IBM DB2 10.1
for Linux, UNIX, and Windows

数据库管理概念和配置参考

更新时间 2013 年 1 月

S151-1758-01

IBM DB2 10.1
for Linux, UNIX, and Windows

数据库管理概念和配置参考

更新时间 2013 年 1 月

S151-1758-01

注意

使用此信息及其支持的产品前，请先阅读第 811 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: http://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: http://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）。

您发送信息给 IBM 后，即授予 IBM 非独占权限，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。

目录

关于本书

数据库管理概念和配置参考提供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

本书由四个部分构成。第一部分到第三部分提供了对 DB2 数据库产品的概念性概述，开始自数据服务器的常规概念，然后一步步深入对组成 DB2 数据库的常用对象进行解释。第四部分包括参考信息。

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。可在 *分区和集群指南*中找到有关数据库分区的详细信息。

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。它还列示了各种命名规则以及 SQL 和 XML 限制。

第 1 部分 数据服务器

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 是关系数据服务器和 XML 数据服务器的混合体。

数据服务器是指安装了 DB2 数据库引擎的计算机。DB2 引擎是功能强大且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描述和规范的完整列表，请访问下列 URL 上的产品页面: http://www.ibm.com/software/data/db2/linux-unix-windows/。

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逻辑分区。添加处理器可以获得更大的处理能力。在带有多个处理器的单一分区数据库配置 (SMP) 中，处理器共享内存和存储系统资源。所有处理器都在一个系统中，所以系统之间的通信及任务协调不会计入工作负载。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，请复审 **DB2TCPCONNMGRS** 注册表变量的值。

添加附加的计算机

如果已有分区数据库环境，那么可以通过对环境添加其他的计算机（单处理器计算机或多处理器计算机）和存储器资源来提高处理能力和数据存储器容量。在计算机之间，不会对内存和存储器资源进行共享。这一选择的优点是，可以在存储器和计算机之间平衡数据和用户访问。

在添加新计算机和存储器之后，请使用 **START DATABASE MANAGER** 命令对新计算机添加新的数据库分区服务器。对于您添加的每个新数据库分区服务器上的实例中的每个数据库，都将创建和配置一个新的数据库分区。在大多数情况下，添加新的数据库分区服务器后不需要重新启动实例。

启用大页支持（AIX）

要在 AIX 操作系统上的 DB2 数据库系统中启用大页支持，必须配置一些操作系统参数并设置 **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。

关于此任务

System z® 上的 POWER4 和更高级别的处理器除了支持传统的 4 KB 页大小以外，还支持 16 MB 页大小。IBM DB2 V10.1 for AIX 之类需要大量内存访问并使用大量虚拟内存的应用程序可通过使用大页获得性能改进。

注:

1. 设置 **DB2_LARGE_PAGE_MEM** 注册表变量还暗示已将内存固定。
2. 在当配置系统以固定内存和支持大型页时，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。如果没有足够内存来支持 4 KB 页，那么对大页分配太多物理内存会降低系统性能。

限制

启用大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页。

过程

要在 AIX 操作系统上的 DB2 数据库系统中启用大页支持，请执行以下操作:

1. 通过发出带以下标志的 **vmo** 命令为 AIX 服务器配置大页支持:

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

   其中 *LargePageSize* 指定硬件支持的大页的大小（以字节计），而 *LargePages* 指定要保留的大页的数目。 例如，如果要为大页支持分配 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，使它能够使用固定的内存。 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页内存。

固定 DB2 数据库共享内存 (AIX)

在 AIX 操作系统上，要固定 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。

关于此任务

将某些部分的内存固定的优点在于，当您访问被固定的页时，无需完成页替换算法就可以检索该页。但是，这样做也存在缺点，会导致操作系统在管理内存方面的灵活程度下降，因此您必须注意确保不过度使用系统。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** 注册表变量设为 YES，那么不能对数据库共享内存启用自调整。

过程

在 AIX 操作系统上，要固定 DB2 数据库共享内存:

1. 配置 AIX 操作系统以启用已固定的内存。 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）如果您要使用中等大小的页面（这是缺省行为），那么应确保 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 数据库系统能够使用固定的内存。
   a. 发出 **db2set** 命令以将 **DB2_PINNED_BP** 注册表变量设为 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 副本概述

对于 V9 及更高版本，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 V9 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 数据库版本的应用程序
- 能够运行独立的 DB2 数据库产品副本来实现不同的功能
- 在将生产数据库移至更高版本的 DB2 数据库产品之前，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 数据库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，应对您的应用程序使用和分发 IBM Data Server Driver for ODBC and CLI 而不是 Data Server Runtime Client，因为一次只能对 COM+ 应用程序使用一个 Data Server Runtime Client。IBM Data Server Driver for ODBC and CLI 没有此限制。

表 1 列示了每个类别中的相关主题。

表 1. 多个 DB2 副本信息的概述

类别	相关主题
一般信息和限制	• 『缺省 IBM 数据库客户机接口副本』
安装	• *安装 DB2 服务器*中的『安装 DB2 服务器（Linux 和 UNIX）』 • *安装 DB2 服务器*中的『安装 DB2 服务器 (Windows)』
配置	• 第 10 页的『运行多个 DB2 副本时设置 DAS』 • 第 11 页的『使用多个 DB2 副本时设置缺省实例（Windows）』 • *Command Reference*中的『dasupdt - 更新 DAS 命令』
管理	• 第 15 页的『更新 DB2 副本（Windows）』 • 第 13 页的『更新 DB2 副本（Linux 和 UNIX）』 • *Command Reference*中的『db2iupdt - 更新实例命令』 • *Command Reference*中的『db2swtch - 切换缺省 DB2 副本命令』 • *Administrative API Reference*中的『db2SelectDB2Copy API - 选择应用程序要使用的 DB2 副本』

缺省 IBM 数据库客户机接口副本

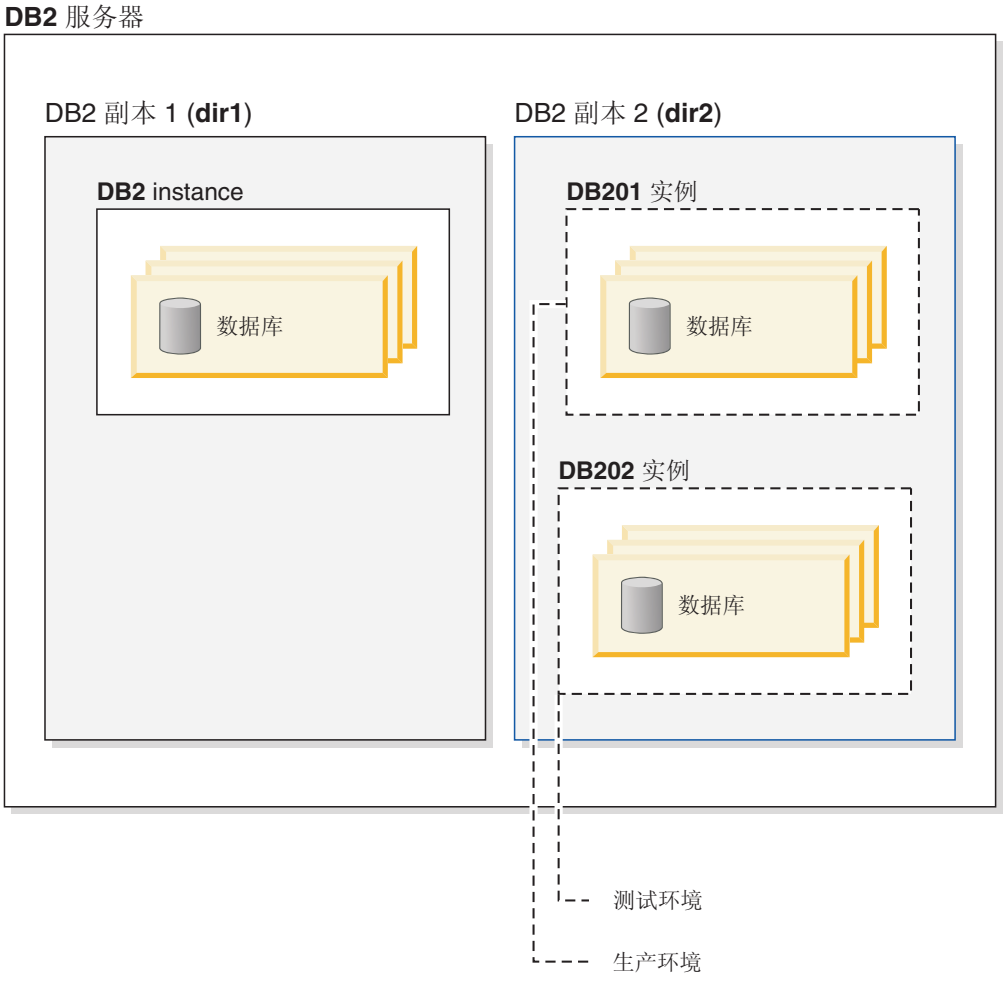
单台计算机上可以有多个 DB2 副本和一个缺省 IBM 数据库客户机接口副本，客户机应用程序通过该接口副本具有缺省情况下与数据库交互所需的 ODBC 驱动程序、CLI 驱动程序和 .NET 数据提供程序代码。

在 V9.1（及更高版本）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 V9.5（及更高版本），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 Data Server Driver Package（DSDRIVER）。对于 V9.5（及更高版本），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 V9.1 后，可以在计算机上安装多个 DB2 副本；在 V9.5 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：



V8 和 V9（或更高版本）副本可以在同一台计算机上共存，但 V8 必须是缺省 DB2 和 IBM 数据库客户机接口副本。除非首先升级到 V9（或更高版本）或者卸载 V8 副本，否则不能在安装期间将缺省的 DB2 副本或缺省 IBM 数据库客户机接口副本从 V8 副本更改为 V9（或更高版本）副本，也不能在以后运行切换缺省副本命令 **db2swtch**。如果在系统上存在 V8 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 V8，因此您不能更改缺省副本。

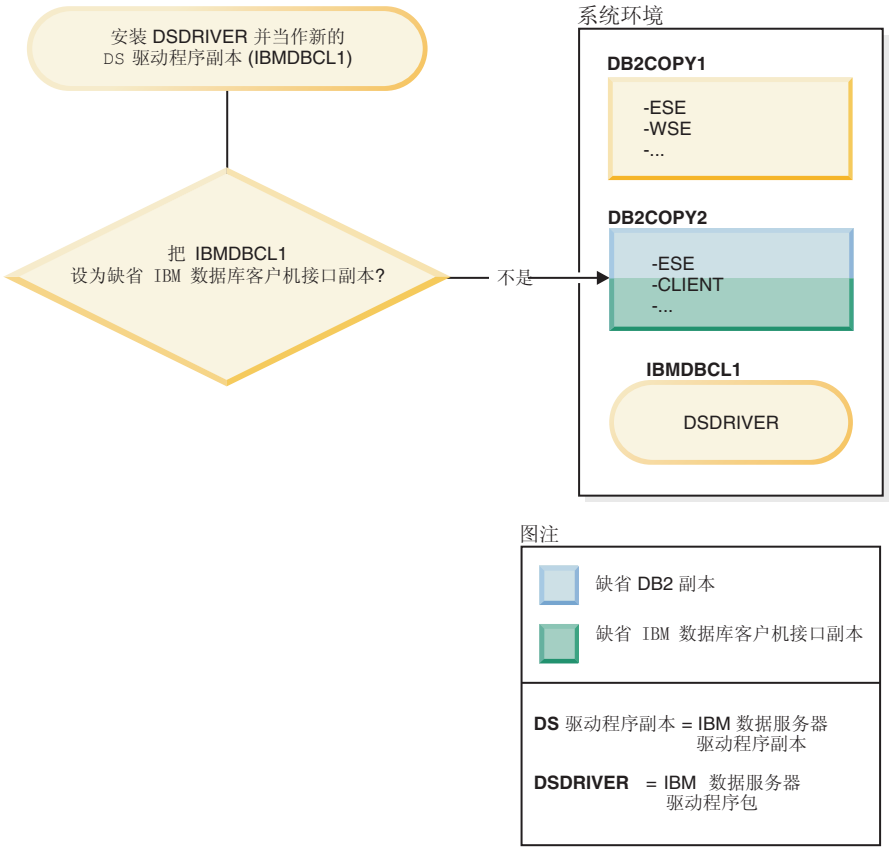
有时，在安装多个 DB2 副本或多个 IBM 数据服务器驱动程序副本之后，您可能要更改缺省的 DB2 副本或缺省的 IBM 数据库客户机接口副本。如果已安装 V8，那么必须先卸载此产品或者将其升级到 V9 或更高版本，然后才能更改缺省 DB2 副本或者更改缺省 IBM 数据库客户机接口副本。

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 V9.5 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。

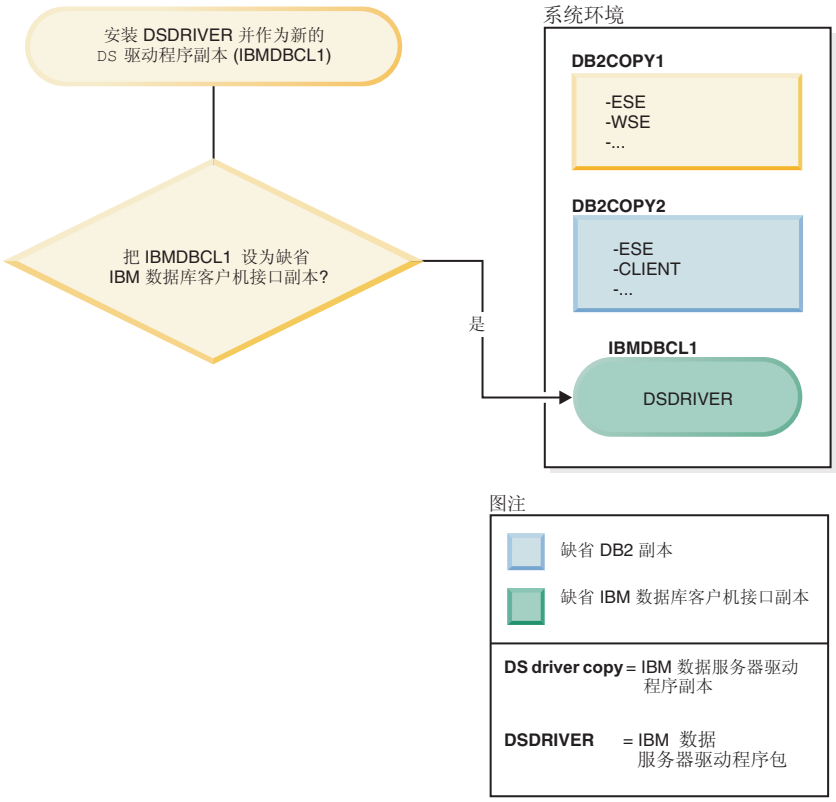


在新的 IBM 数据服务器驱动程序副本上安装 IBM Data Server Driver Package（DSDRIVER）。

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 V9.1 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。

关于此任务

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。因此，下列限制和功能需求适用。

限制

要点： V9.7 中已经不推荐使用“DB2 管理服务器 (DAS)”，在以后的发行版中可能会将其除去。DAS 在 DB2 pureScale® 环境中不受支持。通过使用安全 Shell 协议的软件程序进行远程管理。有关更多信息，请参阅『不推荐使用DB2 管理服务器 (DAS)』，网址为: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html。

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 V9.1 或 V9.5 上运行，那么它可以管理 V8、V9.1 或 V9.5 实例。

- 如果 DAS 在 V8 上运行，那么它只能管理 V8 实例。要管理 V8 或更高版本的实例，您可以升级 V8 DAS 或者将其删除并创建新的 V9.5 DAS。

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 V9.1 或更高版本中，DAS 可以属于当前安装的任何 DB2 副本。

如果 DAS 正在 V9.5 副本中运行，并且您希望它在另一个 V9.5 副本中运行，请使用 **dasupdt** 命令。如果 DAS 正在 V8、V9.1 或 V9.5 副本中运行，并且您希望它在 V9.7 副本中运行，那么无法使用 **dasupdt**，请使用 **dasmigr** 命令将 DAS 升级到 V9.7。

在 Windows 操作系统上，如果需要在版本相同的新缺省 DB2 副本中运行 DAS，那么也可以使用 **dasupdt** 命令。

过程

要在其中一个 DB2 副本中设置 DAS，请执行以下操作:

选择下列其中一项操作:

- 如果尚未创建 DAS，请在其中一个 DB2 副本中创建 DAS。
- 使用 **dasupdt** 命令来仅更新 DAS，以使它在另一个具有相同发行版的 DB2 副本中运行。
- 使用 **dasmigr** 命令从 V8、V9.1 或 V9.5 DAS 升级到 V9.7 DAS。

使用多个 DB2 副本时设置缺省实例（Windows）

DB2INSTANCE 环境是根据您的环境当前设为要使用的 DB2 副本设置的。如果未将它显式设为当前副本中的实例，那么它缺省设为使用 **DB2INSTDEF** 概要文件注册表变量指定的缺省实例。

关于此任务

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**，那么会将 **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 **DB2INSTANCE** *未设置*并且 **DB2INSTDEF**=XYZ，那么将使用 XYZ
  - 如果 **DB2INSTANCE** *未设置*并且 **DB2INSTDEF** *未设置*，那么依赖于有效 **DB2INSTANCE** 的任何应用程序或命令将不工作。
- 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**），但是您必须确保正确设置它们。

  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。

当同一台计算机上有多个 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：

- 仅限于 UNIX 操作系统：要防止两个或更多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。
- 仅限于 Windows 操作系统：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。可以连接 (**ATTACH**) 至其他实例，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，并且每个实例都指回该实例所属的副本，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（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有所变化；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，将其设为 `c:\DB2PROFS`:
  - 使用 **db2set.exe -g** 命令将 **DB2INSTPROF** 设为 `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，用户数据文件的缺省位置（例如，实例目录和 `db2cli.ini` 文件）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（及更高版本）操作系统上: `Program Data\IBM\DB2\`*`Copy Name`*

  其中，*Copy Name* 表示 DB2 副本名。

  注: 根据是否使用 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已设置注册表变量 **DB2CLIINIPATH**，`db2cli.ini` 文件的位置可能会有所变化。

更新 DB2 副本（Linux 和 UNIX）

可更新现有 DB2 副本以及正在该副本上运行的所有实例。还可以选择安装新的 DB2 副本，并在安装之后有选择地更新要在此新副本上运行的实例。

开始之前

- 请确保您具有 root 用户权限。
- 下载修订包并将它解压缩。修订包与您要更新的 DB2 副本必须处于同一发行版。

关于此任务

请遵循下列指示信息将 DB2 副本从一个修订包级别更新为另一个修订包级别（但是处于同一版本级别），或者安装其他功能。

如果您有 DB2 V8、V9.1、V9.5 或 V9.7 副本，那么无法将这些副本从前发行版更新为 DB2 V9.8，您必须对它们执行升级。

限制

- 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，必须重新运行安装。

过程

要更新 DB2 副本，请完成下列步骤:

1. 以具有 root 用户权限的用户登录。
2. 停止所有 DB2 进程。
3. 使用下列其中一个选项来更新每个 DB2 副本:
   - 要更新现有的 DB2 副本并更新所有对此 DB2 副本运行的实例，请发出 **installFixPack** 命令。使用此命令时，无法安装其他功能。
   - 要安装新的 DB2 副本，并且要在安装完成后有选择地更新对现有 DB2 副本运行的实例以使它们对新副本运行，请发出 **db2setup** 命令并在**安装产品**面板中选择**安装新副本**。要安装新的副本，您还可以执行响应文件安装以指定新位置作为安装路径。这些选项中的任何一个都允许您同时安装其他功能。
   - 要对现有的 DB2 副本添加功能，请在**安装产品**面板中选择**使用现有副本**。然后，选择要通过**添加新功能**操作更新的 DB2 副本。仅当 DB2 副本与安装映像处于同一个发行版级别时，此操作才可用。要添加功能，您还可以执行响应文件安装或发出 **db2_install** 命令。

   要点: 建议不要使用命令 **db2_install**，将来的发行版可能会将其除去。请改用带响应文件的 **db2setup** 命令。
4. 如果安装了新的 DB2 副本，请使用 **db2iupdt** 命令来更新任何正在另一个处于相同发行版的 DB2 副本中运行但您希望它们在新副本中运行的实例。下表列示了多个有关更新实例的示例:

实例	DB2 副本	更新为使用另一副本的示例
db2inst1	/opt/IBM/db2/V9.1/	cd /opt/IBM/db2/V9.1_FP3/instance ./db2iupdt db2inst1
db2inst2	/opt/IBM/db2/V9.5FP2/	cd /home/db2/myV9.5_FP1/instance ./db2iupdt -D db2inst2[a]
db2inst3	/opt/IBM/db2/V9.7/	cd /home/db2/myV9.7/instance ./db2iupdt -k db2inst3[b]

注:

a. 使用 **-D** 参数将实例由使用较高发行版级别的副本更新为使用较低发行版级别的副本。

b. 如果要在更新为使用具有较高级别实例类型的 DB2 副本期间保留当前实例类型，请使用 **-k** 参数。如果已从 WSE 更新为 ESE，并且更新实例时未指定此参数，那么实例类型 wse 将转换为 ese。

结果

在安装或更新 DB2 副本之后，始终可以通过发出 **db2iupdt** 命令来更新正在其他处于相同发行版的 DB2 副本中运行的实例，以使它们对这个新的 DB2 副本运行。

更新 DB2 副本（Windows）

可以将现有 DB2 副本以及正在该副本上运行的所有实例都更新为新的修订包级别。还可以选择安装新的 DB2 副本，并在安装之后有选择地更新要在此新副本上运行的实例。

开始之前

- 确保您具有本地管理员访问权。
- 下载修订包并将它解压缩。修订包与您要更新的 DB2 副本必须处于同一发行版。

关于此任务

请遵循下列指示信息将 DB2 副本从一个修订包级别更新为另一个修订包级别（但是处于同一版本级别），或者安装其他功能。

限制

- 只能将发行版相同的实例从较低发行版级别的副本更新为较高发行版级别的副本。无法将实例从较高发行版级别的副本更新为较低发行版级别的副本。
- 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，必须重新运行安装。
- 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。不可能从 V8 的 32 位 x64 DB2 安装版本直接升级到 V9.8 的 64 位安装版本。

过程

要更新 DB2 副本，请完成下列步骤:

1. 作为具有本地管理员权限的用户登录。
2. 停止所有 DB2 实例、服务和应用程序。
3. 运行 `setup.exe` 以启动 DB2 向导来安装 DB2 副本。您可以选择下列选项:
   - 要更新现有的 DB2 副本并更新所有对此 DB2 副本运行的实例，请在**安装产品**面板中选择**使用现有副本**。然后，选择要通过**更新**操作更新的 DB2 副本。使用此操作时，无法安装其他功能。
   - 要安装新的 DB2 副本，并且要在安装完成后有选择地更新对现有 DB2 副本运行的实例以使其对新副本运行，请在**安装产品**面板中选择**安装新副本**。此选项允许您同时安装其他功能。
   - 要对现有的 DB2 副本添加功能，请在**安装产品**面板中选择**使用现有副本**。然后，选择要通过**添加新功能**操作更新的 DB2 副本。仅当 DB2 副本与安装映像处于同一个发行版级别时，此操作才可用。
4. 如果安装了新的 DB2 副本，请使用 **db2iupdt** 命令来更新任何正在另一个处于相同发行版的 DB2 副本中运行但您希望它们在新副本中运行的实例。下表列示了多个有关更新实例的示例:

实例	DB2 副本	更新为使用另一副本的示例
db2inst1	C:\Program Files\IBM\SQLLIB_91\BIN	cd D:\Program Files\IBM\SQLLIB_91_FP5\BIN db2iupdt db2inst1 /u: *user-name*,*password*
db2inst2	C:\Program Files\IBM\SQLLIB_97\BIN	cd D:\Program Files\IBM\SQLLIB_97\BIN db2iupdt db2inst2 /u: *user-name*,*password*

结果

在安装或更新 DB2 副本之后，始终可以通过发出 **db2iupdt** 命令来更新正在其他处于相同发行版的 DB2 副本中运行的实例，以使它们对这个新的 DB2 副本运行。

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。

过程

1. 要使用命令行在同一 DB2 副本中同时运行多个实例:
   a. 输入以下命令，将 **DB2INSTANCE** 变量设为要启动的另一个实例的名称:

      ```
      set db2instance=another_instName
      ```

   b. 通过输入 **db2start** 命令来启动实例。
2. 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**:
     a. 打开命令窗口。
     b. 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，使用以上部分“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。

在同一个或不同的 DB2 副本中使用实例

可以在同一 DB2 副本或不同 DB2 副本中同时运行多个实例。

关于此任务

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 创建数据库时，会在称为 C:\DB2 的目录中创建数据库文件。同样，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。

过程

- 要使用同一 DB2 副本中的多个实例，您必须:
  1. 创建所有实例或将所有实例升级至同一 DB2 副本。
  2. 将 **DB2INSTANCE** 环境变量设为您要使用的实例的名称。 对实例发出命令之前，必须执行此操作。
- 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**开始 > 程序 > IBM DB2 > *DB2 副本名称* > 命令行工具 > 命令窗口**使用“命令”窗口。系统已使用正确环境变量针对所选特定 DB2 副本设置“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 自主计算概述

DB2 自主计算环境能够自我配置、自我修复、自我优化和自我保护。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更改为通过一些技术来管理。

第 20 页的『自动功能』提供了 DB2 自主计算环境所包含的功能的高级别摘要；下表对产品的自主功能进行更详细的分类概述:

表 2. 自主计算信息概述

类别	相关主题
自调整内存功能	• *故障诊断和调整数据库性能*中的『内存使用情况』 • *故障诊断和调整数据库性能*中的『自调整内存』 • *故障诊断和调整数据库性能*中的『自调整内存概述』 • 第 682 页的『auto_maint -"自动维护"』 • *数据库监视指南和参考*中的『db_storage_path -"自动存储器路径"监视元素』 • *数据库监视指南和参考*中的『num_db_storage_paths -"自动存储器路径数"监视元素』 • *数据库监视指南和参考*中的『tablespace_using_auto_storage -"使用自动存储器"监视元素』 • 第 34 页的『配置内存和内存堆』 • 第 37 页的『代理程序、进程技术模型和内存配置概述』 • 第 40 页的『共享文件句柄表』 • 第 40 页的『在受防护方式进程中运行供应商库函数』 • *Administrative Routines and Views*中的『ADMIN_GET_MEM_USAGE 表函数 - 获取实例的总内存消耗』 • 第 36 页的『代理程序和进程技术模型配置』 • 第 39 页的『配置跨多个分区的数据库』
自动存储器	• 第 41 页的『数据库在缺省情况下使用自动存储器』 • 第 144 页的『自动存储器表空间』 • 第 141 页的『自动重新调整 DMS 表空间的大小』
数据压缩	• 第 42 页的『数据压缩』 – 第 268 页的『表压缩』 – 第 391 页的『索引压缩』 – *数据恢复及高可用性指南与参考*中的『备份压缩』 • 第 276 页的『创建表级别压缩字典』 • *数据移动实用程序指南和参考*中的『在装入操作期间创建压缩字典』
自动数据库备份	• *数据恢复及高可用性指南与参考*中的『自动数据库备份』 • *数据恢复及高可用性指南与参考*中的『启用自动备份』 • *数据恢复及高可用性指南与参考*中的『开发备份和恢复策略』

表 2. 自主计算信息概述 (续)

类别	相关主题
自动重组	*故障诊断和调整数据库性能*中的『自动重组』
自动收集统计信息	• *故障诊断和调整数据库性能*中的『自动收集统计信息』 • *故障诊断和调整数据库性能*中的『使用自动收集统计信息』 • *故障诊断和调整数据库性能*中的『自动收集统计信息和概要分析使用的存储器』 • *故障诊断和调整数据库性能*中的『自动收集统计信息活动日志记录』
配置顾问程序	• 第 47 页的『生成数据库配置建议』 – 第 46 页的『使用配置顾问程序调整配置参数』 – 第 47 页的『示例: 使用配置顾问程序请求配置建议』 – *Command Reference*中的『AUTOCONFIGURE 命令』 – *Administrative Routines and Views*中的『使用 ADMIN_CMD 过程的 AUTOCONFIGURE 命令』 – *Administrative API Reference*中的『db2AutoConfig API - 访问配置顾问程序』 • *故障诊断和调整数据库性能*中的『有关性能调整的快速入门技巧』
运行状况监视器	• *数据库监视指南和参考*中的『运行状况监视器』 • *数据库监视指南和参考*中的『运行状况指示器进程循环』 – *数据库监视指南和参考*中的『启用运行状况警报通知』 – *数据库监视指南和参考*中的『使用客户机应用程序配置运行状况指示器』 • *数据库监视指南和参考*中的『运行状况指示器摘要』
实用程序调速	• 第 50 页的『实用程序调速』 • *故障诊断和调整数据库性能*中的『异步索引清除』 • *故障诊断和调整数据库性能*中的『MDC 表的异步索引清除』 – *Command Reference*中的『LIST UTILITIES 命令』 – *Command Reference*中的『SET UTIL_IMPACT_PRIORITY 命令』 – 第 673 页的『util_impact_lim -"实例影响策略"』 – *数据库监视指南和参考*中的『utility_priority -"实用程序优先级"监视元素』

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，但是其他自动功能必须您手动启用:

自调整内存功能（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用来放置表空间数据的缺省存储器组的存储器路径。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。还可创建新存储器组或改变现有存储器组。

数据压缩

可以对表和索引进行压缩以节省存储器。压缩完全自动；一旦使用 CREATE TABLE、ALTER TABLE、CREATE INDEX 或 ALTER INDEX 语句的 COMPRESS YES 子句指定应该对表或索引进行压缩，您就不必执行其他操作来管理压缩。（将未处于压缩状态的现有表或索引转换为处于压缩状态并不要求执行 REORG 来压缩现有数据）。临时表将自动进行压缩；缺省情况下，处于压缩状态的表的索引也将自动进行压缩。

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。确保有最新的完整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。

自动重组

对表数据进行许多更改后，表及其索引可能会碎片化。逻辑上按顺序排列的数据可能会驻留在非顺序页中，从而导致数据库管理器必须执行附加的读操作才能访问数据。自动重组过程会定期评估已经更新了统计信息的表和索引，以便了解是否需要重组并在有必要执行这些操作时进行安排。

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必须更新哪些统计信息。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存取方案。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存取方案。
- **统计信息概要分析。**自动统计信息概要分析功能通过执行以下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。通过设置自动维护数据库配置参数，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。

在 IBM Data Studio V3.1 或更高版本中，可以使用以下工具的任务助手: 配置自动维护. 任务助手可以指导您执行以下过程: 设置选项、查看自动生成的命令以执行任务以及运行这些命令。有关更多详细信息，请参阅使用任务助手管理数据库。

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。

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。会强制断开当前已连接的所有应用程序。
- 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，自动收集统计信息和概要分析、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收集统计信息和概要分析。

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。

自调整内存功能

内存调整功能通过自动设置若干内存配置参数的值来简化内存配置任务。启用此功能之后，内存调整器将在下列内存使用者之间动态分配可用的内存资源: 缓冲池、锁定内存、程序包高速缓存和排序内存。

调整器在 **database_memory** 配置参数所定义的内存限制范围内工作。此参数的值也可以自动调整。启用自调整功能（将 **database_memory** 的值设为 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，那么将为数据库共享内存分配较多的内存。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，那么将释放一些数据库共享内存。

如果 **database_memory** 配置参数未设为 AUTOMATIC，那么数据库将使用您对此参数指定的内存量，从而根据需要在内存使用者之间分配内存。您可以通过两种方法来指定此内存量: 将 **database_memory** 设为某个数值或者将其设为 COMPUTED。在后一种情况下，总内存量基于数据库启动时的数据库内存堆初始值之和。

您还可以对内存使用者启用自调整功能，如下所示:

- 对于缓冲池，使用 ALTER BUFFERPOOL 或 CREATE BUFFERPOOL 语句（指定 AUTOMATIC 关键字）
- 对于锁定内存，使用 **locklist** 或 **maxlocks** 数据库配置参数（指定 AUTOMATIC 值）
- 对于程序包高速缓存，使用 **pckcachesz** 数据库配置参数（指定 AUTOMATIC 值）

- 对于排序内存，使用 **sheapthres_shr** 或 **sortheap** 数据库配置参数（指定 AUTOMATIC 值）

自调整操作所作的更改将记录在 stmmlog 子目录中的内存调整日志文件中。这些日志文件包含每个内存使用者在特定调整时间间隔内的资源需求摘要，这些时间间隔由日志条目中的时间戳记确定。

如果可用内存量不多，那么自调整功能的性能增益有限。由于调整决策基于数据库工作负载，因此内存需要快速变化的工作负载可能会限制自调整内存管理器 (STMM) 的效能。如果工作负载的内存特征不断变化，那么 STMM 将以较低的频率在多变的目标条件下进行调整。在这种情况下，STMM 将无法实现绝对汇合，而是尝试维护针对当前工作负载进行调整的内存配置。

自调整内存功能

内存调整功能通过自动设置若干内存配置参数的值来简化内存配置任务。启用此功能之后，内存调整器将在下列内存使用者之间动态分配可用的内存资源：缓冲池、锁定内存、程序包高速缓存和排序内存。

调整器在 **database_memory** 配置参数所定义的内存限制范围内工作。此参数的值也可以自动调整。启用自调整功能（将 **database_memory** 的值设为 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，那么将为数据库共享内存分配较多的内存。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，那么将释放一些数据库共享内存。

如果 **database_memory** 配置参数未设为 AUTOMATIC，那么数据库将使用您对此参数指定的内存量，从而根据需要在内存使用者之间分配内存。您可以通过两种方法来指定此内存量：将 **database_memory** 设为某个数值或者将其设为 COMPUTED。在后一种情况下，总内存量基于数据库启动时的数据库内存堆初始值之和。

您还可以对内存使用者启用自调整功能，如下所示：

- 对于缓冲池，使用 ALTER BUFFERPOOL 或 CREATE BUFFERPOOL 语句（指定 AUTOMATIC 关键字）
- 对于锁定内存，使用 **locklist** 或 **maxlocks** 数据库配置参数（指定 AUTOMATIC 值）
- 对于程序包高速缓存，使用 **pckcachesz** 数据库配置参数（指定 AUTOMATIC 值）
- 对于排序内存，使用 **sheapthres_shr** 或 **sortheap** 数据库配置参数（指定 AUTOMATIC 值）

自调整操作所作的更改将记录在 stmmlog 子目录中的内存调整日志文件中。这些日志文件包含每个内存使用者在特定调整时间间隔内的资源需求摘要，这些时间间隔由日志条目中的时间戳记确定。

如果可用内存量不多，那么自调整功能的性能增益有限。由于调整决策基于数据库工作负载，因此内存需要快速变化的工作负载可能会限制自调整内存管理器 (STMM) 的效能。如果工作负载的内存特征不断变化，那么 STMM 将以较低的频率在多变的目标条件下进行调整。在这种情况下，STMM 将无法实现绝对汇合，而是尝试维护针对当前工作负载进行调整的内存配置。

自调整内存功能概述

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。启用此功能之后，内存调整器将在下列内存使用者之间动态分配可用的内存资源：缓冲池、锁定内存、程序包高速缓存和排序内存。

要启用自调整内存功能，请使用 **self_tuning_mem** 数据库配置参数。

可以自动调整下列与内存相关的数据库配置参数：

- **database_memory** - 数据库共享内存大小
- **locklist** - 锁定列表的最大存储量
- **maxlocks** - 升级之前锁定列表的最大百分比
- **pckcachesz** - 程序包高速缓存大小
- **sheapthres_shr** - 共享排序的排序堆阈值
- **sortheap** - 排序堆大小

内存分配

内存分配和释放在各个时间进行。可以在发生特定事件（例如应用程序建立连接）时将内存分配给特定内存区，也可以为了响应配置更改而重新分配内存。

图 1 显示了数据库管理器为不同用途分配的各个内存区以及允许您控制这些内存区的大小的配置参数。请注意，在分区数据库环境中，每个数据库分区都将设置自己的数据库管理器共享内存。

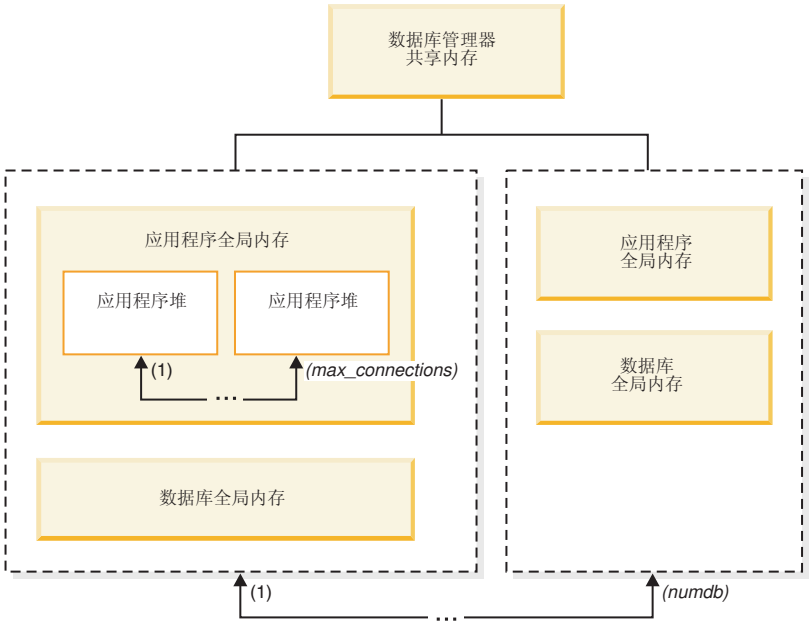


图 1. 数据库管理器分配的内存的类型

每当发生下列其中一个事件时，数据库管理器都将分配内存：

数据库管理器启动时（db2start）

数据库管理器共享内存（也称为*实例共享内存*）在数据库管理器停止

（**db2stop**）前将保持处于已分配状态。此区域包含数据库管理器在管理通过所有数据库连接进行的活动时所需的信息。DB2 将自动控制数据库管理器共享内存的大小。

第一次激活数据库或者连接到数据库时

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** 数据库配置参数指定。缺省情况下，此参数设为 automatic，从而允许 DB2 计算为数据库分配的初始内存量以及在运行时根据数据库的需要自动配置数据库内存大小。

可以对下列内存区进行动态调整:

- 缓冲池（使用 ALTER BUFFERPOOL 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

此外，还可以动态地更新 **sortheap**、**sheapthres_shr** 和 **sheapthres** 配置参数。唯一的限制是，不能动态地将 **sheapthres** 由 0 更改为大于 0 的值，反之亦然。

缺省情况下，将执行共享排序操作，排序内存使用者在任何一个时间可以使用的数据库共享内存量由 **sheapthres_shr** 数据库配置参数值确定。仅当分区内并行性、数据库分区和连接集中器都处于禁用状态，并且 **sheapthres** 数据库管理器配置参数设为非零值时，才能执行专用排序操作。

应用程序连接至数据库时

每个应用程序都有自己的*应用程序堆*，这是*应用程序全局内存*的组成部分。您可以使用 **applheapsz** 数据库配置参数来限制任何一个应用程序可以分配的内存量，也可以使用 **appl_memory** 数据库配置参数来限制应用程序内存总耗用量。

创建代理程序时

当分区数据库环境中出现连接请求或新的 SQL 请求时，系统将指定代理程序并为其分配*代理程序专用内存*。代理程序专用内存包含仅供此特定代理程序使用的内存。如果已启用专用排序操作，那么将从代理程序专用内存中分配专用排序堆。

下列配置参数用于限制为每种类型的内存区分配的内存量。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 此数据库管理器配置参数指定各个应用程序可以使用的并行活动数据库的最大数目。因为每个数据库都有自己的全局内存区，所以增大此参数的值将增加可以分配的内存量。

maxappls

此数据库配置参数指定可以同时连接到特定数据库的应用程序的最大数目。此参数的值将影响可以为该数据库分配的代理程序专用内存量和应用程序全局内存量。

max_connections

此数据库管理器配置参数用于限制任何时候可以访问数据服务器的数据库连接或实例连接的数目。

max_coordagents

此数据库管理器配置参数用于限制，一个实例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的数目（在分区数据库环境中，将对每个数据库分区实施此限制）。与 **maxappls** 和 **max_connections** 相配合，此参数将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量。

可使用 **db2mtrk** 命令调用的内存跟踪程序来查看实例中的当前内存分配量。您还可以使用 ADMIN_GET_MEM_USAGE 表函数来确定整个实例或单一数据库分区的内存总耗用量。使用 MON_GET_MEMORY_SET 和 MON_GET_MEMORY_POOL 表函数在实例、数据库或应用程序级别检查当前内存使用情况。

在 UNIX 和 Linux 操作系统上，尽管 **ipcs** 命令可用于列出所有共享内存段，但是该命令不会精确地反映已消耗的资源量。可使用 **db2mtrk** 命令替代 **ipcs**。

内存参数交互和局限性

虽然您可以启用自调整内存功能并对大多数与内存相关的配置参数使用缺省的 AUTOMATIC 设置，但了解不同内存参数的局限性以及它们之间的交互非常有用，这样您就可以更好地控制它们的设置并了解特定情况下仍可能发生“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** 配置参数设为性能内存的最大容量，也可以将 **database_memory** 设为 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为应用程序请求提供服务而分配的功能内存（即应用程序内存）的最大容量。缺省情况下，此参数设为 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限制的 DB2 数据库产品，或者您将 **instance_memory** 设为特定值，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，执行针对另一个堆的另一个语句时也可能会发生“内存不足”错误。现在，可以使用缺省的 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（要避免工作方式欠佳的数据库应用程序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 设为 AUTOMATIC，并且您要使用具有内存使用限制的 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_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将供应用程序内存（**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（使用连接集中器或者 DB2 V9.5 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。

关于此任务

启用此功能之后，内存调整器将在多个内存使用者（其中包括缓冲池、锁定内存、程序包高速缓存和排序内存）之间动态分配可用的内存资源。

过程

1. 要对数据库启用自调整内存功能，请使用 **UPDATE DATABASE CONFIGURATION** 命令或 db2CfgSet API 将 **self_tuning_mem** 数据库配置参数设为 ON。
2. 要对内存配置参数所控制的内存区启用自调整功能，请使用 **UPDATE DATABASE CONFIGURATION** 命令或 db2CfgSet API 将相关配置参数设为 AUTOMATIC。
3. 要对缓冲池启用自调整功能，请使用 CREATE BUFFERPOOL 语句或 ALTER BUFFERPOOL 语句将缓冲池大小设为 AUTOMATIC。在分区数据库环境中，缓冲池在 SYSCAT.BUFFERPOOLDBPARTITIONS 中不应该有任何条目。

结果

注:

1. 由于在不同的内存使用者之间分配自调整内存，因此在任意给定时间，必须至少对两个内存区（例如锁定内存和数据库共享内存）同时启用自调整功能。当满足下列其中一个条件时，内存调整器将主动调整系统上的内存（**self_tuning_mem** 数据库配置参数的值为 ON）：
   - 一个配置参数或缓冲池大小设为 AUTOMATIC，并且 **database_memory** 数据库配置参数设为数字值或者 AUTOMATIC
   - **locklist**、**sheapthres_shr**、**pckcachesz** 或缓冲池大小中的任意两个设为 AUTOMATIC
   - **sortheap** 数据库配置参数设为 AUTOMATIC
2. **locklist** 数据库配置参数的值将与 **maxlocks** 数据库配置参数一起进行调整。对 **locklist** 参数禁用自调整功能将自动地对 **maxlocks** 参数禁用自调整功能，而对 **locklist** 参数启用自调整功能将自动地对 **maxlocks** 参数启用自调整功能。
3. 仅当数据库管理器配置参数 **sheapthres** 设为 0 时，才允许自动调整 **sortheap** 或 **sheapthres_shr** 数据库配置参数。
4. **sortheap** 的值将与 **sheapthres_shr** 一起进行调整。对 **sortheap** 参数禁用自调整功能将自动地对 **sheapthres_shr** 参数禁用自调整功能，而对 **sheapthres_shr** 参数启用自调整功能将自动地对 **sortheap** 参数启用自调整功能。
5. 自调整内存功能只能在高可用性灾难恢复 (HADR) 主服务器上运行。在 HARD 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，只有在正确设置配置的情况下，此功能才会在主服务器上运行。如果切换 HADR 数据库角色，那么自调整内存操作也将进行切换，从而在新的主服务器上运行。在主数据库启动之后，或者在备用数据库通过接管操作转换为主数据库之后，自调整内存管理器（STMM）引擎可分派单元 (EDU) 可能直到第一个客户机建立连接后才会启动。

禁用自调整内存功能

可以对整个数据库或者一个或多个配置参数或缓冲池禁用自调整内存功能。

关于此任务

即使对整个数据库禁用自调整内存功能，设为 AUTOMATIC 的内存配置参数和缓冲池也仍然支持自动调整；但是，内存区将保持当前大小不变。

过程

1. 要对数据库禁用自调整内存功能，请使用 **UPDATE DATABASE CONFIGURATION** 命令或 db2CfgSet API 将 **self_tuning_mem** 数据库配置参数设为 OFF。
2. 要对内存配置参数所控制的内存区禁用自调整功能，请使用 **UPDATE DATABASE CONFIGURATION** 命令或 db2CfgSet API 将相关配置参数设为 MANUAL 或指定数字参数值。
3. 要对缓冲池禁用自调整功能，请使用 ALTER BUFFERPOOL 语句将缓冲池大小设为特定的值。

结果

注:

- 在某些情况下，要对一个内存配置参数启用自调整功能，还必须同时对另一个相关的内存配置参数启用此功能。例如，这意味着对 **locklist** 或 **sortheap** 数据库配置参数禁用自调整内存功能时，还将分别对 **maxlocks** 或 **sheapthres_shr** 数据库配置参数禁用自调整内存功能。

确定已启用自调整功能的内存使用者

您可以查看由配置参数控制或者应用于缓冲池的自调整内存功能设置。

关于此任务

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，为排序内存调整缓冲池内存大小所产生的性能增益可能不会立即体现。

过程

- 要查看配置参数的设置，请使用下列其中一种方法。
  - 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
-------------------------------------------------------------------------------------------
自调整内存功能                      (SELF_TUNING_MEM) =   ON（活动）             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)
```

  - 使用 db2CfgGet API。

    将返回下列值:

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

    SQLF_ON_ACTIVE 表明自调整功能已启用并处于活动状态，而 SQLF_ON_INACTIVE 表明自调整功能已启用但当前处于不活动状态。

- 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，请使用下列查询:

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

    对缓冲池启用自调整功能之后，该特定缓冲池的 SYSCAT.BUFFERPOOLS 视图中的 NPAGES 字段将设为 -2。当自调整功能处于禁用状态时，NPAGES 字段将设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，请使用 **GET SNAPSHOT** 命令并检查缓冲池的当前大小（**bp_cur_buffsz** 监视元素的值）:

```
GET SNAPSHOT FOR BUFFERPOOLS ON database-alias
```

    指定特定数据库分区上的缓冲池大小的 ALTER BUFFERPOOL 语句为 SYSCAT.BUFFERPOOLDBPARTITIONS 目录视图中的该缓冲池创建异常条目（或更新现有条目）。如果某个缓冲池的例外条目已存在，并且缺省缓冲池大小设为 AUTOMATIC，那么该缓冲池将不会参与自调整操作。

分区数据库环境中的自调整内存功能

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。

对分区数据库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。在该分区中作出调整决策之后，会将内存调整分发到其他数据库分区，以确保所有数据库分区都维护类似的配置。

单调整分区模型假定，仅当所有数据库分区具有类似内存需求时，才会使用该功能。在确定是否对分区数据库启用自调整内存功能时，请使用下列准则。

建议对分区数据库使用自调整内存功能的情况

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运行，并且多个逻辑数据库分区均匀地分布在多个物理数据库分区中
- 数据分布情况最佳或者接近最佳
- 工作负载均匀地分布在各个数据库分区中，这意味着，各个数据库分区中一个或多个堆的内存需求均相同

在这种环境中，如果所有数据库分区的配置相同，那么自调整内存功能将正确地配置系统。

建议对分区数据库使用自调整内存功能并进行限定的情况

在环境中的大部分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。

应该对所有包含数据的数据库分区启用自调整内存功能，并且应该将其中的一个数据库分区指定为调整分区。由于协调程序分区和目录分区的配置可能不同，因此应对那些分区禁用自调整内存功能。要对协调程序分区和目录分区禁用自调整内存功能，请对这些分区将 **self_tuning_mem** 数据库配置参数设为 OFF。

建议不要对分区数据库使用自调整内存功能的情况

如果各个数据库分区的内存需求有所不同，或者不同的数据库分区正在极不相同的硬件上运行，那么最好禁用自调整内存功能。要禁用此功能，请对所有分区将 **self_tuning_mem** 数据库配置参数设为 OFF。

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近的最佳方法是查看快照监视器。如果下列快照元素在所有数据库分区中都相近（差别不超过 20%），那么可以认为这些数据库分区的内存需求极为相近。

通过发出以下命令来收集下列数据: `get snapshot for database on <dbname>`

```
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
锁定升级次数                          = 0
互斥锁定升级次数                            = 0

已分配的共享排序堆总数                      = 0
共享排序堆高水位标记                        = 0
超出阈值后的排序次数（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

OLAP 功能数目                                = 0
OLAP 功能溢出数目                            = 0
活动 OLAP 功能数                             = 0
```

通过发出以下命令来收集下列数据：`get snapshot for bufferpools on <dbname>`

```
缓冲池数据逻辑读取次数                = 0
缓冲池数据物理读取次数                = 0
缓冲池索引逻辑读取次数                = 0
缓冲池索引物理读取次数                = 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 DB2 pureScale 环境中执行自调整内存功能

在 DB2 pureScale 环境中启用自调整内存时，调整成员会监视内存配置并将所有配置更改传播至所有其他成员。

在 DB2 pureScale 环境中启用自调整内存功能时，有一个成员（称为调整成员）监视内存配置并将所有配置更改传播至所有其他成员，以维护实例中所有成员间的一致配置。

在 DB2 pureScale 环境中，调整成员指定为 -1 时，每次激活数据库时都会随机选择调整成员。而且，如果运行调整器的成员取消激活，那么此调整器将自动在当前能够运行它的另一成员上启动。为了让成员运行此调整器，数据库必须在该成员上处于活动状态，并且该成员必须将 **self_tuning_mem** 设为 ON。

- 要确定当前已指定为调整成员的成员，请调用 ADMIN_CMD 过程，如下所示：

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

- 要更改调整成员，请调用 ADMIN_CMD 过程，如下所示：

  ```
  CALL SYSPROC.ADMIN_CMD('update stmm tuning member membernum')
  ```

  调整成员可在随机选择调整成员与根据需要使用显式指定的调整成员之间切换。

请注意，调整成员更改时，从运行调整器的成员中收集的一些数据会被废弃。必须针对新调整成员重新收集此数据。在重新收集此数据的短暂时间段内，内存调整器将仍然调整系统；但是，调整的运行方式可能与在原始成员上的运行方式稍有不同。

在 DB2 pureScale 环境中启动内存调整器

在 DB2 pureScale 环境中，每当数据库在 **self_tuning_mem** 设为 ON 的一个或多个成员上处于活动状态时，将运行内存调整器。

对特定成员禁用自调整内存功能

- 要对一部分数据库成员禁用自调整内存功能，请将这些成员的 **self_tuning_mem** 数据库配置参数设为 OFF。
- 要对由特定成员上的配置参数控制的一部分内存使用者禁用自调整内存功能，请在该成员上将相关配置参数的值设为固定值。建议使自调整内存功能的配置参数值在所有正运行的成员中保持一致。
- 要对特定成员上的特定缓冲池禁用自调整内存功能，请发出 ALTER BUFFERPOOL 语句，以指定大小值和要在其上禁用自调整内存功能的成员。

指定特定成员上的缓冲池大小的 ALTER BUFFERPOOL 语句将在 SYSCAT.BUFFERPOOLEXCEPTIONS 目录视图中为该缓冲池创建例外条目（或更新现有条目）。如果某个缓冲池的例外条目已存在，并且缺省缓冲池大小设为 AUTOMATIC，那么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使用自调整功能，请执行以下操作:

1. 通过发出 ALTER BUFFERPOOL 语句并将缓冲池大小设为特定值，对此缓冲池禁用自调整功能。
2. 发出另一 ALTER BUFFERPOOL 语句以将此成员上的缓冲池大小设为缺省值。
3. 通过发出另一个 ALTER BUFFERPOOL 语句并将缓冲池大小设为 AUTOMATIC，对此缓冲池启用自调整功能。

在不均匀的环境中启用自调整内存功能

在每个成员上运行的工作负载应该具有类似内存需求。成员之间的内存需求可能会有偏差，例如，仅在一个成员上执行消耗大量资源的排序时，或者一些成员与不同硬件相关联并且具有的可用内存比其他成员更多时。仍可在某些成员上激活自调整内存功能。要在各种不同的内存环境中使用自调整内存功能，请标识具有类似内存需求的一组成员并在这些成员上激活自调整内存功能。然后可在余下成员上禁用自调整内存功能，并可手动配置其内存。

在分区数据库环境中使用自调整内存功能

在分区数据库环境中启用自调整内存功能之后，将出现一个单独的数据库分区（称为*调整分区*），此分区将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。

调整分区是根据多个特征选择的，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，请调用 **ADMIN_CMD** 过程，如下所示:

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

- 要更改调整分区，请调用 **ADMIN_CMD** 过程，如下所示:

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

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选择调整分区，请输入 -1 作为 *partitionnum* 的值。

在分区数据库环境中启动内存调整器

由于自调整内存功能要求所有分区都处于活动状态，因此在分区数据库环境中，仅当数据库由显式的 **ACTIVATE DATABASE** 命令激活时，才会启动内存调整器。

对特定数据库分区禁用自调整内存功能

- 要对部分数据库分区禁用自调整内存功能，请对那些数据库分区将 **self_tuning_mem** 数据库配置参数设为 OFF。
- 要对特定数据库分区中由配置参数控制的部分内存使用者禁用自调整内存功能，请对该数据库分区将相关配置参数值或缓冲池大小设为 MANUAL 或某个特定值。建议使自调整内存功能的配置参数值在所有运行中的分区中保持一致。
- 要对特定数据库分区中的特定缓冲池禁用自调整内存功能，请发出 ALTER BUFFERPOOL 语句并指定大小值以及要在其中禁用自调整内存功能的分区。

  对特定数据库分区指定缓冲池大小的 ALTER BUFFERPOOL 语句将在 SYSCAT.BUFFERPOOLDBPARTITIONS 目录视图中为该缓冲池创建例外条目或更新现有条目。如果某个缓冲池的例外条目已存在，并且缺省缓冲池大小设为 AUTOMATIC，那么该缓冲池将不会参与自调整操作。要除去例外条目，以便可以对缓冲池启用自调整功能:

  1. 通过发出 ALTER BUFFERPOOL 语句并将缓冲池大小设为特定值，对此缓冲池禁用自调整功能。
  2. 发出另一个 ALTER BUFFERPOOL 语句，以便将此数据库分区中缓冲池的大小设为缺省大小。
  3. 通过发出另一个 ALTER BUFFERPOOL 语句并将缓冲池大小设为 AUTOMATIC，对此缓冲池启用自调整功能。

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，并且每个分区中运行的工作负载的内存需求应该比较接近。如果数据分布不均匀，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中的内存需求不均匀的情况。例如，如果只在一个分区中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其他分区不同并且有更多的可用内存，那么将发生这种情况。在此类环境中，仍然可以对某些数据库分区启用自调整内存功能。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，应该以手动方式进行内存配置。

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。
- 可使用 **db2mtrk** 命令来监视堆使用情况，并使用 ADMIN_GET_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 3. 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	从 V9.5 开始，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。对于分区数据库环境、集中器或 SMP 配置而言，这意味着除非使用 AUTOMATIC 设置，否则可能需要增大在先前发行版中使用的 **applheapsz** 值。	The AUTOMATIC 设置允许应用程序堆大小根据需要增加。如果存在 **appl_memory** 限制或 **instance_memory** 限制，就可能会强制施加限制。
database_memory	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。从 V9.5 开始，AUTOMATIC 是所有 DB2 服务器产品的缺省设置。
dbheap	确定数据库堆使用的最大内存。	AUTOMATIC 设置允许数据库堆大小根据需要增加。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	如果您使用的是具有内存使用量限制的 DB2 数据库产品，或者您将此参数设为特定值，那么此参数指定可以为数据库分区分配的最大内存量。	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加。如果存在 **instance_memory** 限制，就可能会强制施加限制。

表 3. 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
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加。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加。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。

下表显示了在升级或创建实例期间以及在升级或创建数据库期间，各个配置参数是否设为缺省的 AUTOMATIC 值。

表 4. 在实例和数据库升级以及创建期间将设为 AUTOMATIC 的配置参数

配置参数	在升级或创建实例之后设为 AUTOMATIC	在升级数据库之后设为 AUTOMATIC	在创建数据库之后设为 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，建议不要使用下列元素:

- 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。

代理程序和进程技术模型配置

从 V9.5 开始，DB2 数据库提供了一种较简单但更灵活的机制来配置与过程模型相关的参数。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、进程技术模型和内存配置概述

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。

下表按类别列示了代理程序、进程和内存配置主题:

表 5. 代理程序、进程和内存配置信息概述

类别	相关主题
一般信息、限制和不兼容性	• 第 34 页的『配置内存和内存堆』 • 第 36 页的『代理程序和进程技术模型配置』 • *故障诊断和调整数据库性能*中的『DB2 过程模型』 • 第 39 页的『配置跨多个分区的数据库』
安装和升级	• *DB2 Connect User's Guide*中的『连接集中器』 • *DB2 Connect User's Guide*中的『DB2 Connect™ 调整』 • *DB2 Connect User's Guide*中的『使用 OS/390® 和 zSeries® SYSPLEX 时的注意事项』 • *安装 DB2 服务器*中的『磁盘和内存需求』 • *安装 DB2 服务器*中的『修改内核参数 (Linux)』 • *升级到 DB2 V10.1*中的『DB2 服务器行为更改』
性能	• *故障诊断和调整数据库性能*中的『连接集中器的客户机连接改进』 • *故障诊断和调整数据库性能*中的『数据库代理程序』 • *故障诊断和调整数据库性能*中的『数据库代理程序管理』 • *故障诊断和调整数据库性能*中的『数据库管理器共享内存』 • *故障诊断和调整数据库性能*中的『DB2 中的内存分配』 • *故障诊断和调整数据库性能*中的『调整内存分配参数』

表 5. 代理程序、进程和内存配置信息概述 (续)

类别	相关主题
命令、API、注册表变量、函数和例程	• *Command Reference*中的『db2pd - 监视和诊断 DB2 数据库命令』 • *Command Reference*中的『GET DATABASE MANAGER CONFIGURATION 命令』 • *Command Reference*中的『RESET DATABASE MANAGER CONFIGURATION 命令』 • *Command Reference*中的『UPDATE DATABASE MANAGER CONFIGURATION 命令』 • *Command Reference*中的『db2mtrk - 内存跟踪程序命令』 • *Administrative API Reference*中的『sqlfupd 数据结构』 • • 第 40 页的『共享文件句柄表』 • 第 40 页的『在受防护方式进程中运行供应商库函数』 • *Administrative Routines and Views*中的『ADMIN_GET_MEM_USAGE 表函数 - 获取实例的总内存消耗』 • *SQL Reference Volume 1*中的『SQL 和 XML 限制』 • *SQL Reference Volume 1*中的『SYSCAT.PACKAGES 目录视图』 • *Administrative Routines and Views*中的『DBMCFG 管理视图 - 检索数据库管理器配置参数信息』 • *Administrative Routines and Views*『ADMIN_CMD 过程 - 运行管理命令』
配置参数	• 第 581 页的『配置参数摘要』 • 第 680 页的『appl_memory -“应用程序内存”配置参数』 • 第 680 页的『applheapsz -“应用程序堆大小”』 • 第 696 页的『database_memory -“数据库共享内存大小”』 • 第 698 页的『dbheap -“数据库堆”』 • 第 638 页的『instance_memory -“实例内存”』 • 第 721 页的『locklist -“锁定列表的最大存储量”』 • 第 643 页的『max_connections -“最大客户机连接数”』 • 第 645 页的『max_coordagents -“最大协调代理程序数”』 • 第 736 页的『maxappls -“最大活动应用程序数”』 • 第 649 页的『mon_heap_sz -“数据库系统监视器堆大小”』 • 第 652 页的『num_poolagents -“代理程序池大小”』 • 第 772 页的『stat_heap_sz -“统计信息堆大小”』 • 第 773 页的『stmtheap -“语句堆大小”』

表 5. 代理程序、进程和内存配置信息概述 (续)

类别	相关主题
监视元素	• *数据库监视指南和参考*中的『代理程序和连接』 • *数据库监视指南和参考*中的『agents_from_pool - 从池中分配代理程序数』 • *数据库监视指南和参考*中的『agents_registered - 已注册代理程序数』 • *数据库监视指南和参考*中的『agents_registered_top - 最大已注册代理程序数』 • *数据库监视指南和参考*中的『agents_stolen - 失窃代理程序数』 • *数据库监视指南和参考*中的『appls_in_db2 - 当前在数据库中执行的应用程序数』 • *数据库监视指南和参考*中的『associated_agents_top - 最大关联代理程序数』 • *数据库监视指南和参考*中的『coord_agents_top - 最大协调代理程序数』 • *数据库监视指南和参考*中的『local_cons - 本地连接数』 • *数据库监视指南和参考*中的『local_cons_in_exec - 正在数据库管理器中执行的本地连接数』 • *数据库监视指南和参考*中的『num_gw_conn_switches - 最大代理程序溢出数』 • *数据库监视指南和参考*中的『rem_cons_in - 数据库管理器的远程连接数』 • *数据库监视指南和参考*中的『rem_cons_in_exec - 正在数据库管理器中执行的远程连接数』

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重置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，用于更新或重置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** 注册表变量设为 TRUE，或者将 *versionNumber* 设为低于版本 9.5 的版本号的任意版本号。

将数据库升级到版本 9.7 时，现有的数据库配置参数在数据库升级后通常会保留它们的值。但是，将添加使用其缺省值的新参数，并且会将一些现有参数设为新的版本 9.7 缺省值。有关对现有数据库配置参数所作的更改的详细信息，请参阅*升级到 DB2 V10.1*中的『DB2 服务器行为更改』主题。缺省情况下，对升级后的数据库发出的任何后续更新或重置数据库配置请求都将应用于所有数据库分区。

对于现有的更新或重置命令脚本，前面提到的规则同样适用于所有数据库分区。您可以修改脚本，以便包括 **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。

对于调用了 **db2CfgSet** API 的现有应用程序而言，必须使用版本 9.5 或更高版本的指示信息。如果要采用版本 9.5 以前的行为，那么可以设置 **DB2_UPDDBCFG_SINGLE_DBPARTITION** 注册表变量，也可以修改应用程序以调用具有版本 9.5 或更高版本号的 API，其中包括新的 db2CfgSingleDbpartition 标志以及用于更新或重置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重置每个数据库分区。

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 V9.5 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 V9.5 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，从 V9.5 开始，**maxfilop** 配置参数具有另一个缺省值以及新的最小值和最大值。在数据库升级期间，如果正在从 V9.5 以前的发行版升级，那么 **maxfilop** 配置参数将自动设为这个缺省值。

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。

关于此任务

在 V9.5 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 V9.5 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。可创建存储器组，它们由数据库管理器将数据放在其上的路径组成。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。可在创建数据库时指定缺省存储器组的路径。

数据库在缺省情况下使用自动存储器

自动存储器使存储器管理变得更轻松。存储器在存储器组级别进行管理，并且创建、扩展和添加容器的责任由数据库管理器接管，而不是使用显式容器定义在表空间级别管理存储器。

注: 尽管可在创建数据库时指定 AUTOMATIC STORAGE NO 子句，但建议不要使用 AUTOMATIC STORAGE 子句，将来发行版可能会除去该子句。
缺省情况下，所有数据库都是使用自动存储器创建的。但是，如果数据库是通过指定 AUTOMATIC STORAGE NO 子句创建的，那么它不会使用自动存储器管理的表空间。

如果创建数据库，那么缺省情况下，会自动创建缺省存储器组。可为其建立一个或多个初始存储器路径。随着数据库增大，数据库管理器将在这些存储器路径中创建容器并根据需要扩展容器或自动创建新容器。可使用 ADMIN_GET_STORAGE_PATHS 管理视图来显示存储器路径列表。

如果数据库没有存储器组，那么可使用 CREATE STOGROUP 语句创建存储器组。新创建的存储器组是缺省存储器组，并且所有新自动存储器管理的表空间会添加至使用此存储器组的数据库。可使用 CREATE STOGROUP 语句或 ALTER STOGROUP 语句的 SET AS DEFAULT 子句来更改缺省存储器组。

要点:

- 添加存储器路径不会将现有非自动存储器表空间转换为使用自动存储器。您可以将数据库管理的表空间 (DMS) 转换为使用自动存储器。无法将系统管理的表空间 (SMS) 转换为使用自动存储器。有关更多信息，请参阅第 148 页的『转换表空间以使用自动存储器』。
- 数据库创建存储器组后，它始终至少有一个存储器组。不能从数据库管理器中除去最后一个存储器组。
- 为帮助确保可预测性能，但添加至存储器组的存储器路径应具有类似介质特征。

数据压缩

通过使用内置到 DB2 for Linux, UNIX, and Windows中的压缩功能来减小表、索引甚至备份映像的大小，就可以减少存储数据所需要的存储器。

表和索引通常包含重复信息。这种重复信息可能是单个列值或组合列值，也可能是列值的公共前缀，或者是 XML 数据中的重复模式。您可以通过多种压缩功能来减少存储表和索引所需要的空间量，还可以利用一些功能部件来确定通过压缩可以节省的存储空间。

还可以对备份进行压缩以减小备份的大小。[1]

DB2 V9.7 的大多数版本中具备的压缩功能包括:

- 值压缩
- 备份压缩。

如果具备 DB2 Storage Optimization Feature 的许可证，就可以获得下列附加压缩功能:

- 行压缩，其中包括压缩 XML 存储器对象。
- 临时表压缩
- 索引压缩。

自动收集统计信息

DB2 优化器使用目录统计信息来确定查询的最佳存取方案。过期或者不完整的表或索引统计信息可能会导致优化器选择并非最佳的方案，这会导致查询执行速度下降。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，使这些统计信息保持最新也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（这是 DB2 的自动表维护功能的组成部分），可以让数据库管理器确定是否需要更新统计信息。可以使用实时统计信息 (RTS) 功能在编译语句时*以同步方式*执行自动收集统计信息操作，也可以启用 **RUNSTATS** 命令以便在后台运行此程序以执行*同步*收集。虽然可以在实时收集统计信息功能处于禁用状态时启用后台收集统计信息功能，但必须启用后台收集统计信息功能才能执行实时收集统计信息功能。缺省情况下，当您创建数据库时，就会启用自动后台统计信息收集 **auto_runstats** 和自动实时统计信息收集 **auto_stmt_stats**。

从 DB2 V9 开始，可以使用配置顾问程序来确定新数据库的初始配置，包括 **auto_stmt_stats** 数据库配置参数的适当设置。

在 IBM Data Studio V3.1 或更高版本中，可以使用以下工具的任务助手: 配置自动收集统计信息. 任务助手可以指导您执行以下过程: 设置选项、查看自动生成的命令以执行任务以及运行这些命令。有关更多详细信息，请参阅使用任务助手管理数据库。

了解异步收集和实时收集统计信息

启用实时统计信息收集之后，还可以使用一些元数据来生成统计信息。*生成*表示派生或创建统计信息，而不是作为正常 **RUNSTATS** 命令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，那么可以知道该表中包含的行数。在

1. 有关更多信息，请参阅*数据恢复及高可用性指南与参考*中的『备份压缩』。

某些情况下，并未派生统计信息，而是由索引和数据管理器维护统计信息，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层数。

查询优化器根据查询需求和表更新活动数量（更新、插入或删除操作的数目）来确定统计信息的收集方式。

实时收集统计信息功能能够提供更及时更准确的统计信息。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。不管是否启用实时统计信息收集，异步统计信息收集操作将每隔两小时执行一次。此时间间隔可能太长从而不足以为某些应用程序提供准确的统计信息。

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，因此同步统计信息收集操作进行了采样
- 已生成同步统计信息
- 同步统计信息收集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。一个请求必须由实时收集统计信息功能发起，而另一个请求必须由异步统计信息收集检查操作发起。

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** 实用程序执行异步统计信息收集操作。调速功能根据当前数据库活动来控制 **RUNSTATS** 实用程序耗用的资源量: 随着数据库活动的增加，此实用程序的运行速度将变慢，从而减少资源需求。
- 对于每个查询，同步统计信息收集操作的时间限制为 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。
- 同步统计信息收集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。此存储顺序可避免更新系统目录时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，只能执行一个同步统计信息收集操作。其他需要执行同步统计信息收集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。
- 要定制所收集的统计信息的类型，您可以启用统计信息概要分析功能（此功能使用有关先前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），也可以为特定的表创建自己的统计信息概要文件。
- 只对缺少统计信息或具有高级别活动（根据更新、插入或删除操作的数目衡量）的表收集统计信息。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，查询优化器可以在不使用统计信息的情况下选择存取方案。
- 对于异步统计信息收集检查操作而言，还将对大型表（超过 4000 页的表）进行采样，以确定高级表活动是否更改了统计信息。只有在保证已更改统计信息的情况下，才会收集这种大型表的统计信息。

- 对于异步统计信息收集操作而言，自动将 **RUNSTATS** 实用程序安排在维护策略中指定的联机维护时间段运行。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会按照维护策略指定的联机维护时间段进行，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特定策略，该策略指定了自动收集统计信息作用域内的一组表。
- 执行自动收集统计信息操作时，受影响的表仍然可用于常规的数据库活动（更新、插入或删除操作）。
- 不收集昵称的实时统计信息（同步统计信息或生成的统计信息）。要在系统目录中刷新异步统计信息收集的昵称统计信息，请调用 SYSPROC.NNSTAT 过程。对于异步统计信息收集，DB2 for Linux, UNIX, and Windows 会自动调用 SYSPROC.NNSAT 过程以刷新系统目录中的昵称统计信息。
- 不会对统计视图收集实时统计信息（同步或已生成）。
- 已声明临时表 (DGTT) 只能收集实时统计信息。

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。在某些联机事务处理 (OLTP) 方案中可能存在负面影响，对查询运行时间设置了上限的情况下尤其如此。

对常规表、具体化查询表 (MQT) 和全局临时表执行实时同步统计信息收集操作。不收集全局临时表的异步统计信息。

不会对下列对象执行自动收集统计信息（同步或异步）操作:

- 已被标记为 VOLATILE 的表（在 SYSCAT.TABLES 目录视图中设置了 VOLATILE 字段的表）
- 创建临时表 (CGTT)
- 您已通过直接对 SYSSTAT 目录视图发出 UPDATE 语句手动更新其统计信息的表

  以手动方式修改表统计信息时，数据库管理器假定您现在负责维护其统计信息。为了让数据库管理器维护已经以手动方式更新其统计信息的表的统计信息，请使用 **RUNSTATS** 命令来收集统计信息，或者在使用 **LOAD** 命令时指定收集统计信息。在 V9.5 之前创建并且在升级前已手动更新其统计信息的表不受影响，其统计信息将由数据库管理器自动维护，直到您以手动方式更新这些统计信息为止。

不会对下列对象生成统计信息:

- 统计视图
- 您已通过直接对 SYSSTAT 目录视图发出 UPDATE 语句手动更新其统计信息的表。如果未启用实时统计信息收集，那么仍会对已手动更新其统计信息的表生成一些统计信息。

在分区数据库环境中，将在单个数据库分区中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中收集统计信息（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。

使用 TRUNCATE 语句或 **IMPORT** 命令截断的表会因为包含过时统计信息而自动重组。

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。

在取消激活数据库的情况下，“以异步方式自动收集统计信息”操作可能会中断。如果未使用 **ACTIVATE DATABASE** 命令或 API 显式激活此数据库，那么在最后一个用户与此数据库断开连接时，就会取消激活此数据库。如果操作中断，那么可能会将错误消息记录在 DB2 诊断日志文件中。为了避免中断“以异步方式自动收集统计信息”操作，请显式激活此数据库。

实时统计信息和说明处理

不会对仅仅使用说明工具进行说明（而未执行）的查询进行实时处理。下表概述了 CURRENT EXPLAIN MODE 专用寄存器具有不同值时的行为。

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 V9.5 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，统计信息高速缓存仅驻留在目录数据库分区中，即使每个数据库分区都有目录高速缓存也是如此。启用实时统计信息收集后，目录高速缓存需求更高。如果启用了实时统计信息收集，那么应考虑调整 **catalogcache_sz** 数据库配置参数的值。

从 DB2 V9 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** 数据库配置参数设为 ON。

自动收集统计信息和统计概要文件

同步和异步统计信息是根据表的有效统计概要文件收集的，但下列情况例外:

- 为了使同步统计信息收集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。
- 同步统计信息收集操作可以选择生成统计信息，但是可能无法生成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。

如果同步统计信息收集操作无法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。

启用自动收集统计信息

为了进行高效率的数据访问和实现最佳工作负载性能，具有准确而完整的数据库统计信息极为关键。您可以使用自动化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。

关于此任务

对于在单一处理器上运行单一数据库分区的环境，可以通过以下方法增强此功能：收集查询数据并生成统计信息概要文件，这可以帮助 DB2 服务器自动收集工作负载所需的一组准确统计信息。在分区数据库环境、某些联合数据库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，必须首先通过将 **auto_maint** 和 **auto_tbl_maint** 数据库配置参数设为 ON 来配置数据库。

过程

将 **auto_maint** 和 **auto_tbl_maint** 数据库配置参数设为 ON 后，您有以下选择：

- 要启用在后台收集统计信息，请将 **auto_runstats** 数据库配置参数设为 ON。
- 要对统计视图启用后台统计信息收集，请将 **auto_stats_views** 和 **auto_runstats** 数据库配置参数设为 ON。
- 为允许后台统计信息收集自动对大型表和统计视图使用抽样，还应将 **auto_sampling** 数据库配置参数设为 ON。除 **auto_runstats**（仅适用于表）或 **auto_runstats** 和 **auto_stats_views**（适用于表和统计视图）以外，还应使用此设置。
- 要启用实时收集统计信息，请将 **auto_stmt_stats** 和 **auto_runstats** 数据库配置参数都设为 ON。

配置顾问程序

您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。

可显示建议值或通过在 **CREATE DATABASE** 和 **AUTOCONFIGURE** 命令中指定 APPLY 参数来应用建议值。这些建议是根据您提供的输入以及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。

关于此任务

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。

过程

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
```

示例

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，如第 49 页的图 2 中所示:

```
        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
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

        数据库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列表的百分比            (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
自调整内存功能                          (SELF_TUNING_MEM) =          ON                  ON
    自动运行统计                        (AUTO_RUNSTATS) =           ON                  ON
共享排序的排序堆阈值（4KB）             (SHEAPTHRES_SHR) =          5000                AUTOMATIC

        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 已成功完成 AUTOCONFIGURE。可能已更改数据库管理器或数据库配置值。在任何更改生效之前，必须重新启动实例。在新的配置参数生效之后，还可能要重新绑定程序包，以便使用新值。
```

图 2. 配置顾问程序样本输出

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）是在导致索引条目失效的操作后执行的延迟型索引清除操作。根据索引类型的不同，条目可以是记录标识（RID）或块标识（BID）。在后台以异步方式运行的索引清除程序将除去无效的索引条目。

AIC 将加快从分区表中拆离数据分区这一进程的速度，并且在分区表包含一个或多个非分区索引时启动。在这种情况下，AIC 将除去所有引用了已拆离的数据分区的非分区索引条目以及任何伪删除的条目。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。在 DB2 V9.7 FP1 及更高版本的发行版中，通过异步分区拆离任务来启动 AIC。

在 DB2 V9.7 FP1 之前，如果该分区表有从属具体化查询表 (MQT)，那么要在执行 SET INTEGRITY 语句之后才会启动 AIC。

在 AIC 执行过程中，将维持正常的表访问。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，将对与分区表相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 任务分发到适当的表分区并指定数据库代理程序。分发守护程序和清除程序代理程序是内部系统应用程序，它们分别以应用程序名称 **db2taskd** 和 **db2aic** 出现在 **LIST APPLICATIONS** 命令输出中。为了防止意外中断，不能强制执行系统应用程序。只要数据库处于活动状态，分发守护程序就一直处于联机状态。清除程序在完成清除之前将保持活动状态。如果数据库在清除期间被停用，那么在您重新激活数据库之后，AIC 将继续进行。

AIC 对性能的影响

AIC 对性能的影响很小。

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。您可以使用 **SET UTIL_IMPACT_PRIORITY** 命令或 db2UtilityControl API 来更改此优先级。

监视 AIC

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输出中。以下是 **LIST UTILITIES SHOW DETAIL** 命令输出的示例:

```
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
进度监视:
      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
进度监视:
      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，另一个清除程序正在处理索引 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。正常状态是“正在执行”，但如果清除程序正在等待被指定到可用的数据库代理程序，或者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。

注意，由于每个数据库分区仅对该数据库分区中运行的任务指定标识，因此不同数据库分区中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) 功能来增强转出删除性能。转出删除是一种从多维集群 (MDC) 表中删除合格数据块的高效方法。AIC 是指在执行使索引条目失效的操作之后延迟清除索引。

在标准转出删除期间，将以同步方式清除索引。如果表包含许多记录标识 (RID) 索引，那么需要花费相当多时间来除去引用了正被删除的表行的索引键。您可以通过指定在删除操作落实后清除这些索引来提高转出速度。

要对 MDC 表利用 AIC，必须显式地启用*延迟索引清除转出*机制。可以通过两种方法来指定延迟转出：将注册表变量 **DB2_MDC_ROLLOUT** 设为 DEFER 或者发出 SET CURRENT MDC ROLLOUT MODE 语句。在执行延迟索引清除转出操作期间，会将各个块标记为已转出，但不会更新 RID 索引，直到该事务落实之后才会进行更新。在删除操作期间将清除块标识 (BID) 索引，这是因为它们并不需要执行行级别处理。

转出删除操作落实时将调用 AIC 转出；如果数据库已关闭，那么在数据库重新启动之后首次访问该表时也会调用 AIC 转出。在 AIC 进行期间，对索引执行的查询（包括那些需要访问正被清除的索引的查询）将成功。

每个 MDC 表都有一个协调清除程序。多个转出的索引清除操作都合并在清除程序中，后者将为每个 RID 索引生成一个清除代理程序。清除代理程序以并行方式更新 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）。您可以使用 **SET UTIL_IMPACT_PRIORITY** 命令或 db2UtilityControl API 来更改此优先级。

注: 在 DB2 V9.7 及更高版本的发行版中，不支持对具有分区 RID 索引的数据分区 MDC 表执行延迟清除转出。仅支持 NONE 和 IMMEDIATE 方式。如果 **DB2_MDC_ROLLOUT** 注册表变量设为 DEFER，或者 CURRENT MDC ROLLOUT MODE 专用寄存器设为 DEFERRED 以覆盖 **DB2_MDC_ROLLOUT** 设置，那么清除转出类型将为 IMMEDIATE。

如果 MDC 表仅存在非分区 RID 索引，那么支持执行延迟索引清除转出。MDC 块索引可以是分区索引，也可以是非分区索引。

监视延迟索引清除转出操作的进度

由于直到完成清除之后才能复用 MDC 表中已转出的块，因此监视延迟索引清除转出操作的进度将很有用。请使用 **LIST UTILITIES** 命令来显示被清除的每个索引的实用程序监视器条目。您还可以通过使用 ADMIN_GET_TAB_INFO 表函数或 **GET SNAPSHOT** 命令，来检索数据库中在转出删除后处于暂挂异步清除状态的 MDC 表块的总数 (BLOCKS_PENDING_CLEANUP)。

在 **LIST UTILITIES SHOW DETAIL** 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。每个阶段都代表一个 RID 索引。

```
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
进度监视:
   估算完成百分比                    = 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。不能创建其他实例。

可使用多个实例来执行以下操作:

- 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。

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，可以在同一台计算机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 3 显示了系统、实例和数据库之间的分层关系。

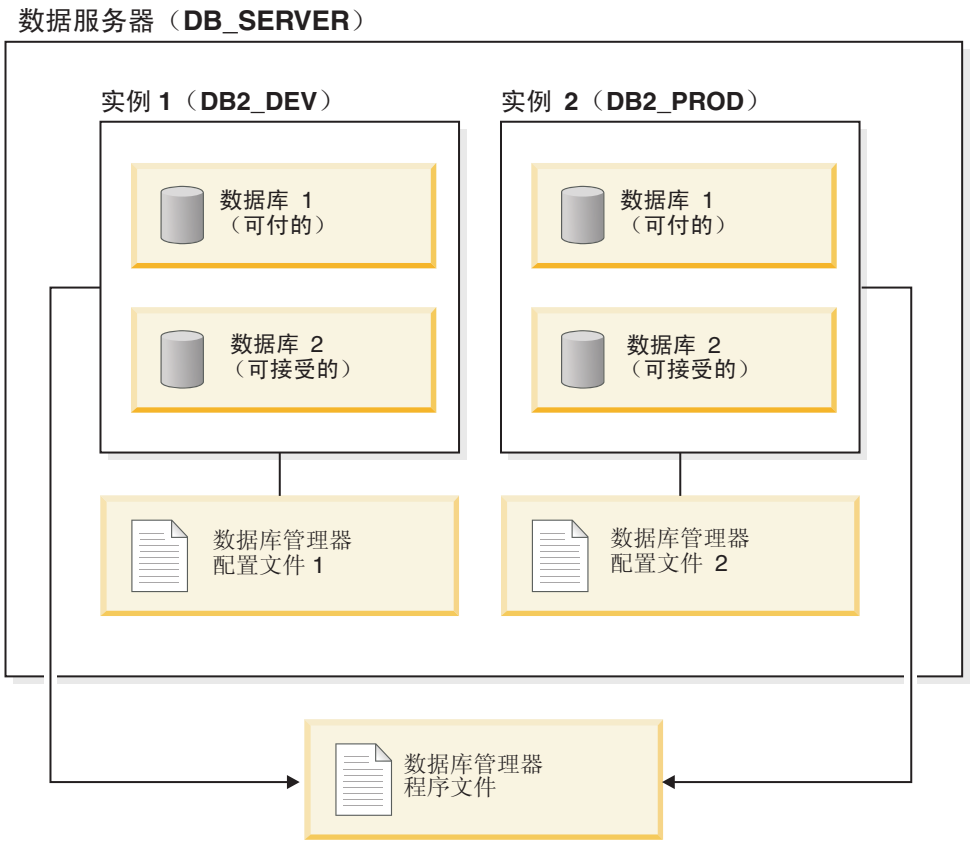


图 3. DB2 系统、实例和数据库之间的分层关系

您还必须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，IBM Data Studio），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。

要点: V9.7 中已经不推荐使用“DB2 管理服务器 (DAS)”，在以后的发行版中可能会将其除去。DAS 在 DB2 pureScale 环境中不受支持。通过使用安全 Shell 协议的软件程序进行远程管理。有关更多信息，请参阅『不推荐使用DB2 管理服务器 (DAS)』，网址为: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html。

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，您可以创建数据库管理器的初始实例。在 V9.5 或更高发行版中，缺省名称是 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。

要支持立即使用此实例，请在安装期间设置下列注册表变量:

- 环境变量 **DB2INSTANCE** 设为 DB2_01。
- 注册表变量 **DB2INSTDEF** 设为 DB2_01。

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 程序的 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。在使用 **db2iupgrade** 命令进行的实例升级期间，将复制用户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**: 在实例所有者的主目录的 sqllib 子目录下的 bin、adm 和 misc 子目录。
- 将 **DB2INSTANCE** 环境变量设为实例名。

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。创建实例目录后，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: 要防止两个或更多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（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有所变化；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，将其设为 c:\DB2PROFS:
  - 使用 **db2set.exe -g** 命令将 **DB2INSTPROF** 设为 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，用户数据文件的缺省位置（例如，实例目录和 db2cli.ini 文件）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（及更高版本）操作系统上: Program Data\IBM\DB2*Copy Name*

  其中，*Copy Name* 表示 DB2 副本名。

  注: 根据是否使用 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已设置注册表变量 **DB2CLIINIPATH**，db2cli.ini 文件的位置可能会有所变化。

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。

开始之前

如果您在 Windows 上属于“Administrative”组，或者您在 Linux 或 UNIX 操作系统上具有 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。实例标识不应是 root，也不应具有到期密码。

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

过程

要使用命令行添加实例:

输入命令: 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），还必须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 可使用的端口号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）

要更新 Linux 或 UNIX 操作系统上根实例的配置，请使用 **db2iupdt** 命令。要更新非根实例，请运行 **db2nrupdt** 命令。

关于此任务

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的 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。如果创建了新的数据库管理器配置文件，那么旧文件会备份到实例所有者主目录下的 sqllib 子目录的 backup 子目录中。

db2iupdt 命令位于 *DB2DIR*/instance 目录，*DB2DIR* 是当前版本的 DB2 数据库产品的安装位置。

限制

此任务仅适用于根实例。

过程

要从命令行更新实例，请输入:

```
db2iupdt InstName
```

InstName 是实例所有者的登录名。

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。

关于此任务

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的 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。如果创建了新的数据库管理器配置文件，那么旧文件会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。

过程

要更新实例配置，请发出 **db2iupdt** 命令。 例如:

```
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。

管理实例

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。

关于此任务

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

可允许实例在每次系统重新启动后自动启动。完成此任务所需的步骤因操作系统不同而变化。

关于此任务

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为自动启动。

在 Linux、UNIX 和 Windows 操作系统上，使用 **db2icrt** 创建的实例设为手动启动。

过程

要将实例配置为自动启动，请执行以下操作:

- 在 Windows 操作系统上，必须转至“服务”面板并在该处更改 DB2 服务的属性。
- 在 Linux 和 UNIX 操作系统上，输入以下命令:

  ```
  db2iauto -on instance_name
  ```

  其中 *instance_name* 是实例的登录名。

启动实例（Linux 和 UNIX）

您可能需要在正常企业运营期间启动或停止 DB2 数据库。例如，必须先启动实例，才能执行下列某些任务：连接至实例上的数据库、预编译应用程序、将包绑定至数据库或访问主机数据库。

开始之前

在 Linux 或 UNIX 操作系统上启动实例之前，请执行以下操作：

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）
source INSTHOME/sqllib/db2cshrc（对于 C shell）
```

过程

要启动实例，请执行以下操作：

- 从命令行输入 **db2start** 命令。DB2 数据库管理器会将该命令应用于当前实例。
- 在 IBM Data Studio 中，打开任务助手以启动该实例。

启动实例（Windows）

您可能需要在正常企业运营期间启动或停止 DB2 实例。例如，必须先启动实例，才能执行下列某些任务：连接至实例上的数据库、预编译应用程序、将包绑定至数据库或访问主机数据库。

开始之前

为成功将 DB2 数据库实例作为服务启动，用户帐户必须有 Windows 操作系统定义的正确特权才能启动 Windows 服务。用户帐户可以是 Administrators、Server Operators 或 Power Users 组的成员。启用扩展安全性之后，缺省情况下，只有 DB2ADMNS 和 Administrators 组的成员才能启动数据库。

关于此任务

缺省情况下，**db2start** 命令将 DB2 数据库实例作为 Windows 服务来启动。通过在 **db2start** 命令上指定 **/D** 参数，Windows 上的 DB2 数据库实例仍可作为进程运行。通过使用“控制面板”或 **NET START** 命令，DB2 数据库实例还可作为服务运行。

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** 参数以将 DB2 实例作为进程启动。

过程

要启动实例，请执行以下操作：

- 从命令行输入 **db2start** 命令。DB2 数据库管理器会将该命令应用于当前实例。
- 在 IBM Data Studio 中，打开任务助手以启动该实例。

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。

开始之前

必须存在多个实例。

过程

- 要连接实例，请执行以下操作:
  - 从命令行输入 **ATTACH** 命令。
  - 从客户机应用程序调用 sqleatin API。
- 要与实例拆离，请执行以下操作:
  - 从命令行输入 **DETACH**。
  - 从客户机应用程序调用 sqledtin API。

示例

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
```

对 testdb2 实例执行维护活动后，要从实例拆离，请输入:

```
db2 detach
```

在同一个或不同的 DB2 副本中使用实例

可以在同一 DB2 副本或不同 DB2 副本中同时运行多个实例。

关于此任务

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 创建数据库时，会在称为 C:\DB2 的目录中创建数据库文件。同样，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。

过程

- 要使用同一 DB2 副本中的多个实例，您必须:
  1. 创建所有实例或将所有实例升级至同一 DB2 副本。
  2. 将 **DB2INSTANCE** 环境变量设为您要使用的实例的名称。 对实例发出命令之前，必须执行此操作。
- 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**开始 > 程序 > IBM DB2 > *DB2 副本名称* > 命令行工具 > 命令窗口**使用“命令”窗口。系统已使用正确环境变量针对所选特定 DB2 副本设置“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

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。为确保没有重要或关键应用程序在运行，请使用 LIST APPLICATIONS 命令。
3. 通过使用 FORCE APPLICATION 命令强制所有应用程序和用户断开与数据库的连接。
4. 如果命令行处理器会话连接至实例，那么必须在运行 **db2stop** 命令前运行 **TERMINATE** 命令来结束每个会话。

关于此任务

运行命令以启动或停止实例时，DB2 数据库管理器会将该命令应用于当前实例。有关更多信息，请参阅第 497 页的『标识当前实例』。

限制

db2stop 命令只能在服务器上运行。

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。

过程

要在 Linux 或 UNIX 操作系统上停止实例，请执行以下操作:

- 从命令行输入 **db2stop** 命令。DB2 数据库管理器会将该命令应用于当前实例。
- 从 IBM Data Studio 中，打开任务助手以停止该实例。

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

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。为确保没有重要或关键应用程序在运行，请使用 LIST APPLICATIONS 命令。
3. 通过使用 FORCE APPLICATION 命令强制所有应用程序和用户断开与数据库的连接。
4. 如果命令行处理器会话连接至实例，那么必须在运行 **db2stop** 命令前运行 **TERMINATE** 命令来结束每个会话。

关于此任务

注: 运行命令以启动或停止实例时，数据库管理器会将该命令应用于当前实例。有关更多信息，请参阅第 497 页的『标识当前实例』。

限制

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。

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。要停止实例，必须停止所有服务。

过程

要停止实例，请执行以下操作:

- 从命令行输入 **db2start** 命令。DB2 数据库管理器会将该命令应用于当前实例。
- 从命令行输入 **NET STOP** 命令。
- 从 IBM Data Studio 中，打开任务助手以停止该实例。

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，请完成下列步骤:

1. 停止当前使用该实例的所有应用程序。
2. 在每个命令窗口中，通过运行 **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，然后作为具有 root 用户权限的用户登录。
6. 发出 **db2idrop** 命令。 例如:

   ```
   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，如果发出 **db2idrop** 命令并接收到一条消息，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）如果所删除的实例是缺省实例，请通过发出 **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）如果实例所有者的用户标识和组仅用于该实例，请除去该用户标识和组。 如果您打算重新创建该实例，那么请不要将它们除去。

   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。

DB2 pureScale 环境中的实例管理

本节包含有关管理 DB2 pureScale实例的信息。此处讨论的主题特定于 DB2 pureScale Feature，不涉及更广范围的 DB2 数据库产品的管理。

本节提供有关以下方面的基本管理概念、任务和用户场景的信息:

- 启动和停止成员和集群高速缓存设施。
- 升级集群高速缓存设施或成员主机以及向共享文件系统集群添加资源之类的维护任务。
- 自动重新启动失败的成员和集群高速缓存设施
- 配置共享文件系统集群、集群高速缓存设施和缓冲池。

DB2 pureScale 环境中的多个活动数据库

从 DB2 V9.8 FP3 开始，DB2 pureScale 环境中现在可包含多个活动数据库。

在 DB2 V9.8 的先前迭代中，DB2 pureScale 环境与 DB2 Enterprise Server Edition 和分区数据库环境的不同之处在于: 任一时间只能有一个数据库处于活动状态。对于 DB2 V9.8 FP3 发行版，该限制已被除去。

现在用户在 DB2 pureScale 环境中的体验几乎与 DB2 Enterprise Server Edition 和分区数据库环境中有多个活动数据库的体验完全相同。

在 DB2 pureScale 环境外部，任何给定时间可处于活动状态的数据库的最大数目为 255。但是，在 DB2 pureScale 环境中，任何给定时间可处于活动状态的数据库的最大数目为 200。

DB2 pureScale 环境中的活动数据库的缺省最大数目为 32。这也是 DB2 Enterprise Server Edition 和分区数据库环境的新缺省值。

要使 DB2 pureScale 环境包含多个活动数据库，请参阅第 792 页的『DB2 pureScale CF 内存参数配置』中概述的配置参数更改。

要更改 DB2 pureScale 环境中的活动数据库数目，请修改 **numdb** 配置参数，更改会在下次全局重新启动后生效。

除了 DB2 pureScale 环境中的 **numdb** 限制外，还对实例中所有成员中的数据库激活最大数有所限制。此数据库激活数最大为 512。例如，如果由 4 个成员组成的 DB2 pureScale 环境中的每个成员激活了 200 个数据库，那么数据库成员激活总数为 800。因为此数据库激活数 (800) 超过最大上限，所以系统会返回错误。

在多个数据库环境中，如果需要成员崩溃恢复 (MCR)，那么在每个成员上并行恢复的数据库数由 **numdb** 配置参数或 **DB2_MCR_RECOVERY_PARALLELISM_CAP** 注册表变量的值（两者中的较小者）设置。

在 DB2 pureScale 环境中启动和停止集群组件及数据库

在 DB2 pureScale 环境中，集群高速缓存设施或成员将作为全局 **db2start** 或 **db2stop** 命令的一部分启动或停止。

发出 **db2start** 或 **db2stop** 命令时，所有当前定义的成员和集群高速缓存设施将启动或停止。但是，在某些情况下，在更详细的级别启动和停止这些集群组件及数据库很有用。

启动集群高速缓存设施

集群高速缓存设施 (CF) 在执行全局 **db2start** 或个别 **db2start** CF 命令时启动。此任务重点在于启动单个集群高速缓存设施。

关于此任务

在全局 **db2start** 上，集群管理器尝试在首选主集群高速缓存设施上启动主角色（又称为 PRIMARY 状态），另一集群高速缓存设施将作为辅助角色（又称为 PEER 状态）启动。

如果一个集群高速缓存设施已启动并以 PRIMARY 状态运行，那么您启动的下一个集群高速缓存设施（辅助集群高速缓存设施）将进入 CATCHUP 阶段，这将确保在辅助集群高速缓存设施转换为 PEER 状态之前，其内存中具有主集群高速缓存设施中的所有相关信息的副本。

过程

要启动特定集群高速缓存设施，请执行以下操作:

发出以下命令:

```
db2start CF CF-identifier
```

示例

John 是一个 DBA，他中的描述将另一集群高速缓存设施添加至 DB2 pureScale实例。他使用以下代码查询了当时实例中所有集群高速缓存设施的状态:

```
SELECT ID,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       varchar(STATE,17) AS STATE,
       ALERT
FROM SYSIBMADM.DB2_CF
```

并获得以下输出:

```
ID     CUR_HOST     STATE      ALERT
------ ------------ ---------- -----
   128 so5          PRIMARY    NO
   129 so6          STOPPED    NO

     2 record(s) selected.
```

集群高速缓存设施 128 是唯一的活动集群高速缓存设施，所以它也处于 PRIMARY 状态。

John 发出 db2start CF 129，使用以下代码查询集群高速缓存设施的状态:

```
SELECT ID,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       varchar(STATE,17) AS STATE,
       ALERT
FROM SYSIBMADM.DB2_CF
```

并获得以下输出:

```
ID     CUR_HOST     STATE        ALERT
------ ------------ ----------   -----
   128 so5          PRIMARY      NO
   129 so6          CATCHUP(50%) NO

     2 record(s) selected.
```

集群高速缓存设施 129 现在正从集群高速缓存设施 128 获取所有相关信息的副本，所以它可在集群高速缓存设施 128 失败时作为主集群高速缓存设施接管。

他使用以下代码查询所有集群高速缓存设施的状态:

```
SELECT ID,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       varchar(STATE,17) AS STATE,
       ALERT
FROM SYSIBMADM.DB2_CF
```

并获得以下输出:

```
ID     CUR_HOST     STATE      ALERT
------ ------------ ---------- -----
   128 so5          PRIMARY    NO
   129 so6          PEER       NO

     2 record(s) selected.
```

现在集群高速缓存设施 129 处于 PEER 状态，它可在当前主集群高速缓存设施失败时作为主集群高速缓存设施接管。

停止集群高速缓存设施

集群高速缓存设施 (CF) 作为全局 **db2stop** 或个别 **db2stop CF** 命令的一部分停止。本主题重点讨论如何停止单个集群高速缓存设施。

关于此任务

如果要在主机上停止集群高速缓存设施但让同一主机上的其他实例进程保持启动并运行，请发出 **db2stop CF** 命令。可在关闭主机前使用 **db2stop CF**。

限制

如果出现下列任何情况，那么不能停止主集群高速缓存设施:

- 实例中有一些活动 成员。
- 主集群高速缓存设施包含脏页。
- 主集群高速缓存设施包含锁定。
- 辅助集群高速缓存设施未处于 PEER 状态。

但是，即使发生其中任何一种情况，都可使用 **FORCE** 选项停止辅助 集群高速缓存设施。

过程

要停止特定集群高速缓存设施，请发出以下命令:

```
db2stop CF CF-identifier
```

启动成员

DB2成员是作为全局 **db2start** 或单个 **db2start** 命令的一部分启动的。本主题重点讨论启动单个成员。

关于此任务

发出 **db2start**（对于实例或成员）时，数据库管理器会启动主机上的所有 DB2 空闲进程和实例中定义的所有集群高速缓存设施（带有活动主机），只要这些空闲进程和集群高速缓存设施未在运行。如果无法启动集群高速缓存设施，那么成员启动操作将失败。如果无法在成员的原始主机上启动这些空闲进程，那么启动操作将失败，但 DB2 集群服务将以轻量级重新启动方式在另一主机上启动该成员。（有关更多信息，请参阅轻量级重新启动。）

过程

要启动特定成员，请发出以下命令:

```
db2start member member-id
```

结果

数据库管理器启动当前在 db2nodes.cfg 文件中指定的主机上的个别成员，即使此主机并非目标成员的原始主机。换言之，先前在另一主机上作为访客成员运行的成员将以轻量级重新启动方式在该主机上启动。如果该成员的原始主机处于活动状态，并且准备接收其常驻成员，那么 DB2 集群服务会将该成员故障恢复至其原始主机。

示例

John 是 DBA，他已在定义了一个成员 (成员 0) 的主机上完成了维护操作。现在他已在主机上重新启动该实例 (db2start instance on host so1)。此成员在维护操作前已关闭。然后他使用以下命令查询实例中所有成员的状态:

```
SELECT ID,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       varchar(STATE,21) AS STATE,
       ALERT
FROM SYSIBMADM.DB2_MEMBER
```

并获得以下输出:

```
ID     HOME_HOST CUR_HOST     STATE      ALERT
------ --------- ------------ ---------- -----
     0 so1       so1          STOPPED    NO
     2 so2       so2          STARTED    NO
     4 so3       so3          STARTED    NO

     3 record(s) selected.
```

John 发出以下命令来启动成员 0: `db2start member 0`，他使用以下命令查询实例中所有成员的状态:

```
SELECT ID,
       varchar(HOME_HOST,10) AS HOME_HOST,
       varchar(CURRENT_HOST,10) AS CUR_HOST,
       varchar(STATE,21) AS STATE,
       ALERT
FROM SYSIBMADM.DB2_MEMBER
```

并获得以下输出:

```
ID     HOME_HOST CUR_HOST     STATE      ALERT
------ --------- ------------ ---------- -----
     0 so1       so1          STARTED    NO
     2 so2       so2          STARTED    NO
     4 so3       so3          STARTED    NO

     3 record(s) selected.
```

停止成员

DB2 成员在执行全局 **db2stop** 或个别 **db2stop** 命令时停止。本主题重点讨论如何停止单个成员。

关于此任务

如果要在主机上停止成员但让该主机上的其他实例进程保持启动并运行，请发出 `db2stop member` 命令。这意味着同一主机上的其他成员或集群高速缓存设施不受影响。但是，请注意，停止成员并非在主机上执行维护的充分必要条件，因为此主机仍是其他成员的可用故障转移目标。

限制

如果成员上有任何活动数据库连接，那么不能停止成员。如果成员处于轻量级重新启动方式，并且尚未完成成员崩溃恢复（即，至少其中一个数据库在该成员上处于不一致状态），那么不能使用 **db2stop** 命令将其停止。但是，可使用 **FORCE** 选项将其停止。强烈建议不停止接受崩溃恢复的成员。

过程

要停止成员，请发出以下命令:

```
db2stop member member-id
```

使用 DB2 pureScale Feature 激活数据库

可使用 **ACTIVATE DATABASE** 命令显式激活 DB2 pureScale 环境中的数据库，也可在第一个客户机连接时隐式激活此数据库。

开始之前

确保您具有下列其中一个权限级别:

- SYSMAINT
- SYSCTRL
- SYSADM

关于此任务

在 DB2 pureScale 环境中，最后一个用户断开连接后，数据库仍保持激活状态。为关闭数据库，您必须发出 **DEACTIVATE DATABASE** 命令或 **db2stop** 命令。

过程

要在 DB2 pureScale 环境中激活数据库，请发出 **ACTIVATE DATABASE** 命令，此命令将在所有成员中自动激活此数据库。

示例

要在所有成员中激活 TEST 数据库，请运行以下命令:

```
DB2 ACTIVATE DATABASE TEST
```

使用 DB2 pureScale Feature 取消激活数据库

在 DB2 pureScale 环境中，通过 **ACTIVATE DATABASE** 命令显式激活或由用户连接隐式激活的数据库只能通过 **DEACTIVATE DATABASE** 命令取消激活。

开始之前

确保您具有下列其中一个权限级别:

- SYSMAINT
- SYSCTRL
- SYSADM

关于此任务

在 DB2 pureScale 环境中，最后一个用户断开连接后，数据库仍保持激活状态。为关闭数据库，您必须发出 **DEACTIVATE DATABASE** 命令或 **db2stop** 命令。

过程

1. 要在 DB2 pureScale 环境中取消激活数据库，请发出 **DEACTIVATE DATABASE** 命令，此命令将在所有成员中自动取消激活此数据库。
2. 要取消激活特定成员，请在发出 **DEACTIVATE DATABASE** 命令时指定 **MEMBER** 参数。

示例

示例 1

要取消激活 TEST 数据库，请运行以下命令:

```
DB2 DEACTIVATE DATABASE TEST
```

示例 2

要取消激活特定成员，请运行以下命令:

```
DB2 DEACTIVATE DATABASE TEST MEMBER 10
```

在 DB2 pureScale 环境中进行维护

IBM DB2 pureScale Feature 的一个优点是它会在计划内维护活动期间维护数据库的持续可用性。

DB2 集群服务允许您在 DB2 pureScale实例中的集群高速缓存设施上执行滚动维护。同样，可轻松从 DB2 pureScale实例中除去要维护的成员或成员主机或将它们重新集成至 DB2 pureScale实例。

在 IBM Data Studio V3.1 或更高版本中，可以使用以下工具的任务助手: 使目标主机置于或脱离维护方式. 任务助手可以指导您执行以下过程: 设置选项、查看自动生成的命令以执行任务以及运行这些命令。有关更多详细信息，请参阅使用任务助手管理数据库。

在成员主机上执行维护

可在成员主机上执行维护或对其应用更新而不影响 DB2 pureScale实例中的数据库的可用性。

开始之前

在运行此任务之前，必须先关闭所有 DB2 服务器进程以及所有访问由 DB2 集群服务管理的文件系统的其他进程。主机上的所有实例必须停止，主机才能进入维护方式。在主机上停止该实例时，该主机不再是失败成员的可用恢复目标。

要执行此任务，您必须是 DB2 集群服务管理员。

过程

1. 作为实例用户，请执行下列步骤:
   a. 使用 **db2stop** 命令的 **QUIESCE** 参数来执行 成员 放弃操作，如下所示:

      ```
      db2stop member member-id quiesce 30
      ```

      其中 *member-id* 表示您要置于维护方式的 成员。

      有关停止 成员 的更多信息，请参阅 第 76 页的『停顿成员』。
   b. 在您停止该主机前，确保仅 成员 以主机上的维护为目标。在成员主机上停止此实例，如下所示:

      ```
      db2stop instance on host-name
      ```

      其中 *host-name* 表示指定 成员 或 CF 的主机名。
2. 作为 DB2 集群服务 管理员，请执行下列步骤:
   a. 在主机上通过发出以下命令将集群管理器置于维护方式:

      ```
      DB2DIR/bin/db2cluster -cm -enter -maintenance
      ```

      其中，*DB2DIR* 表示 DB2 副本的安装位置。
   b. 通过发出以下命令将主机上的集群文件系统服务置于维护方式:

      ```
      DB2DIR/bin/db2cluster -cfs -enter -maintenance
      ```

其中，*DB2DIR* 表示 DB2 副本的安装位置。

c. 重新引导主机。

下一步做什么

1. 作为 DB2 集群服务 管理员，请执行下列步骤:
   a. 执行任何计划的维护活动。例如，更改硬件配置。请注意，更新到 DB2 pureScale 软件组件需要您将所有主机置于维护方式。
   b. 通过发出以下命令在主机上退出集群管理器维护方式:

      ```
      DB2DIR/bin/db2cluster -cm -exit -maintenance
      ```

      其中，*DB2DIR* 表示 DB2 副本的安装位置。
   c. 通过发出以下命令在主机上退出集群文件系统维护方式:

      ```
      DB2DIR/bin/db2cluster -cfs -exit -maintenance
      ```

      其中，*DB2DIR* 表示 DB2 副本的安装位置。
2. 作为实例用户，请执行下列步骤:
   a. 通过发出以下命令重新启动主机上的实例:

      ```
      db2start instance on host-name
      ```

      其中 *host-name* 表示指定 成员 或 CF 的主机名。
   b. 通过发出以下命令重新启动 成员:

      ```
      db2start member member-id
      ```

      其中 *member-id* 表示您要置于维护方式的成员。

替换两个集群高速缓存设施

可使用卷动升级技巧来替换两个集群高速缓存设施，这样可确保 DB2 pureScale实例不会遇到停机。

开始之前

您必须有 root 用户权限才能执行此任务。

确保您知道哪个集群高速缓存设施当前是主集群高速缓存设施以便您不会无意中将其停止。而且，辅助集群高速缓存设施需要处于 PEER 状态。可通过查询 DB2_CF 管理视图来完成这两项，如下所示: `SELECT * FROM SYSIBMADM.DB2_CF`

过程

使用以下步骤来替换这两个集群高速缓存设施 (CF):

1. 使用以下命令来停止辅助集群高速缓存设施: `db2stop CF CF-identifier` 有关停止集群高速缓存设施的更多信息，请参阅第 69 页的『停止集群高速缓存设施』。
2. 使用以下命令来停止辅助集群高速缓存设施上的实例以确保没有其他 DB2 进程在运行: `db2stop instance on host-name`
3. 使用以下命令来删除集群高速缓存设施: `db2iupdt -drop -cf host-name instance-name`
4. 使用以下命令来添加新主机: `db2iupdt -add -cf host-name:net-name instance-name`

5. 使用以下语句来查找新添加的集群高速缓存设施的标识: `SELECT * from SYSIBMADM.DB2_CF`
6. 使用以下命令来启动新添加的集群高速缓存设施: `db2start CF CF-identifier` 有关启动集群高速缓存设施的更多信息，请参阅第 68 页的『启动集群高速缓存设施』。
7. 一旦新集群高速缓存设施（现在是辅助集群高速缓存设施）变为 PEER 状态后，按步骤 1 和步骤 2 来停止主集群高速缓存设施。 DB2 集群服务会将主角色故障转移至辅助集群高速缓存设施。
8. 重复步骤 3 到 6 以替换旧的主集群高速缓存设施并将其作为辅助集群高速缓存设施重新启动。

向共享文件系统添加磁盘

创建共享文件系统后，可能会出现需要将其他磁盘添加至文件系统的情况。

开始之前

运行此命令的用户标识必须拥有这些磁盘并对它们具有读写访问权。本地主机上的任何其他文件系统当前不能使用这些磁盘。要执行此任务，您必须是创建此文件系统的用户标识或 DB2 集群服务管理员。

过程

此磁盘对主机实际可用后，可使用 **db2cluster** 命令将此磁盘添加至需要更多空间的文件系统。 例如:

```
db2cluster -add -filesystem filesystem-name -disk disk-name
```

结果

此任务完成时，该磁盘会添加至该文件系统，并且该文件系统的条带集包括此磁盘。

下一步做什么

如果要确保此文件系统均衡分布在此文件系统中的所有磁盘上，应执行重新平衡操作。

从共享文件系统除去磁盘:

如果决定文件系统不需要当前安装在其上的所有磁盘或者给定磁盘使用需要更多存储空间的文件系统部署可能更好，那么可使用 **db2cluster** 命令轻松除去磁盘。

开始之前

确保文件系统中余下磁盘上有足够的空间来容纳要除去的磁盘上的文件。

关于此任务

此任务从文件系统中除去磁盘并使其可在其他任何位置可用。

限制

- 仅当实例所有者创建文件系统时，实例所有者才能执行此任务；否则只有 DB2 集群服务才能从文件系统中除去磁盘。

- 如果文件系统中只余下一个磁盘，那么不能使用此方法除去此磁盘。按com.ibm.db2.luw.admin.sd.doc/doc/t0056124.dita中所述改为删除该文件系统。
- 不能除去仲裁磁盘。要查明磁盘是不是仲裁磁盘，请参阅com.ibm.db2.luw.admin.sd.doc/doc/c0056704.dita。
- 一次只能从文件系统中除去一个磁盘。

过程

要从共享文件系统中除去磁盘，请使用 **db2cluster** 命令:

```
db2cluster -cfs -remove -filesystem filesystem-name -disk disk-name
```

下一步做什么

从文件系统中除去磁盘时，运行重新平衡操作以确保此文件系统均衡分布在此文件系统中的所有余下磁盘上，如『重新平衡文件系统』中所述。

重新平衡文件系统:

您可在添加或除去磁盘后重新平衡文件系统。使用 **db2cluster** 命令执行重新平衡操作。

开始之前

确保此文件系统是在运行此命令之前安装的。因为重新平衡操作需要大量文件输入和输出，所以建议您在集群中的系统活动较少时运行此操作。

关于此任务

此任务通过在文件系统上的所有磁盘中重新分割数据来重新平衡文件系统。仅当实例所有者创建文件系统时，此实例所有者才能执行此任务；否则，只有 DB2 集群服务管理员才能重新平衡文件系统。

过程

使用 **db2cluster** 命令来重新平衡文件系统:

```
db2cluster -rebalance -filesystem filesystem-name
```

停顿成员

有时可能需要您临时从集群中除去成员（例如，维护操作）。

关于此任务

在 DB2 pureScale 环境中，**db2stop** 和 **STOP DATABASE MANAGER** 命令提供了可选 **QUIESCE** 参数。可选 **QUIESCE** 参数允许您针对单个成员放弃所有活动并关闭此成员。放弃该成员时，自动客户机重新路由 (ACR) 和工作负载均衡会将新事务和新连接重新路由至其他成员。

针对成员发出 **db2stop QUIESCE** 命令时，应用程序可能会有以下行为:

- 如果指定超时值，那么应用程序必须在此时间内完成其活动工作单元。
- 如果未指定超时（或值为 -1），那么服务器将无限期等待直到成员上的所有活动事务和关联连接结束。
- 如果指定超时值 0，那么会立即强制连接关闭。

注: 发出 stop member quiesce（不指定超时或指定非零超时）时，系统会标记活动事务以延迟连接终止。事务成功落实或回滚时，此连接将终止，除非发生下列其中一种情况:

- 此连接使用全局变量
- 使用加密密码
- 有一个打开的 WITH HOLD 游标
- 使用了已声明临时表 (DGTT)
- 设置了 TRANSFORM GROUP
- 已更改 SESSION AUTHID
- 使用了 PL/SQL 程序包或 SQL/PL 模块
- 使用了游标变量
- 使用了顺序值
- 使用了创建临时表 (CGTT) 和 PRESERVE ROWS
- 程序包中进行了预编译的动态 SQL 与 KEEPDYNAMIC YES 绑定在一起。预编译存储过程或用户定义的函数中的语句或 IBM 非嵌入式 API（如 CLI/JDBC/ODBC/.NET）预编译语句时，此限制不适用。

如果出现上述任何情况，那么您需要除去它（例如，WITH HOLD 游标）或显式停止连接。如果未出现上述任何条件，那么客户机连接将在下一个事务端点终止。如果客户机（.NET、CLI 和 JDBC）支持无缝 ACR，那么连接会自动重新路由至另一成员。

过程

运行 **db2stop** 命令或 **STOP DATABASE MANAGER** 命令并指定 **QUIESCE** 参数。
以下示例使成员 2 脱机并给出 30 分钟来完成活动工作负载。30 分钟后，成员上的所有应用程序被强制关闭。

```
db2stop MEMBER 2 QUIESCE 30
```

为使成员 2 变回联机，请对其发出 **db2start** 命令。

```
db2start MEMBER 2
```

示例

以下示例使用 db2InstanceStop API 来使成员 10 脱机。

```
struct sqlca sqlca;     // sqlca to carry the sqlcode
struct db2InstanceStopStruct instanceStopStruct;
struct db2StopOptionsStruct  stopOptions;

instanceStopStruct.iIsRemote = FALSE; // demo local instance
instanceStopStruct.piRemoteInstName = NULL;
instanceStopStruct.piCommData = NULL; // don't care DAS
instanceStopStruct.piStopOpts = &stopOptions;

stopOptions.iOption = SQLE_QUIESCE; // Member quiesce option
stopOptions.iIsType = TRUE;
stopOptions.iType = DB2_NODE_MEMBER;
stopOptions.iIsNodeNum = TRUE;
stopOptions.iNodeNum = 10;
stopOptions.iQuiesceDeferMinutes = 0; // no explicit timeout

// Finally, invoke the API to shut down the instance
db2InstanceStop(db2Version1010, &instanceStopStruct, &sqlca);
```

以下示例使用 db2InstanceStop API 以使成员 10 脱机并指定 5 分钟超时。

```
struct sqlca sqlca;     // sqlca to carry the sqlcode
struct db2InstanceStopStruct instanceStopStruct;
struct db2StopOptionsStruct  stopOptions;

instanceStopStruct.iIsRemote = FALSE; // demo local instance
instanceStopStruct.piRemoteInstName = NULL;
instanceStopStruct.piCommData = NULL; // don't care DAS
instanceStopStruct.piStopOpts = &stopOptions;

stopOptions.iOption = SQLE_QUIESCE; // Member quiesce option
stopOptions.iIsType = TRUE;
stopOptions.iType = DB2_NODE_MEMBER;
stopOptions.iIsNodeNum = TRUE;
stopOptions.iNodeNum = 10;
stopOptions.iQuiesceDeferMinutes = 5; // timeout of 5 minutes

// Finally, invoke the API to shut down the instance
db2InstanceStop(db2Version1010, &instanceStopStruct, &sqlca);
```

将主机置于维护方式

如果要对 DB2 集群服务应用软件更新，那么必须将目标主机置于维护方式。如果要确保更新主机上的操作系统或硬件时不在主机上重新启动成员或集群高速缓存设施，那么也可将主机置于维护方式。

开始之前

在运行此命令之前，必须先关闭任何 DB2 服务器进程以及访问由 DB2 集群服务管理的文件系统的任何其他进程。主机上的所有实例必须停止，主机才能进入维护方式。实例在主机上停止时，此主机不再是失败成员的可用恢复目标。

要执行此任务，您必须是 DB2 集群服务管理员。

如果任何服务器进程仍在主机上处于活动状态，那么主机不能进入维护方式。即使您对此命令指定 **-force** 选项，也只会关闭集群管理器和共享文件系统。

为避免 DB2 pureScale实例遇到停机，请遵循以下建议:

- 不要同时将两个 集群高速缓存设施 的主机都置于维护方式。
- 不要同时将具有 成员 的所有主机都置于维护方式。
- 在集群中的总共 N 个主机中，(N/2) + 1 个主机必须处于联机状态，并且未处于维护方式。

关于此任务

同时置于维护方式的主机越多，停机或故障时可用于恢复的主机就越少。因此，建议不要将多个主机同时置于维护方式。

过程

- 要将主机置于维护方式，请参阅 第 73 页的『在成员主机上执行维护』。
- 要在所有主机上输入维护方式，请参阅 第 79 页的『将集群置于维护方式』。

将集群置于维护方式

当您对集群中主机上的操作系统或硬件进行更新时，可以将集群置于维护方式。

要执行此任务，您必须是 DB2 集群服务管理员。

过程

1. 作为实例用户，请执行下列步骤:
   a. 通过在单个主机上发出以下命令停止所有主机上的数据库管理器:

      ```
      su -iname
      db2stop
      exit
      ```

      其中 *iname* 表示实例所有者的名称。
   b. 在每个主机上，通过发出以下命令停止 DB2 实例:

      ```
      db2stop instance on hostname
      ```

      其中，*hostname* 表示指定 成员 或 CF 的主机名，且对于集群上的每个主机运行 `db2stop instance on hostname` 命令。
2. 作为 DB2 集群服务 管理员，请执行下列步骤:
   a. 在每个资源组上发出以下命令以卸下资源组:

      ```
      rgreq -o stop resource-group-name
      ```

      其中，*resource-group-name* 表示通过运行 `lsrg | grep db2mnt` 命令返回的每个资源组的名称。您可以通过使用 `lssam -g resource-group-name` 命令验证资源组是否脱机。
   b. 在所有主机上通过发出以下命令将集群管理器置于维护方式:

      ```
      DB2DIR/bin/db2cluster -cm -enter -maintenance -all
      ```

      其中，*DB2DIR* 表示 DB2 副本的安装位置。
   c. 通过发出以下命令将主机上的集群文件系统服务置于维护方式:

      ```
      DB2DIR/bin/db2cluster -cfs -enter -maintenance -all
      ```

      其中，*DB2DIR* 表示 DB2 副本的安装位置。
   d. 通过发出 **mmlsconfig** 命令检查 **autoload** 配置参数的值:

      ```
      mmlsconfig | grep autoload
      ```

      如果将 autoload 配置参数的值设为 YES，那么发出以下命令以将该值更改为 NO，以阻止节点自动启动:

      ```
      mmchconfig autoload=no -N all
      ```
   e. 停止所有主机。

下一步做什么

1. 作为 DB2 集群服务 管理员，请执行下列步骤:
   a. 执行任何计划的维护活动。例如，安装修订包更新，或对您的 DB2 pureScale 环境进行网络拓扑的更改。
   b. 通过发出 **mmlsconfig** 命令检查 **autoload** 配置参数的值:

      ```
      mmlsconfig | grep autoload
      ```

如果将 **autoload** 配置参数的值设为 NO，那么发出以下命令以将该值更改为 YES:

```
mmchconfig autoload=yes -N all
```

c. 通过发出以下命令在主机上退出集群管理器维护方式:

```
DB2DIR/bin/db2cluster -cm -exit -maintenance
```

其中，*DB2DIR* 表示 DB2 副本的安装位置。

d. 通过发出以下命令，确保所有 成员 和域为活动状态:

```
DB2DIR/bin/db2cluster -cm -list -host -state
```

其中，*DB2DIR* 表示 DB2 副本的安装位置。

e. 通过在每个资源组上发出以下命令重置资源组状态:

```
rgreq -o cancel resource-group-name
```

其中，*resource-group-name* 表示通过运行 `lsrg | grep db2mnt` 命令返回的每个资源组的名称。

f. 通过发出以下命令在主机上退出集群文件系统维护方式:

```
DB2DIR/bin/db2cluster -cfs -exit -maintenance -all
```

其中，*DB2DIR* 表示 DB2 副本的安装位置。如果超时，通过发出 *DB2DIR*/bin/db2cluster -cfs -list -host -state 命令以检查集群文件系统的状态。

2. 作为实例用户，请执行下列步骤:

a. 在每个主机上，通过发出以下命令启动 DB2 实例:

```
db2start instance on hostname
```

其中 *hostname* 表示指定 成员 或 CF 的主机名，且对于集群上的每个主机运行 db2start instance on *hostname* 命令。

b. 通过发出以下命令启动数据库管理器:

```
su -iname
   db2start exit
```

其中 *iname* 表示实例所有者的名称。

移动 DB2 成员或集群高速缓存设施

您可能因为许多原因要将 DB2 成员或集群高速缓存设施从一个主机移至另一个主机。此任务概述许多可能的场景以及一些要考虑的其他因素。

关于此任务

使用 **db2iupdt -add** 或 **-drop** 操作在DB2 pureScale环境中执行更改时，可能需要备份数据库。必须通过其中一个预先存在的实例成员来执行此备份操作。

限制

系统最多支持两个集群高速缓存设施。如果要移动一个，那么您必须先删除另一个，然后在新主机上添加第二个。如果您尝试在删除第一个集群高速缓存设施之前删除第三个，那么您将接收到错误。

必须始终至少有一个 DB2 成员和一个集群高速缓存设施。如果要移动一个，那么必须先在新主机上添加新 DB2 成员或集群高速缓存设施，然后删除原始那一个。如果您尝试删除最后一个余下成员或集群高速缓存设施，那么您将接收到错误。

过程

- 要仅移动 DB2 成员，请执行以下操作:
  1. 通过使用 **db2stop** 命令在所有主机上停止 DB2 pureScale 实例。 此步骤是必需的，因为 **db2iupdt -add** 和 **-drop** 命令必须脱机运行。
  2. 添加新的成员。 此添加操作确保始终至少有一个 DB2 成员。 例如，将名称为 Member2 且网络名为 Netname2 的 DB2 成员添加至DB2 pureScale 实例 sdinstA:

     ```
     db2iupdt -add -m Member2:Netname2 sdinstA
     ```

  3. 删除原始 DB2 成员。 例如，从DB2 pureScale 实例 sdinstA 中删除名称为 Member1 且网络名为 Netname1 的成员:

     ```
     db2iupdt -drop -m Member1:Netname1 sdinstA
     ```

  4. 如果数据库可恢复，请通过启动此进程之前存在的成员在此 DB2 pureScale 实例上备份此数据库。 可恢复数据库的 **logarchmeth1** 或 **logarchmeth2** 数据库配置参数设为 OFF 以外的值。
- 要移动多个 DB2 成员的其中一个，请执行以下操作:
  1. **db2iupdt** -add 或 -drop 命令必须脱机运行，以便使用 **db2stop** 命令在所有主机上停止该实例。
  2. 删除要移动的 DB2 成员。 例如，从 DB2 pureScale 实例 sdinstA 中删除网络名为 Netname2 的成员 Member2:

     ```
     db2iupdt -drop -m Member2:Netname2 sdinstA
     ```

  3. 添加新的 DB2 成员。 例如，将网络名为 Netname3 的 Member3 作为成员添加至DB2 pureScale 实例 sdinstA:

     ```
     db2iupdt -add -m Member3:Netname3 sdinstA
     ```

  4. 如果数据库可恢复，请通过启动此进程之前存在的成员在此 DB2 pureScale 实例上备份此数据库。 可恢复数据库的 **logarchmeth1** 或 **logarchmeth2** 数据库配置参数设为 OFF 以外的值。
- 要仅移动集群高速缓存设施，请执行以下操作:
  1. **db2iupdt** -add 或 -drop 命令必须脱机运行，以便使用 **db2stop** 命令在所有主机上停止该实例。
  2. 添加新的集群高速缓存设施。 此添加操作确保始终至少有一个集群高速缓存设施。 例如，将名称为 cf2 且网络名为 NetCF2 的集群高速缓存设施添加至DB2 pureScale 实例 sdinstA:

     ```
     db2iupdt -add -cf cf2:NetCF2 sdinstA
     ```

  3. 删除初始集群高速缓存设施。 例如，从DB2 pureScale 实例 sdinstA 中删除名称为 cf1 且网络名为 NetCF1 的集群高速缓存设施:

     ```
     db2iupdt -drop -cf cf1:NetCF1 sdinstA
     ```
- 要移动两个集群高速缓存设施的其中一个，请执行以下操作:
  1. **db2iupdt** -add 或 -drop 命令必须脱机运行，以便使用 **db2stop** 命令在所有主机上停止该实例。

2. 系统最多支持两个集群高速缓存设施，所以必须先删除一次。 例如，从DB2 pureScale 实例 sdinstA 中删除名称为 cf1 且网络名为 NetCF1 的集群高速缓存设施:

   ```
   db2iupdt -drop -cf cf1:NetCF1 sdinstA
   ```

3. 在另一主机上添加第二个集群高速缓存设施。 例如，将名称为 cf2 且网络名为 NetCF2 的集群高速缓存设施添加至DB2 pureScale 实例 sdinstA:

   ```
   db2iupdt -add -cf cf2:NetCF2 sdinstA
   ```

在 DB2 pureScale 环境中从失败的 db2iupdt -add 或 -drop 操作恢复

有许多原因可能会导致 **db2iupdt** -add 或 -drop 操作失败，从而使 DB2 pureScale 环境拓扑处于不一致状态。

如果此问题要求硬件修订（例如，主机故障），请先完成该任务。硬件启动并运行后，可将 **db2iupdt** 命令与 -fixtopology 选项配合使用以自动回滚先前的 **db2iupdt** -add 操作，或完成先前的 **db2iupdt** -drop 操作。

如果尝试删除带有多个 DB2 成员的成员主机的操作失败，那么 **db2iupdt** -fixtopology 命令仅对失败的一个特定成员完成先前的 **db2iupdt** -drop。可能需要重新运行原始 **db2iupdt** -drop 命令来完成删除成员主机上的另一 DB2 成员的操作。

清除资源模型警报

在 DB2 pureScale 环境中，与硬件或软件相关的警报可存在于成员、集群高速缓存设施 (CF) 及主机的状态表中。在其他情况下，警报条件是瞬态条件，所以警报字段可自动消失；但是，在其他情况下，警报字段将保留设置直到管理员解决该问题并手动清除警报。

关于此任务

运行此命令时，指定的成员、集群高速缓存设施或主机上针对该实例的所有警报将被清除。

要执行此任务，您必须是 SYSADM、SYSMAINT 或 SYSADM 组中的用户。

过程

要清除成员、集群高速缓存设施或主机的警报，请发出以下命令:

```
db2cluster -clear -alert  [-member member-id | -cf cf-id | -host host-name]
```

结果

命令完成后，系统可能会也可能不会采取与已清除其警报的组件相关的特定措施。在某些情况下，成员将返回至其原始主机，但如果轻量级重新启动失败导致警报，那么不会采取任何措施。

更改集群管理器定额类型

DB2 集群服务仲裁磁盘是创建第一个实例时在安装 GUI 中一开始设置的。仅当指定磁盘发生问题或配置更改时，才更改集群管理器定额类型。

开始之前

- 必须在所有主机上停止 DB2 pureScale实例，才能更改定额类型。集群管理器必须保持联机。
- 如果要切换至仲裁磁盘，那么可以指定设备路径格式中的磁盘或将其作为 PVID 或 WWN 编号指定。您指定的磁盘不能用于任何其他用途。
- 可使用设备路径格式输入集群管理器的磁盘选项（例如，/dev/hdisk0）或作为 PVID（端口虚拟局域网标识）或 WWN（全球名称）编码输入。
- 要执行此任务，您必须是 DB2 集群服务管理员。

过程

- 要将定额类型更改为仲裁磁盘，请发出以下命令:

  ```
  db2cluster -cm -set -tiebreaker -disk diskname
  ```

- 要将定额类型更改为主节点集，请发出以下命令:

  ```
  db2cluster -cm -set -tiebreaker -majority
  ```

更改共享文件系统定额类型

共享文件系统定额类型是在设置 IBM DB2 pureScale Feature 期间自动指定的。因为所选定额类型基于集群中的主机数，所以在 DB2 实例中添加或除去主机后您必须更改定额类型。

开始之前

- 必须先在所有主机上关闭 DB2 实例和共享文件系统集群，才能更改定额类型。如果共享文件系统集群仍处于活动状态，请发出 **db2cluster -cfs -stop** 命令。
- 如果要切换至仲裁磁盘，那么您指定的磁盘必须为设备路径格式（例如: `/dev/hdisk2`）并且它在所有节点上必须可访问。与集群管理器的仲裁磁盘不同，此磁盘可在共享文件系统中重复使用，也可在已定义文件系统上的磁盘中重复使用。
- 要执行此任务，您必须是 DB2 集群服务管理员。

关于此任务

GPFS™ 有两种定额类型:

- *仲裁磁盘*: 用于确定 GPFS 集群中的哪些存活主机组具有可运行定额的共享磁盘
- *主节点集*: 主项具有可运行定额的 GPFS 中存活主机组

定额类型由 DB2 安装程序根据 表 7 中的规则自动选择。

表 7. GPFS 集群中的主机数与定额类型之间的关系

GPFS 集群中的主机总数	GPFS 定额类型
等于或小于 8	仲裁磁盘
大于 8	主节点集

GPFS 中的主机数从 8 个开始增加或从 9 个开始减少时，将返回一条错误消息，指示应更改定额类型。

过程

- 要将定额类型更改为仲裁磁盘，请发出以下命令:

  ```
  db2cluster -cfs -set -tiebreaker -disk diskname
  ```

- 要将定额类型更改为主节点集，请发出以下命令:

  ```
  db2cluster -cfs -set -tiebreaker -majority
  ```

更改主机故障检测时间

快速故障检测对 DB2 pureScale实例非常重要，因为越快检测到主机故障，就可越早开始从故障恢复。也就是说，主机故障检测时间的主动设置不适用于高延迟环境，所以用户可能想要调整此设置。

开始之前

- 必须在所有主机上关闭 DB2 实例和共享文件系统集群，才能更改主机故障检测时间。如果共享文件系统集群仍处于活动状态，请发出 **db2cluster -cfs -stop -all** 命令。
- 要执行此任务，您必须是 DB2 集群服务管理员。

关于此任务

可使用 **db2cluster** 命令来调整 DB2 pureScale实例的快速故障检测时间。此命令指定它在检测主机故障或网络分区时所耗用的时间。要确定当前设置，请使用以下命令:

```
db2cluster -cm -list -HostFailureDetectionTime
```

过程

要更改主机故障检测时间，请使用以下命令:

```
db2cluster -cm -set -option HostFailureDetectionTime -value value
```

结果

该命令完成后，将对 DB2 集群服务域中的所有 DB2 pureScale实例应用新设置。

第 2 部分 数据库

第 5 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（内置对象）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，您必须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，在上面描述的方案中不存在一对一关系。

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，所以必须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，它必须考虑许多因素，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。

建议的文件系统

DB2 数据库可在 DB2 产品的运行平台支持的许多文件系统中运行。

您可使用的文件系统取决于您是否要使用 IBM DB2 pureScale Feature。

- 『没有 DB2 pureScale Feature』
- 第 89 页的『DB2 pureScale 环境』

没有 DB2 pureScale Feature

的 DB2 环境

IBM 为 DB2 for Linux, UNIX, and Windows建议了第 89 页的表 8 中所显示的文件系统。

注: 第 89 页的表 8中未列示的文件系统可能存在未记录的限制。对未列示文件系统的支持可能受限。

表 8. 为 DB2 for Linux, UNIX, and Windows建议的文件系统

平台	操作系统	建议的文件系统
Linux	Red Hat Enterprise Linux (RHEL)	• IBM 通用并行文件系统[1] (GPFS) • ext3 • VERITAS File System (VxFS) • 带有 IBM N-series 的网络文件系统 (NFS[2]) • 带有网络设备文件程序的网络文件系统 (NFS[2])
	SUSE Linux Enterprise Server (SLES)	• GPFS • ext3 • VERITAS File System (VxFS) • 带有 IBM N-series 的 NFS[2] • 带有网络设备文件程序的 NFS[2]
UNIX	AIX	• GPFS • 增强型日志文件系统 (JFS2) • 带有 IBM N-series 的 NFS[2] • 带有网络设备文件程序的 NFS[2] • VxFS
	HP-UX	• HP JFS[3] (VxFS)
	Solaris	• UNIX 文件系统 (UFS) • ZFS • VxFS
Windows	所有 Windows 产品	NTFS

注意:

[1] 有关 GPFS 的更多信息，请参阅 http://www-03.ibm.com/systems/clusters/software/gpfs/index.html。

[2] 有关要验证哪些 NFS 版本以供在各种操作系统上使用的详细信息，请参阅 http://www-01.ibm.com/support/docview.wss?uid=swg21169938。

[3] HP JFS on HP-UX 是 VxFS 的 OEM 版本。

DB2 pureScale 环境

DB2 pureScale Feature 需要 IBM General Parallel File System (GPFS)。此文件系统由 DB2 pureScale Feature 安装程序自动安装并配置（如果它尚未安装）。有关所需特定 GPFS 版本的更多信息，请参阅安装先决条件。有关 GPFS 的更多信息，请参阅http://www-03.ibm.com/systems/clusters/software/gpfs/index.html。

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。

系统已为您创建分层目录结构。可通过对 **CREATE DATABASE** 命令指定目录路径或驱动器来指定该结构的位置；如果未指定位置，那么会使用缺省位置。

在您在 **CREATE DATABASE** 命令中您指定为数据库路径的目录中，会创建使用该实例名称的子目录。

在实例名子目录内，会创建*分区全局目录*。分区全局目录包含与新数据库相关联的全局信息。分区全局目录名为 NODE*xxxx*/SQL*yyyyy*，其中 *xxxx* 是数据分区号，*yyyyy* 是数据库标记（编号大于等于 1）。

在*分区全局目录*下，会创建*特定于成员*的目录。特定于成员的目录包含本地数据库信息。特定于成员的目录名为 MEMBER*xxxx*，其中 *xxxx* 是成员号。

- 在 DB2 pureScale 环境中，每个成员都有一个名为 MEMBER0000、MEMBER0001 等特定于成员的目录。
- 在分区数据库环境中，成员编号与其对应分区号之间存在一对一映射，因此，每个成员和分区对应一个 NODE*xxxx* 目录。特定于成员的目录始终名为 MEMBERxxxx，并且它们始终驻留在分区全局目录下。
- 企业服务器版环境在单个成员上运行，并且有一个特定于成员的目录，名为 MEMBER0000。

分区全局目录

分区全局目录具有以下路径：*your_instance*/*NODExxxx*/SQLxxxxx。

分区全局目录包含以下文件：

- 全局死锁写至文件事件监视器文件，它指定相对路径或根本不指定任何路径。
- 表空间信息文件。

  SQLSPCS.1 和 SQLSPCS.2 文件都包含表空间信息。这两个文件互为副本以实现备份。
- 存储器组控制文件。

  文件 SQLSGF.1 和 SQLSGF.2 包含与数据库的自动存储器功能相关联的存储器组信息。这两个文件互为副本，旨在进行维护和备份。这些文件是在您使用 **CREATE DATABASE** 命令创建数据库或将非自动存储器数据库升级至 DB2 V10.1 或更高版本时创建的。
- 临时表空间容器文件。

  新容器的缺省目录是 instance/NODExxxx/<db-name>。这些文件由每个成员在本地管理。系统会使这些表空间文件名对每个成员唯一，方法是将成员号插入到文件名中，例如：*/storage path*/SAMPLEDB/T0000011/C0000000.TMP/SQL00002.MEMBER0001.TDA
- 全局配置文件。

  全局配置文件 SQLDBCONF 包含数据库配置参数，它们引用在数据库中必须保持一致的单个共享资源。切勿编辑此文件。要更改配置参数，请使用 **UPDATE DATABASE CONFIGURATION** 和 **RESET DATABASE CONFIGURATION** 命令。
- 历史记录文件。

  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、表空间更改和其他数据库更改的历史记录信息。

  DB2TSCHG.HIS 文件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。可在文本编辑器中检查历史记录文件的内容。

- 与日志记录相关的文件。

  全局日志控制文件 SQLOGCTL.GLFH.1 和 SQLOGCTL.GLFH.2 包含数据库级别的恢复信息，例如，与数据库脱机时添加新成员及维护成员间的公共日志链相关的信息。日志文件本身存储在分区全局目录内的 LOGSTREAM*xxxx* 目录（每个成员对应一个目录）中。
- 锁定文件。

  实例数据库锁定文件 SQLINSLK 和 SQLTMPLK 有助于确保数据库仅被数据库管理器的一个实例使用。
- 自动存储器容器

特定于成员的目录

特定于成员的目录具有以下路径: /NODE*xxxx*/SQL*xxxx*/MEMBER*xxxx*

此目录包含与创建的第一个数据库相关联的对象，后续数据库被给予更高的编号 SQL00002，以此类推。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。

数据库目录包含以下文件:

- 缓冲池信息文件。

  SQLBP.1 和 SQLBP.2 文件都包含缓冲池信息。这两个文件互为副本以实现备份。
- 本地事件监视器文件。
- 与日志记录相关的文件。

  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。在 DB2 pureScale 环境中，每个成员都有自己的日志流和本地 LFH 文件集，它们存储在每个特定于成员的目录中。

  提示: 将日志子目录映射至您未用于存储数据的磁盘。通过这样做，您可将磁盘问题限制为数据或日志方面，从而避免数据和日志同时出现磁盘问题。将日志子目录映射至未用于存储数据的磁盘可大幅提高性能，因为日志文件和数据库容器不会为同一磁盘头的移动而竞争。要更改日志子目录的位置，请使用 **newlogpath** 数据库配置参数。
- 本地配置文件。

  本地 SQLDBCONF 文件包含数据库配置信息。切勿编辑此文件。要更改配置参数，请使用 **UPDATE DATABASE CONFIGURATION** 和 **RESET DATABASE CONFIGURATION** 命令。

在创建数据库的同时，还创建了详细死锁事件监视器。在企业服务器版环境和分区数据库环境中，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。在 DB2 pureScale 环境中，详细死锁事件监视器文件存储在分区全局目录中。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。这样做可避免事件监视器使用太多磁盘空间。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。

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* 子目录包含缺省系统管理的空间 (SMS) 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（包含块分配信息，如果它是 MDC 或 ITC 表）
- 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）

数据库配置文件

为每个数据库创建*数据库配置文件*。在 V8.2 之前，此文件称为 SQLDBCON，而在 V8.2 及更高版本中则称为 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 V8.2 之前）或 SQLDBCONF（在 V8.2 及更高版本中）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，并创建一个存取方案以满足 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存取方案，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。

注: 在DB2 pureScale 环境中，不必运行 CATALOG NODE 命令，因为 DB2 pureScale Feature 充当单个分区。

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。对于具有客户机可访问的一个或多个数据库的每个远程数据库分区服务器，该目录包含一个对应条目。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

在每条定义了数据库的路径（或者 Windows 操作系统的“驱动器”）中，都存在*本地数据库目录*文件。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 Linux 和 UNIX 平台上及分区数据库环境中，应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。

关于此任务

要启用数据库分区，在启动数据库管理器之前 db2nodes.cfg 文件必须位于实例主目录的 sqllib 子目录中。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，分发映射（它指定数据分布方式）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

用于分区间通信的 IP 地址的主机名。对主机名使用标准名称。/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。在这种情况下，netname 用于大多数通信，而 host 名称仅用于 **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 来覆盖此行为。

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 9. 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，必须小心保护此文件中的信息的完整性，这是因为数据库分区功能要求您发出 **START DBM** 时将节点配置文件锁定，而在发出 **STOP DBM** 结束数据库管理器之后将其解锁。将此文件锁定之后，**START DBM** 命令就可以在必要时对其进行更新。例如，您可以发出 **START DBM** 并指定 **RESTART** 选项或 **ADD DBPARTITIONNUM** 选项。

注: 如果 **STOP DBM** 命令未成功并且未将节点配置文件解锁，请发出 **STOP DBM FORCE** 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。

关于此任务

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。如果计划更改数据库，请复查配置参数的值。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 V9.1 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 20 页的『自动功能』。

过程

要更改数据库或数据库管理器配置参数，请执行以下操作:

- 使用配置顾问程序。
  - 从命令行使用 **AUTOCONFIGURE** 命令。
  - 从客户机应用程序调用 db2AutoConfig API。
- 从 IBM Data Studio 中，右键单击实例或数据库以打开任务助手来更改数据库管理器或数据库配置参数。
- 从命令行使用 **UPDATE DATABASE MANAGER CONFIGURATION** 和 **UPDATE DATABASE CONFIGURATION** 命令。

  例如，要更新数据库管理器配置中的各个参数，请输入:

  ```
  UPDATE DBM CFG USING config_keyword value
  ```

  要更新数据库配置中的各个参数，请输入:

  ```
  UPDATE DB CFG FOR database_alias
  USING config_keyword value
  ```

  可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。
- 从客户机应用程序调用 db2CfgSet API。

下一步做什么

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。

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、可用空间以及使用可变长度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
- XML 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

根据您的需要以及配置参数设置的不同，日志文件的空间需求也将有所变化。

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

$$(logprimary + logsecond) \times (logfilsiz + 2) \times 4096$$

其中:

- `logprimary` 是数据库配置文件中定义的主日志文件数
- `logsecond` 是数据库配置文件中定义的辅助日志文件数；在此计算中，不能将 `logsecond` 设为 `-1`。（如果 `logsecond` 设为 `-1`，那么表明您正在请求获取不受限的活动日志空间。）
- `logfilsiz` 是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

前滚恢复

如果已对数据库启用前滚恢复，那么可考虑特殊日志空间需求:

- 如果 **logarchmeth1** 配置参数设为 LOGRETAIN，那么将在日志路径目录中归档日志文件。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。
- 如果 **logarchmeth1** 配置参数设为 USEREXIT、DISK 或 VENDOR，那么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文件
  - 格式化新的日志文件，以供将来使用
- 如果对归档日志文件启用压缩，那么可减少存储这些文件的成本。
  - 例如，如果 **logarchmeth1** 配置参数设为 DISK、TSM 或 VENDOR，并且您将 **logarchcompr1** 配置参数设为 ON，那么系统会压缩归档日志文件并且可减少存储这些文件的成本。如果以动态方式启用压缩，那么不会压缩已存储的现有归档日志文件。启用压缩后，压缩从当前活动日志文件开始。

循环日志记录

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分配所有空间以进行日志记录；即，不会分配更多空间，并且对于任何日志文件，您都不会接收到“磁盘空间不足”错误。

无限日志记录

如果已对数据库启用无限日志记录功能（即，将 **logsecond** 配置参数设为 -1），那么必须将 **logarchmeth1** 配置参数设为除 OFF 或 logretain 以外的值才能启用归档日志记录功能。数据库管理器将在日志路径中至少保留 **logprimary** 配置参数所指定数目的活动日志文件，因此，不得对以上公式中 **logsecond** 配置参数使用值 -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。

镜像日志路径

如果正在镜像日志路径，那么必须将估计的日志文件空间需求增大一倍。

当前已落实

如果查询返回数据的当前已落实值，并且 **cur_commit** 配置参数未设为 DISABLED，那么需要更多的日志空间才能记录事务运行期间对数据行进行的第一次更新。根据工作负载大小的不同，所使用的日志空间总量可能会有很大的变化。此方案影响任何工作负载所需的日志 I/O、所需的活动日志空间量以及所需的日志归档空间量。

注: 如果将 **cur_commit** 配置参数设为 DISABLED，那么将保持前发行版的行为，并且所需的日志空间量没有变化。

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。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

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。除非您另行指定，否则创建的所有数据库都将具有缺省存储器组 IBMSTOGROUP。自动存储器管理的表空间对其存储器定义使用存储器组。

开始之前

DB2 数据库管理器必须正在运行。使用 **db2start** 命令来启动数据库管理器。

在创建数据库之前，规划数据库至关重要，您务必牢记数据库的内容、布局、潜在增长和使用方式。创建数据库并填充数据之后，可以进行更改。

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，如果有 **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。

限制

- 不能使用相对路径名来指定存储器路径；必须使用绝对路径名。存储器路径的长度可达 175 个字符。
- 在 Windows 操作系统上，除非 **DB2_CREATE_DB_ON_PATH** 注册表变量设为 YES，否则数据库路径必须只是一个盘符。
- 如果未使用 **CREATE DATABASE** 命令的 **DBPATH ON** 子句来指定数据库路径，那么数据库管理器将使用您对 **ON** 子句指定的第一个存储器路径作为数据库路径。（在 Windows 操作系统上，如果指定了此子句作为路径，并且 **DB2_CREATE_DB_ON_PATH** 注册表变量未设为 YES，那么您将接收到 SQL1052N 错误消息。）如果未指定 **ON** 子句，

那么将在数据库管理器配置文件（**dftdbpath** 参数）中指定的缺省数据库路径中创建数据库。此路径还将用作与该数据库相关联的单一存储器路径的位置。

- 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器路径（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** 子句显式指定的，还是通过在第一个存储器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。
- 存储器组必须至少有一个关联存储器路径。

注: 尽管可以通过指定 **AUTOMATIC STORAGE NO** 子句创建数据库，但建议不要使用 **AUTOMATIC STORAGE** 子句，且将来发行版可能会除去该子句。

关于此任务

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 person1
    WITH "Personnel DB for BSchiefer Co"
  ```

- 要使用 IBM Data Studio 创建数据库，请右键单击要在其上创建数据库的实例，然后选择任务助手以创建该实例。

示例

示例 1: 在 UNIX 或 Linux 操作系统上创建数据库:

要在使用 /DATA1 和 /DATA2 作为定义至缺省存储器组 IBMSTOGROUP 的存储器路径的情况下在路径 /DPATH1 中创建名为 TESTDB1 的数据库，请使用以下命令:

```
CREATE DATABASE TESTDB1 ON '/DATA1','/DATA2' DBPATH ON '/DPATH1'
```

示例 2: 在 Windows 操作系统上创建数据库，指定存储器路径和数据库路径:

要在驱动器 D: 上创建名为 TESTDB2 的数据库，并且存储器在 E:\DATA 中，请使用以下命令:

```
CREATE DATABASE TESTDB2 ON 'E:\DATA' DBPATH ON 'D:'
```

在此示例中，E:\DATA 用作定义至缺省存储器组 IBMSTOGROUP 的存储器路径和数据库路径。

示例 3: 在 Windows 操作系统上创建数据库，仅指定存储器路径:

要在驱动器 F: 上创建带有存储器的名为 TESTDB3 的数据库，请使用以下命令:

```
CREATE DATABASE TESTDB3 ON 'F:'
```

在此示例中，F: 用作定义至缺省存储器组 IBMSTOGROUP 的存储器路径和数据库路径。

如果指定目录名（例如 F:\DATA）作为存储器路径，那么此命令将失败，这是因为:

1. 如果未指定 **DBPATH**，那么此存储器路径（在这种情况下为 F:\DATA）用作数据库路径
2. 在 Windows 上，数据库路径只能是盘符（除非将 **DB2_CREATE_DB_ON_PATH** 注册表变量由缺省值 NO 更改为 YES）。

在 Windows 操作系统上，如果要指定一个目录作为存储器路径，那么还必须包括 **DBPATH ON** drive 子句，如示例 2 中所示。

示例 4: 在 UNIX 或 Linux 操作系统上创建数据库，并且不指定数据库路径:

要创建名为 TESTDB4 并且在 /DATA1 和 /DATA2 上有存储器的数据库，请使用以下命令:

```
CREATE DATABASE TESTDB4 ON '/DATA1','/DATA2'
```

在此示例中，将 /DATA1 和 /DATA2 用作定义至缺省存储器组 IBMSTOGROUP 的存储器路径，并且 /DATA1 是数据库路径。

下一步做什么

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。

可以使用下列任一方法来覆盖此缺省行为，以便不会自动调用配置顾问程序:

- 发出带有 **AUTOCONFIGURE APPLY NONE** 参数的 **CREATE DATABASE** 命令。
- 将 **DB2_ENABLE_AUTOCONFIG_DEFAULT** 注册表变量设为 NO:

  ```
  db2set DB2_ENABLE_AUTOCONFIG_DEFAULT=NO
  ```

  但是，如果您对 **CREATE DATABASE** 命令指定 **AUTOCONFIGURE** 参数，那么将忽略此注册表变量的设置。

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相关的额外处理时间和资源。如果不需要详细死锁事件监视器，那么可使用以下命令来删除此事件监视器:

```
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停用事件监视器，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。

要覆该数据库的缺省代码页，需要在创建数据库时指定需要的代码集和地域。有关设置代码集和地域的信息，请参阅 **CREATE DATABASE** 命令或 sqlecrea API。

将非自动存储器数据库转换为使用自动存储器

不能通过使用 CREATE STOGROUP 语句在数据库内定义缺省存储器组将现有非自动存储器数据库转换为使用自动存储器。

开始之前

必须存在可以通过路径（对于 Windows 操作系统: 路径或盘符）进行标识的存储器位置可用作自动存储器表空间的存储器路径。

限制

- 创建存储器组后，不能删除数据库的所有存储器组。
- 只能将 DMS 表空间转换为使用自动存储器。

注: 尽管可在创建数据库时指定 AUTOMATIC STORAGE NO 子句，但建议不要使用 AUTOMATIC STORAGE 子句，将来发行版可能会除去该子句。

关于此任务

通过指定 **CREATE DATABASE** 命令的 AUTOMATIC STORAGE NO 子句创建的数据库没有关联存储器组。而是，存储器与数据库的表空间相关联。为数据库定义存储器组时，现有表空间不会自动转换为使用自动存储器。缺省情况下，只有您创建的将来表空间才是自动存储器表空间。必须使用 ALTER TABLESPACE 语句才能将现有表空间转换为使用自动存储器。

过程

可通过使用 CREATE STOGROUP 语句在现有数据库中创建存储器组将该数据库转换为自动存储器数据库。

要在数据库内创建存储器组，请使用以下语句:

```
CREATE STOGROUP sg ON storagePath
```

其中 *sg* 是存储器组，*storagePath* 是要用于自动存储器表空间的路径。

示例

示例 1: 在 UNIX 或 Linux 操作系统上转换数据库

假定数据库 EMPLOYEE 是非自动存储器数据库，并且 /data1/as 和 /data2/as 是要用于自动存储器表空间的路径。要将 EMPLOYEE 转换为自动存储器数据库，请创建将 /data1/as 和 /data2/as 作为路径的存储器组:

```
CREATE STOGROUP sg ON '/data1/as', '/data2/as'
```

示例 2: 在 Windows 操作系统上转换数据库

假定数据库 SALES 是非自动存储器数据库，并且 F:\DB2DATA 和 G: 是要用于自动存储器表空间的路径。要将 SALES 转换为自动存储器数据库，请创建将 F:\DB2DATA 和 G: 作为路径的存储器组:

```
CREATE STOGROUP sg ON 'F:\DB2DATA', 'G:'
```

下一步做什么

如果要将现有的 DMS 表空间转换为使用自动存储器，请使用带有 MANAGED BY AUTOMATIC STORAGE 子句的 ALTER TABLESPACE 语句对其进行更改。如果未指定 USING STOGROUP 子句，那么该表空间使用指定缺省存储器组中的存储器路径。

创建存储器组后，可使用 CREATE TABLESPACE 语句来创建用于存储表、索引和其他数据库对象的自动存储器表空间。

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，也可以重新定义与存储器组相关联的存储器路径。通过将 **TO**、**ON** 和 **DBPATH ON** 的组合与 **RESTORE DATABASE** 命令配合使用或使用 **SET STOGROUP PATHS** 命令来设置数据库路径和存储器路径。

例如，以下是一些有效 **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（而不仅仅是存储器路径）。
  - 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。
- 每个*存储器组*的*存储器路径*（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器路径，并且将使用这些路径而不是存储在备份映像里的路径。如果数据库包含多个存储器组，那么每个已定义存储器组使用新存储器组路径。
– 如果使用了 **SET STOGROUP PATHS** 命令，那么提供的存储器路径用于指定的存储器组而不是备份映像中存储的存储器组。
– 如果未指定 **ON** 并且未使用 **SET STOGROUP PATHS** 命令，那么不会对存储器路径作出更改（会保留存储在备份映像中的存储器路径）。

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器路径:

表 10. 与数据库路径和存储器路径有关的复原含义

	磁盘上不存在同名的数据库		磁盘上存在同名的数据库	
RESTORE DATABASE 命令	**数据库路径**	**存储器路径**	**数据库路径**	**存储器路径**
RESTORE DATABASE TEST1	**dftdbpath**	使用在备份映像中定义的存储器路径	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器路径
RESTORE DATABASE TEST2 TO X:	X:	使用在备份映像中定义的存储器路径	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器路径
RESTORE DATABASE TEST3 DBPATH ON /db2/databases	/db2/databases	使用在备份映像中定义的存储器路径	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器路径
RESTORE DATABASE TEST4 ON /path1, /path2, /path3	/path1	所有存储器组使用 /path1、/path2 和 /path3 来表示它们的存储器路径	使用现有数据库的数据库路径	所有存储器组使用 /path1、/path2 和 /path3 来表示它们的存储器路径
RESTORE DATABASE TEST5 ON E:\newpath1, F:\newpath2 DBPATH ON D:	D:	所有存储器组使用 E:\newpath1 和 F:\newpath2 来表示它们的存储器路径	使用现有数据库的数据库路径	所有存储器组使用 E:\newpath1 和 F:\newpath2 来表示它们的存储器路径

对于已将存储器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** 选项来显示对备份映像中的数据库是否存在存储器组。将显示存储器组及它们的存储器路径。

对于多分区数据库，**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器路径。
2. 仅可在目录数据库分区上使用新的存储器路径发出 **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为 RESTORE_PENDING。

表 11. 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	存储器路径	其他数据库分区上的结果	存储器路径
RESTORE DATABASE TEST1	目录数据库分区	在目录数据库分区上使用存储器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器路径	无。尚未更改存储器路径，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器路径
	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。尚未更改存储器路径，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器路径
RESTORE DATABASE TEST2 ON /path1, /path2, /path3	目录数据库分区	使用在 **RESTORE** 命令中指定的存储器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。	所有存储器组使用 /path1、/path2 和 /path3 来表示它们的存储器路径		所有存储器组使用 /path1、/path2 和 /path3 来表示它们的存储器路径
	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** 中指定存储器路径。	无	返回 SQL1172N。不能在非目录数据库分区的 **RESTORE** 中指定新的存储器路径。	无

编目数据库

创建数据库时，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。

关于此任务

可使用 **CATALOG DATABASE** 命令来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。启用目录高速缓存之后，另一个应用程序对目录所作的更改（例如，使用 **CATALOG DATABASE** 或 **UNCATALOG DATABASE** 命令进行的更改）可能要到您的应用程序重新启动之后才会生效。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。

过程

- 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。

关于此任务

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。

过程

要将实用程序绑定或重新绑定至数据库，请调用下列命令:

```
connect to sample
   bind @db2ubind.lst
```

其中 *sample* 是数据库的名称。

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*并行事务*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*的其中一个:

- 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。

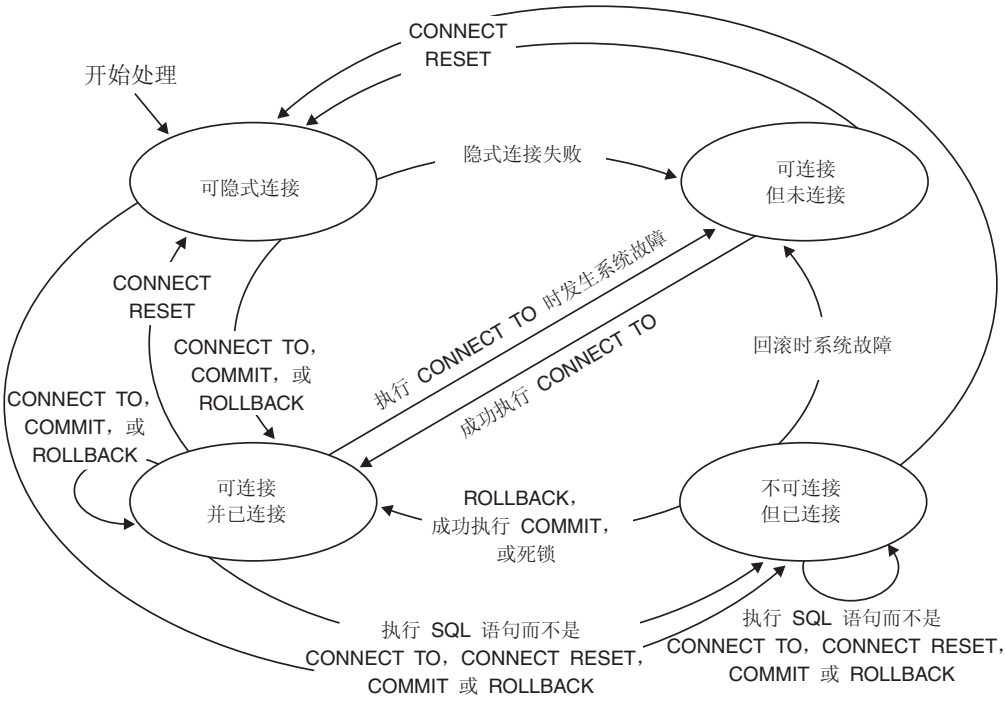


图 4. 隐式连接可用时的连接状态过渡

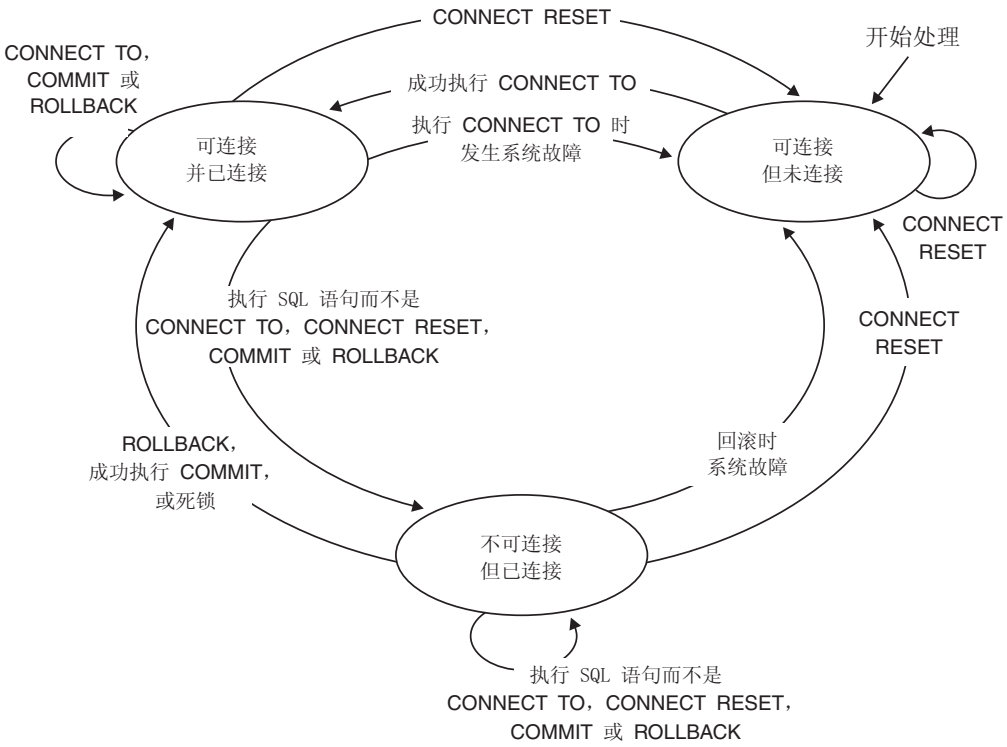


图 5. 隐式连接不可用时的连接状态过渡

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。

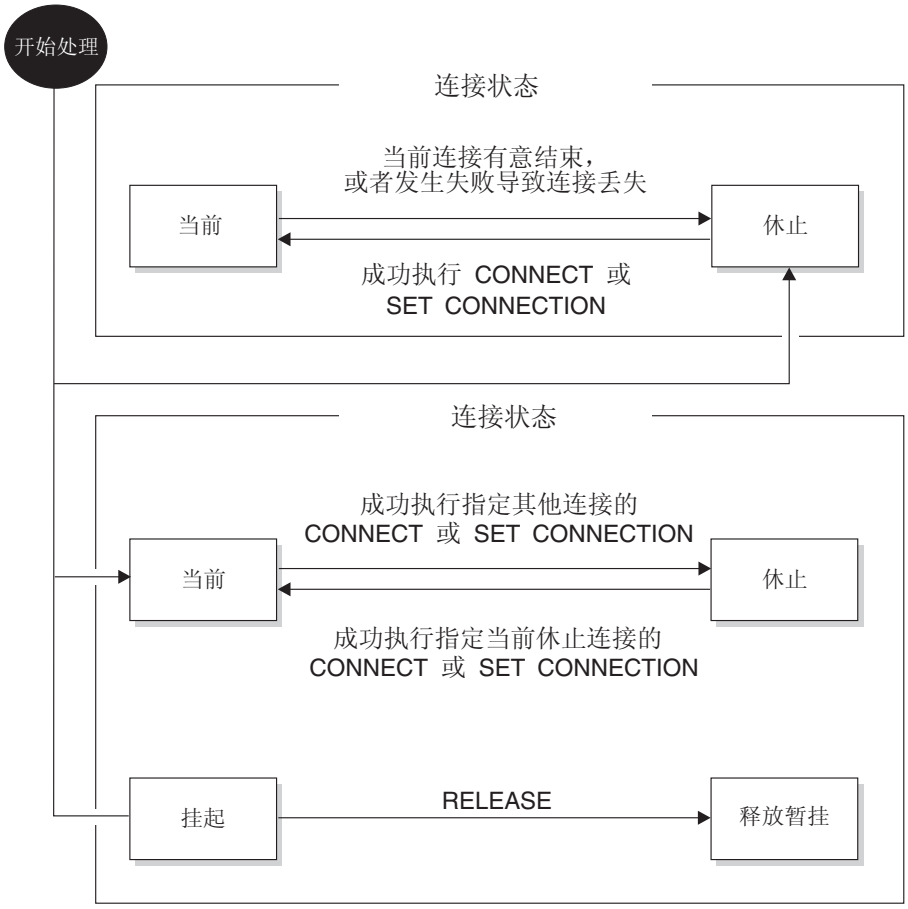


图 6. 应用程序导向的分布式工作单元状态过渡

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 116 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 预编译程序选项设为 AUTOMATIC，那么所有连接都将结束。如果它设为 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。

当前状态和休止状态。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。

使用连接过程定制应用程序环境

连接过程为您提供了一种方法，允许环境中的应用程序通过连接来隐式执行特定过程。此过程允许您从中央控制点定制数据库的应用程序环境。

例如，在连接过程中，可以通过调用 SET CURRENT PATH 语句将诸如 CURRENT_PATH 的专用寄存器设为非缺省值。现在，此新的 CURRENT_PATH 值将是所有应用程序的有效缺省 CURRENT_PATH。

可以将数据库中所创建的任何符合命名限制和参数限制的过程用作该数据库的连接过程。您按照在同一数据库中创建过程的形式提供了定制逻辑，并且允许执行过程的任何常见操作（例如，发出 SQL 语句）。

数据库配置参数 **connect_proc** 指定要用于数据库的所有连接的连接过程。更新 **connect_proc** 参数以设置连接过程的名称并启用连接过程。需要建立数据库连接才能更新 **connect_proc** 参数的长度不为零的值。设置 **connect_proc** 参数后，任何新连接的会话授权标识必须具有对指定的连接过程的 EXECUTE 特权，无论是直接方式还是间接方式（通过与连接过程相关联的组、角色的其中一个或 PUBLIC）。

在成功连接处理结束时且处理同一连接上的任何后续请求之前，将在服务器上隐式执行连接过程。成功运行连接过程之后，数据库管理器会落实由连接过程进行的任何更改。如果连接过程失败，那么将回滚由连接过程进行的任何更改，且连接尝试失败并带有错误。

注：对连接过程中专用寄存器的任何更改都将反映在生成的会话中（即使在过程完成后）。

要点：连接过程返回的任何错误都将使所尝试的连接失败。连接过程的执行所返回的错误将返回到应用程序。如果您要修改连接过程并更正错误，那么在问题被修正之前，必须取消设置 **connect_proc** 参数以允许成功连接。

关于连接过程的建议

为了避免连接过程出现问题，请确保连接过程符合下列建议：

- 使连接过程的逻辑保持简单。

  对于每个连接，使用连接过程会因引入附加处理而影响 CONNECT 命令的性能。如果过程的效率不高或出现延误（如锁定争用），那么性能影响可能很明显。

  将过程作为连接过程来建立之前，务必对此过程进行充分测试。
- 避免访问将被删除或变更的连接过程中的对象。

  如果删除连接过程中的某个从属对象或者撤销对访问从属对象的特权，那么连接过程可能会失败。从过程返回的错误会根据过程的逻辑阻止与数据库建立新连接。
- 避免从连接过程中调用另一个过程。

与连接过程本身不同，连接过程所调用的过程可删除或改变。如果连接过程所调用的过程失效或者被删除，那么连接过程可能会失败。从连接过程返回的错误会根据过程的逻辑阻止与数据库建立新连接。此外还请注意，与在连接过程自身中更改的专用寄存器（这会在应用程序中生效）相反，从连接过程调用的过程中更改的专用寄存器不会更改调用环境的专用寄存器。

- 请避免在连接过程中指定 COMMIT ON RETURN 子句。

  内部落实将在隐式调用连接过程后进行处理。如果指定了 COMMIT ON RETURN YES 子句，那么数据库管理器将处理多个落实调用，而这会影响性能。指定 COMMIT ON RETURN NO 对连接过程处理没有任何影响。
- 在退出连接过程之前，请释放所有资源并关闭所有游标。

  应用程序无法通过连接过程访问仍然打开的任何资源（例如，WITH HOLD 游标）。仅当应用程序完成后，才可以释放在处理落实后由连接过程保持的资源。
- 将连接过程的 EXECUTE 特权授予 PUBLIC。

  连接过程并非用于控制数据库访问。访问控制是通过将数据库权限授予用户而进行的。
- 避免对不同数据库分区使用不同的 **connect_proc** 参数值。

  根据与用户相连的数据库分区，对不同的数据库分区使用不同的连接过程会产生不一致的应用程序行为。这还会使数据库环境更复杂并且难以管理。

连接过程的用法说明

连接过程具有下列限制和局限性:

- 在设置 **connect_proc** 参数的情况下，您不能创建与连接过程同名的过程。
- 只有参数个数正好为零的过程才能用作连接过程。只要设置了 **connect_proc** 参数，数据库中就不能存在任何其他具有由两部分组成的相同名称的过程。
- 连接过程名称（模式名称及过程名称）只能包含以下字符:
  - A-Z
  - a-z
  - _（下划线）
  - 0-9

  此外，模式名称和过程名称需要遵循普通标识符的规则。
- 在设置了 **connect_proc** 参数的情况下，您无法删除或改变连接过程。

  要改变或删除连接过程，请将 **connect_proc** 参数更改为空值或其他过程的名称。
- 连接过程不能使用由 sqleseti API 或 SQLSetConnectAttr CLI 函数设置的客户机信息字段。

  在连接过程运行前，这些字段的专用寄存器包含这些字段的缺省服务器值。当连接过程运行时，尚未更新通过调用 sqleseti API、SQLSetConnectAttr CLI 函数或 SQLSetEnvAttr CLI 函数（例如，CLIENT USERID、CLIENT ACCTNG、CLIENT APPLNAME 和 CLIENT WRKSTNNAME）而设置的客户机信息字段或 SQL 专用寄存器。

- 通过调用 sqleseti API、SQLSetConnectAttr CLI 函数或者 SQLSetEnvAttr CLI 函数以及 IBM Data Server Driver for JDBC and SQLJ 方法集 ClientAccountingInformation 所设置的客户机信息字段或 SQL 专用寄存器具有优先权，并且会覆盖在连接过程中设置的专用寄存器值。
- 从连接过程返回之后，将只保持设置直接由连接过程设置的专用寄存器。连接过程中嵌套的例程调用不会更改执行调用的环境中的专用寄存器的设置。

实现连接过程的示例

下列示例显示了一些连接过程样本以及如何在数据库中启用连接过程:

示例 1

1. 定义 SQL 过程 NEWTON.CONNECTPROC 以根据 SESSION_USER 设置专用寄存器。

```
CREATE PROCEDURE NEWTON.CONNECTPROC ( )
READS SQL DATA
LANGUAGE SQL
   BEGIN

   --set the special register based on session user id
      CASE SESSION_USER
        WHEN 'USERA' THEN
         SET CURRENT LOCALE LC_TIME 'fr_FR';

        WHEN 'USERB' THEN
         SET CURRENT LOCALE LC_TIME 'de_DE';
   ELSE
         SET CURRENT LOCALE LC_TIME 'au_AU';

   END CASE;

   END %
```

   此过程通过用户 USERA 和 USERB 的特例值建立 CURRENT LOCALE LC_TIME 专用寄存器的设置。

2. 将连接过程上的 EXECUTE 特权授予组 PUBLIC:

```
GRANT EXECUTE ON PROCEDURE NEWTON.CONNECTPROC TO PUBLIC
```

   .

3. 更新 **connect_proc** 参数以指出将为所有新连接调用这一新过程:

```
db2 update db cfg using connect_proc "NEWTON.CONNECTPROC"
```

   对于新连接的所有后续 CONNECT 请求，现在将自动调用 NEWTON.CONNECTPROC 连接过程。将根据 SESSION USER 设置专用寄存器 CURRENT LOCALE LC_TIME。

示例 2

1. 设置并调用新连接的过程以定制新连接的初始专用寄存器值。

```
CREATE PROCEDURE MYSCHEMA.CONNECTPROC
  (   )
  EXTERNAL NAME 'parts!connectproc'
  DBINFO
  READS SQL DATA
  LANGUAGE C     PARAMETER STYLE SQL
```

此过程读取自数据库表 MYSCHEMA.CONNECTDEFAULTS，用以根据与新连接的授权标识相关联的组来确定要在 CURRENT SCHEMA、CURRENT PATH 和 CURRENT QUERY OPTIMIZATION 专用寄存器中设置的值。此过程还根据同一表中的信息来设置全局变量 MYSCHEMA.SECURITY_SETTING 的值。

2. 将连接过程上的 EXECUTE 特权授予组 PUBLIC:

```
GRANT EXECUTE ON PROCEDURE MYSCHEMA.CONNECTPROC TO PUBLIC
```

.

3. 更新 **connect_proc** 参数以指出将为所有新连接调用这一新过程:

```
db2 update db cfg using connect_proc "MYSCHEMA.CONNECTPROC"
```

对于新连接的所有后续 CONNECT 请求，现在将自动调用 MYSCHEMA.CONNECTPROC 连接过程。

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。以下列表中概述了这些选项，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）。

要在运行时覆盖上面列示的任何选项，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。

开始之前

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。

过程

- 要查看本地数据库目录文件的内容，请发出以下命令:

  ```
  LIST DATABASE DIRECTORY ON location
  ```

  其中 *location* 指定数据库的位置。
- 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）。

过程

- 要使用命令行删除数据库，请输入：`DROP DATABASE name`
- 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。
- 要使用 IBM Data Studio 删除数据库，请右键单击数据库，然后选择任务助手来删除该数据库。

示例

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。

删除别名

删除别名时，将从目录中删除其描述，引用该别名的任何程序包和高速缓存动态查询将失效。依赖于该别名的所有视图和触发器将标记为不可操作。

过程

要删除别名，请在命令行中发出以下 DROP 语句:

```
DROP ALIAS employee-alias
```

第 6 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。

第 7 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。

在 DB2 pureScale 环境中，每个成员都有自己的本地缓冲池 (LBP)。但是，有一个附加组缓冲池 (GBP)，此缓冲池由集群高速缓存设施维护。GBP 由所有成员共享。它被用作 DB2 pureScale实例中的个别成员使用的页的高速缓存，以改进性能并确保一致性。

缓冲池的使用方法

注: 下面的信息讨论 DB2 pureScale 环境以外的环境中的缓冲池。缓冲池在 DB2 pureScale 环境中以不同方式工作。有关更多信息，请参阅 *数据库监视指南和参考* 中的“在 DB2 pureScale 环境中监视缓冲池”。

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。

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，还是在下一次停用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？请参阅第 122 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『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”。

表空间与缓冲池之间的关系

设计缓冲池时，您必须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更改页大小了。

缓冲池内存注意事项

内存需求

设计缓冲池时，还应根据计算机上已安装的内存量以及与数据库管理器在同一计算机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 V9.5，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 125 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，必须升级到 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。

DB2 pureScale 环境中的缓冲池

在 DB2 pureScale 环境中，集群高速缓存设施提供了由所有成员共享的公共组缓冲池 (GBP)。每个成员还可管理它自己的一组本地缓冲池 (LBP)。

GBP 是支持所有 DB2 页大小并且所有成员都可使用的单个缓冲池。成员会将页高速缓存在它们自己的 LBP 中，并使用 GBP 来维护成员之间的页一致性。每个成员上可存在不同页大小的 LBP（例如，4K、8K、16K 或 32K）。

GBP 存储两种类型的信息、目录条目和数据元素。目录条目存储有关缓冲池页的元数据信息，数据元素存储页数据。目录条目与数据元素之比由 DB2 for Linux, UNIX, and Windows 自动调整。GBP 内存大小由 **cf_gbp_sz** 配置参数定义。

DB2 缓冲池服务

因为每个成员上有 LBP 以及并且存在由所有成员共享的 GBP，所以同一页的多个副本可存在于多个缓冲池中。用于访问页、更改页以及将更改传播至其他成员的全局并行性和相干性控制由 DB2 缓冲池服务处理。此服务还会处理缓冲池中的 I/O 数据，包括将 GBP 中的页写至磁盘。

GBP 依赖性

需要由 DB2 pureScale 环境中的不同成员访问的缓冲池页依赖于 GBP。对于这些相关页，GBP 会在不同缓冲池中协调此页的副本。只有一个成员可访问的页不是依赖于 GBP 的页，并且仅存在于成员 LBP 中。在 DB2 pureScale 环境中，不同成员未共享临时表空间，所以此临时表空间的所有缓冲池页并非依赖于 GBP。

P-lock 控制 DB2 pureScale 环境中的缓冲池页的访问以更新和读取某个页版本。与由特别事务所有的逻辑锁定（例如行锁定或表锁定）不同，用于控制缓冲池页访问的 P-lock 由集群的成员所有。系统使用了以下 P-lock:

- 要更新页，成员必须以 X 方式持有 P-lock。
- 要读取页的最新版本，成员必须以 S 方式持有 P-lock。

要读取此页的一致版本（但不一定是最新版本），不需要任何 P-lock。DB2 for Linux, UNIX, and Windows 在内部决定访问页时使用哪种类型的读取。

GBP 相干性

缓冲池页依赖于 GBP 时，它可能存在于磁盘上的 GBP、多个成员上的 LBP 或两者的组合中。以下协议规则协调多个页副本的相干性:

- 成员将某页访存到其 LBP 中时，它会向 GBP 注册该页。
- 尝试将某页访存到 LBP 中时，成员会先检查 GBP，然后仅当 GBP 中不存在此页时才从磁盘读取此页的内容（GBP 中存在的此页的版本决不会比磁盘上此页的版本旧）。
- 成员具有针对某页的更新锁定（X 方式下的 P-lock）时，成员的 LBP 中的该页的版本可能比 GBP 中该页的版本新。
- 在成员释放更新锁定（或降低 P-lock 级别）之前，系统会使用该页的较新版本更新 GBP。
- 修改某页的事务结束（通过落实或回滚）时，已修改页会写至 GBP。
- 如果另一成员请求 P-lock 以读取某页的最新版本或更新该页，那么还可在事务结束之前通过页面协商将该页写至 GBP。另一 成员 请求 P-lock 时，此锁定冲突会导致拥有锁定的 成员 将已修改页写至 GBP 以便它可释放 P-lock 或使其降级，之后发出请求的 成员 可请求此锁定。

GBP 控制

要由 GBP 使用的总内存量由 **cf_gbp_sz** 数据库配置参数控制。第一次在成员上激活数据库时，如果 CF 上还没有 GBP，那么会分配 GBP。CF 停止时、数据库在整个集群中被删除或一致关闭时或在数据库恢复操作期间，此 GBP 被释放。

释放操作会将 GBP 中的页写至磁盘并在成员之间进行协调。释放操作类似于在 LBP 中清除页并实现两个功能:

- 将脏页写至磁盘以确保有足够的干净目录条目和数据元素可用于新页注册和写入。
- 通过确保 GBP 中没有页的保留时间超过指定时间来维护特定恢复窗口。这样做可以减少在恢复时要重演的日志记录数。

必要时，可使用 **softmax** 数据库配置参数来控制释放行为。在 DB2 pureScale 环境中，此参数确定每个工作阶段必须从 GBP 释放至磁盘的页数。

LBP 控制

本地缓冲池配置通过 DDL 语句控制。通过其发出 DDL 语句的成员充当协调程序，负责将该语句分发至集群中的其他成员以便本地执行及协调整体执行。在 DB2 pureScale 实例中执行 LBP DDL 语句的行为与在非 DB2 pureScale 环境中执行 LBP DDL 语句的行为有很大差异。在 DB2 pureScale 环境中，并非用其定义 LBP 的所有成员（协调程序除外）都必须处于可用状态或者已激活数据库才能使 DDL 语句成功执行。对于当前不可用（例如，因为已安排维护）或未激活数据库的成员，它们不会处理 DDL 语句。DDL 继续按正常方式执行。落实事务时，协调程序将在磁盘上的缓冲池文件中对所有适用成员（包括未处理该语句的对象）更新缓冲池更改。这些成员下次激活数据库时会应用已落实缓冲池更改。但是，如果任何活动成员因为内存不足的情况或其他错误而未能运行 DDL 语句，那么此语句会回滚，并且会向用户或客户机应用程序返回错误。

缓冲池监视元素

可复查若干特定于 GBP 和 LBP 的监视元素以监视 DB2 pureScale Feature 的整体性能。有一些特定于数据库配置参数 **cf_gbp_sz** 的监视元素。有关查看集群高速缓存设施的内存使用情况级别的更多信息，请参阅“MON_GET_CF 表函数 - 获取集群高速缓存设施度量值”主题。

还有一些监视元素用于跟踪 GBP 和 LBP 的物理页读取数、逻辑页读取数以及无效页数。有关 DB2 pureScale Feature 的监视元素的列表，请参阅“新的和已更改的监视元素”。

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。

*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；仅当任何特定代理程序需要访问时，该代理程序才具有对缓冲池页的访问权。

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，就会撤销访问权。此行为确保只有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。

要启用缓冲池内存保护功能，以便提高数据库引擎的弹性，请启用 **DB2_MEMORY_PROTECT** 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。如果启用了 **DB2_MEMORY_PROTECT**（设为 YES），并且 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。

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 的更多信息，请参阅 http://publib.boulder.ibm.com/infocenter/systems/scope/aix/index.jsp?topic=/com.ibm.aix.genprogc/doc/genprogc/storage_protect_keys.htm。

如果 **DB2_LGPAGE_BP** 设为 YES，那么不能使用内存保护。即使 **DB2_MEMORY_PROTECT** 设为 YES，DB2 数据库管理器也将无法保护缓冲池内存和禁用此功能。

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。

开始之前

计算机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。

关于此任务

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 子句将在数据库中的所有数据库分区上创建缓冲池。

过程

要使用命令行创建缓冲池，请执行以下操作:

1. 获取数据库中存在的缓冲池名称列表。 发出以下 SQL 语句:

   ```
   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。 基本 CREATE BUFFERPOOL 语句为:

   ```
   CREATE BUFFERPOOL buffer-pool-name
     PAGESIZE 4096
   ```

结果

如果有足够内存可用，那么缓冲池可立即变为活动状态。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。如果数据库管理器无法分配更多内存，那么数据库管理器会返回警告情况，指出未能启动缓冲池。此警告将在后续数据库启动时提供。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。

示例

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。

开始之前

语句的授权标识必须具有 SYSCTRL 或 SYSADM 权限。

关于此任务

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。

要使用命令行来更改缓冲池，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. 确定必须进行的更改。
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 语句来更改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。

计算机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。

关于此任务

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。

过程

要删除缓冲池，请使用 DROP BUFFERPOOL 语句。

```
DROP BUFFERPOOL buffer-pool-name
```

第 8 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。它们用来将数据库中的数据组织成与数据在系统上的存储位置相关的逻辑存储器分组。表空间存储在数据库分区组中。

使用表空间来组织存储器有多个好处:

可恢复性

由于可以通过单一命令来备份或复原表空间中的所有对象，因此，通过将必须一起备份和复原的对象放入同一个表空间，可以使备份和复原操作更为方便。如果分区表和索引分布在多个表空间中，那么可以只备份或复原驻留在给定表空间中的数据分区和索引分区。

更多的表

可以在任何一个表空间中存储的表数是有限的；如果需要的表数超出表空间可以包含的表数，那么只需要为它们创建附加的表空间。

自动存储器管理

借助自动存储器表空间，可以自动管理存储器。数据库管理器根据需要创建并扩展容器。

能够在缓冲池中隔离数据，以便提高性能或内存利用率

如果您有一组需要频繁查询的对象（例如表或索引），那么可以使用单一 CRETAE 或 ALTER TABLESPACE 语句为这些对象所驻留在的表空间分配缓冲池。可以为临时表空间分配它们自己的缓冲池，以便提高排序或连接之类的活动的性能。在某些情况下，最好为很少访问的数据或者需要非常随机地访问非常大的表的应用程序定义较小的缓冲池；在这种情况下，不需要使数据保存在缓冲池中的时间超出单个查询的运行时间。

表空间可以由一个或多个*容器*组成。容器可以是目录名、设备名或文件名。单个表空间可以有多个容器。可以在同一个物理存储设备上创建多个容器（来自一个或多个表空间），尽管您创建的各个容器使用不同存储设备时的性能最佳。如果您使用自动存储器表空间，那么容器的创建和管理工作由数据库管理器自动处理。如果未使用自动存储器表空间，那么必须自己定义和管理容器。

第 130 页的图 7 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。

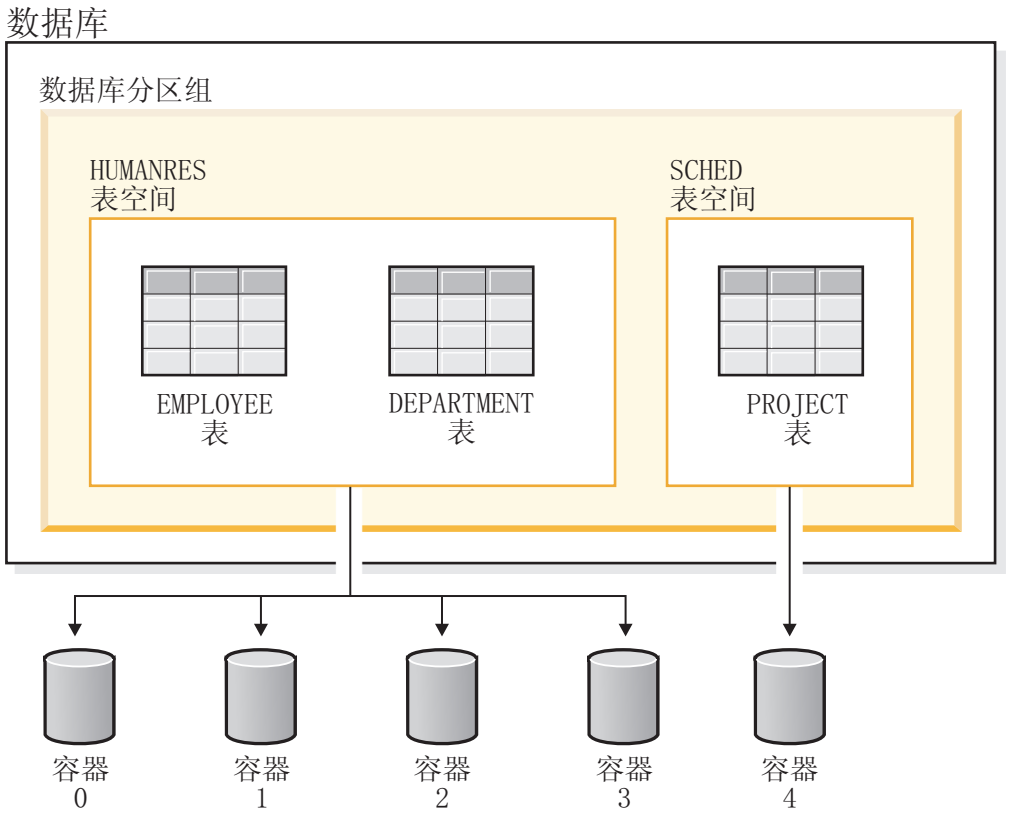


图 7. 数据库中的表空间和表

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。

第 131 页的图 8 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。

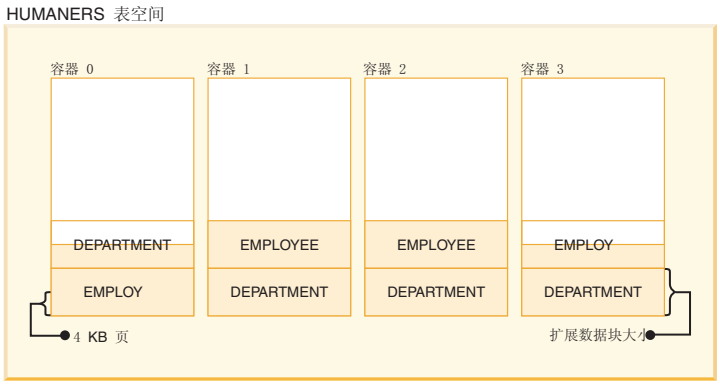


图 8. 表空间中的容器和扩展数据块

系统数据、用户数据和临时数据的表空间

每个数据库都必须要有最小的一组表空间，这些表空间用于存储系统数据、用户数据和临时数据。

一个数据库至少必须包含三个表空间:

- *目录表空间*
- 一个或多个*用户表空间*
- 一个或多个*临时表空间*

目录表空间

目录表空间包含数据库的所有系统目录表。此表空间称为 SYSCATSPACE，它不能被删除。

用户表空间

用户表空间包含用户定义的表。缺省情况下，将创建一个用户表空间 USERSPACE1。

如果创建表时未对其指定表空间，那么数据库管理器将自动进行选择。有关更多信息，请参阅 CREATE TABLE 语句的 IN *tablespace-name* 子句的文档。

表空间的页大小确定了表中的最大行长度或列数。CREATE TABLE 语句的文档说明了页大小与最大行大小和列数之间的关系。在 V9.1 之前，缺省页大小为 4 KB。在 V9.1 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。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）。

临时表空间

临时表空间包含临时表。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。

处理查询时，数据库管理器可能需要访问页大小足以处理与查询相关的数据的系统临时表空间。例如，如果查询返回的数据包含长度为 8KB 的行，并且没有页大小至少为 8KB 的系统临时表空间，那么该查询将失败。您可能需要创建具有更大页大小的系统临时表空间。通过定义页大小等于用户表空间的最大页大小的临时表空间，可以帮助您避免这些类型的问题。

用户临时表空间存放使用 DECLARE GLOBAL TEMPORARY TABLE 或 CREATE GLOBAL TEMPORARY TABLE 语句创建的表的临时数据。缺省情况下，创建数据库时不会创建这些表空间。它们还存放已创建临时表的实例化版本。为了能够定义已声明临时表或已创建临时表，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。

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。如果存在多个具有该页大小的临时表空间，那么将以循环方式来选择表空间，即，先选择具有该页大小的表空间，然后为将要分配的下一个对象选择下一个表空间并依此类推，直到用尽所有合适的表空间后回到第一个表空间。但是，在大多数情况下，建议您不要创建多个具有相同页大小的临时表空间。

分区数据库环境中的表空间

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。

将表空间分配给数据库分区组时，该数据库分区组必须已存在。表空间与数据库分区组之间的关联由您使用 CREATE TABLESPACE 语句创建表空间时定义。

您无法更改表空间与数据库分区组之间的关联。您只能使用 ALTER TABLESPACE 语句来更改数据库分区组中各个数据库分区的表空间规范。

在单分区环境中，每个表空间都与缺省数据库分区组相关联，如下所示:

- 目录表空间 SYSCATSPACE 与 IBMCATGROUP 相关联
- 用户表空间与 IBMDEFAULTGROUP 相关联
- 临时表空间与 IBMTEMPGROUP 相关联

在分区数据库环境中，IBMCATGROUP 分区将包含全部这三个缺省表空间，而其他每个数据库分区将只包含 TEMPSPACE1 和 USERSPACE1。

DB2 pureScale Feature的表空间注意事项

DB2 pureScale 环境需要特定类型的表空间。

表空间类型支持

DB2 pureScale 环境仅支持使用自动存储器类型的表空间。从 DB2 pureScale 以外的环境迁移至 DB2 pureScale 环境之前，不得有系统管理的空间 (SMS) 表空间、常规数据

库管理的空间 (DMS) 表空间或自动存储器混合表空间。如果对 DB2 pureScale 环境中的非自动存储器类型发出 CREATE TABLESPACE 命令，那么会返回错误 (SQL1419N)。

临时表空间支持

在 DB2 pureScale配置中，临时表空间的存储器路径是在集群文件系统内定义的。这些临时表空间由每个成员在本地管理。此容器路径由所有成员共享，并且该目录中的表空间文件对每个成员都是唯一的。成员标识编号与每个文件路径名相关联。

瞬态表空间状态

成员失败后，在其他成员的正常运行时活动期间，可在 DB2 pureScale 环境中以并行方式进行崩溃恢复。这些瞬态状态必须保留设为阻止其他不兼容操作启动，直到进行了该失败成员的成员崩溃恢复 (MCR)。有两种类型的瞬态表空间状态。第一种类型可在成员失败后恢复启动前的任何时间清除。因此，如果失败成员设置了 SQLB_BACKUP_IN_PROGRESS、SQLB_DISABLE_PENDING 或 SQLB_MOVE_IN_PROGRESS，那么它们将保留设置，直到出现下列其中一种情况:

1. 此数据库的 MCR 已启动，或者
2. 另一成员需要将该表空间装入到它的内存高速缓存。

另一类型的瞬态表空间状态需要保留设置直到 MCR 结束，以保护发生故障时运行的操作（例如，表重组和某些装入命令）。

因此，如果失败成员设置了 SQLB_REORG_IN_PROGRESS 或 SQLB_LOAD_IN_PROGRESS，那么它们将保留设置直到 MCR 成功完成。

发出表空间 DDL 请求

在分区数据库环境中，如果数据库分区的主机或 DB2 实例关闭，那么表空间 DDL SQL 事务将失败，并且语句将回滚。如果主机或 DB2 实例已启动但数据库基础结构未在该处激活，那么 DDL 请求将隐式激活数据库以处理该请求。这些常规行为在 DB2 pureScale 环境中会有所不同，如下所示:

1. **CREATE TABLESPACE**、**ALTER TABLESPACE** 和 **DROP TABLESPACE** 操作不需要所有成员启动就能成功完成。
2. 如果数据库基础结构未在成员上激活，那么表空间 DDL 请求将不执行隐式激活。

表空间和存储器管理

可以通过不同的方式来设置表空间，这取决于您希望它们以何种方式使用可用的存储器。可以让操作系统管理空间分配，也可以让数据库管理器根据您指定的参数为数据分配空间。另外，您也可以创建自动分配存储器的表空间。

三种类型的表空间分别被称为:

- 系统管理的空间 (SMS)，在此空间中，操作系统的文件管理器在您定义用于存储数据库文件的位置后控制存储器空间。
- 数据库管理的空间 (DMS)，在此空间中，数据库管理器在您分配存储器容器后控制对存储器空间的使用。
- 自动存储器表空间，在此空间中，数据库管理器根据需要创建容器。

在数据库中，这三种空间可以按任意组合配合使用。

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。根据需要分配存储空间。

SMS 存储模型由表示数据库对象的文件组成；例如，每个表都至少有一个与其相关联的物理文件。当您设置表空间时，您通过创建容器确定文件的位置。SMS 表空间中的每个容器都与绝对或相对目录名相关联。其中每一个目录都可以位于不同的物理存储设备或文件系统上。为每个容器中的对象创建的文件名由数据库管理器控制，文件系统负责对其进行管理。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。

要点: 在适用于用户定义的永久表空间的 V10.1 中已经不推荐使用 SMS 表空间类型，在以后的发行版中可能会将其除去。没有不推荐使用适用于目录和临时表空间的 SMS 表空间类型。有关更多信息，请参阅 《*DB2 V10.1 新增内容*》中的『不推荐使用 SMS 永久表空间』

分配空间的方式

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为 YES（缺省值），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。

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，用于长数据（LONG VARCHAR 和 LONG VAR GRAPHIC）和大对象 (LOB) 的文件并不是每次扩展一个扩展数据块。

注: 多页文件分配功能不适用于使用系统管理的空间的临时表空间。

在 SMS 表空间中，当单个容器中的所有空间都被耗用时，就认为该表空间“已满”，即使其他容器中还有剩余空间亦如此。与 DMS 表空间不同，在创建 SMS 表空间之后，不能对其添加容器。要为 SMS 容器提供更多空间，请对底层文件系统添加更多空间。

规划 SMS 表空间

考虑使用 SMS 表空间时，必须考虑两个因素:

- **该表空间所需的容器数。**创建 SMS 表空间时，必须指定要让该表空间使用的容器数。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。分区数据库环境是此规则的例外；对 SMS 表空间的数据库分区组添加新的数据库分区时，可以使用 ALTER TABLESPACE 语句将容器添加至新的数据库分区。

  表空间的最大大小可以按以下公式估算:

  $n \times maxFileSystemSize$

  其中，n 是容器数，*maxFileSystemSize* 表示操作系统所支持的最大文件系统大小。

  此公式假定每个容器都映射到不同的文件系统，假定每个文件系统都有最大量的可用空间，并假定各个文件系统的大小相同。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，这可能影响表空间的最大大小。

警告: 对 SMS 表空间指定的路径不能包含任何其他文件或目录。

- **表空间的扩展数据块大小。***扩展数据块大小*是指，数据库管理器在使用另一容器前写入容器的页数。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。如果未对 **CREATE DATABASE** 命令指定 **dft_extent_sz** 的值，那么缺省扩展数据块大小将设为 32。

容器和扩展数据块大小

要为表空间选择适当的容器数和扩展数据块大小，您必须了解:

- **操作系统对逻辑文件系统大小施加的限制**。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。**第一个表数据文件（按照惯例，这是 SQL00002.DAT）将在其中一个表空间容器中创建。数据库管理器将根据同时考虑容器总数和表标识的算法来确定该表空间容器。允许此文件增大到扩展数据块大小。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起始容器。此过程（称为*条带分割*）将继续在容器目录中进行，直到一个容器装满 (SQL0289N) 或操作系统中不再有空间可分配（磁盘已满错误）为止。条带分割适用于块映射文件 (SQL*nnnnn*.BKM)、索引对象以及用来存储表数据的其他对象。如果您选择实现磁盘条带分割以及数据库管理器提供的条带分割功能，那么表空间的扩展数据块大小与磁盘的分割大小应该相同。

  注: 只要任何一个容器已满，就认为 SMS 表空间已满。因此，每个容器具有相同容量的可用空间是很重要的。

  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。

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。与 SMS 表空间不同，系统将根据您创建 DMS 表空间时指定的容器定义在文件系统中预先分配存储器空间。

要点: 从 V10.1 FP1 开始，在用户定义的永久表空间中不推荐使用 DMS 表空间类型，在以后的发行版中可能会将其除去。对于目录和临时表空间，不推荐使用 DMS 表空间类型。有关更多信息，请参阅《*DB2 V10.1 新增内容*》中的『不推荐使用 DMS 永久表空间』。

DMS 存储模型由数目有限的文件或设备组成，这些文件或设备的空间由数据库管理器管理。您决定在创建容器时要使用哪些文件和设备，并且为那些文件和设备管理空间。

包含用户定义的表和数据的 DMS 表空间可以被定义为*大型*（缺省）或*常规*表空间，这些表空间可以存储任何表数据或索引数据。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是 64 TB。有关采用其他页大小的常规表空间的最大大小，请参阅 *SQL Reference*中的『SQL 和 XML 限制』。

使用 DMS 表空间时，容器有两个选项: 文件和原始设备。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，并总是确保*扩展数据块*中的页连续。（*扩展数据块*大小是指数据库管理器在使用另一容器前写入容器的页数。）

规划 DMS 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。
- 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，如果设备的存储容量相当于 5000 页，并且将设备容器大小定义为 3000 页，那么在该设备上，将有 2000 页不可用。
- 在缺省情况下，每个容器中都保留一个扩展数据块作为附加的必需空间。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。
  - 在表空间中，有三个扩展数据块保留给开销使用:
  - 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。）

  试图创建小于五个扩展数据块的表空间将产生错误 (SQL1422N)。
- 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。V9.5 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 162 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。
- 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。

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。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*空间映射页* (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图。位图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射页 (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。图 9 说明了如何在逻辑表空间地址映射中分配扩展数据块。

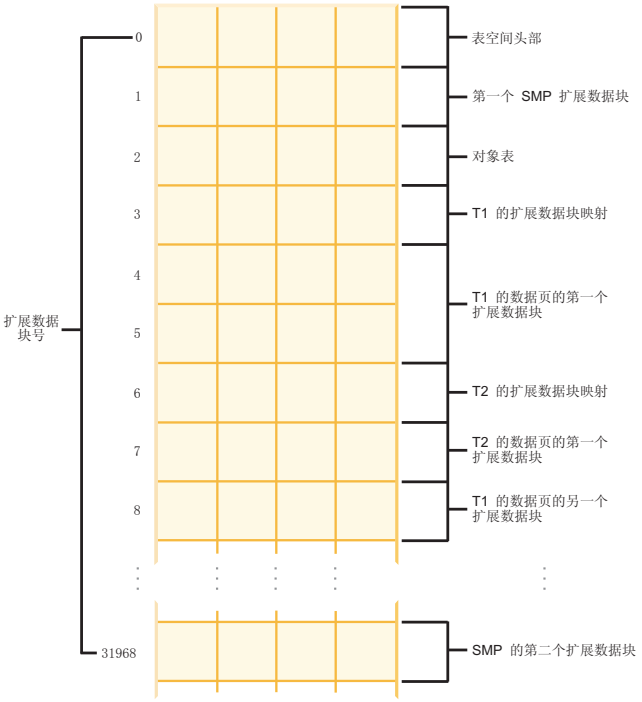


图 9. 逻辑表空间地址映射

数据库管理的表空间的表空间映射:

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 10 显示了 DMS 表空间的逻辑地址映射。

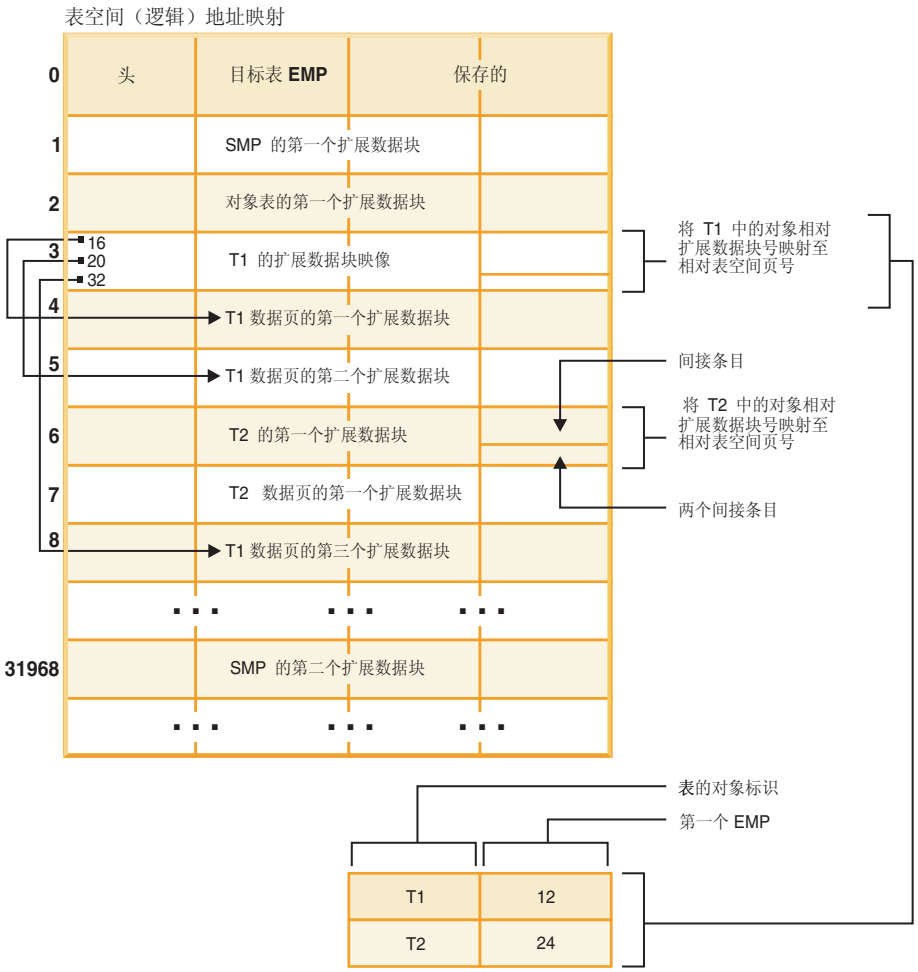


图 10. DMS 表空间

在表空间地址映射中，有两种类型的映射页：扩展数据块映射页 (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供使用。

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 11 中所示。

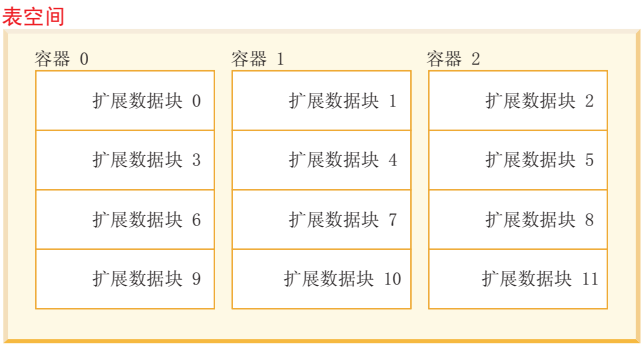


图 11. 带有三个容器和 12 个扩展数据块的表空间

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:

```
Range   Stripe Stripe Max      Max   Start   End    Adj.  Containers
Number  Set    Offset Extent   Page  Stripe  Stripe
   [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 12 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。

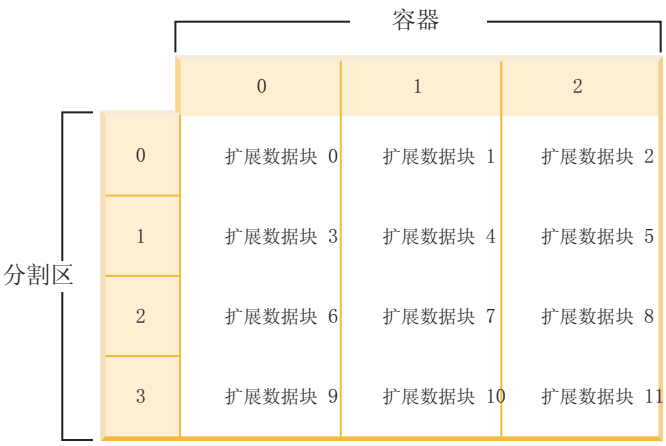


图 12. 带有三个容器和 12 个扩展数据块，突出显示分割区的表空间

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 13 中所示。

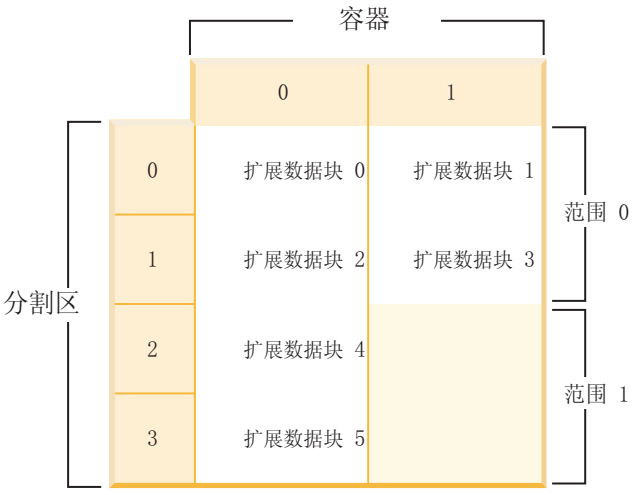


图 13. 带有两个容器，突出显示范围的表空间

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:

```
Range   Stripe Stripe   Max      Max   Start   End    Adj.  Containers
Number  Set    Offset   Extent   Page  Stripe  Stripe
    [0]    [0]      0        3     99       0       1  0    2 (0, 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，所以在这种情况下它是 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。

自动重新调整 DMS 表空间的大小:

通过启用使用文件容器进行自动大小调整的数据库管理的表空间 (DMS)，可以使数据库管理器能够通过扩展现有容器自动处理表空间变满情况。

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 语句来手动添加或扩展容器，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 表空间中的空间被耗用时，数据库系统将通过扩展一个或多个文件容器来增大该表空间的大小。

DMS 表空间的自动调整大小能力与自动存储器表空间的功能相关，但有所不同。有关更多信息，请参阅 第 157 页的『自动存储器表空间、SMS 表空间和 DMS 表空间的比较』。

要点：从 V10.1 FP1 开始，在用户定义的永久表空间中不推荐使用 DMS 表空间类型，在以后的发行版中可能会将其除去。对于目录和临时表空间，不推荐使用 DMS 表空间类型。有关更多信息，请参阅《*DB2 V10.1 新增内容*》中的『不推荐使用 DMS 永久表空间』。

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。表空间可增长至达到文件系统限制或表空间限制（请参阅*SQL Reference*中的『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
```

百分比值意味着每次需要增大表空间时都要计算增大大小；即，增大量基于该时间点的表空间大小的百分比。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。

关于对 DMS 表空间使用 AUTORESIZE 的限制

- 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，那么必须首先禁用自动调整大小功能。
- 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。
- 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)。
- 除了限制数据库管理器自动增大表空间的方式以外，最大大小还将限制您以手动方式增大表空间的程度。如果您执行向表空间添加空间的操作，那么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。

如何扩展表空间

如果 AUTORESIZE 处于启用状态，那么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，以便不需要对该表空间中的数据进行重新平衡。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 - 有关更多信息，请参阅第 137 页的『数据库管理的表空间的表空间映射』）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。）

```
Table space map:

 Range  Stripe Stripe  Max         Max  Start  End    Adj.   Containers
 Number Set    Offset  Extent      Page Stripe Stripe
 [   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:

```
Auto-resize enabled                     = Yes or No
Current tablespace size (bytes)         = ###
Maximum tablespace size (bytes)         = ### or NONE
Increase size (bytes)                   = ###
Increase size (percent)                 = ###
Time of last successful resize          = DD/MM/YYYY HH:MM:SS.SSSSSS
Last resize attempt failed              = Yes or No
```

自动存储器表空间

借助自动存储器表空间，可以自动管理存储器。数据库管理器根据需要创建并扩展容器。

注: 尽管可在创建数据库时指定 AUTOMATIC STORAGE NO 子句，但建议不要使用 AUTOMATIC STORAGE 子句，将来发行版可能会除去该子句。

您创建的任何表空间将作为自动存储器表空间管理，除非您另行指定或数据库是使用 AUTOMATIC STORAGE NO 子句创建的。有了自动存储器表空间，您就不需要提供容器定义；数据库管理器将负责创建和扩展容器，以便利用分配给数据库的存储器。如果向存储器组添加存储器，那么现有容器达到其最大容量时会自动创建新容器。如果您希望立即使用新添加的存储器，那么可以对表空间进行重新平衡，从而在扩充后的

这组新容器和分割集之间重新分配数据。或者，如果您不怎么关心 I/O 并行性，而只希望对表空间添加容量，那么可以提前进行重新平衡；在这种情况下，当需要新的存储器时，将创建新分割集。

可使用 CREATE TABLESPACE 语句在数据库中创建自动存储器表空间。缺省情况下，数据库中的新表空间是自动存储器表空间，因此 MANAGED BY AUTOMATIC STORAGE 子句是可选的。创建自动存储器表空间时，您还可以指定选项，例如，它的初始大小、表空间变满时将增加的表空间量、该表空间可以增长至的最大大小以及它使用的存储器组。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 LARGE TABLESPACE LONGTS
CREATE TABLESPACE TS3 INITIALSIZE 8K INCREASESIZE 20 PERCENT MANAGED BY AUTOMATIC STORAGE
CREATE TABLESPACE TS4 MAXSIZE 2G   CREATE TABLESPACE TS5 USING STOGROUP SG_HOT
```

其中每个示例假定正为其创建这些表空间的数据库具有一个或多个已定义存储器组。在未定义存储器组的数据库中创建表空间时，不能使用 MANAGED BY AUTOMATIC STORAGE 子句；必须创建存储器组，然后再次尝试创建自动存储器表空间。

自动存储器表空间管理存储器扩充的方式:

如果您正在使用*自动存储器表空间*，那么数据库管理器将根据需要来创建和扩展容器。如果您向表空间使用的存储器组添加存储器，那么将自动创建新容器。但是，使用新存储空间的方式取决于您是否重新平衡（REBALANCE）表空间。

创建自动存储器表空间时，如果空间足够的话，数据库管理器将在它定义为使用的存储器组的每个存储器路径上创建一个容器。在耗尽表空间中的所有空间之后，数据库管理器将通过扩展现有容器或者通过添加新的容器分割集来自动增大表空间大小。

自动表空间的存储器是在存储器组级别进行管理的；即，您向数据库的*存储器组*添加存储器，而不是像处理 DMS 表空间那样对表空间添加存储器。向表空间使用的存储器组添加存储器时，自动存储器功能将根据需要创建新容器以容纳数据。但是，已存在的表空间将不会立即开始耗用新路径中的存储器。需要增大表空间时，数据库管理器将首先尝试在表空间的最后一个*范围*中扩展那些容器。“范围”是指给定分割集的所有容器。如果此操作成功，那么应用程序将开始使用这个新空间。但是，如果扩展容器的尝试失败（例如，当一个或多个文件系统已满时，就会发生这种情况），那么数据库管理器将尝试创建新的容器分割集。只有在这个时刻，数据库管理器才会考虑将新添加的存储器路径用于表空间。第 146 页的图 14 演示了此过程。

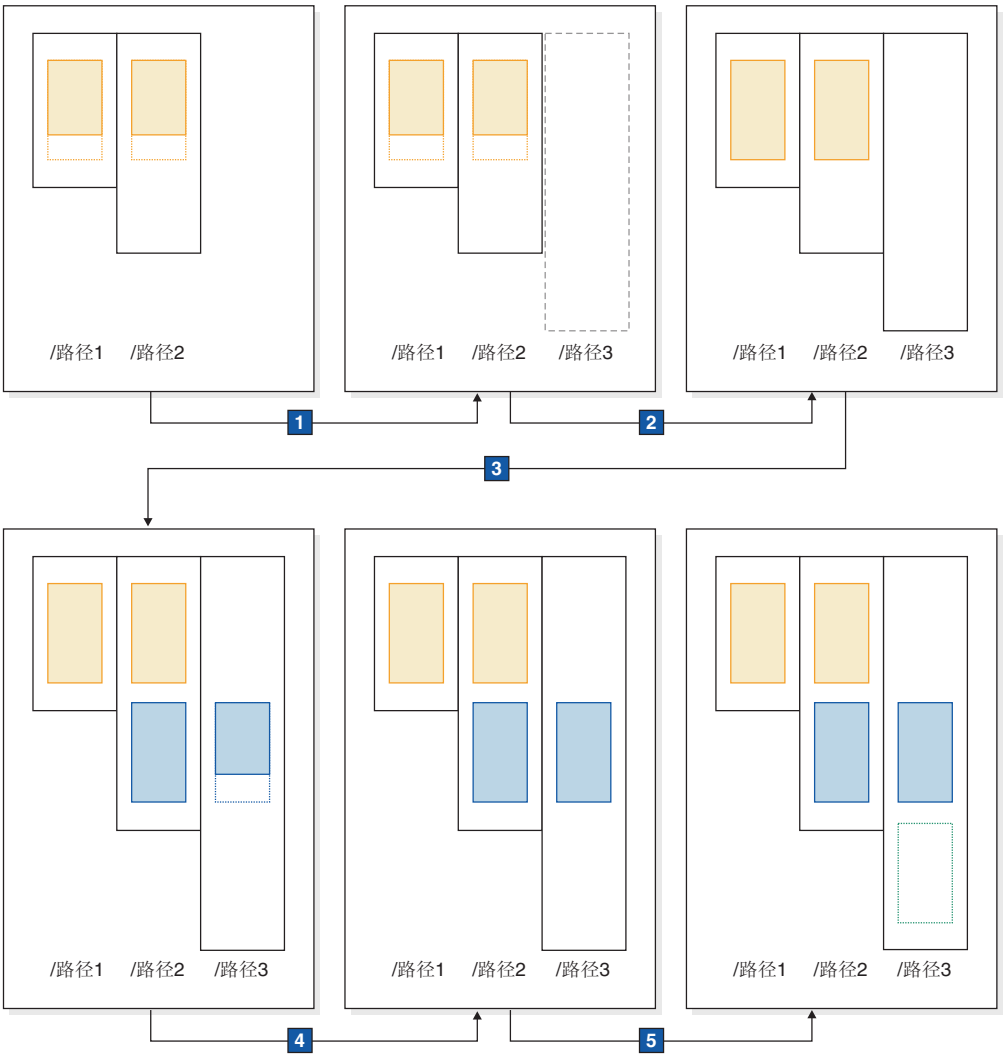


图 14. 自动存储器根据需要添加容器的方式

在上图中:

1. 最初，表空间包含两个尚未达到最大容量的容器。新存储器路径是您将 ALTER STOGROUP 语句与 ADD 子句配合使用添加至存储器组的。但是，新的存储器路径尚未被使用。
2. 最初的两个容器达到它们的最大容量。
3. 添加了新的容器分割集，它们开始填充数据。
4. 新分割集中的容器达到它们的最大容量。
5. 由于没有空间可供容器增大，因此添加新的分割集。

如果要让自动存储器表空间立即开始使用新添加的存储器路径，可以使用 ALTER TABLESPACE 命令的 REBALANCE 子句来执行重新平衡。如果对表空间进行重新平衡，那么将在新添加的存储器中的容器和分割集之间重新分配数据。第 147 页的图 15 对此作了说明。

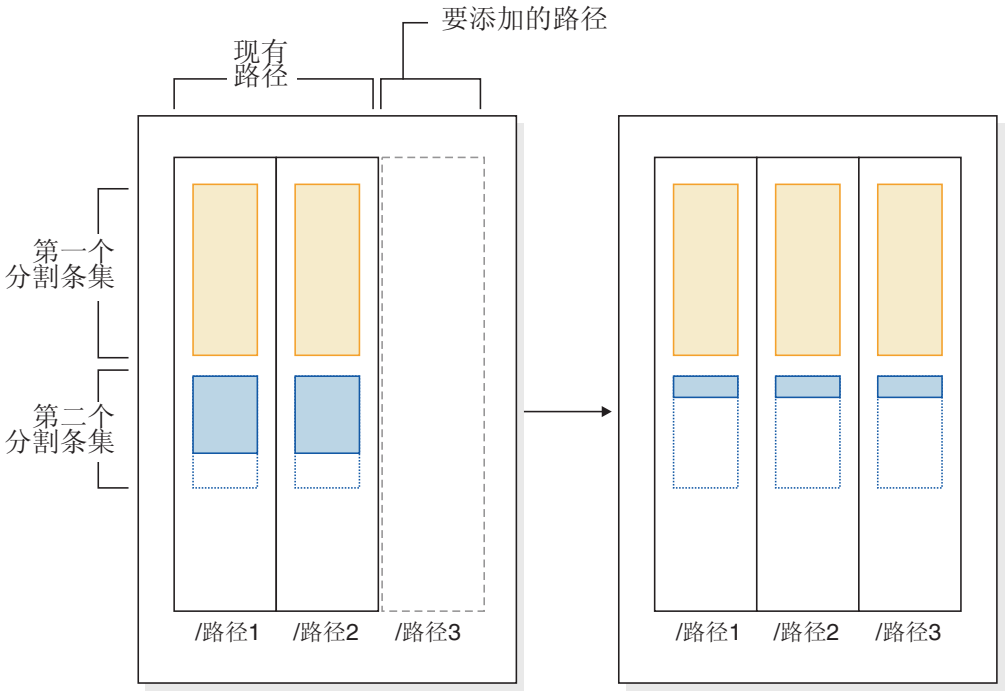


图 15. 添加新存储器并对表空间进行重新平衡的结果

在此示例中，重新平衡操作将现有分割集扩展到新的存储器路径（而不是创建新的分割集）以便根据需要创建容器，然后在所有容器之间重新分配数据。

自动存储器表空间中的容器名称:

虽然自动存储器表空间的容器名称由数据库管理器指定，但您可以通过运行运行（例如，**LIST TABLESPACE CONTAINERS** 或 **GET SNAPSHOT FOR TABLESPACES** 命令）来显示这些名称。本主题描述用于容器名称的约定，以便您识别这些名称。

对自动存储器表空间中的容器指定的名称按如下方式构造:

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

其中:

storage path
是与存储器组相关联的存储器路径

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

示例

例如，假定已在数据库 SAMPLE 中创建自动存储器表空间 TBSAUTO。运行 LIST TABLESPACES 命令时，它将显示为具有表空间标识 **10**:

```
 Tablespace ID                        = 10
 Name                                 = TBSAUTO
 Type                                 = Database managed space
 Contents                             = All permanent data. Large table space.
 State                                = 0x0000
   Detailed explanation:
     Normal
```

现在，如果对标识为 10 的表空间运行 **LIST TABLESPACE CONTAINERS** 命令，那么您可以查看对此表空间的容器指定的名称:

```
LIST TABLESPACE CONTAINERS FOR 10 SHOW DETAIL

            Tablespace Containers for Tablespace 10

 Container ID                         = 0
 Name                                 = D:\DB2\NODE0000\SAMPLE\T0000010\C0000000.LRG
 Type                                 = File
 Total pages                          = 4096
 Useable pages                        = 4064
 Accessible                           = Yes
```

在此示例中，您可以看到，此表空间中容器标识为 0 的容器的名称是

```
D:\DB2\NODE0000\SAMPLE\T0000010\C0000000.LRG
```

转换表空间以使用自动存储器:

您可以将数据库中的某些或全部数据库管理的空间 (DMS) 表空间转换为使用自动存储器。使用自动存储器将简化存储器管理任务。

开始之前

确保数据库至少有一个存储器组。为此，查询 SYSCAT.STOGROUPS 并发出 CREATE STOGROUP 语句（如果结果为空）。

过程

要将 DMS 表空间转换为使用自动存储器，请使用下列其中一个方法:

- **更改单一表空间。** 此方法将保持表空间处于联机状态，但将执行重新平衡操作，该操作将耗用一些时间将数据从非自动存储器容器移至新的自动存储器容器。
  1. 指定要转换为自动存储器的表空间。指示希望该表空间使用的存储器组。发出以下语句:

```
ALTER TABLESPACE tbspc1 MANAGED BY AUTOMATIC STORAGE USING STOGROUP sg_medium
```

其中 tbspc1 是表空间，sg_medium 是定义该表空间的存储器组。

2. 通过发出以下语句，将用户定义的数据从旧容器移至存储器组 sg_medium 中的存储器路径:

```
ALTER TABLESPACE tbspc1 REBALANCE
```

注: 如果现在未指定 REBALANCE 选项，并且以后发出带有 REDUCE 选项的 ALTER TABLESPACE 语句，那么自动存储器容器将被除去。要从此问题中恢复，请发出 ALTER TABLESPACE 语句并指定 REBALANCE 选项。

3. 要监视重新平衡操作的进度，请使用以下语句:

```
SELECT * from table (MON_GET_REBALANCE_STATUS( 'tbspc1', -2))
```

- **使用重定向的复原操作。** 当正在执行重定向的复原操作时，您无法访问正在转换的表空间。对于完全数据库重定向复原，在完成恢复之前，所有表空间都不可访问。

1. 运行 **RESTORE DATABASE** 命令，并指定 **REDIRECT** 参数。如果要转换单一表空间，那么还要指定 **TABLESPACE** 参数:

```
RESTORE DATABASE database_name TABLESPACE (table_space_name) REDIRECT
```

2. 运行 **SET TABLESPACE CONTAINERS** 命令，并对每个要转换的表空间指定 **USING AUTOMATIC STORAGE** 参数:

```
SET TABLESPACE CONTAINERS FOR tablespace_id USING AUTOMATIC STORAGE
```

3. 再次运行 **RESTORE DATABASE** 命令，这次指定 **CONTINUE** 参数:

```
RESTORE DATABASE database_name CONTINUE
```

4. 运行 **ROLLFORWARD DATABASE** 命令，并指定 **TO END OF LOGS** 和 **AND STOP** 参数:

```
ROLLFORWARD DATABASE database_name TO END OF LOGS AND STOP
```

如果使用重定向复原操作，那么必须发出另一 ALTER TABLESPACE 语句以使用表空间的正确存储器组关联来更新数据库目录。表空间与存储器组之间的关联记录在系统目录表中，并且在重定向复原期间不会更新。发出 ALTER TABLESPACE 语句仅更新目录表，不需要重新平衡操作的额外处理。如果未发出 ALTER TABLESPACE 语句，那么查询性能可能会受影响。如果在重定向复原操作期间修改了表空间的缺省存储器组，那么要使所有数据库分区和系统目录保持一致，请发出带有 **USING STOGROUP** 参数的 **RESTORE DATABASE** 命令。

示例

要在重定向复原期间将数据库管理的表空间 *SALES* 转换为自动存储器，请执行以下操作:

1. 要将重定向复原设为 testdb，请发出以下命令:

```
RESTORE DATABASE testdb REDIRECT
```

2. 将表空间 *SALES* 修改为由自动存储器管理。*SALES* 表空间的标识值为 5。

```
SET TABLESPACE CONTAINERS FOR 5 USING AUTOMATIC STORAGE
```

注: 要确定重定向复原期间表空间的标识值，请使用 **RESTORE DATABASE** 命令的 GENERATE SCRIPT 选项。

3. 要继续复原，请发出以下命令:

```
RESTORE DATABASE testdb CONTINUE
```

4. 更新目录表中的存储器组信息。

   ```
   CONNECT TO testdb
   ALTER TABLESPACE SALES MANAGED BY AUTOMATIC STORAGE
   ```

5. 如果在重定向复原操作期间修改了表空间的存储器组，请发出以下命令:

   ```
   RESTORE DATABASE testdb USING STOGROUP sg_default
   ```

表空间高水位标记

*高水位标记*是指位于所分配的最后一个扩展数据块之后的扩展数据块中第一页的页号。

例如，如果表空间有 1000 页，并且扩展数据块大小为 10，那么将有 100 个扩展数据块。如果第 42 个扩展数据块是表空间中最高的已分配扩展数据块，那么意味着高水位标记是 420。

提示: 扩展数据块的索引从 0 开始。因此，高水位标记是*最高已分配扩展数据块的最后一页* + 1。
实际上，您几乎不可能自己确定高水位标记；您可以通过管理性视图和表函数来确定当前高水位标记，尽管它可能随着行操作的进行而不断变化。

请注意，高水位标记并不是已用页数的指示器，这是因为高水位标记下方的某些扩展数据块可能已由于数据删除操作而被释放。在这种情况下，尽管高水位标记下方可能存在空闲的页，但高水位标记仍然是表空间中的最高已分配页。

您可以通过执行缩小表空间大小操作来合并扩展数据块，从而降低该表空间的高水位标记。

示例

第 151 页的图 16 显示了表空间中一系列已分配的扩展数据块。

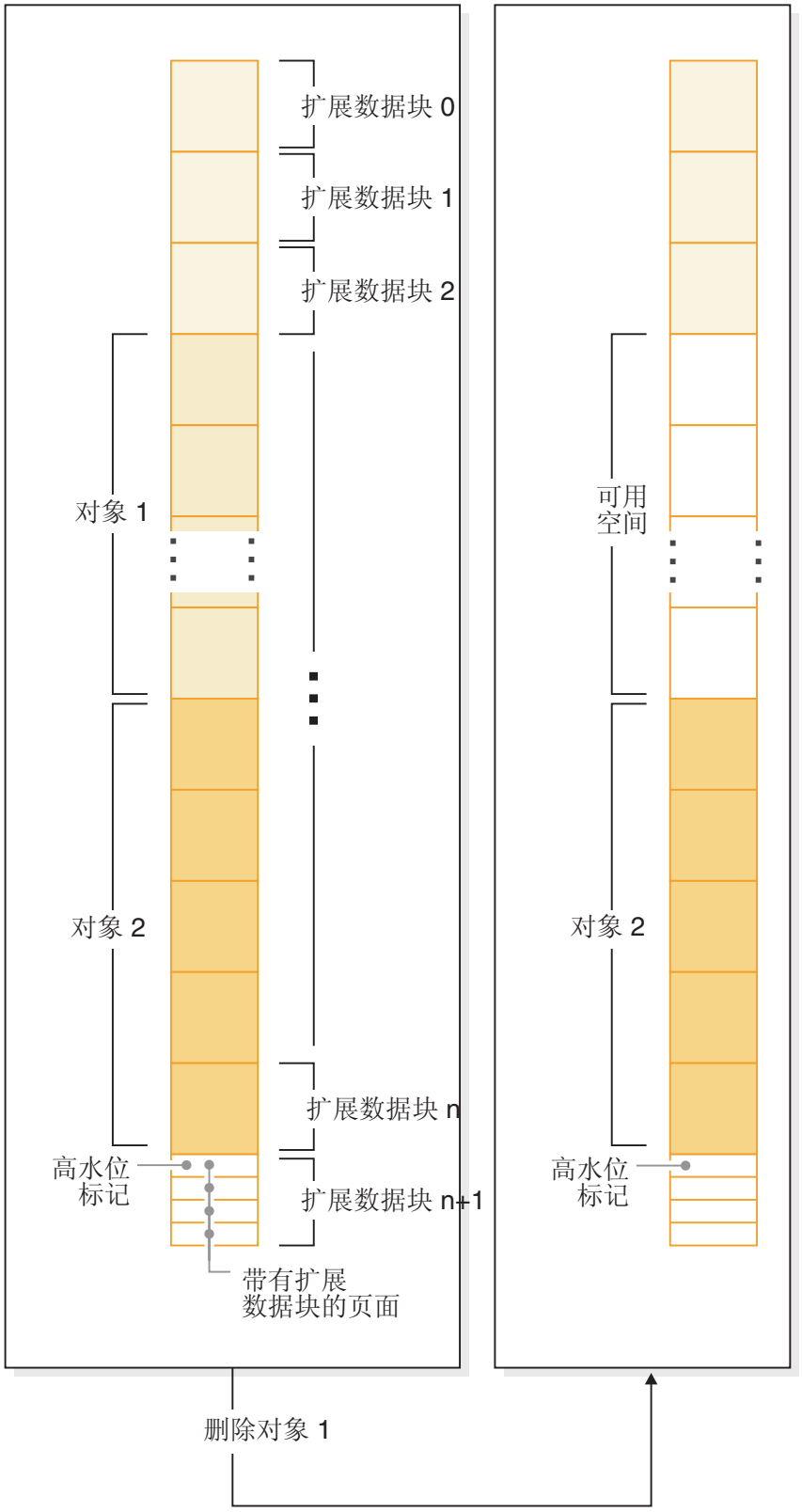

扩展数据块 0
扩展数据块 1
扩展数据块 2
对象 1
对象 2
扩展数据块 n
高水位
标记
扩展数据块 n+1
带有扩展
数据块的页面
可用
空间
对象 2
高水位
标记
删除对象 1

图 16. 高水位标记

删除对象时，将在表空间中释放空间。但是，在执行任何类型的存储器合并操作之前，高水位标记将一直处于先前水平。根据将新的扩展数据块添加到容器的方式不同，高水位标记甚至会处于更高水平。

可回收存储器

*可回收存储器*是 DB2 V9.7 及更高版本中的非临时自动存储器和 DMS 表空间的一个功能部件。可以使用它来整合正在使用且低于*高水位标记*的扩展数据块，并将表空间中未使用的扩展数据块返回至系统以供复用。

对于在 DB2 V9.7 之前创建的表空间，为系统释放存储器的唯一方法是 - 删除容器，或者通过消除*高于*高水位标记的未使用扩展数据块来减小容器大小。没有可直接降低高水位标记的机制。可以通过先卸载数据，然后将数据重新装入空白表空间来降低高水位标记；或者通过间接操作（例如，执行表和索引重组）来降低高水位标记。使用此万不得已的方法时，即使有低于高水位标记的可用扩展数据块，仍然有可能未能降低高水位标记。

在扩展数据块整合过程中，包含数据的扩展数据块将移至低于高水位标记的未使用扩展数据块。在移动扩展数据块之后，如果仍然存在低于高水位标记的可用扩展数据块，那么会将它们释放为可用存储器。接下来，高水位标记将移至表空间中刚好位于最后一个正在使用的扩展数据块后面的页。在包含可回收存储器的表空间中，可以使用 ALTER TABLESPACE 语句来回收未使用的扩展数据块。第 153 页的图 17 提供了有关可回收存储器工作方式的高级别视图。

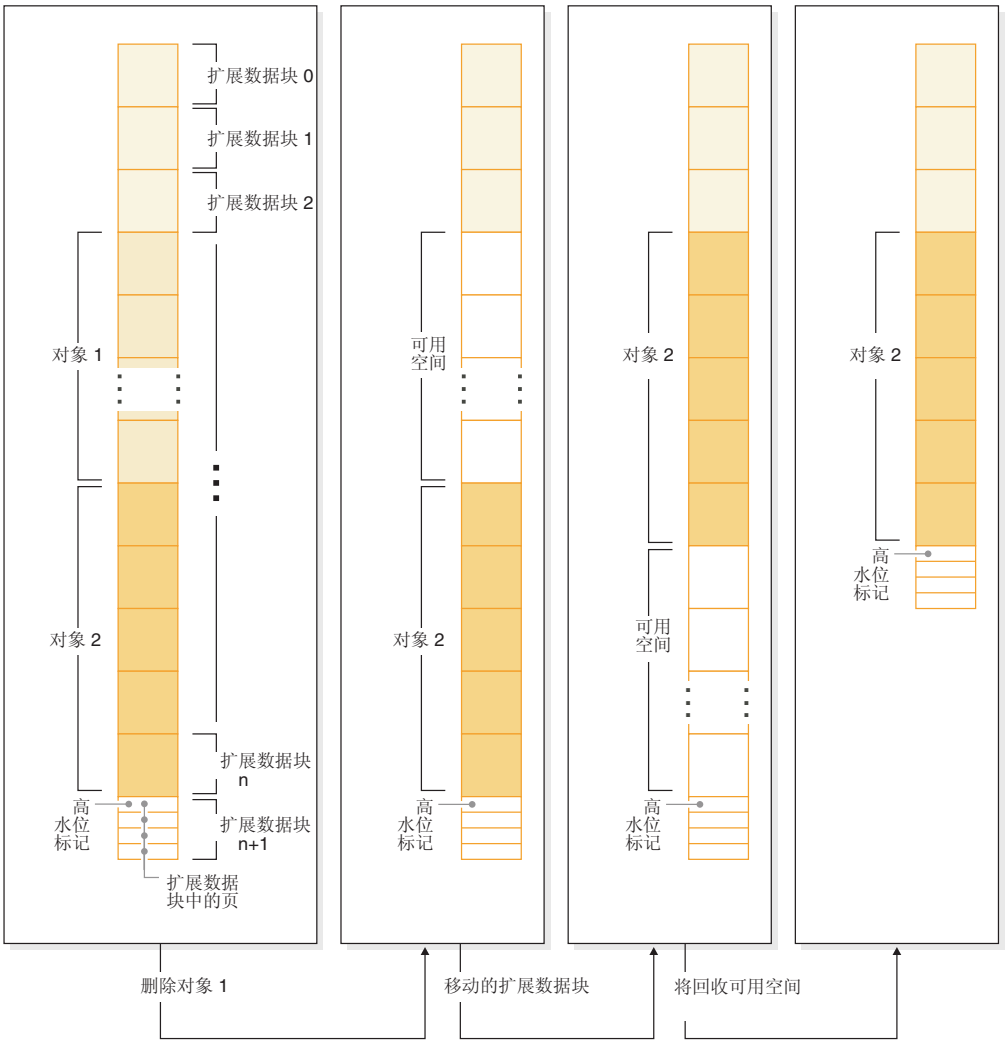


图 17. 可回收存储器的工作方式. 对表空间启用可回收存储器之后，就可以移动使用中的扩展数据块以使其占用表空间中位置较低的未使用扩展数据块。

在 DB2 V9.7 及更高版本中创建的所有非临时自动存储器和 DMS 表空间都能够合并低于高水位标记的扩展数据块。对于在较低版本中创建的表空间，必须首先将此表空间替换为使用 DB2 V9.7 创建的新表空间。您可以卸载并重新装入数据，也可以使用 SYSPROC.ADMIN_MOVE_TABLE 过程通过联机表移动操作来移动数据。但是，不是必须进行这种迁移。对其启用了可回收存储器的表空间可与没有可回收存储器的表空间共存于同一数据库。

通过移动扩展数据块来减小表空间大小是联机操作。换而言之，在执行减小操作期间，使用数据操作语言 (DML) 和数据定义语言 (DDL) 编写的代码可以继续运行。某些操作（例如，备份或复原）无法与扩展数据块移动操作同时运行。在这些情况下，需访问要移动的扩展数据块的进程（例如，备份）将一直等到移动了许多扩展数据块（用户不可配置此数目）为止；这时，备份进程将获得对于所讨论的扩展数据块的锁定并继续工作。

您可以使用 MON_GET_EXTENT_MOVEMENT_STATUS 表函数来监视扩展数据块移动操作的进度。

提示: 要使 ALTER TABLESPACE 语句最大限度地回收空间，首先应对表空间中的表和索引执行 REORG 操作。

自动存储器表空间

您可以通过多种方法来减小自动存储器表空间:

仅减小容器

选择此选项时，不会移动扩展数据块。数据库管理器将尝试通过首先释放存在暂挂删除的扩展数据块来减小容器大小。（某些“暂挂删除”扩展数据块可能由于可恢复性原因而无法被释放，因此，其中一些这样的扩展数据块可能会保留下来。）如果高水位标记先前在那些所释放的扩展数据块之间，那么将降低高水位标记，否则不会对其进行更改。接着，将调整容器的大小，以使表空间中的空间总量等于或略大于高水位标记。此操作本身通过带有 REDUCE 子句的 ALTER TABLESPACE 执行。

仅降低高水位标记

选择此选项时，将移动最大数目的扩展数据块以降低高水位标记，但是，不会执行容器大小调整操作。此操作本身通过带有 LOWER HIGH WATER MARK 子句的 ALTER TABLESPACE 执行。

降低高水位标记并将容器减小特定的容量

选择此选项时，可以指定要将表空间减小的绝对容量（以千字节、兆字节或吉字节计）。此外，也可以通过输入百分比值来指定要减小的相对容量。无论采用哪种方法，数据库管理器都会首先尝试在不移动扩展数据块的情况下将空间减小所请求的容量。即，它会尝试通过仅减小容器大小（如仅减小容器中所述）、释放删除暂挂扩展数据块以及尝试降低高水位标记来减小表空间。如果此方法未将表空间减小足够的容量，那么数据库管理器会开始将表空间中已使用的扩展数据块向下移，以便降低高水位标记。完成扩展数据块移动操作之后，将调整容器的大小，以使表空间中的空间总量等于或略大于高水位标记。如果由于没有足够的扩展数据块可移动，因此无法将表空间减小所请求的容量，那么高水位标记将尽可能降低。此操作是使用带有 REDUCE 子句的 ALTER TABLESPACE 执行的，该子句指定要将表空间大小减小的容量。

降低高水位标记并尽可能减小容器

在这种情况下，数据库管理器将尽可能移动扩展数据块，以减小表空间及其容器的大小。此操作通过带有 REDUCE MAX 子句的 ALTER TABLESPACE 执行。

启动扩展数据块移动过程之后，可以使用带有 REDUCE STOP 子句的 ALTER TABLESPACE 语句将其停止。这会落实已移动的任何扩展数据块，高水位标记会尽可能降低，并将容器大小调整为新的已降低的高水位标记。

DMS 表空间

可以通过两种方法来减小 DMS 表空间:

仅减小容器

选择此选项时，不会移动扩展数据块。数据库管理器将尝试通过首先释放存在暂挂删除的扩展数据块来减小容器大小。（某些“暂挂删除”扩展数据块可能由于可恢复性原因而无法被删除，因此，其中一些这样的扩展数据块可能会保留下来。）如果高水位标记先前位于所释放的那些扩展数据块之间，那么高水位标记将降低。否则，不会更改高水位标记。接着，将调整容器的大小，以使表

空间中的空间总量等于或略大于高水位标记。此操作本身通过带有 REDUCE *database-container* 子句的 ALTER TABLESPACE 执行。

仅降低高水位标记

选择此选项时，将移动最大数目的扩展数据块以降低高水位标记，但是，不会执行容器大小调整操作。此操作本身通过带有 LOWER HIGH WATER MARK 子句的 ALTER TABLESPACE 执行。

降低高水位标记和减小容器大小对于自动存储器表空间而言是结合在一起的自动化操作。相比之下，对于 DMS 表空间而言，要降低高水位标记和减小容器大小，必须执行以下两项操作:

1. 首先，必须使用带有 LOWER HIGH WATER MARK 子句的 ALTER TABLESPACE 语句来降低表空间的高水位标记。
2. 接着，必须单独使用带有 REDUCE *database-container* 子句的 ALTER TABLESPACE 语句来执行容器大小调整操作。

启动扩展数据块移动过程之后，可以使用带有 LOWER HIGH WATER MARK STOP 子句的 ALTER TABLESPACE 语句将其停止。这将落实任何已移动的扩展数据块，并将高水位标记减小为它的新值。

示例

示例 1: 最大程度地减小自动存储器表空间的大小。

假定数据库带有一个自动存储器表空间 TS 以及三个表 T1、T2 和 T3，那我们会删除表 T1 和 T3:

```
DROP TABLE T1
DROP TABLE T3
```

现在，假定现在扩展数据块处于空闲状态，那么以下语句将导致回收以前被 T1 和 T3 占用的扩展数据块并降低该表空间的高水位标记:

```
ALTER TABLESPACE TS REDUCE MAX
```

示例 2: 将自动存储器表空间的大小减小特定容量。

假定数据库带有一个自动存储器表空间 TS 以及两个表 T1 和 T2。接着，删除表 T1:

```
DROP TABLE T1
```

现在，要将表空间的大小减小 1 MB，请使用以下语句:

```
ALTER TABLESPACE TS REDUCE SIZE 1M
```

此外，可以使用类似如下的语句按表空间现有大小的百分比来减小表空间:

```
ALTER TABLESPACE TS REDUCE SIZE 5 PERCENT
```

示例 3: 在高水位标记以下存在可用空间时减小自动存储器表空间的大小。

与示例 1 类似，假定数据库带有一个自动存储器表空间 TS 以及三个表 T1、T2 和 T3。这次，当我们删除 T2 和 T3 时，在高水位标记正下方有 5 个空闲的扩展数据块组成的集合。现在，假定本例中的每个扩展数据块都由两个 4K 页组成，实际上有 40 KB 的可用空间位于高水位标记下方。如果发出类似如下的语句:

```
ALTER TABLESPACE TS REDUCE SIZE 32K
```

数据库管理器可以降低高水位标记并减小容器大小，而不需要执行任何扩展数据块移动操作。图 18 对此方案进行了说明。

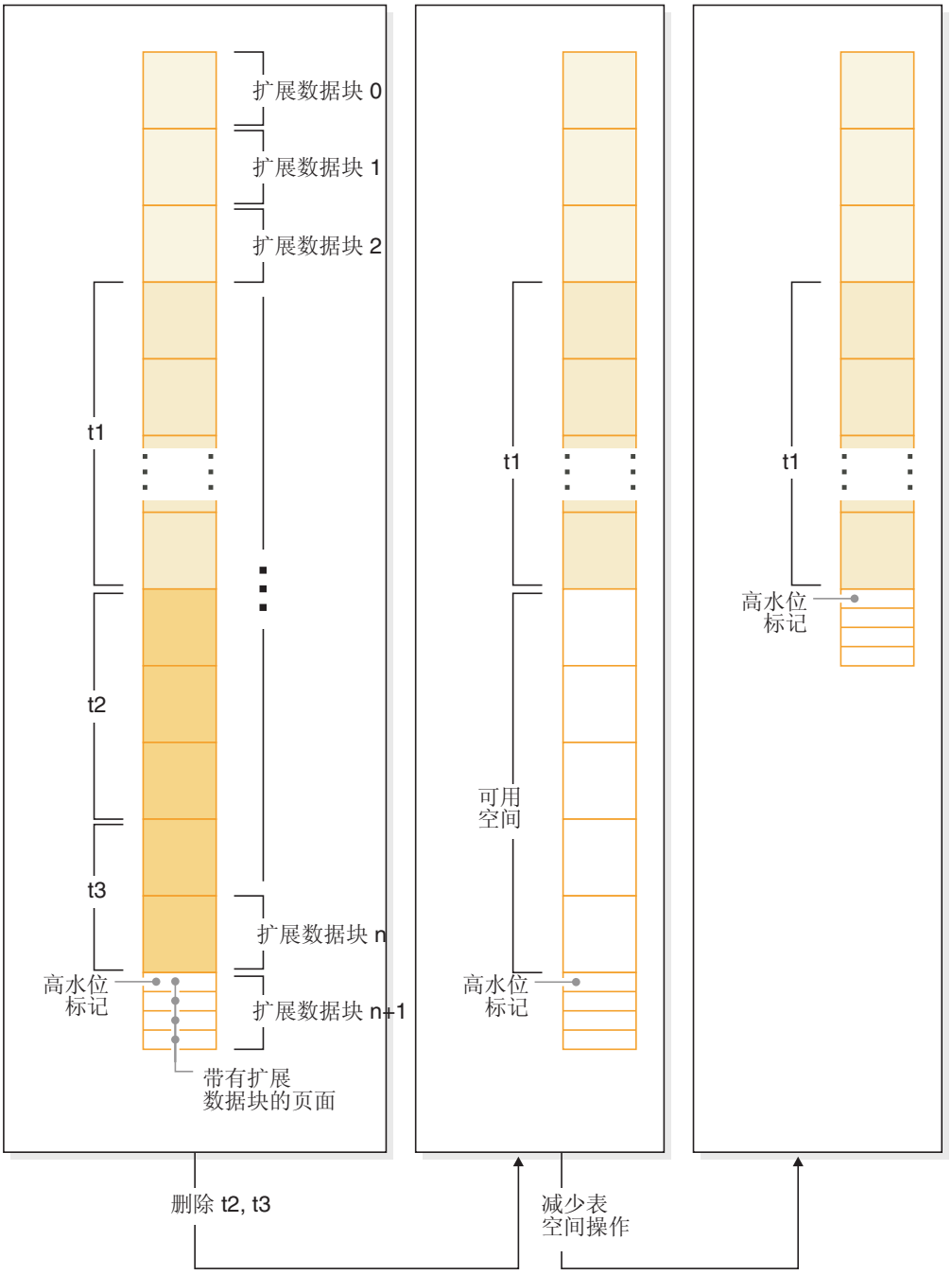


图 18. 降低高水位标记而不需要移动扩展数据块。

示例 4: 减小 DMS 表空间的大小。

假定数据库带有一个 DMS 表空间 TS 以及三个表 T1、T2 和 T3。接着，我们删除表 T1 和 T3:

```
DROP TABLE T1
DROP TABLE T3
```

对于 DMS 表空间而言，降低高水位标记和减小容器大小的操作分为两个步骤。首先，使用以下语句来移动扩展数据块，从而降低高水位标记:

```
ALTER TABLESPACE TS LOWER HIGH WATER MARK
```

接着，使用类似如下的语句来减小容器大小:

```
ALTER TABLESPACE TS REDUCE (ALL CONTAINERS 5 M)
```

自动存储器表空间、SMS 表空间和 DMS 表空间的比较

自动存储器表空间、SMS 表空间和 DMS 表空间提供了在不同环境中可能非常有益的不同功能。

要点: 在适用于用户定义的永久表空间的 V10.1 中已经不推荐使用 SMS 表空间类型，在以后的发行版中可能会将其除去。没有不推荐使用适用于目录和临时表空间的 SMS 表空间类型。有关更多信息，请参阅 《*DB2 V10.1 新增内容*》中的『不推荐使用 SMS 永久表空间』

要点: 从 V10.1 FP1 开始，在用户定义的永久表空间中不推荐使用 DMS 表空间类型，在以后的发行版中可能会将其除去。对于目录和临时表空间，不推荐使用 DMS 表空间类型。有关更多信息，请参阅《*DB2 V10.1 新增内容*》中的『不推荐使用 DMS 永久表空间』。

表 12. SMS 表空间、DMS 表空间与自动存储器表空间的比较

	自动存储器表空间	SMS 表空间	DMS 表空间
创建方式	使用 CREATE TABLESPACE 语句的 MANAGED BY AUTOMATIC STORAGE 子句或者通过完全省略 MANAGED BY 子句创建。如果创建数据库时启用了自动存储器，那么对于您创建的任何表空间，除非您另有指定，否则缺省情况是将其创建为自动存储器表空间。	使用 CREATE TABLESPACE 语句的 MANAGED BY SYSTEM 子句创建	使用 CREATE TABLESPACE 语句的 MANAGED BY DATABASE 子句创建
初始容器定义和位置	在创建自动存储器表空间时，您不需要提供容器列表。而是，数据库管理器在与该数据库相关联的所有存储器路径中自动创建容器。数据在所有容器之间均匀地进行分割，以使各条存储器路径的使用程度相同。	要求将容器定义为目录名。	• 要求将容器定义为文件或设备。 • 必须指定每个容器的初始大小。
空间的初始分配	• 对于非临时自动存储器表空间: – 在创建表空间时分配空间 – 您可以指定表空间的初始大小 • 对于临时自动存储器表空间而言，将根据需要来分配空间。	根据需要进行。由于文件系统控制存储器的分配，因此降低了页面连续的可能性，这可能会影响某些类型的查询的性能。	在创建表空间时进行。 • 与 SMS 表空间的情况相比，扩展数据块更有可能连续。 • 对于设备容器而言，扩展数据块中的页始终连续。

表 12. SMS 表空间、DMS 表空间与自动存储器表空间的比较 (续)

	自动存储器表空间	SMS 表空间	DMS 表空间
对表空间容器进行更改	• 在缩小表空间大小时，可以删除或减小容器。 • 对数据库添加新存储器或者从中删除存储器时，可以对表空间进行重新平衡，以便在各个容器之间均匀地分布数据。	创建后不进行更改，而不是在添加新数据分区时为其添加容器。	• 可以扩展或添加容器。如果在表空间的高水位标记之下添加新空间，那么将对表空间数据进行重新平衡。 • 可以缩小或删除容器。如果所删除的空间中包含数据，那么将进行重新平衡。
处理增加存储器需求	• 容器将自动地进行扩展，直到达到文件系统所施加的约束为止。 • 如果对数据库添加存储器路径，那么将自动地扩展或创建容器。	容器可以增大，直到达到文件系统所施加的容量限制为止。当任何一个容器达到其最大容量时，通常认为表空间已满。	可以手动地或自动地（需启用自动调整大小功能）对容器进行扩展，以使其超出最初分配的大小，直到达到文件系统所施加的约束为止。
能够将不同类型的对象放入不同的表空间	表、相关大对象 (LOB) 的存储器以及索引都可以驻留在不同的表空间中。	（仅限于分区表）索引和索引分区与表数据可以驻留在不同的表空间中。	表、相关大对象 (LOB) 的存储器以及索引都可以驻留在不同的表空间中。
持续维护需求	• 减小表空间的大小 • 降低高水位标记 • 重新平衡	无	• 添加或扩展容器 • 删除或减小容器 • 降低高水位标记 • 重新平衡
使用复原功能来重新定义容器	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，这是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	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
性能	类似于 DMS	通常低于 DMS 和自动存储器，对于较大的表而言尤其如此。	通常高于 SMS

自动存储器表空间是最容易设置和维护的表空间，建议用于大多数应用场合。在下列情况下，这些表空间尤其有益:

- 表较大，并且有可能迅速增大
- 您不想进行有关如何管理容器增长的常规决策。
- 您希望能够将不同类型的相关对象（例如表、LOB 和索引）存储在不同的表空间中，以便提高性能。

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。

要点: 在适用于用户定义的永久表空间的 V10.1 中已经不推荐使用 SMS 表空间类型，在以后的发行版中可能会将其除去。没有不推荐使用适用于目录和临时表空间的 SMS 表空间类型。有关更多信息，请参阅 《*DB2 V10.1 新增内容*》中的『不推荐使用 SMS 永久表空间』

要点: 从 V10.1 FP1 开始，在用户定义的永久表空间中不推荐使用 DMS 表空间类型，在以后的发行版中可能会将其除去。对于目录和临时表空间，不推荐使用 DMS 表空间类型。有关更多信息，请参阅《*DB2 V10.1 新增内容*》中的『不推荐使用 DMS 永久表空间』。

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，那么必须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。

要点: 从 V10.1 FP1 开始，在用户定义的永久表空间中不推荐使用 DMS 表空间类型，在以后的发行版中可能会将其除去。对于目录和临时表空间，不推荐使用 DMS 表空间类型。有关更多信息，请参阅《*DB2 V10.1 新增内容*》中的『不推荐使用 DMS 永久表空间』。

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 LONG 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，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。

  要点: 建议不要使用那些使用系统管理的空间 (SMS) 的用户表空间，将来的发行版可能会将其除去。请改用数据库管理的空间 (DMS) 或自动存储器表空间 (AMS)。

- **使用 LOB 或 LONG 数据**

  应用程序检索 LOB 或 LONG 数据时，数据库管理器不会将该数据高速缓存到其缓冲区中。每次应用程序需要其中一页时，数据库管理器都必须从磁盘检索该页。但是，如果 LOB 或 LONG 数据存储在 DMS 文件容器中，那么文件系统高速缓存可能会提供缓冲和（因此）更好的性能。

  因为系统目录包含一些 LOB 列，所以应将它们保留在 DMS 文件表空间中。

临时表空间

临时表空间存放数据库管理器在执行排序或连接之类的操作时所需的临时数据，这是因为，这些活动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。TEMPSPACE1 的页大小 TEMPSPACE1 是在创建数据库本身时指定的（缺省大小为 4 千字节）。

用户临时表空间存放使用 DECLARE GLOBAL TEMPORARY TABLE 或 CREATE GLOBAL TEMPORARY TABLE 语句创建的表的临时数据。用户临时表空间不是在创建数据库时缺省创建的。它们还存放已创建临时表的实例化版本。

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。由于这个原因，您应确保为现有表使用的每种不同的页大小定义了临时表空间，这样才可使用临时表空间重组这些表。

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。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。

自动存储器临时表空间（例如常规自动存储器表空间和大型自动存储器表空间）与存储器组相关联。但是，自动存储器临时表空间不能更改其存储器组关联。如果尝试对自动存储器临时表空间执行重新平衡操作，那么会返回 SQL0109N。要将临时表空间与存储器组相关联，可删除该临时表空间并使用另一存储器组重新创建该临时表空间。如果向存储器组添加存储器路径，那么直到下一次激活数据库，临时表空间才会使用新路径。

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。

要点: 建议不要使用那些使用系统管理的空间 (SMS) 的用户表空间，将来的发行版可能会将其除去。请改用数据库管理的空间 (DMS) 或自动存储器表空间 (AMS)。

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，还可以将表数据分布在四个不同的表空间中：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；最后一个存储 XML 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 表空间不支持在表空间之间进行此类型的数据分发。

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。

不使用文件系统高速缓存的表空间

建议在表空间级启用或禁用 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）还是并行 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，V9 之前未提供此功能。

要禁用或启用文件系统高速缓存，请在 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）。

要确定是否已启用文件系统高速缓存功能，请在 MON_GET_TABLESPACE 表中查询表空间的 **fs_caching** 监视元素值。

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并非来自数据库管理器的文件，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 V8.1 FP4 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 V9.1 或更高版本中设置此注册表变量相当于使用 NO FILE SYSTEM CACHING 子句更改所有表空间、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。 **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 V9.5 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下降，但性能下降不限于这种情况。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，那么更改表空间以使用“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，表空间容器使用并行 I/O 或直接 I/O

在大多数 AIX、Linux、Solaris 和 Windows 操作系统上创建的表空间容器的缺省 I/O 机制为 CIO/DIO（并行 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 属性指定是否要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。

如果已直接插入大对象 (LOB) 数据，那么会将它作为规则数据来访问，并使用为表空间 FILE SYSTEM CACHING 属性指定的 I/O 方法（已高速缓存或者未高速缓存）。

如果未直接插入大对象 (LOB) 数据，那么存在下列情况:

- 对于 SMS 表空间，即使设置了 NO FILE SYSTEM CACHING 表空间属性，也不会对长字段 (LF) 数据和大对象 (LOB) 数据请求未高速缓存的 I/O 访问。遵从操作系统配置和行为在文件系统高速缓存中进行缓存，并且可能会提高性能。
- 对于 DMS 表空间，当执行 I/O 时，DB2 不会区分不同的数据类型。除非在启用了 FILE SYSTEM CACHING 的情况下配置了表空间，否则不会缓存 LF 或 LOB 数据。如果您为了提高性能而希望缓存 DMS 表空间中的 LF 或 LOB 数据，那么可以将此数据放在单独的一个 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 165 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驻留在 AIX JFS 文件系统上并且发出了以下语句:

```
DB2 CREATE TABLESPACE JFS2
```

如果未指定此属性，那么数据库管理器将使用 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 V9.5 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 V9.5，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 V9.5 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。

要点： 在适用于用户定义的永久表空间的 V10.1 中已经不推荐使用 SMS 表空间类型，在以后的发行版中可能会将其除去。没有不推荐使用适用于目录和临时表空间的 SMS 表空间类型。有关更多信息，请参阅 《*DB2 V10.1 新增内容*》中的『不推荐使用 SMS 永久表空间』

受支持的配置

表 13 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 13. 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 6.1 及更高版本	日志文件系统 (JFS)	DIO	FILE SYSTEM CACHING（请参阅注 1.）
AIX 6.1 及更高版本	General Parallel File System (GPFS)	DIO	NO FILE SYSTEM CACHING
AIX 6.1 及更高版本	并行日志文件系统 (JFS2)	CIO	NO FILE SYSTEM CACHING
AIX 6.1 及更高版本	VERITAS Storage Foundation for DB2 4.1 (VxFS)	CIO	NO FILE SYSTEM CACHING
HP-UX V11i v3 (Itanium)	VERITAS Storage Foundation 4.1 (VxFS)	CIO	FILE SYSTEM CACHING
Solaris 10 和 11	UNIX 文件系统 (UFS)	CIO	FILE SYSTEM CACHING（请参阅注 2.）
Solaris 10 和 11	VERITAS Storage Foundation for DB2 4.1 (VxFS)	CIO	NO FILE SYSTEM CACHING
Linux 分发 SLES 10 SP3 或更高版本, 以及 RHEL 5.2 或更高版本 （在这些体系结构上: x86、x64 和 POWER®）	ext2、ext3 和 reiserfs	DIO	NO FILE SYSTEM CACHING
Linux 分发 SLES 10 SP3 或更高版本, 以及 RHEL 5.2 或更高版本 （在这些体系结构上: x86、x64 和 POWER）	VERITAS Storage Foundation 4.1 (VxFS)	CIO	NO FILE SYSTEM CACHING
Linux 分发 SLES 10 SP3 或更高版本, 以及 RHEL 5.2 或更高版本 （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 而不是上面指定的最低级别，那么必须使用 SFDB2 5.0 MP1 RP1 发行版。此发行版包括特定于版本 5.0 的修正。
5. 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
```

表空间中的扩展数据块大小

*扩展数据块*是表空间容器中的存储器块。它表示写下一个容器之前写入上一个容器的数据页数。在创建表空间时，您可以根据对性能以及存储器管理的要求来选择扩展数据块大小。

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。但是，如果您有一个增长速率高的非常大的表，且您正使用具有较小扩展数据块大小的 DMS 表空间，那么可能会更频繁地分配不必要的额外扩展数据块。

- 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。

页大小、表大小和表空间大小

对于 DMS、临时 DMS 和非临时自动存储器表空间而言，您为数据库选择的页大小确定了表空间大小的上限。对于 SMS 和临时自动存储器表空间中的表而言，页大小约束表本身的大小。

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 14 列示了 DMS 和非临时自动存储器表空间的表空间大小限制（按页大小排列）:

表 14. DMS 和非临时自动存储器表空间的大小限制. DMS 和非临时自动存储器表空间受页大小约束。

表空间类型	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 和非临时自动存储器表空间（常规）	64G	128G	256G	512G
DMS、临时 DMS 和非临时自动存储器表空间（大型）	8192G	16384G	32768G	65536G

第 169 页的表 15 列示了 SMS 和临时自动存储器表空间中的表大小限制（按页大小排列）:

表 15. SMS 和临时自动存储器表空间中的表的大小限制. 对于 SMS 和临时自动存储器表空间中的表而言，表对象本身（而不是表空间）受页大小约束。

表空间类型	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	64G	128G	256G	512G
临时 SMS 和临时自动存储器	8192G	16384G	32768G	65536G

要了解不同类型的表空间的数据库和索引页大小限制，请参阅 *SQL Reference*中的『SQL 和 XML 限制』中的特定于数据库管理器页大小的限制。

磁盘 I/O 效率和表空间设计

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

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。

创建表空间

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。

关于此任务

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器路径将容器指定给表空间。

对于非自动存储器表空间而言，在创建表空间时，您必须知道将要使用的容器的路径名、设备名或文件名。另外，对于您为 DMS 表空间创建的每个设备容器或文件容器，您还必须知道可以为每个容器分配的存储器空间量。

如果要指定 PREFETCHSIZE，那么使用的值必须是 EXTENTSIZE 值的倍数。例如，如果 EXTENTSIZE 是 10，那么 PREFETCHSIZE 应为 20 或 30。应该通过指定 AUTOMATIC 作为值，让数据库管理器自动确定预取大小。

将关键字 NO FILE SYSTEM CACHING 和 FILE SYSTEM CACHING 用作 CREATE TABLESPACE 语句的一部分来指定数据库管理器是使用直接 I/O (DIO) 还是使用并行 I/O (CIO) 来访问表空间。如果指定 NO FILE SYSTEM CACHING，那么只要可能，数据库管理器就会尝试使用 CIO。在不支持 CIO 的情况下（例如，如果使用的是 JFS），数据库管理器会改用 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。

如果计划删除许多表并且使用循环日志记录或者不想恢复任何已删除的表，请在发出 CREATE TABLESPACE 语句时通过将 DROPPED TABLE RECOVERY 选项显式设为 OFF 来禁用已删除的表的恢复功能。此外，也可以在创建表空间之后使用 ALTER TABLESPACE 语句来关闭已删除的表的恢复功能。

过程

- 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
  ```

  或者

  ```
  CREATE TABLESPACE name
        MANAGED BY AUTOMATIC STORAGE
  ```

  假定在自动存储器数据库中创建表空间，以上两条语句是等同的；缺省情况下，除非您另有指定，否则在此类数据库中创建的表空间将是自动存储器表空间。

- 要使用命令行来创建 SMS 表空间，请输入:

  ```
  CREATE TABLESPACE name
        MANAGED BY SYSTEM
        USING ('path')
  ```

  要点: 在适用于用户定义的永久表空间的 V10.1 中已经不推荐使用 SMS 表空间类型，在以后的发行版中可能会将其除去。没有不推荐使用适用于目录和临时表空间的 SMS 表空间类型。有关更多信息，请参阅 《*DB2 V10.1 新增内容*》中的『不推荐使用 SMS 永久表空间』

- 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)
  ```

  注意，在缺省情况下，会将 DMS 表空间创建为大型表空间。

  在创建 DMS 表空间之后，可以使用 ALTER TABLESPACE 语句对 DMS 表空间添加容器、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。

  要点: 从 V10.1 FP1 开始，在用户定义的永久表空间中不推荐使用 DMS 表空间类型，在以后的发行版中可能会将其除去。对于目录和临时表空间，不推荐使用 DMS 表空间类型。有关更多信息，请参阅《*DB2 V10.1 新增内容*》中的『不推荐使用 DMS 永久表空间』。

示例

示例 1: 在 Windows 上创建自动存储器表空间。

以下 SQL 语句在称为 STOGROUP1 的存储器组中创建称为 RESOURCE 的自动存储器表空间:

```
CREATE TABLESPACE RESOURCE
      MANAGED BY AUTOMATIC STORAGE
      USING STOGROUP STOGROUP1
```

示例 2: 在 Windows 上创建 SMS 表空间。

以下 SQL 语句将创建名为 RESOURCE 的 SMS 表空间，这个表空间使用三个不同驱动器上的三个不同目录中的容器:

```
CREATE TABLESPACE RESOURCE
      MANAGED BY SYSTEM
      USING ('d:\acc_tbsp', 'e:\acc_tbsp', 'f:\acc_tbsp')
```

示例 3: 在 Windows 上创建 DMS 表空间。

以下 SQL 语句将创建一个 DMS 表空间，这个表空间有两个文件容器，每个容器的大小均为 5000 页:

```
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，可能会出现下列方案:

- 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
```

示例 4: 在 AIX 上创建 DMS 表空间。

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。

示例 5: 在 UNIX 系统上创建 DMS 表空间。

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。

示例 6: 创建页大小大于缺省值的 SMS 表空间。

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。

创建临时表空间

临时表空间存放数据库管理器在执行排序或连接之类的操作时所需的临时数据，这是因为，这些活动需要额外的空间来处理结果集。使用 CREATE TABLESPACE 语句的变体来创建临时表空间。

关于此任务

*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中的一个便是名为“TEMPSPACE1”的系统临时表空间。至少应该有一个具有数据库中存在的用户表空间的各个页大小的系统临时表空间，否则某些查询可能会失败。有关更多信息，请参阅第 131 页的『系统数据、用户数据和临时数据的表空间』。

缺省情况下，创建数据库时并不会创建*用户临时表空间*。如果应用程序需要使用临时表，那么您必须创建临时表所驻留在的用户临时表空间。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。

限制

对于分区环境中的系统临时表空间而言，在创建系统临时表空间时，只能指定数据库分区组 IBMTEMPGROUP。

过程

- 要创建除缺省的 TEMPSPACE1 以外的系统临时表空间，请使用包含关键字 SYSTEM TEMPORARY 的 CREATE TABLESPACE 语句。 例如:

```
CREATE SYSTEM TEMPORARY TABLESPACE tmp_tbsp
  MANAGED BY SYSTEM
  USING ('d:\tmp_tbsp','e:\tmp_tbsp')
```

- 要创建用户临时表空间，请使用包含关键字 USER TEMPORARY 的 CREATE TABLESPACE 语句。 例如:

```
CREATE USER TEMPORARY TABLESPACE usr_tbsp
  MANAGED BY AUTOMATIC STORAGE
```

创建数据库时定义初始表空间

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。同时，您还可以创建其他用户表空间。

关于此任务

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
PAGESIZE page size
     CATALOG TABLESPACE
  MANAGED BY AUTOMATIC STORAGE              EXTENTSIZE value PREFETCHSIZE value
     USER TABLESPACE
```

```
  MANAGED BY AUTOMATIC STORAGE              EXTENTSIZE value PREFETCHSIZE value
     TEMPORARY TABLESPACE
  MANAGED BY AUTOMATIC STORAGE          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
PAGESIZE 16384
     CATALOG TABLESPACE
  MANAGED BY AUTOMATIC STORAGE              EXTENTSIZE 16 PREFETCHSIZE 32
     USER TABLESPACE
  MANAGED BY AUTOMATIC STORAGE              EXTENTSIZE 32 PREFETCHSIZE 64
     TEMPORARY TABLESPACE
  MANAGED BY AUTOMATIC STORAGE          WITH "Personnel DB for BSchiefer Co"
```

在本示例中，缺省页大小设为 16384 字节，并且明确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）。

关于此任务

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，无格式的数据库分区），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指定表空间定义所需的其余信息。**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 V5.2 系统管理概念: 操作系统和设备*。

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）。

开始之前

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。

关于此任务

在 V9 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，建议不要使用此方法，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，**db2diag** 日志文件中将写入一条消息以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)
```

在 Linux 环境中工作时使用以下信息。在 Linux/390 上，数据库管理器不支持直接磁盘访问设备。

过程

要在 Linux 上配置原始 I/O，请执行以下操作:

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
   ```

   在此示例中，要使用的原始数据库分区为 /dev/sda5。它应该不包含任何有用的数据。

   表 16. 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 中的页数为:

   ```
   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。

更改表空间

要使用命令行来更改表空间，可使用 ALTER TABLESPACE 语句。

关于此任务

根据表空间类型的不同，您可以执行以下操作:

- 通过添加附加的容器增大表空间的大小
- 调整现有容器的大小
- 删除容器
- 对表空间进行重新平衡，以便开始使用新容器或者从已删除的容器中移走数据
- 降低表空间的高水位标记
- 减小表空间的整体大小

还可以重命名表空间，并将它从脱机方式切换至联机方式。

计算表空间的使用情况

可以通过 MON_GET_TABLESPACE 表函数来确定当前使用中的表空间量。此函数返回的信息可以帮助您确定是否应该尝试回收空闲存储器。

关于此任务

本任务提供的信息可以用于确定，表空间的高水位标记下方的哪些扩展数据块包含未使用的空间。根据此情况，您可以确定回收空闲存储器是否有益。

限制

虽然您可以确定有关所有表空间的各种使用情况属性，但只有通过 DB2 V9.7 或更高版本创建的表空间才具有可回收存储器功能。如果您希望能够回收通过先前版本的 DB2 数据库产品创建的表空间中的存储器，那么必须卸载数据并接着将其重新装入到通过 DB2 V9.7 创建的表空间或者通过联机移动操作来移动数据。

过程

要确定高水位标记下方的空闲空间量，请完成下列步骤:

1. 构造包含 MON_GET_TABLESPACE 表函数的 SELECT 语句，以便报告表空间的状态。例如，以下语句将显示所有数据库分区中所有表空间的总页数、空闲页数和已用页数:

```
SELECT varchar(tbsp_name, 30) as tbsp_name,
  reclaimable_space_enabled,
  tbsp_free_pages,
  tbsp_page_top,
  tbsp_usable_pages
FROM TABLE(MON_GET_TABLESPACE('',-2)) AS t
ORDER BY tbsp_free_pages ASC
```

2. 运行此语句。 您将看到类似如下的输出:

```
TBSP_NAME                      RECLAIMABLE_SPACE_ENABLED TBSP_FREE_PAGES      TBSP_PAGE_TOP        TBSP_USABLE_PAGES
------------------------------ ------------------------- -------------------- -------------------- --------------------
TEMPSPACE1                                             0                    0                    0                    1
SYSTOOLSTMPSPACE                                       0                    0                    0                    1
TBSP1                                                  1                    0                 1632                 1632
SMSDEMO                                                0                    0                    0                    1
SYSCATSPACE                                            1                 2012                10272                12284
```

```
USERSPACE1                                1            2496            1696            4064
IBMDB2SAMPLEREL                           1            3328             736            4064
TS1                                       1            3584             480            4064
TS2                                       1            3968              96            4064
TBSP2                                     1            3968              96            4064
TBSAUTO                                   1            3968              96            4064
SYSTOOLSPACE                              1            3976             116            4092

  12 record(s) selected.
```

3. 使用以下公式来确定高水位标记下方的空闲页数:

   freeSpaceBelowHWM = tbsp_free_pages - (tbsp_usable_pages - tbsp_page_top)

结果

通过使用步骤 第 178 页的2 中的报告中的信息，对于 USERSPACE1，高水位标记下方的空闲空间量将是 2496 - (4064 - 1696) = 128 页。此数目仅略超出该表空间中空闲页总数的 5%。

下一步做什么

在这种情况下，可能不值得尝试回收此空间。但是，如果您希望回收那 128 页，那么可以运行 ALTER TABLESPACE USERSPACE1 REDUCE MAX 语句。如果您已执行此操作，然后再次运行 MON_GET_TABLESPACE 表函数，那么将看到以下输出:

```
TBSP_NAME                    RECLAIMABLE_SPACE_ENABLED TBSP_FREE_PAGES      TBSP_PAGE_TOP        TBSP_USABLE_PAGES
---------------------------- ------------------------- -------------------- -------------------- --------------------
TEMPSPACE1                                           0                    0                    0                    1
USERSPACE1                                           1                    0                 1568                 1568
SYSTOOLSTMPSPACE                                     0                    0                    0                    1
TBSP1                                                1                    0                 1632                 1632
SMSDEMO                                              0                    0                    0                    1
SYSCATSPACE                                          1                 2012                10272                12284
IBMDB2SAMPLEREL                                      1                 3328                  736                 4064
TS1                                                  1                 3584                  480                 4064
TS2                                                  1                 3968                   96                 4064
TBSP2                                                1                 3968                   96                 4064
TBSAUTO                                              1                 3968                   96                 4064
SYSTOOLSPACE                                         1                 3976                  116                 4092

  12 record(s) selected.
```

更改 SMS 表空间

在创建 SMS 表空间之后，除以下例外情况以外，无法对其添加容器或者更改其容器大小: 在添加新的数据分区时，可以为那些分区对 SMS 表空间添加新容器。

更改 DMS 表空间

对于 DMS 表空间，可以添加、扩展、重新平衡、调整容器大小、删除或减少容器。

添加 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。

关于此任务

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。

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。

要点: 从 V10.1 FP1 开始，在用户定义的永久表空间中不推荐使用 DMS 表空间类型，在以后的发行版中可能会将其除去。对于目录和临时表空间，不推荐使用 DMS 表空间类型。有关更多信息，请参阅《*DB2 V10.1 新增内容*》中的『不推荐使用 DMS 永久表空间』。

示例

以下示例说明如何将两个新设备容器（各含 10000 页）添加到 Linux 或 UNIX 操作系统上的表空间:

```
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。

删除 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。

关于此任务

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，才允许删除容器。此限制是必须的，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。

如果删除容器，那么余下容器会重新编号，其容器标识从 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。

过程

要删除容器，在 ALTER TABLESPACE 语句上使用 DROP 选项。

调整 DMS 容器的大小

数据库管理的表空间 (DMS) 中的容器可以随着存储器需求的变化而调整大小。如果您对 DMS 容器启用自动调整大小功能，那么数据库管理器将自动进行大小调整。即使未启用自动调整大小选项，您也可以手动地进行调整。

关于此任务

要将 DMS 表空间中的一个或多个容器增大指定的大小，请使用 ALTER TABLESPACE 命令的 EXTEND 选项；要减小现有容器的大小，请使用 REDUCE 选项。使用 EXTEND 或 REDUCE 时，请指定要将大小增大的数量或者要将当前大小减小的数量。换而言之，大小将相对于当前大小进行调整。

还可以使用 ALTER TABLESPACE 语句的 RESIZE 选项。使用 RESIZE 时，请对受影响的容器指定新大小。换而言之，此大小将被解释成所指定容器的绝对大小。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。

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。

限制

- 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，请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。
- 在 DMS 表空间中，容器大小必须至少为扩展数据块大小页的两倍。容器的最大大小与操作系统有关。

示例

示例 1: 增大文件容器的大小。以下示例说明如何在 Windows 操作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)
```

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。

示例 2: 增大设备容器的大小。以下示例说明如何在 Linux 和 UNIX 操作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)
```

两个设备的大小都从 1000 页增大至 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。

示例 3: 使用 REDUCE 选项来减小容器大小。以下示例说明如何在 Windows 操作系统上的表空间中缩小文件容器（包含 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。

重新平衡 DMS 容器

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。通常，您在对数据库添加存储器路径或者从中删除存储器路径时对表空间进行重新平衡。

添加或删除容器对重新平衡的影响

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（“之前”）。）

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。

对表空间添加新容器或者扩展现有容器时，如果新空间被添加到表空间的*高水位标记*下方，那么将对表空间数据进行重新平衡。如果新空间被添加到高水位标记上方，或者您正在创建新的分割集，那么不会自动执行重新平衡。为了利用所添加的存储器而进行的重新平衡称为*正向重新平衡*；在这种情况下，扩展数据块移动处理将从扩展数据块 0（表空间中的第一个扩展数据块）开始并向上持续到紧跟在高水位标记下方的扩展数据块。

添加容器将始终在高水位标记下方添加空间，这就是添加容器时通常有必要执行重新平衡的原因。可以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。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。

从表空间中删除容器时，如果所删除的空间包含数据，那么将自动执行重新平衡。在这种情况下，此重新平衡称为*反向重新平衡*；扩展数据块移动处理将从高水位标记开始并向下持续到表空间中的第一个扩展数据块。

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。

正向重新平衡

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，当前映射每次更改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，那么将使用该方法来对它进行放置（请参阅示例 1（“之后”））。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。

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。

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。

反向重新平衡

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，当前映射每次更改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。

示例

示例 1（之前）：添加容器之前的表空间布局

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，这相当于 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 184 页的图 19 中所示的图解。

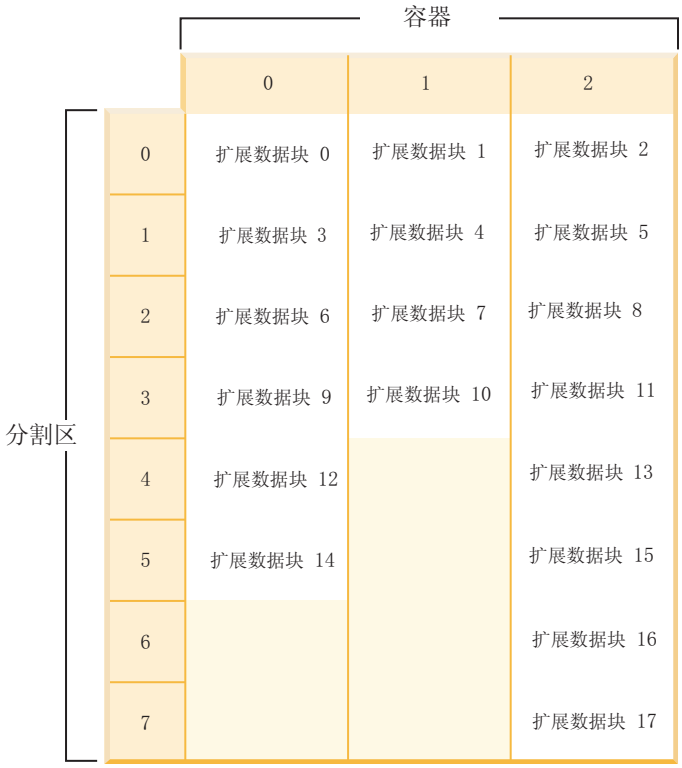


图 19. 带有三个容器和 18 个扩展数据块的表空间

相应的表空间映射，如表空间快照中所示，类似如下:

```
Range     Stripe  Stripe  Max        Max     Start    End     Adj.   Containers
Number    Set     Offset  Extent     Page    Stripe   Stripe
   [0]       [0]      0      11      119        0        3    0   3 (0, 1, 2)
   [1]       [0]      0      15      159        4        5    0   2 (0, 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 1（之后）：添加容器导致执行正向重新平衡

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 185 页的图 20 中所示的图解。

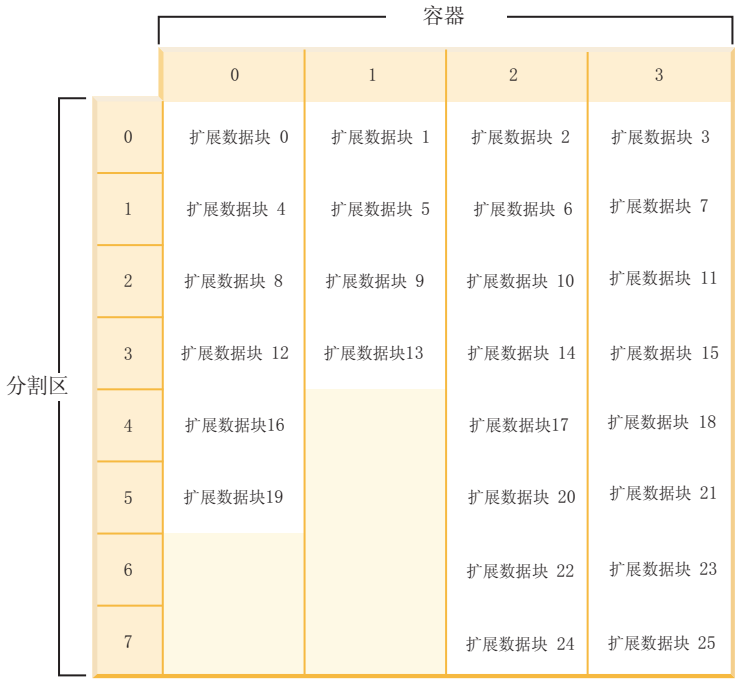


图 20. 带有四个容器和 26 个扩展数据块的表空间

相应的表空间映射，如表空间快照中所示，类似如下:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]      0      15      159       0        3    0   4 (0, 1, 2, 3)
   [1]       [0]      0      21      219       4        5    0   3 (0, 2, 3)
   [2]       [0]      0      25      259       6        7    0   2 (2, 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。

如果更改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。如果更改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 2: 扩展容器

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第 186 页的图 21。

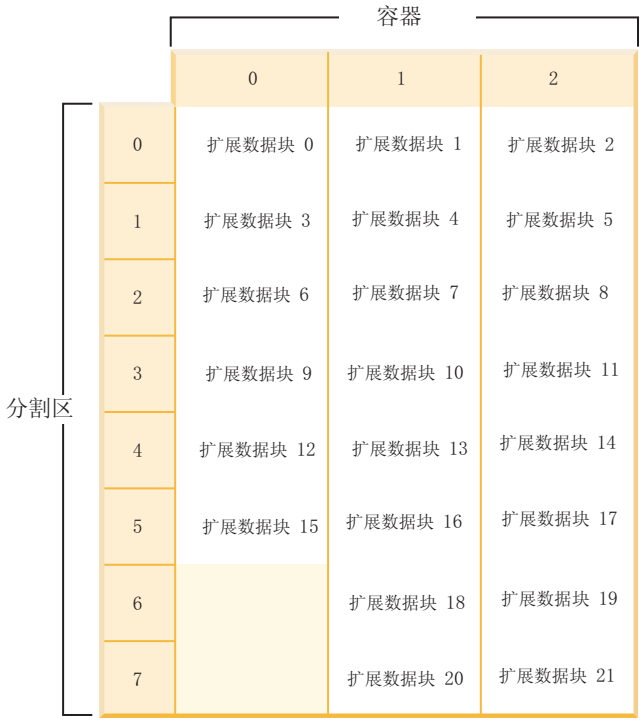


图 21. 带有三个容器和 22 个扩展数据块的表空间

相应的表空间映射，如表空间快照中所示，类似如下:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]      [0]       0      17      179        0        5    0   3 (0, 1, 2)
   [1]      [0]       0      21      219        6        7    0   2 (1, 2)
```

示例 3: 添加大小不足以确保从第一个分割区开始并在最后一个分割区结束的容器

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 187 页的图 22。）

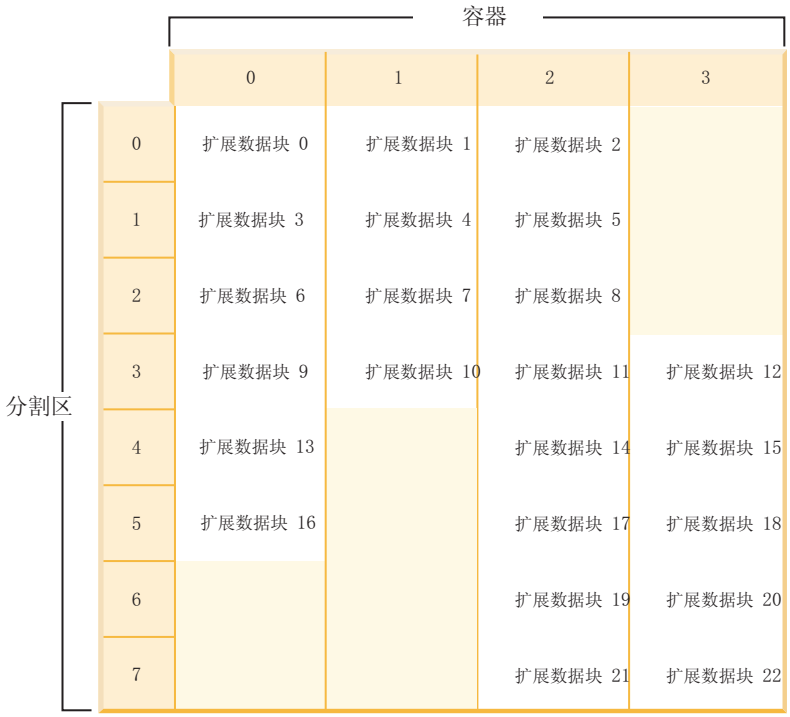


图 22. 带有四个容器和 23 个扩展数据块的表空间

相应的表空间映射，如表空间快照中所示，类似如下:

```
Range     Stripe  Stripe  Max        Max     Start    End     Adj.   Containers
Number    Set     Offset  Extent     Page    Stripe   Stripe
   [0]      [0]       0       8       89         0        2     0   3 (0, 1, 2)
   [1]      [0]       0      12      129         3        3     0   4 (0, 1, 2, 3)
   [2]      [0]       0      18      189         4        5     0   3 (0, 2, 3)
   [3]      [0]       0      22      229         6        7     0   2 (2, 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，例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必须移动多少个扩展数据块。

示例 4: 删除容器导致执行反向重新平衡

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 23 中。

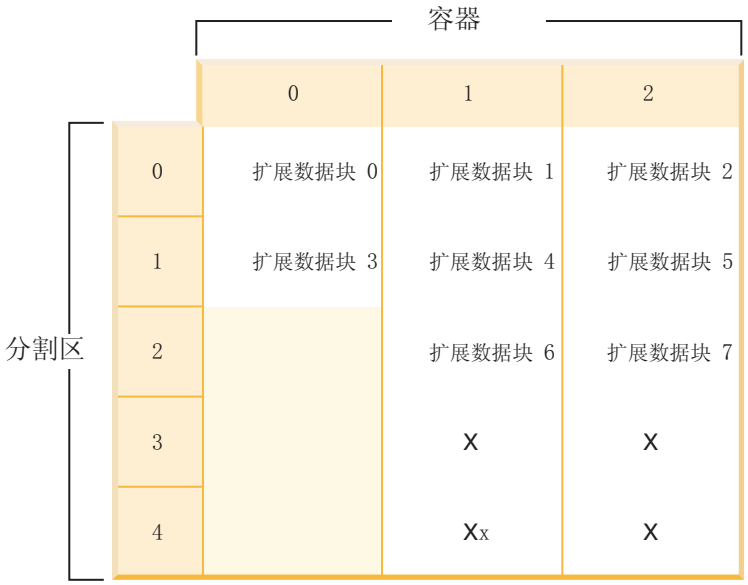


图 23. 带有 12 个扩展数据块（包括四个无数据的扩展数据块）的表空间

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:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]       [0]      0       5       59        0        1    0   3 (0, 1, 2)
   [1]       [0]      0      11      119        2        4    0   2 (1, 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 189 页的图 24 中。

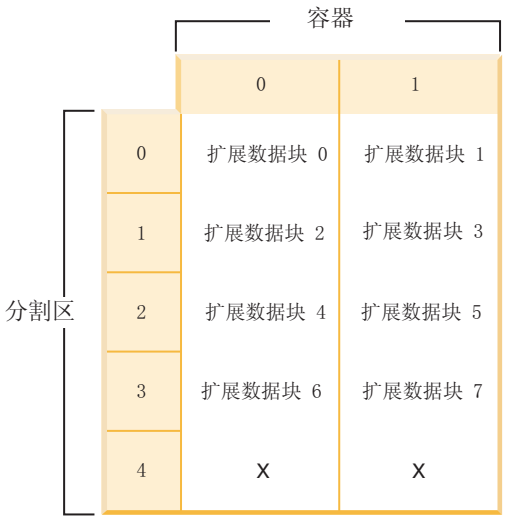


图 24. 删除容器之后的表空间

相应的表空间映射，如表空间快照中所示，类似如下:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]      [0]       0       9       99        0       4    0   2 (0, 1)
```

示例 5: 添加新的分割区

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 25 中所示的图解。

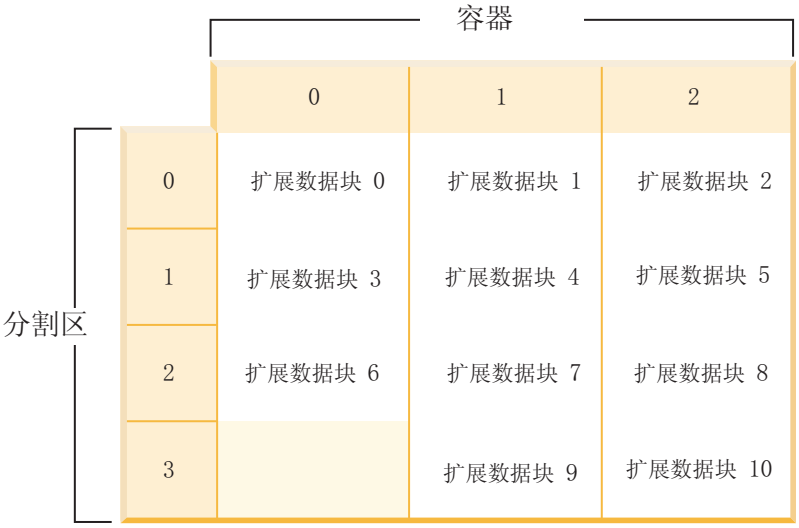


图 25. 带有三个容器和 11 个扩展数据块的表空间

相应的表空间映射，如表空间快照中所示，类似如下:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]      [0]       0       8       89        0       2    0   3 (0, 1, 2)
   [1]      [0]       0      10      109        3       3    0   2 (1, 2)
```

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 26。

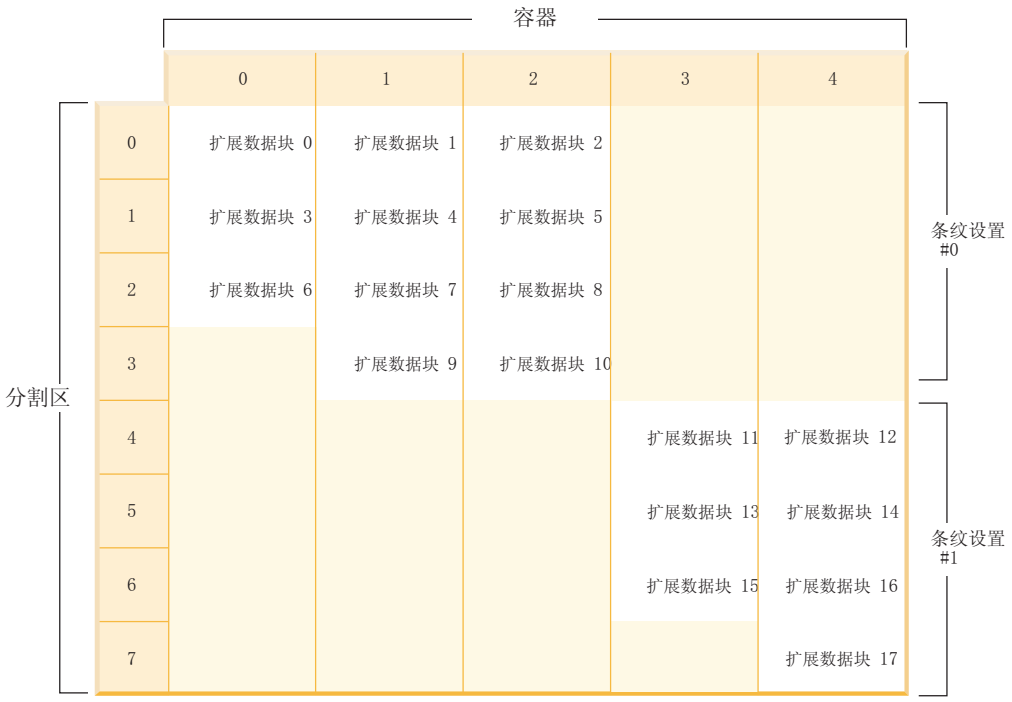


图 26. 带有两个分割集的表空间

相应的表空间映射，如表空间快照中所示，类似如下:

```
Range     Stripe  Stripe  Max        Max    Start    End    Adj.   Containers
Number    Set     Offset  Extent     Page   Stripe   Stripe
   [0]      [0]       0       8       89        0        2    0   3 (0, 1, 2)
   [1]      [0]       0      10      109        3        3    0   2 (1, 2)
   [2]      [1]       4      16      169        4        6    0   2 (3, 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。

监视表空间重新平衡操作

可使用 **MON_GET_REBALANCE_STATUS** 表函数来监视针对数据库的重新平衡操作的进度。

关于此任务

仅当正在进行重新平衡操作时，此过程才会返回有关表空间的数据。否则，不会返回任何数据。

过程

要监视表空间重新平衡操作，请执行以下操作:

发出带有 **tbsp_name** 和 **dbpartitionnum** 参数的 **MON_GET_REBALANCE_STATUS** 表函数:

```
select
   varchar(tbsp_name, 30) as tbsp_name,
   dbpartitionnum,
   member,
   rebalancer_mode,
   rebalancer_status,
   rebalancer_extents_remaining,
   rebalancer_extents_processed,
   rebalancer_start_time
from table(mon_get_rebalance_status(NULL,-2)) as t
```

结果

以下输出是监视表空间重新平衡操作进度的典型输出:

```
TBSP_NAME                      DBPARTITIONNUM MEMBER REBALANCER_MODE
------------------------------ -------------- ------ ------------------------------
SYSCATSPACE                                 0      0 REV_REBAL

REBALANCER_STATUS REBALANCER_EXTENTS_REMAINING REBALANCER_EXTENTS_PROCESSED REBALANCER_START_TIME
----------------- ---------------------------- ---------------------------- --------------------------
ACTIVE                                  6517                            4 2011-12-01-12.08.16.000000

  1 record(s) selected.
```

回收 DMS 表空间中未使用的存储器

您可以通过指示数据库管理器合并 DMS 表空间中位置较低且在使用中的扩展数据块来回收该表空间中未使用的存储器。这还将降低高水位标记。要减小 DMS 表空间中的容器大小，还必须执行单独的 REDUCE 操作。

开始之前

您必须要有使用 DB2 V9.7 或更高版本创建的 DMS 表空间。可回收存储器在使用先前版本的 DB2 产品创建的表空间中不可用。您还可以使用 MON_GET_TABLESPACE 表函数来查看数据库中的哪些表空间支持可回收存储器。

关于此任务

要回收 DMS 表空间中未使用的存储器，必须先启动一项操作对表中的扩展数据块进行重新安排，以便利用表空间中位置较低的空闲扩展数据块。要完成此任务，请使用 ALTER TABLESPACE 语句的 LOWER HIGH WATER MARK 子句。接着，可以将表空间中的容器大小减小指定的容量。

减小 DMS 表空间中容器的大小时，必须指定要减小的容器的名称或者使用 ALL CONTAINERS 子句。

限制

- 只能回收使用 DB2 V9.7 及更高版本创建的表空间中的存储器。
- 在 ALTER TABLESPACE 语句中指定 REDUCE 或 LOWER HIGH WATER MARK 子句时，无法指定其他参数。
- 如果包含当前被指定为高水位标记的页的扩展数据块处于“暂挂删除”状态，那么通过移动扩展数据块降低高水位标记的尝试可能会失败，此时将记录消息 ADM6008I。

为了能够进行恢复，处于“暂挂删除”状态的扩展数据块始终无法移动。这些扩展数据块最终将通过正常的数据库维护过程被释放，释放后即可移动。

- 在 DB2 数据共享环境中执行时，ALTER TABLESPACE 语句不支持下列子句:
  - ADD *database-container-clause*
  - BEGIN NEW STRIPE SET *database-container-clause*
  - DROP *database-container-clause*
  - LOWER HIGH WATER MARK
  - LOWER HIGH WATER MARK STOP
  - REBALANCE
  - REDUCE *database-container-clause*
  - REDUCE + LOWER HIGH WATER MARK 操作
  - RESIZE *database-container-clause*
  - USING STOGROUP

过程

1. 使用带有 LOWER HIGH WATER MARK 子句的 ALTER TABLESPACE 语句，通过重新安排表空间容器中的扩展数据块来尽可能降低高水位标记。
2. 使用带有 REDUCE 子句的 ALTER TABLESPACES 语句将某些或全部容器的大小减小指定的容量。

示例

*示例 1: 降低高水位标记并将所有容器减小 5 兆字节。*以下示例降低表空间 ts 的高水位标记并将该表空间中所有容器的大小减小 5 兆字节。

```
ALTER TABLESPACE ts LOWER HIGH WATER MARK
ALTER TABLESPACE ts REDUCE (ALL CONTAINERS 5 M)
```

*示例 2: 降低高水位标记并将容器“Container1”缩小 2000 页。*以下示例降低表空间 ts 的高水位标记并将“Container1”的大小缩小 2000 页。

```
ALTER TABLESPACE ts LOWER HIGH WATER MARK
ALTER TABLESPACE ts REDUCE (FILE "Container1" 2000)
```

添加或删除容器时调整预取大小

将为使用 DB2 V8.2 及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，那么您可能未发挥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优势。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。

可以手动设置表空间的预取大小；但是，一旦您这样做，并且希望保持最佳预取性能，那么在更改表空间中的容器时就务必更新此预取大小。使用 CREATE

TABLESPACE 或 ALTER TABLESPACE 语句时，通过将表空间的 PREFETCHSIZE 设为 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 语句手动更改预取大小。

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。将预取大小设为 4 × 1 × 8 = 32 将会得到 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。

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)。在启用并行 I/O 时，预取大小会增大为原来的六倍。
  - 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。

转换表空间以使用自动存储器

您可以将数据库中的某些或全部数据库管理的空间 (DMS) 表空间转换为使用自动存储器。使用自动存储器将简化存储器管理任务。

开始之前

确保数据库至少有一个存储器组。为此，查询 SYSCAT.STOGROUPS 并发出 CREATE STOGROUP 语句（如果结果为空）。

过程

要将 DMS 表空间转换为使用自动存储器，请使用下列其中一个方法:

- **更改单一表空间。** 此方法将保持表空间处于联机状态，但将执行重新平衡操作，该操作将耗用一些时间将数据从非自动存储器容器移至新的自动存储器容器。
  1. 指定要转换为自动存储器的表空间。指示希望该表空间使用的存储器组。发出以下语句:

     ```
     ALTER TABLESPACE tbspc1 MANAGED BY AUTOMATIC STORAGE USING STOGROUP sg_medium
     ```

     其中 tbspc1 是表空间，sg_medium 是定义该表空间的存储器组。
  2. 通过发出以下语句，将用户定义的数据从旧容器移至存储器组 sg_medium 中的存储器路径:

     ```
     ALTER TABLESPACE tbspc1 REBALANCE
     ```

     注: 如果现在未指定 REBALANCE 选项，并且以后发出带有 REDUCE 选项的 ALTER TABLESPACE 语句，那么自动存储器容器将被除去。要从此问题中恢复，请发出 ALTER TABLESPACE 语句并指定 REBALANCE 选项。
  3. 要监视重新平衡操作的进度，请使用以下语句:

     ```
     SELECT * from table (MON_GET_REBALANCE_STATUS( 'tbspc1', -2))
     ```
- **使用重定向的复原操作。** 当正在执行重定向的复原操作时，您无法访问正在转换的表空间。对于完全数据库重定向复原，在完成恢复之前，所有表空间都不可访问。
  1. 运行 **RESTORE DATABASE** 命令，并指定 **REDIRECT** 参数。如果要转换单一表空间，那么还要指定 **TABLESPACE** 参数:

```
RESTORE DATABASE database_name TABLESPACE (table_space_name) REDIRECT
```

2. 运行 **SET TABLESPACE CONTAINERS** 命令，并对每个要转换的表空间指定 **USING AUTOMATIC STORAGE** 参数:

```
SET TABLESPACE CONTAINERS FOR tablespace_id USING AUTOMATIC STORAGE
```

3. 再次运行 **RESTORE DATABASE** 命令，这次指定 **CONTINUE** 参数:

```
RESTORE DATABASE database_name CONTINUE
```

4. 运行 **ROLLFORWARD DATABASE** 命令，并指定 **TO END OF LOGS** 和 **AND STOP** 参数:

```
ROLLFORWARD DATABASE database_name TO END OF LOGS AND STOP
```

如果使用重定向复原操作，那么必须发出另一 ALTER TABLESPACE 语句以使用表空间的正确存储器组关联来更新数据库目录。表空间与存储器组之间的关联记录在系统目录表中，并且在重定向复原期间不会更新。发出 ALTER TABLESPACE 语句仅更新目录表，不需要重新平衡操作的额外处理。如果未发出 ALTER TABLESPACE 语句，那么查询性能可能会受影响。如果在重定向复原操作期间修改了表空间的缺省存储器组，那么要使所有数据库分区和系统目录保持一致，请发出带有 **USING STOGROUP** 参数的 **RESTORE DATABASE** 命令。

示例

要在重定向复原期间将数据库管理的表空间 *SALES* 转换为自动存储器，请执行以下操作:

1. 要将重定向复原设为 testdb，请发出以下命令:

```
RESTORE DATABASE testdb REDIRECT
```

2. 将表空间 *SALES* 修改为由自动存储器管理。*SALES* 表空间的标识值为 5。

```
SET TABLESPACE CONTAINERS FOR 5 USING AUTOMATIC STORAGE
```

注: 要确定重定向复原期间表空间的标识值，请使用 **RESTORE DATABASE** 命令的 GENERATE SCRIPT 选项。

3. 要继续复原，请发出以下命令:

```
RESTORE DATABASE testdb CONTINUE
```

4. 更新目录表中的存储器组信息。

```
CONNECT TO testdb
ALTER TABLESPACE SALES MANAGED BY AUTOMATIC STORAGE
```

5. 如果在重定向复原操作期间修改了表空间的存储器组，请发出以下命令:

```
RESTORE DATABASE testdb USING STOGROUP sg_default
```

更改自动存储器表空间

自动存储器表空间的许多维护工作是自动处理的。对于自动存储器表空间，您可以进行的更改限于重新平衡以及减小整个表空间的大小。

自动存储器表空间为您管理存储器的分配，从而根据需要来创建和扩展容器，直到达到存储器路径所施加的限制为止。对于自动存储器空间，您可以执行的维护操作只有:

- 重新平衡
- 通过降低高水位标记来回收未使用的存储器
- 减小整个表空间的大小

- 更改自动存储器表空间的存储器组

向存储器组添加存储器路径时，可重新平衡自动存储器表空间。这会导致该表空间立即开始使用新存储器路径。同样，从存储器组中删除存储器路径时，重新平衡操作会将数据移出您要删除的存储器路径中的容器并将其分配到其余容器中。

添加新存储器路径或者删除存储器路径都是在存储器组级别处理的。要向数据库添加存储器路径，请使用 ALTER STOGROUP 语句的 ADD 子句。您可以随意选择是否进行重新平衡，但是，如果不进行重新平衡，那么新存储器路径在先前存在的容器装满（达到容量限制）之前将不会被使用。如果进行重新平衡，那么任何新添加的存储器路径都将立即可供使用。

要删除存储器路径，请使用 ALTER STORGOUP 语句的 DROP 子句。此操作将存储器路径置于“删除暂挂”状态。所指定存储器路径中的容器将停止增大。但是，在可以从数据库中完全除去该路径之前，必须使用 ALTER TABLESPACE 命令的 REBALANCE 子句对所有使用该存储器路径的表空间进行重新平衡。如果某个临时表空间在处于删除暂挂状态的存储器路径中有容器，那么可以删除并重新创建该表空间，也可以重新启动数据库以将其从存储器路径中除去。

限制：无法对临时自动存储器表空间进行重新平衡；只有常规自动存储器表空间和大型自动存储器表空间才支持重新平衡操作。

可以使用 ALTER TABLESPACE 语句的 LOWER HIGH WATER MARK 子句来回收表空间的高水位标记下方的存储器。此操作的效果是，将尽量多的扩展数据块移至表空间中位置较低的未使用扩展数据块。此过程将会降低表空间的高水位标记，但是，容器将保持它们在此操作执行前具有的大小。

可以使用 ALTER TABLESPACE 语句的 REDUCE 选项来减小自动存储器表空间的大小。当您减小自动存储器表空间的大小时，数据库管理器将尝试降低该表空间的高水位标记并减小表空间容器的大小。在尝试降低高水位标记时，数据库管理器可能会删除空容器并将已使用的扩展数据块移至靠近表空间开头的空闲空间。接着，将调整容器的大小，以使表空间中的空间总量等于或略大于高水位标记。

回收自动存储器表空间中未使用的存储器

当您减小自动存储器表空间的大小时，数据库管理器将尝试降低该表空间的*高水位标记*并减小表空间容器的大小。在尝试降低高水位标记时，数据库管理器可能会删除空容器并将已使用的扩展数据块移至靠近表空间开头的空闲空间。接着，将调整容器的大小，以使表空间中的空间总量等于或略大于高水位标记。

开始之前

您必须要有使用 DB2 V9.7 或更高版本创建的自动存储器表空间。可回收存储器在使用先前版本的 DB2 产品创建的表空间中不可用。您还可以使用 MON_GET_TABLESPACE 表函数来查看数据库中的哪些表空间支持可回收存储器。

关于此任务

您可以减小以各种方式启用了可回收存储器的自动存储器表空间的大小。您可以指定数据库管理器通过下列方式来减小表空间:

- 最大的可能容量

- 您以千字节、兆字节、吉字节或页为单位指定的容量
- 表空间当前大小的百分比。

无论在哪种情况下，数据库管理器都将尝试通过将扩展数据块移至表空间开头来减小大小，如果有足够的空闲空间，这将降低表空间的高水位标记。移动扩展数据块完成后，表空间大小将减小到新的高水位标记。

使用 ALTER TABLESPACE 语句的 REDUCE 子句来减小自动存储器表空间的表空间大小。如上所述，您可以指定要将表空间减小的容量。

注:

- 如果您未指定要将表空间减小的容量，那么将在不移动扩展数据块的情况下尽量减小表空间大小。数据库管理器将尝试通过首先释放存在暂挂删除的扩展数据块来减小容器大小。（某些“暂挂删除”扩展数据块可能由于可恢复性原因而无法被释放，因此，其中一些这样的扩展数据块可能会保留下来。）如果高水位标记先前在那些所释放的扩展数据块之间，那么将降低高水位标记，否则不会对其进行更改。接着，将调整容器的大小，以使表空间中的空间总量等于或略大于高水位标记。此操作本身通过带有 REDUCE 子句的 ALTER TABLESPACE 执行。
- 如果您只想降低高水位标记，从而合并表空间中位置较低并且在使用中的扩展数据块，而不执行任何容器操作，那么可以使用带有 LOWER HIGH WATER MARK 子句的 ALTER TABLESPACE 语句。
- 在 REDUCE 或 LOWER HIGH WATER MARK 操作的执行期间，您可以使用 ALTER TABLESPACE 语句的 REDUCE STOP 或 LOWER HIGH WATER MARK STOP 子句将其停止。这将落实任何已移动的扩展数据块，将高水位标记减小为它的新值，并将按新的高水位标记来调整容器大小。

限制

- 只能回收使用 DB2 V9.7 及更高版本创建的表空间中的存储器。
- 在 ALTER TABLESPACE 语句中指定 REDUCE 或 LOWER HIGH WATER MARK 子句时，无法指定其他参数。
- 如果包含当前被指定为高水位标记的页的扩展数据块处于“暂挂删除”状态，那么通过移动扩展数据块降低高水位标记的尝试可能会失败，此时将记录消息 ADM6008I。为了能够进行恢复，处于“暂挂删除”状态的扩展数据块始终无法移动。这些扩展数据块最终将通过正常的数据库维护过程被释放，释放后即可移动。
- 在 DB2 数据共享环境中执行时，ALTER TABLESPACE 语句不支持下列子句:
  - ADD *database-container-clause*
  - BEGIN NEW STRIPE SET *database-container-clause*
  - DROP *database-container-clause*
  - LOWER HIGH WATER MARK
  - LOWER HIGH WATER MARK STOP
  - REBALANCE
  - REDUCE *database-container-clause*
  - REDUCE + LOWER HIGH WATER MARK 操作
  - RESIZE *database-container-clause*
  - USING STOGROUP

过程

要减小自动存储器表空间的大小，请完成下列步骤:

1. 构造包含 REDUCE 子句的 ALTER TABLESPACE 语句。

   ```
   ALTER TABLESPACE table-space-name REDUCE reduction-clause
   ```

2. 运行 ALTER TABLESPACE 语句。

示例

示例 1: 最大程度地减小自动存储器表空间的大小。

```
ALTER TABLESPACE TS1 REDUCE MAX
```

在本例中，指定了关键字 MAX 作为 REDUCE 子句的组成部分，从而表明数据库管理器应该尝试将最大数目的扩展数据块移至表空间开头。

示例 2: 按当前表空间大小的百分比来减小自动存储器表空间。

```
ALTER TABLESPACE TS1 REDUCE 25 PERCENT
```

这将尝试将表空间 TS1 的大小减小到原始大小的 75%（如果有可能的话）。

方案: 通过自动存储器表空间来添加和除去存储器

本节中的三个方案说明添加和除去存储器路径对自动存储器表空间的影响。

在存储器组中添加或除去存储器路径后，可以使用重新平衡操作在新存储器路径上创建一个或多个容器或从所删除路径中除去容器。对表空间进行重新平衡时，应该考虑下列几点:

- 如果数据库管理器由于任何原因而确定不需要添加或删除任何容器，或者由于“空间不足”情况而未能添加容器，那么您将接收到警告。
- 必须单独指定 REBALANCE 子句。
- 无法对临时自动存储器表空间进行重新平衡；只能对常规自动存储器表空间和大型自动存储器表空间进行重新平衡。
- 调用重新平衡是一项将会进行日志记录的操作，在前滚期间，将回放此操作（尽管存储器布局可能有所不同）。
- 在分区数据库环境中，将对表空间所驻留在的每个数据库分区启动重新平衡。
- 添加或删除存储器路径时，不会强制进行重新平衡。实际上，在执行重新平衡操作之前，可以随着时间的推移而执行后续存储器路径操作。如果存储器路径已被删除并处于“未在使用中”状态，那么它将在 ALTER STOGROUP 操作中立即删除。如果存储器路径处于“在使用中”状态并被删除，但未对表空间进行重新平衡，那么不会使用该存储器路径（它现在处于“删除暂挂”状态）来存储其他容器或数据。

方案: 添加存储器路径并对自动存储器表空间进行重新平衡:

本方案说明如何向存储器组添加存储器路径以及 REBALANCE 操作如何在新存储器路径中创建一个或多个容器。

本方案假定要向存储器组添加新的存储器路径，并且通过此新路径对现有表空间进行条带分割。通过将新容器添加到表空间的每个分割集，可以提高 I/O 并行性。

使用 ALTER STOGROUP 语句向存储器组添加新存储器路径。然后，使用 ALTER TABLESPACE 语句的 REBALANCE 子句在新存储器路径中分配容器以及将现有容器中的数据重新平衡到新容器。所要创建的容器的数目和大小取决于表空间的当前分割集的定义以及新存储器路径中的可用空间量。

图 27 说明添加存储器路径以及表空间在重新平衡前后的布局:

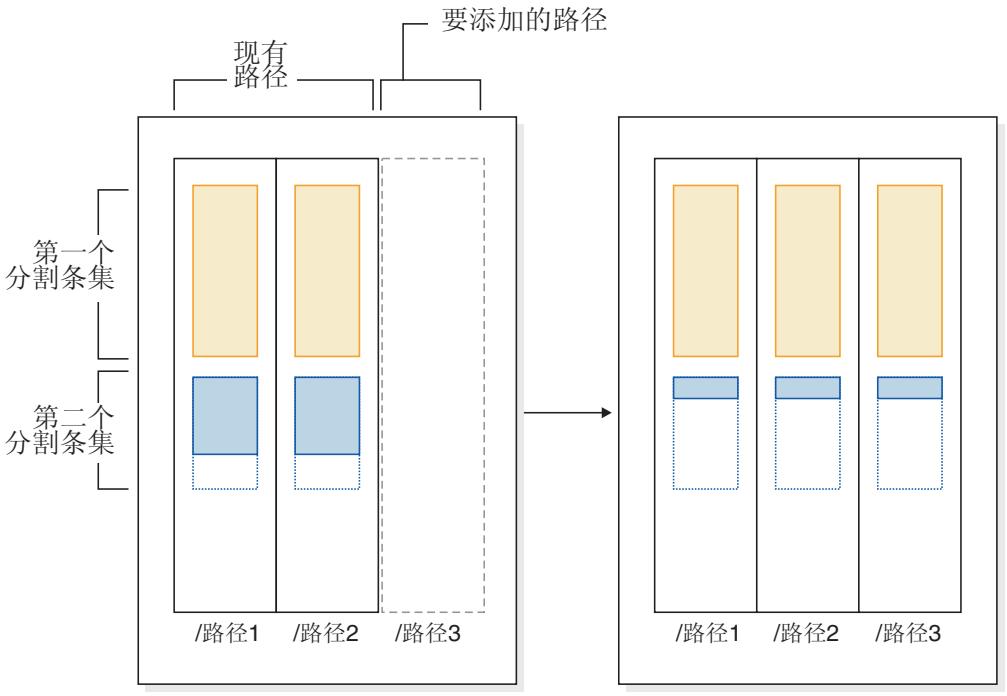


图 27. 添加存储器路径并对自动存储器表空间进行重新平衡

注: 本主题中提供的插图仅供参考。这些插图并未提供有关存储器布局的特定方法或最佳实践的建议。另外，这些插图仅说明单一表空间的情况；在实践中，可能有多个共享同一个存储器路径的自动存储器表空间。

当现有表空间有多个分割集，并且每个分割集包含不同数目的容器时，可能会发生类似的情况。发生这种情况的原因是，在表空间的生存期内，一个或多个存储器路径中的容器进入*磁盘满*状态。在这种情况下，数据库管理器将容器添加到那些现有存储器路径以填充分割集中的“空洞”将十分有益（当然，假定现在有空闲空间可完成此操作）。REBALANCE 操作也可用于执行此操作。

第 200 页的图 28 提供了正在进行重新平衡的表空间的分割集中存在“空洞”（例如，这可能是由于删除表行所致）的示例，并说明了存储器路径在重新平衡前后的布局。

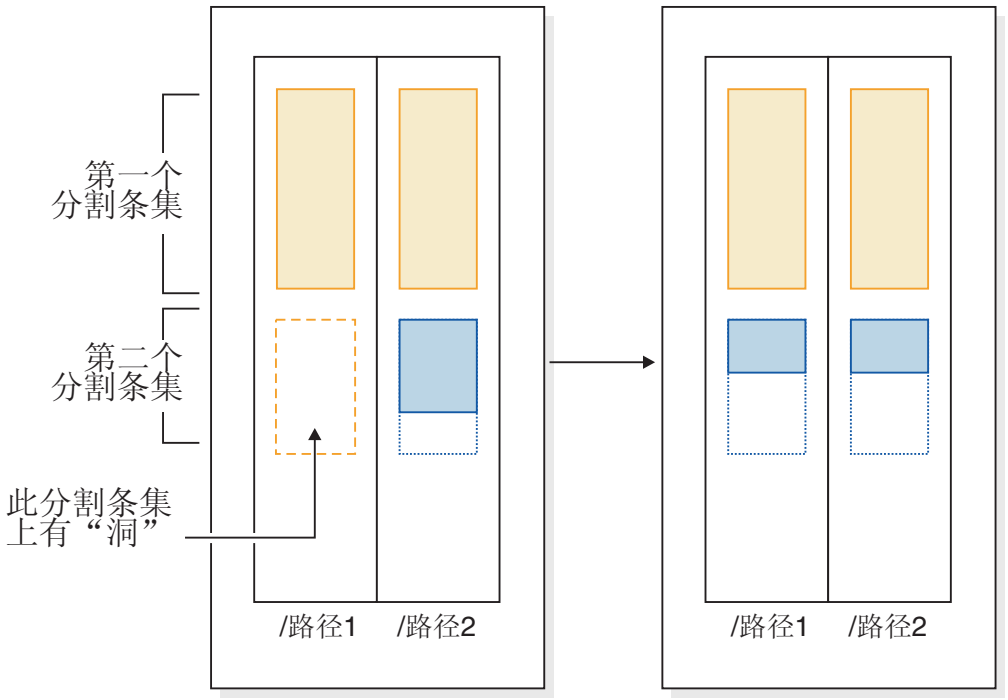


图 28. 重新平衡自动存储器表空间将填充间隔

示例

您已创建具有两个存储器路径的存储器组:

```
CREATE STOGROUP sg ON '/path1', '/path2'
```

在创建数据库之后，系统随后会在此存储器组中创建自动存储器表空间。

您决定向存储器组添加另一个存储器路径 (/path3)，并且要让所有自动存储器表空间使用新的存储器路径。

1. 第一步是向存储器组添加存储器路径:

   ```
   ALTER STOGROUP sg ADD '/path3'
   ```

2. 下一步是确定所有受影响的永久表空间。这可以通过以手动方式扫描表空间快照输出来完成，也可以通过 SQL 完成。以下 SQL 语句将生成存储器组中所有常规自动存储器表空间和大型自动存储器表空间的列表:

   ```
   SELECT TBSP_NAME
      FROM table (MON_GET_TABLESPACE(' ', -2))
      WHERE TBSP_USING_AUTO_STORAGE = 1
         AND TBSP_CONTENT_TYPE IN ('ANY','LARGE')
         AND STORAGE_GROUP_NAME = 'sg'
      ORDER BY TBSP_ID
   ```

3. 确定表空间之后，下一步是对列示的每个表空间执行以下语句。如果其余存储器路径中有足够的空间，那么重新平衡操作的执行顺序通常不重要（并且，它们可以并行地运行）。

   ```
   ALTER TABLESPACE tablespace_name REBALANCE
   ```

在此之后，必须确定如何处理临时表空间。其中一个选项是，先停止（停用）数据库，然后再将其启动（激活）。这将导致重新定义容器。此外，可以删除并重新创建临时表空间，也可以先创建新的临时表空间，然后再删除旧的临时表空间 - 这样，您就

不会试图删除数据库中的最后一个临时表空间，这是不允许的操作。要确定受影响的表空间的列表，您可以手动地扫描表空间快照输出或执行 SQL 语句。以下 SQL 语句将生成数据库中所有系统临时自动存储器表空间和用户临时自动存储器表空间的列表:

```
SELECT TBSP_NAME
   FROM table (MON_GET_TABLESPACE(' ', -2))
   WHERE TBSP_USING_AUTO_STORAGE = 1
      AND TBSP_CONTENT_TYPE IN ('USRTEMP','SYSTEMP')
      AND STORAGE_GROUP_NAME = 'sg'
   ORDER BY TBSP_ID
```

方案: 删除存储器路径并对自动存储器表空间进行重新平衡:

本方案说明如何删除存储器路径以及 REBALANCE 操作如何从使用路径的表空间中删除容器。

必须先除去存储器路径中的表空间容器，然后才能完成删除该路径的操作。如果不再需要整个表空间，那么可以先将其删除，然后再从存储器组中删除存储器路径。在这种情况下，不需要进行重新平衡。但是，如果您想保留该表空间，那么需要执行 REBALANCE 操作。在这种情况下，如果存在处于“暂挂状态”状态的存储器路径，那么数据库管理器将执行*反向重新平衡*，即，从高水位标记扩展数据块（表空间中最后一个有可能包含数据的扩展数据块）开始移动扩展数据块，并以扩展数据块 0 结束。

当运行 REBALANCE 操作时:

- 执行反向重新平衡。将任何处于“删除暂挂”状态的容器中的数据移入其余容器。
- 删除处于“删除暂挂”状态的容器。
- 如果当前表空间是最后一个使用该存储器路径的表空间，那么还将删除该存储器路径。

如果其余存储器路径中的容器不够大，无法容纳要移动的所有数据，那么数据库管理器可能必须先在其余存储器路径中创建或扩展容器，然后再执行重新平衡。

第 202 页的图 29 提供了正被删除的存储器路径的一个示例，并提供了存储器路径在表空间重新平衡前后的布局:

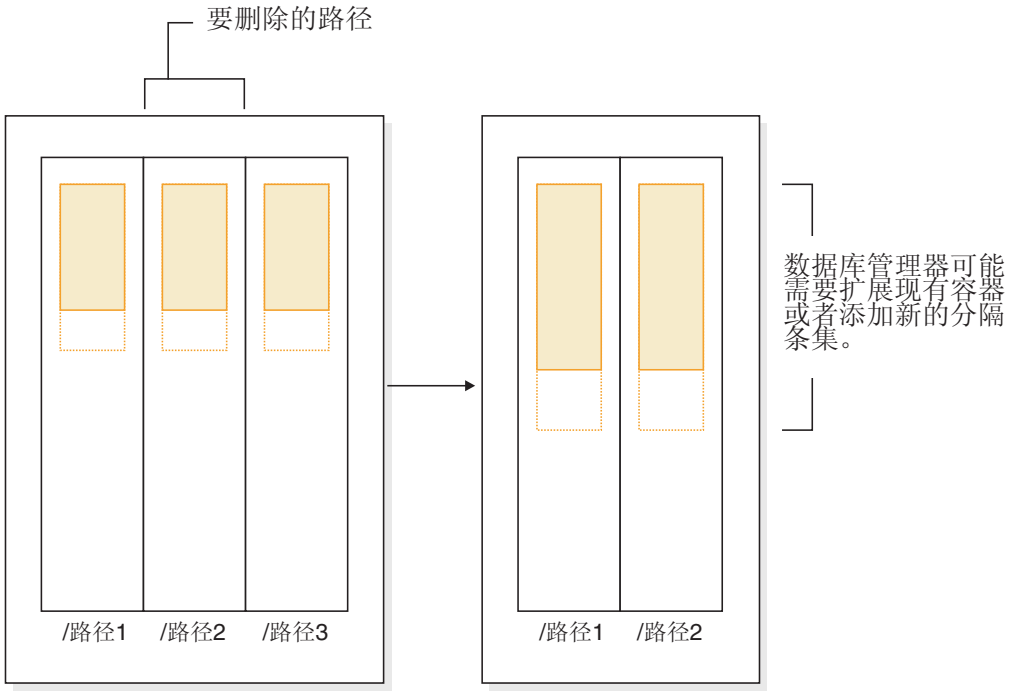


图 29. 删除存储器路径并对自动存储器表空间进行重新平衡

示例

创建具有三个存储器路径的存储器组:

```
CREATE STOGROUP sg ON '/path1', '/path2', '/path3'
```

创建该存储器组后，系统会使用该存储器组创建自动存储器表空间。

您想要通过从存储器组中删除 /path3 存储器路径以将其置于“删除暂挂”状态，然后对所有使用此存储器路径的表空间进行重新平衡以便将其删除。

1. 第一步是从存储器组中删除该存储器路径:

   ```
   ALTER STOGROUP sg DROP '/path3'
   ```

2. 下一步是确定所有受影响的非临时表空间。以下 SQL 语句将生成数据库中所有常规自动存储器表空间和大型自动存储器表空间的列表，这些表空间都带有驻留在处于“删除暂挂”状态的路径中的容器:

   ```
   SELECT TBSP_NAME
      FROM table (MON_GET_TABLESPACE(' ', -2))
      WHERE TBSP_USING_AUTO_STORAGE = 1
         AND TBSP_CONTENT_TYPE IN ('ANY','LARGE')
         AND STORAGE_GROUP_NAME = 'sg'
      ORDER BY TBSP_ID
   ```

3. 确定表空间之后，下一步是对列示的每个表空间执行以下语句:

   ```
   ALTER TABLESPACE <tablespace_name> REBALANCE
   ```

   a. 如果您已从存储器组中删除多个存储器路径，并且要释放特定路径中的存储器，那么可以查询该存储器组中的容器列表，以查找该存储器路径上存在的容器。例如，考虑名为 /path3 的路径。下列查询提供其容器驻留在路径 /path3 上的表空间的列表:

```
SELECT TBSP_NAME FROM SYSIBMADM.SNAPCONTAINER
  WHERE CONTAINER_NAME LIKE '/path3'
  GROUP BY TBSP_NAME;
```

b. 然后，可以对结果集中的每个表空间发出 REBALANCE 语句。

4. 要确定受影响表空间的列表，请生成所删除存储器路径上的所有系统临时自动存储器表空间和用户临时自动存储器表空间的列表:

```
SELECT TBSP_NAME
   FROM table (MON_GET_TABLESPACE(' ', -2))
   WHERE TBSP_USING_AUTO_STORAGE = 1
      AND TBSP_CONTENT_TYPE IN ('USRTEMP','SYSTEMP')
      AND STORAGE_GROUP_NAME = 'sg'
   ORDER BY TBSP_ID
```

方案: 添加和除去存储器路径并对自动存储器表空间进行重新平衡:

本方案说明如何同时添加和除去存储器路径以及 REBALANCE 操作如何对所有自动存储器表空间进行重新平衡。

可同时在存储器组中添加和删除存储器。可使用单个 ALTER STOGROUP 语句或通过按某个时间段（未在其间重新平衡表空间）分隔的多个 ALTER STOGROUP 语句来完成此操作。

如第 198 页的『方案: 添加存储器路径并对自动存储器表空间进行重新平衡』所述，在特定情况下，数据库管理器在删除存储器路径时会填充分割集中的“空洞”。在这种情况下，数据库管理器将在此过程中创建容器和删除容器。在所有这些情况下，数据库管理器都将认识到需要添加某些容器（如果空闲空间允许的话）以及需要除去某些容器。在这些情况下，数据库管理器可能需要执行双程重新平衡操作（快照监视器输出将对此操作的阶段和状态进行描述）:

1. 首先，在新路径中分配新容器（如果正在填充“空洞”，那么将在现有路径中进行分配）。
2. 执行正向重新平衡。
3. 执行反向重新平衡，从而从路径中正被删除的容器中移走数据。
4. 以物理方式删除容器。

第 204 页的图 30 提供了正在添加和删除的存储器路径的示例，并且描述了表空间在重新平衡前后的布局:

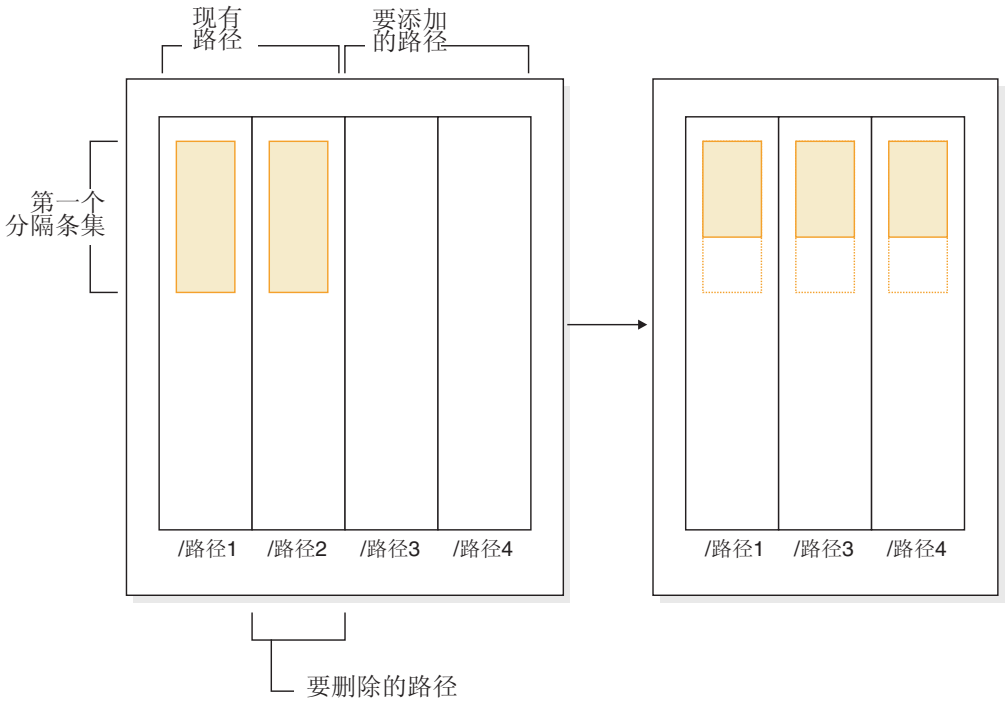


图 30. 添加和删除存储器路径，然后对自动存储器表空间进行重新平衡

示例

创建具有两个存储器路径的存储器组:

```
CREATE STOGROUP sg ON '/path1', '/path2', '/path4'
```

假定您要向存储器组添加另一个存储器路径 (/path3)，除去其中一个现有路径 (/path2) 并且还想重新平衡所有自动存储器表空间。第一步是将新存储器路径 /path3 添加至存储器组并除去 /path2:

```
ALTER STOGROUP sg ADD '/path3'
ALTER STOGROUP sg DROP '/path2'
```

下一步是确定所有受影响的表空间。此分析可通过以手动方式扫描表空间快照输出来完成，也可以通过 SQL 语句完成。以下 SQL 语句将生成数据库中所有常规自动存储器表空间和大型自动存储器表空间的列表:

```
SELECT TBSP_NAME
   FROM table (MON_GET_TABLESPACE(' ', -2))
   WHERE TBSP_USING_AUTO_STORAGE = 1
      AND TBSP_CONTENT_TYPE IN ('ANY','LARGE')
      AND STORAGE_GROUP_NAME = 'sg'
   ORDER BY TBSP_ID
```

标识表空间之后，下一步是对列示的每个表空间执行以下语句:

```
ALTER TABLESPACE tablespace_name REBALANCE
```

其中 *tablespace_name* 是上一步中标识的表空间的名称。

注: 不能重新平衡自动存储器管理的临时表空间。如果要停止使用已被分配给临时表空间的存储器，那么一个选项是删除临时表空间，然后重新创建这些表空间。

使表空间与存储器组相关联

通过使用 CREATE TABLESPACE 语句或 ALTER TABLESPACE 语句，可指定或更改表空间使用的存储器组。如果创建表空间时未指定存储器组，那么会使用缺省存储器组。

关于此任务

如果更改表空间使用的存储器组，那么落实 ALTER TABLESPACE 语句时会发出隐式 REBALANCE 操作。这会将数据从源存储器组移至目标存储器组。

使用 IBM DB2 pureScale Feature时，REBALANCE 不受支持，并且您不能更改所指定存储器组。REBALANCE 操作异步进行，并且不会影响数据的可用性。可使用监视表函数 MON_GET_REBALANCE_STATUS 来监视 REBALANCE 操作的进度。

在 ALTER TABLESPACE 操作期间，基于旧表空间属性的已编译对象会*软失效*。ALTER TABLESPACE 落实后进行的任何新编译使用 ALTER TABLESPACE 语句中指定的新表空间属性。*软失效*支持仅限于动态 SQL，您必须手动检测并重新编译所有静态 SQL 依赖项，才能使用新值。

使用同一存储器组的所有表空间可有不同的 PAGESIZE 和 EXTENTSIZE 值。这些属性与表空间定义相关，与存储器组无关。

过程

要将表空间与存储器组相关联，请发出以下语句:

```
CREATE TABLESPACE tbspc USING STOGROUP storage_group
```

其中 *tbspc* 是新表空间，*storage_group* 是关联存储器组。

方案: 将表空间移至新存储器组

此方案显示如何将表空间从一个存储器组移至另一个存储器组。

此方案中的假定是表空间数据在存储器组内存储器路径上的容器中。ALTER TABLESPACE 语句用于将表空间数据移至新存储器组。

表空间移至新存储器组后，旧存储器组中的容器会标记为删除暂挂。落实 ALTER TABLESPACE 语句后，容器分配到新存储器组的存储器路径上，驻留在旧存储器组中的现有容器标记为删除暂挂，并且会启动隐式 REBALANCE 操作。此操作将容器分配到新存储器路径上，并将现有容器中的数据重新平衡到新容器中。要创建的容器的数目和大小取决于目标存储器组中的存储器路径数以及新存储器路径上的可用空间量。移动所有数据后，会删除旧容器。

下图是将表空间从一个存储器组移至另一个存储器组的示例，其中:

1. 新容器分配到目标存储器组的存储器路径上。
2. 所有原始容器被标记为删除暂挂，新分配请求通过新容器得到满足。
3. 执行了反向重新平衡，从路径上的容器移出的数据被删除。
4. 以物理方式删除容器。

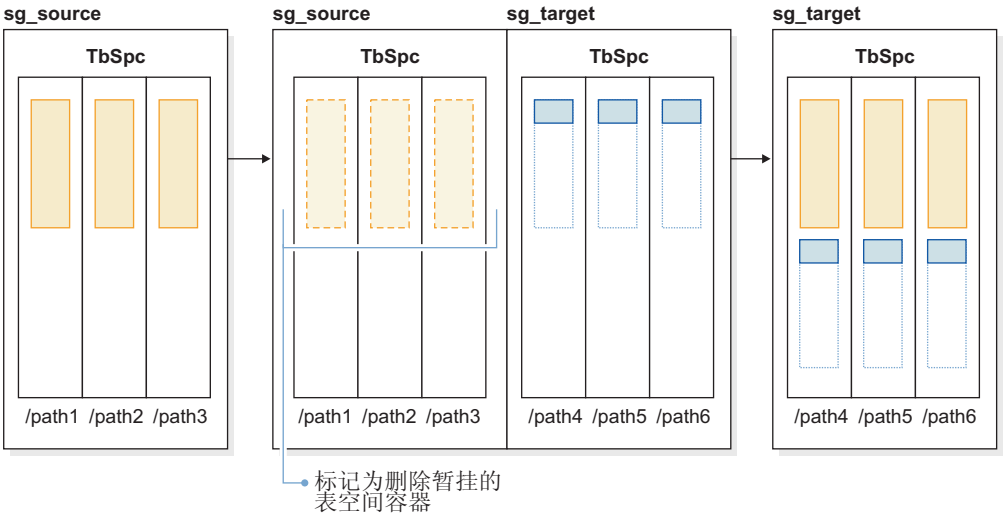


图 31. 将表空间移至新存储器组

要将表空间移至另一存储器组，请执行以下操作:

1. 创建两个存储器组: sg_source 和 sg_target:

```
CREATE STOGROUP sg_source ON '/path1', '/path2', '/path3'
CREATE STOGROUP sg_target ON '/path4', '/path5', '/path6'
```

2. 创建数据库后，创建最初使用该 sg_source 存储器组的自动存储器表空间:

```
CREATE TABLESPACE TbSpc USING STOGROUP sg_source
```

3. 将自动存储器表空间移至 sg_target 存储器组:

```
ALTER TABLESPACE TbSpc USING sg_target
```

重命名表空间

使用 RENAME TABLESPACE 语句来重命名表空间。

关于此任务

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。

表空间状态

本主题提供有关受支持表空间状态的信息。

IBM DB2 数据库产品当前至少支持 25 种表或表空间状态。在特定情况下，这些状态用于控制对数据的访问权，或者根据需要触发特定的用户操作以保护数据库的完整性。它们中的大部分来自与某个 DB2 数据库实用程序（例如，装入实用程序或者备份

和复原实用程序）的操作有关的事件。 下表描述了各种受支持的表空间状态。另外，该表还提供了一些工作示例以说明如何对管理数据库时可能遇到的各种状态作出解释和响应。这些示例均摘自曾在 AIX 上运行的命令脚本；您可以单独复制、粘贴并运行这些示例。如果您是在非 UNIX 系统上运行 DB2 数据库产品，请确保所有路径名均采用适合于您的系统的正确格式。其中的大多数示例均基于随 DB2 数据库产品附带的 SAMPLE 数据库中的表。少数示例需要某些不属于 SAMPLE 数据库的方案，但您可以使用与 SAMPLE 数据库的连接作为起始点。

表 17. 受支持的表空间状态

State	十六进制状态值	描述

备份暂挂	0x20	在执行时间点表空间前滚操作后，或者在对可恢复的数据库执行指定了 **COPY NO** 选项的装入操作后，表空间将处于此状态。必须备份表空间（或者整个数据库），然后才能使用该表空间。如果未备份表空间，那么可以对该表空间中的表执行查询，但不能进行更新。
		注：另外，在启用数据库以执行前滚恢复操作后，还必须立即备份该数据库。如果 **logarchmeth1** 数据库配置参数设为除 OFF 以外的任何值，那么数据库是可恢复的。在备份数据库前，不能激活或连接至此类数据库。完成数据库备份时，**backup_pending** 信息数据库配置参数的值将设为 NO。**示例**
		
		假定 staff_data.del 输入文件包含下列内容:
		`11,"Melnyk",20,"Sales",10,70000,15000:`
		
		在指定 copy no 的情况下将此数据装入 staff 表，如下所示:
		`update db cfg for sample using logarchmeth1 logretain;`
		`backup db sample;`
		`connect to sample;`
		`load from staff_data.del of del messages load.msg insert into staff copy no;`
		`update staff set salary = 69000 where id = 11;`
		`list tablespaces;`
		`connect reset;`
		
		返回的 USERSPACE1 信息显示此表空间处于“备份暂挂”状态。

正在备份	0x800	这是一种瞬态状态，仅在执行备份操作期间有效。
		
		示例
		
		执行如下所示的联机备份:
		`backup db sample online;`
		
		运行备份操作时，从其他会话执行下列其中一个脚本:
		• `connect to sample;`
		`list tablespaces show detail;`
		`connect reset;`
		• `connect to sample;`
		`get snapshot for tablespaces on sample;`
		`connect reset;`
		
		返回的 USERSPACE1 信息显示此表空间处于“正在备份”状态。

表 17. 受支持的表空间状态 (续)

State	十六进制状态值	描述

正在进行 DMS 重新平衡	0x10000000	这是一种瞬态状态，仅在执行数据重新平衡操作期间有效。如果将新容器添加至一个定义为数据库管理的空间 (DMS) 的表空间，或者扩展了现有容器，那么可能会对表空间数据进行重新平衡。*重新平衡*是将表空间扩展数据块从一个位置移到另一个位置以尝试保持数据条带化的过程。*扩展数据块*是容器空间的单位（以页计），而条带是*跨表空间的容器集*的扩展数据块层。
		
		示例
		
		假定 staffdata.del 输入文件包含 20000 个或更多的记录，创建 newstaff 表，使用此输入文件装入该表，然后将新容器添加至 ts1 表空间:
		`connect to sample;`
		`create tablespace ts1 managed by database using`
		`(file '/home/melnyk/melnyk/NODE0000/SQL00001/ts1c1' 1024);`
		`create table newstaff like staff in ts1;`
		`load from staffdata.del of del insert into newstaff nonrecoverable;`
		`alter tablespace ts1 add`
		`(file '/home/melnyk/melnyk/NODE0000/SQL00001/ts1c2' 1024);`
		`list tablespaces;`
		`connect reset;`
		
		返回的 TS1 信息显示此表空间处于“正在进行 DMS 重新平衡”状态。

禁用暂挂	0x200	在执行数据库前滚操作期间，表空间可能处于此状态，但在前滚操作结束时，表空间不应该处于此状态。表空间脱机及事务的补偿日志记录未写入所导致的情况将触发此状态。此表空间状态的出现和后续消失对于用户是透明的。 对此表空间状态的说明不在本文档的范围内，因此未给出示例。
删除暂挂	0x8000	在执行数据库重新启动操作期间，如果发现表空间的一个或多个容器出现问题，那么该表空间将处于此状态。如果数据库的前一个会话异常终止（例如，在电源发生故障期间异常终止），那么必须重新启动此数据库。如果表空间处于“删除暂挂”状态，那么表空间将不可用并且只能删除。 对此表空间状态的说明不在本文档的范围内，因此未给出示例。

正在装入	0x20000	这是一种瞬态状态，仅在对可恢复数据库执行指定了 **COPY NO** 选项的装入操作期间有效。另请参阅“正在装入”表状态。
		
		示例
		
		假定 staffdata.del 输入文件包含 20000 个或更多的记录，创建 newstaff 表，然后在指定 **COPY NO** 和此输入文件的情况下装入该表:
		`update db cfg for sample using logarchmeth1 logretain;`
		`backup db sample;`
		`connect to sample;`
		`create table newstaff like staff;`
		`load from staffdata.del of del insert into newstaff copy no;`
		`connect reset;`
		
		运行装入操作时，从其他会话，通过执行“正在备份”示例中显示的其中一个样本脚本来获取有关表空间的信息。
		
		返回的 USERSPACE1 信息显示此表空间处于“正在装入”状态（及“备份暂挂”）状态。

表 17. 受支持的表空间状态 (续)

State	十六进制状态值	描述

正常	0x0	如果表空间未处于任何其他（异常）表空间状态，那么该表空间处于“正常”状态。“正常”状态是创建表空间后该表空间的初始状态。
		
		示例
		
		创建表空间，然后获取有关该表空间的信息，如下所示:
		
		`connect to sample;`
		`create tablespace ts1 managed by automatic storage;`
		`list tablespaces show detail;`
		
		返回的 USERSPACE1 信息显示此表空间处于“正常”状态。

脱机且不可访问	0x4000	如果表空间的一个或多个容器出现问题，那么该表空间将处于此状态。容器可能意外重命名、移动或损坏。修正问题并且与该表空间相关联的容器可再次访问后，可通过从数据库断开所有应用程序的连接，然后重新连接至该数据库，来除去此异常状态。另外，还可以发出 ALTER TABLESPACE 语句并指定 SWITCH ONLINE 子句，以从表空间中除去“脱机且不可访问”状态，而不必从数据库断开其他应用程序的连接。
		
		示例
		
		创建包含 tsc1 和 tsc2 容器的 ts1 表空间，创建 staffemp 表，然后从 `st_data.del` 文件中导入数据，如下所示:
		
		`connect to sample;`
		`create tablespace ts1 managed by database using`
		`(file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc1' 1024);`
		`alter tablespace ts1 add`
		`(file '/home/melnyk/melnyk/NODE0000/SQL00001/tsc2' 1024);`
		`export to st_data.del of del select * from staff;`
		`create table stafftemp like staff in ts1;`
		`import from st_data.del of del insert into stafftemp;`
		`connect reset;`
		
		将表空间容器 tsc1 重命名为 tsc3，然后尝试查询 STAFFTEMP 表:
		
		`connect to sample;`
		`select * from stafftemp;`
		
		该查询会返回 SQL0290N（不允许进行表空间访问），并且 **LIST TABLESPACES** 命令会返回 TS1 的状态值 0x4000（脱机且不可访问）。将表空间容器 tsc3 重命名为 tsc1。此时，查询可成功运行。

表 17. 受支持的表空间状态 (续)

State	十六进制状态值	描述

停顿互斥	0x4	对表空间停顿函数进行调用的应用程序对表空间具有互斥（读或写）访问权时，该表空间将处于此状态。使用 **QUIESCE TABLESPACES FOR TABLE** 命令，将表空间显式设为“停顿互斥”。
		
		示例
		
		将表空间设为“正常”，然后再将表空间设为“停顿互斥”，如下所示:
		`connect to sample;`
		`quiesce tablespaces for table staff reset;`
		`quiesce tablespaces for table staff exclusive;`
		`connect reset;`
		
		从其他会话，执行下列脚本:
		`connect to sample;`
		`select * from staff where id=60;`
		`update staff set salary=50000 where id=60;`
		`list tablespaces;`
		`connect reset;`
		
		返回的 USERSPACE1 信息显示此表空间处于“停顿互斥”状态。

停顿共享	0x1	对表空间停顿函数进行调用的应用程序及并行应用程序对表空间都具有读访问权（但没有写访问权）时，该表空间将处于此状态。使用 **QUIESCE TABLESPACES FOR TABLE** 命令，将表空间显式设为“停顿共享”。
		
		示例
		
		将表空间设为“正常”，然后再将表空间设为“停顿共享”，如下所示:
		`connect to sample;`
		`quiesce tablespaces for table staff reset;`
		`quiesce tablespaces for table staff share;`
		`connect reset;`
		
		从其他会话，执行下列脚本:
		`connect to sample;`
		`select * from staff where id=40;`
		`update staff set salary=50000 where id=40;`
		`list tablespaces;`
		`connect reset;`
		
		返回的 USERSPACE1 信息显示此表空间处于“停顿共享”状态。

表 17. 受支持的表空间状态 (续)

State	十六进制状态值	描述

停顿更新	0x2	对表空间停顿函数进行调用的应用程序对表空间具有互斥写访问权时，该表空间将处于此状态。使用 **QUIESCE TABLESPACES FOR TABLE** 命令，将表空间显式设为“停顿更新”状态。
		
		示例
		
		将表空间设为“正常”，然后再将表空间设为“停顿更新”，如下所示:
		`connect to sample;`
		`quiesce tablespaces for table staff reset;`
		`quiesce tablespaces for table staff intent to update;`
		`connect reset;`
		
		从其他会话，执行下列脚本:
		`connect to sample;`
		`select * from staff where id=50;`
		`update staff set salary=50000 where id=50;`
		`list tablespaces;`
		`connect reset;`
		
		返回的 USERSPACE1 信息显示此表空间处于“停顿更新”状态。

正在重组	0x400	这是一种瞬态状态，仅在执行重组操作期间有效。
		
		示例
		
		重组 staff 表，如下所示:
		`connect to sample;`
		`reorg table staff;`
		`connect reset;`
		
		运行重组操作时，从其他会话，通过执行“正在备份”示例中显示的其中一个样本脚本来获取有关表空间的信息。
		
		返回的 USERSPACE1 信息显示此表空间处于“正在重组”状态。
		注: 涉及到 SAMPLE 数据库的“重组表”操作可能在很短时间内完成，因此，使用此方法可能很难察看到“重组进度”状态。

复原暂挂	0x100	在“重定向复原”操作的第一部分之后（即，在发出 **SET TABLESPACE CONTAINERS** 命令之前），数据库的表空间将处于此状态。必须复原表空间（或整个数据库），然后才能使用该表空间。在成功完成复原操作前，不能连接至数据库。在成功完成复原操作时，**restore_pending** 信息数据库配置参数的值将设为 NO。 **示例** 在“可定义存储器”状态下完成“重定向复原”操作的第一部分时，所有表空间都将处于“复原暂挂”状态。

表 17. 受支持的表空间状态 (续)

State	十六进制状态值	描述

正在复原	0x2000	这是一种瞬态状态，仅在执行复原操作期间有效。
		
		示例
		
		对样本数据库启用前滚恢复功能，然后备份样本数据库和 USERSPACE1 表空间，如下所示:
		`update db cfg for sample using logarchmeth1 logretain;`
		`backup db sample;`
		`backup db sample tablespace (userspace1);`
		
		假定此备份映像的时间戳记是 20040611174124，复原 USERSPACE1 表空间备份:
		`restore db sample tablespace (userspace1) online taken at 20040611174124;`
		
		运行复原操作时，从其他会话，通过执行“正在备份”示例中显示的其中一个样本脚本来获取有关表空间的信息。
		
		返回的 USERSPACE1 信息显示此表空间处于“正在复原”状态。

前滚暂挂	0x80	对可恢复数据库执行复原操作后，表空间将处于此状态。必须前滚表空间（或整个数据库），然后才能使用该表空间。如果 **logarchmeth1** 数据库配置参数设为除 OFF 以外的任何值，那么数据库是可恢复的。在成功完成前滚操作前，不能激活或连接至数据库。在成功完成前滚操作时，**rollfwd_pending** 信息数据库配置参数的值将设为 NO。 **示例** 在“正在复原”状态下完成联机表空间复原操作时，表空间 USERSPACE1 将处于“前滚暂挂”状态。

正在前滚	0x40	这是一种瞬态状态，仅在执行前滚操作期间有效。
		
		示例
		
		假定 staffdata.del 输入文件包含 20000 个或更多的记录，创建表和表空间，然后进行数据库备份:
		`update db cfg for sample using logarchmeth1 logretain;`
		`backup db sample;`
		`connect to sample;`
		`create tablespace ts1 managed by automatic storage;`
		`create table newstaff like staff in ts1;`
		`connect reset;`
		`backup db sample tablespace (ts1) online;`
		
		假定备份映像的时间戳记是 20040630000715，复原数据库备份并前滚至日志末尾，如下所示:
		`connect to sample;`
		`load from staffdata.del of del insert into newstaff copy yes`
		`to /home/melnyk/backups;`
		`connect reset;`
		`restore db sample tablespace (ts1) online taken at 20040630000715;`
		`rollforward db sample to end of logs and stop tablespace (ts1) online;`
		
		运行前滚操作时，从其他会话，通过执行“正在备份”示例中显示的其中一个样本脚本来获取有关表空间的信息。
		
		返回的 TS1 信息显示此表空间处于“正在前滚”状态。

表 17. 受支持的表空间状态 (续)

State	十六进制状态值	描述

可定义存储器	0x2000000	在“重定向复原”操作的第一部分之后（即，在发出 **SET TABLESPACE CONTAINERS** 命令之前），数据库的表空间将处于此状态。这允许您重定义该容器。
		
		示例
		
		假定备份映像的时间戳记是 20040613204955，复原数据库备份，如下所示:
		
		`restore db sample taken at 20040613204955 redirect;`
		`list tablespaces;`
		
		LIST TABLESPACES 命令返回的信息显示所有表空间都处于“可定义存储器”和“复原暂挂”状态。

必须定义存储器	0x1000	如果在执行“重定向到新数据库复原”操作期间省略了“设置表空间容器”阶段，或者如果在“设置表空间容器”阶段无法获取指定的容器，那么数据库的表空间将处于此状态。在“设置表空间容器”阶段无法获取指定容器的原因可能是指定的路径名无效或磁盘空间不足等。
		
		示例
		
		假定备份映像的时间戳记是 20040613204955，复原数据库备份，如下所示:
		
		`restore db sample taken at 20040613204955 into mydb redirect;`
		`set tablespace containers for 2 using (path 'ts2c1');`
		`list tablespaces;`
		
		LIST TABLESPACES 命令返回的信息显示表空间 SYSCATSPACE 和表空间 TEMPSPACE1 处于“必须定义存储器”、“可定义存储器”和“复原暂挂”状态。“必须定义存储器”状态优先于“可定义存储器”状态。

暂挂写入	0x10000	写入操作暂挂后，表空间将处于此状态。 对此表空间状态的说明不在本文档的范围内，因此未给出示例。

正在创建表空间	0x40000000	这是一种瞬态状态，仅在执行“创建表空间”操作期间有效。
		
		示例
		
		创建 ts1、ts2 和 ts3 表空间，如下所示:
		
		`connect to sample;`
		`create tablespace ts1 managed by automatic storage;`
		`create tablespace ts2 managed by automatic storage;`
		`create tablespace ts3 managed by automatic storage;`
		
		运行“创建表空间”操作时，从其他会话，通过执行“正在备份”示例中显示的其中一个样本脚本来获取有关表空间的信息。
		
		返回的 TS1、TS2 和 TS3 信息说明这些表空间处于“正在创建表空间”状态。

表 17. 受支持的表空间状态 (续)

State	十六进制状态值	描述

正在删除表空间	0x20000000	这是一种瞬态状态，仅在执行“删除表空间”操作期间有效。
		
		示例
		
		创建 ts1、ts2 和 ts3 表空间，然后将这些表空间删除，如下所示:
		
		`connect to sample;`
		`create tablespace ts1 managed by automatic storage;`
		`create tablespace ts2 managed by automatic storage;`
		`create tablespace ts3 managed by automatic storage;`
		`drop tablespaces ts1, ts2, ts3;`
		
		运行删除表空间操作时，从其他会话，通过执行“正在备份”示例中显示的其中一个样本脚本来获取有关表空间的信息。
		
		返回的 TS1、TS2 和 TS3 信息说明这些表空间处于“正在删除表空间”状态。

存储器组和表空间介质属性

自动存储器表空间从该表空间缺省情况下使用的存储器组继承介质属性值、设备读速率和数据标记属性。

使用 CREATE STOGROUP 语句创建存储器组时，可指定以下存储器组属性:

OVERHEAD
此属性指定 I/O 控制器时间、磁盘搜索和等待时间（以毫秒计）。

DEVICE READ RATE
此属性指定设备的读取传输速率（以兆字节/秒计）规范。此值用于确定查询优化期间的 I/O 成本。如果此值对于所有存储器路径不同，那么该数字应是属于该存储器组的所有存储器路径的平均值。

DATA TAG
此属性指定特定存储器组中的数据的标记，WLM 可使用此标记来确定数据库活动的处理优先级。

存储器组属性的缺省值如下所示:

表 18. 存储器组属性的缺省设置

属性	缺省设置
DATA TAG	NONE
DEVICE READ RATE	100 MB/sec
OVERHEAD	6.725 ms

创建自动存储器表空间时，可指定用于标识该表空间中包含的数据的标记。如果该表空间与存储器组相关联，那么表空间的数据标记属性将覆盖可能对该存储器组设置的任何数据标记属性。如果用户未对表空间指定数据标记属性，并且该表空间包含在存储器组中，那么该表空间将从该存储器组继承数据标记值。可对除目录表空间以外的任何常规表空间或大型表空间设置数据标记属性 (SQL0109N)。不能对临时表空间设置数据标记属性，否则将返回 SQL0109N 消息错误。

自动存储器表空间从它使用的存储器组继承开销和传输速率属性。如果表空间从它使用的存储器组继承传输速率属性，那么该存储器组的设备读速率将从“毫秒/所读页”进行转换（考虑表空间的页大小设置），如下所示:

```
TRANSFERRATE = ( 1 / DEVICE READ RATE ) * 1000 / 1024000 * PAGESIZE
```

自动存储器表空间和非自动表空间的页大小设置具有对应的缺省 TRANSFERRATE 值:

表 19. 缺省 TRANSFERRATE 值

PAGESIZE	TRANSFERRATE
4 KB	0.04 毫秒/所读页
8 KB	0.08 毫秒/所读页
16 KB	0.16 毫秒/所读页
32 KB	0.32 毫秒/所读页

自动存储器表空间的数据标记、设备读速率和开销介质属性可更改为从其关联存储器组动态继承值。为动态更新介质属性，请对 CREATE TABLESPACE 或 ALTER TABLESPACE 语句指定 INHERIT 选项。

表空间从存储器组继承属性值时，SYSCAT.TABLESPACES 目录表视图报告该属性的值为 -1。要查看开销、传输率和数据标记属性在运行时的实际值，可使用下列查询:

```
 select tbspace,
  cast(case when a.datatag = -1 then b.datatag else a.datatag end
as smallint) eff_datatag,
  eff_datatag,
  cast(case when a.overhead = -1 then b.overhead else a.overhead
end as double) eff_overhead,
  eff_overhead,
  cast(case when a.transferrate = -1 then (1 / b.devicereadrate) / 1024 * a.pagesize
else a.transferrate end as double) eff_transferrate
  eff_transferrate
   from syscat.tablespaces a left outer join syscat.stogroups b on a.sgid = b.sgid
```

如果升级至 V10.1，那么现有表空间将保留其开销和传输速率设置，并且该存储器组的开销和设备读速率属性设为 undefined（未定义）。存储器组中新创建的表空间（设备读速率设为未定义）使用最初创建 DB2 数据库时定义的数据库缺省值。如果存储器组的介质设置具有有效值，那么新创建的表空间将继承这些值。可使用 ALTER STOGROUP 语句来设置该存储器组的介质属性。对于非自动表空间，介质属性将保留。

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。

过程

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
```

或者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。

结果

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。

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。

关于此任务

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 146 GB 磁盘配置为三个 RAID-5 阵列，每个阵列包含 5 个磁盘。对于 RAID-5，一个磁盘的空间用于奇偶性校验信息，此信息分布在阵列中的所有磁盘上。硬盘驱动器制造商将 GB（千兆字节）定义为 1,000,000,000 个字节，但大多数操作系统使用基于 2 的表示法，这就解释了 146 GB 磁盘为何会显示为 136 GB 磁盘。在初始化 RAID-5 阵列之后，每个阵列都可以存储大约 544 GB [1]。如果表空间需要 300 GB 存储空间，那么创建三个容器并将每个容器放在不同的阵列上。每个容器使用设备上的 100 GB (300 GB/3)，并且每个设备上保留 444 GB (544 GB - 100 GB) 以提供附加表空间。

   注: [1] (146,000,000,000B/1024/1024/1024) = 135.97 GB，(5*136 GB - 1*136 GB) = 544 GB。取决于放置在阵列上的文件系统，可能需要使用其他空间来存储元数据及内部文件系统结构。

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 64 KB，而页大小为 16 KB，那么适当的扩展数据块大小可能是 256 KB。

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，请设置 DB2_PARALLEL_IO = *:5。

   如果将表空间的预取大小设为 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，可将预取大小设为 768 KB。此值等于 RAID 条带大小 (256 KB) 乘以 RAID 并行设备数 (3)。它也是扩展数据块大小 (256 KB) 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。

过程

- 在一组 RAID 设备上创建表空间时，应该为不同设备上的表空间（SMS 或 DMS）创建容器。

  考虑以下示例: 您将 15 个 146 GB 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。

- 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 64 KB，而页大小为 16 KB，那么适当的扩展数据块大小可能是 256 KB。

- 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，请设置 **DB2_PARALLEL_IO** = *:4。

  如果将表空间的预取大小设为 AUTOMATIC，则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，可将预取大小设为 768 KB。此值等于 RAID 条带大小 (256 KB) 乘以 RAID 并行设备数 (3)。它也是扩展数据块大小 (256 KB) 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)。

- 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。

关于此任务

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，那么必须在一个 DROP TABLESPACE 语句中同时删除索引和数据表空间。

过程

- 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。

  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或已创建临时表时，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或已创建临时表。

  注: 已声明临时表或已创建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）。

  1. 要创建系统临时表空间，请发出以下语句:

```
CREATE SYSTEM TEMPORARY TABLESPACE name
  MANAGED BY SYSTEM USING ('directories')
```

2. 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
```

3. 以下 SQL 语句创建一个称为 TEMPSPACE2 的系统临时表空间:

```
CREATE SYSTEM TEMPORARY TABLESPACE TEMPSPACE2
  MANAGED BY SYSTEM USING ('d:\systemp2')
```

4. 创建 TEMPSPACE2 后，可使用以下语句来删除原来的系统临时表空间 TEMPSPACE1:

```
DROP TABLESPACE TEMPSPACE1
```

第 9 章 存储器组

存储器组是可存储数据的存储器路径的指定集合。存储器组配置为表示可供数据库系统使用的不同存储器类。可对存储器组指定最适合于该数据的表空间。只有自动存储器表空间才使用存储器组。

一个表空间只能与一个存储器组相关联，但一个存储器组可有多个表空间关联。要管理存储器组对象，可使用 CREATE STOGROUP、ALTER STOGROUP、RENAME STOGROUP、DROP 和 COMMENT 语句。

通过表分区功能，可将表数据放在多个表空间中。通过使用此功能，存储器组可将表数据的一部分存储在快速存储器上，将表数据的余下部分存储在较慢存储器的一层或多层上。使用存储器组来支持多温度存储器，这会根据存储器类来确定数据的优先级。例如，可创建映射至数据库系统中的不同存储器层的存储器组。于是，所定义表空间与这些存储器组相关联。

定义存储器组时，应确保根据服务质量特征来对存储器路径分组。数据的常见*服务质量*特征遵循帐龄模式，其中最新数据经常被访问，并且要求最短访问时间 (*热数据*)，较旧数据较少被访问，并且可容忍较长访问时间（*温数据*或*冷数据*）。数据的优先级基于:

- 访问频率
- 可接受访问时间
- 数据的易变性
- 应用程序需求

通常，数据的优先级与数据量成反比，冷数据和温数据的量很大，热数据只有一小部分。可使用 DB2 工作负载管理器 (WLM) 来定义有关如何根据可指定给所访问数据的标记（通过表空间或存储器组的定义）来处理活动的规则。

使用多温度存储器来管理数据

可配置数据库以将经常访问的数据（*热数据*）存储在快速存储器上，不经常访问的数据（*温数据*）存储在速度稍慢的存储器上，极少访问的数据（*冷数据*）存储在速度很慢且比较便宜的存储器上。如果热数据变冷并且不经常访问，那么可通过动态方式将其移至速度较慢的存储器。

在数据库系统中，有一种强烈的趋势: 相对较小比例的数据是热数据，大部分数据是温数据或冷数据。如果要通过尽量不将冷数据存储在快速存储器上来优化快速存储器的使用，那么这些*多温度数据*集合会构成相当大的挑战。因为数据仓库使用的存储量日益增长，所以优化快速存储器的使用在管理存储器成本方面变得日益重要。

存储器组是具有相似质量的存储器路径组。 创建或改变存储器组时要考虑的一些低层存储器的关键属性包括可用存储器容量、等待时间、数据传输率和 RAID 保护度。可创建将映射至数据库管理系统中不同存储器类的存储器组。可根据哪些表空间具有热数据、温数据或冷数据以将自动存储器表空间指定给这些存储器组。要将数据库管理的表空间转换为使用自动存储器，必须发出 ALTER TABLESPACE 语句（指定 MANAGED BY AUTOMATIC STORAGE 选项）然后执行重新平衡操作。

因为当前数据通常被视为热数据，所以随时间迁移它通常会变为温数据，然后变为冷数据。可通过使用带 USING STOGROUP 选项的 ALTER TABLESPACE 语句以动态方式将表空间重新指定给另一存储器组。

以下示例说明如何将存储器组与多温度数据配合使用。假定您是公司的 DBA，负责对当前财政季度数据进行大部分处理。如图 32中所示，此公司有足够的固态硬盘 (SSD) 容量来保存整个季度的数据，并具有足够的基于光纤 (FC) 的和串行连接的 SCSI (SAS) 硬盘容量来保存该年度余下时间的数据。早于一年的数据存储在大容量串行 ATA (SATA) RAID 阵列上，该阵列虽然稳定，但运行速度不足以承受较重的查询工作负载。可定义 3 个存储器组：一个用于 SSD 存储器 (sg_hot)，一个用于 FC 和 SAS 存储器 (sg_warm)，一个用于 SATA 存储器 (sg_cold)。然后您执行以下操作：

- 将包含当前季度的数据的表空间指定给 sg_hot 存储器组
- 将包含前三个季度的数据的表空间指定给 sg_warm 存储器组
- 将包含所有更早数据的表空间指定给 sg_cold 存储器组

当前季度过去后，您执行以下操作：

- 将新季度的表空间指定给 sg_hot 存储器组
- 将刚过去的季度的表空间移至 sg_warm 存储器组
- 将 sg_warm 存储器组中最早季度的数据移至 sg_cold 存储器组

可在数据库联机时执行所有这些工作。

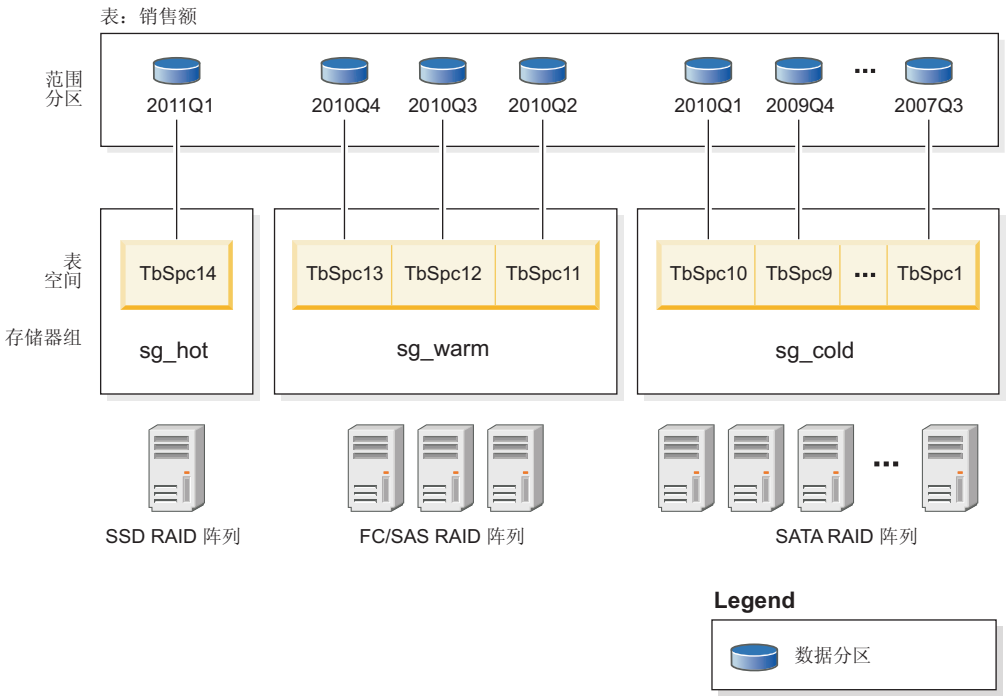


图 32. 使用多温度数据存储器来管理销售数据

以下步骤提供有关如何为当前财政年度的销售数据设置多温度数据存储器的更多详细信息：

1. 创建两个存储器组来反映两类存储器：一个存储器组用于存储热数据，一个存储器组用于存储温数据。

```
CREATE STOGROUP sg_hot ON '/ssd/path1', '/ssd/path2' DEVICE READ RATE 100
        OVERHEAD 6.725;
CREATE STOGROUP sg_warm ON '/hdd/path1', '/hdd/path2';
```

这些语句定义 SSD 存储器组 (sg_hot) 来存储热数据，并定义 FC 和 SAS 存储器组 (sg_warm) 来存储温数据。

2. 创建 4 个表空间（一个财政年度的每个季度数据对应一个表空间），并将这些表空间指定给存储器组。

```
CREATE TABLESPACE tbsp_2010q2 USING STOGROUP sg_warm;
CREATE TABLESPACE tbsp_2010q3 USING STOGROUP sg_warm;
CREATE TABLESPACE tbsp_2010q4 USING STOGROUP sg_warm;
CREATE TABLESPACE tbsp_2011q1 USING STOGROUP sg_hot;
```

此关联将产生继承存储器组属性的表空间。

3. 在 sales 表中设置范围分区。

```
CREATE TABLE sales (order_date DATE, order_id INT, cust_id BIGINT)
PARTITION BY RANGE (order_date)
(PART "2010Q2" STARTING ('2010-04-01') ENDING ('2010-06-30') in "tbsp_2010q2",
 PART "2010Q3" STARTING ('2010-07-01') ENDING ('2010-09-30') in "tbsp_2010q3",
 PART "2010Q4" STARTING ('2010-10-01') ENDING ('2010-12-31') in "tbsp_2010q4",
 PART "2011Q1" STARTING ('2011-01-01') ENDING ('2011-03-31') in "tbsp_2011q1");
```

2011Q1 数据表示当前财政季度并将使用 sg_hot 存储器组。

4. 当前季度过去后，为新季度创建表空间并将该表空间指定给 sg_hot 存储器组。

```
CREATE TABLESPACE tbsp_2011q2 USING STOGROUP sg_hot;
```

5. 将刚过去的季度的表空间移至 sg_warm 存储器组。要更改 tbsp_2011q1 表空间的存储器组关联，请发出带 USING STOGROUP 选项的 ALTER TABLESPACE 语句。

```
ALTER TABLESPACE tbsp_2011q1 USING STOGROUP sg_warm;
```

缺省存储器组

如果数据库有存储器组，那么在未显式指定存储器组的情况下创建自动存储器管理的表空间时会使用缺省存储器组。

创建数据库时，会自动创建名为 IBMSTOGROUP 的缺省存储器组。但是，使用 AUTOMATIC STORAGE NO 子句创建的数据库没有缺省存储器组。使用 CREATE STOGROUP 语句创建的第一个存储器组变为指定的缺省存储器组。只有一个存储器组被指定为缺省存储器组。

注: 尽管可在创建数据库时指定 AUTOMATIC STORAGE NO 子句，但建议不要使用 AUTOMATIC STORAGE 子句，将来发行版可能会除去该子句。

可使用 CREATE STOGROUP 或 ALTER STOGROUP 语句来指定缺省存储器组。如果指定另一存储器组作为缺省存储器组，那么对使用旧缺省存储器组的现有表空间没有影响。要改变与表空间相关联的存储器组，请使用 ALTER TABLESPACE 语句。

可使用 SYSCAT.STOGROUPS 目录视图来确定哪个存储器组是缺省存储器组。

不能删除当前缺省存储器组。如果当时 IBMSTOGROUP 存储器组未指定为缺省存储器组，那么可删除该存储器组。如果删除 IBMSTOGROUP 存储器组，那么可使用该名称创建另一存储器组。

创建存储器组

使用 CREATE STOGROUP 语句来创建存储器组。在数据库内创建存储器组会向该存储器组指定存储器路径。

开始之前

如果使用 AUTOMATIC STORAGE NO 子句创建数据库，那么该数据库没有缺省存储器组。可使用 CREATE STOGROUP 语句来创建缺省存储器组。

注: 尽管可在创建数据库时指定 AUTOMATIC STORAGE NO 子句，但建议不要使用 AUTOMATIC STORAGE 子句，将来发行版可能会除去该子句。

过程

要使用命令行来创建存储器组，请输入以下语句:

```
CREATE STOGROUP operational_sg ON '/filesystem1', '/filesystem2', '/filesystem3'...
```

其中 *operational_sg* 是存储器组的名称，*/filesystem1*、*/filesystem2*、*/filesystem3* 等是要添加的存储器路径。

要点: 为帮助确保可预测性能，您指定给存储器组的所有路径应具有相同介质特征: 等待时间、设备读速率和大小。

改变存储器组

可使用 ALTER STOGROUP 语句来改变存储器组的定义，包括设置介质属性、设置数据标记或设置缺省存储器组。还可在存储器组中添加和除去存储器路径。

如果向存储器组添加存储器路径并且您想要使其表空间的扩展数据块在所有存储器路径间条带化，那么必须对与该存储器组相关联的每个表空间使用带 REBALANCE 选项的 ALTER TABLESPACE 语句。

如果从存储器组中删除存储器路径，那么必须使用带 REBALANCE 选项的 ALTER TABLESPACE 语句从已删除路径中除去已分配扩展数据块。

可使用 DB2 工作负载管理器 (WLM) 来定义有关如何根据与所访问数据相关联的标记来处理活动的规则。可在定义表空间或存储器组时将该标记与数据相关联。

添加存储器路径

可使用 ALTER STOGROUP 语句向存储器组添加存储器路径。

关于此任务

如果为多分区数据库环境添加存储器路径，那么每个数据库分区上必须存在该存储器路径。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。

过程

- 要向存储器组添加存储器路径，请发出以下 ALTER STOGROUP 语句:

  ```
  ALTER STOGROUP sg ADD '/hdd/path1', '/hdd/path2', ...
  ```

其中 *sg* 是存储器组，*/hdd/path1*、*/hdd/path2* 等是要添加的存储器路径。

要点：您指定的存储器组的所有路径都应具有相似介质特征：底层磁盘、等待时间、设备读速率和大小。如果路径具有非唯一介质特征，那么性能可能会不一致。

- 向存储器组添加一个或多个存储器路径后，可选择使用 ALTER TABLESPACE 语句重新平衡表空间以立即开始使用新存储器路径。否则，仅当现有存储器路径上的容器中没有空间时才会使用新存储器路径。要确定存储器组中所有受影响的永久表空间，请运行以下语句:

```
SELECT TBSP_NAME
   FROM table (MON_GET_TABLESPACE(' ', -2))
   WHERE TBSP_USING_AUTO_STORAGE = 1
      AND TBSP_CONTENT_TYPE IN ('ANY','LARGE')
      AND STORAGE_GROUP_NAME = 'sg'
   ORDER BY TBSP_ID
```

标识表空间后，可对所列示的每个表空间执行以下语句:

```
ALTER TABLESPACE tablespace_name REBALANCE
```

其中 *tablespace_name* 是表空间。

删除存储器路径

可从存储器组中删除一个或多个存储器路径，也可从存储器路径中移出数据并进行重新平衡。

开始之前

要确定永久表空间是否在使用该存储器路径，请使用 ADMIN_GET_STORAGE_PATHS 管理视图。此视图显示有关每个存储器组的存储器路径的当前信息。存储器路径可以处于三种状态的其中一种:

NOT_IN_USE
此存储器路径已被添加到数据库，但尚未被任何表空间使用。

IN_USE
一个或多个表空间已将容器置于此存储器路径中。

DROP_PENDING
已发出 ALTER STOGROUP *stogroup_name* DROP 语句以删除该路径，但表空间仍在使用该存储器路径。表空间不再使用该路径时，会从数据库中删除该路径。

如果您删除的存储器路径上存储了数据，并且该存储器路径处于 DROP_PENDING 状态，那么必须先重新平衡使用该存储器路径的所有永久表空间，数据库管理器才能完成路径删除操作。

要获取有关特定数据库分区上的表空间的信息，请使用 MON_GET_TABLESPACE 管理视图。

限制

存储器组必须具有至少一个路径。不能删除存储器组中的所有路径。

关于此任务

如果您打算删除某条存储器路径，那么必须使用 ALTER TABLESPACE tablespace-name REBALANCE（此语句将数据从所要删除的路径中移出）对所有使用了该存储器路径的永久表空间进行重新平衡。在这种情况下，重新平衡操作会将数据从您打算删除的存储器路径移至其余存储器路径，并且会保持数据一致地在那些存储器路径之间进行条带分布，从而最大程度地提高 I/O 并行性。

过程

1. 要从存储器组中删除存储器路径，请发出以下 ALTER STOGROUP 语句:

   ```
   ALTER STOGROUP sg DROP '/db2/filesystem1', '/db2/filesystem2'
   ```

   其中 *sg* 是存储器组，*/db2/filesystem1* 和 */db2/filesystem2* 是要删除的存储器路径。
2. 重新平衡要删除的存储器路径的容器。要确定数据库中所有受影响永久表空间的容器都驻留在“删除暂挂”路径上，请发出以下语句:

   ```
   SELECT TBSP_NAME
      FROM table (MON_GET_TABLESPACE(' ', -2))
      WHERE TBSP_USING_AUTO_STORAGE = 1
         AND TBSP_CONTENT_TYPE IN ('ANY','LARGE')
         AND STORAGE_GROUP_NAME = 'sg'
      ORDER BY TBSP_ID
   ```

   标识表空间后，可对所列示的每个表空间执行以下语句:

   ```
   ALTER TABLESPACE tablespace_name REBALANCE
   ```

   其中 *tablespace_name* 是表空间。

   最后一个重新平衡操作完成后，会从存储器组中除去 /db2/filesystem1 和 /db2/filesystem2。
3. 删除使用该存储器组的临时表空间。如果处于 DROP_PENDING 状态的表空间上存在临时表空间，那么不会删除该表空间。
4. 重新创建使用该存储器组的临时表空间。

下一步做什么

查询 ADMIN_GET_STORAGE_PATHS 管理视图以验证已删除的存储器路径是否未再列示。如果仍列示了该存储器路径，那么表明仍有一个或多个表空间使用该路径。

监视存储器路径

可使用管理视图和表函数来获得所使用存储器路径的信息。

以下管理视图和表函数可用于:

- 使用 ADMIN_GET_STORAGE_PATHS 管理视图来获取每个存储器组的存储器路径列表和每个存储器路径的文件系统信息。
- 在 MON_GET_TABLESPACE 表函数中使用 TBSP_USING_AUTOMATIC_STORAGE 和 STORAGE_GROUP_NAME 监视元素来了解表空间是否在使用自动存储器并标识表空间使用的存储器组。
- 在 MON_GET_CONTAINER 表函数中使用 DB_STORAGE_PATH_ID 监视元素来了解容器被定义在存储器组中的哪个存储器路径上。

替换存储器组的路径

将存储器组中的存储器路径替换为新存储器路径。

过程

要替换存储器组中的现有存储器路径，请执行以下操作:

1. 向现有存储器组添加新存储器路径。

   ```
   ALTER STOGROUP sg_default ADD '/hdd/path3', '/hdd/path4'
   ```

2. 删除旧存储器路径。

   ```
   ALTER STOGROUP sg_default DROP '/hdd/path1', '/hdd/path2'
   ```

   注: 所有存储器组都必须至少有一个路径并且不能删除这最后一个路径。这会将已删除存储器路径标记为 DROP PENDING。

3. 确定受影响的非临时表空间。

   ```
   SELECT TBSP_NAME
      FROM table (MON_GET_TABLESPACE(' ', -2))
      WHERE TBSP_USING_AUTO_STORAGE = 1
         AND TBSP_CONTENT_TYPE IN ('ANY','LARGE')
         AND STORAGE_GROUP_NAME = 'sg_default'
      ORDER BY TBSP_ID
   ```

4. 对返回的每个受影响非临时表空间执行以下语句。

   ```
   ALTER TABLESPACE tablespace-name REBALANCE
   ```

5. 如果对被删除存储器路径定义了任何临时表空间，那么必须先创建新的临时表空间，才能删除旧的临时表空间。

   ```
   SELECT TBSP_NAME
      FROM table (MON_GET_TABLESPACE(' ', -2))
      WHERE TBSP_USING_AUTO_STORAGE = 1
         AND TBSP_CONTENT_TYPE IN ('USRTEMP','SYSTEMP')
         AND STORAGE_GROUP_NAME = 'sg_default'
      ORDER BY TBSP_ID
   ```

重命名存储器组

使用 RENAME STOGROUP 语句来重命名存储器组

过程

使用以下语句来重命名存储器组:

```
RENAME STOGROUP sg_hot TO sg_warm
```

其中 sg_warm 是存储器组的新名称。

示例

如果在创建数据库时创建第一个存储器组，那么缺省存储器组名称为 *IBMSTOGROUP*。可使用以下语句来更改指定的缺省名称:

```
RENAME STOGROUP IBMSTOGROUP TO DEFAULT_SG
```

其中 DEFAULT_SG 是该存储器组的新缺省名称。

删除存储器组

可使用 DROP 语句来除去存储器组。

关于此任务

在删除存储器组之前，必须确定是否有任何表空间使用该存储器组。如果有这样的表空间，那么在删除原始存储器组之前，必须更换这些表空间使用的存储器组并完成重新平衡操作。

限制

不能删除当前缺省存储器组。

过程

要删除存储器组，请执行以下操作:

1. 找到正使用该存储器组的表空间。

   ```
   SELECT TBSP_NAME, TBSP_CONTENT_TYPE
      FROM table (MON_GET_TABLESPACE(' ', -2))
      WHERE TBSP_USING_AUTO_STORAGE = 1
         AND STORAGE_GROUP_NAME = STO_GROUP
      ORDER BY TBSP_ID
   ```

   其中 *STO_GROUP* 是要删除的存储器组。

2. 如果存在使用该存储器组的常规表空间或大型表空间，请将它们指定给另一存储器组:

   ```
   ALTER TABLESPACE tablespace_name USING STOGROUP sto_group_new
   ```

   其中 *sto_group_new* 是另一存储器组。

3. 如果存在使用要删除的存储器组的临时表空间，请执行以下步骤:

   a. 确定哪些临时表空间使用要删除的存储器组:

      ```
      SELECT TBSP_NAME
         FROM table (MON_GET_TABLESPACE(' ', -2))
         WHERE TBSP_USING_AUTO_STORAGE = 1
            AND TBSP_CONTENT_TYPE IN ('USRTEMP','SYSTEMP')
            AND STORAGE_GROUP_NAME = 'STO_GROUP'
         ORDER BY TBSP_ID
      ```

   b. 删除使用该存储器组的临时表空间:

      ```
      DROP TABLESPACE table_space
      ```

   c. 重新创建使用该存储器组的临时表空间。

4. 监视要删除的存储器组的重新平衡活动。

   ```
   SELECT * from table (MON_GET_REBALANCE_STATUS( ' ', -2))
     WHERE REBALANCER_SOURCE_STORAGE_GROUP_NAME = sto_group_old
   ```

   空结果状态指示所有表空间已完成移至新存储器组的操作。

5. 在所有表空间扩展数据块已成功移至目标存储器组后，删除该存储器组。

   ```
   DROP STOGROUP STO_GROUP
   ```

   其中 *STO_GROUP* 是要删除的存储器组的名称。

存储器组和表空间介质属性

自动存储器表空间从该表空间缺省情况下使用的存储器组继承介质属性值、设备读速率和数据标记属性。

使用 CREATE STOGROUP 语句创建存储器组时，可指定以下存储器组属性:

OVERHEAD

此属性指定 I/O 控制器时间、磁盘搜索和等待时间（以毫秒计）。

DEVICE READ RATE

此属性指定设备的读取传输速率（以兆字节/秒计）规范。此值用于确定查询优化期间的 I/O 成本。如果此值对于所有存储器路径不同，那么该数字应是属于该存储器组的所有存储器路径的平均值。

DATA TAG

此属性指定特定存储器组中的数据的标记，WLM 可使用此标记来确定数据库活动的处理优先级。

存储器组属性的缺省值如下所示:

表 20. 存储器组属性的缺省设置

属性	缺省设置
DATA TAG	NONE
DEVICE READ RATE	100 MB/sec
OVERHEAD	6.725 ms

创建自动存储器表空间时，可指定用于标识该表空间中包含的数据的标记。如果该表空间与存储器组相关联，那么表空间的数据标记属性将覆盖可能对该存储器组设置的任何数据标记属性。如果用户未对表空间指定数据标记属性，并且该表空间包含在存储器组中，那么该表空间将从该存储器组继承数据标记值。可对除目录表空间以外的任何常规表空间或大型表空间设置数据标记属性 (SQL0109N)。不能对临时表空间设置数据标记属性，否则将返回 SQL0109N 消息错误。

自动存储器表空间从它使用的存储器组继承开销和传输速率属性。如果表空间从它使用的存储器组继承传输速率属性，那么该存储器组的设备读速率将从“毫秒/所读页”进行转换（考虑表空间的页大小设置），如下所示:

```
TRANSFERRATE = ( 1 / DEVICE READ RATE ) * 1000 / 1024000 * PAGESIZE
```

自动存储器表空间和非自动表空间的页大小设置具有对应的缺省 TRANSFERRATE 值:

表 21. 缺省 TRANSFERRATE 值

PAGESIZE	TRANSFERRATE
4 KB	0.04 毫秒/所读页
8 KB	0.08 毫秒/所读页
16 KB	0.16 毫秒/所读页
32 KB	0.32 毫秒/所读页

自动存储器表空间的数据标记、设备读速率和开销介质属性可更改为从其关联存储器组动态继承值。为动态更新介质属性，请对 CREATE TABLESPACE 或 ALTER TABLESPACE 语句指定 INHERIT 选项。

表空间从存储器组继承属性值时，SYSCAT.TABLESPACES 目录表视图报告该属性的值为 -1。要查看开销、传输率和数据标记属性在运行时的实际值，可使用下列查询:

```
 select tbspace,
  cast(case when a.datatag = -1 then b.datatag else a.datatag end
as smallint) eff_datatag,
  eff_datatag,
  cast(case when a.overhead = -1 then b.overhead else a.overhead
end as double) eff_overhead,
  eff_overhead,
  cast(case when a.transferrate = -1 then (1 / b.devicereadrate) / 1024 * a.pagesize
else a.transferrate end as double) eff_transferrate
  eff_transferrate
   from syscat.tablespaces a left outer join syscat.stogroups b on a.sgid = b.sgid
```

如果升级至 V10.1，那么现有表空间将保留其开销和传输速率设置，并且该存储器组的开销和设备读速率属性设为 undefined（未定义）。存储器组中新创建的表空间（设备读速率设为未定义）使用最初创建 DB2 数据库时定义的数据库缺省值。如果存储器组的介质设置具有有效值，那么新创建的表空间将继承这些值。可使用 ALTER STOGROUP 语句来设置该存储器组的介质属性。对于非自动表空间，介质属性将保留。

使表空间与存储器组相关联

通过使用 CREATE TABLESPACE 语句或 ALTER TABLESPACE 语句，可指定或更改表空间使用的存储器组。如果创建表空间时未指定存储器组，那么会使用缺省存储器组。

关于此任务

如果更改表空间使用的存储器组，那么落实 ALTER TABLESPACE 语句时会发出隐式 REBALANCE 操作。这会将数据从源存储器组移至目标存储器组。

使用 IBM DB2 pureScale Feature时，REBALANCE 不受支持，并且您不能更改所指定存储器组。REBALANCE 操作异步进行，并且不会影响数据的可用性。可使用监视表函数 MON_GET_REBALANCE_STATUS 来监视 REBALANCE 操作的进度。

在 ALTER TABLESPACE 操作期间，基于旧表空间属性的已编译对象会*软失效*。ALTER TABLESPACE 落实后进行的任何新编译使用 ALTER TABLESPACE 语句中指定的新表空间属性。*软失效*支持仅限于动态 SQL，您必须手动检测并重新编译所有静态 SQL 依赖项，才能使用新值。

使用同一存储器组的所有表空间可有不同的 PAGESIZE 和 EXTENTSIZE 值。这些属性与表空间定义相关，与存储器组无关。

过程

要将表空间与存储器组相关联，请发出以下语句:

```
CREATE TABLESPACE tbspc USING STOGROUP storage_group
```

其中 *tbspc* 是新表空间，*storage_group* 是关联存储器组。

方案: 将表空间移至新存储器组

此方案显示如何将表空间从一个存储器组移至另一个存储器组。

此方案中的假定是表空间数据在存储器组内存储器路径上的容器中。ALTER TABLESPACE 语句用于将表空间数据移至新存储器组。

表空间移至新存储器组后，旧存储器组中的容器会标记为删除暂挂。落实 ALTER TABLESPACE 语句后，容器分配到新存储器组的存储器路径上，驻留在旧存储器组中的现有容器标记为删除暂挂，并且会启动隐式 REBALANCE 操作。此操作将容器分配到新存储器路径上，并将现有容器中的数据重新平衡到新容器中。要创建的容器的数目和大小取决于目标存储器组中的存储器路径数以及新存储器路径上的可用空间量。移动所有数据后，会删除旧容器。

下图是将表空间从一个存储器组移至另一个存储器组的示例，其中:

1. 新容器分配到目标存储器组的存储器路径上。
2. 所有原始容器被标记为删除暂挂，新分配请求通过新容器得到满足。
3. 执行了反向重新平衡，从路径上的容器移出的数据被删除。
4. 以物理方式删除容器。

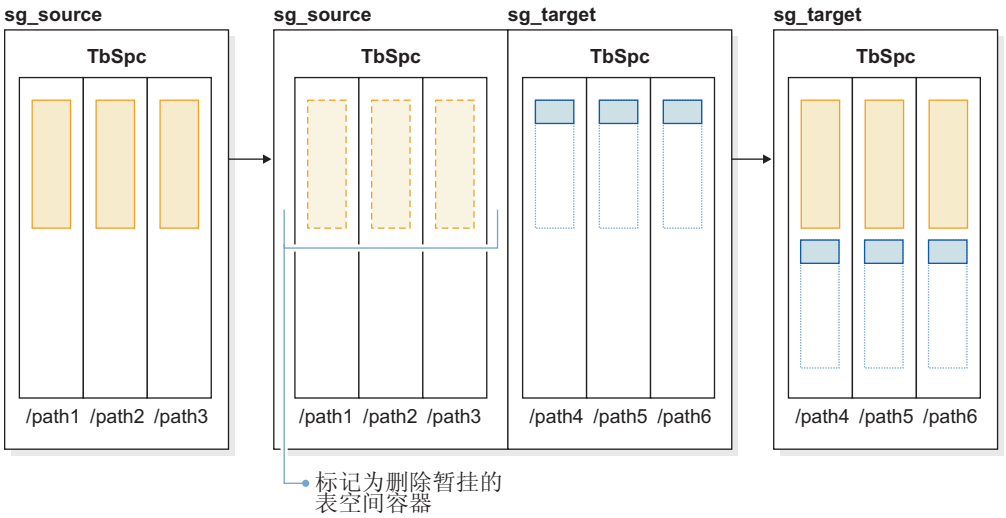


图 33. 将表空间移至新存储器组

要将表空间移至另一存储器组，请执行以下操作:

1. 创建两个存储器组: sg_source 和 sg_target:

```
CREATE STOGROUP sg_source ON '/path1', '/path2', '/path3'
CREATE STOGROUP sg_target ON '/path4', '/path5', '/path6'
```

2. 创建数据库后，创建最初使用该 sg_source 存储器组的自动存储器表空间:

```
CREATE TABLESPACE TbSpc USING STOGROUP sg_source
```

3. 将自动存储器表空间移至 sg_target 存储器组:

```
ALTER TABLESPACE TbSpc USING sg_target
```

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。现在不仅能创建别名、单值类型、函数和触发器之类的对象，还可隐式创建模式。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。

隐式创建的模式的所有者为 SYSIBM。如果数据库是限制性的，那么 PUBLIC 没有对此模式的 CREATEIN 特权。隐式创建此模式的用户具有对此模式的 CREATEIN 特权。如果数据库不是限制性的，那么 PUBLIC 具有对此模式的 CREATEIN 特权。

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具有创建、改变和删除模式中的对象的特权。此能力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其他人。隐式创建的模式由系统拥有，并且所有用户最初被授予在这类模式中创建对象的特权，在限制性数据库环境中时除外。具有 ACCESSCTRL 或 SECADM 权限的用户可以更改用户对任何模式拥有的特权。因此，可以控制在任何模式（即使是隐式创建的模式）中创建、改变和删除对象的特权。

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。

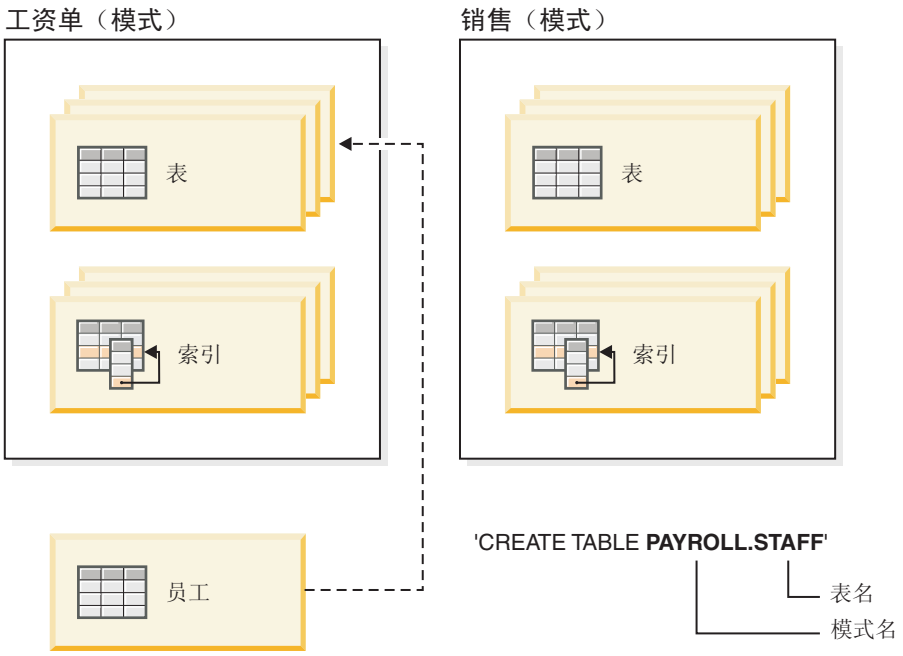
可以将模式比作目录，并且当前模式相当于当前目录。通过这种类比，SET SCHEMA 相当于 **change directory** 命令。

要点：了解除以下一节描述的缺省 CURRENT SCHEMA 设置之外，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 237 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 DBADM 权限，那么该用户可以创建具有任何有效名称的模式。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，必须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，您必须了解一些限制和建议。

- 用户定义的类型 (UDT) 的模式名长度不能超出 *SQL Reference*中的『SQL 和 XML 限制』中列示的模式长度。
- 下列模式名是保留字，不能使用: SYSCAT、SYSFUN、SYSIBM、SYSSTAT 和 SYSPROC。
- 为了避免将来升级数据库时的潜在问题，切勿使用以 SYS 开头的模式名。数据库管理器将不允许您使用以 SYS 开头的模式名来创建模块、过程、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。

开始之前

要创建模式，并且要使另一个用户成为该模式的所有者（可选），您需要 DBADM 权限。即使您不具有 DBADM 权限，也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。

过程

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 过程允许您快速复制数据库模式。在建立模型模式后，可以将它用作创建新版本的模板。

过程

- 使用 ADMIN_COPY_SCHEMA 过程在同一数据库中复制单个模式。
- 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。 源模式中的大部分数据库对象被复制到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。因为该表空间已经处于“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除以下类型以外的所有允许模式对象:

- 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

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一个不受支持的类型，会将一个条目记录到错误文件中。错误文件指示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

按以下示例中所示使用 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 语句的输出如以下示例所示:

```
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 实用程序进行模式复制的示例

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 关键字以指示必须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启动 **db2move** 实用程序。要将模式从一个数据库复制到另一个数据库，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，请从操作系统命令提示符中发出 **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。

关于此任务

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。

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导致实用程序回滚。

过程

要重新启动复制模式操作，请执行以下操作:

解决导致装入操作失败的问题（已在错误文件中描述）后，重新发出 **db2move-COPY** 命令。 将 **-tf** 参数与 LOADTABLE.err 文件名配合使用以指定要复制并使用数据填充的表。
例如:

```
db2move sourcedb COPY -tf LOADTABLE.err -co TARGET_DB mytarget_db
  -mode load_only
```

还可使用 **-tn** 参数来手动输入表名。例如:

```
db2move sourcedb COPY -tn "FOO"."TABLE1","FOO 1"."TAB 444",
  -co TARGET_DB mytarget_db -mode load_only
```

注: load_only 方式需要使用 **-tn** 或 **-tf** 参数来输入至少一个表。

示例

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。为将所有可能的错误收集到错误文件中，尝试重新创建每个对象后在重新创建阶段结束时（而不是第一次失败后）发生了回滚。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。**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 **db2move** 尝试使用装入实用程序重新填充表数据时发生了错误，那么 **db2move** 实用程序不会失败。而是，一般故障信息被记录至 COPYSCHEMA_*timestamp*.err 文件（例如，对象名、对象类型、DDL 文本、时间戳记和 sqlca），表的标准名称被记录至另一文件，即 LOADTABLE_*timestamp*.err。每行列示一个表以满足 **db2move -tf** 参数格式，类似以下所示:

```
"FOO"."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 DDL 重新创建阶段期间发生了内部操作失败，那么会从目标模式回滚所有成功创建的对象。系统会在源数据库上清除所有内部创建的表，例如，DMT 表和 **db2look** 表。

如果装入阶段期间发生了内部操作失败，那么所有成功创建的对象将保留在目标模式上。在装入操作期间遇到失败的所有表以及尚未装入的所有表记录在 LOADTABLE.err 错误文件中。然后，可按示例 2 中的讨论发出 **db2move COPY** 命令并使用 LOADTABLE.err。如果 **db2move** 实用程序异常中止（例如，发生系统崩溃、实用程序落陷或实用程序终止），那么有关仍必须装入哪些表的信息会丢失。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。然后，可重新发出 **db2move COPY** 命令。

删除模式

要删除模式，请使用 DROP 语句。

开始之前

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。

过程

要使用命令行删除模式，请输入:

```
DROP SCHEMA name RESTRICT
```

RESTRICT 关键字强制实施以下规则: 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。RESTRICT 关键字并非可选关键字。

示例

在以下示例中，删除了模式“joeschma”:

```
DROP SCHEMA joeschma RESTRICT
```

第 3 部分 数据库对象

物理数据库设计包括定义数据库对象及其关系。

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
- 用法列表

可以使用数据定义语言 (DDL) 语句或工具（例如 IBM Data Studio）来创建这些数据库对象。DDL 语句通常以关键字 CREATE 或 ALTER 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。

第 11 章 大部分数据库对象的共用概念

别名

*别名*是对象（例如模块、表或另一个别名）的备用名称。它可以用来在对象可以被直接引用时引用该对象。

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临时表，但可以引用已创建临时表。

与其他对象相同，您可以创建和删除别名以及使其与注释相关联。只要不存在循环引用，别名就可以通过称为*链接*的过程来引用其他别名。您无需具有任何特权即可使用别名。但是，访问别名所引用的对象要求具有与该对象相关联的权限。

如果将别名定义为*公用*别名，那么可以通过它的未限定名称对其进行引用，而不受当前缺省模式名影响。另外，还可以使用限定符 SYSPUBLIC 来引用该别名。

*同义词*是对“别名”的另一种称呼。

有关更多信息，请参阅*SQL Reference Volume 1*中的『标识中的别名』。

创建数据库对象别名

*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。

关于此任务

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。

注：DB2 for z/OS® 使用两种不同的别名概念：ALIAS 和 SYNONYM。这两种概念在 DB2 for Linux, UNIX, and Windows 中是有区别的，如下所示：

- DB2 for z/OS 中的 ALIAS:
  - 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 for z/OS 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

过程

要使用命令行来创建别名，请输入:

```
CREATE ALIAS alias_name FOR table_name
```

下列 SQL 语句为 EMPLOYEE 表创建别名 WORKERS:

```
CREATE ALIAS WORKERS FOR EMPLOYEE
```

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
```

使数据库对象软失效

当*软失效*功能处于活动状态时，您可以删除正被其他运行中的事务使用的对象。正在使用所删除对象的事务将被允许继续运行，但任何要访问所删除对象的新事务都将被拒绝。

已进行高速缓存的语句和程序包如果直接或间接地引用了正被删除或更改的对象，那么所有这些语句和程序包都将被标记为无效（称为*失效*）。软失效功能使影响被引用对象的 DDL 能够避免由于它们所引用对象被正在运行的语句锁定而等待，并允许任何活动访问通过使用该对象的高速缓存版本继续进行，从而消除锁定超时的可能性。

相反，如果使用*硬失效*功能，那么引用对象时将挂起互斥锁定。这将确保所有进程使用相同的对象版本，并确保一旦删除对象就不允许对其进行任何访问。

软失效功能是通过 **DB2_DDL_SOFT_INVAL** 注册表变量启用的；缺省情况下，此注册表变量设为 ON。

以下列表显示了支持软失效功能的数据定义语言（DDL）语句:

- ALTER TABLE...DETACH PARTITION
- CREATE OR REPLACE ALIAS
- CREATE OR REPLACE FUNCTION
- CREATE OR REPLACE TRIGGER
- CREATE OR REPLACE VIEW

- DROP ALIAS
- DROP FUNCTION
- DROP TRIGGER
- DROP VIEW

注: 在 DB2 V9.7 FP1 及更高版本的发行版中，ALTER TABLE...DETACH PARTITION 将在所有隔离级别对直接或间接引用分区表的已高速缓存语句执行软失效。在将已拆离的分区转换为独立表之前，后续的异步分区拆离任务将对先前已执行软失效的已高速缓存语句执行硬失效。

DB2_DDL_SOFT_INVAL 注册表变量不会影响由 ALTER TABLE...DETACH PARTITION 完成的失效。

软失效支持仅适用于动态 SQL 以及在“游标稳定性”(CS) 和“未落实的读”(UR) 隔离级别下执行的扫描。对于 ALTER TABLE...DETACH PARTITION 语句，软失效适用于所有隔离级别下的扫描。

示例

假定存在一个名为 VIEW1 的视图。您打开一个游标，然后运行语句 `SELECT * from VIEW1`。此后不久，数据库管理员发出命令 `DROP VIEW VIEW1` 从数据库中删除 VIEW1。在使用硬失效功能的情况下，DROP VIEW 语句将必须等待对 VIEW1 挂起互斥锁定，直到 SELECT 事务完成为止。在使用软失效功能的情况下，DROP VIEW 语句不会对该事务挂起互斥锁定。该视图将被删除，但是，该 SELECT 语句将使用该视图的最新定义继续运行。在该 SELECT 语句完成之后，任何使用 VIEW1 的后续尝试（即使此尝试由刚刚使用该视图的用户或进程进行）都将导致错误（SQL0204N）。

使数据库对象自动重新生效

*自动重新生效*是运行时访问无效的数据库对象时使其自动重新生效的一种机制。

一个数据库对象通常依赖于一个或多个不同的基本对象。如果数据库对象所依赖的基本对象的状态在任何重要的方面作出更改（例如变更或删除基本对象），那么从属数据库对象会变无效。必须使无效的数据库对象重新生效，然后才能再次使用这些对象。重新生效是 DB2 软件据以重新处理无效从属对象的定义的过程，这样对象就会使用其基本对象的当前状态进行更新，从而将无效的从属对象转回为可用的有效对象。自动重新生效是运行时访问无效的数据库对象时使其自动重新生效的一种机制。

通常，数据库管理器将在无效对象下次被使用时尝试使那些对象重新生效。通过 **auto_reval** 配置参数启用了自动重新验证。缺省情况下，此注册表变量设为 DEFERRED（从 V9.5 或更低版本升级的数据库除外，对于这些数据库，**auto_reval** 设为 DISABLED）。

有关删除对象时受影响的从属对象以及那些从属对象何时重新生效的信息，请参阅 *SQL Reference Volume 1*中的『DROP 语句』。

以下列表显示了当前支持自动重新生效功能的数据定义语言 (DDL) 语句:

- ALTER MODULE DROP FUNCTION
- ALTER MODULE DROP PROCEDURE

- ALTER MODULE DROP TYPE
- ALTER MODULE DROP VARIABLE
- ALTER NICKNAME（更改局部名或局部类型）
- ALTER TABLE ALTER COLUMN
- ALTER TABLE DROP COLUMN
- ALTER TABLE RENAME COLUMN
- CREATE OR REPLACE ALIAS
- CREATE OR REPLACE FUNCTION
- CREATE OR REPLACE NICKNAME
- CREATE OR REPLACE PROCEDURE
- CREATE OR REPLACE SEQUENCE
- CREATE OR REPLACE TRIGGER
- CREATE OR REPLACE VARIABLE
- CREATE OR REPLACE VIEW
- DROP FUNCTION
- DROP NICKNAME
- DROP PROCEDURE
- DROP SEQUENCE
- DROP TABLE
- DROP TRIGGER
- DROP TYPE
- DROP VARIABLE
- DROP VIEW
- RENAME TABLE

可以使用 ADMIN_REVALIDATE_DB_OBJECTS 过程来重新验证已经标记为无效的现有对象。

创建和维护数据库对象

创建某些类型的数据库对象时，您应该了解具有出错支持的 CREATE 以及 REPLACE 选项。

用于某些数据库对象的具有出错支持的 CREATE

对于某些对象而言，即使编译其主体期间发生错误，也能够创建那些对象；例如，在创建视图时，即使该视图所引用的表不存在，也能够创建该视图。

此类对象在它们被访问前将保持处于无效状态。目前，具有出错支持的 CREATE 已扩展到视图以及直接插入型 SQL 函数（而不是编译型函数）。如果 **auto_reval** 数据库配置参数设为 IMMEDIATE 或 DEFERRED，那么将启用此功能。

在对象创建期间能够容忍的错误仅限于下列类型的错误:

- 任何名称解析错误，例如：所引用的表不存在（SQLSTATE 42704，SQL0204N）、所引用的列不存在（SQLSTATE 42703，SQL0206N）或者找不到所引用的函数（SQLSTATE 42884，SQL0440N）
- 任何嵌套的重新生效故障。正在创建的对象可以引用无效的对象，对于那些无效的对象，将调用重新生效操作。即使未能使任何所引用的无效对象重新生效，CREATE 语句也将成功，并且所创建的对象在下次被访问前将保持处于无效状态。
- 任何授权错误（SQLSTATE 42501，SQL0551N）

即使对象体包含多个错误，也可以成功地创建该对象。返回的警告消息包含编译期间遇到的第一个未定义、无效或未经授权对象的名称。SYSCAT.INVALIDOBJECTS 目录视图包含有关无效对象的信息。

可以使用 ADMIN_REVALIDATE_DB_OBJECTS 过程来重新验证已经标记为无效的现有对象。

示例

```
create view v2 as select * from v1
```

即使 v1 不存在，CREATE VIEW 语句也将成功完成，但 v2 将保持处于无效状态。

多个 CREATE 语句中的 REPLACE 选项

在用于别名、函数、模块、昵称、过程（包括联合过程）、序列、触发器、变量和视图等多种对象的 CREATE 语句中，**OR REPLACE** 子句允许在该对象已存在时将其替换；如果该对象不存在，那么将创建该对象。这将显著减少更改数据库模式所需的人工。

先前授予的对一个对象的特权在该对象被替换时将被保留。换而言之，CREATE OR REPLACE 在语义上相当于先执行 DROP，然后执行 CREATE。对于函数、过程和触发器而言，此支持既适用于直接插入型对象也适用于编译型对象。

对于函数和过程而言，此支持既适用于 SQL 也适用于外部函数和过程。如果某个模块被替换，那么其中的所有对象都将被删除；新版本的模块不包含任何对象。

直接或间接依赖于被替换对象的对象将失效。在替换操作后对所有从属对象执行的重新生效操作始终在失效操作完成后立即执行，即使 **auto_reval** 数据库配置参数设为 DISABLED 亦如此。

示例

替换具有从属对象的视图 v1。

```
create table t1 (c1 int, c2 int);
create table t2 (c1 int, c2 int);

create view v1 as select * from t1;
create view v2 as select * from v1;

create function foo1()
 language sql
 returns int
 return select c1 from v2;

create or replace v1 as select * from t2;
```

```
select * from v2;

values foo1();
```

替换后的 v1 版本将引用 t2，而不是引用 t1。CREATE OR REPLACE 语句将使 v2 和 foo1 都失效。在延迟重新生效语义下，`select * from v2` 将成功地使 v2 重新生效，但不会使 foo1 重新生效，values foo1() 将使 foo1 重新生效。在立即重新生效语义下，CREATE OR REPLACE 语句将成功地使 v2 和 foo1 重新生效。

第 12 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

DB2 数据库在表中存储数据。除了用于存储持久数据的表以外，还有用于显示结果的表、摘要表和临时表；多维集群表在仓库环境中具有特定的优点。

基本表 这些类型的表将保存持久数据。存在不同类型的基本表，其中包括:

常规表 具有索引的常规表是“常规用途”表选项。

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。MDC 表在组合维中提供*可靠集群*。相反，虽然可以对常规表建立集群索引，并且数据库管理器在这种情况下将尝试进行集群，但此集群不可靠，并且集群程度会随着时间的推移而下降。MDC 表可以与分区表共存，它们本身可以是分区表。

DB2 pureScale 环境不支持多维集群表。

插入时间集群 (ITC) 表

这些类型的表是概念上的，实际上与 MDC 表类似，但是，行按它们插入到表中的时间进行集群，而不是按一个或多个用户指定维进行集群。ITC 表可以是分区表。

DB2 pureScale 环境不支持 ITC 表。

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。

DB2 pureScale 环境不支持范围集群表。

分区表 这些类型的表使用了数据组织方案，即，表数据根据该表中一个或多个表分区键列中的值分布到多个存储对象（称为数据分区或范围）中。可以对分区表添加数据分区、将数据分区与分区表相连以及断开

数据分区与分区表的连接，并且，可以将一个表的多个数据分区范围存储在一个表空间中。分区表可以包含大量数据，并且简化表数据的转入和转出。

临时表 这些类型的表用于将基于时间的状态信息与数据相关联。未使用临时支持的表中的数据表示现在，而临时表中的数据在数据库系统和/或客户应用程序定义的时间段有效。例如，数据库可存储表的历史记录（已删除行或已更新的行的原始值），所以可查询数据的过去状态。还可对数据行指定日期范围以通过应用程序或业务规则指示该数据何时被认为有效。

临时表 这些类型的表用作各种数据库操作的临时工作表。*已声明临时表*（DGTT）不会在系统目录中出现，因此不会被保留下来以供其他应用程序使用，也不可供其他应用程序共享。当使用此表的应用程序终止或与数据库断开连接时，此表中的数据将被删除，此表也将被删除。相反，*已创建临时表*（CGTT）将在系统目录中出现，而不需要在每个使用这些表的会话中进行定义。因此，它们具有持久性，并能够跨不同的连接供其他应用程序共享。

这两种类型的临时表都不支持

- 用户定义的引用或用户定义的结构类型列
- LONG VARCHAR 列

另外，也不能在已创建临时表中使用 XML 列。

具体化查询表

这些类型的表是由查询定义的，也用于确定表中的数据。具体化查询表可用来改进查询的性能。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。摘要表是一种专用类型的具体化查询表。

您可以使用 CREATE TABLE 语句来创建上述所有类型的表。

根据数据的不同，您可能会发现，某种表类型的特定功能能够优化存储和查询性能。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（而不是唯一的）值，那么不应使用范围集群表。并且，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。如果数据有可能按多个维进行集群，例如按地理区域、分公司和供应商来跟踪零售额的表，那么多维集群表能够满足您的需要。

除了上面描述的各种表类型以外，您还可以选择*分区*之类的特征选项，这有助于提高滚入表数据之类的任务的性能。另外，与常规的非分区表相比，分区表能够存放更多信息。您还可以使用*压缩*之类的功能，这可以帮助您显著降低数据数据存储器成本。

设计表

在设计表时，您必须熟悉某些概念、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，压缩和乐观锁定。

设计分区表时，您必须熟悉分区概念，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 288 页的『表分区和数据组织方案』。

表设计概念

在设计表时，您必须熟悉一些相关概念。

数据类型和表列

在创建表时，必须指示每一列将要存储的数据的类型。通过仔细考虑您将要管理的数据的性质，可以对表进行设置以确保查询性能最佳、最大程度地减少所需的物理存储器以及为您提供专用的能力来处理不同类型的数据，例如用于数字数据的算术运算或者在日期或时间值之间进行比较。

第 256 页的图 34 说明了 DB2 数据库所支持的数据类型。

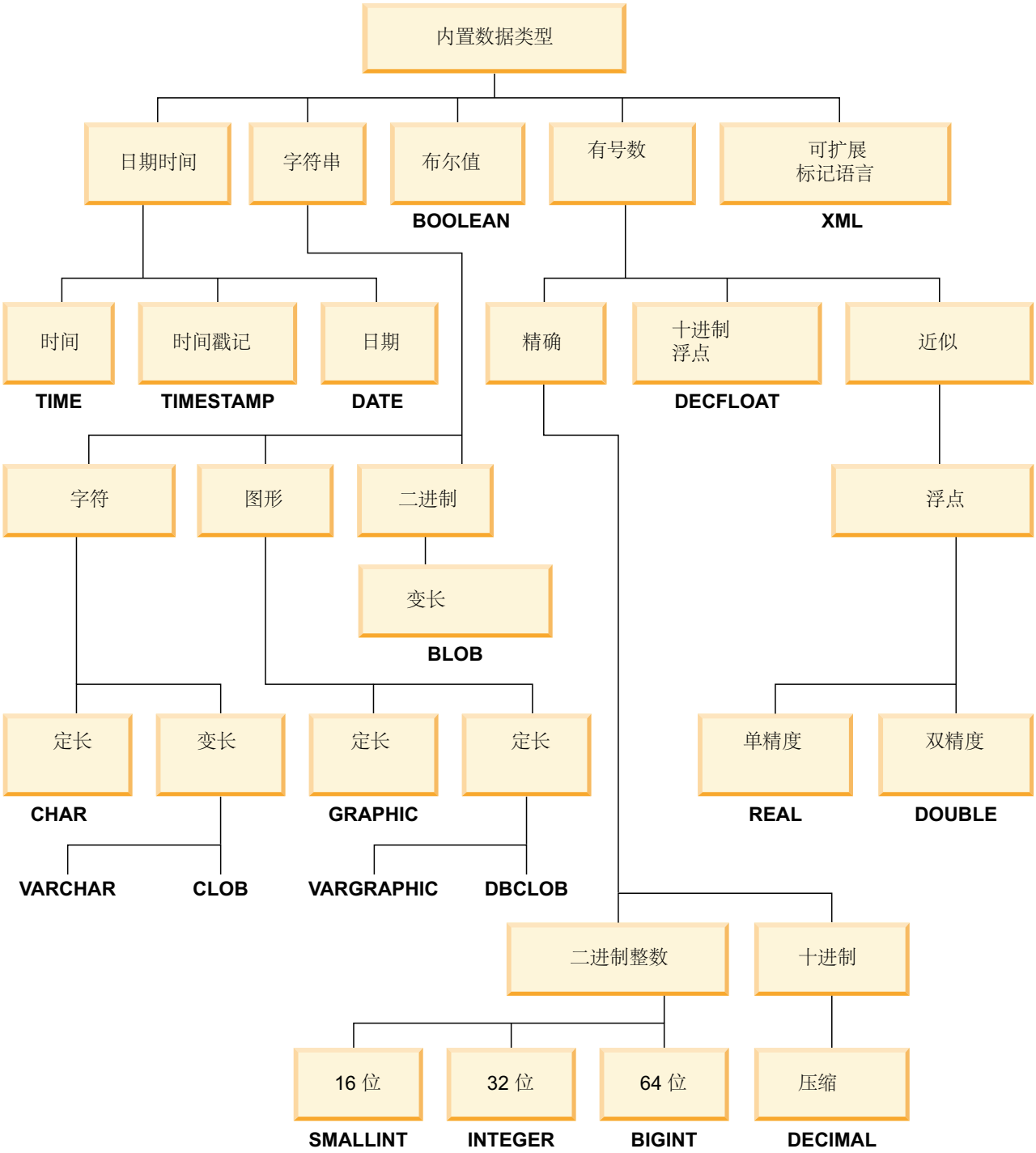


图 34. 内置数据类型

声明数据库列时，所有这些数据类型都可供您选择。除内置类型以外，您还可以创建基于内置类型的*用户定义*数据类型。例如，您可以选择通过包含 VARCHAR（姓名和工作职务）、SMALLINT（工作级别）、DATE（受雇日期）和 DECIMAL（薪水）数据的用户定义*结构*类型来表示具有姓名、工作职务、工作级别、受雇日期和薪水属性的职员。

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更多表中的数据时，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
```

隐藏列

如果定义了带有隐式隐藏属性的表列，那么直到显式引用该列，该列才可用。例如，如果对表运行 SELECT * 查询，那么结果表中不会返回隐式隐藏列。始终可以指定列名的位置显式引用隐式隐藏列。

如果列及其条目由数据库管理器生成，那么将这类列定义为 IMPLICITLY HIDDEN 可尽量减少对应用程序的任何潜在负面影响。例如，系统时间段临时表有 3 列的值由数据

库管理器生成。数据库管理器使用这些列来保存每个表行的历史版本。大部分业务应用程序将使用历史数据，但几乎不会使用这 3 个生成列。对应用程序隐藏这些列可缩短应用程序处理时间。

在表中插入数据时，不带列列表的 INSERT 语句不需要任何隐式隐藏列的值。在这类情况下，如果输入包括隐式隐藏列的值，那么该值没有对应的目标列并且系统会返回错误 (SQLSTATE 42802)。因为不带列列表的 INSERT 语句未包括隐式隐藏列的值，所有定义为隐式隐藏并且“非空”的任何列都必须具有已定义的缺省值

使用输入文件中的数据填充表时，导入、插入或装入之类的实用程序都需要您指定该操作是否包括隐藏列的数据。如果未指定列列表，那么在处理包含隐式隐藏列的表时，数据移动实用程序必须使用 *implicitlyhiddeninclude* 或 *implicitlyhiddenmissing* 文件类型修饰符。数据移动实用程序遇到带隐式隐藏列的表时，还可使用 DB2_DMU_DEFAULT 注册表变量来设置缺省行为。同样，导出操作需要您指定该操作是否包括隐藏列的数据。

可对新表使用 CREATE TABLE 语句或对现有表使用 ALTER TABLE 语句来对表列定义隐式隐藏属性。如果使用带 LIKE 子句的 CREATE TABLE 语句来创建表，那么新表将继承源表中的任何隐式隐藏列。可使用 ALTER TABLE 语句将隐藏列更改为不隐藏或将不隐藏列更改为隐藏。改变表以更改某些列的隐藏属性可能会影响要处理该表的数据移动实用程序的行为。例如，这可能意味着改变该表以定义某些隐藏列之前成功运行的装入操作现在返回错误 (SQLCODE -2437)。

用于标识带有 *exposed-name.** 选项运行 SELECT 查询生成的结果表的列的名称列表不包括任何隐式隐藏列。带 `order-by-clause` 选项运行的 SELECT 查询可在 *simple-column-name* 中包括隐式隐藏列。

如果在具体化查询表定义中显式引用了隐式隐藏列，那么该列将成为具体化查询表的一部分。但是，具体化查询表中的该列不会继承隐式隐藏属性。此相同行为将应用于使用 as-result-table 子句创建的视图和表。

可在 CREATE INDEX 语句、ALTER TABLE 语句或引用约束中显式引用隐式隐藏列。

定义为隐式隐藏的任何列都存在转换变量。在触发器主体中，可引用与隐式隐藏列对应的转换变量。

已创建临时表和已声明临时表中不支持隐式隐藏列。

可使用 DESCRIBE 命令显示表的隐藏列。

```
DESCRIBE TABLE tablename SHOW DETAIL
```

示例

- *示例 1:* 在以下语句中，创建了带隐式隐藏列的表。

```
CREATE TABLE CUSTOMER
(
 CUSTOMERNO       INTEGER NOT NULL,
 CUSTOMERNAME     VARCHAR(80),
 PHONENO          CHAR(8) IMPLICITLY HIDDEN
);
```

  `SELECT *` 仅返回对应 `CUSTOMERNO` 和 `CUSTOMERNAME` 的列条目。例如:

```
A123, ACME
B567, First Choice
C345, National Chain
```

除非显式引用，否则 PHONENO 列的条目会隐藏。

```
SELECT CUSTOMERNO, CUSTOMERNAME, PHONENO
   FROM CUSTOMER
```

- *示例 2:* 如果数据库表包含隐式隐藏列，那么必须指定数据移动操作是否包括隐藏列的数据。以下示例使用装入操作来说明用于指示是否包括隐藏列的数据的不同方法:
  - 使用 *insert-column* 来显式指定要在其中插入数据的列。

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

  - 使用某个隐藏列文件类型修饰符: 指定 **implicitlyhiddeninclude**（如果输入文件包含隐藏列的数据）或 **implicitlyhiddenmissing**（如果输入文件未包含隐藏列的数据）。

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

  - 在服务器端使用 DB2_DMU_DEFAULT 注册表变量来设置数据移动实用程序遇到带隐式隐藏列的表时的行为。

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

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，如果必须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，首先必须使用 DEFAULT 选项来添加列，以获取现有的缺省值。然后，可以更改 (ALTER) 缺省值，以使其成为标识列。

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。

列数据约束、缺省值和空设置

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: 『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 355 页的第 13 章，『约束』或第 355 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、双精度浮点和十进制浮点数据类型。
- *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*: 用于可变长度字符串、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*: 这是用户定义的单值数据类型的基本数据类型的内置缺省值（强制转换为用户定义的单值数据类型）。

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。

有四种不同类型的 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。
- 完整行映像前、更改字节和大小增大日志记录将更新对行末尾追加的新数据。对于支持“当前已落实”的数据库，如果未对该表启用 DATA CAPTURE CHANGES，并且更新是事务中对此行执行的第一项操作时，将进行此日志记录。这将记录“当前已落实”所需的前映像以及重做/撤销所需的最小数据量。通过对末尾经常更新的列进行排序，可以最大程度地减少行的已更改部分的日志记录量。
- 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。

对于上面列示的前两种类型的 UPDATE 日志记录而言，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。

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:

- *全球化指南*中的『Unicode 字符编码』
- *全球化指南*中的『基于整理顺序的字符比较』
- *全球化指南*中的『基于地域代码的日期和时间格式』
- *全球化指南*中的『支持欧元符号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

设计表时，您需要考虑那些表将要包含的数据的空间需求。特别是，必须注意数据类型较大的列，例如 LOB 列或 XML 列。

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，必须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分配页中。这些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。额外空间的计算方式如下:

表 22. 基于页大小的分配页额外空间

页大小	分配页数
4 KB	每 4 MB 一页，另外每 1 GB 一页
8 KB	每 8 MB 一页，另外每 2 GB 一页
16 KB	每 16 MB 一页，另外每 4 GB 一页
32 KB	每 32 MB 一页，另外每 8 GB 一页

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。

注: 通过使用 CREATE 和 ALTER TABLE 语句的 INLINE LENGTH 选项，可以将某些 LOB 数据放入基本表行。

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。

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使用的临时表空间大小。

XML 数据

插入到类型为 XML 的列中的 XML 文档可以驻留在缺省的存储器对象中，也可以直接驻留在基本表行中。基本表行存储器由您控制，并且仅可用于小型文档；较大的文档始终存储在缺省的存储器对象中。

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。除非使用列的 INLINE LENGTH 选项来直接插入 LOB 或 XML 文档，否则表数据页不包含使用 LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB、DCLOB 或 XML 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。

注: 通过使用 CREATE 和 ALTER TABLE 语句的 INLINE LENGTH 选项，可以将某些 LOB 和 XML 数据放入基本表行。

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 64 TB。

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。每页的最大行数为 255，此数目与页大小无关。

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，这样不需要的行耗用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`，那么每一页上都会耗用空间。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，表数据根据该表所在表空间的页大小进行存储。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。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、DBCLOB 或 XML 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。

注: 通过使用 CREATE 和 ALTER TABLE 语句的 INLINE LENGTH 选项，可以将某些 LOB 数据放入基本表行。

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。

如果发出带有 `APPEND ON` 选项的 ALTER TABLE 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更改，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 64 TB。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，因意外的行而耗用的缓冲区空间更少。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。

已声明临时表或已创建临时表只能采用它们自己的用户临时表空间类型进行声明或创建。没有缺省用户临时表空间。当应用程序与数据库断开连接时，临时表将被隐式删除，估计这些表的空间需求时应考虑这一点。

以直接插入方式将 LOB 存储在表行中

通常，大对象 (LOB) 与引用它们的表行存储在不同的位置。但是，您可以选择以直接插入方式将长度不超过 32673 字节的 LOB 存储在基本表行中，以便简化对其进行的访问。

将大数据对象存储在基本表行中可能并不现实（根据数据的不同，可能无法进行此存储）。第 267 页的图 35 提供了尝试将 LOB 存储在表行中的示例，并说明了为何这样做会引起问题。在此示例中，行被定义为包含两个 LOB 列，其长度为 500 和 145 千

字节。但是，DB2 表的最大行大小是 32 千字节；因此，这样的行定义实际上根本无法实现。

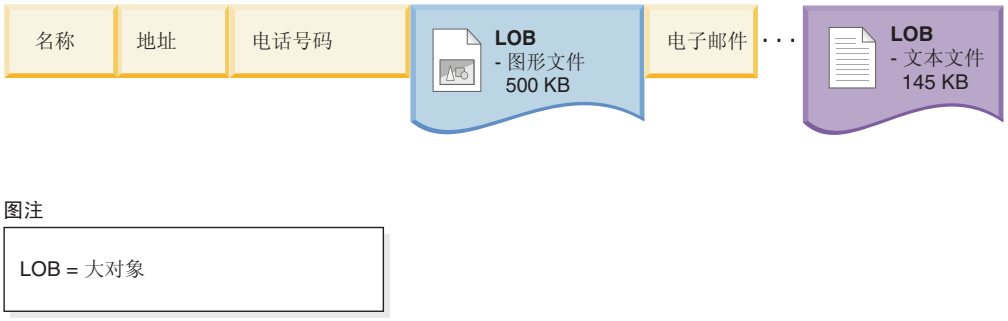


图 35. 将 LOB 数据存储到基本表行时存在的问题

为了降低使用 LOB 时的难度，处理它们的方式与处理其他数据类型不同。图 36 显示只将 LOB 描述符（而不是 LOB 本身）存储在基本表行中。每个 LOB 本身都存储在数据库管理器所控制的另一 LOB 位置。采用这种安排方式后，包含 LOB 描述符的行与包含完整 LOB 的行相比，将缩短在缓冲池与磁盘存储器之间移动它们所需的时间。

但是，由于实际的 LOB 与基本表行存储在不同的位置，因此处理 LOB 数据变得更为困难。

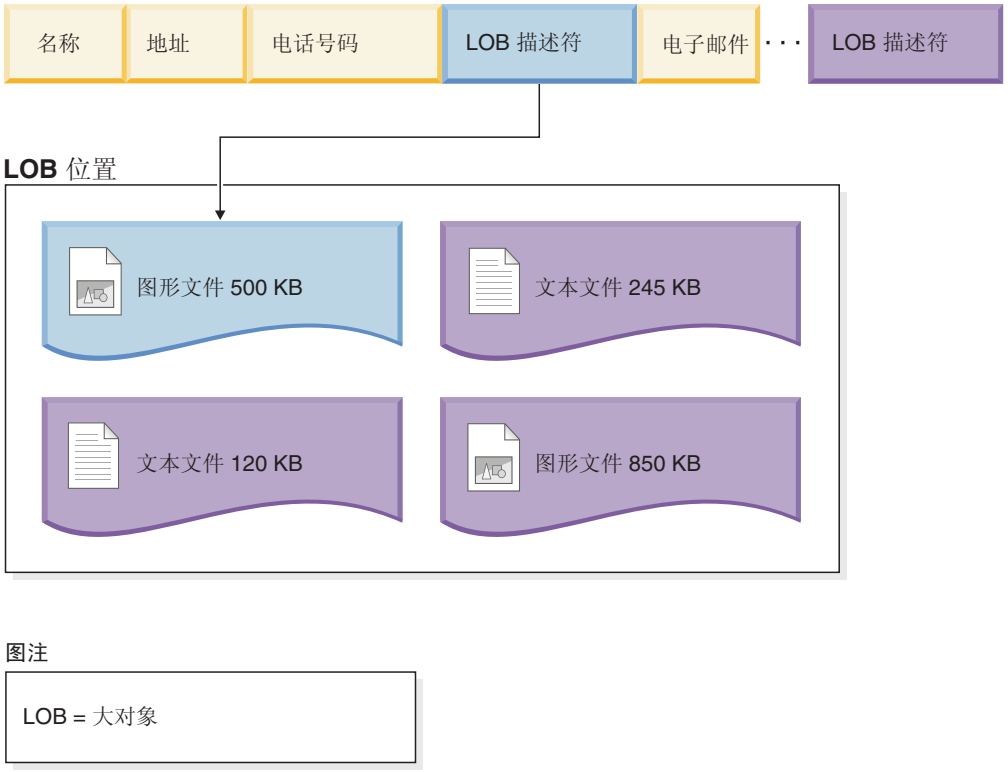


图 36. 基本表行中的 LOB 描述符引用独立 LOB 位置中的 LOB

为了简化对较小的 LOB 的处理，您可以选择将小于所指定大小阈值的 LOB 数据以直接插入方式存储在基本表行中。于是，可以将这些 LOB 数据类型作为基本表行的组成

部分进行处理，从而简化移动到缓冲池以及从缓冲池移动之类的操作。另外，如果已启用行压缩功能，那么将对直接插入式 LOB 进行行压缩。

CREATE 和 ALTER TABLE 语句的 INLINE LENGTH 选项允许将小于所指定长度限制的 LOB 数据存储在基本表行中。缺省情况下，即使未对 INLINE LENGTH 指定显式的值，小于列的最大大小 LOB 描述符的 LOB 也始终存储在基本表行中。

对于直接插入式 LOB，基本表行如图 37 所示。

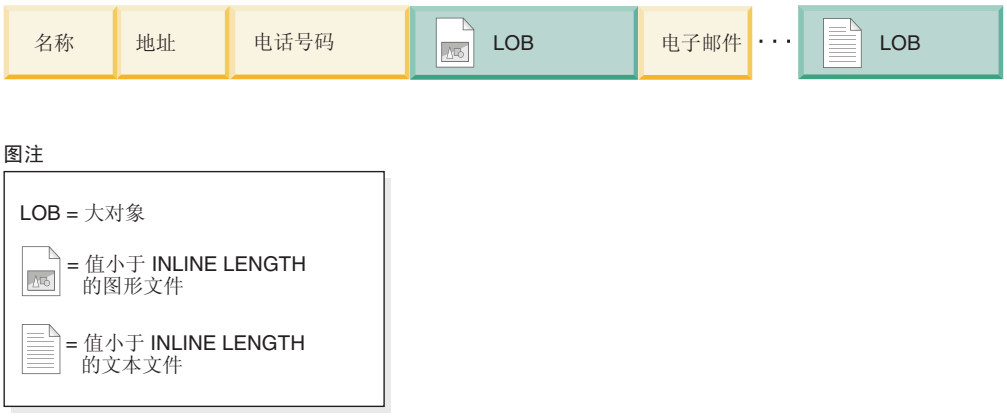


图 37. 存储在基本表行中的小型 LOB

当您考虑选择用于以直接插入方式存储 LOB 的阈值时，请考虑数据库的当前页大小以及直接插入式 LOB 是否会导致行大小超出当前页大小。表中的行的最大大小是 32677 字节。但是，每个直接插入式 LOB 都将产生需要的 4 个字节的额外存储。因此，以直接插入方式存储的每个 LOB 都将使行中的可用存储量减少 4 个字节。所以，直接插入式 LOB 的最大大小是 32673 字节。

注: 与采用直接插入方式存储 LOB 相同，还可以采用直接插入方式来存储 XML 数据。

表压缩

可通过使用 DB2 表压缩功能来减少用于存储表的磁盘空间。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。

而且，因为每页可以存储更多行，因此访问相同数据量时需要读取的页数更少。因此，针对已压缩表的查询在访问相同数据量时需要的 I/O 操作更少。因为缓冲池页上的数据行更多，所以所需行在缓冲池中的可能性增加。因此，压缩可通过已改进的缓冲池命中率来提高性能。同样，压缩还可使备份和复原操作加速，因为备份或复原相同数据量时需要传输的页更少。

可对新表和现有表使用压缩。临时表也是自动压缩的（如果数据库管理器认为这样做有利）。

可对表使用两种主要类型的数据压缩:

- 行压缩（需要具备 DB2 Storage Optimization Feature的许可证才能获得此功能）。
- 值压缩

对于特定表，可一起使用或分别使用行压缩和值压缩。但是，只能对特定表使用一种类型的行压缩。

行压缩

行压缩使用基于字典的压缩算法将数据行中重复出现的字符串替换为较短符号。

可从两种行压缩类型中进行选择:

- “经典”行压缩。
- 自适应压缩

使用行压缩需要具备 DB2 Storage Optimization Feature 许可证。根据您的 DB2 产品版本，可能包括此功能部件，也可能需要您将此功能部件作为选件单独定购。

经典行压缩:

经典行压缩（有时称为*静态压缩*）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。

使用经典行压缩的好处与自适应压缩相似，可以减少用于存储数据的空间，从而可显著节省存储成本。但是，与自适应压缩不同，经典行压缩仅使用表级别字典来存储全局重复出现的模式；它不使用那些用于以动态方式压缩数据的页级别字典。

经典行压缩的工作方式

经典行压缩使用表级别压缩字典来按行压缩数据。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中。

对哪些数据进行压缩?

存储在基本表行和日志记录中的数据符合经典行压缩资格。此外，也可以对 XML 存储对象中的数据进行压缩。可压缩直接插入到表行中的 LOB 数据；但是，不会压缩长数据对象的存储对象。

限制: 不能压缩您使用 DB2 V9.5 或 DB2 V9.1 创建的 XML 列中的数据。但是，可压缩您使用 DB2 V9.7 或更高版本添加至表的直接插入 XML 列，只要在该产品的先前发行版中创建的该表没有 XML 列。如果您在先前发行版中创建的表已有一个或多个 XML 列并且您想要通过使用 DB2 V9.7 或更高版本添加已压缩 XML 列，那么必须使用 ADMIN_MOVE_TABLE 存储过程来迁移该表，才能使用压缩。

启用或禁用经典行压缩

要使用经典行压缩，您必须具有 DB2 Storage Optimization Feature的许可证。通过将表的 COMPRESS 属性设为 YES STATIC 来压缩表数据。通过对 CREATE TABLE 语句指定 COMPRESS YES STATIC 选项，可在创建表时设置此属性。还可对 ALTER TABLE 语句使用同一选项以将现有表改变为使用压缩。启用压缩后，向该表添加数据的操作（例如，**INSERT**、**LOAD INSERT** 或 **IMPORT INSERT** 命令操作）可使用经典行压缩。而且会对表启用索引压缩。如果不另行指定并且索引是可压缩的索引类型，那么索引被创建为压缩索引。

要点: 对表启用经典行压缩时，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，请使用带有 COMPRESS NO 选项的 ALTER TABLE 语句；您以后添加的行不会压缩。要抽取整个表，必须使用 **REORG TABLE** 命令来执行表重组。

如果您有 DB2 Storage Optimization Feature 的许可证，那么可自动启用对临时表的压缩。您不能直接启用或禁用对临时表进行压缩。

更新活动对日志和已压缩表的影响

根据更新活动以及在数据行中更新的列，日志的使用可能会增加。有关如何将针对日志的更新活动的影响降至最低的信息，请参阅『第 261 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。为尽量减少指针溢出记录的形成，应使用带有 PCTFREE 选项的 ALTER TABLE 语句来提高重组后每页留为可用空间的百分比。例如，如果在启用压缩之前将 PCTFREE 选项设为 5%，那么可以在启用压缩时将其更改为 10%。提高每页留为可用空间的百分比对于大量更新的数据特别重要。

临时表的经典行压缩

临时表的压缩是使用 DB2 Storage Optimization Feature自动启用的。执行查询时，DB2 优化器将考虑对临时表进行压缩所节省的存储器容量以及对查询性能的影响，从而确定是否值得使用压缩功能。如果值得使用，那么将自动使用压缩功能。在使用压缩功能之前，临时表必须达到的最小大小比常规表的要大。

可以使用说明工具或 **db2pd** 工具来了解优化器是否对临时表使用了压缩。

回收因为压缩而释放的空间

可以回收已经通过压缩数据而释放的空间。有关更多信息，请参阅第 152 页的『可回收存储器』。

自适应压缩:

自适应压缩改进了可使用经典行压缩本身实现的压缩率。自适应压缩合并经典行压缩；但是，它也可逐页压缩以进一步压缩数据。在 DB2 产品中的各种数据压缩技巧中，自适应压缩可能节省的存储量最高。

自适应压缩的工作方式

自适应压缩实际使用两种压缩方法。第一种方法使用经典行压缩中使用的同一表级别压缩字典，以根据表中的数据抽样的重复情况来整体压缩数据。第二种方法使用基于页级别字典的压缩算法以根据每个数据页中的数据重复情况来压缩数据。这些字典将重复字节模式映射至小得多的符号；然后这些符号会替换表中的较长字节模式。表级别压缩字典存储在为其创建该字典的表对象中，用于压缩整个表的数据。页级别压缩字典与数据页中的数据存储在一起，用于仅压缩该页中的数据。有关每个字典在压缩数据时所起到的作用的更多信息，请参阅第 276 页的『压缩字典』。

注: 只能使用表级别压缩字典来指定使用经典行压缩对表进行压缩。但是，不能仅使用页级别压缩字典来指定对表进行压缩。自适应压缩同时使用表级别压缩字典和页级别压缩字典。

符合压缩资格的数据

可使用自适应压缩和经典行压缩来压缩存储在数据行*内部*的数据（包括直接插入的 LOB 或 XML 值）。可使用静态压缩来压缩 XML 存储对象。但是，不会压缩存储在表行外部的长数据对象的存储对象。此外，尽管不会压缩日志记录本身，作为插入、更新或删除操作的结果写入的日志数据量因为压缩行而减少。

限制：不能压缩您使用 DB2 V9.5 或 DB2 V9.1 创建的 XML 列中的数据。但是，可压缩您使用 DB2 V9.7 或更高版本添加至表的直接插入 XML 列，只要在该产品的先前发行版中创建的该表没有 XML 列。如果您在先前发行版中创建的表已有一个或多个 XML 列并且您想要通过使用 DB2 V9.7 或更高版本添加已压缩 XML 列，那么必须使用 ADMIN_MOVE_TABLE 存储过程来迁移该表，才能使用压缩。

启用或禁用自适应压缩

要使用自适应压缩，必须有 DB2 Storage Optimization Feature 的许可证。通过将表的 COMPRESS 属性设为 YES 来压缩表数据。通过对 CREATE TABLE 语句指定 COMPRESS YES 选项，可在创建表时设置此属性。还可对 ALTER TABLE 语句使用同一选项以将现有表改变为使用压缩。启用压缩后，向该表添加数据的操作（例如，**INSERT**、**LOAD INSERT** 或 **IMPORT INSERT** 命令操作）可使用自适应压缩。而且会对表启用索引压缩。如果不另行指定并且索引是可压缩的索引类型，那么索引被创建为压缩索引。

要点：对表启用自适应压缩时，就对整个表启用了自适应压缩，即使此表由多个表分区组成也是如此。

要对表禁用压缩，请使用带有 COMPRESS NO 选项的 ALTER TABLE 语句；您以后添加的行不会压缩。现有行将保持已压缩状态。要在关闭压缩后抽取整个表，必须使用 **REORG TABLE** 命令来执行表重组。

如果对 DB2 Storage Optimization Feature 应用许可证，并且数据库管理器认为值得，那么会自动启用针对临时表的压缩。您不能直接启用或禁用对临时表进行压缩。

更新活动对日志和已压缩表的影响

根据更新活动以及数据行中更新的位置，日志的使用可能会增加。有关表中的列顺序对更新日志记录的影响的信息，请参阅『第 261 页的『对列进行排序以使更新日志记录最少』』。

如果向一行添加新数据后行大小增大，那么当前数据页可能装不下该行的新版本。而且，行的新映像将存储在溢出页上。为尽量减少指针溢出记录的形成，应使用带有 PCTFREE 选项的 ALTER TABLE 语句来提高重组后每页留为可用空间的百分比。例如，如果在启用压缩之前将 PCTFREE 选项设为 5%，那么可以在启用压缩时将其更改为 10%。提高每页留为可用空间的百分比对于大量更新的数据特别重要。

临时表的压缩

临时表的压缩是使用 DB2 Storage Optimization Feature自动启用的。仅对临时表使用经典行压缩。

系统临时表

执行查询时，DB2 优化器将考虑对系统创建的临时表进行压缩所节省的存储器

容量以及对查询性能的影响，从而确定是否值得使用压缩功能。如果值得使用，那么将自动使用经典行压缩。在使用压缩功能之前，临时表必须达到的最小大小比常规表的要大。

用户创建的临时表

始终使用经典行压缩来压缩已创建全局临时表 (CGTT) 和已声明全局临时表 (DGTT)。

可以使用说明工具或 **db2pd** 工具来了解优化器是否对系统临时表使用了压缩。

回收因为压缩而释放的空间

可以回收已经通过压缩数据而释放的空间。有关更多信息，请参阅第 152 页的『可回收存储器』。

估算自适应压缩或经典行压缩节省的存储量

可使用 ADMIN_GET_TAB_COMPRESS_INFO 表函数来查看可为表提供的自适应压缩或经典行压缩节省的存储量的估算。

开始之前

自适应压缩或经典行压缩提供的估算节省量取决于通过运行 **RUNSTATS** 命令生成的统计信息。要最准确地估算可以节省的存储器容量，请运行 **RUNSTATS** 命令，然后执行以下步骤。

过程

要使用 ADMIN_GET_TAB_COMPRESS_INFO 表函数估算自适应压缩或经典行压缩可节省的存储量:

1. 构造使用 ADMIN_GET_TAB_COMPRESS_INFO 表函数的 SELECT 语句。 例如，对于名为 SAMPLE1.T1 的表，请输入:

```
SELECT * FROM TABLE(SYSPROC.ADMIN_GET_TAB_COMPRESS_INFO('SAMPLE1', 'T1'))
```

2. 执行此 SELECT 语句。 执行步骤 1 中所显示的语句可能会产生与以下内容相似的报告:

```
TABSCHEMA TABNAME       DBPARTITIONNUM DATAPARTITIONID OBJECT_TYPE ROWCOMPMODE ...
--------- ---------- -------------- --------------- ----------- ----------- ...
SAMPLE1   T1            0              0               DATA        A           ...

  1 record(s) selected.

PCTPAGESSAVED_CURRENT AVGROWSIZE_CURRENT PCTPAGESSAVED_STATIC ...
--------------------- ------------------ -------------------- ...
                   96                 24                   81 ...

AVGROWSIZE_STATIC PCTPAGESSAVED_ADAPTIVE AVGROWSIZE_ADAPTIVE
----------------- ---------------------- -------------------
              148                     93                  44
```

创建使用压缩功能的表

通过发出 CREATE TABLE 语句创建新表时，您可选择压缩表行中包含的数据。

开始之前

必须决定要使用的压缩类型: 自适应压缩、经典行压缩、值压缩还是值压缩与两种行压缩类型中的任一种的组合。自适应压缩和经典行压缩几乎始终能节省存储空间，因

为它们会尝试将跨多列的数据模式替换为较短符号字符串。如果许多行的列包含相同值（例如，城市或国家/地区名称），或者列包含该列的数据类型的缺省值，那么值压缩功能也能节省存储空间。

过程

要创建使用压缩的表，请发出 CREATE TABLE 语句。

- 如果要使用自适应压缩，请包括 COMPRESS YES ADAPTIVE 子句。
- 如果要使用经典行压缩，请包括 COMPRESS YES STATIC 子句。
- 如果要使用值压缩功能，请指定 VALUE COMPRESSION 子句。如果要压缩表示系统缺省列值的数据，还应包括 COMPRESS SYSTEM DEFAULT 子句。

结果

创建该表后，您从该时间点开始添加至该表的所有数据都会被压缩。与该表相关联的所有索引也会被压缩，除非您使用 CREATE INDEX 或 ALTER INDEX 语句的 COMPRESS NO 子句另行指定。

示例

示例 1: 以下语句创建客户信息表并启用自适应压缩。在此示例中，该表将使用表级别压缩字典和页级别压缩字典进行压缩。

```
CREATE TABLE CUSTOMER
     (CUSTOMERNUM      INTEGER,
      CUSTOMERNAME     VARCHAR(80),
      ADDRESS          VARCHAR(200),
      CITY             VARCHAR(50),
      COUNTRY          VARCHAR(50),
      CODE             VARCHAR(15),
      CUSTOMERNUMDIM   INTEGER)
      COMPRESS YES ADAPTIVE;
```

示例 2: 以下语句创建客户信息表并启用经典行压缩。在此示例中，该表仅使用表级别压缩字典进行压缩。

```
CREATE TABLE CUSTOMER
     (CUSTOMERNUM      INTEGER,
      CUSTOMERNAME     VARCHAR(80),
      ADDRESS          VARCHAR(200),
      CITY             VARCHAR(50),
      COUNTRY          VARCHAR(50),
      CODE             VARCHAR(15),
      CUSTOMERNUMDIM   INTEGER)
      COMPRESS YES STATIC;
```

示例 3: 以下语句创建职员薪水表。SALARY 列的缺省值为 0，并且对该列指定了行压缩和系统缺省压缩。

```
CREATE TABLE EMPLOYEE_SALARY
     (DEPTNO   CHAR(3)      NOT NULL,
      DEPTNAME VARCHAR(36)  NOT NULL,
      EMPNO    CHAR(6)      NOT NULL,
      SALARY   DECIMAL(9,2) NOT NULL WITH DEFAULT COMPRESS SYSTEM DEFAULT)
      COMPRESS YES ADAPTIVE;
```

请注意，此语句省略了 VALUE COMPRESSION 子句。此语句创建名为 EMPLOYEE_SALARY 的表，但返回警告消息:

```
SQL20140W 由于已对此表取消激活 VALUE COMPRESSION，因此 COMPRESS
列属性被忽略。SQLSTATE=01648
```

在这种情况下，未对 SALARY 列应用 COMPRESS SYSTEM DEFAULT 子句。

示例 4: 以下语句创建职员薪水表。SALARY 列的缺省值为 0，并且对该列启用了行压缩和系统缺省压缩。

```
CREATE TABLE EMPLOYEE_SALARY
     (DEPTNO   CHAR(3)      NOT NULL,
      DEPTNAME VARCHAR(36)  NOT NULL,
      EMPNO    CHAR(6)      NOT NULL,
      SALARY   DECIMAL(9,2) NOT NULL WITH DEFAULT COMPRESS SYSTEM DEFAULT)
      VALUE COMPRESSION COMPRESS YES ADAPTIVE;
```

在此示例中，该语句包含 VALUE COMPRESSION 子句，这会压缩 SALARY 列的缺省值。

在现有的表中启用压缩功能

通过使用 ALTER TABLE 语句，可修改现有表以利用压缩的节省存储空间的优点。

开始之前

必须决定要使用的压缩类型：自适应压缩、经典行压缩、值压缩还是值压缩与两种行压缩类型中的任一种的组合。自适应压缩和经典行压缩几乎始终能节省存储空间，因为它们会尝试将跨多列的数据模式替换为较短符号字符串。如果许多行的列包含相同值（例如，城市或国家/地区名称），或者列包含该列的数据类型的缺省值，那么值压缩功能也能节省存储空间。

过程

要在现有表中启用压缩，请执行以下操作:

1. 发出 ALTER TABLE 语句。
   - 如果要使用自适应压缩，请包括 COMPRESS YES ADAPTIVE 子句。
   - 如果要使用经典行压缩，请包括 COMPRESS YES STATIC 子句。
   - 如果要使用值压缩，请为包含要压缩的值的每个列添加 ACTIVATE VALUE COMPRESSION 子句。如果要压缩包含系统缺省值的列中的数据，还应包括 COMPRESS SYSTEM DEFAULT 子句。

   您后续追加、插入、装入或更新的所有行将使用新的压缩格式。
2. 可选: 要立即对表的所有现有行应用压缩，请使用 **REORG TABLE** 命令来执行表重组。如果此时不对所有行应用压缩，那么直到下一次您更新未压缩行或下一次运行 REORG TABLE 命令，这些行才会以新压缩格式存储。

示例

示例 1: 以下语句对名为 CUSTOMER 的现有表应用自适应压缩:

```
ALTER TABLE CUSTOMER COMPRESS YES ADAPTIVE
```

示例 2: 以下语句对名为 CUSTOMER 的现有表应用经典行压缩:

```
ALTER TABLE CUSTOMER COMPRESS YES STATIC
```

示例 3: 以下语句对名为 EMPLOYEE_SALARY 的现有表的 SALARY 列应用行压缩、值压缩和系统缺省值压缩。然后重组该表。

```
ALTER TABLE EMPLOYEE_SALARY
ALTER SALARY COMPRESS SYSTEM DEFAULT
COMPRESS YES ACTIVATE VALUE COMPRESSION;

REORG TABLE EMPLOYEE_SALARY
```

对已压缩表更改或禁用压缩

通过使用 ALTER TABLE 语句的各种压缩相关子句中的一个或多个，可更改表的压缩方式，或对启用了自适应压缩、经典行压缩或值压缩的表完全禁用压缩。

关于此任务

如果取消激活自适应压缩或经典行压缩，那么索引压缩不会受影响。如果要对索引解压缩，必须使用 ALTER INDEX 语句。

过程

要取消激活针对表的压缩或从一种类型的行压缩更改为另一种类型的行压缩，请执行以下操作:

1. 发出 ALTER TABLE 语句。
   - 如果要取消激活自适应压缩或经典行压缩，请包括 COMPRESS NO 子句。
   - 如果要更改为另一种类型的行压缩，请使用 COMPRESS YES ADAPTIVE 或 COMPRESS YES STATIC 子句来指定想要的压缩类型。例如，如果表当前使用经典行压缩，并且您想要更改为自适应压缩，请执行带 COMPRESS YES ADAPTIVE 子句的 ALTER TABLE 语句
   - 如果要取消激活值压缩功能，请指定 DEACTIVATE VALUE COMPRESSION 子句。
   - 如果要取消激活对系统缺省值进行的压缩功能，请对 ALTER *column name* 子句指定 COMPRESS OFF 选项。
2. 使用 **REORG TABLE** 命令来执行脱机表重组。

结果

- 如果使用 COMPRESS NO 子句关闭行压缩，那么所有行数据都会解压缩。
- 如果从一种类型的行压缩更改为另一种类型的行压缩，那么系统将使用您在 ALTER TABLE 语句中指定的行压缩类型压缩整个表。（请参阅示例 2。）
- 取消激活值压缩具有以下效果:
  - 如果表包含启用了 COMPRESS SYSTEM DEFAULT 的列，那么系统不再对这些列启用压缩。
  - 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。如果发生此情况，那么会返回错误消息 SQL0670N。

示例

示例 1: 关闭行压缩: 以下语句在名为 CUSTOMER 的表中关闭自适应压缩或经典行压缩，然后重组该表以对先前已压缩的数据解压缩:

```
ALTER TABLE CUSTOMER COMPRESS NO
REORG TABLE CUSTOMER
```

示例 2: 从静态压缩更改为自适应压缩: 假定 SALES 表当前使用经典行压缩。以下语句将使用的压缩类型更改为自适应压缩:

```
ALTER TABLE SALES COMPRESS ADAPTIVE YES
REORG TABLE SALES
```

压缩字典

数据库管理器为已启用自适应压缩或经典行压缩的每个表创建表级别压缩字典。对于已启用自适应压缩的表，数据库管理器还会创建页级别压缩字典。

这两种类型的字典用于将重复字节模式从表行映射至小得多的符号；然后这些符号会替换表行中的较长字节模式。

表级别压缩字典

要构建表级别字典，系统会扫描表以查找重复模式。系统将检查所有行（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，数据库管理器将构建一个压缩字典，并为这些条目指定简短的数字键。通常来讲，文本字符串的压缩机率高于数字数据；压缩数字数据涉及将一个数字替换为另一个数字。根据要替换的数字的大小，节省的存储空间可能不像压缩文本时节省的存储空间那么多。

第一次创建表级别字典时，系统使用表中的数据样本来构建该字典。除非您显式指示使用经典脱机表重组来重建字典，否则系统不会再次更新该字典。即使您重建该字典，该字典也只反映整个表中的数据的样本。

切记: 表级别字典是静态字典；除非您手动重建该字典，否则它在初始创建后不会更改。即使您重建该字典（因为用于创建该字典的抽样技巧），它可能也不会反映单页内重复出现的字符串。

表级别压缩字典存储在应用该字典的对象中的隐藏行中，并且会高速缓存在内存中以便快速访问。此字典不会占用太多空间。即使对于极大的表，压缩字典通常也只占用大约 100 KB 空间。

页级别压缩字典

除了表级别字典，自适应压缩还使用页级别字典。但是，与表级别字典不同，页级别字典是在数据库管理器填充页时自动创建或重新创建的。与表级别压缩字典一样，页级别字典也存储在表内的隐藏行中。

创建表级别压缩字典

可对启用了自适应压缩或经典行压缩的表自动或手动构建表级别压缩字典。您启用了自适应压缩的表包括页级别数据字典，这些字典始终是自动创建的。

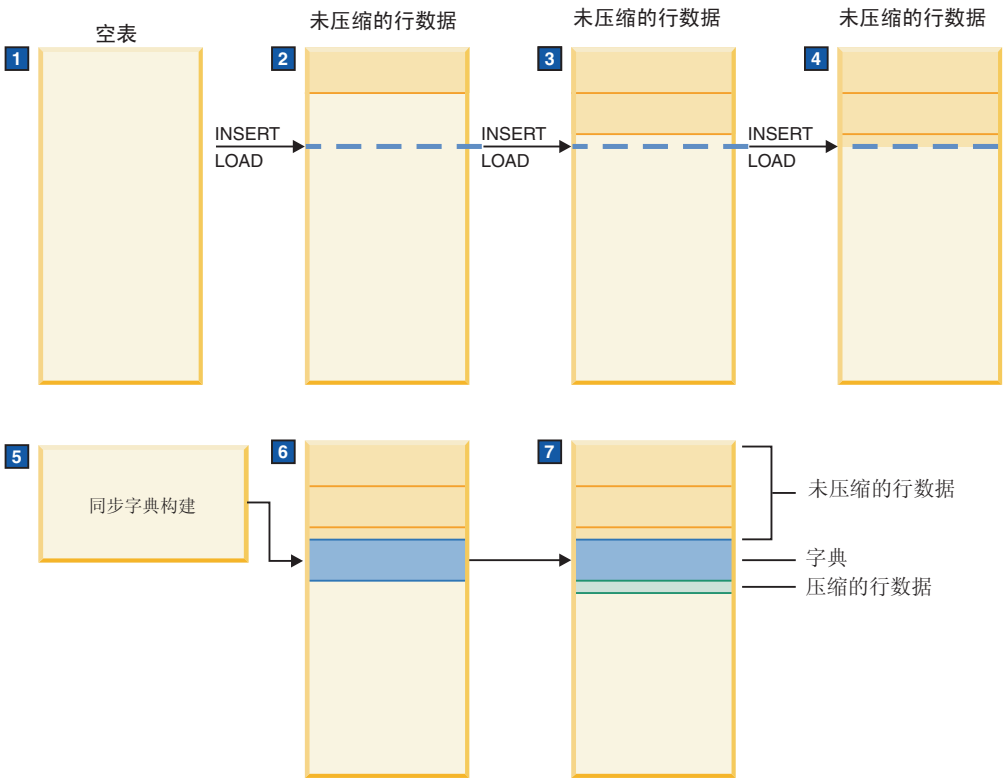
自动创建字典

从 DB2 V9.5 开始，如果满足下面的每个条件，那么将自动创建表级别压缩字典:

- 通过使用带 COMPRESS YES ADAPTIVE 或 COMPRESS YES STATIC 子句的 CREATE TABLE 或 ALTER TABLE 语句为该表设置 COMPRESS 属性。
- 该表尚不存在表级别压缩字典。
- 该表包含足够数据以构造重复数据的字典。

如果压缩保持已启用状态，那么系统会使用该字典压缩创建该字典后移至表的数据。

下图显示了自动创建压缩字典的过程:



图中显示的事件顺序如下所示:

1. 尚未创建压缩字典，因为表为空。
2. 数据插入到表中（通过使用插入或装入操作）并保持未压缩状态。
3. 随着更多数据插入或装入到表中，数据保持未压缩状态。
4. 如果 COMPRESS 属性设为 YES ADAPTIVE 或 YES STATIC，那么达到阈值后系统会自动触发字典创建。
5. 创建了字典。
6. 已将字典追加至表。
7. 从此时开始，系统会启用表级别压缩，并且后续插入或添加的行将由表级别压缩字典压缩。

要点: 创建该字典前表中存在的行将保持未压缩状态，除非您更改这些行或手动重建该字典。

如果使用 DB2 V9.7 或更高版本创建表，并且该表包含类型为 XML 的至少一列，那么系统会创建另一压缩字典。此字典用于压缩存储在与该表相关联的缺省 XML 存储器对象中的 XML 数据。如果符合下列每个条件，那么系统会自动为 XML 数据创建压缩字典:

- 您将表的 COMPRESS 属性设为 YES ADAPTIVE 或 YES STATIC。
- 该 XML 存储器对象中不存在压缩字典。
- XML 存储器对象中有足够的数据。

限制: 不能压缩您使用 DB2 V9.5 或 DB2 V9.1 创建的 XML 列中的数据。但是，可压缩您使用 DB2 V9.7 或更高版本添加至表的直接插入 XML 列，只要在该产品的先前发行版中创建的该表没有 XML 列。如果您在先前发行版中创建的表已有一个或多个 XML 列并且您想要通过使用 DB2 V9.7 或更高版本添加已压缩 XML 列，那么必须使用 ADMIN_MOVE_TABLE 存储过程来迁移该表，才能使用压缩。

用于为临时表创建表级别压缩字典的机制与用于永久表的机制类似。但是，数据库管理器将根据诸如查询复杂程度和结果集大小之类的因素来自动确定是否对临时表使用经典行压缩。

手动创建字典

尽管启用了压缩的表增大到足够大小时就会自动创建字典，但如果不存在压缩字典，那么您也可以强制创建表级别压缩字典（通过使用带有 **RESETDICTIONARY** 参数的 **REORG TABLE** 命令）。如果表中至少有一行数据，那么此命令将强制创建压缩字典。对表进行重组是一项脱机操作；使用“自动创建字典”这项功能的一个好处就是，构建字典时，表仍然处于联机状态。

您还可以使用带有 **ROWCOMPESTIMATE** 参数的 **INSPECT** 命令创建新字典，而不是使用 **REORG TABLE** 命令来强制创建新字典。如果表没有字典，那么此命令就会创建压缩字典。与执行表重组相比，使用此方法的优点在于，创建字典时该表仍然保持联机状态。系统将压缩您后续添加的行；但不会压缩在运行 **INSPECT** 命令之前就已存在的行，直到您执行表重组。但是，如果启用压缩，那么激活压缩后通常很快自动创建字典，就像您以前有机会使用 **INSPECT** 命令一样。

重置压缩字典

无论是自动还是手动创建表级别压缩字典，字典均为静态；在构建之后也不会更改。添加或更新行时，将根据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，考虑数据库中用于维护城市自来水公司客户帐户的表。这样一个表中可能有 STREET_ADDRESS、CITY、PROVINCE、TELEPHONE_NUM、POSTAL_CODE 和 ACCOUNT_TYPE 等列。如果使用这类表中的数据构建了压缩字典，那么即使它只是中等大小的表，也可能存在足够多的重复信息，因此对它进行经典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公共数据，例如 CITY、POSTAL_CODE 和 PROVINCE 列的值或 STREET_ADDRSS 或 TELEPHONE_NUM 列的值的某些部分。

但是，其他表可能会随时间推移而显著更改。考虑用于零售数据的表，如下所示:

- 主表用于按月累积数据。
- 每个月，会将一组新记录装入到表中。

在这种情况下，在某个月份（例如，四月）创建的压缩字典可能不会反映当年后续时间的重复销售数据。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** 参数的 **REORG TABLE** 命令来重置压缩字典。重置压缩字典的优点是，构建该字典时会考虑整个表中的数据。

经典表重组对表级别压缩字典的影响

如果使用经典脱机表重组来重组已启用自适应压缩或经典行压缩的表，那么可以保留表级别压缩字典，也可以强制数据库管理器创建新的压缩字典。

在 DB2 V9.5 及更高版本中，系统通过使用带有 **COMPRESS YES** 子句的 CREATE TABLE 或 ALTER TABLE 语句为启用了自适应压缩或经典行压缩的表自动创建表级别压缩字

典。对于新表，数据库管理器将一直等到该表增大到最小大小之后才会创建字典。对于现有表，将在表增大到足够大小以在明显存在模式重复现象时创建压缩字典。仅对您在启用压缩功能之后插入或更新的行应用压缩。

如果您使用经典表重组来重组表，并且表级别压缩字典已存在，那么将隐式应用 **REORG TABLE** 命令的 **KEEPDICTIONARY** 参数，这会保留该字典。当您执行重组时，处理的所有行将使用该字典进行压缩。如果不存在压缩字典并且表足够大，那么将创建压缩字典，并且这些行使用该字典进行压缩。

可以通过执行使用 **REORG TABLE** 命令的 **RESETDICTIONARY** 参数的经典表重组来强制构建新的表级别压缩字典。当您指定 **RESETDICTIONARY** 参数时，如果表中至少有一行，那么会构建新的压缩字典，用来替换现有的任何字典。

注：可仅使用经典表重组来重建表级别字典。如果尝试对已使用页级别压缩字典压缩任何行的表执行原位置表重组，那么 REORG 命令将失败，并且发生 SQL2219 错误。

用于复制源表的多个压缩字典

对于 CREATE TABLE 和 ALTER TABLE 语句，可将 DATA CAPTURE CHANGES 子句与 COMPRESS YES STATIC 或 COMPRESS YES ADAPTIVE 选项组合到一起以对要复制的源表启用行压缩。

启用压缩时，如果还在命令 **REORG TABLE** 或 **LOAD REPLACE** 中指定了 DATA CAPTURE CHANGES 子句，那么源表可有两个表级别压缩字典：一个处于活动状态的*表级别压缩字典*和一个*历史压缩字典*。换言之，如果启用了 DATA CAPTURE CHANGES，那么当您运行 **REORG TABLE** 或 **LOAD REPLACE** 命令时，不会替换表级别压缩字典。反而会生成新字典，之前字典会保留。

历史压缩字典使得 db2ReadLog API 能够对重建该活动字典（因为使用 **REORG TABLE** 或 **LOAD** 命令指定了 **RESETDICTIONARY** 选项）之前写入的日志记录中的行内容解压缩。

注：要让日志阅读器以非压缩格式（而不是原始压缩格式）返回日志记录中的数据，必须将 db2ReadLog API 的 **iFilterOption** 参数设为 DB2READLOG_FILTER_ON。

如果在用于创建表的 CREATE TABLE 语句中指定了 DATA CAPTURE NONE 选项，那么发出 **REORG TABLE** 命令或执行表截断操作（通过发出 **LOAD REPLACE**、**IMPORT REPLACE** 或 **TRUNCATE TABLE** 命令）会除去该表的历史压缩字典。

要查看该表是否存在历史字典，请检查 ADMIN_GET_TAB_DICTIONARY_INFO 表函数的结果集中的 HISTORICAL_DICTIONARY 列。

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可变长度数据类型（VARCHAR、VARGRAPHICS、LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB 和 DBCLOB）的 NULL 和零长度数据存储在磁盘上。

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、固定长度字符和固定长度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。

乐观锁定概述

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。

表 23 列示了每个类别中的相关主题。

表 23. 乐观锁定信息的概述

类别	相关主题
一般信息和限制	• 第 281 页的『乐观锁定』 • 第 283 页的『行更改标记和被动错误信息的详细程度』 • 第 282 页的『乐观锁定限制和注意事项』
基于时间的更新	• 第 284 页的『基于时间的更新检测』 • 第 285 页的『为 ROW CHANGE TIMESTAMP 生成的时间值』
启用	• 第 287 页的『计划启用乐观锁定』 • 第 288 页的『在应用程序中启用乐观锁定』

表 23. 乐观锁定信息的概述 (续)

类别	相关主题
使用方案	• 第 350 页的『方案: 乐观锁定和基于时间的检测』 – 第 350 页的『方案: 在应用程序中使用乐观锁定』 – 第 353 页的『方案: 基于时间的更新检测』 – 第 352 页的『方案: 使用隐式隐藏列的乐观锁定』
参考信息	• 第 285 页的『RID_BIT() 和 RID() 内置函数』 • *SQL Reference Volume 1*中的 ALTER TABLE 语句 • *SQL Reference Volume 2*中的 CREATE TABLE 语句 • *SQL Reference Volume 2*中的 DELETE 语句 • *SQL Reference Volume 2*中的 SELECT 语句 • *SQL Reference Volume 2*中的 UPDATE 语句

注: 在整个乐观锁定主题中，每次提及插入或更新行时，这指的是可以通过任何方式将行插入到表中或更新行的所有形式的 SQL 语句。例如，INSERT、UPDATE、MERGE 或者甚至 DELETE 语句（带有引用约束）都可以导致创建或更新时间戳记列。

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

下列新增的 SQL 函数、表达式和功能支持更容易且速度更快的乐观锁定，这种乐观锁定不会产生*主动错误信息*:

- 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 ROW CHANGE TOKEN 值与上次访存行时存储的 ROW CHANGE TOKEN 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 RID_BIT() 和 ROW CHANGE TOKEN 添加至 SQL 的。要支持此功能，需要在表中定义一个新生成的列，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该列。该列是否存在也会影响乐观锁定的行为，这是因为，如果该列用来将 ROW CHANGE TOKEN 的粒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 版也已增加此功能。

隐式隐藏列:

为了兼容，此功能不要求现有的表和应用程序使用 RID_BIT 和 ROW CHANGE TOKEN 列。使用隐式列列表时，未外部化隐式隐藏列。例如:

- 对表执行 SELECT * 不会在结果表中返回隐式隐藏列
- 不使用列列表的 INSERT 语句不期望隐式隐藏列的值，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，*乐观并行控制*、*悲观锁定*、*ROWID* 和*更新检测*）的定义。

乐观锁定限制和注意事项

本主题列示您必须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
  - 表分区列
  - 数据库散列分区键
  - DETERMINED BY 约束列
  - 昵称
- 分区数据库配置中不支持 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应该处理这种情况。
- 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列。可以在引用约束中引用隐式隐藏列。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列，那么该列将是具体化查询表的一部分。否则，隐式隐藏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，隐式隐藏列不是视图的一部分，该视图引用包含隐式隐藏列的表。

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，如果更改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，使用以下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 ROW CHANGE TOKEN 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。ROW CHANGE TIMESTAMP 表达式提供此信息。

```
ROW CHANGE TIMESTAMP FOR table-designator
```

返回用类似于 CURRENT TIMESTAMP 的本地时间表示的时间戳记，表示上次更改行的时间。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。

ROW CHANGE TIMESTAMP 的值与 CURRENT TIMESTAMP 的不同之处在于，数据库对每个数据库分区中的每一行指定此值时保证它的唯一性。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。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，在表中必须定义行更改时间戳记列，并且该列必须具有时间戳记数据类型的缺省精度 TIMESTAMP(6)（或者 TIMESTAMP - 缺省精度为 6）。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，使用的 CREATE TABLE 命令包含 FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP 子句。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，未在其中定义行更改时间戳记列，但后来使用 ALTER TABLE 语句的 FOR EACH ROW ON UPDATE AS ROW CHANGE TIMESTAMP 子句对其添加了该列。ROW CHANGE TIMESTAMP 表达式返回该列的值。对于此类别，先前更改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。带有 **INPLACE** 参数的 **REORG TABLE** 操作不够，因为它未具体化模式更改。

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 TABLE** 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，如果系统时钟赶上行更改时间戳记值，那么会有一个插入行的近似时间。在此时间之前，将按 timestamp(6) 数据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 ROW CHANCE TOKEN。可以在应用程序需要的任何隔离级别进行选择。

通过执行以下操作，应用程序可以借助乐观锁定来修改相同（未更改的）行:

- 搜索 RID_BIT() 以直接访问（不扫描）目标行
- ROW CHANGE TOKEN，以确保这是同一个未更改的行

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列示的两个值。

“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行冲突。“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，这样乐观搜索的 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 UPDATE 或 DELETE 语句（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 for z/OS 实现了返回类型为 BIGINT 的内置函数 RID，但这对于 Linux、UNIX 和 Windows RID 而言不够大。为了实现兼容，此 RID() 内置函数不仅返回 RID_BIT()，还返回 BIGINT。

此 RID() 内置函数不在分区数据库环境中工作，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，*带有值的乐观锁定*是使用效率较低的访问方法（通过将所有选中列值添加至谓词并依赖于某些唯一列组合以仅限定单行）完成的。

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 ROW CHANGE TOKEN 值与上次访存行时存储的 ROW CHANGE TOKEN 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 ROW CHANGE TOKEN 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 283 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以及 RID_BIT() 和 RID() 函数

ROW CHANGE TIMESTAMP 表达式返回时间戳记值，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。尽管 RID_BIT() 和 RID() 内置函数以及基于时间的更新检测功能相互相关，但一定要注意不能交换使用 ROW CHANGE TOKEN 和 ROW CHANGE TIMESTAMP 表达式；具体而言，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
- 更改后包含 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很少。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行应用程序产生的。

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 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，必须执行一些步骤。

过程

1. 在初始查询中，选择需要处理的每行的行标识（使用 第 285 页的『RID_BIT() 和 RID() 内置函数』）和 ROW CHANGE TOKEN。
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 ROW CHANGE TOKEN 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。

下一步做什么

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 ROW CHANGE TOKEN 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，请创建或更改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 IMPLICITLY 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。

声明临时表

要在应用程序中定义临时表，请使用 DECLARE GLOBAL TEMPORARY TABLE 语句。

关于此任务

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。在声明临时表之前，必须存在用户临时表空间。

注: 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。
临时表不支持:

- 用户定义的类型列
- LONG VARCHAR 列
- 已创建全局临时表的 XML 列

示例

```
DECLARE GLOBAL TEMPORARY TABLE temptbl
   LIKE empltabl
   ON COMMIT DELETE ROWS
   NOT LOGGED
   IN usr_tbsp
```

此语句定义名为 temptbl 的临时表。对此表定义的列名和列描述与 empltabl 的列名和列描述完全相同。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。借助 ON COMMIT DELETE ROWS（任何 DELETE ROWS 选项），数据库管理器始终删除行，而无论是否已经以 HOLD 方式对该表打开游标。如果未打开 WITH HOLD 游标，那么数据库管理器将通过实现内部 TRUNCATE 优化 NOT LOGGED 删除，否则以每次一行的方式删除行。

在应用程序与数据库断开连接时，此表将被隐式删除。有关更多信息，请参阅 DECLARE GLOBAL TEMPORARY TABLE 语句。

创建以及连接到已创建临时表

*已创建临时表*通过 CREATE GLOBAL TEMPORARY TABLE 语句进行创建。应用程序第一次使用连接来引用已创建临时表时，会将这个已创建临时表的一个专用版本实例化，以供使用该连接的应用程序使用。

关于此任务

与已声明临时表类似，已创建临时表由需要处理数据库数据并且需要将数据处理结果暂时存储在表中的应用程序使用。但是，已声明临时表信息*并不会*保存在系统目录表中，每个使用该信息的会话都必须定义该信息；相反，已创建临时表信息却*保存在*系统目录中并且不需要在每个使用该信息的会话中进行定义，从而使其具有持久性并能够通过不同的连接与其他应用程序共享。在可以创建已创建临时表之前，必须存在用户临时表空间。

注: 任何使用该连接的程序所执行的第一个对已创建临时表的隐式或显式引用都将创建给定已创建临时表的空实例。每个引用这个已创建临时表的连接都有自己的唯一已创建临时表实例，该实例在该连接结束后不再存在。

各个连接对已创建临时表名的引用将引用同一个持久已创建临时表定义，但各个连接将引用当前服务器上不同的已创建临时表实例。如果未对所引用的已创建临时表名进行限定，那么将使用应用于 SQL 语句的标准限定规则隐式地对其进行限定。

所有者隐式地对已创建临时表拥有所有表特权，并且有权删除该表。所有者的表特权可以被授予和撤销（可以逐个特权地进行授予和撤销，也可以通过 ALL 子句进行）。另一个授权标识只有被授予已创建临时表的适当特权之后，才能访问该表。

支持索引以及用于修改数据的 SQL 语句（例如 INSERT、UPDATE 和 DELETE）。只能在已创建临时表所在的表空间中创建索引。

对于 CREATE GLOBAL TEMPORARY TABLE 语句: 锁定和恢复功能不适用; 仅当指定了 LOGGED 子句时，日志记录功能才适用。要了解更多选项，请参阅 CREATE GLOBAL TEMPORARY 语句。

创建临时表不能:

- 与安全策略相关联
- 是分区表
- 是多维集群 (MDC) 表
- 是插入时间集群 (ITC) 表
- 是范围集群 (RCT) 的
- 由复制分发

不能创建基于已创建临时表的具体化查询表 (MQT)。

已创建临时表不支持以下列类型、对象类型以及表或索引操作:

- XML 列
- 结构类型
- 引用类型
- 约束
- 索引扩展
- LOAD
- LOAD TABLE
- ALTER TABLE
- RENAME TABLE
- RENAME INDEX
- REORG TABLE
- REORG INDEX
- LOCK TABLE

有关更多信息，请参阅 CREATE GLOBAL TEMPORARY TABLE 语句。

示例

```
CREATE GLOBAL TEMPORARY TABLE temptbl
   LIKE empltabl
   ON COMMIT DELETE ROWS
   NOT LOGGED
   IN usr_tbsp
```

此语句创建名为 temptbl 的临时表。对此表定义的列名和列描述与 empltabl 的列名和列描述完全相同。隐式定义只包括 empltabl 中的各个列的列名、数据类型、可空性特征和列缺省值属性。未以隐式方式定义所有其他列属性，其中包括唯一约束、外键约束、触发器和索引。

COMMIT 始终从表中删除行。如果已对该表打开任何 HOLD 游标，那么可以使用 TRUNCATE 语句删除那些游标（速度较快），但在“正常情况”下，必须逐行进行删除。系统不记录对临时表所作的更改。临时表将被放入指定的用户临时表空间 usr tbsp。此表空间必须存在，否则创建此表将失败。

将已创建临时表实例化的应用程序与数据库断开连接时，该应用程序的已创建临时表实例将被删除。

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。

开始之前

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）
- DBADM 权限

关于此任务

如果尝试更正不匹配情况失败，将返回 SQL20408N 或 SQL20307N 错误。

过程

要创建新的源表，请执行以下操作:

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** 的输出中除去分区子句，将创建的表的名称更改为新名称（例如，“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。登台表收集必须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，不必完全重新生成具体化查询表。

关于此任务

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，除非执行了其他某些操作。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 语句有目的地修改登台表。修改会将登台表更改为不一致状态。例如，下列语句强制修改称为 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

DB2 基本表与临时表之间的差别

DB2 基本表与两类临时表之间存在多项差别。

下表对基本表、已创建临时表和已声明临时表之间的重要差别作了概述。

表 24. DB2 基本表与 DB2 临时表之间的重要差别

差别领域	差别
创建、持久性以及共享表描述的能力	**基本表**：CREATE TABLE 语句将表的描述放入目录视图 SYSCAT.TABES。表描述具有持久性，并且可以在不同连接之间进行共享。在 CREATE TABLE 语句中，可以对表名进行限定。如果未对表名进行限定，那么将使用应用于 SQL 语句的标准限定规则隐式地对其进行限定。
	已创建临时表：CREATE GLOBAL TEMPORARY TABLE 语句将表的描述放入目录视图 SYSCAT.TABES。表描述具有持久性，并且可以在不同连接之间进行共享。在 CREATE GLOBAL TEMPORARY TABLE 语句中，可以对表名进行限定。如果未对表名进行限定，那么将使用应用于 SQL 语句的标准限定规则隐式地对其进行限定。
	已声明临时表：DECLARE GLOBAL TEMPORARY TABLE 语句不将表的描述放入目录。表描述在发出 DECLARE GLOBAL TEMPORARY TABLE 语句的连接结束后不再存在，所以只有该连接才知道该描述。 因此，对于同一个已声明临时表，每个连接都可以有自己的有可能唯一的描述。在 DECLARE GLOBAL TEMPORARY TABLE 语句中，可以对表名进行限定。如果对表名进行限定，那么必须将 SESSION 用作模式限定符。如果未对表名进行限定，那么将隐式地使用 SESSION 作为限定符。
表实例化和共享数据能力	**基本表**：CREATE TABLE 语句创建表的一个空实例，所有连接都使用表的该实例。表和数据具有持久性。
	已创建临时表：CREATE GLOBAL TEMPORARY TABLE 语句不创建表的实例。在任何使用该连接的程序所执行的打开、选择、插入、更新或删除操作中，对该表的第一个隐式或显式引用都将创建给定表的空实例。每个引用该表的连接都有自己的唯一表实例，该实例在该连接结束后不再存在。
	已声明临时表：DECLARE GLOBAL TEMPORARY TABLE 语句将为连接创建表的空实例。每个声明该表的连接都有自己的唯一表实例，该实例在该连接结束后不再存在。
连接期间对表的引用	**基本表**：各个连接对表名的引用将引用同一个持久表描述以及当前服务器上的同一个实例。如果未对所引用的表名进行限定，那么将使用应用于 SQL 语句的标准限定规则隐式地对其进行限定。
	已创建临时表：各个连接对表名的引用将引用同一个持久表描述，但各个连接将引用当前服务器上不同的表实例。如果未对所引用的表名进行限定，那么将使用应用于 SQL 语句的标准限定规则隐式地对其进行限定。
	已声明临时表：在多个连接中，各个连接对表名的引用将引用当前服务器上不同的表描述和表实例。在除 DECLARE GLOBAL TEMPORARY TABLE 语句以外的 SQL 语句中，对表名的引用必须包括 SESSION 作为模式限定符。如果未通过 SESSION 对表名进行限定，那么将假定引用基本表。

表 24. DB2 基本表与 DB2 临时表之间的重要差别 (续)

差别领域	差别
表特权和授权	**基本表**: 所有者隐式地对表拥有所有表特权，并且有权删除该表。所有者的表特权可以被授予和撤销（可以逐个特权地进行授予和撤销，也可以通过 ALL 子句进行）。 另一个授权标识只有被授予该表的适当特权之后，才能访问该表。
	已创建临时表: 所有者隐式地对表拥有所有表特权，并且有权删除该表。所有者的表特权可以被授予和撤销（可以逐个特权地进行授予和撤销，也可以通过 ALL 子句进行）。 另一个授权标识只有被授予该表的适当特权之后，才能访问该表。
	已声明临时表: PUBLIC 隐式地对表拥有除 GRANT 权限以外的所有表特权，并且还有权删除该表。这些表特权无法被授予或撤销。 任何授权标识都能够访问该表，而不要求授予对该表的任何特权。
索引和其他 SQL 语句支持	**基本表**: 支持索引以及用于修改数据的 SQL 语句（INSERT、UPDATE 和 DELETE 等等）。索引可以在不同的表空间中。
	已创建临时表: 支持索引以及用于修改数据的 SQL 语句（INSERT、UPDATE 和 DELETE 等等）。索引只能在表所在的表空间中。
	已声明临时表: 支持索引以及用于修改数据的 SQL 语句（INSERT、UPDATE 和 DELETE 等等）。索引只能在表所在的表空间中。
锁定、日志记录和恢复	**基本表**: 支持锁定、日志记录和恢复功能。
	已创建临时表: 不支持锁定和恢复功能，但显式地指定了 LOGGED 时，支持日志记录功能。仅当*显式地*指定了 LOGGED 时，才支持撤销恢复功能（将更改回滚到保存点或最近的落实点）。
	已声明临时表: 不支持锁定和恢复功能，但仅当显式或隐式地指定了 LOGGED 时，才支持日志记录功能。在显式或隐式地指定了 LOGGED 时，支持撤销恢复功能（将更改回滚到保存点或最近的落实点）。

修改表

本节提供有关如何修改表的主题。

更改表

更改表时，您应该了解一些有用的选项，例如 ALTER COLUMN SET DATA TYPE 选项以及可以在单一事务中执行的不受限 REORG 建议操作。

ALTER TABLE 的 SET DATA TYPE 支持

ALTER TABLE 语句的 ALTER COLUMN SET DATA TYPE 选项支持所有兼容的类型。

更改列数据类型可能会导致数据丢失。此类丢失的其中一些与强制类型转换规则一致；例如，将从字符串中截断空白而不会返回错误，并且将 DECIMAL 转换为 INTEGER 也将导致截断。为了防止发生意外的错误（例如溢出错误、截断错误或者由强制类

型转换所返回的任何其他类型错误），将扫描现有的列数据并将关于冲突行的消息写入通知日志。另外，还将检查列缺省值以确保它们与新数据类型一致。

如果数据扫描未报告任何错误，那么列类型将设为新数据类型，并且现有的列数据将被强制转换到新数据类型。如果报告了错误，那么 ALTER TABLE 语句将失败。

不支持将 VARCHAR、VARGRAPHIC 或 LOB 列更改为数据类型优先顺序列表中位置靠前的数据类型（请参阅*提升数据类型*主题）。

示例

将 SALES 表中 SALES 列的数据类型由 INTEGER 更改为 SMALLINT。

```
alter table sales alter column sales set data type smallint
DB20000I  The SQL command completed successfully.
```

将 SALES 表中 REGION 列的数据类型由 VARCHAR(15) 更改为 VARCHAR(14)。

```
alter table sales alter column region set data type varchar(14)
...
SQL0190N  ALTER TABLE "ADMINISTRATOR.SALES" specified attributes for column
"REGION" that are not compatible with the existing column.  SQLSTATE=42837
```

更改基本表中的列类型。存在直接或间接依赖于基本表的视图和函数。

```
create table t1 (c1 int, c2 int);

create view v1 as select c1, c2 from t1;
create view v2 as select c1, c2 from v1;

create function foo1 ()
 language sql
 returns int
 return select c2 from t1;

create view v3 as select c2 from v2
 where c2 = foo1();

create function foo2 ()
 language sql
 returns int
 return select c2 from v3;

alter table t1
 alter column c1
  set data type smallint;

select * from v2;
```

将列类型由 INTEGER 向下强制转换为 SMALLINT 的 ALTER TABLE 语句将使 v1、v2、v3 和 foo2 失效。在延迟重新生效语义下，`select * from v2` 将成功地使 v1 和 v2 重新生效，v1 和 v2 中的 c1 列都将更改为 SMALLINT。但是，v3 和 foo2 不会重新生效，这是因为它们在失效后未被引用，并且它们在依赖关系层次结构链中位于 v2 上方。在立即重新生效语义下，ALTER TABLE 语句将成功地使所有被依赖对象重新生效。

单个工作单元中的多项 ALTER TABLE 操作

某些 ALTER TABLE 操作（例如，删除列、改变列类型或者改变列的可空性属性）可能会使表进入 REORG 暂挂状态。在这种状态下，许多类型的查询不能运行；必须先执

行表重组，才能对表执行某些类型的查询。但是，即使对于处于 REORG 暂挂状态的表，您仍然可以在执行重组之前执行多项 ALTER TABLE 操作。

从 DB2 V9.7 开始，可以在单个*工作单元*中执行任意数目的 ALTER TABLE 语句。但是，在执行了三个包含这类操作的工作单元之后，必须运行 REORG TABLE 命令。

更改具体化查询表属性

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。

关于此任务

一旦将常规表更改为具体化查询表，该表便处于设置完整性暂挂状态。当使用此方法进行更改时，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更改为具体化查询表。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，那么不能将其更改为具体化查询表。如果更改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更改该表。

在将常规表更改为具体化查询表时，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，请使用以下命令:

```
ALTER TABLE sumtable
      SET SUMMARY AS DEFINITION ONLY
```

要将常规表更改为具体化查询表，请使用以下命令:

```
 ALTER TABLE regtable
      SET SUMMARY AS <fullselect>
```

将常规表更改为具体化查询表时，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 DATAACCESS 权限，或对要刷新的表具有 CONTROL 特权。

示例

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。

开始之前

要更改表，您必须对要更改的表至少具有下列其中一种特权:

- ALTER 特权
- CONTROL 特权
- DBADM 权限
- 对表模式的 ALTERIN 特权

您必须具有 DBADM 权限才能执行以下操作:

- 更改现有列的定义。
- 更改表列时编辑和测试 SQL。
- 更改表列时验证相关对象。

过程

要更改列属性，请执行以下操作:

发出 ALTER TABLE 语句。 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，可使用带有 ADD COLUMN 或 DROP COLUMN 子句的 ALTER TABLE 语句。表不能是类型表。

关于此任务

对于表中的所有现有列，新列的值将设为其缺省值。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。

过程

- 要添加列，请发出带 ADD COLUMN 子句的 ALTER TABLE 语句。 例如:

```
ALTER TABLE SALES
  ADD COLUMN SOLD_QTY
  SMALLINT NOT NULL DEFAULT 0
```

- 要删除列，请发出带 DROP COLUMN 子句的 ALTER TABLE 语句。 例如:

```
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。

关于此任务

在列定义中省略 DEFAULT 会导致将空值用作该列的缺省值，如第 260 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，可以将该列从 GENERATED ALWAYS 更改为 GENERATED BY DEFAULT。反之亦然，即可将该列从 GENERATED BY DEFAULT 更改为 GENERATED ALWAYS。只有使用标识列时，此操作才有可能。
- 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。

关于此任务

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更改其缺省值。

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。

过程

- 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，使用类似如下的语句:

```
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)
```

不能更改类型表的列。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
```

- 要修改列以允许以直接插入方式包括 LOB，请输入:

```
ALTER TABLE table_name
   ALTER COLUMN column_name
   SET INLINE LENGTH new_LOB_length
```

例如，如果您要将不超过 1000 字节的 LOB 包括在基本表行中，请使用类似如下的语句:

```
ALTER TABLE t1
   ALTER COLUMN colnam1
   SET INLINE LENGTH 1004
```

在本例中，长度设为 1004，而不是设为 1000。这是因为，除 LOB 本身的大小以外，直接插入 LOB 还另外需要 4 字节的存储器。

- 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'
```

重命名表和列

可以使用 RENAME 语句来重命名现有表。要重命名列，请使用 ALTER TABLE 语句。

关于此任务

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。

RENAME COLUMN 子句是 ALTER TABLE 语句的一个选项。您可以将基本表中现有的列重命名为新名称，而不会丢失已存储的数据，也不会影响任何与该表相关联的特权或者基于标签的访问控制（LBAC）策略。

只支持对基本表列进行重命名。不支持对视图、具体化查询表 (MQT)、已声明临时表和已创建临时表以及其他类似于表的对象中的列进行重命名。

重命名列操作的失效和重新生效语义与删除列操作的那些语义类似；即，所有从属对象都将失效。在重命名列操作后对所有从属对象执行的重新生效操作始终在失效操作完成后立即执行，即使 **auto_reval** 数据库配置参数设为 DISABLED 亦如此。

以下示例说明使用 ALTER TABLE 语句将列重命名:

```
ALTER TABLE org RENAME COLUMN deptnumb TO deptnum
```

要更改现有列的定义，请参阅“更改列属性”主题或 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*。

关于此任务

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.TABLES 和 SYSCAT.COLUMNS 目录视图来查看表定义。对于 SYSCAT.COLUMNS 而言，每一行都表示对表、视图或昵称定义的一列。要查看列中的数据，请使用 SELECT 语句。

关于此任务

还可以使用下列视图和表函数来查看表定义:

- ADMINTEMPCOLUMNS 管理视图
- ADMINTEMPTABLES 管理视图
- ADMIN_GET_TEMP_COLUMNS 表函数 - 检索临时表的列信息
- ADMIN_GET_TEMP_TABLES 表函数 - 检索临时表的信息

删除表

可以使用 DROP TABLE 语句删除表。

关于此任务

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。

限制

如果一个表有子表，那么不能删除该表。

过程

- 要删除表，请使用 DROP TABLE 语句。

  以下语句删除 DEPARTMENT 表:

  ```
  DROP TABLE DEPARTMENT
  ```

- 要删除表层次结构中的所有表，请使用 DROP TABLE HIERARCHY 语句。

  DROP TABLE HIERARCHY 语句必须命名要删除的层次结构的根表。 例如:

  ```
  DROP TABLE HIERARCHY person
  ```

结果

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 TABLE 语句将激活的删除触发器。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

不能更改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。

关于此任务

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。

过程

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
```

使用临时表的时间旅行查询

可使用临时表将基于时间的状态信息与数据相关联。未使用临时支持的表中的数据被认为适于存在，而临时表中的数据可在数据库系统和/或用户应用程序定义的时间段有效。

有许多业务需求需要存储和维护基于时间的数据。如果数据库中没有此功能，那么维护以时间为焦点的数据支持基础结构的成本很高并且很复杂。通过临时表，数据库可在没有其他应用程序逻辑的情况下存储并检索基于时间的数据。例如，数据库可存储表的历史记录（已删除行或已更新的行的原始值），所以可查询数据的过去状态。还可对数据行指定日期范围以通过应用程序或业务规则指示该数据何时被认为有效。

临时表记录一行有效的时间段。时间段是临时表中的两个日期或时间列定义的时间间隔。时间段包含开始列和结束列。开始列指示时间段的开始，结束列指示时间段的结束。时间段的开始列包含开始值和结束值，结束列不包含开始值和结束值。例如，时间段从 1 月 1 日至 2 月 1 日的行的有效期为 1 月 1 日至 1 月 31 日午夜。

支持两个时间段类型:

系统时间段

系统时间段由一对列组成，这两列包含指示一行最新的时间段的数据库管理器维护值。开始列包含表示一行创建时间的时间戳记值。结束列包含表示一行更新时间或删除时间的时间戳记值。如果创建系统时间段临时表，那么该表包含当前处于活动状态的行。每个系统时间段临时表与包含任何已更改行的历史记录表相关联。

应用程序时间段

应用程序时间段由一对列组成，这两列包含指示一行有效的时间段的用户值或应用程序提供值。开始列指示一行有效的开始时间。结束列指示一行停止有效的结束时间。带有应用程序时间段的表称为应用程序时间段临时表。

可通过查询 SYSCAT.TABLES 系统目录视图来检查表是否有临时支持。例如:

```
SELECT TABSCHEMA, TABNAME, TEMPORALTYPE FROM SYSCAT.TABLES
```

对 TEMPORALTYPE 返回的值按如下方式定义:

A 应用程序时间段临时表

B 双时态表

N 不是临时表

S 系统时间段临时表

系统时间段临时表

系统时间段临时表是保存其行的历史版本的表。 应使用系统时间段临时表存储数据的当前版本，使用其关联历史记录表透明存储已更新的数据行和已删除的数据行。

系统时间段临时表包括 SYSTEM_TIME 时间段及以下列，这些列捕获一行中的数据为最新版本的开始时间和结束时间。数据库管理器还使用 SYSTEM_TIME 时间段来保存每个表行的历史版本（每当发生更新或删除时）。数据库管理器将这些行存储在历史记录表中，此表以独占方式与系统时间段临时表关联。添加版本控制会在系统时间段临时表与历史记录表之间建立链接。通过系统时间段临时表，查询可访问当前时间点的数据并可检索过去时间点的数据。

系统时间段临时表还包括事务开始标识列。此列捕获影响该行的事务开始执行的时间。如果在单个 SQL 事务内插入或更新多行，那么事务开始标识列的值对于所有行都是相同的，并且不同于其他事务为此列生成的值。此列开始标识列值意味着您可使用事务开始标识列来标识表中由同一事务写入的所有行。

历史记录表

每个系统时间段临时表都需要一个历史记录表。在系统时间段临时表中更新或删除一行时，数据库管理器会将旧行的副本插入到其关联历史记录表中。此旧系统时间段临时表数据的存储允许您从过去时间点检索数据。

为存储行数据，历史记录表列与系统时间段临时表列必须具有相同的名称、顺序和数据类型。通过使用 CREATE TABLE 语句的 LIKE 子句，可创建列名称和描述与系统时间段临时表的列名称和描述相同的历史记录表，例如:

```
CREATE TABLE employees_history LIKE employees IN hist_space;
```

现有表可用作历史记录表，只要它避免 ALTER TABLE 语句 USE HISTORY 子句的描述中列示的限制。

创建历史记录表后，可添加版本控制以在系统时间段临时表与历史记录表之间建立链接。

```
ALTER TABLE employees ADD VERSIONING USE HISTORY TABLE employees_history;
```

如果启用版本控制，那么历史记录表受以下规则和限制约束:

- 不能显式删除历史记录表。仅当删除历史记录表的关联系统时间段临时表后，才能隐式删除该历史记录表。
- 不能显式添加、删除或更改历史记录表列。
- 不能在引用约束中将历史记录表定义为父代、子代或自引用项。对历史记录表的访问被限制以阻止对该历史记录表执行层叠操作。
- 不能删除包含历史记录表但未包含其关联系统时间段临时表的表空间。

您应极少需要显式变更历史记录表。这样做可能会危及您审计系统时间段临时表数据历史记录的能力。您应限制对历史记录表的访问以保护其数据。

在正常操作下，历史记录表遇到的活动大部分是插入和读取。更新和删除极少遇到。缺少更新和删除意味着历史记录表通常没有可用空间可供插入新行的操作重复使用。如果在历史记录表中插入行对工作负载性能有负面影响，那么您可通过改变历史记录表的定义使用 APPEND ON 选项以避免搜索可用空间。此选项避免了与可用空间搜索相关联的处理，并直接将新行追加至表的结尾。

```
ALTER TABLE employees_history APPEND ON;
```

删除系统时间段临时表时，会隐式删除关联历史记录表及对该历史记录表定义的任何索引。为避免删除系统时间段临时表时丢失历史数据，可使用 RESTRICT ON DROP 属性创建该历史记录表或通过添加 RESTRICT ON DROP 属性来改变该历史记录表。

```
CREATE TABLE employees_history LIKE employees WITH RESTRICT ON DROP;
```

因为历史记录表遇到的插入操作比删除操作多，所以历史记录表总是增大，从而导致消耗的存储量越来越大。决定如何修改历史记录表以除去不再需要的行可能是一个很复杂的任务。您需要了解各个记录的价值。某些内容（例如，客户合同）可能不能进行处理，因此决不能删除。而其他记录（例如，Web 站点访问者信息）可放心修改。通常，确定行的修改和归档时间的决定因素是某种业务逻辑而不是行的生存期。以下列表包含某些可行的修改规则:

- 修改由用户提供的查询（反映某些业务规则）选择的行。
- 行的存在时间超过一定生存期时，修改这些行。
- 记录有 N 个以上版本时，修改该记录的历史记录行（仅保留最新的 N 个版本）。
- 从关联系统时间段临时表中删除记录时，修改该记录的历史记录行（如果没有最新版本）。

可使用若干方法来定期从历史记录表中修改旧数据:

- 使用范围分区以及从历史记录表中拆离旧分区。
- 使用 DELETE 语句从表中删除行。如果使用 DELETE 语句，那么您需要遵循以下准则:
  - 定期重组历史记录表会释放在执行删除操作后留下的可用空间。
  - 确保没有更改历史记录表才能使用 APPEND ON 选项，从而允许插入操作来搜索可用空间。

SYSTEM_TIME 时间段

系统时间段临时表的 SYSTEM_TIME 时间段列指示何时一行的版本为最新。

SYSTEM_TIME 时间段包含一对 TIMESTAMP(12) 列，其值由数据库管理器生成。必须使用适用 GENERATED ALWAYS AS 选项将这些列定义为 NOT NULL。时间段的开始列必须是行开始列，时间段的结束列必须是行结束列。

```
CREATE TABLE policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 sys_start   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
 sys_end  TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
 ts_id   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID,
 PERIOD SYSTEM_TIME (sys_start, sys_end)
) IN policy_space;
```

行开始列

此列表示行数据变为最新的时间。数据库管理器通过在生成行的事务中执行第一个数据更改语句时读取系统时钟来为此列生成值。如果在单个 SQL 事务中插入或更新多行，那么对于所有受影响的行，行开始列的值相同。这些行开始列的值不同于为其他事务的行开始列生成的值。行开始列是必需的（作为 SYSTEM_TIME 时间段的开始列），必须对每个系统时间段临时表定义该列。

改变现有常规表以使其成为系统时间段临时表时，会向该表添加行开始列。行开始列被填充缺省值 0001-01-01-00.00.00.000000000000 以表示任何现有行的 TIMESTAMP(12) 数据类型值。

行结束列

此列表示行数据不再为最新的时间。对于历史记录表中的行，行结束列中的值表示将该行添加至历史记录表的时间。系统时间段临时表中的行是最新的（按定义），所以行结束列被填充 TIMESTAMP(12) 数据类型的缺省值（例如，9999-12-30-00.00.00.000000000000）。行结束列是必需的（作为 SYSTEM_TIME 时间段的结束列），必须对每个系统时间段临时表定义该列。

改变现有常规表以使其成为系统时间段临时表时，会向该表添加行结束列。行结束列被填充任何现有行的 TIMESTAMP(12) 数据类型的最大值（缺省值: 9999-12-30-00.00.00.000000000000）。

因为行开始列和行结束列都是生成列，所以未对 SYSTEM_TIME 生成隐式检查约束来确保系统时间段临时表中结束列的值大于其开始列的值。不同的是，应用程序时间段临时表不缺少此检查约束，在该表中，存在与其 BUSINESS_TIME 相关联的检查约束。如果一行的结束列的值小于开始列的值，那么使用时间段规范来查询该表时，无法返回该行。可定义约束来保证结束列的值大于开始列的值。此保证在支持显式将数据输入到这些生成列中的操作（例如，装入操作）时很有用。

systime_period_adj 数据库配置参数用于指定为系统时间段临时表生成历史记录行时，如果该行的结束列值小于开始列值，那么应执行什么操作。

创建系统时间段临时表

创建系统时间段临时表会生成一个表，此表跟踪数据更改的发生时间并保存该数据的历史版本。

关于此任务

创建系统时间段临时表时，应包括指示一行中的数据为当前数据的属性以及指示事务影响了该数据的属性:

- 包括 SYSTEM_TIME 时间段使用的行开始列和行结束列以跟踪一行为最新版本的时间。
- 包括事务开始标识列，该列会捕获影响行的事务的开始时间。
- 创建历史记录表以从系统时间段临时表中接收旧行。
- 添加版本控制以在系统时间段临时表与历史记录表之间建立链接。

行开始列、行结束列和事务开始标识列可定义为 IMPLICITLY HIDDEN。因为这些列及其条目由数据库管理器生成，所以隐藏它们可尽量减少对应用程序的任何潜在负面影响。于是，除非引用这些列，否则这些列不可用，例如:

- 针对表运行的 SELECT * 查询不会在结果表中返回任何隐式隐藏列。
- INSERT 语句不需要任何隐式隐藏列的值。
- LOAD、IMPORT 和 IMPORT 命令可使用 *includeimplicitlyhidden* 修饰符来处理隐式隐藏列。

系统时间段临时表可在引用约束中定义为父代或子代。但是，引用约束仅应用于当前数据，即，系统时间段临时表中的数据。这些约束不会应用于关联历史记录表。系统时间段临时表为引用约束中的子表时，为尽量减少不一致，父表也应是系统时间段临时表。

注: 虽然创建系统时间段临时表时行开始生成列、行结束生成列和事务开始标识生成列是必需的，但您也可使用这些生成列创建常规表。

以下部分中的示例说明如何创建用于存储保险公司客户的保单信息的表。

过程

创建系统时间段临时表。

1. 创建带有 SYSTEM_TIME 属性的表。 例如:

```
CREATE TABLE policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 sys_start   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
 sys_end  TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
 ts_id   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID,
 PERIOD SYSTEM_TIME (sys_start, sys_end)
) IN policy_space;
```

2. 创建历史记录表。 例如:

```
CREATE TABLE hist_policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 sys_start  TIMESTAMP(12) NOT NULL,
 sys_end  TIMESTAMP(12) NOT NULL,
 ts_id    TIMESTAMP(12) NOT NULL
) IN hist_space;
```

   还可通过使用 CREATE TABLE 语句的 LIKE 子句，创建列名称和描述与系统时间段临时表的列名称和描述相同的历史记录表。例如:

```
CREATE TABLE hist_policy_info LIKE policy_info IN hist_space;
```

3. 向系统时间段临时表添加版本控制以建立与历史记录表的链接。 例如:

```
ALTER TABLE policy_info ADD VERSIONING USE HISTORY TABLE hist_policy_info;
```

结果

policy_info 表存储客户的保险覆盖范围级别。与 SYSTEM_TIME 时间段相关的列（sys_start 和 sys_end）显示覆盖范围级别行何时为最新版本。ts_id 列用于列示开始执行影响该行的事务的时间。

表 25. 创建的系统时间段临时表 (policy_info)

policy_id	coverage	sys_start	sys_end	ts_id

hist_policy_info 历史记录表从 policy_info 表接收旧行。

表 26. 创建的历史记录表 (hist_policy_info)

policy_id	coverage	sys_start	sys_end	ts_id

示例

本节包含其他创建系统时间段临时表的示例。

隐藏列 以下示例创建 policy_info 表并将 TIMESTAMP(12) 列（sys_start、sys_end 和 ts_id）标记为隐式隐藏。

```
CREATE TABLE policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 sys_start   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN IMPLICITLY HIDDEN,
 sys_end  TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END IMPLICITLY HIDDEN,
 ts_id   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID IMPLICITLY HIDDEN,
 PERIOD SYSTEM_TIME (sys_start, sys_end)
) in policy_space;
```

使用 CREATE TABLE 语句的 LIKE 子句创建 hist_policy_info 历史记录表会生成从 policy_info 表继承隐式隐藏属性的历史记录表。创建历史记录表时如果没有使用 LIKE 子句，那么在系统时间段临时表中标记为隐藏的所有列在相关历史记录表中也必须标记为隐藏。

将现有表更改为系统时间段临时表

以下示例向现有表 (employees) 添加时间戳记列和 SYSTEM_TIME 时间段以启用系统时间段临时表功能。

```
ALTER TABLE employees
   ADD COLUMN sys_start TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN;
ALTER TABLE employees
   ADD COLUMN sys_end TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END;
ALTER TABLE employees
   ADD COLUMN ts_id TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS TRANSACTION START ID;
ALTER TABLE employees ADD PERIOD SYSTEM_TIME(sys_start, sys_end);
```

可通过在 ALTER TABLE 语句中添加 IMPLICITLY HIDDEN 子句来隐藏这些新列

必须创建历史记录表并添加版本控制才能完成此任务。

在系统时间段临时表中插入数据

对于用户，在系统时间段临时表中插入数据类似在常规表中插入数据。

关于此任务

在系统时间段临时表中插入数据时，数据库管理器会自动为行开始时间戳记列和行结束时间戳记列生成值。数据库管理器还会生成事务开始标识值，它唯一标识要插入该行的事务。

过程

要在系统时间段临时表中插入数据，请使用 INSERT 语句向该表添加数据。 例如，2010 年 1 月 31 日 (2010-01-31) 向“创建系统时间段临时表”主题的示例中创建的表插入了以下数据。

```
INSERT INTO policy_info(policy_id, coverage)
   VALUES('A123',12000);

INSERT INTO policy_info(policy_id, coverage)
   VALUES('B345',18000);

INSERT INTO policy_info(policy_id, coverage)
   VALUES('C567',20000);
```

结果

policy_info 表现在包含以下保险覆盖范围数据。数据库管理器生成 sys_start、sys_end 和 ts_id 列条目。

表 27. 添加至系统时间段临时表 (policy_info) 的数据

policy_id	cover-age	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	20000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000

his_policy_info 历史记录表仍然为空，因为插入操作未生成任何历史记录行。

表 28. 执行插入后的历史记录表 (hist_policy_info)

policy_id	coverage	sys_start	sys_end	ts_id

注: 行开始列 sys_start 表示行数据变为最新的时间。数据库管理器通过在生成行的事务中执行第一个数据更改语句时读取系统时钟来生成此值。数据库管理器还会生成事务开始标识列 ts_id，此列捕获影响该行的事务开始执行的时间。在许多情况下，这两个列的时间戳记值相同，因为它们是通过执行同一事务生成的。

多个事务更新同一行时，可能会发生时间戳记值冲突。数据库管理器可通过调整行开始列时间戳记值来解决这些冲突。在这类情况下，行开始列和事务开始标识列中的值将会不同。“更新系统时间段临时表”中的“示例”部分提供有关时间戳记调整的更多详细信息。

更新系统时间段临时表中的数据

更新系统时间段临时表中的数据会导致向其关联历史记录表添加行。

关于此任务

除更新系统时间段临时表行中指定列的值之外，UPDATE 语句会将现有行的副本插入至关联历史记录表。历史记录行作为更新该行的同一事务的一部分生成。如果单个事务对同一行进行多次更新，那么只会生成一个历史记录行，并且该行反映该事务进行任何更改之前该记录的状态。

注：多个事务更新同一行时，可能发生时间戳记值冲突。这些冲突发生时，『systime_period_adj』数据库配置参数的设置会确定是进行时间戳记调整还是事务应失败。**更多示例**部分中的**不同事务对一行进行的多个更改**示例提供更多详细信息。应用程序员可考虑使用 SQLCODE 或 SQLSTATE 值来处理来自 SQL 语句的与潜在时间戳记值调整相关的返回码。

过程

要更新系统时间段临时表中的数据，请使用 UPDATE 语句。 例如，发现客户的保险覆盖范围级别存在某些错误，并且 2011 年 2 月 28 日 (2011-02-28) 在“向系统时间段临时表插入数据”主题中添加了数据的示例表中更新了以下数据。
下表包含原始 `policy_info` 表数据。

表 29. 系统时间段临时表 (policy_info) 中的原始数据

policy_id	cover-age	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	20000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000

- 保单 C567 的覆盖范围应该为 25000。

```
UPDATE policy_info
   SET coverage = 25000
   WHERE policy_id = 'C567';
```

对保单 C567 的更新会影响系统时间段临时表及其历史记录表，导致出现以下情况:

1. 保单 C567 的该行的 coverage 值更新为 25000。
2. 在系统时间段临时表中，数据库管理器将 `sys_start` 和 `ts_id` 值更新为更新日期。
3. 原始行移至历史记录表。数据库管理器将 `sys_end` 值更新为更新日期。此行可解释为保单 C567 的有效覆盖范围为 2010-01-31-22.31.33.495925000000 至 2011-02-28-09.10.12.649592000000。

表 30. 系统时间段临时表 (policy_info) 中的已更新数据

policy_id	cover-age	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	25000	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

表 31. 执行更新后的历史记录表 (hist_policy_info)

policy_id	cover-age	sys_start	sys_end	ts_id
C567	20000	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

更多示例

本节包含更新系统时间段临时表的更多示例。

时间指定

在以下示例中，表更新中指定了一个时间段。以下更新在先前“过程”部分中的更新完成后运行。

```
UPDATE (SELECT * FROM policy_info
   FOR SYSTEM_TIME AS OF '2010-01-31-22.31.33.495925000000')
   SET coverage = coverage + 1000;
```

此更新返回错误，因为它暗地试图更新历史记录行。SELECT 显式查询 policy_info 表并隐式查询其关联历史记录表 (hist_policy_info)。SELECT 将返回 hist_policy_info 中的 C567 行，但不能更新历史记录表中的隐式访问的行。

不同事务对一行进行的多个更改

在以下示例中，两个事务将同时针对 policy_info 表执行 SQL 语句。在此示例中，时间戳记简化为占位符而不是样本系统时钟值。例如，示例使用 T1 而不是 2010-01-31-22.31.33.495925000000。编号较高的占位符指示事务内较晚执行的操作。例如，T5 晚于 T4。

如果在单个 SQL 事务内插入或更新多行，那么对于所有受影响行，行开始列的值相同。该值通过读取执行事务中的第一个数据更改语句时的系统时钟得出。例如，与事务 ABC 相关联的所有时间都为 T1。

事务 ABC **事务 XYZ**

```
T1: INSERT INTO policy_info(policy_id, coverage)
         VALUES ('S777',7000);
```

事务 ABC

事务 XYZ

```
                                    T2: INSERT INTO policy_info
                                    (policy_id, coverage)
                                            VALUES ('T888',8000);

                                    T3: COMMIT;

T4: UPDATE policy_info
      SET policy_id = 'X999'
      WHERE policy_id = 'T888';

T5: INSERT INTO policy_info(policy_id, coverage)
        VALUES ('Y555',9000);

T6: COMMIT;
```

在 **T1** 和 **T2** 执行插入后，policy_info 表将类似如下所示，并且历史记录表将为空 (hist_policy_info)。sys_end 列中的值 **max** 被填充 TIMESTAMP(12) 数据类型的最大缺省值。

表 32. 不同事务对 policy_info 表执行插入

policy_id	cover-age	sys_start	sys_end	ts_id
S777	7000	T1	max	T1
T888	8000	T2	max	T2

事务 ABC 在时间 T4 执行更新后，保单信息将类似下表所示。policy_info 表中的所有行反映来自事务 ABC 的插入和更新活动。这些行的 sys_start 和 ts_id 列将填充时间 T1，它是执行事务 ABC 中第一个数据更改语句的时间。事务 XYZ 插入的保单信息将更新，并且原始行移至历史记录表。

表 33. 更新 policy_info 表后执行的不同事务

policy_id	cover-age	sys_start	sys_end	ts_id
S777	7000	T1	max	T1
X999	8000	T1	max	T1

表 34. 不同事务更新后的历史记录表 (hist_policy_info)

policy_id	cover-age	sys_start	sys_end	ts_id
T888	8000	T2	T1	T2

历史记录表显示 sys_end 时间，此时间早于 sys_start。在这种情况下，时间 T4 处的更新无法执行，并且事务 ABC 将失败 (SQLSTATE 57062, SQLCODE SQL20528N)。为避免这类失败，可将 **systime_period_adj** 数据库配置参数设为 *YES*，这允许数据库管理器调整行开始时间戳记 (SQLSTATE 01695, SQLCODE SQL5191W)。事务 ABC 中的时间 T4 更新的 sys_start 时间戳记设为时间 T2 加上变化量 (T2+delta)。此调整仅应用于时间 T4 更新，事务 ABC 执行的所有其他更改将继续使用时间 T1 时间戳记（例如，插入 policy_id 为 Y555 的保单）。进行此调整并完成事务 ABC 后，保单表将包含以下数据:

表 35. 时间调整后的不同事务 (policy_info)

policy_id	cover-age	sys_start	sys_end	ts_id
S777	7000	T1	max	T1
X999	8000	T2+delta	max	T1
Y555	9000	T1	max	T1

表 36. 时间调整后的历史记录表 (hist_policy_info)

policy_id	cover-age	sys_start	sys_end	ts_id
T888	8000	T2	T2+delta	T2

同一事务中对一行进行的多个更改

在以下示例中，一个事务对一行进行多个更改。通过使用先前示例中的保单表，事务 ABC 继续并在时间 T6 更新 policy_id 为 X999 的保单（最初 T6 为 COMMIT 语句）。

事务 ABC

```
T6: UPDATE policy_info SET policy_id = 'R111' WHERE policy_id = 'X999';
T7: COMMIT;
```

此行现在遇到了以下更改:

1. 在时间 T2 被事务 XYZ 创建为保单 T888。
2. 在时间 T4 被事务 ABC 更新为保单 X999。
3. 在时间 T6 被事务 ABC 更新为保单 R111。

一个事务对同一行进行多个更新时，数据库管理器仅对第一个更改生成历史记录行。这会生成以下表:

表 37. 更新后的同一事务 (policy_info)

policy_id	cover-age	sys_start	sys_end	ts_id
S777	7000	T1	max	T1
R111	8000	T1	max	T1
Y555	9000	T1	max	T1

表 38. 同一事务更新后的历史记录表 (hist_policy_info)

policy_id	cover-age	sys_start	sys_end	ts_id
T888	8000	T2	T2+delta	T2

数据库管理器使用事务开始标识 (ts_id) 以唯一标识更改该行的事务。如果在单个 SQL 事务内插入或更新多行，那么事务开始标识列的值对于所有行都是相同的，并且不同于其他事务为此列生成的值。生成历史记录行之前，数据库管理器会确定对该行执行的最后一次更新对应时间 T1 开始的事务（ts_id 为 T1），它是进行当前更改的事务的相同事务开始时间，因此不会生成任何历史记录行。policy_info 表中该行的 sys_start 值更改为时间 T1。

更新视图

仅当引用系统时间段临时表或双时态表的视图的定义未包含 FOR SYSTEM_TIME 子句时，才能更新该视图。以下 UPDATE 语句更新 policy_info 表并生成历史记录行。

```
CREATE VIEW viewA AS SELECT * FROM policy_info;
UPDATE viewA SET col2 = col2 + 1000;
```

如果引用系统时间段临时表或双时态表的视图的定义包含 FOR SYSTEM_TIME 子句，那么可通过定义 INSTEAD OF 触发器来更新该视图。以下示例更新 regular_table 表。

```
CREATE VIEW viewB AS SELECT * FROM policy_info;
   FOR SYSTEM_TIME BETWEEN
   TIMESTAMP '2010-01-01 10:00:00' AND TIMESTAMP '2011-01-01 10:00:00';

CREATE TRIGGER update INSTEAD OF UPDATE ON viewB
   REFERENCING NEW AS n FOR EACH ROW
   UPDATE regular_table SET col1 = n.id;

UPDATE viewB set id = 500;
```

从系统时间段临时表中删除数据

从系统时间段临时表中删除数据会从该表中除去行并向关联历史记录表添加行。这些行添加时带有适当的系统时间戳记。

关于此任务

除删除系统时间段临时表的指定行之外，DELETE FROM 语句还会在从系统时间段临时表中删除现有行之前将该行的副本移至关联历史记录表。

过程

要从系统时间段临时表中删除数据，请使用 DELETE FROM 语句。 例如，保单 B345 的所有者决定取消保险覆盖范围。已在 2011 年 9 月 1 日 (2011-09-01) 从“更新双时态表中的数据”主题中已更新的表中删除该数据。

```
DELETE FROM policy_info WHERE policy_id = 'B345';
```

结果

原始 policy_info 表数据如下所示:

表 39. 执行 DELETE 语句前系统时间段临时表 (policy_info) 中的数据

policy_id	cover-age	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	25000	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

删除保单 B345 会影响系统时间段临时表及其历史记录表，导致出现以下情况:

1. 从系统时间段临时表中删除 policy_id 列值为 B345 的行。

2. 原始行移至历史记录表。数据库管理器将 sys_end 列值更新为删除日期。

表 40. 执行 DELETE 语句后系统时间段临时表 (policy_info) 中的数据

policy_id	cover-age	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	25000	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

表 41. 执行删除后的历史记录表 (hist_policy_info)

policy_id	cover-age	sys_start	sys_end	ts_id
C567	20000	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	2011-09-01-12.18.22.959254000000	2010-01-31-22.31.33.495925000000

示例

本节包含针对系统时间段临时表的删除操作的更多示例。

时间指定

在以下示例中，DELETE 语句中指定了一个时间段。以下删除操作在先前“过程”部分中的删除操作完成后运行。

```
DELETE FROM (SELECT * FROM policy_info
   FOR SYSTEM_TIME AS OF '2010-01-31-22.31.33.495925000000')
   WHERE policy_id = C567;
```

此 DELETE 语句返回错误。SELECT 语句显式查询 policy_info 表并隐式查询其关联历史记录表 (hist_policy_info)。SELECT 语句将返回 hist_policy_info 表中 policy_id 列值为 C567 的行，但历史记录表中隐式访问的行不能被删除。

查询系统时间段临时数据

查询系统时间段临时表可返回指定时间段的结果。这些结果可能包括当前值和先前历史值。

关于此任务

查询系统时间段临时表时，可在 FROM 子句中包括 FOR SYSTEM_TIME。通过使用 FOR SYSTEM_TIME 规范，可查询数据的当前状态和过去状态。时间段按如下所示指定:

AS OF *value1*

包括符合后述条件的所有行，这些行的时间段的开始值小于或等于 *value1*，结束值大于 *value1*。这允许您查询从某个时间点开始的数据。

FROM *value1* **TO** *value2*

包括符合后述条件的所有行，这些行的时间段的开始值等于或大于 *value1*，结束值小于 *value2*。这意味着开始时间包括在该时间段内，但结束时间不包括在该时间段内。

BETWEEN *value1* **AND** *value2*

包括符合后述条件的所有行，这些行的任何时间段与 *value1* 与 *value2* 之间的任意时间点重叠。如果该时间段的开始值小于或等于 *value2* 且该时间段的结束值大于 *value1*，那么返回一行。

请参阅以下部分以获取某些样本查询。

过程

要查询系统时间段临时表，请使用 SELECT 语句。例如，以下每个查询请求来自“从系统时间段临时表中删除数据”主题中结果表的保单信息。每个查询使用 FOR SYSTEM_TIME 规范的变体。
`policy_info` 表及其关联历史记录表如下所示:

表 42. 系统时间段临时表: policy_info

policy_id	cover-age	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	25000	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

表 43. 历史记录表: hist_policy_info

policy_id	cover-age	sys_start	sys_end	ts_id
C567	20000	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	2011-09-01-12.18.22.959254000000	2010-01-31-22.31.33.495925000000

- 未指定时间段的查询。 例如:

```
SELECT policy_id, coverage
   FROM policy_info
   where policy_id = 'C567'
```

此查询返回一行。SELECT 仅查询 `policy_info` 表。不会查询历史记录表，因为未指定 FOR SYSTEM_TIME。

```
C567, 25000
```

- 指定了 FOR SYSTEM_TIME AS OF 的查询。 例如:

```
SELECT policy_id, coverage
   FROM policy_info
   FOR SYSTEM_TIME AS OF
      '2011-02-28-09.10.12.649592000000'
```

此查询返回三行。该 SELECT 查询 policy_info 表和 hist_policy_info 表。时间段的开始列包含开始值和结束值，结束列不包含开始值和结束值。sys_end 列值为 2011-02-28-22.31.33.495925000000 的历史记录表行等于 *value1*，但它必须小于 *value1* 才能返回。

```
A123, 12000
C567, 25000
B345, 18000
```

- 指定了 FOR SYSTEM_TIME FROM..TO 的查询。 例如:

```
SELECT policy_id, coverage, sys_start, sys_end
   FROM policy_info
   FOR SYSTEM_TIME FROM
      '0001-01-01-00.00.00.000000' TO '9999-12-30-00.00.00.000000000000'
   where policy_id = 'C567'
```

此查询返回两行。该 SELECT 查询 policy_info 表和 hist_policy_info 表。

```
C567, 25000, 2011-02-28-09.10.12.649592000000, 9999-12-30-00.00.00.000000000000
C567, 20000, 2010-01-31-22.31.33.495925000000, 2011-02-28-09.10.12.649592000000
```

- 指定了 FOR SYSTEM_TIME BETWEEN..AND 的查询。 例如:

```
SELECT policy_id, coverage
   FROM policy_info
   FOR SYSTEM_TIME BETWEEN
      '2011-02-28-09.10.12.649592000000' AND '9999-12-30-00.00.00.000000000000'
```

此查询返回三行。该 SELECT 查询 policy_info 表和 hist_policy_info 表。将返回以 sys_start 列值 2011-02-28-09.10.12.649592000000 开头并且等于 *value1* 的行，因为包括了时间段的开始时间。将不返回 sys_end 列值为 2011-02-28-09.10.12.649592000000 并且等于 *value1* 的行，因为未包括时间段的结束时间。

```
A123, 12000
C567, 25000
B345, 18000
```

更多示例

本节包含其他查询系统时间段临时表的示例。

使用其他有效日期或时间戳记值的查询

创建了时间相关列声明为 TIMESTAMP(12) 的 policy_info 表，所以在执行之前，使用任何其他有效日期或时间戳记值的查询将转换为使用 TIMESTAMP(12)。例如:

```
SELECT policy_id, coverage
   FROM policy_info
   FOR SYSTEM_TIME AS OF '2011-02-28'
```

按如下所示转换并执行:

```
SELECT policy_id, coverage
   FROM policy_info
   FOR SYSTEM_TIME AS OF '2011-02-28-00.00.00.000000000000'
```

查询视图

可像查询系统时间段临时表那样查询视图。可在视图引用后指定 FOR SYSTEM_TIME 规范。

```
CREATE VIEW policy_2011(policy, start_date)
   AS SELECT policy_id, sys_start FROM policy_info;

SELECT * FROM policy_2011 FOR SYSTEM_TIME BETWEEN
   '2011-01-01-00.00.00.000000' AND '2011-12-31-23.59.59.999999999999';
```

针对视图 policy_2011 的 SELECT 查询 policy_info 表和 hist_policy_info 表。将返回在 2011 年任意时间生效的所有保单并包括保单开始日期。

```
A123, 2010-01-31-22.31.33.495925000000
C567, 2011-02-28-09.10.12.649592000000
C567, 2010-01-31-22.31.33.495925000000
B345, 2010-01-31-22.31.33.495925000000
```

如果视图定义包含时间段指定，那么针对该视图的查询不能包含时间段指定。以下语句返回错误，因为指定了多个时间段:

```
CREATE VIEW all_policies AS SELECT * FROM policy_info;
   FOR SYSTEM_TIME AS OF '2011-02-28-09.10.12.649592000000';

SELECT * FROM all_policies FOR SYSTEM_TIME BETWEEN
   '2011-01-01-00.00.00.000000' AND '2011-12-31-23.59.59.999999999999';
```

为会话设置系统时间

使用 CURRENT TEMPORAL SYSTEM_TIME 专用寄存器设置系统时间可减少或消除针对不同时间点运行应用程序时所需的更改。

关于此任务

如果要对系统时间段临时表运行某个应用程序以针对许多不同日期查询业务状态，那么可在专用寄存器中设置日期。如果需要查询截至今天、截至上一季度结束和截至去年同一日期的数据，那么可能无法更改应用程序并向每个 SQL 语句添加 AS OF 指定。使用打包应用程序时，可能存在此限制。要解决这类情况，可使用 CURRENT TEMPORAL SYSTEM_TIME 专用寄存器以在会话级别设置日期或时间戳记。

设置 CURRENT TEMPORAL SYSTEM_TIME 专用寄存器不会影响常规表。只有针对已启用版本控制的临时表（系统时间段临时表和双时态表）的查询才会使用专用寄存器中设置的时间。它对 DDL 语句也没有影响。该专用寄存器不会应用于针对引用完整性处理运行的任何扫描。.

要点: 如果 CURRENT TEMPORAL SYSTEM_TIME 专用寄存器设为非空值，那么针对系统时间段临时表执行的数据修改语句（例如，INSERT、UPDATE 和 DELETE）被阻止。如果专用寄存器设为过去某个时间（例如，五年前），那么允许数据修改操作可能导致历史数据记录更改。如果 CURRENT TEMPORAL SYSTEM_TIME 专用寄存器设为非空值，那么系统还会阻止对系统时间段临时表使用导入和装入之类的实用程序。

BIND 命令包含 SYSTIMESENSITIVE 选项，此选项指示静态和动态 SQL 语句中对系统时间段临时表的引用是否受 CURRENT TEMPORAL SYSTEM_TIME 专用寄存器的值影响。对于 SQL 过程，请使用 SET_ROUTINE_OPTS 过程来设置称为查询编译器变量的类似绑定选项。

过程

如果此专用寄存器设为非空值，那么发出查询的应用程序将返回截至该日期的数据或时间戳记。以下示例请求来自从系统时间段临时表中删除数据主题的结果表中的信息。

- 将专用寄存器设为当前时间戳记并查询一年前的数据。 假定当前时间戳记为 2011-05-17-14.45.31.434235000000:

```
SET CURRENT TEMPORAL SYSTEM_TIME = CURRENT TIMESTAMP - 1 YEAR;
SELECT policy_id, coverage FROM policy_info;
```

- 将专用寄存器设为某个时间戳记并在视图定义中引用系统时间段临时表。

```
CREATE VIEW view1 AS SELECT policy_id, coverage FROM policy_info;
CREATE VIEW view2 AS SELECT * FROM regular_table
   WHERE col1 IN (SELECT coverage FROM policy_info);
SET CURRENT TEMPORAL SYSTEM_TIME = TIMESTAMP '2011-02-28-09.10.12.649592000000';
SELECT * FROM view1;
SELECT * FROM view2;
```

- 将专用寄存器设为当前时间戳记并发出包含时间段指定的查询。 假定当前时间戳记为 2011-05-17-14.45.31.434235000000:

```
SET CURRENT TEMPORAL SYSTEM_TIME = CURRENT TIMESTAMP - 1 YEAR;
SELECT *
  FROM policy_info FOR SYSTEM_TIME AS OF '2011-02-28-09.10.12.649592000000';
```

结果

policy_info 表及其关联历史记录表如下所示:

表 44. 执行 DELETE 语句后系统时间段临时表 (policy_info) 中的数据

policy_id	cover-age	sys_start	sys_end	ts_id
A123	12000	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	25000	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

表 45. 执行删除后的历史记录表 (hist_policy_info)

policy_id	cover-age	sys_start	sys_end	ts_id
C567	20000	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
B345	18000	2010-01-31-22.31.33.495925000000	2011-09-01-12.18.22.959254000000	2010-01-31-22.31.33.495925000000

- 如果请求一年前的数据，那么将查询 policy_info 表中截至 2010-05-17-14.45.31.434235000000 的数据。该查询隐式重写为:

```
SELECT policy_id, coverage FROM policy_info
   FOR SYSTEM_TIME AS OF TIMESTAMP '2010-05-17-14.45.31.434235000000';
```

该 SELECT 查询 policy_info 表和 hist_policy_info 表并返回:

```
A123, 12000
C567, 20000
B345, 18000
```

- 针对 view1 的查询隐式重写为:

  ```
  SELECT * FROM view1 FOR SYSTEM_TIME AS OF CURRENT TEMPORAL SYSTEM_TIME;
  ```

  然后，该查询重写为:

  ```
  SELECT policy_id, coverage FROM policy_info
     FOR SYSTEM_TIME AS OF TIMESTAMP '2011-02-28-09.10.12.649592000000';
  ```

  该 SELECT 查询 policy_info 表和 hist_policy_info 表并返回:

  ```
  A123, 12000
  C567, 25000
  B345, 18000
  ```

  针对 view2 的查询涉及常规表的引用系统时间段临时表的视图，这会导致在常规表与专用寄存器之间形成隐式关系。该查询隐式重写为:

  ```
  SELECT * FROM view2 FOR SYSTEM_TIME AS OF CURRENT TEMPORAL SYSTEM_TIME;
  ```

  然后，该查询重写为:

  ```
  SELECT * FROM regular_table WHERE col1 in (SELECT coverage FROM policy_info
     FOR SYSTEM_TIME AS OF TIMESTAMP '2011-02-28-09.10.12.649592000000');
  ```

  该 SELECT 返回以下行，这些行的 col1 值与 coverage 中的值相匹配。
- 返回了错误，因为指定了多个时间段。专用寄存器设为非空值，并且该查询也指定了时间。

删除系统时间段临时表

删除系统时间段临时表时，还会删除其关联历史记录表及对该历史记录表定义的任何索引。

开始之前

要删除系统时间段临时表，您必须有权删除其历史记录表。

关于此任务

历史记录表在其关联系统时间段临时表被删除时隐式删除。不能使用 DROP 语句显式删除历史记录表。

为避免删除系统时间段临时表时丢失历史数据，可使用 RESTRICT ON DROP 属性创建该历史记录表或通过添加 RESTRICT ON DROP 属性来改变该历史记录表。如果尝试删除系统时间段临时表并且其历史记录表具有 RESTRICT ON DROP 属性，那么删除系统时间段临时表会失败 (SQLSTATE 42893)。在这类情况下，必须通过除去 VERSIONING 属性来中断系统时间段临时表与历史记录表之间的链接，然后重新运行 DROP 语句。

改变表来删除 VERSIONING 时，带有对该表的版本控制依赖性的所有包会失效。其他从属对象（例如，视图或触发器）在系统目录中标记为无效（“N”）。系统会执行自动重新生效。重新生效失败的任何对象在该目录中会保留为无效状态。仅在显式用户操作后，某些对象会变为有效。

过程

要删除系统时间段临时表及其关联历史记录表，请执行以下操作:

1. 可选: 保护历史数据以防删除:
   a. 如果创建的历史记录表没有 RESTRICT ON DROP 属性，请改变该历史记录表以设置 RESTRICT ON DROP 属性。 例如，如果审计需求要求保存保单的历史记录，那么必须保护该历史记录表。

      ```
      ALTER TABLE hist_policy_info ADD RESTRICT ON DROP;
      ```

   b. 通过除去 VERSIONING 属性来中断系统时间段临时表与带有 RESTRICT ON DROP 属性的历史记录表之间的链接。 例如:

      ```
      ALTER TABLE policy_info DROP VERSIONING;
      ```

2. 使用 DROP 语句来删除系统时间段临时表。 例如，不再需要“创建系统时间段临时表”主题的示例中创建的保单表。

   ```
   DROP TABLE policy_info;
   ```

结果

以上命令对 policy_info 和 hist_policy_info 表有如下影响:

- DROP 语句显式删除系统时间段临时表并隐式删除关联历史记录表。policy_info 和 hist_policy_info 表被删除。直接或间接依赖于这些表的任何对象被删除或变为不可操作。
- RESTRICT ON DROP 属性与历史记录表关联后，任何尝试删除 policy_info 表的操作将失败 (SQLSTATE 42893)。还可创建或改变系统时间段临时表以使用 RESTRICT ON DROP 属性。
- 中断系统时间段临时表与其历史记录表之间的链接后，policy_info 表可被删除，hist_policy_info 历史记录表仍将保留。

删除表空间

如果表空间包含历史记录表但未包含关联的系统时间段临时表，那么不能显式删除该表空间。例如，使用 policy_space 和 hist_space 表空间中创建的保单表时，不允许执行以下语句:

```
DROP TABLESPACE hist_space;
```

如果包含历史记录表的表空间以及包含关联系统时间段临时表的表空间包括在一起，那么允许执行该语句。例如，以下语句将成功执行:

```
DROP TABLESPACE policy_space hist_space;
```

删除历史记录表的关联系统时间段临时表的表空间时，会隐式删除该历史记录表。例如，以下语句将删除 hist_policy_info 历史记录表:

```
DROP TABLESPACE policy_space;
```

实用程序和工具

有许多工具和实用程序可用于处理和管理临时表中的数据。

下列工具可用于处理和管理临时表:

- 导入数据（请参阅第 321 页的『导入』）
- 装入数据（请参阅第 321 页的『装入』）

- 联机表移动（请参阅第 322 页的『ADMIN_MOVE_TABLE 过程』）
- 停顿表空间（请参阅第 322 页的『QUIESCE TABLESPACES FOR TABLE 命令』）
- 复制（请参阅第 322 页的『复制』）
- 前滚（请参阅第 322 页的『前滚』）
- ADMIN_COPY_SCHEMA 过程（请参阅 ADMIN_COPY_SCHEMA）

导入

在系统时间段临时表中导入数据时，使用文件类型修饰符以忽略外部文件中可能应用于系统时间段临时表中数据库管理器生成列的任何内容。在系统时间段临时表中导入数据时，以下修饰符可用。

periodignore
使用此修饰符以通知导入实用程序：外部文件中存在 SYSTEM_TIME 时间段列的数据，但应忽略。如果指定此修饰符，那么所有时间段列值由该实用程序生成。

periodmissing
使用此修饰符以告诉导入实用程序：外部数据文件未包含 SYSTEM_TIME 时间段列的任何数据。如果指定此修饰符，那么所有时间段列值由该实用程序生成。

transactionidignore
使用此修饰符以通知导入实用程序：外部文件中存在事务开始标识列的数据，但应忽略。如果指定此修饰符，那么该事务开始标识列的值由该实用程序生成。

transactionidmissing
使用此修饰符以告诉导入实用程序：外部数据文件未包含事务开始标识列的任何数据。如果指定此修饰符，那么该事务开始标识列的值由该实用程序生成。

与装入实用程序不同，导入实用程序没有用于覆盖 GENERATED ALWAYS 列的修饰符。

装入

在系统时间段临时表中装入数据时，使用文件类型修饰符以忽略外部文件中可能应用于数据库管理器生成列的任何数据或将用户提供的值装入至这些生成列。在系统时间段临时表中装入数据时，以下修饰符可用。LOAD REPLACE 在系统时间段临时表上受阻。

periodignore
使用此修饰符以通知装入实用程序：外部文件中存在 SYSTEM_TIME 时间段列的数据，但应忽略。如果指定此修饰符，那么所有时间段列值由该实用程序生成。

periodmissing
使用此修饰符以告诉装入实用程序：外部数据文件未包含 SYSTEM_TIME 时间段列的任何数据。如果指定此修饰符，那么所有时间段列值由该实用程序生成。

periodoverride
使用此修饰符以指示装入实用程序接受 SYSTEM_TIME 时间段行开始列和行结

束列的用户提供值。此修饰符覆盖 GENERATED ALWAYS 子句。如果您想要保留历史记录数据并在系统时间段临时表中装入包括时间戳记的数据，那么此修饰符会很有用。如果使用此修饰符，那么行开始列和行结束列中任何未包含数据或包含空数据的行会被拒绝。

transactionidignore
使用此修饰符以通知装入实用程序: 外部文件中存在事务开始标识列的数据，但应忽略。如果指定此修饰符，那么该事务开始标识列的值由该实用程序生成。

transactionidmissing
使用此修饰符以告诉装入实用程序: 外部数据文件未包含事务开始标识列的任何数据。如果指定此修饰符，那么该事务开始标识列的值由该实用程序生成。

transactionidoverride
使用此修饰符以指示装入实用程序接受事务开始标识列的用户提供值。此修饰符覆盖 GENERATED ALWAYS 子句。如果使用此修饰符，那么事务开始标识列中任何未包含数据或包含空数据的行会被拒绝。

ADMIN_MOVE_TABLE 过程

使用 ADMIN_MOVE_TABLE 存储过程将处于活动状态的系统时间段临时表中的数据移至同名新表时，以下操作被阻止。

- 联机移动操作期间，会更改系统时间段临时表或关联历史记录表定义的改变表操作会被阻止。
- ADMIN_MOVE_TABLE 的 KEEP 选项对系统时间段临时表不可用

历史记录表不支持联机表移动操作。

QUIESCE TABLESPACES FOR TABLE 命令

对系统时间段临时表运行 QUIESCE TABLESPACES FOR TABLE 命令时，与该系统时间段临时表及其历史记录表相关联的所有表空间会停顿。对历史记录表运行该命令时，与该历史记录表及关联系统时间段临时表相关联的所有表空间会停顿。

复制

当复制系统时间段临时表时，如果该目标是另一系统时间段临时表，那么具有以下生成属性的列无法参与到该复制中:

- GENERATED ALWAYS AS ROW BEGIN
- GENERATED ALWAYS AS ROW END
- GENERATED ALWAYS AS TRANSACTION START ID

类似地，当复制双时态表时，如果目标是另一双时态表，那么具有以下生成属性的列无法参与复制过程:

前滚

系统时间段临时表或双时态表的表空间前滚至某个时间点时，关联历史记录表的表空间也必须前滚至同一 ROLLFORWARD 语句中的同一时间点。同样，当历史记录表的表空间前滚到时间点时，系统时间段临时表或双时态表的表空间必须前滚到同一时间

点。但是，可单独将系统时间段临时表的表空间或历史记录表的表空间恢复至日志结尾。

ADMIN_COPY_SCHEMA 过程

ADMIN_COPY_SCHEMA 过程用于复制特定模式，且其中包含所有对象。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。ADMIN_COPY_SCHEMA 过程受系统时间段临时表支持。该过程要求系统时间段临时表和历史记录表处于同一模式下，否则，不会复制这两个表，且会记录错误。

模式更改

为维护系统时间段临时表与其关联历史记录表之间的关系的完整性，只允许对系统时间段临时表的模式执行某些更改。所有将导致数据丢失的更改受到限制。

可对系统时间段临时表进行以下更改。如果您有适当特权，那么这些更改会隐式传播至关联历史记录表。不能显式对历史记录表进行这些更改。

- ADD COLUMN（生成列除外）
- RENAME COLUMN
- ALTER COLUMN（在未丢失任何历史数据的情况下）。例如，允许将列的数据类型从 VARCHAR(50) 改变为 VARCHAR(100)，或从 INTEGER 改变为 DECIMAL。但是，不允许从 VARCHAR(100) 改变为 VARCHAR(50) 或从 DECIMAL 改变为 INTEGER，因为这会降低列的长度或精度并可能导致数据丢失。

不能对系统时间段临时表进行以下更改，因为数据可能会丢失:

- DROP COLUMN
- ADD COLUMN（已生成）
- ALTER COLUMN（在可能丢失历史记录数据的情况下）。例如，允许将列的数据类型从 VARCHAR(50) 改变为 VARCHAR(100)，或从 INTEGER 改变为 DECIMAL。但是，不允许从 VARCHAR(100) 改变为 VARCHAR(50) 或从 DECIMAL 改变为 INTEGER，因为这会降低列的长度或精度。

版本控制在系统时间段临时表与其关联历史记录表之间建立链接。版本控制处于活动状态时，针对系统时间段临时表及历史记录表的某些 ALTER TABLE 操作被阻止。

- ALTER TABLE DROP PERIOD
- ALTER TABLE ADD MATERIALIZED
- ALTER TABLE ACTIVATE NOT LOGGED INITIALLY
- ALTER TABLE ADD SECURITY POLICY
- ALTER TABLE DROP SECURITY POLICY
- ALTER TABLE SECURED WITH ALTER

以上列表中未显示的 ALTER TABLE 操作受支持，但未从系统时间段临时表传播至其历史记录表。

此外，RENAME INDEX 和 RENAME TABLE 受支持，但未从系统时间段临时表传播至其历史记录表。

游标和系统时间段临时表

用于为在历史记录表中潜在引用行的查询更新或删除行的游标必须为只读。

在以下示例中，该语句成功运行，因为游标 appcur 是只读游标。

```
DECLARE appcur CURSOR FOR SELECT * FROM policy_info FOR SYSTEM_TIME AS OF '2011-02-28';
```

但是，以下语句会导致错误，因为对历史记录行的任何游标引用都不是只读游标:

```
DECLARE appcur CURSOR FOR SELECT * FROM policy_info
FOR SYSTEM_TIME AS OF '2011-02-28' FOR UPDATE;
```

表分区和系统时间段临时表

系统时间段临时表可将其表数据划分到多个存储对象（称为数据分区）中。还可对与系统时间段临时表相关联的历史记录表进行分区。

如果启用了版本控制，那么将分区连接至系统时间段临时表或使分区与系统时间段临时表拆离时以下行为适用:

连接分区

- 如果启用了版本控制，那么表可连接至系统时间段临时表。
- 要连接的表必须包含所有三个时间戳记列（ROW BEGIN、ROW END 和 TRANSACTION START ID）。这些时间戳记列的定义必须与系统时间段临时表中对应列的定义相同。
- 要连接的表不需要 SYSTEM_TIME 时间段定义。
- 如果启用了版本控制，那么不能运行 SET INTEGRITY ... FOR EXCEPTION 语句，因为将异常行移至异常表将导致丢失历史记录。因为异常行未记录在历史记录表中，所以会损害系统时间段临时表及其关联历史记录表中数据的可审核性。可临时停止版本控制，运行 SET INTEGRITY ... FOR EXCEPTION 语句，然后再次启用版本控制。

拆离分区

- 如果启用了版本控制，那么无法从系统时间段临时表拆离表。可停止版本控制，然后从基本表拆离分区。已拆离分区变为独立表。从历史记录表拆离分区不需要您停止版本控制。
- 已拆离分区保留所有三个时间戳记列（ROW BEGIN、ROW END 和 TRANSACTION START ID），但未保留 SYSTEM_TIME 时间段定义。
- 已拆离分区中的行不会自动移至历史记录表。如果要保留历史记录，那么必须手动移动这些行。如果将这些行手动移至历史记录表，那么应该将 ROW END 时间戳记更改为行从当前变为历史的时间点。如果没有这些更改，那么与时间相关的查询可能返回意外结果。

系统时间段临时表的数据访问控制

可对系统时间段临时表及其关联历史记录表定义行访问控制和列访问控制。

行访问控制和列访问控制 (RCAC) 是数据安全层，用于在行级别和/或列级别控制对表的访问。RCAC 可应用于系统时间段临时表和历史记录表。如果仅对系统时间段临时表激活 RCAC，那么数据库管理器会自动对历史记录表激活行访问控制并为该历史记录表创建缺省行许可权。

当历史记录表受缺省行许可权保护时，更新和删除操作仍会在历史记录表中生成历史记录行。如果对系统时间段临时表发出 AS OF 查询，那么系统时间段临时表的 RCAC 行许可权和列屏蔽还将应用于从该历史记录表返回的行。

如果直接访问历史记录表，那么会应用对该历史记录表定义的任何行规则和列规则。

对系统时间段临时表的限制

系统时间段临时表受到许多限制。这些限制可能会影响系统时间段临时表的实现。

以下是对系统时间段临时表的限制:

- 不支持对系统时间段临时表执行基于标记的访问控制 (LBAC)。虽然启用了系统时间段数据版本控制，但向系统时间段临时表或历史记录表添加行标签和列标签的操作会被阻止。如果使用 ALTER TABLE 语句启用版本控制，那么数据库管理器会确保系统时间段临时表和历史记录表都没有使用标签保护的行或列。
- 不支持对系统时间段临时表执行导致数据可能丢失的 ALTER 操作。
- 对于系统时间段临时表和历史记录表的 ALTER TABLE ACTIVATE NOT LOGGED INITIALLY 语句会被阻止。
- 不能将系统时间段临时表改变为具体化查询表 (MQT)。
- 阻止从系统时间段临时表删除数据的实用程序，包括 LOAD REPLACE 和 IMPORT REPLACE。
- 不支持对系统时间段临时表执行 TRUNCATE 语句。
- 不支持对系统时间段临时表执行以下模式更改操作:
  - ALTER TABLE DROP COLUMN
  - ALTER TABLE ALTER COL（不支持将字符串数据类型改变为需要截断数据的类型。也不支持将数字数据类型改变为较低精度的类型）。
  - ALTER TABLE ADD GENERATED COLUMN
- 对于时间点恢复，如果包含系统时间段临时表的表空间正前滚至某个时间点，那么包含关联历史记录表的表空间也必须前滚至一个集合中的同一时间点。同样，当历史记录表的表空间前滚到时间点时，系统时间段临时表或双时态表的表空间必须前滚到同一时间点。但是，允许仅将系统时间段临时表的表空间（或历史记录表的表空间）滚动至日志结尾。
- 可对远程系统时间段临时表创建昵称，但不会展示临时信息，也不会支持针对昵称的临时操作。例如，针对联合昵称的临时数据定义操作和临时查询被阻止。
- 如果 CURRENT TEMPORAL SYSTEM_TIME 专用寄存器设为非空值，那么对系统时间段临时表的 IMPORT 和 LOAD 操作被阻止。

应用程序时间段临时表

应用程序时间段临时表是用于存储应用程序数据的生效部分。使用应用程序时间段临时表以通过定义数据有效的时间段来管理基于时间条件的数据。

与系统时间段临时表相似，应用程序时间段临时表包括带有以下列的 BUSINESS_TIME 时间段，这些列指示该行中的数据有效或生效的时间段。为与每行相关联的 BUSINESS_TIME 时间段提供开始时间和结束时间。但是，与系统时间段临时表不同，没有单独的历史记录表。过去、现在和将来生效日期及其关联业务数据保留在单个表中。可通过 BUSINESS_TIME 时间段来控制数据值，并使用应用程序时间段临时表对过去、现在和将来的数据建模。

BUSINESS_TIME 时间段

应用程序时间段临时表的 BUSINESS_TIME 时间段列记录一行的版本何时有效（从用户或业务应用程序的角度来看）。

BUSINESS_TIME 时间段 包含一对 DATE 或 TIMESTAMP(*p*) 列，其中 *p* 可以是 0 到 12。这些列由您或业务应用程序填充。BUSINESS_TIME 时间段中的两列指示有效期的开始和结束。这些列与 SYSTEM_TIME 时间段列的不同之处在于时间段值由数据库管理器生成。BUSINESS_TIME 时间段列必须定义为 NOT NULL 并且不得是生成列。

A BUSINESS_TIME 时间段是“包含/排除”式的。有效期的开始包括在 BUSINESS_TIME 中，结尾未包括在 BUSINESS_TIME 中。

每当对表定义 BUSINESS_TIME 时间段时，会生成名为 DB2_GENERATED_CHECK_CONSTRAINT_FOR_BUSINESS_TIME 的隐式检查约束以确保有效期的结束值大于有效期的开始值。如果存在同名约束，那么会返回错误。此约束在支持显式将数据输入到这些列中的操作（例如，插入或装入操作）时很有用。

可定义应用程序时间段临时表以使带有相同键的行的 BUSINESS_TIME 时间段没有任何重叠。例如，此约束将阻止一个保单的两个版本在任意时间点同时生效。这些控制都可通过向主键添加 BUSINESS_TIME WITHOUT OVERLAPS 子句、唯一约束规范或创建唯一索引语句来实现。实施由数据库管理器控制。此控制是可选的。

创建应用程序时间段临时表

创建应用程序时间段临时表会生成根据数据有效或生效的时间来管理数据的表。

关于此任务

创建应用程序时间段临时表时，应包括用于指示一行中的数据何时有效的 BUSINESS_TIME 时间段。可选择定义不允许 BUSINESS_TIME 的时间段重叠，并且值对于任何时间段都是唯一。以下部分中的示例说明如何创建用于存储保险公司客户的保单信息的表。

过程

要创建应用程序时间段临时表，请执行以下操作:

1. 创建带 BUSINESS_TIME 时间段的表。 例如:

```
CREATE TABLE policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 bus_start       DATE NOT NULL,
 bus_end         DATE NOT NULL,
 PERIOD BUSINESS_TIME (bus_start, bus_end)
 );
```

2. 可选: 创建阻止同一 policy_id 的 BUSINESS_TIME 时间段重叠的唯一索引。 例如:

```
CREATE UNIQUE INDEX ix_policy
   ON policy_info (policy_id, BUSINESS_TIME WITHOUT OVERLAPS);
```

结果

policy_info 表存储客户的保险覆盖范围级别。与 BUSINESS_TIME 时间段相关的列（bus_start 和 bus_end）指示保险覆盖范围级别何时有效。

表 46. 创建的应用程序时间段临时表 (policy_info)

policy_id	coverage	bus_start	bus_end

将 BUSINESS_TIME WITHOUT OVERLAPS 作为索引键列的列表中的最终列的 ix_policy 索引确保客户保险覆盖范围级别没有重叠时间段。

示例

本节包含其他创建应用程序时间段临时表的示例。

将现有表更改为应用程序时间段临时表

以下示例向现有表 (employees) 添加时间列和 BUSINESS_TIME 时间段以启用应用程序时间段临时表功能。添加 BUSINESS_TIME WITHOUT OVERLAPS 子句可确保某个职员在任意时间段内仅列示一次。

```
ALTER TABLE employees ADD COLUMN bus_start DATE NOT NULL;
ALTER TABLE employees ADD COLUMN bus_end DATE NOT NULL;
ALTER TABLE employees ADD PERIOD BUSINESS_TIME(bus_start, bus_end);
ALTER TABLE employees ADD CONSTRAINT uniq
   UNIQUE(employee_id, BUSINESS_TIME WITHOUT OVERLAPS);
```

阻止时间段重叠

在“过程”部分中，索引确保没有重叠 BUSINESS_TIME 时间段。在以下备用示例中，创建 policy_info 表时使用了 PRIMARY KEY 声明，以确保不允许 BUSINESS_TIME 时间段重叠。这意味着同一保单不能有两个版本同时有效。

```
CREATE TABLE policy_info
(
  policy_id CHAR(4) NOT NULL,
  coverage INT NOT NULL,
  bus_start DATE NOT NULL,
  bus_end DATE NOT NULL,
  PERIOD BUSINESS_TIME(bus_start, bus_end),
  PRIMARY KEY(policy_id, BUSINESS_TIME WITHOUT OVERLAPS)
);
```

确保时间段的唯一性

以下示例创建 product_availability 表，公司在其中跟踪它分发的产品、这些产品的供应商以及供应商收取的价格。多个供应商可同时提供同一产品，但 PRIMARY KEY 声明确保单个供应商在任何给定时间点只能收取一个价格。

```
CREATE TABLE product_availability
(
  product_id CHAR(4) NOT NULL,
  supplier_id INT NOT NULL,
  product_price DECIMAL NOT NULL
  bus_start DATE NOT NULL,
  bus_end DATE NOT NULL,
  PERIOD BUSINESS_TIME(bus_start, bus_end),
  PRIMARY KEY(product_id, supplier_id, BUSINESS_TIME WITHOUT OVERLAPS)
);
```

If the PRIMARY KEY was defined as

```
PRIMARY KEY(product_id, BUSINESS_TIME WITHOUT OVERLAPS)
```

，于是两个供应商不能同时提供同一产品。

在应用程序时间段临时表中插入数据

在应用程序时间段临时表中插入数据类似在常规表中插入数据。

关于此任务

如果在应用程序时间段临时表中插入数据，那么唯一需要的特殊注意事项是包括用于捕获该行有效（从关联业务应用程序的角度看）的时间的开始列和结束列。此有效时间段称为 BUSINESS_TIME 时间段。数据库管理器自动生成隐式检查约束，该检查约束确保 BUSINESS_TIME 时间段的开始列小于其结束列。如果为该表创建了带有 BUSINESS_TIME WITHOUT OVERLAPS 的唯一约束或索引，那么必须确保没有 BUSINESS_TIME 时间段重叠。

过程

要在应用程序时间段临时表中插入数据，请使用 INSERT 语句向该表添加数据。 例如，在“创建应用程序时间段临时表”主题的示例中创建的表内插入以下数据。

```
INSERT INTO policy_info VALUES('A123',12000,'2008-01-01','2008-07-01');

INSERT INTO policy_info VALUES('A123',16000,'2008-07-01','2009-01-01');

INSERT INTO policy_info VALUES('A123',16000,'2008-06-01','2008-08-01');

INSERT INTO policy_info VALUES('B345',18000,'2008-01-01','2009-01-01');

INSERT INTO policy_info VALUES('C567',20000,'2008-01-01','2009-01-01');
```

结果

发出了 5 个 INSERT 语句，但仅向该表添加了 4 行。第 2 个和第 3 个 INSERT 语句尝试对保单 A123 添加行，但其 BUSINESS_TIME 时间段重叠，这会导致以下结果:

- 第 2 个插入操作对 policy_id A123 添加 bus_start 值为 2008-07-01 并且 bus_end 值为 2009-01-01 的行。
- 第 3 个插入操作尝试对 policy_id A123 添加一行但失败，因为其 BUSINESS_TIME 时间段与上一插入操作的 BUSINESS_TIME 时间段重叠。创建了带有 BUSINESS_TIME WITHOUT OVERLAPS 索引的 policy_info 表，第 3 个插入操作的 bus_end 值为 2008-08-01，此值在之前插入操作的时间段内。

时间段的开始列包括开始值和结束值，结束列不包括开始值和结束值，这意味着 bus_end 值为 2008-07-01 的行的 BUSINESS_TIME 时间段未与 bus_start 值为 2008-07-01 的行的时间段重叠。因此，policy_info 表现在包含以下保险覆盖范围数据:

表 47. 添加至应用程序时间段临时表 (policy_info) 的数据

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-07-01
A123	16000	2008-07-01	2009-01-01
B345	18000	2008-01-01	2009-01-01
C567	20000	2008-01-01	2009-01-01

更新应用程序时间段临时表中的数据

更新应用程序时间段临时表中的数据可能与更新常规表中的数据相似，但还可以为过去、现在或将来的指定的时间点更新数据。时间点更新会导致拆分行，并自动将新行插入到该表中。

关于此任务

除了常规 UPDATE 语句外，应用程序时间段临时表还支持时间范围更新，在这类更新中，UPDATE 语句包括 FOR PORTION OF BUSINESS_TIME 子句。如果一行的时间段开始列和/或时间段结束列在 FOR PORTION OF BUSINESS_TIME 子句中指定的范围内，那么该行是更新操作的候选项。

过程

要更新应用程序时间段临时表中的数据，请使用 UPDATE 语句。 例如，您发现某些客户的保险覆盖范围信息存在一些错误，并在样本表上执行了以下更新（这在『在应用程序时间段临时表中插入数据』主题中有介绍）。
下表包含原始 policy_info 表数据。

表 48. 应用程序时间段临时表 (policy_info) 中的原始数据

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-07-01
A123	16000	2008-07-01	2009-01-01
B345	18000	2008-01-01	2009-01-01
C567	20000	2008-01-01	2009-01-01

创建了带有 BUSINESS_TIME WITHOUT OVERLAPS 索引的 policy_info 表。使用常规 UPDATE 语句时，必须确保没有 BUSINESS_TIME 时间段重叠。使用 FOR PORTION OF BUSINESS_TIME 子句更新应用程序时间段临时表可避免时间段重叠问题。此子句导致更改行，并可能导致插入行（如果要更新的行的现有时间段未完全包含在 UPDATE 语句中指定的范围内）。

- 保单 B345 的覆盖范围实际从 2008 年 3 月 1 日 (2008-03-01) 开始，且覆盖范围应该是 18500:

```
UPDATE policy_info
  SET coverage = 18500, bus_start = '2008-03-01'
  WHERE policy_id = 'B345'
  AND coverage=18000
```

  对保单 B345 的更新使用常规 UPDATE 语句。policy_info 表中只有一行对应 policy_id B345，所以没有潜在的 BUSINESS_TIME 时间段重叠。因此，将 bus_start 列值更新为 2008-03-01，且 coverage 值更新为 18500。请注意，对 BUSINESS_TIME 时间段列的更新不能包括 FOR PORTION OF BUSINESS_TIME 子句。

表 49. 保单 B345 已更新

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-07-01
A123	16000	2008-07-01	2009-01-01
B345	18500	2008-03-01	2009-01-01
C567	20000	2008-01-01	2009-01-01

- 保单 C567 对应 2008 年的覆盖范围应该为 25000:

```
UPDATE policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-01-01' TO '2009-01-01'
   SET coverage = 25000
   WHERE policy_id = 'C567';
```

对保单 C567 的更新会应用于 2008-01-01 至 2009-01-01 的 BUSINESS_TIME 时间段。policy_info 表中只有一行对应 policy_id C567 并包括此时间段。BUSINESS_TIME 时间段完全包含在该行的 bus_start 和 bus_end 列值内。因此，coverage 值更新为 25000。bus_start 和 bus_end 列值保持不变。

表 50. 保单 C567 已更新

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-07-01
A123	16000	2008-07-01	2009-01-01
B345	18500	2008-03-01	2009-01-01
C567	25000	2008-01-01	2009-01-01

- 保单 A123 的覆盖范围显示在 7 月 1 日 (2008-07-01) 从 12000 增加至 16000，但之前增加至 14000 的记录已丢失:

```
UPDATE policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-06-01' TO '2008-08-01'
   SET coverage = 14000
   WHERE policy_id = 'A123';
```

对保单 A123 的更新会应用于 2008-06-01 至 2008-08-01 的 BUSINESS_TIME 时间段。policy_info 表中有 2 行对应 policy_id A123 并且包括此时间段的部分。

1. BUSINESS_TIME 时间段部分包含在 bus_start 值为 2008-01-01 且 bus_end 值为 2008-07-01 的行的时间段内。此行与指定时间段的开始重叠，因为 BUSINESS_TIME 时间段中的最早时间值大于行 bus_start 值但小于其 bus_end 值。
2. BUSINESS_TIME 时间段部分包含在 bus_start 值为 2008-07-01 且 bus_end 值为 2009-01-01 的行的时间段内。此行与指定时间段的结束重叠，因为 BUSINESS_TIME 时间段中的最晚时间值大于行 bus_start 值但小于其 bus_end 值。

因此，更新会导致出现以下情况:

1. 如果 bus_end 值与指定时间段的开始重叠，那么该行的 coverage 值更新为 14000。在此更新行中，bus_start 值设为 2008-06-01（这是 UPDATE 指定时间段的开始值），bus_end 值保持不变。系统会插入带有该行的原始值的附加行，但其 bus_end 值设为 2008-06-01。此新行反映 coverage 为 12000 时的 BUSINESS_TIME 时间段。
2. 如果 bus_start 值与指定时间段的结束重叠，那么该行的 coverage 值更新为 14000。在此更新行中，bus_start 值保持不变，bus_end 值设为 2008-08-01（这是 UPDATE 指定时间段的结束值）。系统会插入带有该行的原始值的附加行，但其 bus_start 值设为 2008-08-01。此新行反映 coverage 为 16000 时的 BUSINESS_TIME 时间段。

表 51. 保单 A123 已更新

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-06-01
A123	14000	2008-06-01	2008-07-01
A123	14000	2008-07-01	2008-08-01
A123	16000	2008-08-01	2009-01-01
B345	18500	2008-03-01	2009-01-01
C567	25000	2008-01-01	2009-01-01

更多示例

本节包含其他更新应用程序时间段临时表的示例。

合并内容

在以下示例中，MERGE 语句使用 FOR PORTION OF 子句以通过另一个表 (merge_policy) 的内容来更新 policy_info 表。

表 52. merge_policy 表的内容

policy_id	coverage	bus_start	bus_end
C567	30000	2008-10-01	2010-05-01
H789	16000	2008-10-01	2010-05-01

1. 创建全局变量以保存 FOR PORTION OF 子句的 FROM 日期和 TO 日期。

```
CREATE VARIABLE sdate DATE default '2008-10-01';
CREATE VARIABLE edate DATE default '2010-05-01';
```

2. 发出 MERGE 语句，该语句将 merge_policy 的内容合并到通过先前的“过程”部分更新产生的 policy_info 表中。

```
MERGE INTO policy_info pi1
   USING (SELECT policy_id, coverage, bus_start,
bus_end FROM merge_policy) mp2
   ON (pi1.policy_id = mp2.policy_id)
WHEN MATCHED THEN
     UPDATE FOR PORTION OF BUSINESS_TIME FROM sdate TO edate
        SET pi1_coverage = mp2.coverage
WHEN NOT MATCHED THEN
      INSERT (policy_id, coverage, bus_start, bus_end)
       VALUES (mp2.policy_id, mp2.coverage, mp2.bus_start, mp2.bus_end)
```

policy_id C567 对两个表是公共的。merge_policy 中的 C567 bus_start 值与 policy_info 中的 C567 bus_end 值重叠。此语句产生以下项:

- coverage 为 25000 的 bus_end 值设为 2008-10-01。
- 将插入新行，其 coverage 为 30000 且带有来自 merge_policy 的 bus_start 和 bus_end 值。

policy_id H789 仅存在于 merge_policy 中，因此会向 policy_info 添加新行。

表 53. 应用程序时间段临时表 (policy_info) 中已合并的已更新数据

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-06-01

表 53. 应用程序时间段临时表 (policy_info) 中已合并的已更新数据 (续)

policy_id	coverage	bus_start	bus_end
A123	14000	2008-06-01	2008-07-01
A123	14000	2008-07-01	2008-08-01
A123	16000	2008-08-01	2009-01-01
B345	18000	2008-03-01	2009-01-01
C567	25000	2008-01-01	2008-10-01
C567	30000	2008-10-01	2010-05-01
H789	16000	2008-10-01	2010-05-01

更新目标

仅当 UPDATE 语句的目标为表或视图时，才能使用 FOR PORTION OF BUSINESS_TIME 子句。以下 UPDATE 返回错误。

```
UPDATE (SELECT * FROM policy_info) FOR PORTION OF BUSINESS_TIME
   FROM '2008-01-01' TO '06-15-2008' SET policy_id = policy_id + 1;

UPDATE (SELECT * FROM policy_info FOR BUSINESS_TIME AS OF '2008-01-01')
   FOR PORTION OF BUSINESS_TIME FROM '2008-01-01' TO '06-15-2008'
   SET policy_id = policy_id + 1;
```

更新视图

带有对应用程序时间段临时表的引用的视图可进行更新。以下 UPDATE 将更新 policy_info 表。

```
CREATE VIEW viewC AS SELECT * FROM policy_info;
UPDATE viewC SET coverage = coverage + 5000;
```

其 FROM 子句中指定了时间段且带有应用程序时间段临时表的视图也可进行更新。此条件不同于系统时间段临时表和双时态表的视图。

```
CREATE VIEW viewD AS SELECT * FROM policy_info
   FOR BUSINESS_TIME AS OF CURRENT DATE;
UPDATE viewD SET coverage = coverage - 1000;
```

可针对带有对应用程序时间段临时表或双时态表的引用的视图包括 FOR PORTION OF 更新子句。这类更新会传播至视图定义的 FROM 子句中引用的临时表。

```
CREATE VIEW viewE AS SELECT * FROM policy_info;
UPDATE viewE FOR PORTION OF BUSINESS_TIME
   FROM '2009-01-01' TO '2009-06-01' SET coverage = coverage + 500;
```

从应用程序时间段临时表中删除数据

从应用程序时间段临时表中删除数据会从该表中删除行，并可能导致向应用程序时间段临时表本身插入新行。

关于此任务

除了常规 DELETE 语句外，应用程序时间段临时表还支持时间范围删除，在这类删除中，DELETE 语句包括 FOR PORTION OF BUSINESS_TIME 子句。如果一行的时间段开始列和/或时间段结束列在 FOR PORTION OF BUSINESS_TIME 子句中指定的范围内，那么该行是删除操作的候选项。

过程

要从应用程序时间段临时表中删除数据，请使用 DELETE FROM 语句来删除数据。 例如，发现保单 A123 不应提供 2008 年 6 月 15 日至 2008 年 8 月 15 日的覆盖范围，因此应从在『更新应用程序时间段临时表中的数据』主题中已更新的表中删除该数据。

```
DELETE FROM policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-06-15' TO '2008-08-15'
   WHERE policy_id = 'A123';
```

结果

原始 policy_info 表数据如下所示:

表 54. 执行 DELETE 语句前应用程序时间段临时表 (policy_info) 中的数据

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-06-01
A123	14000	2008-06-01	2008-07-01
A123	14000	2008-07-01	2008-08-01
A123	16000	2008-08-01	2009-01-01
B345	18000	2008-03-01	2009-01-01
C567	25000	2008-01-01	2009-01-01

通过使用 FOR PORTION OF BUSINESS_TIME 子句从应用程序时间段临时表中删除数据会导致删除行，并可能导致插入行（如果一行的时间段覆盖 DELETE FROM 语句中指定的范围的一部分）。删除与保单 A123 相关的数据的操作会应用于 2008-06-15 至 2008-08-15 的 BUSINESS_TIME 时间段。policy_info 表中有 3 行对应 policy_id A123 并且包括该时间段的全部或部分。

对保单 A123 的更新会影响系统时间段临时表及其历史记录表，导致出现以下情况:

- 有 1 行符合以下条件: DELETE FROM 语句中的 BUSINESS_TIME 时间段覆盖一行的整个时间段。bus_start 值为 2008-07-01 且 bus_end 值为 2008-08-01 的行被删除。
- 如果只有 bus_end 值在指定时间段内，那么该行被删除。系统会插入带有所删除行的原始值的新行，但其 bus_end 值设为 2008-06-15。
- 如果只有 bus_start 值在指定时间段内，那么该行被删除。系统会插入带有所删除行的原始值的新行，但其 bus_start 值设为 2008-08-15。

表 55. 执行 DELETE 语句后应用程序时间段临时表 (policy_info) 中的数据

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-06-01
A123	14000	2008-06-01	2008-06-15
A123	16000	2008-08-15	2009-01-01
B345	18000	2008-03-01	2009-01-01
C567	25000	2008-01-01	2009-01-01

示例

本节包含其他删除应用程序时间段临时表的示例。

删除目标

仅当 DELETE 语句的目标为表或视图时，才能使用 FOR PORTION OF BUSINESS_TIME 子句。以下 DELETE 语句返回错误:

```
DELETE FROM (SELECT * FROM policy_info) FOR PORTION OF BUSINESS_TIME
   FROM '2008-01-01' TO '2008-06-15';
```

查询应用程序时间段临时数据

查询应用程序时间段临时表可返回指定时间段的结果。

关于此任务

查询应用程序时间段临时表时，可在 FROM 子句中包括 FOR BUSINESS_TIME。通过使用 FOR BUSINESS_TIME 规范，可查询数据的当前状态、过去状态和将来状态。时间段按如下所示指定:

AS OF *value1*

包括符合后述条件的所有行，这些行的时间段的开始值小于或等于 *value1*，结束值大于 *value1*。

FROM *value1* **TO** *value2*

包括符合后述条件的所有行，这些行的时间段的开始值大于或等于 *value1*，结束值小于 *value2*。这意味着开始时间包括在该时间段内，但结束时间不包括在该时间段内。

BETWEEN *value1* **AND** *value2*

包括符合后述条件的所有行，这些行的任何时间段与 *value1* 与 *value2* 之间的任意时间点重叠。如果该时间段的开始值小于或等于 *value2* 且该时间段的结束值大于 *value1*，那么返回一行。

请参阅以下部分以获取某些样本查询。

过程

要查询应用程序时间段临时表，请使用 SELECT 语句。例如，以下每个查询请求来自『更新应用程序时间段临时表中的数据』主题的结果表中对应 policy_id A123 的保单信息。每个查询指定的时间段不同。
policy_info 表如下所示:

表 56. 应用程序时间段临时表: policy_info

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-06-01
A123	14000	2008-06-01	2008-06-15
A123	16000	2008-08-15	2009-01-01
B345	18000	2008-03-01	2009-01-01
C567	25000	2008-01-01	2009-01-01

- 未指定时间段的查询。 例如:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   where policy_id = 'A123'
```

此查询将保单 A123 的三行全部返回。

```
A123, 12000, 2008-01-01, 2008-06-01
A123, 14000, 2008-06-01, 2008-06-15
A123, 16000, 2008-08-15, 2009-01-01
```

- 指定了 FOR BUSINESS_TIME AS OF 的查询。 例如:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR BUSINESS_TIME AS OF '2008-07-15'
   where policy_id = 'A123'
```

此查询不返回任何行。A123 没有以下对应行，这些行的时间段的开始值小于或等于 *2008-07-15*，结束值大于 *2008-07-15*。保单 A123 没有对应 *2008-07-15* 的覆盖范围。

- 指定了 FOR BUSINESS_TIME FROM...TO 的查询。 例如:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR BUSINESS_TIME FROM
      '2008-01-01' TO '2008-06-15'
   where policy_id = 'A123'
```

此查询返回两行。时间段的开始列包含开始值和结束值，结束列不包含开始值和结束值。bus_end 值为 2008-06-15 的行的有效期直到 2008 年 6 月 14 日午夜，因此小于 *value2*。

```
A123, 12000, 2008-01-01, 2008-06-01
A123, 14000, 2008-06-01, 2008-06-15
```

- 指定了 FOR BUSINESS_TIME BETWEEN...AND 的查询。 例如:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR BUSINESS_TIME BETWEEN
      '0001-01-01' AND '2008-01-01'
```

此查询返回两行。具有 bus_start 的值为 2008-01-01 的行等于 *value1* 并将返回，因为包括了时间段的开始时间。注意，如果某个行具有 bus_end 列，其值为 2008-01-01，那么将返回该行，因为其结束时间等于 *value1*，且包括时间段的结束时间。

```
A123, 12000, 2008-01-01, 2008-06-01
C567, 25000, 2008-01-01, 2009-01-01
```

更多示例

本节包含其他查询应用程序时间段临时表的示例。

查询视图

可像查询应用程序时间段临时表那样查询视图。可在视图引用后指定时间段 (FOR BUSINESS_TIME)。

```
CREATE VIEW policy_year_end(policy, amount, start_date, end_date)
   AS SELECT * FROM policy_info;

SELECT * FROM policy_year_end FOR BUSINESS_TIME AS OF '2008-12-31';
```

针对视图 policy_year_end 的 SELECT 会查询 policy_info 表并返回 2008 年底生效的所有保单。

```
A123, 16000, 2008-08-15, 2009-01-01
B345, 18000, 2008-03-01, 2009-01-01
C567, 25000, 2008-01-01, 2009-01-01
```

如果视图定义包含时间段指定，那么针对该视图的查询不能包含时间段指定。以下语句返回错误，因为指定了多个时间段:

```
CREATE VIEW all_policies AS SELECT * FROM policy_info;
   FOR BUSINESS_TIME AS OF '2008-02-28';

SELECT * FROM all_policies FOR BUSINESS_TIME BETWEEN
   FOR BUSINESS_TIME AS OF '2008-10-01';
```

为会话设置应用程序时间

在 CURRENT TEMPORAL BUSINESS_TIME 专用寄存器中设置应用程序时间可减少或消除对不同时间点运行应用程序时所需的更改。

关于此任务

如果要对应用程序时间段临时表运行某个应用程序以针对许多不同日期查询业务状态，那么可在专用寄存器中设置日期。如果需要查询截至今天、截至上一季度结束或截至某个将来日期（如果您要模拟将来事件）的数据，那么可能无法更改应用程序并向每个 SQL 语句添加 AS OF 指定。使用打包应用程序时，可能存在此限制。要解决这类情况，可使用 CURRENT TEMPORAL BUSINESS_TIME 专用寄存器以在会话级别设置日期。

设置 CURRENT TEMPORAL BUSINESS_TIME 专用寄存器不会影响常规表。只有针对已启用 BUSINESS_TIME 时间段的临时表（应用程序时间段临时表和双时态表）的查询才会使用专用寄存器中设置的时间。它对 DDL 语句也没有影响。

注: 如果 CURRENT TEMPORAL BUSINESS_TIME 专用寄存器设为非空值，那么针对应用程序时间段临时表执行的数据修改语句（例如，INSERT、UPDATE、DELETE 和 MERGE）受支持。此行为与 CURRENT TEMPORAL SYSTEM_TIME 专用寄存器的行为不同，后者会阻止对系统时间段临时表和双时态表执行数据修改语句。

BUSTIMESENSITIVE 绑定选项的设置决定程序包中静态和动态 SQL 语句中对应用程序时间段临时表和双时态表的引用是否受 CURRENT TEMPORAL BUSINESS_TIME 专用寄存器的值影响。可将该绑定选项设为 YES 或 NO。对于 SQL 过程，请使用 SET_ROUTINE_OPTS 过程来设置称为查询编译器变量的类似绑定选项。

过程

如果此专用寄存器设为非空值，那么发出查询的应用程序返回截至该日期的数据。以下示例请求来自“从应用程序时间段临时表中删除数据”主题的结果表中的信息。

- 将专用寄存器设为非空值并查询截至该日期的数据。 例如:

  ```
  SET CURRENT TEMPORAL BUSINESS_TIME = '2008-01-01';
  SELECT * FROM policy_info;
  ```

- 将专用寄存器设为某个时间并在视图定义中引用应用程序时间段临时表。

  ```
  CREATE VIEW view1 AS SELECT policy_id, coverage FROM policy_info;
  CREATE VIEW view2 AS SELECT * FROM regular_table
     WHERE col1 IN (SELECT coverage FROM policy_info);
  SET CURRENT TEMPORAL BUSINESS_TIME = '2008-01-01';
  SELECT * FROM view1;
  SELECT * FROM view2;
  ```

- 将专用寄存器设为过去日期并发出包含时间段指定的查询。 例如:

```
SET CURRENT TEMPORAL BUSINESS_TIME = CURRENT DATE - 1 YEAR;
SELECT * FROM policy_info FOR BUSINESS_TIME AS OF '2008-01-01';
```

结果

policy_info 表如下所示:

表 57. 执行 DELETE 语句后应用程序时间段临时表 (policy_info) 中的数据

policy_id	coverage	bus_start	bus_end
A123	12000	2008-01-01	2008-06-01
A123	14000	2008-06-01	2008-06-15
A123	16000	2008-08-15	2009-01-01
B345	18000	2008-03-01	2009-01-01
C567	25000	2008-01-01	2009-01-01

- 如果请求截至 2008-01-01 的数据，那么将查询 policy_info 表。该查询隐式重写为:

```
SELECT * FROM policy_info FOR BUSINESS_TIME AS OF '2008-01-01';
```

  该查询返回:

```
A123, 12000, 2008-01-01, 2008-06-01
C567, 25000, 2008-01-01, 2009-01-01
```

- 针对 view1 的查询隐式重写为:

```
SELECT * FROM view1 FOR BUSINESS_TIME AS OF CURRENT TEMPORAL BUSINESS_TIME;
```

  然后重写为:

```
SELECT policy_id, coverage FROM policy_info
   FOR BUSINESS_TIME AS OF '2008-01-01';
```

  该查询返回:

```
A123, 12000
C567, 25000
```

  针对 view2 的查询涉及常规表的引用应用程序时间段临时表的视图，这会导致在常规表与专用寄存器之间形成隐式关系。该查询隐式重写为:

```
SELECT * FROM view2 FOR BUSINESS_TIME AS OF CURRENT TEMPORAL BUSINESS_TIME;
```

  然后重写为:

```
SELECT * FROM regular_table WHERE col1 in (SELECT coverage FROM policy_info
   FOR BUSINESS_TIME AS OF '2008-01-01');
```

  该 SELECT 返回以下行，这些行的 col1 值与 coverage 中的值相匹配。

- 返回了错误，因为指定了多个时间段。专用寄存器设为非空值，并且该查询也指定了时间。

双时态表

双时态表是将系统时间段临时表的历史跟踪功能与应用程序时间段临时表的特定于时间的数据存储功能组合到一起的表。可使用双时态表保存基于用户的时间段信息以及基于系统的历史信息。

双时态表的行为方式如同系统时间段临时表与应用程序时间段临时表的组合。适用于系统时间段临时表和应用程序时间段临时表的所有限制也适用于双时态表。

创建双时态表

创建双时态表将产生以下表，该表将系统时间段临时表的历史跟踪功能与应用程序时间段临时表的特定于时间的数据存储功能组合到一起。

关于此任务

创建双时态表时，您将用于创建系统时间段临时表的步骤与用于创建应用程序时间段临时表的步骤组合到一起。

- 在 CREATE TABLE 语句中包括 SYSTEM_TIME 时间段和 BUSINESS_TIME 时间段。
- 创建历史记录表以从双时态表中接收旧行。
- 添加版本控制以在双时态表与历史记录表之间建立链接。
- 可选择定义不允许 BUSINESS_TIME 的时间段重叠，并且值对于任何时间段都是唯一。

以下部分中的示例说明如何创建用于存储保险公司客户的保单信息的表。

过程

要创建双时态表，请执行以下操作:

1. 创建带有 SYSTEM_TIME 属性和 BUSINESS_TIME 属性的表。 例如:

```
CREATE TABLE policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 bus_start       DATE NOT NULL,
 bus_end         DATE NOT NULL,
 sys_start   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW BEGIN,
 sys_end  TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END,
 ts_id    TIMESTAMP(12) NOT NULL
             GENERATED ALWAYS AS TRANSACTION START ID,
 PERIOD BUSINESS_TIME (bus_start, bus_end),
 PERIOD SYSTEM_TIME (sys_start, sys_end)
) in policy_space;
```

2. 创建历史记录表。 例如:

```
CREATE TABLE hist_policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 bus_start       DATE NOT NULL,
 bus_end         DATE NOT NULL,
 sys_start  TIMESTAMP(12) NOT NULL,
 sys_end  TIMESTAMP(12) NOT NULL,
 ts_id    TIMESTAMP(12)
) in hist_space;
```

   还可通过使用 CREATE TABLE 语句的 LIKE 子句，创建列名称和描述与系统时间段临时表的列名称和描述相同的历史记录表。例如:

```
CREATE TABLE hist_policy_info LIKE policy_info in hist_space;
```

3. 向双时态表添加版本控制。 例如:

```
ALTER TABLE policy_info ADD VERSIONING USE HISTORY TABLE hist_policy_info;
```

4. 可选: 创建包括 BUSINESS_TIME 时间段的唯一索引。 例如:

```
CREATE UNIQUE INDEX ix_policy
   ON policy_info (policy_id, BUSINESS_TIME WITHOUT OVERLAPS);
```

结果

policy_info 表存储客户的保险覆盖范围级别。与 BUSINESS_TIME 时间段相关的列（bus_start 和 bus_end）指示保险覆盖范围级别何时有效。与 SYSTEM_TIME 时间段相关的列（sys_start 和 sys_end）显示覆盖范围级别行何时为最新版本。ts_id 列用于列示开始执行影响该行的事务的时间。

表 58. 创建的双时态表 (policy_info)

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id

hist_policy_info 历史记录表从 policy_info 表接收旧行。

表 59. 创建的历史记录表 (hist_policy_info)

policy_id	coverage	bus_start	bus_end	sys_start	sys_end	ts_id

将 BUSINESS_TIME WITHOUT OVERLAPS 作为索引键列的列表中的最终列的 ix_policy 索引确保客户保险覆盖范围级别没有重叠时间段。

示例

本节包含其他创建双时态表的示例。

隐藏列 以下示例创建 policy_info 表并将 TIMESTAMP(12) 列（sys_start、sys_end 和 ts_id）标记为隐式隐藏。

```
CREATE TABLE policy_info
(
 policy_id       CHAR(4) NOT NULL,
 coverage        INT NOT NULL,
 bus_start       DATE NOT NULL,
 bus_end         DATE NOT NULL,
 sys_start   TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW
BEGIN IMPLICITLY HIDDEN,
 sys_end  TIMESTAMP(12) NOT NULL GENERATED ALWAYS AS ROW END
IMPLICITLY HIDDEN,
 ts_id    TIMESTAMP(12)
             GENERATED ALWAYS AS TRANSACTION START ID IMPLICITLY HIDDEN,
 PERIOD BUSINESS_TIME (bus_start, bus_end),
 PERIOD SYSTEM_TIME (sys_start, sys_end)
) in policy_space;
```

使用 CREATE TABLE 语句的 LIKE 子句创建 hist_policy_info 历史记录表会生成从 policy_info 表继承隐式隐藏属性的历史记录表。

在双时态表中插入数据

在双时态表中插入数据类似在应用程序时间段临时表中插入数据。

关于此任务

在双时态表中插入数据时，应包括用于捕获该行有效（从关联业务应用程序的角度看）的时间的开始列和结束列。此有效时间段称为 BUSINESS_TIME 时间段。数据库管理器自动生成隐式检查约束，该检查约束确保 BUSINESS_TIME 时间段的开始列小于其结束列。如果为该表创建了带有 BUSINESS_TIME WITHOUT OVERLAPS 的唯一约束或索引，请确保没有 BUSINESS_TIME 时间段重叠。

过程

要在双时态表中插入数据，请使用 INSERT 语句向该表添加数据。 例如，2010 年 1 月 31 日 (2010-01-31) 向“创建双时态表”的示例中创建的表插入了以下数据。

```
INSERT INTO policy_info(policy_id, coverage, bus_start, bus_end)
   VALUES('A123',12000,'2008-01-01','2008-07-01');

INSERT INTO policy_info(policy_id, coverage, bus_start, bus_end)
   VALUES('A123',16000,'2008-07-01','2009-01-01');

INSERT INTO policy_info(policy_id, coverage, bus_start, bus_end)
   VALUES('B345',18000,'2008-01-01','2009-01-01');

INSERT INTO policy_info(policy_id, coverage, bus_start, bus_end)
   VALUES('C567',20000,'2008-01-01','2009-01-01');
```

结果

policy_info 表现在包含以下保险覆盖范围数据。数据库管理器生成 sys_start、sys_end 和 ts_id 列条目。时间段的开始列包括开始值和结束值，结束列不包括开始值和结束值，这意味着 bus_end 值为 2008-07-01 的行的 BUSINESS_TIME 时间段未与 bus_start 值为 2008-07-01 的行的时间段重叠。

表 60. 添加至双时态表 (policy_info) 的数据

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000

hist_policy_info 历史记录表仍然为空，因为插入操作未生成任何历史记录行。

表 61. 执行插入后的历史记录表 (hist_policy_info)

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id

更新双时态表中的数据

更新双时态表中的数据会导致向其关联历史记录表添加行，并可能导致向双时态表本身添加行。

关于此任务

除了常规 UPDATE 语句外，双时态表还支持时间范围更新，在这类更新中，UPDATE 语句包括 FOR PORTION OF BUSINESS_TIME 子句。如果一行的时间段开始列和/或时间段结束列在 FOR PORTION OF BUSINESS_TIME 子句中指定的范围内，那么该行是更新操作的候选项。所有现有受影响行在被更新之前会复制到历史记录表中。

过程

要更新双时态表中的数据，请使用 UPDATE 语句来更改数据行。例如，发现两个客户的保险覆盖范围级别存在某些错误，并且 2011 年 2 月 28 日 (2011-02-28) 在“在双时态表中插入数据”主题中添加了数据的示例表中更新了以下数据。
下表为原始 `policy_info` 表数据。

表 62. 双时态表 (policy_info) 中的原始数据

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000

通过使用 FOR PORTION OF BUSINESS_TIME 子句更新双时态表会导致更改行，并可能导致插入行（如果已更新行的现有时间段未完全包含在 UPDATE 语句中指定的范围内）。

- 保单 B345 的覆盖范围实际从 2008 年 3 月 1 日 (2008-03-01) 开始:

```
UPDATE policy_info
  SET bus_start='2008-03-01'
  WHERE policy_id = 'B345'
  AND coverage = 18000;
```

对保单 B345 的更新使用常规 UPDATE 语句。policy_info 表中只有一行对应 policy_id B345，所以没有潜在的 BUSINESS_TIME 时间段重叠。因此会出现以下情况:

1. bus_start 列值更新为 2008-03-01。请注意，对 BUSINESS_TIME 时间段列的更新不能包括 FOR PORTION OF BUSINESS_TIME 子句。
2. 数据库管理器将 sys_start 和 ts_id 值更新为更新日期。
3. 原始行移至历史记录表。数据库管理器将 sys_end 值更新为更新日期。

下表显示保单 B345 的更新。

表 63. 更新保单 B345 后的双时态表 (policy_info)

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000

表 64. 更新保单 B345 后的历史记录表 (hist_policy_info)

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

- 保单 C567 对应 2008 年的覆盖范围应该为 25000:

```
UPDATE policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-01-01' TO '2009-01-01'
   SET coverage = 25000
   WHERE policy_id = 'C567';
```

对保单 C567 的更新会应用于 2008-01-01 至 2009-01-01 的 BUSINESS_TIME 时间段。policy_info 表中只有一行对应 policy_id C567 并包括该时间段。BUSINESS_TIME 时间段完全包含在该行的 bus_start 和 bus_end 列值内。因此会出现以下情况:

1. policy_id 为 C567 的行的 coverage 值更新为 25000。
2. bus_start 和 bus_end 列值保持不变。
3. 数据库管理器将 sys_start 和 ts_id 值更新为更新日期。
4. 原始行移至历史记录表。数据库管理器将 sys_end 值更新为更新日期。

下表显示保单 C567 的更新。

表 65. 更新保单 C567 后的双时态表 (policy_info)

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	9999-12-30-00.00.00.000000000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

表 66. 更新保单 C567 后的历史记录表 (hist_policy_info)

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

- 保单 A123 的覆盖范围显示在 7 月 1 日 (2008-07-01) 从 12000 增加至 16000，但之前增加至 14000 的记录已丢失:

```
UPDATE policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-06-01' TO '2008-08-01'
   SET coverage = 14000
   WHERE policy_id = 'A123';
```

对保单 A123 的更新会应用于 2008-06-01 至 2008-08-01 的 BUSINESS_TIME 时间段。policy_info 表中有 2 行对应 policy_id A123 并且包括更新时间段的部分。

1. BUSINESS_TIME 时间段部分包含在 bus_start 值为 2008-01-01 且 bus_end 值为 2008-07-01 的行的时间段内。此行与指定时间段的开始重叠，因为 BUSINESS_TIME 时间段中的最早时间值大于行 bus_start 值但小于其 bus_end 值。
2. BUSINESS_TIME 时间段部分包含在 bus_start 值为 2008-07-01 且 bus_end 值为 2009-01-01 的行的时间段内。此行与指定时间段的结束重叠，因为 BUSINESS_TIME 时间段中的最晚时间值大于行 bus_start 值但小于其 bus_end 值。

因此会出现以下情况:

1. 如果 bus_end 值与指定时间段的开始重叠，那么该行的 coverage 值更新为 14000。在此更新行中，bus_start 值设为 2008-06-01（这是 UPDATE 指定时间段的开始值），bus_end 值保持不变。系统会插入带有该行的原始值的附加行，但

其 bus_end 值设为 2008-06-01。此新行反映 coverage 为 12000 时的 BUSINESS_TIME 时间段。数据库管理器生成 sys_start、sys_end 和 ts_id 列条目。

2. 如果 bus_start 值与指定时间段的结束重叠，那么该行的 coverage 值更新为 14000。在此更新行中，bus_start 值保持不变，bus_end 值设为 2008-08-01（这是 UPDATE 指定时间段的结束值）。系统会插入带有该行的原始值的附加行，但其 bus_start 值设为 2008-08-01。此新行反映 coverage 为 16000 时的 BUSINESS_TIME 时间段。数据库管理器生成 sys_start、sys_end 和 ts_id 列条目。
3. 原始行移至历史记录表。数据库管理器将 sys_end 值更新为更新日期。

下表显示保单 A123 的更新。

表 67. 更新保单 A123 后的双时态表 (policy_info)

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-06-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-06-01	2008-07-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-07-01	2008-08-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	16000	2008-08-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

表 68. 更新保单 A123 后的历史记录表 (hist_policy_info)

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

表 68. 更新保单 A123 后的历史记录表 (hist_policy_info) (续)

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

从双时态表中删除数据

从双时态表中删除数据会导致从该表中删除行，向其关联历史记录表添加行并（可能）在双时态表本身插入新行。

关于此任务

除了常规 DELETE 语句外，双时态表还支持时间范围删除，在这类删除中，DELETE 语句包括 FOR PORTION OF BUSINESS_TIME 子句。如果一行的时间段开始列和/或时间段结束列在 FOR PORTION OF BUSINESS_TIME 子句中指定的范围内，那么该行是删除操作的候选项。所有现有受影响行在被删除之前会复制到历史记录表中。

过程

要从双时态表中删除数据，请使用 DELETE FROM 语句。 例如，发现保单 A123 没有 2008 年 6 月 15 日至 2008 年 8 月 15 日的覆盖范围。已在 2011 年 9 月 1 日 (2011-09-01) 从“更新双时态表中的数据”主题中已更新的表中删除该数据。

```
DELETE FROM policy_info
   FOR PORTION OF BUSINESS_TIME FROM '2008-06-15' TO '2008-08-15'
   WHERE policy_id = 'A123';
```

结果

原始 policy_info 表和 hist_policy_info 表数据如下所示:

表 69. 执行 DELETE 语句前双时态表 (policy_info) 中的数据

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-06-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-06-01	2008-07-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	14000	2008-07-01	2008-08-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
A123	16000	2008-08-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

表 69. 执行 DELETE 语句前双时态表 (policy_info) 中的数据 (续)

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

表 70. 执行 DELETE 语句前历史记录表 (hist_policy_info) 中的数据

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000

通过使用 FOR PORTION OF BUSINESS_TIME 子句从双时态表中删除数据会导致删除行，并可能导致插入行（如果一行的时间段覆盖 DELETE FROM 语句中指定的范围的一部分）。删除与保单 A123 相关的数据的操作会应用于 2008-06-15 至 2008-08-15 的 BUSINESS_TIME 时间段。policy_info 表中有 3 行对应 policy_id A123 并且包括该时间段的全部或部分。

因此会出现以下情况:

- 有 1 行符合以下条件: DELETE FROM 语句中的 BUSINESS_TIME 时间段覆盖一行的整个时间段。bus_start 值为 2008-07-01 且 bus_end 值为 2008-08-01 的行被删除。
- 如果只有 bus_end 值在指定时间段内，那么该行被删除。系统会插入带有所删除行的原始值的新行，但其 bus_end 值设为 2008-06-15。数据库管理器生成 sys_start、sys_end 和 ts_id 列条目。
- 如果只有 bus_start 值在指定时间段内，那么该行被删除。系统会插入带有所删除行的原始值的新行，但其 bus_start 值设为 2008-08-15。数据库管理器生成 sys_start、sys_end 和 ts_id 列条目。
- 原始行移至历史记录表。数据库管理器将 sys_end 值更新为删除日期。

表 71. 执行 DELETE 语句后双时态表 (policy_info) 中的数据

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-06-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

表 71. 执行 DELETE 语句后双时态表 (policy_info) 中的数据 (续)

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	14000	2008-06-01	2008-06-15	2011-09-01-12.18.22.959254000000	9999-12-30-00.00.00.000000000000	2011-09-01-12.18.22.959254000000
A123	16000	2008-08-15	2009-01-01	2011-09-01-12.18.22.959254000000	9999-12-30-00.00.00.000000000000	2011-09-01-12.18.22.959254000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.649592000000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.649592000000

表 72. 执行 DELETE 语句后的历史记录表 (hist_policy_info)

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.649592000000	2010-01-31-22.31.33.495925000000
A123	14000	2008-06-01	2008-07-01	2011-02-28-09.10.12.649592000000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000
A123	14000	2008-07-01	2008-08-01	2011-02-28-09.10.12.649592000000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000
A123	16000	2008-08-01	2009-01-01	2011-02-28-09.10.12.649592000000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000

查询双临时数据

查询双时态表可返回指定时间段的结果。这些结果可能包括当前值、先前历史值和将来值。

关于此任务

查询双时态表时，可在 FROM 子句中包括 FOR BUSINESS_TIME 和/或 FOR SYSTEM_TIME。通过使用这些时间段指定，可查询数据的当前状态、过去状态和将来状态。时间段按如下所示指定:

AS OF *value1*
包括符合后述条件的所有行，这些行的时间段的开始值小于或等于 *value1*，结束值大于 *value1*。这允许您查询从某个时间点开始的数据。

FROM *value1* **TO** *value2*
包括符合后述条件的所有行，这些行的时间段的开始值等于或大于 *value1*，结束值小于 *value2*。这意味着开始时间包括在该时间段内，但结束时间不包括在该时间段内。

BETWEEN *value1* **AND** *value2*
包括符合后述条件的所有行，这些行的任何时间段与 *value1* 与 *value2* 之间的任意时间点重叠。如果该时间段的开始值小于或等于 *value2* 且该时间段的结束值大于 *value1*，那么返回一行。

请参阅以下部分以获取某些样本查询。

过程

要查询双时态表，请使用 SELECT 语句。例如，以下每个查询请求来自“从双时态表中删除数据”主题的结果表中对应 `policy_id` A123 的保单信息。每个查询指定的时间段不同。
`policy_info` 表及其关联历史记录表如下所示:

表 73. 双时态表: policy_info

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-06-01	2011-02-28-09.10.12.64959200000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.64959200000
A123	14000	2008-06-01	2008-06-15	2011-09-01-12.18.22.959254000000	9999-12-30-00.00.00.000000000000	2011-09-01-12.18.22.959254000000
A123	16000	2008-08-15	2009-01-01	2011-09-01-12.18.22.959254000000	9999-12-30-00.00.00.000000000000	2011-09-01-12.18.22.959254000000
B345	18000	2008-03-01	2009-01-01	2011-02-28-09.10.12.64959200000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.64959200000
C567	25000	2008-01-01	2009-01-01	2011-02-28-09.10.12.64959200000	9999-12-30-00.00.00.000000000000	2011-02-28-09.10.12.64959200000

表 74. 历史记录表: hist_policy_info

policy_id	cover-age	bus_start	bus_end	sys_start	sys_end	ts_id
A123	12000	2008-01-01	2008-07-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.64959200000	2010-01-31-22.31.33.495925000000
A123	16000	2008-07-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.64959200000	2010-01-31-22.31.33.495925000000
B345	18000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.64959200000	2010-01-31-22.31.33.495925000000
C567	20000	2008-01-01	2009-01-01	2010-01-31-22.31.33.495925000000	2011-02-28-09.10.12.64959200000	2010-01-31-22.31.33.495925000000
A123	14000	2008-06-01	2008-07-01	2011-02-28-09.10.12.64959200000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000
A123	14000	2008-07-01	2008-08-01	2011-02-28-09.10.12.64959200000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000
A123	16000	2008-08-01	2009-01-01	2011-02-28-09.10.12.64959200000	2011-09-01-12.18.22.959254000000	2011-09-01-12.18.22.959254000000

- 未指定时间段的查询。 例如:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   where policy_id = 'A123'
```

此查询返回三行。该 SELECT 语句仅查询 policy_info 表。不会查询历史记录表，因为未指定 FOR SYSTEM_TIME。

```
A123, 12000, 2008-01-01, 2008-06-01
A123, 14000, 2008-06-01, 2008-06-15
A123, 16000, 2008-08-15, 2009-01-01
```

- 指定了 FOR SYSTEM_TIME FROM...TO 的查询。 例如:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR SYSTEM_TIME FROM
      '0001-01-01-00.00.00.000000' TO '9999-12-30-00.00.00.000000000000'
   where policy_id = 'A123'
```

此查询返回八行。该 SELECT 语句查询 policy_info 表和 hist_policy_info 表。

```
A123, 12000, 2008-01-01, 2008-06-01
A123, 14000, 2008-06-01, 2008-06-15
A123, 16000, 2008-08-15, 2009-01-01
A123, 12000, 2008-01-01, 2008-07-01
A123, 16000, 2008-07-01, 2009-01-01
A123, 14000, 2008-06-01, 2008-07-01
A123, 14000, 2008-07-01, 2008-08-01
A123, 16000, 2008-08-01, 2009-01-01
```

- 指定了 FOR BUSINESS_TIME AS OF 的查询。 例如:

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR BUSINESS_TIME AS OF '2008-07-15'
   where policy_id = 'A123'
```

此查询不返回任何行。该 SELECT 语句仅查询 policy_info 表，A123 没有符合以下条件的对应行：这些行的时间段的开始值小于或等于 *2008-07-15*，结束值大于 *2008-07-15*。保单 A123 没有对应 2008-07-15 的覆盖范围。不会查询历史记录表，因为未指定 FOR SYSTEM_TIME。

- 指定了 FOR BUSINESS_TIME AS OF 和 FOR SYSTEM_TIME FROM...TO 的查询。 例如：

```
SELECT policy_id, coverage, bus_start, bus_end
   FROM policy_info
   FOR BUSINESS_TIME AS OF '2008-07-15'
   FOR SYSTEM_TIME FROM
      '0001-01-01-00.00.00.000000' TO '9999-12-30-00.00.00.000000000000'
   where policy_id = 'A123'
```

此查询返回两行。该 SELECT 查询 policy_info 表和 hist_policy_info 表。所返回行可在历史记录表中找到。

```
A123, 16000, 2008-07-01, 2009-01-01
A123, 14000, 2008-07-01, 2008-08-01
```

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 288 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存取方案，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 75 列示了在启用乐观锁定后可能出现的情况。

表 75. 启用乐观锁定后可能出现的情况

情况标识	操作	结果
方案 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。
方案 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。 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。
情况 5	更改了表以便包含行更改时间戳记列，并且在更改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。

表 75. 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 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列实施乐观锁定，隐式隐藏列是使用 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 13 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*（也称为*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 $20000。

*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。

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 38 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:

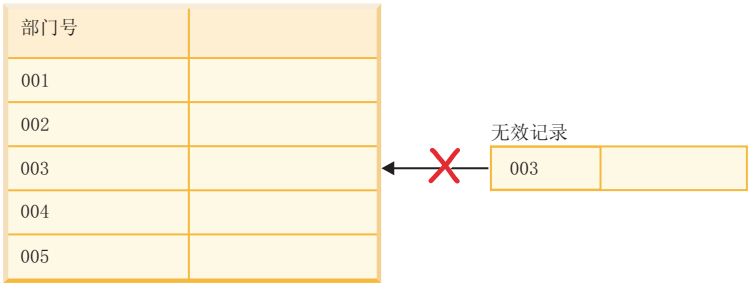


图 38. 唯一约束防止出现重复数据

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，可以在创建或更改表时定义。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 358 页的图 39 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:

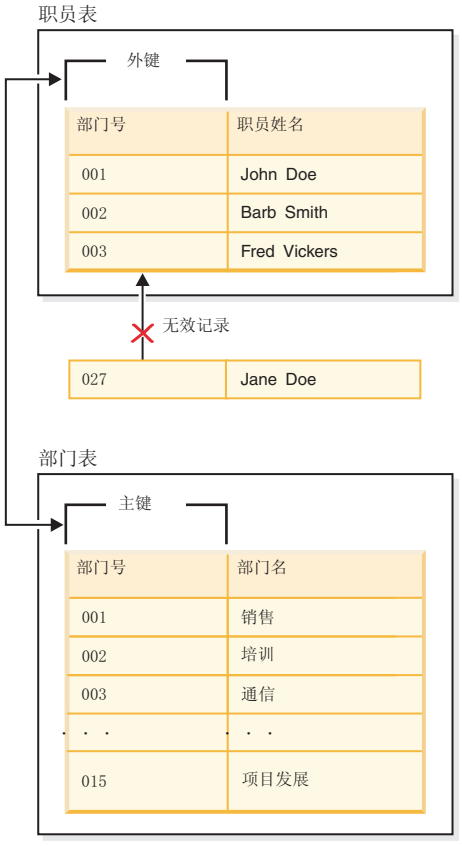


图 39. 外键和主键约束

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 76. 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*。
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（在以下一节中定义）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、表分区键列或 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。先添加引用完整性或检查约束，然后将约束属性与它们相关联，以指定数据库管理器是否强制实施该约束。对于检查约束，可进一步指定能否信任该约束。对于引用完整性约束，如果未强制实施该约束，那么可进一步指定能否信任该约束。未强制实施并且不可信的约束又称为统计引用完整性约束。指定该约束后，可指定是否将该约束用于查询优化。

参考 RI（引用完整性）约束用于对查询性能以及REFRESH IMMEDIATE MQT 的增量处理和登台表进行优化。如果违反参考约束，MQT 数据和登台表可能会不正确。

例如，维护父/子表的顺序很重要。如果要向父/子表中添加行，必须首先将行插入父表。要从父/子表中除去行，必须先从子表中删除行。这样就能始终确保子表中没有孤行。否则，参考约束违例就可能会影响在表维护期间执行的查询的正确性，以及从属 MQT 数据和登台表的增量维护的正确性。

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。

关于此任务

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。

关于此任务

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** 或 **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。此操作避免了装入该表时维护索引。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。

限制

- 不能对子表定义唯一约束。
- 每个表只能有一个主键。

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。

关于此任务

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，可以在创建或更改表时使用 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 361 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的一种: (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。

关于此任务

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）。以下示例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，必须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，它也会产生一定量的系统活动，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。

关于此任务

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 367 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 40 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。

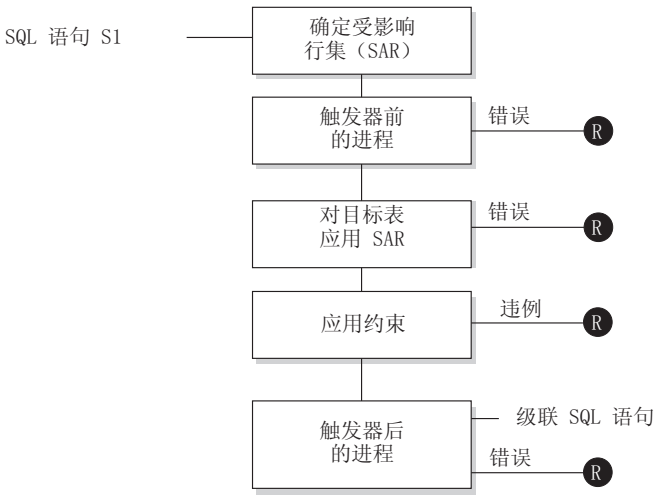


图 40. 处理包含相关触发器和约束的语句

图 40 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 40 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，您可以执行以下操作:

- 创建带有或不带外键的表。
- 在创建或更改表时定义外键。
- 在更改表时删除外键。

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，如果删除服务器或更改服务器定义，那么任何引用了该服务器的相关联昵称的程序包或高速缓存动态 SQL 都将失效。

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，必须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活动，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,
 );
```

此示例包含两个选项，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。可进一步将此选项指定为 TRUSTED 或 NOT TRUSTED。如果参考约束指定为 TRUSTED，那么数据库管理器可信任数据将符合该约束。这是缺省选项。如果指定 NOT TRUSTED，那么数据库管理器知道大部分数据（但不是所有数据）将不符合该约束。在此示例中，缺省情况下该选项为 NOT ENFORCED TRUSTED，因为未指定该选项可信或不可信。

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 和 TRUSTED。对该表执行 SELECT 语句的结果如以下示例所示:

```
SELECT * FROM APPLICANTS
  WHERE GENDER = 'C';
```

```
APPLICANT  GENDER  AGE
---------  ------  ---

0 record(s) selected.
```

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，如以下示例所示:

```
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.
```

注: 如果一开始就对表 APPLICANTS 指定约束属性 NOT ENFORCED NOT TRUSTED 和 ENABLE QUERY OPTIMIZATION，那么发出第一个 SELECT 语句后将返回上述正确结果。

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 NOT ENFORCED TRUSTED 参考约束的最佳情形。如果应用程序事先已检查所有信息（例如，以上示例中的性别和年龄），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。而且，如果无法保证表中的数据将始终符合该约束，那么您可将约束设为 NOT ENFORCED 和 NOT TRUSTED。外键中的值与主键中的值之间不需要严格匹配时，可使用此类型的约束。也可仍在允许优化某些 SQL 查询的统计视图中使用此约束。

创建和修改约束

可以使用 ALTER TABLE 语句将约束添加至现有表。

关于此任务

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。与 NOT ENFORCED 选项一起，可进一步将约束指定为 TRUSTED 或 NOT TRUSTED。

限制: 对表定义参考约束后，只有先除去参考约束才能更改该表的列名。

要使用命令行对表指定参考约束，请对新表输入下列其中一个命令:

```
ALTER TABLE <name> <constraint attributes> NOT ENFORCED

ALTER TABLE <name> <constraint attributes> NOT ENFORCED TRUSTED

ALTER TABLE <name> <constraint attributes> NOT ENFORCED NOT TRUSTED
```

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。还可指定 NOT ENFORCED 约束是 TRUSTED 还是 NOT TRUSTED。
  - TRUSTED: 通知数据库管理器可信任数据符合该约束。这是缺省选项。仅当绝对知道数据符合该约束时，才能使用此选项。
  - NOT TRUSTED: 通知数据库管理器不信任该数据符合该约束。此选项适用于数据的大部分行符合该约束但未绝对知道符合该约束的情况。只能对引用完整性约束指定 NOT TRUSTED (SQLSTATE 42613)。

要修改此约束，必须先删除此约束，然后重新创建。

复用具有唯一键约束或主键约束的索引

如果使用 ALTER TABLE 命令对具有分区索引的分区表添加唯一键约束或主键约束，那么根据已存在的索引的不同，可能会更改索引以强制实施新约束，也可能会创建新索引。

当您运行 ALTER TABLE 语句对一个表添加或更改唯一键或主键时，系统将执行检查，以确定是否任何现有索引与您正在定义的唯一键或主键匹配（忽略 INCLUDE 列）。索引定义如果标识了同一组列，那么将被认为匹配，而与这些列的顺序或方向（例如 ASC/DESC）无关。

对于具有非唯一分区索引的分区表而言，如果正在更改的表的索引列未包括在构成分区键的那些列中，那么该索引不会被认为是匹配的索引。

如果该表确实有匹配的索引定义，并且该索引还不是唯一索引，那么它将更改为唯一索引并且将被标记成系统必需的索引。如果该表有多个匹配的现有索引，那么将选择现有的唯一索引。如果有多个匹配的唯一索引，或者有多个匹配的非唯一索引并且不存在匹配的唯一索引，那么将选择使用分区索引。否则，将任意地选择索引。

如果找不到匹配的索引，那么将为各个列自动创建唯一的双向索引。

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。

关于此任务

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。

关于此任务

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。

过程

- 要显式删除唯一约束，请使用 ALTER TABLE 语句的 DROP UNIQUE 子句。

  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。
- 要删除主键约束，请使用 ALTER TABLE 语句的 DROP PRIMARY KEY 子句。

  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
  ```
- 要删除（表）检查约束，请使用 ALTER TABLE 语句的 DROP CHECK 子句。

  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* 必须标识对表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。

- 要删除外键（引用）约束，请使用 ALTER TABLE 语句的 DROP CONSTRAINT 子句。

  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
  ```

  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。

示例

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
```

第 14 章 索引

*索引*是一组按一个或多个键的值以逻辑方式进行排序的指针。这些指针可以引用表中的行、MDC 或 ITC 表中的块以及 XML 存储器对象中的 XML 数据等等。

索引用于:

- 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。

对表添加数据时，该数据将被追加到底部（除非已对该表和正在添加的数据执行了其他操作）。不存在固有的数据顺序。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。

通常，在表中搜索数据时，您将查找列具有特定值的行。可以使用数据行中的列值来标识整个行。例如，职员编号可能唯一地定义特定的职员。另外，也可能需要多个列才能标识一行。例如，客户名称与电话号码的组合。索引中用于标识数据行的列称为*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。

并不是所有索引都指向表中的行。MDC 和 ITC 块索引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 和 ITC 块索引及 XML 索引都是由系统生成的索引。

示例

在第 376 页的图 41 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 376 页的图 41 显示了索引与表之间的关系。

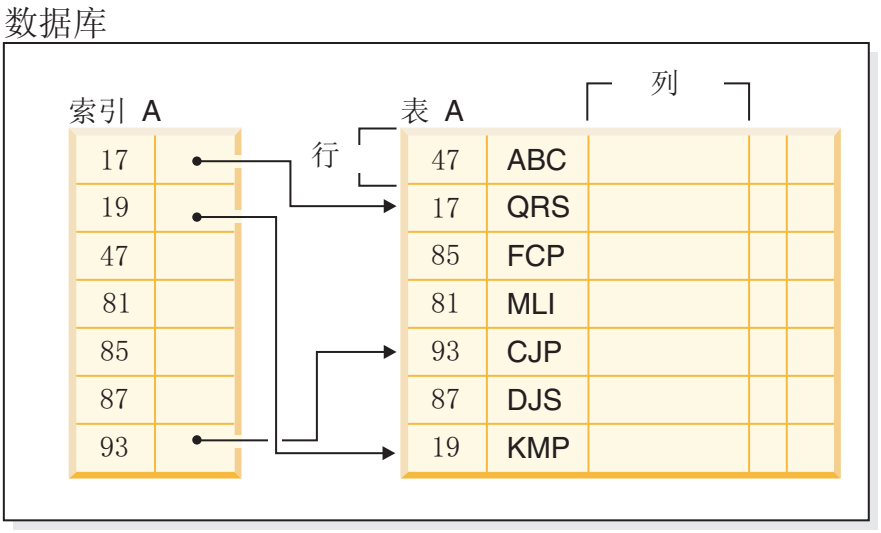


图 41. 索引与表之间的关系

图 42 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。

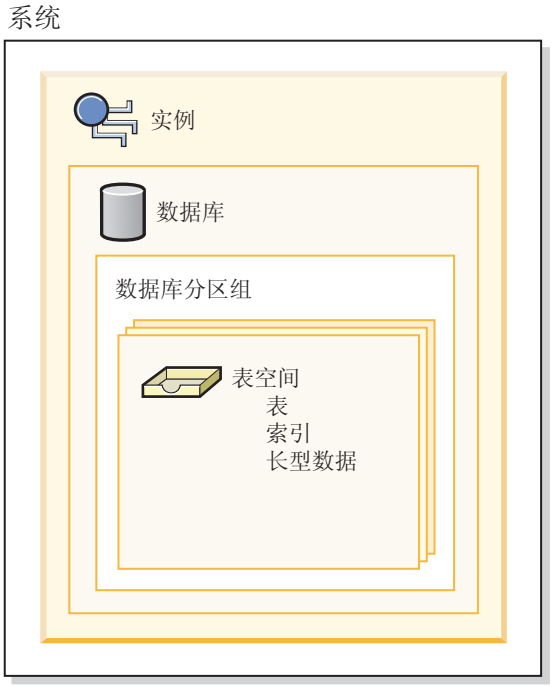


图 42. 所选择的数据库对象之间的关系

索引的类型

您可以创建不同类型的索引以便用于不同的用途。例如，唯一索引将对索引键强制实施唯一性约束；双向索引允许按正向和反向进行扫描；集群索引有助于提高按键顺序遍历表的查询的性能。

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。

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。

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顺序检测预取效率通常更高。

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，有助于使表保持集群。有关更多信息，请参阅 *SQL Reference*中的『ALTER TABLE 语句』以及*故障诊断和调整数据库性能*中的『减少对重组表和索引的需要』。

使用集群索引提高性能

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。）

分区索引和非分区索引

分区数据可以具有非分区索引（存在于一个数据库分区中的单一表空间中）、可以具有本身在一个数据库分区中的一个或多个表空间之间进行分区的索引或者同时具有这两种索引。对分区表执行滚入操作时（使用 ALTER TABLE 语句的 ATTACH PARTITION 子句将一个数据分区连接到另一个表），分区索引特别有用。

分区表的索引

分区表可以具有非分区索引（存在于一个数据库分区中的单一表空间中）、可以具有本身在一个数据库分区中的一个或多个表空间之间进行分区的索引或者同时具有这两种索引。

对分区表执行滚入操作时（使用 ALTER TABLE 语句的 ATTACH PARTITION 子句将一个数据分区连接到另一个表），分区索引具有优势。借助分区索引，您可以避免必须对非分区索引执行的索引维护工作。如果分区表使用非分区索引，那么您必须使用 SET INTEGRITY 语句来维护非分区索引（通过合并新连接的分区中的索引键）。此操作不仅耗时，而且可能需要大量的日志空间，这取决于正在滚入的行数。

某些类型的索引不能进行分区:

- 基于非分区数据的索引
- 基于空间数据的索引
- XML 列路径索引（由系统生成）

您必须始终作为非分区索引来创建这些索引。另外，分区唯一索引的索引键必须包含表分区键中所有的列，而无论它们是由用户生成还是由系统生成。后者将是系统创建索引以便对数据强制实施唯一约束或主约束的情况。

从 DB2 V9.7 FP1 开始，可以对分区表创建基于 XML 数据的分区索引或者非分区索引。缺省情况下将创建分区索引。基于 XML 数据的唯一索引始终为非分区索引。

分区表的非分区索引

*非分区索引*是单一的索引对象，它引用分区表中所有的行。非分区索引始终作为单一表空间中的独立索引对象进行创建，即使表数据分区跨多个表空间亦如此。

当您为分区表创建索引时，该索引在缺省情况下为*分区索引*，除非您创建下列其中一种类型的索引:

- 索引键未包含所有表分区列的唯一索引
- 空间索引

在这些情况下，您创建的索引为非分区索引。但是，在某些情况下，最好或有必要创建非分区索引，即使数据进行了分区也是如此。在这些情况下，请使用 CREATE INDEX 语句的 NOT PARTITIONED 子句对分区表创建非分区索引。缺省情况下，创建非分区索引时，它将存储在第一个可视的或者所连接的数据分区所在的表空间中。图 43 显示了单一索引 X1 的示例，此索引引用表中的所有分区。将在表的第一个可视分区所在的表空间中创建此索引。

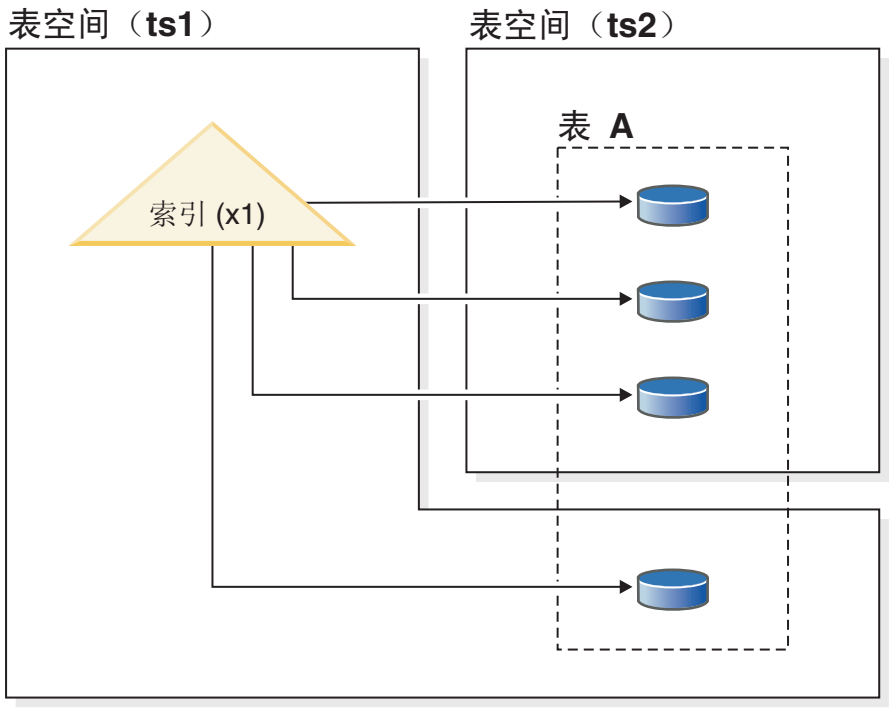


图 43. 分区表的非分区索引

第 380 页的图 44 显示了两个非分区索引的示例。在本例中，每个索引分区与数据分区在不同的表空间中。请再次注意每个索引引用表中所有分区的方式。

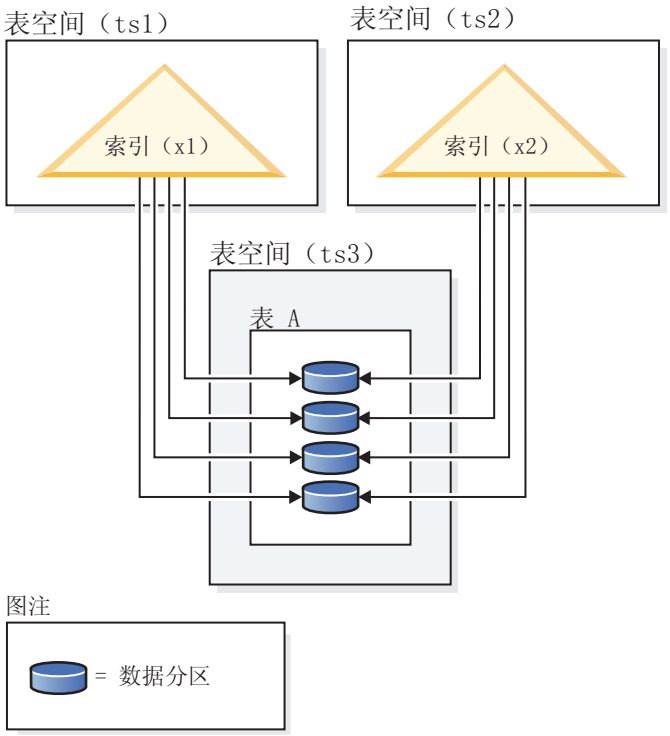


图 44. 分区表的非分区索引（索引在它们自己的表空间中）

在下列时候，您可以更改非分区索引的位置:

- 在创建表时，使用 CREATE TABLE 语句的 INDEX IN 子句进行更改
- 在创建索引时，使用 CREATE INDEX 语句的 IN 子句进行更改

第二种方法始终优先于第一种方法。

如果使用 ALTER TABLE 语句的 ATTACH PARTITION 子句将数据滚动到分区表，那么必须运行 SET INTEGRITY 语句以使相连分区的数据进入联机状态以供查询。如果索引是非分区索引，那么使相连分区的数据进入联机状态这一操作可能相当耗时，此操作将使用相当多的日志空间，这是因为 SET INTEGRITY 必须将新连接的分区中的数据插入到非分区索引中。

在拆离分区之后，不必运行 SET INTEGRITY。

分区表的分区索引

*分区索引*由一组*索引分区*构成，每个索引分区都包含单一数据分区的索引条目。每个索引分区都只包含对相应数据分区中的数据的引用。系统生成的索引和用户生成的索引都可以是分区索引。

在下列情况下，分区索引变得很有益:

- 您使用 ALTER TABLE 语句的 ATTACH PARTITION 或 DETACH PARTITION 子句将数据转入或转出分区表。对于非分区索引，您必须在新连接的分区中的数据可用之前运行的 SET INTEGRITY 语句可能相当耗时并且需要大量日志空间。当您连接一个使用分区索引的表分区时，仍然必须发出 SET INTEGRITY 语句以执行诸如验证范围和检查约束等任务。

提示: 如果在连接操作前可通过独立于数据服务器的应用程序逻辑完成数据完整性检查（包括范围验证和其他约束检查），那么新连接的数据可更快供用户使用。可通过使用 SET INTEGRITY...ALL IMMEDIATE UNCHECKED 语句来跳过范围和约束违例检查以优化数据转入进程。在这种情况下，该表会脱离 SET INTEGRITY 暂挂状态，并且新数据可供应用程序立即使用，只要目标表没有非分区用户索引。

如果源表的索引与目标表的索引分区，那么 SET INTEGRITY 处理不会产生与索引维护相关的性能和日志记录处理；可以更快的速度访问新转入的数据（与使用非分区索引相比）。有关索引匹配的更多信息，请参阅*分区和集群指南*中的『在 ATTACH PARTITION 期间使源表索引与目标表分区索引相匹配的条件』。

- 您正在对需要执行索引重组的特定分区中的数据执行维护。例如，以一个具有 12 个分区（每个分区都对应于一年中的特定月份）的表为例。您可能要更新或删除特定于一年中的某个月的许多行。此操作可能导致索引分段，这可能需要您执行索引重组。借助分区索引，您可以仅重组与进行更改的数据分区相对应的索引分区，与重组整个非分区索引相比，这会节省大量时间。

某些类型的索引不能进行分区:

- 基于非分区数据的索引
- 基于空间数据的索引
- XML 列路径索引（由系统生成）

您必须始终作为非分区索引来创建这些索引。另外，分区唯一索引的索引键必须包含表分区键中所有的列，而无论它们是由用户生成还是由系统生成。后者将是系统创建索引以便对数据强制实施唯一约束或主约束的情况。

第 382 页的图 45 显示了分区索引的示例。

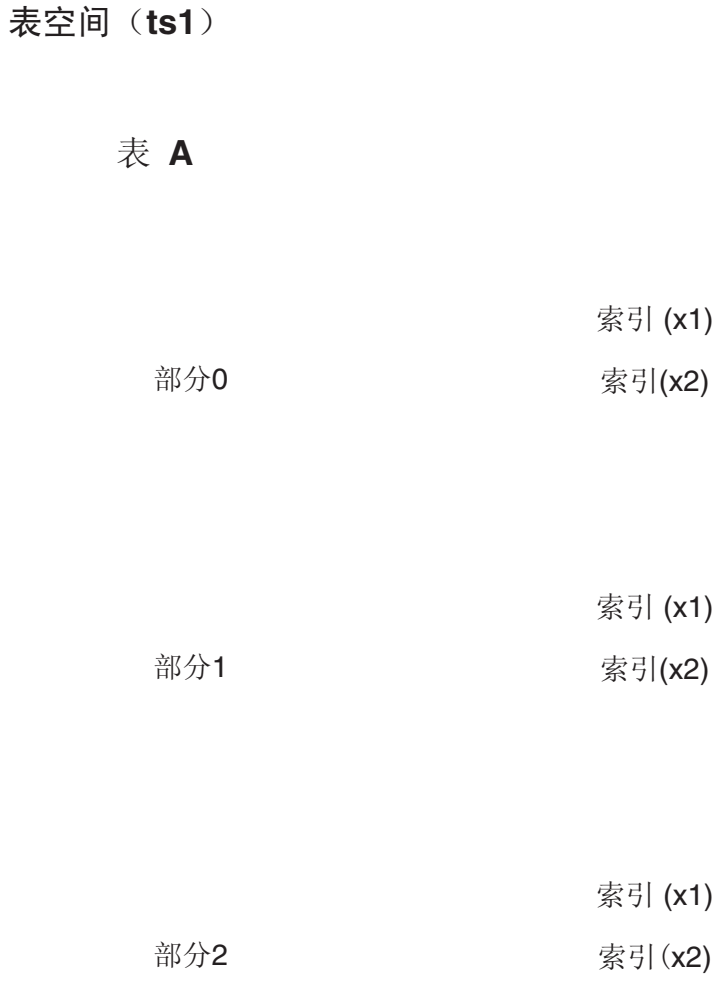


图 45. 与表的数据分区共享表空间的分区索引

在此示例中，表 A 的所有数据分区与表 A 的所有索引分区在单一表空间中。索引分区只引用与其相关联的数据分区中的行。（请将分区索引与索引引用*所有*数据分区中*所有*的行的非分区索引进行对比。）而且，数据分区的索引分区在同一个索引对象中。索引和索引分区的这种特定安排是使用类似于以下的语句建立的:

```
CREATE TABLE A (columns) in ts1
      PARTITION BY RANGE (column expression)
      (PARTITION PART0 STARTING FROM constant ENDING constant,
       PARTITION PART1 STARTING FROM constant ENDING constant,
       PARTITION PART2 STARTING FROM constant ENDING constant,

   CREATE INDEX x1 ON A (...) PARTITIONED;
   CREATE INDEX x2 ON A (...) PARTITIONED;
```

第 383 页的图 46 显示了分区索引的另一个示例。

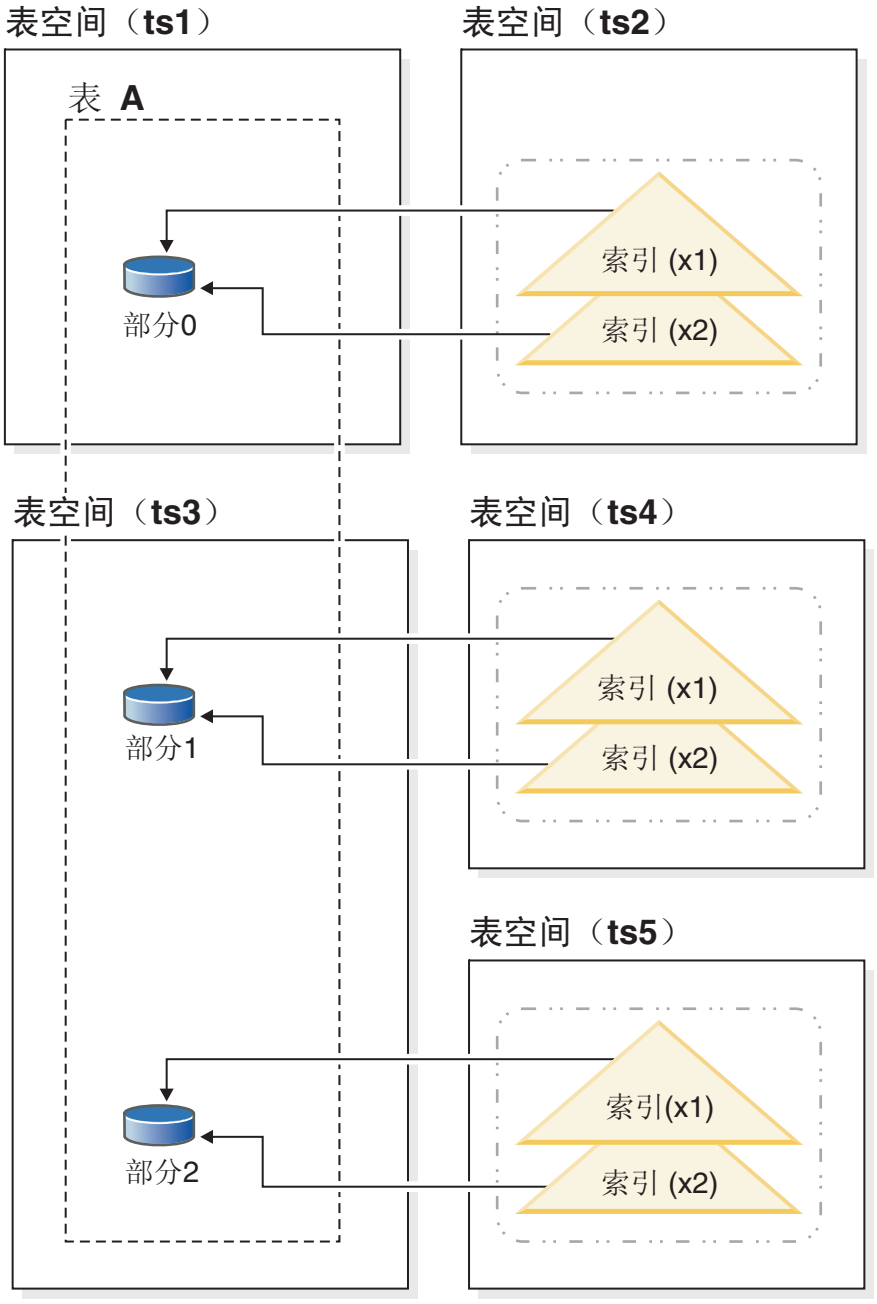


图 46. 数据分区与索引分区在不同表空间中的分区索引。

在此示例中，表 A 的数据分区分布在两个表空间 TS1 和 TS3 中。索引分区也在不同的表空间中。索引分区只引用与其相关联的数据分区中的行。索引和索引分区的这种特定安排是使用类似于以下的语句建立的:

```
CREATE TABLE A (columns)
      PARTITION BY RANGE (column expression)
      (PARTITION PART0 STARTING FROM constant ENDING constant IN ts1 INDEX IN ts2,
       PARTITION PART1 STARTING FROM constant ENDING constant IN ts3 INDEX IN ts4,
       PARTITION PART2 STARTING FROM constant ENDING constant IN ts3,INDEX IN ts5)

   CREATE INDEX x1 ON A (...);
   CREATE INDEX x2 ON A (...);
```

在这种情况下，CREATE INDEX 语句中省略了 PARTITIONED 子句；索引仍作为分区索引创建，因为此设置是分区表的缺省设置。

图 47 图提供了同时带有非分区索引和分区索引的分区表的示例。

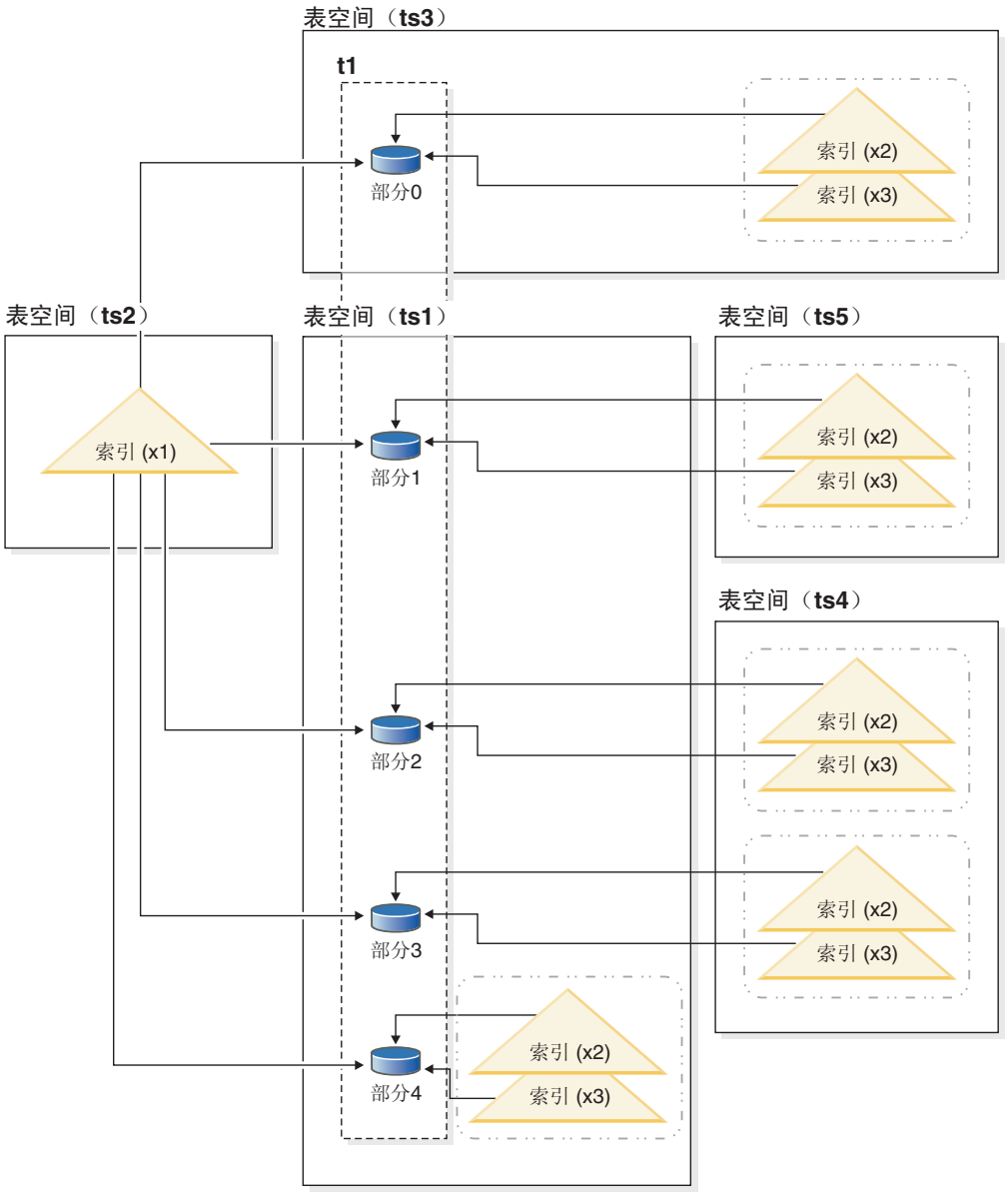


图 47. 分区表的非分区索引与分区索引的组合

在此图中，索引 X1 是非分区索引，它引用了表 T1 的所有分区。索引 X2 和 X3 是分区索引，它们驻留在各种表空间中。索引和索引分区的这种特定安排是使用类似于以下的语句建立的:

```
CREATE TABLE t1 (columns) in ts1 INDEX IN ts2 1
   PARTITION BY RANGE (column expression)
   (PARTITION PART0 STARTING FROM constant ENDING constant  IN ts3, 2
    PARTITION PART1 STARTING FROM constant ENDING constant  INDEX IN ts5,
    PARTITION PART2 STARTING FROM constant ENDING constant  INDEX IN ts4,
    PARTITION PART3 STARTING FROM constant ENDING constant  INDEX IN ts4,
    PARTITION PART4 STARTING FROM constant ENDING constant)
```

```
CREATE INDEX x1 ON t1 (...) NOT PARTITIONED;
CREATE INDEX x2 ON t1 (...) PARTITIONED;
CREATE INDEX x3 ON t1 (...) PARTITIONED;
```

请注意:

- 非分区索引 X1 存储在表空间 TS2 中，因为此位置是对表 T1 的非分区索引指定的缺省位置（请参阅 **1** ）。
- 数据分区 0 (Part0) 的索引分区存储在表空间 TS3 中，这是因为，索引分区的缺省位置与它所引用的数据分区的位置相同（参见 **2** ）。
- Part4 存储在 TS1 中，这是表 T1 中各个数据分区的缺省表空间（参见 **1** ）；此数据分区的索引分区还驻留在 TS1 中，这也是因为索引分区的缺省位置与它所引用的数据分区的位置相同。

要点: 与非分区索引不同，对于分区索引而言，不能使用 CREATE INDEX 语句的 INDEX IN 子句来指定用于存储索引分区的表空间。更改索引分区的缺省存储位置的唯一方法是，在创建表时使用 CREATE TABLE 语句的*分区级* INDEX IN 子句来指定位置。表级别 INDEX IN 子句不影响索引分区的存储位置。

您通过在 CREATE INDEX 语句中指定 PARTITIONED 选项为分区表创建分区索引。例如，对于在将 sales_date 用作表分区键的情况下进行分区的名为 SALES 的表，要创建分区索引，可以使用类似于以下的语句:

```
CREATE INDEX partIDbydate on SALES (sales_date, partID) PARTITIONED
```

如果您正在创建分区唯一索引，那么必须将表分区列包括在索引键列中。因此，对于上一个示例，如果您尝试使用以下语句来创建分区索引:

```
CREATE UNIQUE INDEX uPartID on SALES (partID) PARTITIONED
```

那么此语句将失败，这是因为未将构成表分区键的列 sales_date 包括在索引键中。

如果对分区表创建索引时省略了 PARTITIONED 关键字，那么数据库管理器在缺省情况下将创建分区索引，除非符合以下条件:

- 您正在创建唯一索引，并且索引键未包含所有表分区键。
- 您正在创建本主题开头描述的其中一种无法被创建成分区索引的索引。

在其中任一情况下，该索引将被创建为非分区索引。

尽管创建定义与现有非分区索引匹配的非分区索引返回 SQL0605W，但分区索引可以与具有类似定义的非分区索引共存。此共存是为了更便于采用分区索引。

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。

要点: 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。出现此负面影响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，只有在能够明显提高整体性能时，才应创建索引。

在创建索引时，还应考虑表结构和最常对这些表执行的查询的类型。例如，频繁发出的查询的 WHERE 子句中出现的列很适合作为索引。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。

在创建索引时，请记住，它们也可以进行压缩。以后，您可以使用 ALTER INDEX 语句来修改索引，从而启用或禁用压缩功能。

要除去或删除索引，可以使用 DROP INDEX 命令。删除索引的要求与插入索引相反；即，除去索引条目（或者将其标记为“已删除”）。

设计索引时的准则和注意事项

- 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。例如，如果查询包含 `ORDER BY col1,col2` 子句，那么可以使用对 `(col1,col2)` 创建的索引，但对 `(col2,col1)` 创建的索引没什么帮助。同样，如果查询指定了条件，例如 `where col1 >= 50 and col1 <= 100` 或 `where col1=74`，那么对 `(col1)` 或 `(col1,col2)` 创建的索引将起作用，但基于 `(col2,col1)` 的索引的作用不大。

  注：每当有可能时，您都应该按最独特到最不独特的顺序对索引键中的列进行排序。此排序提供最佳性能。
- 可以对特定的表定义任意数目的索引（最大数目为 32767），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。此索引将用作主键索引。
- 如果您有*分区表*，缺省情况下，您创建的任何索引都是*分区索引*，除非您创建不包括分区键的唯一索引。您还可以将索引作为*非分区索引*来创建。

  从 DB2 V9.7 FP1 开始，可以对分区表创建基于 XML 数据的分区索引或者非分区索引。缺省情况下将创建分区索引。

  对分区表执行滚入操作时（使用 ALTER TABLE 语句的 ATTACH PARTITION 子句将一个数据分区连接到另一个表），分区索引具有优势。借助分区索引，您可以避免必须对非分区索引执行的索引维护工作。如果分区表使用非分区索引，那么您

必须使用 SET INTEGRITY 语句来维护非分区索引（通过合并新连接的分区中的索引键）。此操作不仅耗时，而且可能需要大量的日志空间，这取决于正在滚入的行数。

- 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 或 ITC 表时还会将创建系统生成的块索引。
  - XML 列将始终导致创建由系统生成的索引（其中包括列路径索引和区域索引）。
  - 对外键约束列创建索引通常会有好处。
  - 是否使用 COMPRESS 选项对索引进行压缩。

  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（包括所有组件）为 *IndexPageSize* ÷ 4。表允许的最大索引数为 32,767。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。
- 在数据库升级期间，不会对现有索引进行压缩。如果对表启用数据行压缩功能，那么除非在 CREATE INDEX 语句中指定 COMPRESS NO 选项，否则将对升级后创建的新索引进行压缩。

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 设计顾问程序是可以帮助您显著提高工作负载性能的工具。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，您必须了解它们的空间需求。对于处于压缩状态的索引而言，可以使用根据本主题中的公式派生的估算值作为上限，但是，实际大小很可能小得多。

处于未压缩状态的索引的空间需求

对于每个处于未压缩状态的索引，可以按如下公式估算所需的空间:

(*平均索引键大小* + *索引键开销*) × *行数* × 2

其中:

- *平均索引键大小*是索引键中每一列的字节数。估算 VARCHAR 和 VARGRAPHIC 列的平均列大小时，请使用当前数据大小的平均值加上两个字节。
- *索引键开销*取决于对其创建索引的表的类型:

表 77. 不同的表的索引键开销

表空间的类型	表类型	索引类型	索引键开销
任意	任意	XML 路径或区域	11 字节
常规	非分区	任意	9 字节
	分区	分区	9
		非分区	11
大型	分区	分区	11
		非分区	13

- *number of rows* 是表中的行数或给定数据分区中的行数。通过在此计算中使用整个表中的行数，您可以估算索引的大小（对于非分区索引）或者所有索引分区的总大小（对于分区索引）。通过使用数据分区中的行数，可以估算索引分区的大小。
- 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) 表或插入时间集群 (ITC) 表创建的块索引，“行数”将被替换为“块数”。

XML 索引的空间需求

对于 XML 列的每个索引，可以按如下公式估算所需的空间:

(*平均索引键* + *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:

  14 + *可变开销* + *sql-data-type 的字节数*

  其中:

  – 14 表示固定开销的字节数
  – *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 的字节数*与 SQL 遵循相同的规则。

- *建立索引的节点数*是指，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。*建立索引的节点数*可以是分区或整个表中的节点数。

创建索引时的临时空间需求

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算:

(*平均索引键大小* + *索引键开销*) × *行数* × 3.2

对于那些每行可以有多个索引键的索引（例如空间索引、基于 XML 列的索引以及内部 XML 区域索引），可以按如下方式来估算所需的临时空间量:

(*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。*行数*或*建立索引的节点数*是整个表中的数目或者给定数据分区中的数目。

注: 对于非唯一索引，在任何给定的叶子节点上，将只存储给定重复键条目的一个副本。对于 LARGE 表空间中的表的索引，重复键的大小是 9（对于非分区索引）或 7（对于分区索引以及非分区表的索引）。对于 REGULAR 表空间中的表的索引，这些值是 7（对于非分区索引）或 5（对于分区索引以及非分区表的索引）。这些规则的例外情况只有 XML 路径和 XML 区域索引，对于这些索引，重复键的大小始终是 7。上面提供的估算值假定没有重复项。因此，以上公式可能会过高地估算存储索引所需的空间量。

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。可以按如下公式估算所需的临时空间:

平均索引键大小 × *建立索引的节点数* × 1.2

估算每个叶子页的键数

可以使用下面这两个公式来估算每个索引叶子页的键数（第二个公式的估算更为准确）。这些估算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量是页大小的三倍。对于 DMS 表空间，最小值是一个扩展数据块。

1. 每个叶子页的平均键数的粗略估算是:

   ((.9 * (*U* - (*M*×2))) × (*D* + 1)) ÷ (*K* + 7 + (*Ds* × *D*))

   其中:

   - *U*（一页中的可用空间量）大约等于页大小减 100。例如，页大小为 4096 时，U 将是 3996。
   - *M* = *U* ÷ (9 + *最小键大小*)
   - *Ds* = *重复键大小*（请参阅“创建索引时的临时空间需求”下的说明。）
   - *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，表示每个可变长度键部分的长度。

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。

*最小键大小*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 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 的字节数*值与 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。

2. 每个叶子页的平均键数的更准确估算是:

   叶子页数 = x / (叶子页中的平均键数)

   其中，*x* 是表或分区中的总行数。

   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:

   $(L + 2L/(\text{叶子页中的平均键数})) \times \text{页大小}$

   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:

   $((.9 \times (U - (M \times 2))) \times (D + 1)) \div (K + 13 + (9 * D))$

   其中:

   - *U*（一页中的可用空间量）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（这将比在叶子页上的小很多，您可能想将该值设为 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 的值大于 .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）。

注: 以上计算适用于单一非分区索引或者分区索引的单一索引分区。

总页数是:

T = (*L* + *P* + 2) × 1.0002

附加的 0.02%（1.0002）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

索引压缩

可以压缩索引（包括已声明的临时表或者已创建的临时表的索引），以便降低存储器开销。对于大型 OLTP 和数据仓库环境而言，此功能特别有用。

缺省情况下，索引压缩功能对于已压缩的表处于启用状态，对于未压缩的表处于禁用状态。您可以使用 CREATE INDEX 语句的 **COMPRESS YES** 选项来更改此缺省行为。处理现有索引时，使用 ALTER INDEX 语句来启用或禁用索引压缩；然后必须执行索引重组以重建索引。

限制: 下列类型的索引不支持索引压缩功能:

- 块索引
- XML 路径索引。

另外:

- 不能对索引规范进行压缩。
- 无法使用 ALTER INDEX 命令对临时表的索引更改压缩属性。

启用索引压缩功能后，将根据数据库管理器所选择的压缩算法对索引页在磁盘上和内存中的格式进行修改，以便最大程度地减少存储空间耗用量。根据所创建索引类型以及索引所包含数据的不同，实现的压缩程度也会有所变化。例如，通过存储重复键的记录标识 (RID) 的缩写格式，数据库管理器可以对包含大量重复键的索引进行压缩。在索引键前缀的公共程度很高的索引中，数据库管理器可以根据索引键前缀的相似性来进行压缩。

存在一些与压缩相关联的限制和权衡。如果索引未共享公共索引列值或者不完整的公共前缀，那么索引压缩在减少耗用存储量方面的优势可以忽略不计。并且，尽管时间

戳记列的唯一索引可能由于同一个叶子页包含年、月、日、小时、分钟甚至秒的公共值而具有非常高的压缩能力，但检查是否存在公共前缀也会导致性能下降。

如果您相信压缩不会对您所处的特定情况带来好处，那么可以在不进行压缩的情况下重新创建索引，也可以更改索引并接着执行索引重组以禁用索引压缩功能。

在考虑使用索引压缩功能时，您应该记住下列事项:

- 如果使用 CREATE TABLE 或 ALTER TABLE 命令的 **COMPRESS YES** 选项启用行压缩功能，那么在缺省情况下，将对此后为该表创建的所有支持压缩的索引启用压缩功能，除非通过 CREATE INDEX 或 ALTER INDEX 命令显式禁用该功能。同样，如果通过 CREATE TABLE 或 ALTER TABLE 命令禁用行压缩功能，那么将对此后为该表创建的所有索引禁用索引压缩功能，除非通过 CREATE INDEX 或 ALTER INDEX 命令显式启用该功能。
- 如果使用 ALTER INDEX 命令来启用索引压缩功能，那么直到执行索引重组之后才会进行压缩。同样，如果禁用压缩功能，那么在您执行索引重组之前，该索引将保持处于已压缩状态。
- 在数据库迁移期间，不会对任何迁移后的索引启用压缩功能。如果要启用压缩功能，那么必须使用 ALTER INDEX 命令，然后执行索引重组。
- 索引压缩或解压所需进行的处理可能会导致 CPU 使用率略微提高。如果这种情况不可接受，那么您可以对新索引或现有索引禁用索引压缩功能。

示例

示例 1: 检查索引是否处于压缩状态。

下面这两个语句将创建支持行压缩功能的新表 T1 并对 T1 创建索引 I1。

```
CREATE TABLE T1 (C1 INT, C2 INT, C3 INT) COMPRESS YES
CREATE INDEX I1 ON T1(C1)
```

缺省情况下，T1 的索引处于压缩状态。可以使用目录表或管理表函数来检查索引 T1 的*压缩属性*，此属性将指示是否已启用压缩功能:

```
SELECT COMPRESSION FROM SYSCAT.INDEXES WHERE TABNAME='T1'

COMPRESSION
-----------
Y

  1 record(s) selected.
```

示例 2: 确定已压缩的索引是否要求进行重组。

要确定已压缩的索引是否要求进行重组，请使用 **REORGCHK** 命令。第 393 页的图 48 显示如何对名为 T1 的表运行此命令:

```
REORGCHK ON TABLE SCHEMA1.T1

Doing RUNSTATS ....


Table statistics:

F1: 100 * OVERFLOW / CARD < 5
F2: 100 * (Effective Space Utilization of Data Pages) > 70
F3: 100 * (Required Pages / Total Pages) > 80

SCHEMA.NAME                     CARD     OV     NP     FP ACTBLK    TSIZE  F1  F2  F3 REORG
-------------------------------------------------------------------------
Table: SCHEMA1.T1
                    879      0     14     14      -    51861   0 100 100 ---
-------------------------------------------------------------------------

Index statistics:

F4: CLUSTERRATIO or normalized CLUSTERFACTOR > 80
F5: 100 * (Space used on leaf pages / Space available on non-empty leaf pages) >
MIN(50, (100 - PCTFREE))
F6: (100 - PCTFREE) * (Amount of space available in an index with one
less level / Amount of space required for all keys) < 100
F7: 100 * (Number of pseudo-deleted RIDs / Total number of RIDs) < 20
F8: 100 * (Number of pseudo-empty leaf pages / Total number of leaf pages) < 20

SCHEMA.NAME       INDCARD LEAF ELEAF LVLS NDEL KEYS LEAF_RECSIZE NLEAF_RECSIZE...
-------------------------------------------------------------------------------...
Table: SCHEMA1.T1
Index: SCHEMA1.I1
                     879   15     0    2    0  682           20            20...
-------------------------------------------------------------------------------...

...LEAF_PAGE_OVERHEAD NLEAF_PAGE_OVERHEAD PCT_PAGES_SAVED  F4  F5  F6  F7  F8 REORG
...---------------------------------------------------------------------------------


...               596                 596             28  56  31   -   0   0 -----
...---------------------------------------------------------------------------------
```

图 48. **REORGCHK** *命令的输出*

已对 **REORGCHK** 命令的输出进行格式化以适应页面。

示例 3: 确定索引压缩功能有可能节省的空间量。

要获取有关如何计算索引压缩功能有可能节省的空间量的示例，请参阅 ADMIN_GET_INDEX_COMPRESS_INFO 表函数的文档。

创建索引

可因为许多原因创建索引，包括以下原因：为允许更高效地运行查询；为按列中的值的升序或降序顺序对表行排序；为强制实施诸如索引键的唯一性之类的约束。可使用 CREATE INDEX 语句、DB2 设计顾问程序或 **db2advis** 设计顾问程序命令来创建索引。

开始之前

在 Solaris 平台上，要创建使用原始设备的索引，必须安装用于 Solaris 9 的补丁 122300-11 或者用于 Solaris 10 的补丁 125100-07。如果没有此补丁，并且使用了原始设备，那么 CREATE INDEX 语句将被挂起。

关于此任务

本任务假定您正在对非分区表创建索引。

过程

要从命令行创建索引，请使用 CREATE INDEX 语句。
例如:

```
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。

下一步做什么

删除或更新行时，索引键会标记为“已删除”，并且直到落实删除或更新操作后的某个时间执行清除时，才会实际从页中删除。这些键被称为伪删除键。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。可通过在 **REORG INDEXES** 命令中使用 **CLEANUP ONLY ALL** 参数来显式触发伪删除键的清除。

对分区表创建非分区索引

对分区表创建*非分区索引*时，您将创建引用表中所有的行的单一索引对象。非分区索引始终在单一表空间中进行创建，即使表数据分区跨多个表空间亦如此。

开始之前

本任务假定分区表已创建完毕。

过程

1. 通过使用 NOT PARTITIONED 子句，为表构造 CREATE INDEX 语句。 例如:

   ```
   CREATE INDEX indexName ON tableName(column) NOT PARTITIONED
   ```

2. 从受支持的 DB2 接口中执行此 CREATE INDEX 语句。

示例

示例 1: 在数据分区所在的表空间中创建非分区索引。

假定 SALES 表的定义如下所示:

```
CREATE TABLE sales(store_num INT, sales_date DATE, total_sales DECIMAL (6,2)) IN ts1
   PARTITION BY RANGE(store_num)
   (STARTING FROM (1) ENDING AT (100),
    STARTING FROM (101) ENDING AT (150),
    STARTING FROM (151) ENDING AT (200))
```

SALES 表的三个分区都存储在表空间 TS1 中。缺省情况下，任何为此表创建的索引也将存储在 TS1 中，其原因在于，这是对此表指定的表空间。要根据 STORE_NUM 列来创建非分区索引 STORENUM，请使用以下语句:

```
CREATE INDEX StoreNum ON sales(store_num) NOT PARTITIONED
```

注意，必须指定 NOT PARTITIONED 子句，否则该索引将被创建成分区索引，这是分区表的缺省情况。

示例 2: 在除缺省表空间以外的表空间中创建非分区索引

假定已按如下方式来定义名为 PARTS 的表:

```
    CREATE TABLE parts(part_number INT, manufacturer CHAR, description CLOB, price
DECIMAL (4,2)) IN ts1 INDEX in ts2
       PARTITION BY RANGE (part_number)
       (STARTING FROM (1) ENDING AT (10) IN ts3,
        STARTING FROM (11) ENDING AT (20) INDEX IN ts1,
        STARTING FROM (21) ENDING AT (30) IN ts2 INDEX IN ts4);
```

PARTS 表包含 3 个分区: 第一个分区在表空间 TS3 中，第二个在 TS2 中，第三个在 TS3 中。如果发出以下语句，那么将创建按制造商名称的降序对行进行排序的非分区索引:

```
CREATE INDEX manufct on parts(manufacturer DESC) NOT PARTITIONED IN TS3;
```

此索引将在表空间 TS3 中进行创建; CREATE TABLE 语句的 INDEX IN 子句将被 CREATE INDEX 语句的 IN *tablespace* 子句覆盖。由于表 PARTS 已分区，因此必须在 CREATE INDEX 语句中指定 NOT PARTITIONED 子句才能创建非分区索引。

创建分区索引

为分区表创建*分区索引*时，每个数据分区都将在它自己的索引分区中建立索引。缺省情况下，索引分区与对应的数据分区存储在同一个表空间中。索引中的数据根据表的分布键进行分布。

开始之前

本任务假定分区表已创建完毕。

关于此任务

限制

某些类型的索引不能进行分区:

- 基于非分区数据的索引
- 基于空间数据的索引
- XML 列路径索引（由系统生成）

您必须始终作为非分区索引来创建这些索引。另外，分区唯一索引的索引键必须包含表分区键中所有的列，而无论它们是由用户生成还是由系统生成。后者将是系统创建索引以便对数据强制实施唯一约束或主约束的情况。

并且，创建分区索引时，不支持使用 CREATE INDEX 语句的 IN 子句。缺省情况下，索引分区将在对应数据分区所在的表空间中进行创建。要指定使用另一个表空间来存

储索引分区，您必须使用 CREATE TABLE 语句的分区级 INDEX IN 子句逐个分区地为索引指定表分区。如果省略此子句，那么索引分区将驻留在对应数据分区所在的表空间中。

过程

1. 通过使用 PARTITIONED 子句，为表构造 CREATE INDEX 语句。
2. 从受支持的 DB2 接口中执行此 CREATE INDEX 语句。

示例

注: 下列示例仅供参考，它们并不是创建分区表或分区索引方面的最佳实践。
示例 1: 在数据分区所在的表空间中创建分区索引。

在此示例中，假定 SALES 表的定义如下所示:

```
CREATE TABLE sales(store_num INT, sales_date DATE, total_sales DECIMAL (6,2)) IN ts1
 PARTITION BY RANGE(store_num)
   (STARTING FROM (1) ENDING AT (100),
    STARTING FROM (101) ENDING AT (150),
    STARTING FROM (151) ENDING AT (200))
```

在此例中，表 SALES 的三个分区存储在表空间 ts1 中。为这个表创建的任何分区索引也将存储在 ts1 中，其原因在于，这是用于存储此表的每个分区的表空间。要根据存储器编号来创建分区索引，请使用以下语句:

```
CREATE INDEX StoreNum ON sales(store_num) PARTITIONED
```

示例 2: 为所有索引分区选择另一个位置。

在此示例中，假定 EMPLOYEE 表的定义如下所示:

```
CREATE TABLE employee(employee_number INT, employee_name CHAR,
             job_code INT, city CHAR, salary DECIMAL (6,2))
             IN ts1 INDEX IN ts2
   PARTITION BY RANGE (job_code)
   (STARTING FROM (1) ENDING AT (10) INDEX IN ts2,
    STARTING FROM (11) ENDING AT (20) INDEX IN ts2,
    STARTING FROM (21) ENDING AT (30) INDEX IN ts2)
```

要根据作业代码来创建分区索引，请使用以下语句:

```
CREATE INDEX JobCode ON employee(job_code) PARTITIONED
```

在此示例中，EMPLOYEE 表的分区存储在表空间 ts1 中，但是，所有索引分区都将存储在 ts2 中。

示例 3: 在多个分区中创建索引。

假定已按如下方式来定义名为 PARTS 的表:

```
CREATE TABLE parts(part_number INT, manufacturer CHAR, description CLOB,
price DECIMAL (4,2)) IN ts1 INDEX in ts2
   PARTITION BY RANGE (part_number)
   (STARTING FROM (1) ENDING AT (10) IN ts3,
    STARTING FROM (11) ENDING AT (20) INDEX IN ts1,
    STARTING FROM (21) ENDING AT (30) IN ts2 INDEX IN ts4);
```

在本例中，PARTS 表包含 3 个分区: 第一个分区在表空间 ts3 中，第二个分区在 ts1 中，第三个分区在 ts2 中。发出了下列语句:

```
CREATE INDEX partNoasc ON parts(part_number ASC)  PARTITIONED
CREATE INDEX manufct on parts(manufacturer DESC) NOT PARTITIONED IN TS3;
```

于是，创建了两个索引。第一个索引是分区索引，用于按部件号的升序对行进行排序。第一个索引分区将在表空间 ts3 中创建，第二个索引分区在 ts1 中创建，第三个索引分区在 ts4 中创建。第二个索引是非分区索引，用于按制造商名称的降序对行进行排序。此索引将在 ts3 中创建。注意，对于非分区索引而言，允许在 CREATE INDEX 语句中使用 IN 子句。并且，在本例中，由于表 PARTS 是分区表，因此，要创建非分区索引，必须在 CREATE INDEX 语句中指定 NOT PARTITIONED 子句。

修改索引

如果要修改索引，那么除了使用 ALTER INDEX 语句来启用或禁用索引压缩功能以外，必须先删除该索引，然后再次创建该索引。

示例

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。但是，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。

关于此任务

重命名索引时，源索引不能是系统生成的索引。

过程

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。它不能是已声明临时表或已创建临时表的索引的名称。模式名不能是 SYSIBM、SYSCAT、SYSFUN 或 SYSSTAT。
target_index_name 指定索引的新名称（不带模式名）。源对象的模式名用于限定对象的新名称。限定名不得标识数据库中已存在的索引。

结果

如果 RENAME 语句成功，那么系统目录表会更新以反映新索引名。

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。

关于此任务

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

要删除索引，请使用 DROP 语句。

关于此任务

除了更改索引的 COMPRESSION 属性以外，您无法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包无效。

限制

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。

过程

要使用命令行删除索引，请输入：

```
DROP INDEX index_name
```

结果

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 15 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语句。仅当搜索条件求值为 true 时，才执行这些语句。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语句的触发器，这些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操作应用引

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，会使用冲突解决机制（如临时表）来解决访问冲突。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。如果必须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。

关于此任务

*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。在这种情况下，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。

关于此任务

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。

关于此任务

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。

示例

下图说明了前触发器和后触发器的执行模型:

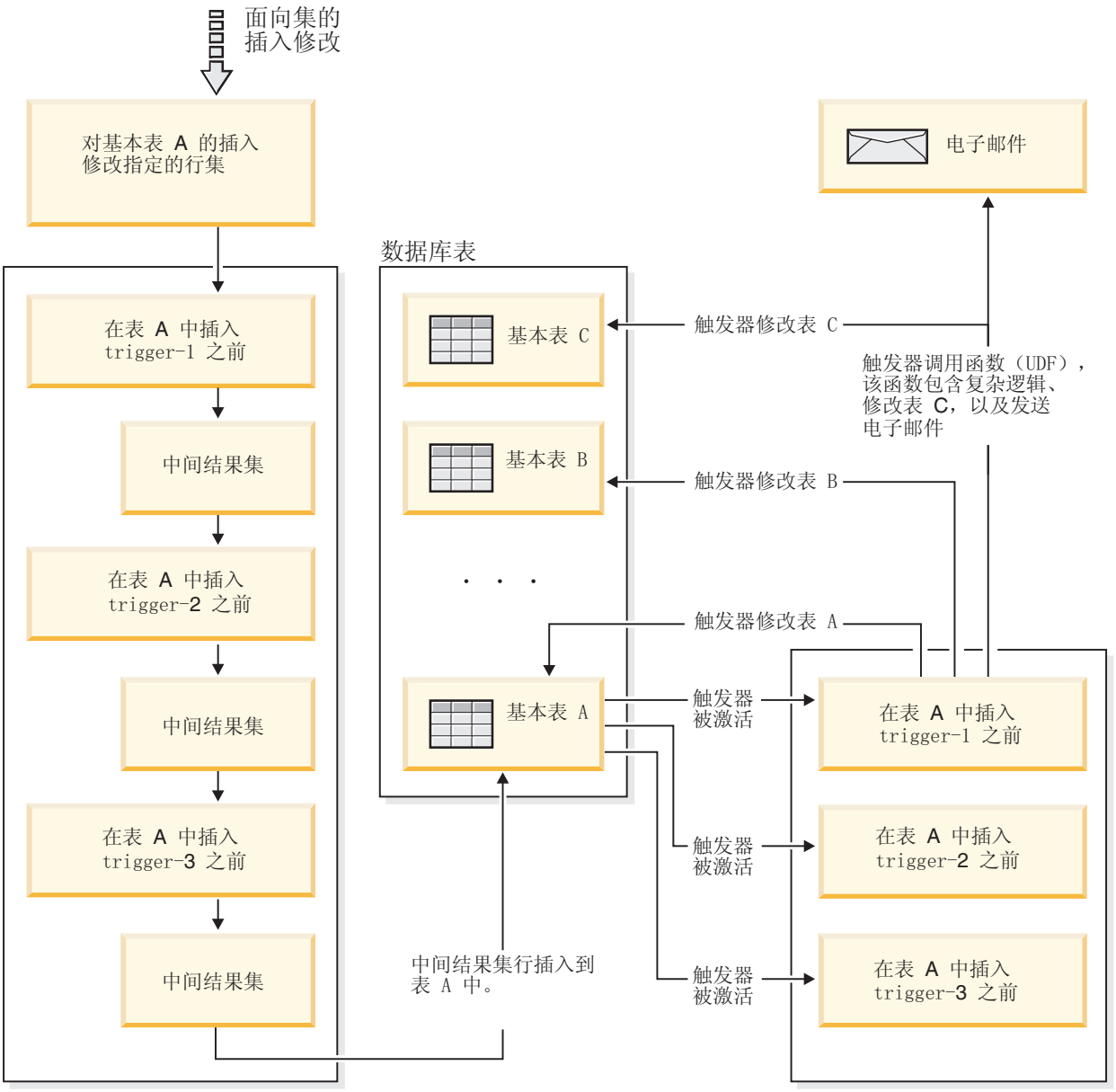


图 49. 触发器执行模型

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。

关于此任务

*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，那么总是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。

示例

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。在此示例中，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。

关于此任务

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。

示例

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
```

下一步做什么

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:

UPDATE

更新触发器可以同时引用 OLD 和 NEW 转换变量。

INSERT

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。

DELETE

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。

关于此任务

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。

示例

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。

下一步做什么

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:

UPDATE

更新触发器可以引用 OLD_TABLE 和 NEW_TABLE 转换表。

INSERT

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。

DELETE

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。

关于此任务

使用触发器来执行以下操作:

- 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: 定义了某个触发器后，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: 如果业务策略更改，只需更改触发器而不必更改每个应用程序。

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，以下语法可将其复原:

```
--#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。

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。

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。

过程

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
```

示例

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;
```

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。

开始之前

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。

关于此任务

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 365 页的图 40 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。

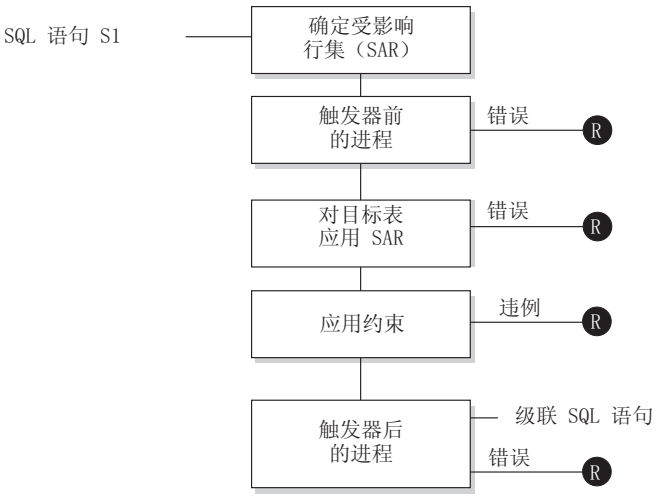


图 50. 处理包含相关触发器和约束的语句

第 365 页的图 40 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 365 页的图 40 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

假定您所在的公司具有的策略，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，您必须防止执行某些 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 16 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，必须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，那么您必须熟悉可用的选项和参数。

关于此任务

在设计序列之前，请参阅第 421 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）

即使在生成序列后，这些值中的许多值也可以更改。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额外处理，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，那么必须将高速缓存值设为 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 Reference*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停用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下降一些，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更改它，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。

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。

关于此任务

在可使用 NEXT VALUE 或 PREVIOUS VALUE 表达式的位置有几个限制。可以创建或更改序列，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。

序列修改

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**: 将序列重置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***: 将序列重置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。

限制

不能更改序列的数据类型。而是必须删除当前序列，然后创建序列，指定新的数据类型。

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。

在更改序列时，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。

过程

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。即使另一个连接在同时使用序列值也是如此。

示例

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.
```

删除序列

要删除序列，请使用 DROP 语句。

开始之前

删除序列时，语句的授权标识必须具有 DBADM 权限。

限制

不能使用 DROP SEQUENCE 语句删除系统为 IDENTITY 列创建的序列。

过程

要删除特定序列，请输入:

```
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。

结果

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必须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

序列引用是一个表达式，该表达式引用在应用程序服务器定义的序列。

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

- ***权限:*** 如果在某个语句中使用序列引用，该语句的授权标识持有的特权必须至少包括下列其中一项特权:
  - 序列上的 USAGE 特权
  - DATAACCESS 权限
- 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，如以下示例所示:

```
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。

- ***PREVIOUS VALUE 的范围:*** PREVIOUS VALUE 的值将持续存在，直到在当前会话中生成序列的下一个值，或者序列被删除或更改，或者应用程序会话结束。该值不受 COMMIT 或 ROLLBACK 语句影响。PREVIOUS VALUE 的值无法直接设置，并且是执行序列的 NEXT VALUE 表达式的结果。

  经常使用的技术，尤其对性能而言，应用程序或产品用它来管理一系列连接并将事务传递至任意连接。在这些情况中，序列的 PREVIOUS VALUE 应该仅在事务结尾时才可用。此种情况的示例包括使用 XA 协议、连接池、连接集中器和 HADR 进行故障转移的应用程序。
- 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。
- ***语法替换选择***: 与先前版本的 DB2 及其他数据库产品的兼容性支持下列替换。这些替换选择是非标准替换，不应该使用。
  - 可指定 NEXTVAL 和 PREVVAL 来代替 NEXT VALUE 和 PREVIOUS VALUE
  - 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 17 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 51 显示了表与视图之间的关系。

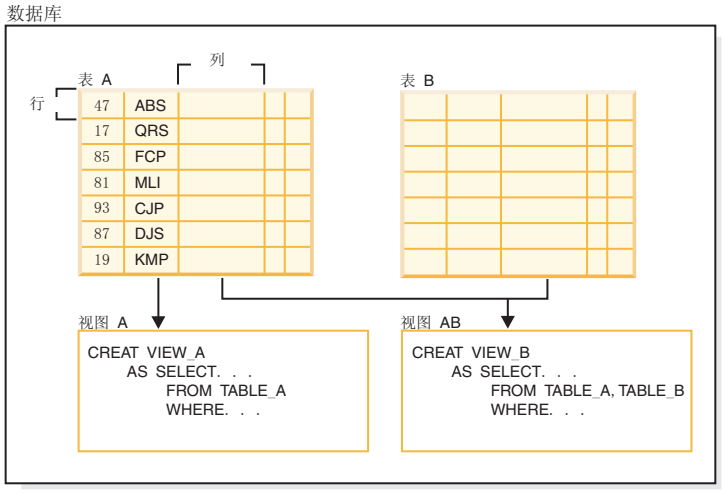


图 51. 表与视图之间的关系

因为视图允许多个用户查看同一数据的不同表示，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

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。

示例 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。

以下示例显示可插入视图。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。

以下示例未显示可删除视图，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。

关于此任务

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:

- 更改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。

以下示例显示了样本 CREATE VIEW 语句。基础表 EMPLOYEE 具有名为 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。如以上示例所示，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。如果必须挑选新的 UDF，那么必须重新创建视图。

关于此任务

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。

关于此任务

ALTER VIEW 语句通过更改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
- 更改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）。

过程

要使用命令行改变带类型视图，请发出 ALTER VIEW 语句。 例如:

```
ALTER VIEW view_name ALTER column_name
   ADD SCOPE typed_table_or_view_name
```

结果

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。

恢复不可用视图

不可用视图是不再能用于 SQL 语句的视图。

关于此任务

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。

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。

过程

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. 将当前模式设为 QUALIFIER 列的内容。
3. 将函数路径设为 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。）

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。

过程

要使用命令行删除视图，请输入:

```
DROP VIEW view_name
```

示例

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 18 章 游标

*游标*在应用程序中用于选择一组行，然后以一次一行的方式处理返回的数据。嵌入式 SQL 应用程序中的 SELECT 语句返回多行数据时，您需要一个机制，此机制使此返回数据或结果集对应用程序可用（逐行进行）。

游标类似与查询相关联的名称。游标是使用 DECLARE CURSOR 语句创建的，此语句定义游标的名称并指定其关联查询。三个其他 SQL 语句作用于游标。

OPEN 执行一个查询，此查询构建结果集并预编译游标以检索第一行。

FETCH
检索结果集的一行并将该行的值分配给目标变量。访存通常会重复执行，直到已检索结果集的所有行。

CLOSE
终止游标并释放它使用的所有资源。如果再次需要此游标，那么可重新打开此游标。

可将游标声明为挂起或未挂起、可返回或不可返回以及可滚动或不可滚动。

可挂起性
挂起的游标在落实操作后不会关闭。未挂起的游标会在落实操作后关闭。

可返回性
可返回游标会将结果集返回给过程或客户机应用程序的调用者。

可滚动性
不可滚动游标在结果集中顺序向前移动，并且每行只能被访问一次。可使用 FETCH 语句在结果集中随机移动可滚动游标。请注意，可滚动游标仅在 CLI、JDBC 和 SQLJ 应用程序中受支持。

第 19 章 用法列表

*用法列表*是一个数据库对象，用于记录每个引用特定表或索引的 DML 语句段。语句段是查询的可执行形式。每个语句段执行时，系统会捕获它的统计信息。如果要确定哪些 DML 语句（如果存在）影响了表或索引，请使用一些用法列表。

仅当用法列表处于活动状态时，才会在用法列表中收集数据。用法列表中的每个条目包含每个 DML 语句（该语句引用对其创建用法列表的表或索引）的数据。每个条目包括有关执行该段的次数以及聚集统计信息（用于指示该片段的所有执行对表或索引的影响）的信息。

可按下表中列示的值聚集列表中的引用。

表 78. 用法列表引用的聚集值

值	描述
executable_ID	标识所执行的 SQL 语句。
mon_interval_ID	标识将 *executable_ID* 添加至用法列表时的监视时间间隔。

考虑使用用法列表的以下示例。监视例程时，您发现 MON_GET_TABLE 表函数输出中特定表的 **rows_read** 监视元素的值很高。可对该表使用用法列表来标识哪些 DML 语句导致此高值。如果确定存在问题，那么可使用用法列表中的统计信息来确定哪些特定语句可能需要进一步监视或调整。

可为一个表或索引创建多个用法列表。但是，一次激活多个用法列表可能会对数据库性能和内存用法有负面影响。

限制

用法列表存在以下限制:

- 用法列表可仅捕获有关 DML 语句的信息。
- 可仅对无类型表创建用法列表。以下表类型和对象不受支持:
  - 别名
  - 创建临时表
  - 拆离表
  - 层次结构表
  - 昵称
  - 类型表
  - 视图
- 可仅对以下类型的索引创建用法列表:
  - 块索引
  - 集群索引
  - 维块索引
  - 常规索引

- **db2look** 实用程序不会抽取创建用法列表副本所需的 DDL 语句。

用法列表内存注意事项和验证依赖性

激活用法列表后，数据库管理器分配内存以存储第一次某节引用对其定义用法列表的对象时所收集的数据。在用法列表的整个生存期，各种操作可能会影响此内存和/或使用法列表失效。

常规内存注意事项如下所示:

- *用法列表大小注意事项:* 选择合理列表大小或将 **mon_heap_sz** 配置参数设为 AUTOMATIC 以便数据库管理器管理监视堆大小。
- *性能注意事项:* 为维护高性能，请创建用法列表以便可将它们限制为收集所需信息时需要的量。每个用法列表都需要系统内存；系统性能可能会降低，因为激活了其他用法列表。

下表更具体地说明各种操作对已分配内存的影响。

操作	影响	用法列表用于分区表或索引时的影响	分区数据库环境 或 DB2 pureScale 环境中的影响
第一次激活用法列表后，某节引用对其定义用法列表的对象。	对用法列表分配内存。	对每个数据分区分配内存。例如，如果用法列表需要 2 MB 内存，并且存在 3 个数据分区，那么会分配 6 MB 总内存。	对每个成员分配内存。例如，如果用法列表需要 2 MB 内存，并且存在 3 个成员，那么会分配 6 MB 总内存。
更改用法列表的大小。	下次激活用法列表时，与用法列表相关联的内存量会更改。	下次激活用法列表时，对于每个数据分区，与用法列表相关联的内存量会更改。	下次激活用法列表时，对于每个成员，与用法列表相关联的内存量会更改。
向对其定义用法列表的表或索引添加或连接新数据分区。	不适用。	下次某节引用表或索引时为新数据分区分配内存。	不适用。
删除用法列表。	将释放与该用法列表相关联的内存。	对所有数据分区释放与该用法列表相关联的内存。	在所有成员上释放与该用法列表相关联的内存。
删除对其定义用法列表的表或索引。	释放与该用法列表相关联的内存并使该用法列表的目录条目失效。可使用 ADMIN_REVALIDATE_DB_OBJECTS 过程使该目录条目再次生效。	对所有数据分区释放与该用法列表相关联的内存并使该用法列表的目录条目失效。可使用 ADMIN_REVALIDATE_DB_OBJECTS 过程使该目录条目再次生效。	在所有成员上释放与该用法列表相关联的内存并使该用法列表的目录条目失效。可使用 ADMIN_REVALIDATE_DB_OBJECTS 过程使该目录条目再次生效。
取消激活实例或数据库。	将释放与该用法列表相关联的内存。	对所有数据分区释放与该用法列表相关联的内存。	在所有成员上释放与该用法列表相关联的内存。
使用 SET USAGE LIST STATE 语句来释放与该用法列表相关联的内存。	将释放与该用法列表相关联的内存。	对所有数据分区释放与该用法列表相关联的内存。	在所有成员上释放与该用法列表相关联的内存。
从对其创建用法列表的表或索引拆离数据分区。	不适用。	释放与您拆离的数据分区相关联的内存。	不适用。
删除或取消激活数据库成员。	不适用。	不适用。	释放与您删除或取消激活的成员相关联的内存。

第 4 部分 参考

第 20 章 符合命名规则

一般命名规则

为所有数据库对象、用户名、密码、组、文件和路径命名时都要遵循一些规则。其中某些规则特定于所用的平台。

例如，关于在文件系统中的可视对象（数据库和实例等等）的名称中使用大小写字母的规则:

- 在 UNIX 平台上，名称区分大小写。例如，`/data1` 与 `/DATA1` 或 `/Data1` 不是同一个目录。
- 在 Windows 平台上，名称不区分大小写。例如，`\data1` 与 `\DATA1` 和 `\Data1` 表示同一个目录。

除非另有指定，否则所有名称均可以包括下列字符:

- 字母 A 到 Z 以及 a 到 z，如基本（7 位）ASCII 字符集中定义的那样。在使用 SQL 语句创建的对象的标识中使用这些字母时，除非使用引号 (″) 对小写字符“a”到“z”进行定界，否则它们将转换为大写。
- 0 至 9。
- ! % ( ) { } . - ^ ~ _（下划线） @、#、$ 和空格。
- \（反斜杠）。

限制

- 名称不要以数字或者下划线字符开头。
- 不要使用 SQL 保留字来命名表、视图、列、索引或授权标识。
- 对于目录名和文件名，仅使用在基本 ASCII 字符集中定义的字母。虽然您的计算机操作系统可能支持使用不同的代码页，但是，非 ASCII 字符可能不会稳定工作。在分布式环境中使用非 ASCII 字符可能是一个特殊问题。在分布式环境中，不同的计算机可能使用不同的代码页。
- 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。
- 用户名和组名还必须遵循特定操作系统强制实施的规则。例如，在 Linux 和 UNIX 平台上，用户名和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。
- **对 AUTHID 标识的限制**: 在 DB2 V9.5 及更高版本中，您可以具有一个 128 字节的授权标识。然而，当授权标识被解释为操作系统用户标识或组名时，存在操作系统命名限制。例如，在 Linux 和 UNIX 操作系统上，用户标识和组名最多可以包含 8 个字符，在 Windows 操作系统上，用户标识和组名最多可以包含 30 个字符。因此，虽然您可以授予一个 128 字节的授权标识，但是您无法作为具有该授权标识的用户进行连接。如果您编写自己的安全插件，那么可以使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，该用户标识会在认证期间返回一个您可以连接至的 128 字节的授权标识。

还必须考虑对象命名规则、多元文化支持的环境中的命名规则以及 Unicode 环境中的命名规则。

DB2 对象命名规则

所有对象都遵从一般命名规则。此外，某些对象具有下表所示的其他限制。

表 79. 数据库、数据库别名和实例命名规则

对象	准则
• 数据库 • 数据库别名 • 实例	• 在编目数据库名称的位置中，这些数据库名称必须唯一。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。 • 在系统数据库目录中，数据库别名必须唯一。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。 • 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。 • 在 Windows 上，任何实例都不能与服务名称同名。 **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 80. 数据库对象命名规则

对象	准则
• 别名 • 审计策略 • 缓冲池 • 列 • 事件监视器 • 索引 • 方法 • 节点组 • 程序包 • 程序包版本 • 角色 • 模式 • 存储过程 • 表 • 表空间 • 触发器 • 可信上下文 • UDF • UDT • 视图	• 这些对象的标识的长度必须小于或等于*SQL Reference*中的『SQL 和 XML 限制』中列示的长度。对象名还可以包括: – 有效的重音字符（例如，ö） – 除了多字节空格之外的多字节字符（对于多字节环境） • 程序包名和程序包版本还可以包括句号 (.)、连字符 (-) 和冒号 (:) 有关更多信息，请参阅*SQL Reference*中的『标识』。

表 81. 联合数据库对象命名规则

对象	准则
• 函数映射 • 索引规范 • 昵称 • 服务器 • 类型映射 • 用户映射 • 包装器	这些对象的长度必须小于或等于*SQL Reference*中的『SQL 和 XML 限制』中列示的长度。联合数据库对象的名称还可以包括: • 有效的强调字母（例如 ö） • 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) 的模式名长度不能超出*SQL Reference*中的『SQL 和 XML 限制』中列示的长度。
- 下列模式名是保留字，不能使用: SYSCAT、SYSFUN、SYSIBM、SYSSTAT 和 SYSPUBLIC。
- 为了避免将来升级数据库时的潜在问题，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 82. 用户、用户标识和组命名规则

对象	准则
• 组名 • 用户名 • 用户标识	• 组名长度必须小于或等于*SQL Reference* 中的『SQL 和 XML 限制』中列示的组名长度。 • 在 Linux 和 UNIX 操作系统上，用户标识最多可以包含 8 个字符。 • 在 Windows 上，用户名最多可以包含 30 个字符。 • 未使用“客户机”认证方法时，如果明确指定用户名和密码，那么支持使用长度超过*SQL Reference* 中的『SQL 和 XML 限制』所列示用户名长度的用户名将非 Windows 32 位客户机连接至 Windows。 • 名称和标识不能: – 是 USERS、ADMINS、GUESTS、PUBLIC、LOCAL 或任何 SQL 保留字。 – 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. 成功的 CONNECT 或 ATTACH 所返回的授权标识将被截断为*SQL Reference* 中的『SQL 和 XML 限制』所列示的授权名长度。将省略号 (...) 追加至授权标识并且 SQLWARN 字段包含指示截断的警告。
3. 从用户标识和密码中除去尾部空格。

4. **对 AUTHID 标识的限制**: 在 DB2 V9.5 及更高版本中，您可以具有一个 128 字节的授权标识。然而，当授权标识被解释为操作系统用户标识或组名时，存在操作系统命名限制。例如，在 Linux 和 UNIX 操作系统上，用户标识和组名最多可以包含 8 个字符，在 Windows 操作系统上，用户标识和组名最多可以包含 30 个字符。因此，虽然您可以授予一个 128 字节的授权标识，但是您无法作为具有该授权标识的用户进行连接。如果您编写自己的安全插件，那么可以使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，该用户标识会在认证期间返回一个您可以连接至的 128 字节的授权标识。

多国语言环境中的命名规则

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。在多国语言环境中使用特殊字符 #、 和 $ 时应小心，因为它们未包括在多国语言主机 (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 21 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，在本地计算机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 并且 **dir_cache**=NO，那么始终将从 LDAP 中读取信息。
- 如果 **DB2LDAPCACHE**=NO 并且 **dir_cache**=YES，那么将从 LDAP 中读取一次信息，然后将它插入到 DB2 高速缓存中。
- 如果 **DB2LDAPCACHE**=YES 或者未设置此注册表变量，那么将从 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。

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，这些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 83. cimManagedElement

类	cimManagedElement
Active Directory LDAP 显示名	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 84. cimSetting

类	cimSetting
Active Directory LDAP 显示名	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 85. eProperty

类	eProperty
Active Directory LDAP 显示名	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 85. *eProperty (续)*

类	eProperty
可选属性	propertyType cisPropertyType cisProperty cesPropertyType cesProperty binPropertyType 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 86. *DB2Node*

类	DB2Node
Active Directory LDAP 显示名	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 db2instanceName db2Type host/dNSHostName（请参阅注释 2） 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
特殊注意事项	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。 2. *host* 在 IBM Tivoli Directory Server 环境中使用。*dNSHostName* 属性用于 Microsoft Active Directory 中。 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 87. DB2Database

类	**DB2Database**
Active Directory LDAP 显示名	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 db2nodePtr
可选属性	db2databaseAlias db2additionalParameters db2ARLibrary db2authenticationLocation db2gwPtr db2databaseRelease DCEPrincipalName db2altgwPtr 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 88. db2additionalParameters

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

表 88. *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 89. *db2authenticationLocation*

属性	db2authenticationLocation
Active Directory LDAP 显示名	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	有效值有：CLIENT、SERVER、DCS、DCE、KERBEROS、SVRENCRYPT 或 DCSENCRYPT

表 90. *db2ARLibrary*

属性	db2ARLibrary
Active Directory LDAP 显示名	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 91. *db2databaseAlias*

属性	db2databaseAlias
Active Directory LDAP 显示名	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
多值	多值

表 91. db2databaseAlias（续）

属性	db2databaseAlias
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 92. db2databaseName

属性	db2databaseName
Active Directory LDAP 显示名	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 93. db2databaseRelease

属性	db2databaseRelease
Active Directory LDAP 显示名	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 94. db2nodeAlias

属性	db2nodeAlias
Active Directory LDAP 显示名	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 95. db2nodeName

属性	db2nodeName
Active Directory LDAP 显示名	ibm-db2NodeName
Active Directory 公共名（cn）	ibm-db2NodeName

表 95. *db2nodeName（续）*

属性	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 96. *db2nodePtr*

属性	db2nodePtr
Active Directory LDAP 显示名	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 97. *db2altnodePtr*

属性	db2altnodePtr
Active Directory LDAP 显示名	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 98. *db2gwPtr*

属性	db2gwPtr
Active Directory LDAP 显示名	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
最大长度	1000

表 98. db2gwPtr (续)

属性	db2gwPtr
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 99. db2altgwPtr

属性	db2altgwPtr
Active Directory LDAP 显示名	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 100. db2instanceName

属性	db2instanceName
Active Directory LDAP 显示名	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 101. db2Type

属性	db2Type
Active Directory LDAP 显示名	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	数据库服务器的有效类型是：SERVER、MPP 和 DCS

表 102. DCEPrincipalName

属性	DCEPrincipalName
Active Directory LDAP 显示名	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 103. cesProperty

属性	cesProperty
Active Directory LDAP 显示名	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 格式的数据。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 104. cesPropertyType

属性	cesPropertyType
Active Directory LDAP 显示名	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 105. cisProperty

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

表 105. 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 106. cisPropertyType

属性	cisPropertyType
Active Directory LDAP 显示名	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 107. binProperty

属性	binProperty
Active Directory LDAP 显示名	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 108. binPropertyType

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

表 108. 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 109. PropertyType

属性	PropertyType
Active Directory LDAP 显示名	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 110. settingID

属性	settingID
Active Directory LDAP 显示名	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。

IBM Tivoli Directory Server 是一款 LDAP V6.2 服务器并可用于 Windows、AIX、Solaris、Linux 和 HP-UX，而在 AIX 和 System i® 上，它与 OS/390 Security Server 一起作为基本操作系统的组成部分交付。

DB2 数据库支持安装在 AIX、Solaris、HP-UX 11.11、Windows 和 Linux 上的 IBM LDAP 客户机。

Microsoft Active Directory 服务器是一款 LDAP V3 服务器，可作为操作系统的 Windows 2000 Server 和 Windows Server 2003 系列的部件提供。

Microsoft LDAP 客户机是 Windows 操作系统附带包括的。

表 111. 受支持的 LDAP 客户机和服务器配置

受支持的 LDAP 客户机和服务器配置	IBM Tivoli Directory Server	Microsoft Active Directory 服务器	Sun One LDAP 服务器
IBM LDAP 客户机	是否受支持	是否受支持	是否受支持
Microsoft LDAP/ADSI 客户机	是否受支持	是否受支持	是否受支持

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。

关于此任务

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。

过程

通常，可在 LDAP 中手动配置主机数据库信息，以便不要求每台客户机在每台计算机上以本地方式手动编目该数据库和节点。过程如下所示:

1. 在 LDAP 中注册主机数据库服务器。

   在 **REGISTER** 命令中，必须使用 **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** 子句可以设为主机名或主机服务器的 LU 名。**INSTANCE** 子句可以设为长度不超过 8 个字符的任何字符串。（例如，可以将实例名设为 DB2。）必须将 **NODETYPE** 子句设为 DCS 以指示这是主机数据库服务器。
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为 SERVER。

示例

以下示例显示与主机数据库的直接连接以及通过网关与主机数据库的连接。有一个称为 NIAGARA_FALLS 的主机数据库，它可接受使用 TCP/IP 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。

必须完成下列步骤:

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 TCP/IP 连接至主机。

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 V8.2 及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 ) )
```

可以在以下 URL 处找到 `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。

扩展 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**查看**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 复制到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，需要满足某些先决条件。

开始之前

- 运行 DB2 数据库的机器必须属于 Windows 2000 或 Windows Server 2003 域。
- 必须安装 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。

过程

1. 启用 LDAP 支持。
2. 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，必须具有足够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，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，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. 通过使用**高级**设置，设置许可权以应用于“此对象和所有子对象”。
8. 选中**允许可继承许可权从父代传播至此对象**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*。

关于此任务

必须通过运行“DB2 模式安装”程序 **db2schex** 来扩展 Active Directory 的模式。您应当在安装 DB2 产品并创建数据库之前运行此命令，否则您必须手动注册节点和编目数据库。

db2schex 程序位于产品 CD-ROM 上的以下位置: `x:\db2\windows\utilities\`，其中 x: 是 DVD 驱动器盘符。

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 Windows 域中包含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 驱动器盘符。

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 V8.2 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
```

如果在 Windows 上尚未运行较早版本的 DB2 数据库管理系统中的 **db2schex** 命令，那么当您在 DB2 V9.7 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。

此命令还有其他相关联的可选子句。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
  ```

完成安装之后启用 LDAP 支持

必须在完成 DB2 数据库产品的安装并启用 LDAP 之后，才能使用 LDAP。

过程

要启用 LDAP 支持:

1. 在任何属于 Windows 域的机器上，请执行以下步骤:
   a. 如果您在安装 DB2 数据库产品之前未启用 LDAP 支持，而您又想使用 Microsoft Active Directory，那么您必须扩展目录模式。 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“利用 LDAP 目录”支持来安装 LDAP 支持二进制文件。 “**安装程序**”会将 DB2 注册表变量 **DB2_ENABLE_LDAP** 自动设为 YES；要启用 LDAP 支持，必须满足此设置。
   c. 可选: 要使用 IBM LDAP 客户机而不使用 Microsoft LDAP 客户机，请将 **DB2LDAP_CLIENT_PROVIDER** 注册表变量设为 IBM。
2. 在每台 LDAP 客户机中，请执行以下步骤:

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，*[:port_number]* 是端口号。如果您未指定端口号，那么将使用缺省 LDAP 端口号 389。对于支持 SSL 的 LDAP 服务器，请运行以下命令：`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 数据库产品之后扩展了目录模式，请执行以下步骤:

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
```

下一步做什么

现在，您就可以注册 LDAP 条目。

注: 使用 SSL 从 LDAP 服务器检索目录信息的功能当前不受支持。要确保使用 SSL 对客户机与服务器之间交换的数据进行加密，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。

关于此任务

当注册数据库服务器的实例时，必须指定*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 Enterprise Server Edition 创建实例时，必须对每台计算机都调用一次 **REGISTER** 命令。使用 **rah** 命令在所有计算机上发出 **REGISTER** 命令。

注: 每台计算机都不能使用相同的 *ldap_node_name*，因为在 LDAP 中该名称必须唯一。将要以每台计算机的主机名替换 **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** 命令的每台计算机的主机名。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计算机上的服务器，而不必为每台计算机编目独立的 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。

过程

- 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，如果本地计算机上的数据库目录中不存在该数据库，那么搜索 LDAP 目录。

关于此任务

如果该名称存在于 LDAP 目录中，那么仍然会在本地计算机上创建该数据库，但是会返回一条警告消息，说明在 LDAP 目录中存在命名冲突。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。

过程

- 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。

关于此任务

在本地计算机或远程计算机上注销 DB2 服务器时，需要为服务器指定 LDAP 节点名。

过程

要从 LDAP 注销 DB2 服务器，请执行以下操作:

- 从命令行使用 **DEREGISTER** 命令:

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

- 从客户机应用程序调用 db2LdapDeregister API。

结果

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。

过程

要从 LDAP 目录注销数据库，请执行以下操作:

- 从命令行使用 **UNCATALOG LDAP DATABASE** 命令:

  ```
  db2 uncatalog ldap database dbname
  ```

- 从客户机应用程序调用 **db2LdapUncatalogDatabase** API。

配置 LDAP 用户

创建 LDAP 用户

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。

关于此任务

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。

用于 IBM Tivoli Directory Server 的 **LDIF** 实用程序是 **LDIF2DB**。

包含人员对象属性的 LDIF 文件类似如下内容:

```
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。

过程

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。

关于此任务

DB2 for Linux, UNIX, and Windows 具有高速缓存机制。将用户级 DB2 概要文件注册表变量高速缓存到本地计算机上。

当发生下列情况时，会刷新高速缓存:

- 更新或重置用户级 DB2 注册表变量。
- 您发出在用户级刷新 LDAP 概要文件变量的命令:

  ```
  db2set -ur
  ```

过程

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。
如果指定了 **-ul** 参数，DB2 for Linux, UNIX, and Windows 将始终从高速缓存中读取 DB2 注册表变量。

禁用 LDAP 支持

过程

要禁用 LDAP 支持，请执行以下操作:

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。

关于此任务

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。

过程

- 要更新本地计算机上的 LDAP 中的 DB2 服务器，请使用 **UPDATE LDAP** 命令：
- 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。

开始之前

DB2_ENABLE_LDAP 注册表变量必须设为“YES”。

关于此任务

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*。

关于此任务

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计算机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。

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，会根据数据库管理器配置参数 **dir_cache** 和注册表变量 **DB2LDAPCACHE** 的值，在本地计算机上存储或高速缓存此信息。

- 如果 **DB2LDAPCACHE**=NO 并且 **dir_cache**=NO，那么始终将从 LDAP 中读取信息。
- 如果 **DB2LDAPCACHE**=NO 并且 **dir_cache**=YES，那么将从 LDAP 中读取一次信息，然后将它插入到 DB2 高速缓存中。
- 如果 **DB2LDAPCACHE**=YES 或者未设置此注册表变量，那么将从 LDAP 服务器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: 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。

关于此任务

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，在本地计算机上存储或高速缓存此信息。

- 如果 **DB2LDAPCACHE**=NO 并且 **dir_cache**=NO，那么始终将从 LDAP 中读取信息。
- 如果 **DB2LDAPCACHE**=NO 并且 **dir_cache**=YES，那么将从 LDAP 中读取一次信息，然后将它插入到 DB2 高速缓存中。
- 如果 **DB2LDAPCACHE**=YES 或者未设置此注册表变量，那么将从 LDAP 服务器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: 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，因此可能必须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。

过程

- 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
  ```

- 要刷新现有本地数据库和节目条目并从 LDAP 添加新条目，请使用以下命令:

  ```
  db2 refresh ldap immediate all
  ```

指定 **IMMEDIATE ALL** 参数可将随 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 for Linux, UNIX, and Windows 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
  ```

结果

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。

下一步做什么

为了确保及时进行刷新，可能要采用以下几种方法:

- 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。
- 设置 **DB2LDAPCACHE**=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。

关于此任务

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，一旦在每台计算机上记录了该条目，如果启用了高速缓存，那么搜索范围就可更改为 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 22 章 SQL 和 XML 限制

此主题中的表描述了 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 112 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 112. 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

表 112. 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	128
程序包版本标识	64
参数名称	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
存储器组	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

表 112. 标识长度限制 (续)

描述	最大值（以字节计）
用户定义的类型名[2]	128
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: 1. 各个主语言编译器可能对变量名具有更严格的限制。 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 113. 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	1.0000000000000000E-383

表 113. 数字限制 (续)

描述	限制
最大的负 DECFLOAT(16) 值	-1.000000000000000E-383
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 114. 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 114. 字符串限制 (续)

描述	限制
注: 1. 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。	

表 115. 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 116. 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	0001-01-01-00.00.00.000000000000
最大的 TIMESTAMP 值	9999-12-31-24.00.00.000000000000
最小的时间戳记精度	0
最大的时间戳记精度	12

表 117. 数据库管理器限制

类别	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	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

表 117. 数据库管理器限制 (续)

类别	描述	限制
约束	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
	UNIQUE 约束中的最大列组合长度（通过 UNIQUE 索引受支持，以字节计）[8]	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）[8]	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
	DB2 pureScale 环境中最大量的 成员	128
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）[6] [8]	*indexpagesize*/4
	可变索引键部件的最大长度（以字节计）[7]	1022 或存储器
	SMS 表空间中每个数据库分区的最大索引大小（以千兆字节计）[6]	64
	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）[6]	512
	大型 DMS 表空间中每个数据库分区的最大索引大小（以千兆字节计）[6]	64
	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 FFFF FFFF FFFF
监视	同时处于活动状态的事件监视器的最大数目	128
	在 DB2 分区数据库环境中，同时处于活动状态的 GLOBAL 事件监视器的最大数目	32
例程	带 LANGUAGE SQL 的进程中的最大参数数	32767
	带 PROGRAM TYPE MAIN 的外部进程中的最大参数数	32767
	带 PROGRAM TYPE SUB 的外部进程中的最大参数数	90
	游标值构造函数中的最大参数数	32767
	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

表 117. 数据库管理器限制 (续)

类别	描述	限制
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
	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[6]	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[6]	1012
	GROUP BY 子句中的列的最大总长度（以字节计）[6]	32 677
	ORDER BY 子句中的最大列数 [6]	1012
	ORDER BY 子句中的列的最大总长度（以字节计）[6]	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 [6]	1012
存储器组	数据库中最大数量的存储器组	256
	存储器组中最大数量的存储路器径	128
	最大长度的存储器路径（以字节计）	175
表格和视图	表中的最大列数 [6]	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] [6]	32 677
	每个数据库分区的非分区表中的最大行数	128×10^{10}
	每个数据库分区的数据分区中的最大行数	128×10^{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] [6]	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）[6]	64
	单个表的最大数据分区数	32 767
	最大表分区列数	16
	用户定义的行类型中的最大字段数	1012

表 117. 数据库管理器限制 (续)

类别	描述	限制
表空间	每个表或表分区中 LOB 对象的最大大小（以太字节计）	4
	每个表或表分区中 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] [6]	512
	大型 DMS 表空间的最大大小（以太字节计）[3] [6]	64
	临时 DMS 表空间的最大大小（以太字节计）[3] [6]	64
	DMS 表空间中的最大表对象数	请参阅表 118
触发器	级联触发器的最大运行时深度	16
用户定义的类型	结构化类型中的最大属性数	4082
工作负载管理器	每个数据库最大数量的用户定义的服务超类	64
	每个服务超类最大数量的用户定义的服务子类	61

注:

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。
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。
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。
4. 实际值由 **max_connections** 和 **max_coordagents** 数据库管理器配置参数控制。
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。
6. 对于特定于页大小的值，请参阅表 118。
7. 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。
8. 最大值可能小一些，这取决于索引选项。

表 118. 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] 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

表 118. 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 表空间中的每个数据库分区的最大表大小（以千兆字节计）	8	16	32	64
包括所有开销在内的最长索引键（以字节计）	1024	2048	4096	8192
SMS 表空间中每个数据库分区的最大索引大小（以千兆字节计）	8	16	32	64
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）	8	16	32	64
常规 DMS 表空间的最大大小（以千兆字节计）	64	128	256	512
大 DMS 表空间的最大大小（以太字节计）	8	16	32	64
临时 DMS 表空间的最大大小（以太字节计）	8	16	32	64
选择列表中的最大元素数	500[4]	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

表 118. 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
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。 4. 假如仅有的系统临时表空间为 4KB 且数据溢出至排序缓冲区，那么会产生错误。如果结果集可以适合内存，那么将不会产生错误。				

第 23 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。使用 DB2 概要文件注册表来查看和更新有关变量和实例的信息。

在引入 DB2 数据库概要文件注册表之前，要设置环境变量，就会要求您为环境变量指定值并重新启动计算机。现在，您可以使用 DB2 概要文件注册表来控制会影响 DB2 数据库环境的大多数变量。

从一台计算机中使用概要文件注册表来控制环境变量。通过不同的概要文件提供了不同级别的支持。可以使用 DB2 管理服务器来远程管理环境变量。

DB2 数据库会受到下列概要文件注册表的影响:

- DB2 实例级别的概要文件注册表包含实例的注册表变量。在此注册表中定义的值将覆盖它们在全局注册表中的设置。
- DB2 全局级别的概要文件注册表包含在没有为实例设置注册表变量的情况下使用的设置。与 DB2 Enterprise Server Edition 的特定副本有关的所有实例都可以访问此注册表。
- DB2 实例节点级别的概要文件注册表包含特定于分区数据库环境中的数据库分区的变量设置。在此注册表中定义的值将覆盖它们在实例级别和全局级别的设置。
- DB2 用户级别的概要文件注册表包含特定于每个用户的设置。在此注册表中定义的值将覆盖它们在其他注册表中的设置。

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. 在概要文件注册表外部设置的环境变量。
2. 对用户级别的概要文件设置的注册表变量。
3. 对实例节点级别的概要文件设置的注册表变量。
4. 对实例级别的概要文件设置的注册表变量。
5. 对全局级别的概要文件设置的注册表变量。

DB2 实例概要文件注册表包含与当前副本相关联的所有实例的列表。每个 DB2 副本都存在一个列表。通过运行 **db2ilist** 命令，即可查看系统上提供的所有实例的完整列表。此概要文件注册表不包含变量设置。

概要文件注册表位置和权限要求

DB2 概要文件注册表在每个操作系统上具有不同的位置和权限要求。需要具有权限才能更新每个概要文件注册表中的变量值。

表 119. 概要文件注册表位置和权限要求

概要文件注册表	Windows 上的位置	Linux 和 UNIX 上的位置	Linux 和 UNIX 权限要求	Windows 权限要求
实例级别的概要文件注册表	\HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\PROFILES*instance_name*	*instance_home*/sqllib/profile.env 其中 *instance_home* 是实例所有者的主路径。	-rw-rw-r-- *instance_owner instance_owner_group* profile.env	您必须是 DB2 管理员组 (DB2ADMNS) 的成员。
全局级别的概要文件注册表	\HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\GLOBAL_PROFILE	对于 root 用户安装：/var/db2/global.reg 对于非 root 用户安装：*home_directory*/sqllib/global.reg	要在 root 用户安装版本中修改全局注册表变量，您必须以具有 root 用户权限的用户身份登录。	您必须是 DB2 管理员组 (DB2ADMNS) 的成员。
实例节点级别的概要文件注册表	...\SOFTWARE\IBM\DB2\PROFILES*instance_name*\NODES*node_number*	*instance_home*/sqllib/nodes/*node_number*.env 其中 *instance_home* 是实例所有者的主路径。	对于包含此文件的目录： drwxrwsr-w *instance_owner instance_owner_group* 节点 对于此文件： -rw-rw-r-- *instance_owner instance_owner_group node_number*.env	您必须是 DB2 管理员组 (DB2ADMNS) 的成员。
用户级别的概要文件注册表	轻量级目录访问协议 (LDAP) 目录。	不适用。	不适用。	您必须是 DB2 管理员组 (DB2ADMNS) 的成员。
实例概要文件注册表	\HKEY_LOCAL_computer\SOFTWARE\IBM\DB2\PROFILES*instance_name*	对于 root 用户安装：/var/db2/global.reg 对于非 root 用户安装：*home_directory*/sqllib/global.reg	不需要任何权限。	不需要任何权限。

设置登记表变量和环境变量

大多数环境变量是使用 **db2set** 命令在 DB2 数据库概要文件注册表中设置的。根据您使用的操作系统，少量在概要文件注册表外部设置的变量需要使用另外的命令来设置。

开始之前

请确保您具有设置注册表变量所需要的特权。

在 Linux 和 UNIX 操作系统上，您必须具有下列特权：

- 具有在实例级别的注册表中设置变量的 SYSADM 权限
- 具有在全局级别的注册表中设置变量的 root 用户权限

在 Windows 操作系统上，您必须具有下列其中一种特权：

- 本地管理员权限
- 满足下列条件的 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，并且在 Windows 注册表中为 SYSADM 用户授予了适当的许可权，那么这些 SYSADM 用户就可以进行更新。

关于此任务

当您使用 **db2set** 命令在概要文件注册表中设置变量时，不需要重新启动计算机即可使变量值生效。但是，所作的更改不会影响当前正在运行的 DB2 应用程序或者处于活动状态的用户。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。

如果 DB2 变量是在注册表外部设置的，那么您无法远程管理这些变量。此外，您必须重新启动计算机才能使变量值生效。

DB2INSTANCE 和 **DB2NODE** DB2 环境变量未存储在 DB2 概要文件注册表中。请参阅有关在概要文件注册表外部设置环境变量的主题，以了解有关设置这些变量的信息。

在 Linux 和 UNIX 操作系统上，实例级别的概要文件注册表存储在 `profile.env` 文本文件中。如果两个或更多用户几乎同时使用 **db2set** 命令来设置注册表变量，那么此文件的大小将减小为零。此外，**db2set -all** 命令的输出将显示不一致的值。

过程

要设置注册表变量，请执行以下操作:

发出附带有关参数的 **db2set** 命令。
下表说明了您使用 **db2set** 命令设置注册表变量时可以采用的一些方式。请参阅 **db2set** 命令参考主题，以了解有关此命令的参数及其用法的更多信息。

表 120. 用于设置注册表变量的常见命令

期望执行的操作	命令
为当前实例或缺省实例设置注册表变量。	db2set *registry_variable_name=new_value*
为实例中的所有数据库设置注册表变量。	db2set *registry_variable_name=new_value* -i *instance_name*
为实例中的特定数据库分区设置注册表变量。	db2set *registry_variable_name=new_value* -i *instance_name database_partition_number*
为与 DB2 Enterprise Server Edition 安装有关的所有实例设置注册表变量。	db2set *registry_variable_name=new_value* -g
在轻量级目录访问协议 (LDAP) 环境中，在用户级别设置注册表变量。	db2set *registry_variable_name=new_value* -ul
在 LDAP 环境中，在全局级别设置注册表变量。**DB2LDAP_KEEP_CONNECTION** 和 **DB2LDAP_SEARCH_SCOPE** 是仅有的两个可以在 LDAP 全局级别设置的注册表变量。	db2set *registry_variable_name=new_value* -gl

提示: 如果注册表变量要求将布尔值作为自变量，那么 YES、1、TRUE 和 ON 这些值都等价，并且 NO、0、FALSE 和 OFF 这些值也等价。对于任何变量，可以指定任何适当的等价值。

在 Windows 上，在概要文件注册表外部设置环境变量

在 Windows 操作系统上，只能在概要文件注册表外部设置 **DB2INSTANCE**、**DB2NODE** 和 **DB2PATH** 环境变量。您仅需要设置 **DB2PATH** 变量。

关于此任务

在 Windows 操作系统上，将在概要文件注册表外部设置下列环境变量:

- **DB2INSTANCE** 环境变量指定缺省情况下处于活动状态的实例。如果未设置此变量，那么 DB2 数据库管理器使用 **DB2INSTDEF** 变量的值作为当前实例。
- **DB2NODE** 环境变量指定将向其传递请求的数据库分区服务器的目标逻辑节点。
- **DB2PATH** 环境变量指定在 32 位 Windows 操作系统上安装了 DB2 数据库产品的目录。

如果您想要设置任何其他变量，那么必须在一个或多个概要文件注册表中设置这些变量。

可以使用 **echo** 命令来确定环境变量的值。例如，要检查 **DB2NODE** 环境变量的值，请发出以下命令:

```
echo %db2path%
```

过程

要在概要文件注册表外部设置环境变量，请执行以下操作:

使用下列其中一个选项来设置环境变量。

选项	描述
在实例级别设置此环境变量。	1. 遵循适用于 Windows 操作系统的适当过程。 2. 重新启动计算机。

为当前会话设置此环境变量。	发出以下命令:
	`set env_variable_name=new_value`
	`db2start`

为 C shell 的当前会话设置此环境变量。	发出以下命令: `setenv env_variable_name new_value`

在 Linux 和 UNIX 操作系统上，在概要文件注册表外部设置环境变量

在 Linux 和 UNIX 操作系统上，必须在概要文件注册表外部设置 **DB2INSTANCE** 系统环境变量。如果您想要设置任何其他变量，那么必须在一个或多个概要文件注册表中设置这些变量。

关于此任务

可以使用 db2profile 脚本（对于 Bourne 或 Korn shell）和 db2cshrc 脚本（对于 C shell）将 **DB2INSTANCE** 变量设为您指定的实例名称。这些脚本位于 *instance_home*/sqllib 目录中，其中 *instance_home* 是实例所有者的主目录。

实例所有者和具有 SYSADM 特权的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。

要设置 DB2 数据库管理器不支持的变量，请在 userprofile 和 usercshrc 脚本文件中定义这些变量。*instance_home*/sqllib 目录中也存在这些文件。

过程

要在概要文件注册表外部设置环境变量，请执行以下操作:

使用下列其中一种方法来设置环境变量:

选项	描述
在实例级别为 Bourne 或 Korn shell 设置环境变量。	运行 db2profile 脚本。
在实例级别为 C shell 设置环境变量。	运行 db2cshrc 脚本。
为 Bourne shell 的当前会话设置此环境变量。	发出以下命令: export *env_variable_name=new_value*
为 C shell 的当前会话设置此环境变量。	发出以下命令: setenv *env_variable_name new_value*
为 Korn shell 的当前会话设置此环境变量。	发出以下命令: *environment_variable_name=new_value* export *environment_variable_name*

标识当前实例

缺省情况下，您发出的大多数命令或者您对配置所作的更改将应用于当前实例。可以通过检查某些环境变量的值来标识当前实例。

关于此任务

当您运行命令以对实例启动或停止数据库管理器时，数据库管理器将该命令应用于当前实例。要确定当前实例，数据库管理器将按以下顺序检查某些环境变量的值:

1. 当前会话的 **DB2INSTANCE** 环境变量的值。
2. **DB2INSTANCE** 系统环境变量的值。
3. 在 Windows 操作系统上，**DB2INSTDEF** 注册表变量的值。

过程

要标识当前实例，请执行下列操作:

检查有关的环境变量的值。

选项	描述
查看当前会话的 DB2INSTANCE 环境变量的值。	发出以下命令: `db2 get instance`

查看 DB2INSTANCE 系统环境变量的值。	• 在 Windows 操作系统上，请发出以下命令:
	`echo %DB2INSTANCE%`
	• 在 Linux 和 UNIX 操作系统上，请发出以下命令:
	`echo $DB2INSTANCE`

查看 DB2INSTDEF 注册表变量的值。	发出以下命令: `db2set DB2INSTDEF`

在分区数据库环境中设置实例级别的变量

在分区数据库环境中，您在实例级别概要文件注册表中设置注册表变量所采用的方式取决于您使用的操作系统。

关于此任务

在 Linux 和 UNIX 操作系统中，实例级别的概要文件注册表存储在 `sqllib` 目录下的一个文本文件中。因为 `sqllib` 目录在所有物理数据库分区共享的文件系统中，所以您可以从任何数据库分区中设置注册表变量。

在 Windows 操作系统中，DB2 数据库管理器将实例级别的概要文件注册表存储在 Windows 注册表中。因此，物理数据库分区之间不共享数据。要对所有数据库分区设置某个变量，必须使用 **rah** 命令以确保您用来设置此变量的命令在所有计算机上运行。如果您从数据库分区中设置注册表变量并且不使用 **rah** 命令，那么在当前实例中将只为该数据库分区设置此变量。

还可以使用 **DB2REMOTEPREG** 注册表变量将不是实例所有者的计算机配置为使用拥有实例的计算机上的注册表变量的值。

过程

要为当前实例的所有数据库分区设置注册表变量，请执行以下操作:

从任何数据库分区针对您所使用的操作系统发出命令。

- 在 Linux 和 UNIX 操作系统上，请发出以下命令:

  `db2set` *`registry_variable_name=new_value`*
- 在 Windows 操作系统上，请发出以下命令:

  `rah db2set` *`registry_variable_name=new_value`*

聚集注册表变量

使用聚集注册表变量将几个注册表变量组成一个配置，由另一个注册表变量名称来标识此配置。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。

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。要控制数据库环境，可将聚集注册表变量 **DB2_WORKLOAD** 设为 SAP 并将 **DB2_SKIPDELETED** 注册表变量修改为 NO。通过运行 **db2set** 命令，您将接收到下列结果:

```
DB2_WORKLOAD=SAP
DB2_SKIPDELETED=NO
```

在另一种情况下，您可以设置 **DB2ENVLIST**，将聚集注册表变量 **DB2_WORKLOAD** 设为 SAP，并将 **DB2_SKIPDELETED** 注册表变量修改为 NO。当您发出 **db2set** 命令时，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显示聚集的名称（显示在方括号内）。**DB2_SKIPDELETED** 注册表变量显示 NO 值，并且 [O] 显示在其值旁边。

当您不再需要与 **DB2_WORKLOAD** 相关联的配置时，通过删除聚集注册表变量的值来删除组中每个注册表变量的隐式值。使用以下命令来删除 **DB2_WORKLOAD** 变量的值:

```
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。

删除聚集注册表变量的值时并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已删除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。要确定 **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，在启动和运行此产品前，您可能需要了解这些变量。

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
```

在执行 **db2start** 命令之前，必须设置您想要更新的注册表变量的值。

下表按类别列示了所有注册表变量。

表 121. 注册表变量和环境变量摘要. 第二列分割为两列。

变量类别	注册表或环境变量名	
常规	**DB2ACCOUNT** **DB2BIDI** **DB2_CAPTURE_LOCKTIMEOUT** **DB2CODEPAGE** **DB2_COLLECT_TS_REC_INFO** **DB2_CONNRETRIES_INTERVAL** **DB2CONSOLECP** **DB2DBDFT** **DB2DISCOVERYTIME** **DB2_ENFORCE_MEMBER_SYNTAX** **DB2FODC** **DB2_FORCE_APP_ON_MAX_LOG**	**DB2GRAPHICUNICODESERVER** **DB2INCLUDE** **DB2INSTDEF** **DB2INSTOWNER** **DB2_LIC_STAT_SIZE** **DB2LOCALE** **DB2_MAX_CLIENT_CONNRETRIES** **DB2_OBJECT_TABLE_ENTRIES** **DB2_SYSTEM_MONITOR_SETTINGS** **DB2TERRITORY** **DB2_VIEW_REOPT_VALUES**

表 121. 注册表变量和环境变量摘要 (续). 第二列分割为两列。

变量类别	注册表或环境变量名	
系统环境	DB2_ALTERNATE_GROUP_LOOKUP DB2CONNECT_ENABLE_EURO_CODEPAGE DB2CONNECT_IN_APP_PROCESS DB2_COPY_NAME DB2_CPU_BINDING DB2DBMSADDR DB2_DIAGPATH DB2DOMAINLIST DB2ENVLIST DB2INSTANCE DB2INSTPROF DB2LDAPSecurityConfig DB2LIBPATH DB2LOGINRESTRICTIONS	DB2NODE DB2OPTIONS DB2_PARALLEL_IO DB2PATH DB2_PMAP_COMPATIBILITY DB2PROCESSORS DB2RCMD_LEGACY_MODE DB2RESILIENCE DB2_RESTORE_GRANT_ADMIN_AUTHORITIES DB2SYSTEM DB2_UPDDBCFG_SINGLE_DBPARTITION DB2_USE_PAGE_CONTAINER_TAG DB2_WORKLOAD
通信	DB2CHECKCLIENTINTERVAL DB2COMM DB2FCMCOMM DB2_FORCE_NLS_CACHE DB2_PMODEL_SETTINGS DB2RSHCMD DB2RSHTIMEOUT	DB2SORCVBUF DB2SOSNDBUF DB2TCP_CLIENT_CONTIMEOUT DB2TCP_CLIENT_KEEPALIVE_TIMEOUT DB2TCP_CLIENT_RCVTIMEOUT DB2TCPCONNMGRS DB2TCP_SERVER_KEEPALIVE_TIMEOUT
命令行	DB2BQTIME DB2BQTRY DB2_CLP_EDITOR DB2_CLP_HISTSIZE	DB2_CLPPROMPT DB2IQTIME DB2RQTIME
分区数据库环境	DB2CHGPWD_EEE DB2_FCM_SETTINGS DB2_FORCE_OFFLINE_ADD_PARTITION	DB2_NUM_FAILOVER_NODES DB2_PARTITIONEDLOAD_DEFAULT DB2PORTRANGE
DB2 pureScale 环境	DB2_DATABASE_CF_MEMORY	DB2_MCR_RECOVERY_PARALLELISM_CAP
查询编译器	DB2_ANTIJOIN DB2_DEFERRED_PREPARE_SEMANTICS DB2_INLIST_TO_NLJN DB2_LIKE_VARCHAR DB2_MINIMIZE_LISTPREFETCH	DB2_NEW_CORR_SQ_FF DB2_OPT_MAX_TEMP_SIZE DB2_REDUCED_OPTIMIZATION DB2_SELECTIVITY DB2_SQLROUTINE_PREPOPTS

表 121. 注册表变量和环境变量摘要 (续). 第二列分割为两列。

变量类别	注册表或环境变量名	
性能	DB2_ALLOCATION_SIZE	DB2_MMAP_WRITE
	DB2_APM_PERFORMANCE	DB2_NO_FORK_CHECK
	DB2ASSUMEUPDATE	DB2NTMEMSIZE
	DB2_AVOID_PREFETCH	DB2NTNOCACHE
	DB2_BACKUP_USE_DIO	DB2NTPRICLASS
	DB2BPVARS	DB2NTWORKSET
	DB2CHKPTR	DB2_OVERRIDE_BPF
	DB2CHKSQLDA	DB2_PINNED_BP
	DB2_EVALUNCOMMITTED	DB2PRIORITIES
	DB2_EXTENDED_IO_FEATURES	DB2_RCT_FEATURES
	DB2_EXTENDED_OPTIMIZATION	DB2_RESOURCE_POLICY
	DB2_IO_PRIORITY_SETTING	DB2_SELUDI_COMM_BUFFER
	DB2_KEEP_AS_AND_DMS_CONTAINERS_OPEN	DB2_SET_MAX_CONTAINER_SIZE
	DB2_KEEPTABLELOCK	DB2_SKIPDELETED
	DB2_LARGE_PAGE_MEM	DB2_SKIPINSERTED
	DB2_LOGGER_NON_BUFFERED_IO	DB2_SMS_TRUNC_TMPTABLE_THRESH
	DB2MAXFSCRSEARCH	DB2_SORT_AFTER_TQ
	DB2_MAX_INACT_STMTS	DB2_SQLWORKSPACE_CACHE
	DB2_MAX_NON_TABLE_LOCKS	DB2_TRUSTED_BINDIN
	DB2_MDC_ROLLOUT	DB2_USE_ALTERNATE_PAGE_CLEANING
	DB2MEMDISCLAIM	DB2_USE_FAST_PREALLOCATION
	DB2_MEM_TUNING_RANGE	DB2_USE_IOCP
	DB2_MMAP_READ	

表 121. 注册表变量和环境变量摘要 (续). 第二列分割为两列。

变量类别	注册表或环境变量名	
其他	DB2ADMINSERVER	DB2_INDEX_PCTFREE_DEFAULT
	DB2_ATS_ENABLE	DB2LDAP_BASEDN
	DB2AUTH	DB2LDAPCACHE
	DB2_BCKP_INCLUDE_LOGS_WARNING	DB2LDAP_CLIENT_PROVIDER
	DB2_BCKP_PAGE_VALIDATION	DB2LDAPHOST
	DB2CLIINIPATH	DB2LDAP_KEEP_CONNECTION
	DB2_COMMIT_ON_EXIT	DB2LDAP_SEARCH_SCOPE
	DB2_COMMON_APP_DATA_PATH	DB2_LOAD_COPY_NO_OVERRIDE
	DB2_COMPATIBILITY_VECTOR	DB2_LIMIT_FENCED_GROUP
	DB2_CREATE_DB_ON_PATH	DB2LOADREC
	DB2_DDL_SOFT_INVAL	DB2LOCK_TO_RB
	DB2_DISABLE_FLUSH_LOG	DB2_MAX_LOB_BLOCK_SIZE
	DB2_DISPATCHER_PEEKTIMEOUT	DB2_MEMORY_PROTECT
	DB2_DJ_INI	DB2_MIN_IDLE_RESOURCES
	DB2_DMU_DEFAULT	DB2_NCHAR_SUPPORT
	DB2_DOCHOST	DB2NOEXITLIST
	DB2_DOCPORT	DB2_NUM_CKPW_DAEMONS
	DB2DSDRIVER_CFG_PATH	DB2_OPTSTATS_LOG
	DB2DSDRIVER_CLIENT_HOSTNAME	DB2REMOTEPREG
	DB2_ENABLE_AUTOCONFIG_DEFAULT	DB2_RESOLVE_CALL_CONFLICT
	DB2_ENABLE_LDAP	DB2_RESTRICT_DDF
	DB2_EVMON_EVENT_LIST_SIZE	DB2_SAS_SETTINGS
	DB2_EVMON_STMT_FILTER	DB2SATELLITEID
	DB2_EXTSECURITY	DB2_SERVER_CONTIMEOUT
	DB2_FALLBACK	DB2_SERVER_ENCALG
	DB2_FMP_COMM_HEAPSZ	DB2SORT
	DB2_GRP_LOOKUP	DB2_STANDBY_ISO
	DB2_HADR_BUF_SIZE	DB2STMM
	DB2_HADR_NO_IP_CHECK	DB2_TRUNCATE_REUSESTORAGE
	DB2_HADR_PEER_WAIT_LIMIT	DB2_UTIL_MSGPATH
	DB2_HADR_ROS	DB2_XBSA_LIBRARY
	DB2_HADR_SORCVBUF	DB2_XSLT_ALLOWED_PATH
	DB2_HADR_SOSNDBUF	
	DB2_HISTORY_FILTER	

常规注册表变量

您将发现了解 DB2 环境中的常规注册表变量（特别是如果您计划在将来变更这些变量）很有用。请注意，这里列出的某些注册表变量适用于特定的操作系统环境。

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 User's Guide。

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，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。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

要点：由于已有通过使用 CREATE EVENT MONITOR FOR LOCKING 语句来收集锁定超时事件的新方法，因此建议不要使用此变量，将来的发行版可能会将其除去。

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。

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。

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_ENFORCE_MEMBER_SYNTAX

- 操作系统：所有操作系统
- 缺省值：OFF，值：OFF 或 ON
- 此变量允许您控制是否检查 SQL 语句的语法、DB2 命令和 API 来了解是否正确使用了数据库分区关键字，从而确定是否必须改用 MEMBER 关键字。在 DB2 pureScale 环境中，缺省行为是容许使用特定于数据库分区的关键字，例如，DBPARTITIONNUM 或 DATABASE PARTITION，即使该操作的目标是DB2 成员。但是，如果 **DB2_ENFORCE_MEMBER_SYNTAX** 设为 ON，那么必须正确指定 MEMBER 关键字，否则会返回 SQL1538N。此变量的设置被忽略，并且在 DB2 pureScale 环境外部不起任何作用。

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** 注册表变量的设置用于控制 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，不将任何过滤操作推到包含此函数的表达式之后。这可能会致使不在表访问期间应用任何有可能导致表达式计算量过高、锁定过多以及查询性能不佳的谓词。

  在某些情况下，此行为过于严格，这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，那么此行为并不重要。例如，在行级别安全性实现的上下文中，通常存在格式如下的表达式：

  ```
  CASE WHEN <用于验证对此行的访问权的条件>
       THEN NULL
       ELSE RAISE_ERROR(...)
  END
  ```

  此应用程序可能只希望验证对此查询所选择的行的访问权，而不希望验证对表中每一行的访问权。因此，可以在基本表访问期间应用谓词，并且只需要在执行所有过滤操作之后执行包含 RAISE_ERROR 的表达式。在这种情况下，适当的值可能是 **DB2_EXPRESSION_RULES**=RAISE_ERROR_PERMIT_SKIP。

  另一种情况是列级安全性的上下文。在这种情况下，通常存在格式如下的表达式：

  ```
  CASE WHEN <用于验证对此行和此列的访问权的条件>
       THEN <table.column>
       ELSE RAISE_ERROR(...)
  END
  ```

  在这种情况下，此应用程序可能只希望在用户尝试接收特定行的数据并且列包含不允许用户检索的值时才引起错误。在这种情况下，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。

DB2FODC

- 操作系统：所有操作系统

- 缺省值: 所有 FODC 参数的并置（请参阅以下列表）
  - 对于 Linux 和 UNIX: ″CORELIMIT=*val* DUMPCORE=ON DUMPDIR=*diagpath*″
  - 对于 Windows: ″DUMPDIR=*diagpath*″

  请注意，参数之间用空格分隔。
- 此注册表变量控制在“首次出现数据收集”(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。

  COS

  - 操作系统: 所有操作系统
  - 缺省值: ON，值: ON 或 OFF
  - 此选项指定是否启用了 **db2cos** 脚本。可以将下列参数与此参数配合使用:

    COS_SLEEP

    - 缺省值: 3，值: 0 到无限大
    - 此选项指定两次检查所生成输出文件的大小之间休眠的时间量（以秒计）。

    COS_TIMEOUT

    - 缺省值: 30，值: 0 到无限大
    - 此选项指定在完成脚本之前要等待的时间量（以秒计）。

COS_COUNT

- 缺省值：255，值：0 到 255
- 此选项指定在数据库管理器陷阱期间执行 **db2cos** 的次数。

COS_SQLO_SIG_DUMP

- 缺省值：ON，值：ON 或 OFF
- 此选项指定在接收到 **SQLO_SIG_DUMP** 信号时是否启用了 **db2cos**。

DUMPCORE

– 操作系统：Linux、Solaris 和 AIX

– 缺省值：AUTO，值：AUTO、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。某些操作系统还有与核心转储相关的配置设置，这些设置可能会指定应用程序核心转储的行为。如果 **DB2RESILIENCE** 注册表变量设为 ON，那么 AUTO 设置将导致无法承受陷阱时生成核心文件。**DUMPCORE**=ON 设置将覆盖 **DB2RESILIENCE** 注册表变量设置，从而始终生成核心文件。

建议通过将 **DUMPCORE** 设为 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。

FODCPATH

– 操作系统：所有操作系统

– 缺省值：由 **DIAGPATH** 数据库管理器配置参数定义的路径，值：*fodc_path_name*

– 此选项指定要将 FODC 程序包定向至的绝对路径名。*fodc_path_name* 必须是一个现有目录，并且为其设置此选项的一个或多个成员以及正在这些成员上运行的 fmp 进程必须可写入此目录。

SERVICELEVEL

– 操作系统：所有操作系统

– 缺省值：AUTOMATIC ulimit 设置，值：AUTOMATIC、BASIC 或 FULL

– 此选项指定在发生紧急情况、陷阱或错误（可能指示数据损坏）期间如何收集数据。DB2 被设计为生成适合配置和问题环境的诊断。例如，如果可承受陷阱，那么只生成最低级别的基本诊断以回滚事务并尽快响应该应用程序，释放其他应用程序可能正在等待的资源。如果不可承受陷阱，那么可限制诊断（例如，db2cos

数据收集脚本和核心转储）并转而在 DB2 pureScale 配置中提供。生成诊断的缺省行为由 SERVICELEVEL 设置 AUTOMATIC 表示。

此参数支持以下选项:

AUTOMATIC
此设置指定分别在运行时为成员以及在启动时为 CF 进程选择生效的 SERVICELEVEL 设置（即: BASIC 或 FULL）。当前，仅当 DB2 pureScale 环境有多个成员时以及存在陷阱弹性时，才会选择 BASIC。

BASIC 此 SERVICELEVEL 设置指定要转储最少量的 FODC 数据。缺省情况下，核心转储处理被禁用（但可被 COREDUMP 设置覆盖），诊断被限制为仅针对受影响线程，并且调出脚本被禁用。

FULL 此 SERVICELEVEL 设置指定要转储最大数量的 FODC 数据。这包括核心转储、任何相关联的组件转储和调出脚本的调用。此外，不会尝试承受陷阱。

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** 设为 TRUE，并且发出带有 **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。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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 **DB 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。请参阅开发嵌入式 SQL 应用程序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。零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_MAX_GLOBAL_SNAPSHOT_SIZE

- 操作系统: 所有操作系统
- 缺省值: 未设置, 值: 0 到快照的最大大小。

- 此变量指定快照或快照估计值可以达到的字节数。可以使用此变量来防止大型全局快照导致内存使用峰值，这会导致性能下降和系统挂起。

  缺省情况下，未设置 **DB2_MAX_GLOBAL_SNAPSHOT_SIZE**，这意味着对快照的最大大小实行有效限制（2 GB 减去 512 个字节）。如果需要设置此变量，那么建议的起始点为分配给快速通信管理器 (FCM) 缓冲区的内存的 25%。此变量是动态的，且仅适用于分区数据库环境。

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。通过保留连续的存储器，可以降低需要更新元数据条目的操作（例如 CREATE INDEX 和 **IMPORT REPLACE**）被联机备份阻塞的机率。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
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。
- 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
  ```

  OLD_CPU_USAGE 将恢复为其缺省设置。

  注：当前，此注册表变量只具有适用于 Linux 的设置；在将来的发行版中将添加适用于其他操作系统的其他设置。
- 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 设置将加以忽略 - 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。当此变量设为 NO 时，只允许 DBADM 将这些值保存在 EXPLAIN_PREDICATE 表中。

系统环境变量

使用系统环境变量将配置值传递至 DB2 环境中正在运行的应用程序。请注意，列出的某些注册表变量适用于特定的操作系统环境。

DB2_ALTERNATE_GROUP_LOOKUP

- 操作系统：AIX 和 Linux
- 缺省值：NULL，值：NULL、GETGRSET（在 AIX 上）或 GETGROUPLIST（在 Linux 上）
- 此变量允许 DB2 数据库系统从操作系统提供的备用源中获取组信息。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
- 在所有连接到需要支持欧元符号的 DB2 z/OS 版服务器或 DB2 IBM i 版服务器的 DB2 Connect 客户机和服务器上，请将此变量设为 YES。如果将此变量设为 YES，那么当前应用程序代码页将映射到明确指示支持欧元符号的等效编码字符集标识 (CCSID)。因此，DB2 Connect 将使用特定 CCSID 来连接到 DB2 z/OS 版服务器或 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，如果客户机正在使用映射到 CCSID 1252 的代码页，那么该客户机将使用 CCSID 5348 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，那么无法使用 **DB2_COPY_NAME** 来切换到另一个 DB2 副本，而必须运行 *INSTALLPATH*\bin\db2envar.bat 命令才能切换当前正在使用的副本，其中 *INSTALLPATH* 是 DB2 副本的安装位置。

DB2_CPU_BINDING

- 操作系统：Linux
- 缺省值：
  - 如果 DB2 成员和集群高速缓存设施 (CF) 在同一主机上：
    - 对于成员，NUM_CORES=max(1, floor(0.8*totalCores))
    - 对于集群高速缓存设施，NUM_CORES=totalCores - 上面列示的数目。
  - 如果 DB2 成员和集群高速缓存设施未共用主机，那么不会设置此变量
- 此注册表变量控制 CPU 分配。要使对此变量的更改生效，您需要重新启动 DB2 实例。

  参数：

  NUM_CORES

  - 操作系统：Linux
  - 缺省值：如果成员或 CF 在同一主机上，那么总可用核心数的大约 80% 指定给 DB2，余下部分指定给 CF。值：0 < x <（主机上的物理核心数）
  - 此选项指定分配给成员或 CF 进程的核心数。可使用 NUM_CORES 来配置 DB2 产品的次级容量许可。核心数可以是整数或分数，这允许您在启用了同时多线程 (SMT) 的情况下添加一个或多个硬件线程。

  PROCESSOR_LIST

– 操作系统: Linux
– 缺省值: 未设置，值: 任何处理器数
– 此选项指定 DB2 将绑定至的逻辑处理器数，这允许您完全控制逻辑处理器（或核心）数以及它们将驻留的 CPU 包（或套接字）。如果尝试使用 **DB2_CPU_BINDING** 设置 PROCESSOR_LIST 和 NUM_CORES，那么 NUM_CORES 会被忽略

轻量级重新启动注意事项

如果成员作为主机上的访客成员重新启动，并且该主机上已有成员在运行，那么轻量级重新启动成员将分配至已被常驻成员占用的核心（最多为 **DB2_CPU_BINDING** 指定的核心数）。如果成员作为主机上的访客成员重新启动，并且该主机的核心数少于 **DB2_CPU_BINDING** 指定的核心数，那么成员将绑定至主机上的核心数。

每次设置 **DB2_CPU_BINDING** 时，系统会在实例级别概要文件中清除未显式设置的任何参数。请将每个参数及其值括在引号中，如以下示例所示。

示例 1

用户想要将DB2 实例 db2inst1 的第一个成员成员（其标识为 0）分配至有 2 个核心的主机上的 1 个核心:

```
db2set -i db2inst1 0 DB2_CPU_BINDING="NUM_CORES=1"
```

示例 2

用户想要将 db2inst1 上的所有成员分配至有 8 个核心且启用了 Intel HTT（意味着该主机有 16 个处理器）的主机上的 5 个逻辑处理器:

```
db2set -i db2inst1 DB2_CPU_BINDING="NUM_CORES=2.5"
```

示例 3

用户要指定主CF（其标识为 128）将绑定至多少个核心:

```
db2set -i db2inst1 128 DB2_CPU_BINDING="NUM_CORES=4"
```

示例 4

用户想要将成员 0 上的 DB2 实例 db2inst1 绑定至特定逻辑处理器组:

```
db2set -i db2inst1 0 DB2_CPU_BINDING="PROCESSOR_LIST=2,10,6,14"
```

DB2DBMSADDR

- 操作系统: Linux on x86、Linux on zSeries（31 位）和 Windows 32 位
- 缺省值: NULL（在 Linux 操作系统上）或0x20000000（在 Windows 操作系统上），值: 0x09000000 到 0xB0000000 范围内的虚拟地址，增量为 0x10000（在 Linux 操作系统上）或0x20000000 到 0xB0000000 范围内的虚拟地址，增量为 0x10000（在 Windows 操作系统上）
- **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。

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。

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，可使用以下命令将 **DB2DBMSADDR** 注册表变量重置为 NULL:

```
db2set DB2DBMSADDR=
```

注: 在更改此变量的设置之前，必须停止该实例和所有 DB2 进程。如果在设置此变量时该实例正在运行，那么任何后续的 **db2stop** 命令都将失败。

DB2_DIAGPATH

- 操作系统: 所有操作系统
- 缺省值: 缺省值是 UNIX 和 Linux 操作系统上的实例 db2dump 目录以及 Windows 操作系统上的实例 db2 目录。
- 此参数仅适用于 ODBC 和 CLI 应用程序。

此参数允许您指定 DB2 诊断信息的标准路径。根据您的平台，此目录可能包含转储文件、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，各个域名之间由逗号（“,”）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。

如果客户机或服务器正在 Windows 环境中运行，那么支持 **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: **DB2INSTDEF**（Windows 32 位操作系统）。
- 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: 不能使用 **db2set** 命令来更新此注册表变量。有关更多信息，请参阅第 497 页的『标识当前实例』和第 496 页的『在 Windows 上，在概要文件注册表外部设置环境变量』。

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。实例目录和其他用户数据文件不能在 sqllib 目录中。

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 V9.1 只通过强制实施 **su** 限制来验证是否允许使用 **su** 命令，以及验证当前进程是否具有可通过调用 **su** 命令来切换到帐户的组标识。
  - NONE

    DB2 不强制实施任何登录限制。
  - LOCAL（或者不设置此变量）

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：不能使用 **db2set** 命令来更新此注册表变量。有关更多信息，请参阅第 496 页的『在 Windows 上，在概要文件注册表外部设置环境变量』。

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 或 *（在 DB2 pureScale 环境中），值：*TablespaceID*:[*n*],... - 已定义的表空间（由其数字表空间标识识别）的逗号分隔列表。如果表空间的预取大小设为 AUTOMATIC，那么可以通过指定表空间标识，后面跟一个冒号，再加上每个容器的磁盘数 *n*，向 DB2 数据库管理器指示该表空间中每个容器的磁盘数。如果未指定 *n*，那么将使用缺省值 6。

  可以将 *TablespaceID* 替换为星号 (*) 以指定所有表空间。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。如果启用了 I/O 并行性（通过使用多个容器来隐式启用，或者通过设置 **DB2_PARALLEL_IO** 来显式启用），那么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。如果设置了此注册表变量，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。下表概述了可用的不同选项和在每种情况下计算并行性的方式:

表 122. 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*
非 AUTOMATIC	未设置	容器数
非 AUTOMATIC	*表空间标识*	预取大小/扩展数据块大小
非 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: 随操作系统的不同而有所变化
- 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。

DB2_PMAP_COMPATIBILITY

- 操作系统: 所有操作系统
- 缺省值: ON，值: ON 或 OFF

- 此变量允许用户继续使用 sqlugtpi 和 sqlugrpn API 来分别返回某一行的表以及数据库分区号和数据库分区服务器号的分布信息。缺省设置 ON 表示分布映射大小仍为 4096 个条目（V9.7 以前的行为）。如果此变量设为 OFF，那么新数据库或已升级数据库的分布映射大小将增大为 32768 个条目（V9.7 的行为）。如果使用 32K 分布映射，那么需要使用新的 db2GetDistMap 和 db2GetRowPartNum API。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

DB2PROCESSORS

- 操作系统：Windows
- 缺省值：NULL，值：0-*n*-1（其中，*n* 为处理器数）
- 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** 进程将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。

DB2RESILIENCE

- 操作系统：所有操作系统
- 缺省值：ON，值：ON（TRUE 或 1）或者 OFF（FALSE 或 0）
- 此注册表变量可用于控制是否容忍物理读错误以及是否激活扩展陷阱恢复功能。缺省行为是，容忍读错误并激活扩展陷阱恢复功能。要恢复前发行版的行为并强制数据库管理器关闭实例，请将此注册表变量设为 OFF。此注册表变量不会影响现有的存储密钥支持。

DB2_RESTORE_GRANT_ADMIN_AUTHORITIES

- 操作系统：所有操作系统
- 缺省值：OFF，值：ON 或 OFF
- 如果 **DB2_RESTORE_GRANT_ADMIN_AUTHORITIES** 设为 ON，并且您正在复原到新的或现有数据库，那么将对您授予 SECADM、DBADM、DATAACCESS 和 ACCESSCTRL 权限。

- 当 **DB2_RESTORE_GRANT_ADMIN_AUTHORITIES** 设为 ON 时，支持下列复原方法:
  - 分割镜像备份
  - ACS 快照备份
  - 使用 **RESTORE DATABASE** 命令进行联机和脱机数据库备份

  注: 注意，此变量对表空间复原不起作用；不会将任何其他权限授予发出复原操作的用户。
- 如果 **DB2_WORKLOAD** 设为 SAP，那么 **DB2_RESTORE_GRANT_ADMIN_AUTHORITIES** 将设为 ON。

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
- **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。

  当此注册表变量设为 1、TRUE 或 YES 时，它允许您指定对数据库所进行的任何更新或重置仅影响特定分区。如果未设置变量（缺省值），那么在未指定分区的情况下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。

注: 此变量不适用于通过调用 ADMIN_CMD 例程发出的更新或重置请求。

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** 设为 ON。

  但是，如果在对容器使用 RAID 设备时将此注册表变量设为 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 V8 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 如果要在数据库中为 1C 应用程序配置一组注册表变量，请使用此设置。

  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。

  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。

  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。

  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。

  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。

  MDM 当您想在数据库中为 Master Data Management 配置一组注册表变量时使用此设置。

  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
  – 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 当您要在数据库中为 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。

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。

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。

通信变量

使用通信变量来帮助控制 DB2 网络连接上以及 DB2 环境自身内部的数据流。

DB2CHECKCLIENTINTERVAL

- 操作系统: 所有操作系统，仅适用于服务器
- 缺省值: 100，值: 大于或等于零的数字值。
- 此变量指定活动事务期间 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耗用的时间。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。

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。可以将此变量设为值的任何组合（用逗号来分隔值）。

  注: 值 NPIPE 仅在 Windows 操作系统中有效。

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。

DB2_PMODEL_SETTINGS

- 操作系统: 所有操作系统
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。
- 此注册表变量控制一组参数，这些参数允许您修改 DB2 内部基础结构的各个方面的行为。使用分号分隔参数，如以下示例中所示:

  ```
  db2set
  DB2_PMODEL_SETTINGS=MLN_REMOTE_LISTENER:TRUE;ENHANCED_ROLLBACK:TRUE;
  SRVLST_EQUAL_WEIGHT:TRUE
  ```

- 参数:

  ENHANCED_ROLLBACK

  - 缺省值: FALSE
  - 值: TRUE 或 FALSE
  - 使用 **ENHANCED_ROLLBACK** 参数来改进分区数据库环境中 DB2 服务器的工作单元回滚行为。如果您将此选项设为 TRUE，那么只会将工作单元的回滚请求发送至参与事务的逻辑数据库分区。

  MLN_REMOTE_LISTENER

- 缺省值：FALSE
- 值：TRUE 或 FALSE
- 使用 **MLN_REMOTE_LISTENER** 参数对每个逻辑数据库分区启动 TCP/IP 侦听器。如果您将此选项设为 TRUE，那么应用程序可以直接连接至每个逻辑数据库分区，而不通过分配给逻辑端口 0 的数据库分区服务器来传递请求。

  如果您将此选项设为 TRUE，请确保其他 TCP/IP 侦听器不使用其他服务所需要的端口。

SRVLST_EQUAL_WEIGHT

- 缺省值：FALSE
- 值：TRUE 或 FALSE
- 如果希望服务器列表中的非零成员权重始终完全相同（从而覆盖根据负载来计算成员权重的缺省行为），请使用 **SRVLST_EQUAL_WEIGHT** 参数。如果启用了工作负载均衡 (WLB)，那么远程客户机使用包含在服务器列表中的成员权重来分配工作负载。

  如果将此选项设为 TRUE，那么客户机上的 WLB 会转换为在成员间平均分配工作负载而不考虑成员负载。

DB2RSHCMD

- 操作系统：UNIX 和 Linux
- 缺省值：rsh（在 HP-UX 上为 remsh），值：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、**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单分区环境，以及能够启动、停止或监视 DB2 实例（例如，**db2gcf**）的 rsh 或 ssh 从属实用程序，才需要此变量。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** 注册表变量以将 ssh shell 与 DB2 配合使用的详细指示信息，请参阅“配置 DB2 Universal Database™ for UNIX 以使用 OpenSSH”一文。

DB2RSHTIMEOUT

- 操作系统：UNIX 和 Linux
- 缺省值：30 秒，值：1 - 120
- 仅当 **DB2RSHCMD** 设为非空值时，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: 65 536
- 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: 65 536
- 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 - 32 767 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。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

DB2TCP_CLIENT_KEEPALIVE_TIMEOUT

- 操作系统: AIX、HP-UX、Linux, Windows（仅适用于客户机）
- 缺省值: 15, 值: 0 至 32767 秒
- **DB2TCP_CLIENT_KEEPALIVE_TIMEOUT** 注册表变量指定在检测到无响应的 TCP/IP 客户机连接或附件不再处于活动状态之前经过的最长时间（按秒计）。它是相当于 **DB2TCP_SERVER_KEEPALIVE_TIMEOUT** 的客户端。如果未设置此变量, 那么会使用为 15 秒的缺省设置。如果设置此变量, 那么此变量会优先于 db2dsdriver.cfg 文件中指定的任何 keepAliveTimeout 设置。

  对此变量的更改针对所有将此服务器的 TCP/IP 连接和附件立即生效

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 - 32 767 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。

- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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向上舍入为最大连接管理器数 (16)。值: 1 到 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: 将连接管理器数设为 1 会导致具有许多用户的系统中的远程连接的性能下降、频繁连接和/或断开连接。

DB2TCP_SERVER_KEEPALIVE_TIMEOUT

- 操作系统: AIX、HP-UX、Linux, Windows（仅适用于服务器）
- 缺省值: 60，值: 0 至 32767 秒
-

  DB2TCP_SERVER_KEEPALIVE_TIMEOUT 注册表变量指定在检测到无响应的 TCP/IP 客户机连接或附件不再处于活动状态之前经过的最长时间（按秒计）。它是相当于 **DB2TCP_CLIENT_KEEPALIVE_TIMEOUT** 和 keepAliveTimeout 的服务器端。如果未设置此变量，那么会使用为 60 秒的缺省设置。

  对此变量的更改针对所有将此服务器的 TCP/IP 连接和附件立即生效。不需要重新启动该服务器实例。

命令行变量

可以设置命令行变量以控制 DB2 产品环境中命令行的缺省行为。

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: 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为缺省值。如果未设置此注册表变量或者将其设为有效范围以外的值，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 且包含零个或零个以上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。该变量可设为长度小于 100 个字符且包含零个或零个以上可选标记 %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。（仅适用于 Windows 平台）如果未设置 **DB2INSTANCE**，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

使用分区数据库环境变量来控制分区数据库环境的缺省行为, 包括授权、故障转移和网络行为。

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: YES, 值:
  - FCM_MAXIMIZE_SET_SIZE:[YES|TRUE|NO|FALSE]。 FCM_MAXIMIZE_SET_SIZE 的缺省值为 YES。
  - FCM_CFG_BASE_AS_FLOOR: [YES|TRUE|NO|FALSE]。FCM_CFG_BASE_AS_FLOOR 的缺省值为 NO。
- 可以使用 FCM_MAXIMIZE_SET_SIZE 标记来设置 **DB2_FCM_SETTINGS** 注册表变量, 以便为快速通信管理器 (FCM) 缓冲区预先分配缺省的 4 GB 空间。此标记的值必须是 YES 或 TRUE 才能启用此功能。

  可将 FCM_CFG_BASE_AS_FLOOR 选项与 **DB2_FCM_SETTINGS** 注册表变量配合使用, 以将基值设为 *fcm_num_buffers* 和 *fcm_num_channels* 数据库管理器配置参数的最低值。当 FCM_CFG_BASE_AS_FLOOR 选项设为 YES 或 TRUE, 且这些参数设为 AUTOMATIC 并具有初始或起始值时, DB2 不会将这些参数调整为低于此值。

DB2_FORCE_OFFLINE_ADD_PARTITION

- 操作系统: 所有操作系统
- 缺省值: FALSE; 值: FALSE 或 TRUE

- 此变量允许您指定以脱机方式执行添加数据库分区服务器操作。缺省设置 FALSE 表明不必使数据库进入脱机状态即可添加 DB2 数据库分区服务器。但是，如果要以脱机方式执行此操作，或者某些限制导致无法在数据库处于联机状态时添加数据库分区服务器，请将 **DB2_FORCE_OFFLINE_ADD_PARTITION** 设为 TRUE。如果此变量设为 TRUE，那么将按照版本 9.5 和更低版本的行为来添加新的 DB2 数据库分区服务器；即，新数据库分区服务器直到实例关闭并重新启动之后才对该实例可见。

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 上都设为 2。

  执行 START DBM 时，FCM 将在机器 A 和机器 B 上保留足够的内存，以便管理多达四个数据库分区，这样在一台机器发生故障时，可以在另一台机器上重新启动故障机器上的那两个数据库分区。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 实用程序当前连接指的数据库分区上尝试装入。

  注：建议您不要使用此变量，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其他数据库分区也有相同的端口范围。

DB2 pureScale 环境变量

可以为 DB2 pureScale 系统设置两个环境变量。

DB2_DATABASE_CF_MEMORY

- 操作系统: 所有操作系统
- 缺省值: 100, 值: -1 或 0 到 100
- 类型: 浮点
- 此变量专门用于 DB2 pureScale 环境。
- **DB2_DATABASE_CF_MEMORY** 用于指示将指定给其 **cf_db_mem_sz** 数据库配置参数设为 AUTOMATIC 的每个数据库的总 CF 内存 (**CF_MEM_SZ**) 比例。**cf_db_mem_sz** 设为特定值的任何数据库将忽略此注册表变量。
- 如果超过 100 个数据库处于活动状态时所有数据库要具有相等份额的 CF 内存资源, 那么 **DB2_DATABASE_CF_MEMORY** 注册表变量必须使用小于 1 的值。例如, 如果有 200 个处于活动状态的数据库, 并且每个数据库具有相等份额的 CF 内存, 那么此注册表变量必须设为 0.5。
- 必须将 **DB2_DATABASE_CF_MEMORY** 注册表变量与 **cf_db_mem_sz** 和 **numdb** 配置参数协调使用。有关其他信息, 请参阅 DB2 pureScale CF 内存参数配置。

DB2_MCR_RECOVERY_PARALLELISM_CAP

- 操作系统: 所有操作系统
- 此变量专门用于 DB2 pureScale 环境。
- 在多个数据库环境中, 如果需要成员崩溃恢复 (MCR), 那么在每个成员上并行恢复的数据库数由 **numdb** 配置参数或 **DB2_MCR_RECOVERY_PARALLELISM_CAP** 注册表变量的值（两者中的较小者）设置。

查询编译器变量

使用查询编译器变量来控制 DB2 查询优化过程中的查询编译器优化决策。这些决策包括防止特定的优化决策并强制执行特定的 SQL 查询操作。

DB2_ANTIJOIN

- 操作系统: 所有操作系统
- 在 ESE 环境中, 缺省值为 NO; 在非 ESE 环境中, 缺省值为 EXTEND。值: YES、NO 或 EXTEND
- 如果此变量设为 YES, 那么优化器会搜索将 NOT EXISTS 子查询变换为 DB2 可更有效地处理的反连接的机会。

  如果此变量设为 NO, 那么优化器会限制将 NOT EXISTS 子查询变换为反连接的机会。

  如果此变量设为 EXTEND, 那么优化器会搜索将 NOT IN 和 NOT EXISTS 子查询变换为反连接的机会。

  如果带有 **-immediate** 参数发出 **db2set** 命令, 那么对此变量的更改会在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例。

DB2_DEFERRED_PREPARE_SEMANTICS

- 操作系统: 所有操作系统

- 缺省值: NO, 值: YES 或 NO
- 如果将此注册表变量设为 YES, 那么将启用延迟准备语义, 以使 PREPARE 语句中使用的所有隐式类型参数标记都根据与后续 OPEN 或 EXECUTE 语句相关联的输入描述符来派生其数据类型和长度属性。与以前的情况相比, 这样可以在更多位置使用隐式类型参数标记。

  注: 将 **DB2_DEFERRED_PREPARE_SEMANTICS** 设为 YES 可能会导致非预期的影响或结果。如果输入描述符中的数据类型与使用"确定隐式类型表达式的数据类型"的规则所派生的数据类型不同, 那么可能会发生下列情况:

  – 由于增加了强制类型转换操作, 使得查询性能降低。

  – 由于无法转换数据类型而使得查询失败。

  – 查询可能会返回不同的结果。

  例如, 假定表 t1 中有 char_col 列, 此列被定义为 VARCHAR(10) 并且具有值"1"、"100"、"200"和"xxx"。用户运行下列查询:

  ```
  select * from t1 where char_col = ?
  ```

  如果输入参数的数据类型为 INTEGER, 并且正在使用延迟准备, 那么会将 char_col 列强制转换为数字类型。但是, 由于此表中的其中一行包含非数字数据 ("xxx"), 无法转换为数字值, 因此查询失败。

  如果设为 YES_DBCS_GRAPHIC_TO_CHAR, 那么此注册表变量会指定参数标记要使用 VARCHAR 类型而不是 VARGRAPHIC 类型。如果符合下列所有条件, 那么 **DB2_DEFERRED_PREPARE_SEMANTICS** 注册表变量隐式具有此设置:

  – 未设置 **DB2_DEFERRED_PREPARE_SEMANTICS** (即, 设为 NULL)。

  – **The DB2_COMPATIBILITY_VECTOR** 注册表变量设为 ORA、MYS 或 MSS。

  – 您处于双字节字符集 (DBCS) 环境中。

  必须在发出 **db2start** 命令之前设置 **DB2_DEFERRED_PREPARE_SEMANTICS** 注册表变量。

  仅建议对 Unicode 数据库和 SBCS 数据库使用此注册表变量。

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, 在下列查询中:

  ```
  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** 设为 YES 时，它们都保持为活动状态，即使指定了 REOPT(ONCE) 亦如此。

如果带有 **-immediate** 参数发出 **db2set** 命令，那么对此变量的更改会在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例。

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。

  要点：建议不要使用此变量，将来的发行版可能会将其除去，因为您仅应在 IBM 服务人员的建议下更改这些设置。

  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，估算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。将可变长度子元素算法与算法参数一起使用。
    - N - 使用固定长度子元素算法。
    - num1 - 将 num1 的值与可变长度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。使用可变长度子元素算法，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 - 使用可变长度子元素算法，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。

如果带有 **-immediate** 参数发出 **db2set** 命令，那么对此变量的更改会在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例。

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** 设为 YES 时，它们都保持为活动状态，即使指定了 REOPT(ONCE) 亦如此。

  如果带有 **-immediate** 参数发出 **db2set** 命令，那么对此变量的更改会在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例。

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 ...)
  ```

  如果带有 **-immediate** 参数发出 **db2set** 命令，那么对此变量的更改会在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例。

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

如果带有 **-immediate** 参数发出 **db2set** 命令，那么对此变量的更改会在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例。

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、JUMPSCAN、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级别来使用优化功能。如果减少所使用的优化技巧的数目，那么还将减少优化期间耗用的时间和资源。

  如果设置此变量，那么以下语法规则适用:

  – 用逗号 (,) 分隔每个选项，并确保逗号前后没有空格。

  – 用单个空格分隔选项与其值。

  – 如果设置包括空格，请将该设置括在双引号（“”）中。

  以下示例显示正确语法:

  ```
  db2set DB2_REDUCED_OPTIMIZATION="NO_SORT_NLJOIN,JUMPSCAN ON"
  ```

  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存取方案的风险。仅当 IBM 或其合作伙伴建议时，才使用此注册表变量。

  – 如果设为 NO

    优化器将不更改它的优化技术。

  – 如果设为 YES

    如果优化级别为 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存取方案。

    如果优化级别恰好是 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存取方案的风险。对于小于 5 的优化级别，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，那么它们将保持有效。

  – 如果设为任何整数

    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 优化级别所述。这意味将以与优化级别 2 类似的级别优化查询。

  – 如果设为 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级别 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。

– 如果设为 JUMPSCAN

使用此选项来控制 DB2 优化器能否使用跳跃扫描操作。可指定以下值:

- OFF = DB2 优化器不创建使用跳跃扫描的方案。
- ON = DB2 优化器使用基于成本的分析来确定是否生成使用跳跃扫描的方案（缺省值）。

– 如果设为 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：优化级别 5 的动态优化减少优先于对于在 **DB2_REDUCED_OPTIMIZATION** 设为 YES 时恰好是 5 的优化级别所描述的行为以及整数设置所描述的行为。

– 如果设为 ZZJN，请执行以下操作:

使用此选项来控制 DB2 优化器如何对包含一个事实表的基于星型模式的查询使用 zigzag 连接方法。可指定以下值:

- OFF 表示 DB2 优化器不使用 zigzag 连接方法。
- ON 表示 DB2 优化器使用基于成本的分析来确定是使用 zigzag 连接方法还是另一连接方法（缺省）。
- FORCE 表示如果 zigzag 连接方法可行，那么 DB2 优化器使用 zigzag 连接方法。

– 如果设为 ZZJN_MULTI_FACT，请执行以下操作:

使用此选项来控制 DB2 优化器如何对包含多个事实表的基于星型模式的查询使用 zigzag 连接方法。可指定以下值:

- OFF 表示 DB2 优化器不使用 zigzag 连接方法。
- ON 表示 DB2 优化器使用基于成本的分析来确定是使用 zigzag 连接方法还是另一连接方法（缺省）。
- FORCE 表示如果 zigzag 连接方法可行，那么 DB2 优化器使用 zigzag 连接方法。

如果带有 **-immediate** 参数发出 **db2set** 命令，那么对此变量的更改会在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例。

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。

  如果此注册表变量设为 NO，那么只能在用户定义的谓词中指定 SELECTIVITY 子句。

  当此注册表变量设为 YES 时，可以为下列谓词指定 SELECTIVITY 子句:
  - 用户定义的谓词
  - 至少一个表达式包含主变量或参数标记的基本谓词

  如果带有 **-immediate** 参数发出 **db2set** 命令，那么对此变量的更改会在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例。

DB2_SQLROUTINE_PREPOPTS

- 操作系统: 所有操作系统
- 缺省值: 空字符串，值:
  - APREUSE {YES | NO}
  - BLOCKING {UNAMBIG | ALL | NO}
  - CONCURRENTACCESSRESOLUTION { USE CURRENTLY COMMITTED | WAIT FOR OUTCOME }
  - 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}
  - STATICREADONLY {YES|NO|INSENSITIVE}
  - VALIDATE {RUN | BIND}
- **DB2_SQLROUTINE_PREPOPTS** 注册表变量可以用来定制 SQL 和 XQuery 过程及函数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

可以设置这些变量来改进 DB2 产品的性能。采用此方式，通过指定内存调整操作和操作系统资源策略，可以影响的性能改进包括存取方案优化。

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：建议不要使用 **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
- 将此变量设为 ON 来启用存取方案管理器 (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。NO PACKAGE LOCK 方式可能会使性能有较小的提高，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 语句中提供的所有固定长度列。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_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。

DB2_BACKUP_USE_DIO

- 操作系统: 所有操作系统
- 缺省值: OFF, 值: ON 或 OFF
- 指定操作系统是否对备份映像进行高速缓存。缺省行为是对该映像文件进行高速缓存。如果 **DB2_BACKUP_USE_DIO** 设为 ON, 那么备份映像文件直接写至磁盘（绕过文件高速缓存）。

  将此变量设为 ON 可能会让操作系统更好地利用内存资源, 因为对备份映像文件进行高速缓存没有任何好处。对于 Linux 平台, 此性能影响带来的好处最大。但是, 备份本身的速度可能会稍微减慢, 所以应在 **DB2_BACKUP_USE_DIO** 设为 ON 时测量备份性能的变化。

  注: 更改此注册表变量的值不会影响已在运行的备份操作的行为。对该值的更改将在下一次运行备份时生效, 并且不需要实例重新启动。

DB2BPVARS

- 操作系统: 对每个参数指定的操作系统
- Default=Path
- 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。

  要点: V9.5 中已经不推荐使用此性能变量, 在以后的发行版中可能会将其除去。有关更多信息, 请参阅。

  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 的文件可能包含以下行:

  ```
  NO_NT_SCATTER = 1
  ```

  假定 bpvars.vars 存储在 F:\vars\ 中, 要设置这些变量, 请执行以下命令:

  ```
  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。

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，它将进行扫描以延迟或避免行锁定，直到知道数据满足谓词求值为止。当启用此变量时，可对未落实的数据进行谓词求值。

  只有在“当前已落实”语义对避免锁定争用没有帮助时，**DB2_EVALUNCOMMITTED** 才适用。如果设置了此变量，并且“当前已落实”语义适用于扫描，那么将不会跳过已删除的行，并且不会对尚未落实的数据进行谓词求值；而是，将处理行和数据的当前已落实版本。

  并且，**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。并且，执行表扫描访问时将无条件地跳过已删除的行，执行索引扫描时不会跳过已删除键，除非还设置了注册表变量 **DB2_SKIPDELETED**。

  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 和 OFF
- 将此变量设为 ON 可启用增强 I/O 性能的功能。此增强功能包括改进内存高速缓存的命中率和缩短高优先级 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）
  - 存储子系统: 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。

  HIGH、MEDIUM 和 LOW 的缺省 I/O 优先级设置分别为 3、8 和 12；可使用 **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、ENHANCED_MULTIPLE_DISTINCT 和 IXOR 值指定不同的优化扩展功能。如果要使用多个值，请使用逗号分隔列表。

缺省行为（由 OFF 或 IXOR 值指定）用于优化器扩展索引 ORing 数据访问方法以包括引用任意索引列的 OR 谓词，即使出现非索引列谓词也是如此。例如，考虑下面这两个索引定义:

```
INDEX IX2:  dept    ASC
INDEX IX3:  job     ASC
```

如果设置 IXOR 选项，那么通过使用这两个索引，可以满足下列谓词:

```
WHERE
    dept = :hv1 OR
    (job = :hv2 AND
    years >= :hv3)
```

通常，**DB2_EXTENDED_OPTIMIZATION** 设置可能不会在所有环境中提高查询性能。您应执行测试，以确定各个查询的性能提高情况。

要点:

- 建议不要在 V10.1 中使用 ENHANCED_MULTIPLE_DISTINCT 和 IXOR 值，将来的发行版可能会将其除去。除去 ENHANCED_MULTIPLE_DISTINCT 提供了可改进多个不同查询的性能的新增强功能。IXOR 值是冗余值，因为它指定该缺省行为。有关更多详细信息，请参阅*DB2 V10.1 新增内容*中的『带有已更改行为的注册表变量』。
- 仅当上次启动实例时启用了 ENHANCED_MULTIPLE_DISTINCT 值时，此值才会动态生效。

DB2_IO_PRIORITY_SETTING

- 操作系统: AIX
- 值: HIGH:#、MEDIUM:# 和 LOW:#，其中 # 可以是 1 到 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。必须在启动实例之前设置此注册表变量；对此变量进行任何修改都需要重新启动实例。请注意，单独设置此注册表变量并不会启用增强的 I/O 功能，还必须设置 **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
- 如果将此变量设为 ON，那么每个 DMS 表空间容器都将打开一个文件句柄，直到数据库被取消激活为止。因为免除了打开容器的开销，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

- 如果将此变量设为 ON 或 TRANSACTION，那么此变量允许 DB2 数据库系统在“未落实的读”或“游标稳定性”隔离级别处于关闭状态期间维护表锁定。事务结束时将释放所保留的表锁定，就像为“读稳定性”和“可重复读”扫描释放表锁定一样。

  如果将此变量设为 CONNECTION，那么将为应用程序释放一个表锁定，直到该应用程序回滚事务或者连接被重置为止。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。然而，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 设置还具有上面描述的 ON 或 TRANSACTION 设置所具有的行为。

  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 * 来表示应使用大页内存的所有适用内存区域，或应使用大页内存的特定内存区域的逗号分隔列表。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页支持或者巨页支持。设置 **DB2_LARGE_PAGE_MEM**=DB 将对数据库共享内存区域启用大页内存；如果 **database_memory** 设为 AUTOMATIC，那么将禁止 STMM 对此共享内存区域进行自动调整。在 AIX 上，设置 DB2_LARGE_PAGE_MEM=DB:16GB 将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页或巨页来提高性能。要使 DB2 数据库系统可以使用大页或巨页，必须先将操作系统配置为使用大页或巨页。

  要对 64 位 DB2 AIX 版上的代理程序专用内存启用大页（**DB2_LARGE_PAGE_MEM**=PRIVATE 设置），必须在操作系统中配置大页，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** 注册表变量的值，以便对 FCM 内存启用大页。

  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，那么应该显示下面三行（服务器上配置的内存量不同，显示的数字也会不同）：

```
HugePages_Total:   200
   HugePages_Free:    200
   Hugepagesize:    16384 kB
```

如果没有看到这几行，或者 HugePages_Total 为 0，那么需要配置操作系统或内核。

O在 Windows 上，系统上可用的大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_ALLOCATION_SIZE** 注册表变量应设为较高值（例如，256 MB），以便在 Windows 上实现一致的性能整理大内存页。（请注意，**DB2_ALLOCATION_SIZE** 需要您停止然后重新启动实例才能使更改生效。）

DB2_LOGGER_NON_ BUFFERED_IO

- 操作系统：所有操作系统
- 缺省值：AUTOMATIC，值：AUTOMATIC、ON 或 OFF
- 此变量允许您控制是否对日志文件系统执行直接 I/O (DIO) 操作。如果 **DB2_LOGGER_NON_BUFFERED_IO** 设为 AUTOMATIC，那么将通过 DIO 来打开活动日志窗口（即，主日志文件），并且将对所有其他记录器文件进行缓存。如果它设为 ON，那么将通过 DIO 来打开所有日志文件句柄。如果它设为 OFF，那么将对所有日志文件句柄进行缓存。

DB2MAXFSCRSEARCH

- 操作系统：所有操作系统
- 缺省值：5，值：-1 以及 1 到 33 554
- 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) 的数量。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 4000 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，或者如果有许多并行执行的应用程序，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 **locklist** 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。
- 在 DB2 V9.7 及更高版本的发行版中，不支持将 DEFER 值用于具有分区 RID 索引的数据分区 MDC 表。仅支持 OFF 和 IMMEDIATE 值。如果 **DB2_MDC_ROLLOUT** 注册表变量设为 DEFER，或者 CURRENT MDC ROLLOUT MODE 专用寄存器设为 DEFERRED 以覆盖 **DB2_MDC_ROLLOUT** 设置，那么清除转出类型将为 IMMEDIATE。

  如果 MDC 表仅存在非分区 RID 索引，那么支持执行延迟索引清除转出。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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。是否会这样取决于操作系统的（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，那么设为 NO 会提供较小的性能提高。

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** 设为 AUTOMATIC，否则此变量的设置不起作用。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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 的替代方法。

  如果这些变量设为 ON，那么会使用内存映射 I/O 在 DB2 缓冲池中读取和写入数据，并且随后从文件系统高速缓存中除去此数据。这可避免对 DB2 数据进行重复高速缓存。但是，绕过文件系统高速缓存的建议方法是在表空间级别指定 NO FILE SYSTEM CACHING 子句，并将这些变量保留为缺省设置 OFF。

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 的替代方法。

  如果这些变量设为 ON，那么会使用内存映射 I/O 在 DB2 缓冲池中读取和写入数据，并且随后从文件系统高速缓存中除去此数据。这可避免对 DB2 数据进行重复高速缓存。但是，绕过文件系统高速缓存的建议方法是在表空间级别指定 NO FILE SYSTEM CACHING 子句，并将这些变量保留为缺省设置 OFF。

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。这样可以提高不使用 fork() API 的 DB2 应用程序的性能。

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:*number_of_bytes*
  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:*number_of_bytes*
  3. 消息查询内存: 缺省大小为 4 MB（在 32 位和 64 位平台上）; 覆盖选项是 QUE:<number of bytes>。

  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
  ```

  要将消息队列内存增加到 64 MB，请使用:

  ```
  db2set DB2NTMEMSIZE=QUE:67108864
  ```

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** 设为 ON，那么不需要文件系统高速缓存。如果 **DB2NTNOCACHE** 设为 OFF，那么操作系统会高速缓存 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。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

注: 对于表空间容器，将 NO FILE SYSTEM CACHING 子句与 ALTER TABLESPACE 或 CREATE TABLESPACE 语句配合使用与将 **DB2NTNOCACHE** 设为 ON 有相同的优点。

DB2NTPRICLASS

- 操作系统: Windows
- 缺省值: NULL，值: R、H 和任何其他值

- 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 来设置）
  - HIGH_PRIORITY_CLASS（使用 H 来设置）

  此变量与使用 **DB2PRIORITIES** 设置的各个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: 建议不要使用 **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 **DB2NTWORKSET**=*min*, *max* 语法来指定 **DB2NTWORKSET**，其中 *min* 和 *max* 以兆字节表示。

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> 为 <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。当内存约束导致在激活数据库、前滚恢复或崩溃恢复期间出现故障时，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，当数据库管理器连 16 页的最小缓冲池都无法启动时，请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> 为 <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，以便它们可以启动。

DB2_PINNED_BP

- 操作系统: AIX、HP-UX 和 Linux
- 缺省值: NO，值: YES 或 NO
- 将此变量设为 YES 将致使 DB2 请求操作系统锁定 DB2 的数据库共享内存。将 DB2 配置为锁定数据库共享内存之后，应注意确保系统不会被过度使用，这是因为操作系统在管理内存方面的灵活程度将下降。

  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** 配置参数设为 AUTOMATIC 来激活此功能）。

对于 64 位环境中的 HP-UX，除了修改此注册表变量外，还必须对 DB2 实例组授予 MLOCK 特权。为此，具有 root 用户访问权限的用户要执行以下操作:

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: 建议不要使用 **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。
- 当仅在前导“键序列”列及其子集上指定了等于谓词时，此变量允许对搜索式 UPDATE 语句（它以范围集群表中的多行作为目标）进行经过优化和简化的更新处理。由于对于页面上的所有行只更新单个日志记录，而不是对于每一行都要更新日志记录，因此也简化了日志记录。

  用法:

  ```
  db2set DB2_RCT_FEATURES=GROUPUPDATE=ON
  ```

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: 配置文件的有效路径，在运行 AIX 6.1 Technology Level (TL) 5 或更高版本的 POWER7® 系统上为 AUTOMATIC
- 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 进程与特定资源集绑定。这对于一些性能调整方案很有用。

  在运行 AIX 6.1 Technology Level (TL) 5 或更高版本的 POWER7 系统上，此变量可设为 AUTOMATIC。通过此设置，DB2 数据库系统自动确定硬件拓扑

并按以下方式将引擎可分派单元 (EDUs) 指定给各种硬件模块: 在此方式下，可在需要访问相同内存区域的多个 EDU 之间更高效地共享内存。此设置适用于带有 16 个或更多核心的较大 POWER7 系统，并且可能在运行某些工作负载时实现更高查询性能。最好在将此变量设为 AUTOMATIC 之前和之后对工作负载运行性能分析以验证是否存在任何性能改进。

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 进程与操作系统资源的策略绑定。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 进程都与该资源集中的单个资源绑定。资源分配以循环方式进行。

样本配置文件:

示例 1: 将所有 DB2 进程与 CPU 1 或 3 绑定。

```
<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 进程与下列其中一个资源集绑定: sys/node.03.00000、sys/node.03.00001、sys/node.03.00002 或 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>
```

注意: 仅适用于 AIX: 使用 RSET 方法需要 CAP_NUMA_ATTACH 功能。

示例 3: （仅适用于 Linux）将与 SAMPLE 数据库相关联的缓冲池标识 2 和 3 的所有内存与 NUMA 节点 3 绑定。并且，将数据库内存总量的 80% 用于与 NUMA 节点 3 绑定，并保留 20% 的内存量分布在所有节点中作为并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）定义两个分别由 CPU 掩码 0x0F 和 0xF0 指定的不同处理器集。将 DB2 进程和缓冲池标识 2 与处理器集 0x0F 绑定，并将 DB2 进程和缓冲池标识 3 与处理器集 0xF0 绑定。对于每个处理器集，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）。

```
<RESOURECE_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>0x0F</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_FIFO2 调度策略，并提高 DB2 日志写程序和阅读器进程的优先级。

```
<RESOURCE_POLICY>
  <SCHEDULING_POLICY>
   <POLICY_TYPE>SCHED_FIFO2</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_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，那么将发出 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。

  在启用了分区内并行性时，不支持此注册表变量。

  如果带有 **-immediate** 参数发出 **db2set** 命令，那么对此变量的更改会在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例。

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 65 536 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: 尽管您可以使用字节、千字节或兆字节单位来指定 **DB2_SET_MAX_CONTAINER_SIZE**，但是 **db2set** 将以字节为单位来指示它的值。
- 如果该值设为 -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。启用 **DB2_EVALUNCOMMITTED** 之后，将自动跳过已删除的行，但不会跳过索引中未落实的伪删除键，除非还启用了 **DB2_SKIPDELETED**。

  只有在“当前已落实”语义对避免锁定争用没有帮助时，**DB2_SKIPDELETED** 才适用。如果设置了此变量，并且“当前已落实”语义适用于扫描，那么将不会跳过已删除的行；而是，将处理它们的当前已落实版本。

  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。

  如果正在使用“当前已落实”语义，那么此注册表变量没有任何作用。也就是说，即使将 **DB2_SKIPINSERTED** 设为 OFF 并启用“当前已落实”行为，也仍然会跳过未落实的已插入行。

  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

v98_u3

- 操作系统: 所有操作系统
- 缺省值: -2, 值: -2、-1 或 0 到 *n*，其中 *n* 为 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。

  此变量的缺省设为 -2，这意味着对于其大小小于或等于 1 个扩展数据块 * 容器数的任何溢出的 SMS 临时对象，将没有任何不需要的文件系统访问。超过此大小的临时对象会被截断为 0 个扩展数据块。

  如果此变量设为 0，那么不执行任何特殊阈值处理。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。超过此阈值的对象将被截断为阈值大小。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。

  如果带有 **-immediate** 参数发出 **db2set** 命令，那么对此变量的更改会在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例。

DB2_SQLWORKSPACE_CACHE

- 操作系统：所有操作系统
- 缺省值：30，值：10 到 2000
- 此变量允许您控制在 SQL 工作空间中对先前使用的段进行高速缓存的数量。

  SQL 工作空间包含执行 SQL 时的分配（以段的形式）。正在为应用程序执行的每个 SQL 语句（静态或动态）都必须在 SQL 工作空间中保留该语句的执行期间所使用的段的唯一副本。在完成该语句的执行后，该段将成为不活动段，并且可以释放与不活动段相关联的内存分配，也可以继续在 SQL 工作空间中高速缓存这些内存分配。当从任何连接再次执行相同 SQL 语句时，将在 SQL 工作空间中查找上次执行该语句时保留的段的高速缓存副本，从而节省与分配和初始化该段的新副本相关联的成本。因此，SQL 工作空间将同时包含活动段（对应于当前执行的 SQL）和当前未在执行的高速缓存的段。

  此注册表变量的值指定允许在 SQL 工作空间中高速缓存的内存分配的百分比。此高速缓存将表示为活动段的内存分配的百分比。例如，值 50 表示 SQL 工作空间包含所有活动段（当前在执行的段）以及最多 50% 的先前执行了高速缓存的段（可以复用这些段）。应该根据您希望 SQL 工作空间中可供复用的多少来调整 **DB2_SQLWORKSPACE_CACHE** 的设置。例如，增加此变量的大小可以在一定程度上提高 OLTP 工作负载的性能。在另一方面，较高的设置也意味着将增加应用程序共享堆的大小。

  注：如果未将 **appl_memory** 数据库配置参数设为 AUTOMATIC，那么 SQL 工作空间的大小还会受 **appl_memory** 的限制，因此 SQL 工作空间可能并不会提供 **DB2_SQLWORKSPACE_CACHE** 设置所允许的高速缓存量；在这样的情况下，您可能需要考虑增大 **appl_memory**（或将其设为 AUTOMATIC）。
  此注册表变量不是动态的

DB2_TRUSTED_BINDIN

- 操作系统：所有操作系统
- 缺省值：OFF，值：OFF、ON 或者 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: Veritas VxFS、JFS2、GPFS 和 ext4（仅适用于 Linux）文件系统上的 AIX、Linux 和 Solaris
- 缺省值: ON（对于 Veritas VxFS、JFS2、GPFS 和 ext4），值: ON 或 OFF
- 允许快速预分配文件系统功能部件保留表空间，并可以提高创建或改变大型表空间的过程以及数据库恢复操作的速度。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。请注意，禁用快速预分配后，必须复原数据库。

DB2_USE_IOCP

- 操作系统: AIX
- 缺省值: ON，值: ON 或 OFF
- 此变量允许在提交和收集异步 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。

其他变量

可以设置其他 DB2 变量，包括对缺省管理服务器、客户机路径以及退出应用程序时落实对 DB2 数据所作的更改的能力进行控制的那些变量。

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。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

DB2AUTH

- 操作系统: 所有操作系统
- 缺省值: 未设置。值: DISABLE_CHGPASS、OSAUTHDB、SQLADM_NO_RUNSTATS_REORG、TRUSTEDCLIENT_DATAENC 或 TRUSTEDCLIENT_SRVRENC
- 此变量允许您调整用户认证的行为。有效值如下所示:
  - ALLOW_LOCAL_FALLBACK: 此值允许 DB2 服务器在配置为使用类属安全性服务 (GSS) 插件时回退到使用 SERVER 认证进行本地隐式连接。如果启用了 ALLOW_LOCAL_FALLBACK，那么对于本地隐式连接，会使用

srvcon_pw_plugin 数据库管理器配置参数指定的用户标识和密码插件来认证用户，而不是使用指定的 GSS 认证，例如，KERBEROS、KRB_SERVER_ENCRYPT、GSSPLUGIN 或 GSS_SERVER_ENCRYPT。

发出与本地数据库的连接（请注意，*本地* 意味着仅 IPC 而不是 TCP/IP）时会创建本地隐式连接而不提供用户标识或密码。DB2 使用当前会话或进程的用户标识作为此连接的用户标识。DB2 提供的密码插件假定操作系统已认证从操作系统检索到的用户标识，因此不需要验证密码。

注: 如果提供用户标识和密码，那么不会将此连接视为本地隐式连接，并且 ALLOW_LOCAL_FALLBACK 选项不适用。

DB2 使用的密码插件由 **srvcon_pw_plugin** 数据库管理器配置参数确定。如果 **srvcon_pw_plugin** 参数设为 IBMLDAPauthserver，那么 IBMLDAPauthserver 插件会处理本地隐式连接。如果 **srvcon_pw_plugin** 参数设为定制安全插件，那么此定制插件会处理本地隐式连接。如果未设置 **srvcon_pw_plugin** 参数，那么缺省插件 (IBMOSauthserver) 会处理此本地隐式连接。DB2 提供的安全插件始终允许本地隐式连接，因为它们假定操作系统已验证该用户。

– DISABLE_CHGPASS: 此值禁用从客户机更改密码的能力。
– OSAUTHDB: 此值指示 DB2 数据库管理器使用用户在 AIX 操作系统上的认证和组设置。透明 LDAP 支持也已扩展到 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
– SQLADM_NO_RUNSTATS_REORG: 此值是在 DB2 V9.7 FP5 中引入的，用于使具有 SQLADM 权限的用户无法执行 reorg 或 runstats 操作。
– TRUSTEDCLIENT_DATAENC: 此值强制不可信客户机使用 DATA_ENCRYPT。此值不适用于 DB2 Connect 网关。
– TRUSTEDCLIENT_SRVRENC: 此值强制不可信客户机使用 SERVER_ENCRYPT。此值不适用于 DB2 Connect 网关。

DB2_BCKP_INCLUDE_LOGS_WARNING

- 操作系统: 所有操作系统
- 缺省值: FALSE，值: FALSE 或 TRUE
- 指定无法包括所有必需日志文件的联机备份是否应该仍可以成功完成。缺省情况下，未显式指定 INCLUDE LOGS 或 EXCLUDE LOGS 选项的联机备份在未成功包括所有日志时会失败。将此变量设为 TRUE 时，这些备份将可以成功，但会发出警告 (SQL2440W)。

  在 SAP 环境中，如果设置了 **DB2_WORKLOAD**=SAP，那么此注册表变量的缺省值为 TRUE。

DB2_BCKP_PAGE_VALIDATION

- 操作系统: 所有操作系统
- 缺省值: FALSE，值: FALSE 或 TRUE

- 指定备份期间是否进行 DMS 和 AS 页面验证。

DB2CLIINIPATH

- 操作系统: 所有操作系统
- 缺省值: NULL
- 用于替换 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 V8 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 V8 中, 该行为更改为退出时将回滚未完成的事务。此注册表变量使依赖于先前行为的嵌入式 SQL 应用程序的用户能够在 DB2 V9 中继续运行该应用程序。此注册表变量不影响 JDBC、CLI 和 ODBC 应用程序。

  请注意, 建议您不要使用此注册表变量, 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 DB2 V9 之前开发的任何应用程序是否继续依赖于此功能, 并根据需要在应用程序中添加适当的显式 COMMIT 或 ROLLBACK 语句。如果打开此注册表变量, 那么应该注意不要实现在退出前未能显式地执行落实 (COMMIT) 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_COMMON_APP_DATA_PATH

- 操作系统: Windows
- 缺省值: Windows 的公共应用程序数据路径。
  - 对于 Windows XP 和 Windows 2003 操作系统: C:\Documents and Settings\All Users\Application Data\
  - 对于 Windows Vista 及更高版本操作系统: C:\ProgramData\
- 指向用于保存 DB2 副本的 DB2 公共应用程序数据的用户定义位置。如果在安装响应文件期间指定了 **DB2_COMMON_APP_DATA_TOP_PATH** 或在定制安装步骤期间填充了"DB2 公共应用程序数据顶部路径"字段, 那么会填充此注册表变量。

  从 V9.7 FP5 开始, 此注册表变量在 **db2set** 命令输出中可视, 但不可更改。尝试更改给定注册表值将导致 DBI1301E 无效值错误。

DB2_COMPATIBILITY_VECTOR

- 操作系统: 所有操作系统
- 缺省值: NULL, 值: NULL 或者 00 到 FFF
- **DB2_COMPATIBILITY_VECTOR** 注册表变量用来启用自从 DB2 V9.5 以来引入的一个或多个 DB2 兼容性功能部件。通过这些功能部件, 更容易完成将对其他关系数据库供应商编写的应用程序迁移到 DB2 V9.5 或更高版本的任务。
- **DB2_COMPATIBILITY_VECTOR** 被表示为一个十六进制值, 该变量中的每个位启用其中一个 DB2 兼容性功能部件, 如 DB2_COMPATIBILITY_VECTOR 值

表中所述。要启用所有受支持的兼容性功能部件，请将此注册表变量设为值 ORA（此值等价于十六进制值 FFF）。这是建议设置。

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
- 当此变量设为 YES（TRUE 或 1）时，它指定在发生中断时，应立即断开与 V8（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 V8（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** 设为 YES。
  - 如果正在通过 DB2 Connect 网关对 V8（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** 设为 YES。

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** 时以及复原数据库时，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:
  如果使用路径支持功能来创建新数据库，那么在 DB2 V9.1 之前编写的使用 db2DbDirGetNextEntry() API 和其旧版本的应用程序可能无法正常工作。有关各种方案和正确操作过程的详细信息，请参阅 http://www.ibm.com/software/data/db2/support/db2_9/。

DB2_DDL_SOFT_INVAL

- 操作系统：所有操作系统
- 缺省值：ON，值：ON 或 OFF
- 允许在适用的数据库对象被删除或更改时使其软失效。

  如果 **DB2_DDL_SOFT_INVAL** 设为 ON，那么任何 DDL 操作（例如删除、更改或拆离）不必等待引用相同对象的事务完成即可启动。依赖于这些对象的当前执行将继续使用原始的对象定义，而新执行将使用更改后的对象。这将提高发出 DDL 语句时的并行性。

  注：新的软失效功能仅适用于动态程序包。任何使用静态程序包的对象仍要求执行硬失效操作。

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，此注册表变量将加以忽略，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。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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® 包装器以及 InfoSphere® 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。

除非数据库管理器参数 **federated** 设为 YES，否则此变量将加以忽略。

DB2_DMU_DEFAULT

- 操作系统: 所有操作系统
- 缺省值: NULL，值: IMPLICITLYHIDDENMISSING 和 IMPLICITLYHIDDENINCLUDE
- 此变量允许您设置在装入、导入、INGEST 和导出实用程序省略列列表时是否包含隐式隐藏列的缺省行为。有效值如下所示:

  NULL
  这意味着未指定缺省行为。如果表包含隐式隐藏列，那么实用程序必须显式指定列列表或必须指定隐藏列选项。否则，会发生错误。

  IMPLICITLYHIDDENMISSING
  实用程序假定缺省情况下未包含隐式隐藏列，除非指定了列列表或隐藏列选项。

  IMPLICITLYHIDDENINCLUDE
  实用程序假定缺省情况下包含隐式隐藏列，此时既未指定列列表也未指定隐藏列选项。

  考虑说明 **DB2_DMU_DEFAULT** 的设置对装入操作结果的影响的以下示例:

  – **DB2_DMU_DEFAULT** 设为 IMPLICITLYHIDDENMISSING

  ```
  db2 load from delfile1 of del insert into table1
  ```

  如果 table1 包含隐式隐藏列，那么装入实用程序假定隐式隐藏列的数据不在输入文件中。

  – **DB2_DMU_DEFAULT** 设为 IMPLICITLYHIDDENINCLUDE

  ```
  db2 load from delfile1 of del insert into table1
  ```

  如果 table1 包含隐式隐藏列，那么装入实用程序假定隐式隐藏列的数据在输入文件中并尝试装入该数据。

DB2_DOCHOST

- 操作系统: 所有操作系统
- 缺省值: 未设置（但 DB2 仍将尝试从 IBM Web 站点访问信息中心），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。

- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

DB2DSDRIVER_CFG_PATH

- 操作系统: 所有操作系统
- 缺省值: NULL
- 用于覆盖 db2dsdriver.cfg 配置文件的缺省目录。
- 在 V10.1 FP2 及更高版本的修订包中，可以使用定界字符来指定 db2dsdriver.cfg 配置文件的多个目录。
  - 在 Windows 操作系统上，定界字符是分号 (;)。
  - 在 Linux 和 UNIX 操作系统上，定界字符是分号 (;) 或冒号 (:)。分号字符和冒号字符不能一起用作 **DB2DSDRIVER_CFG_PATH** 变量值的定界符。

  不要在目录名称中使用定界字符。会按 **DB2DSDRIVER_CFG_PATH** 变量值中指定的顺序来搜索目录。
- 句点 (.) 指定当前目录。
- 有关 db2dsdriver.cfg 配置文件的更多信息，请参阅“相关参考”部分。

DB2DSDRIVER_CLIENT_HOSTNAME

- 操作系统: 所有操作系统
- 缺省值: NULL
- 用于覆盖 (db2dsdriver.cfg) 配置文件的缺省客户机主机名。该变量强制 CLI 从 db2dsdriver.cfg 文件的客户机自动重新路由部分中选取客户机主机名条目。

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。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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，请将 **mon_heap_sz** 配置参数设为 AUTOMATIC。如果将 **mon_heap_sz** 设为特定值，那么确保 **DB2_EVMON_EVENT_LIST_SIZE** 设为更小的值。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* 和 *op2* 等等）都表示指向特定 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 123. 要用于 **DB2_EVMON_STMT_FILTER** 以对特定 SQL 操作限制事件监视器输出的值*

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'
```

在此示例中，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
```

在此示例中，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**，其值在 DB2 数据库服务器产品上解释为 YES，在客户机上解释为 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。

如果要运行 SAS 数据库内分析（通过设置 **DB2_SAS_SETTINGS** 注册表变量来启用），那么用于与 SAS 嵌入式进程 (EP) 的连接的内存也会从 FMP 堆分配。调整该堆以容纳那些运行查询（这些查询包括数据库内分析）的连接时，受防护例程的准则适用。一般来说，您可能期望 FMP 堆内存需求增大 120 KB。但是，如果对 **DB2_SAS_SETTINGS** 注册表变量指定 COMM_BUFFER_SZ 选项，那么 FMP 堆内存需求的增加幅度为 COMM_BUFFER_SZ 选项值乘以您希望支持的并行 SAS 查询数的两倍。

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。可以指定的最大大小是 4 GB。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。

DB2_HADR_NO_IP_CHECK

- 操作系统：所有操作系统
- 缺省值：OFF，值：ON IOFF
- 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）
- 如果设置了 **DB2_HADR_PEER_WAIT_LIMIT** 注册表变量，并且由于将日志复制到备用数据库而导致对 HADR 主数据库执行的日志记录操作被阻塞指定的秒数，那么主数据库将脱离对等状态。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。

  注：如果设置 **DB2_HADR_PEER_WAIT_LIMIT**，请使用最小值 10 以避免触发假警报。

  此参数对备用数据库没有影响，但建议对主数据库和备用数据库使用相同值。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_ROS

- 操作系统：所有操作系统
- 缺省值：OFF，值：OFF 或 ON
- 此变量启用对备用功能部件进行 HADR 读取。对 HADR 备用数据库启用了 **DB2_HADR_ROS** 时，此备用数据库将接受客户机连接并允许对它运行只读查询。**DB2_HADR_ROS** 是静态注册表变量，因此它要求重新启动 DB2 实例以使已更改的设置生效。

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** 日志文件中。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** 日志文件中。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。

DB2_HISTORY_FILTER

- 操作系统：所有操作系统
- 缺省值：NULL，值：NULL、G、L、Q、T 或 U
- 此变量指定不会修改历史记录文件的操作。可使用 **DB2_HISTORY_FILTER** 注册表变量来减少对历史记录文件的潜在争用（通过过滤掉操作）。使用逗号分隔列表来指定不能修改历史记录文件的操作:

  ```
  db2set DB2_HISTORY_FILTER=T, L
  ```

  DB2_HISTORY_FILTER 的可能值包括:

  – **G**：重组操作
  – **L**：装入操作
  – **Q**：停顿操作
  – **T**：改变表空间操作
  – **U**：卸载操作

DB2_INDEX_PCTFREE_DEFAULT

- 操作系统：所有操作系统
- 缺省值：未设置，值：0 到 99
- 此注册表变量指定构建索引时每个索引页要留为可用空间的百分比。如果您在 CREATE INDEX 语句上显式指定 PCTFREE 子句，那么 **DB2_INDEX_PCTFREE_DEFAULT** 的设置会被覆盖。此注册表变量不会影响 CREATE INDEX 语句上的 LEVEL2 PCTFREE 子句。

  此注册表变量在数据库升级时不会应用，即使升级期间重新创建索引。它仅应用于新安装或升级完成后应用。此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。

  如果 **DB2_WORKLOAD** 设为 SAP，那么 **DB2_INDEX_PCTFREE_DEFAULT** 将设为 0。

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。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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，请执行以下操作:

  ```
  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 Microsoft 可用，那么使用此操作系统；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]* 或者 *base_domain_name*:SSL:636（当使用支持 SSL 的 LDAP 主机时）
- 指定 LDAP 目录所在位置的主机名和可选的端口号，其中 *base_domain_name* 是 TCP/IP 主机名，*[:port_number]* 是端口号。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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** 设为 NO。

  要获得最佳性能，缺省情况下，应将此变量设为 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_LIMIT_FENCED_GROUP

- 操作系统：Windows
- 缺省值：NULL，值：ON 或 OFF
- 如果您启用了扩展安全性，那么可以通过将此注册表变量设为 ON 以及通过将 DB2 服务帐户（用来运行 DB2 服务的用户名）添加至 DB2USERS 组，将采用受防护方式的进程 (**db2fmp**) 的操作系统特权限制为对 DB2USERS 组指定的特权。

  注：如果使用了 LocalSystem 作为 DB2 服务帐户，那么设置 **DB2_LIMIT_FENCED_GROUP** 将不起作用。

  要对 **db2fmp** 进程授予其他操作系统特权，可以通过对具备这些附加特权的操作系统组添加 DB2 服务帐户来实现。

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。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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**。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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：建议您不要使用 **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。

  注：如果 **DB2_LGPAGE_BP** 设为 YES，您将无法使用内存保护。即使 **DB2_MEMORY_PROTECT** 设为 YES，DB2 也无法保护缓冲池内存并禁用此功能部件。

DB2_MIN_IDLE_RESOURCES

- 操作系统：Linux
- 缺省值：OFF，值：OFF 或 ON
- 此变量指定已激活的数据库在空闲时将使用最少的处理资源。这可能在某些虚拟 Linux 环境（例如，zVM）中很有用。在这些环境中，节省少量的资源有助于主机虚拟机监视器在它的所有虚拟机之间更有效地调度其 CPU 和内存资源。

DB2_NCHAR_SUPPORT

- 操作系统：所有操作系统
- 缺省值：ON，值：ON 或 OFF
- 当此变量设为 ON（缺省值）时，图形数据类型的 NCHAR、NVARCHAR 和 NCLOB 拼写可供在 Unicode 数据库中使用。还提供了各种与本地语言字符相关的函数，例如，NCHAR() 和 TO_NCHAR()。

如果现有数据库具有名为 NCHAR、NVARCHAR 或 NCLOB 的用户定义类型，那么此变量只应设为 OFF。

注: 在将来的发行版中可能会除去 **DB2_NCHAR_SUPPORT** 注册表变量，那时，数据库中将不能具有用户定义的类型 NCHAR、NVARCHAR 或 NCLOB。

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: Linux 和 UNIX
- 缺省值: 3，值: 1[:FORK] 到 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** 执行期间创建的，在使用缺省的 **IBMOSauthserver** 安全插件时，它们用于处理检查密码请求。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。

  可以将 **DB2_NUM_CKPW_DAEMONS** 设为介于 1 与 100 之间的值。数据库管理器所创建的守护程序的数目由 **DB2_NUM_CKPW_DAEMONS** 指定。每个守护程序都可以直接处理检查密码请求。

  您可以添加可选的 FORK 参数，以使检查密码守护程序能够显式地衍生外部检查密码程序 (**db2ckpw**) 以处理检查密码请求。这与前发行版中将 **DB2_NUM_CKPW_DAEMONS** 设为 0 相似。在 FORK 方式下，每个检查密码守护程序都将为每个请求衍生检查密码程序以检查密码。在 FORK 方式下，守护程序将作为实例所有者启动。

  如果 **DB2_NUM_CKPW_DAEMONS** 设为零，那么有效值将设为 3:FORK，即，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** 未设置或设为 ON 时，将启用统计事件日志记录，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为 OFF 来将其禁用。

- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

显式启用统计信息事件日志记录时（设为 ON），您可以修改一些选项:

- NUM: 轮换日志文件的最大数目。缺省值: 5，值: 1 - 15
- SIZE: 轮换日志文件的最大大小。（每个旋转文件的大小为 SIZE/NUM。）缺省值 = 15 Mb，值 1 Mb – 4096 Mb
- NAME: 轮换日志文件的基本名称。缺省值: db2optstats.*number*.log，例如 db2optstats.0.log 和 db2optstats.1.log 等等。
- DIR: 轮换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 计算机名称
- 指定包含 DB2 实例概要文件和 DB2 实例的 Win32 注册表列表的远程计算机名称。安装 DB2 数据库产品后，**DB2REMOTEPREG** 的值只能设置一次，并且设置后不得更改。使用此变量要格外小心。
- 在分区数据库环境中，可使用 **DB2REMOTEPREG** 注册表变量将不是实例所有者的计算机配置为使用拥有实例的计算机上的注册表变量的值。有关何时使用此变量的更多信息，请参阅第 498 页的『在分区数据库环境中设置实例级别的变量』。

  如果 DB2 数据库管理器在 Windows 操作系统上读取这些注册表变量，那么它会先读取 **DB2REMOTEPREG** 值。如果设置了 **DB2REMOTEPREG** 变量，那么数据库管理器会在远程计算机上打开 **DB2REMOTEPREG** 变量指定的注册表。这些注册表变量的后续读取和更新会重定向至指定的远程计算机。

  如果并非实例所有者的计算机要访问远程注册表，那么“远程注册表服务”必须正在目标计算机上运行。而且，用户登录帐户和所有 DB2 服务登录帐户必须对远程注册表有足够的访问权。要使用 **DB2REMOTEPREG** 变量，必须在 Windows 域环境中运行才能向域帐户授予必需的注册表访问权。
- 不要在 Microsoft Cluster Server 环境中使用 **DB2REMOTEPREG**。

DB2_RESOLVE_CALL_ CONFLICT

- 操作系统: AIX、HP-UX、Solaris、Linux 和 Windows
- 缺省值: YES，值: YES 或 NO

- 当触发器调用的例程尝试访问已由触发器主体中的其他语句和例程修复的表时，这会中断嵌套的 SQL 语句规则。设置 **DB2_RESOLVE_CALL_CONFLICT** 以在执行 CALL 语句之前根据触发器的 SQL 标准规则强制完成对表的所有修改。

  在重置 **DB2_RESOLVE_CALL_CONFLICT** 之前，必须停止实例，然后将其重新启动。然后重新绑定导致调用触发器的任何程序包。要重新绑定 SQL 过程，使用：CALL SYSPROC.REBIND_ROUTINE_PACKAGE ('P','*procedureschema.procedurename*','CONSERVATIVE');

  您需要了解，**DB2_RESOLVE_CALL_CONFLICT** 会对性能产生影响。将 **DB2_RESOLVE_CALL_CONFLICT** 设为 YES 会导致 DB2 数据库管理器通过在需要时插入临时表来解决所有潜在读/写冲突。通常，这样做的影响很小，因为最多插入一个临时表。这将在 OLTP 环境中产生很小影响，因为只有一（或少量）行被触发语句修改。通常，当遵循一般建议将 SMS（系统管理的空间）用作临时表空间时，认为设置 **DB2_RESOLVE_CALL_CONFLICT** 所产生的性能影响很小。

  如果带有 **-immediate** 参数发出 **db2set** 命令，那么对此变量的更改会在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例。

DB2_RESTRICT_DDF

- 操作系统：所有操作系统
- 缺省值：FALSE，值：TRUE 或 FALSE
- 指定是否应禁用动态数据格式化功能（又称为*顺序流式传输*）。如果 **DB2_RESTRICT_DDF** 设为 TRUE，那么服务器会通知 JDBC 驱动程序将禁用动态数据格式化功能。

  在 SAP 环境中，如果设置了 **DB2_WORKLOAD**=SAP，那么此注册表变量的缺省值为 TRUE。

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：建议您不要使用 **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。

DB2_SAS_SETTINGS

- 操作系统：所有操作系统
- 缺省值：未设置。值：ENABLE_SAS_EP、LIBRARY_PATH、COMM_BUFFER_SZ、COMM_TIMEOUT、RESTART_RETRIES、DIAGPATH 或 DIAGLEVEL
- 此变量是带 SAS 嵌入式进程 (EP) 的数据库内分析的配置的主要点。除 ENABLE_SAS_EP 选项以外的所有选项都可在线配置。

  ENABLE_SAS_EP
  如果将此选项设为 TRUE，那么 SAS EP 会在您发出 **db2start** 命令时自动启动。此选项的缺省值为 FALSE。

LIBRARY_PATH

下次 SAS EP 进程启动时从中装入 SAS EP 库的标准路径。如果未指定路径，那么 DB2 数据库管理器会在 sqllib/function/sas 目录下查找 SAS EP 库。为安全起见，应将 SAS EP 库安装在未授权用户无法修改或替换该文件的位置。选择下列其中一个选项:

- 请确保库路径和 SAS EP 库文件由实例所有者拥有并且只能由实例所有者写入。
- 请将该文件放在设置了粘性位的目录（例如，sqllib/function）中。

只有具有 SYSADM 权限的用户才能使用 **db2set** 命令来配置库路径。

COMM_BUFFER_SZ

一个整数值，用于指定要用于 DB2 数据服务器与 SAS EP 之间的通信会话的共享内存缓存量（以 4 KB 页计）。此参数的有效值范围为 1 到 32767。缺省值为 15。通信缓冲区是从 FMP 通信堆分配的。有关更多信息，请参阅 **DB2_FMP_COMM_HEAPSZ**。

COMM_TIMEOUT

一个超时值，DB2 数据库管理器在交换控制消息时用它来确定是否应将 SAS EP 视为无法响应。如果达到此值，那么数据库管理器会终止 SAS EP 以便可再次衍生 SAS EP。缺省值为 300 秒。

RESTART_RETRIES

一个整数值，用于指定检测到 SAS EP 异常终止后 DB2 数据库管理器应尝试重新衍生该 SAS EP 的次数。此参数的有效值范围为 0 到 100。缺省值为 10。达到重试计数后，数据库管理器会等待 15 分钟，然后再次重试该操作。

DIAGPATH

一个标准路径，用于指定 SAS EP 的诊断日志的位置。缺省值为 **diagpath** 数据库管理器配置参数的值。

DIAGLEVEL

一个整数值，用于指定 SAS 诊断日志中捕获的消息的最低严重性级别。此选项的有效值如下所示:

1 严重

2 错误

3 警告

4 参考

缺省值为 **diaglevel** 数据库管理器配置参数的值。

MEMSIZE

一个整数值，用于指定 SAS EP 可在特定主机上消耗的最大内存量（以 4 KB 页计）。此选项的有效值范围为 1 到 4294967295。如果有多个逻辑分区，那么应用于每个分区的值将除以对应主机上的逻辑分区数。缺省值为 **instance_memory** 数据库管理器配置参数值的 20%。如果将 **instance_memory** 参数设为固定值，请确保此值会将 SAS EP 的额外内存需求考虑在内。

示例:

```
db2set DB2_SAS_SETTINGS="ENABLE_SAS_EP:TRUE;
  LIBRARY_PATH:/home/instowner/sqllib/function/sas"
```

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
-

  注: 在V9.7 中，建议不要使用 **DB2_SERVER_ENCALG**，将来的发行版中可能会将它除去。

  如果将实例升级到 DB2 V9.7 时设置了 **DB2_SERVER_ENCALG** 注册表变量，那么 **alternate_auth_enc** 配置参数将根据 **DB2_SERVER_ENCALG** 的设置来设为 AES_ONLY 或 AES_CMP。因此，要指定用于对用户标识和密码进行加密的加密算法，请更新 **alternate_auth_enc** 配置参数。如果设置了 **alternate_auth_enc** 配置参数，那么它的值将优先于 **DB2_SERVER_ENCALG** 注册表变量值。

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_STANDBY_ISO

- 操作系统: 所有操作系统
- 缺省值: NULL, 值: UR 或 OFF
- 此变量将正在活动 HADR 备用数据库上运行的应用程序和语句所请求的隔离级别强制设为“未落实的读 (UR)”。当 **DB2_STANDBY_ISO** 设为 UR 时，高于 UR 的隔离级别会被强制设为 UR，并且不会返回警告。如果此 HADR 备用数据库转变为 HADR 主数据库，那么此变量将不起作用。

DB2STMM

- 操作系统: UNIX
- 此注册表变量控制一组参数，这些参数允许您修改自调整内存管理器 (STMM) 的某些特征。
- 参数:

  GLOBAL_BENEFIT_SEGMENT_COMPATIBLE

  - 缺省值: 未设置，值: YES 或 NO
  - 仅当数据库的 **database_memory** 配置参数设为 AUTOMATIC 时，GLOBAL_BENEFIT_SEGMENT_COMPATIBLE 参数才会起作用。

    此参数将影响 STMM 共享内存段的许可权设置。仅在具有多个实例且其中某些实例是下级实例的系统上才应该将此参数设为 YES，并且将 **database_memory** 设为 AUTOMATIC，以减轻下级实例兼容性问题，这些问题会影响调整数据库的整体数据库内存使用量。属于下列任何 DB2 发行版和修订包级别的实例都是下级实例: 所有修订包级别的 DB2 V9.1、DB2 V9.5 FP7 和更低版本以及 DB2 V9.7 FP4 和更低版本。

    对于非 root 用户安装的 DB2 中的实例，仅当您希望系统上的所有实例都使用相同的 STMM 共享内存段时，才应该设置此变量。不设置此变量或将其设为 NO 将导致非 root 用户实例使用其自己的特定于实例的 STMM 共享内存段，这可能会影响任何将 **database_memory** 设为 AUTOMATIC 的数据库调整整体数据库内存使用量。

    此注册表变量在 DB2 实例启动期间读取一次。注意，您需要在所有已升级的实例（即，非下级实例）上设置此参数，并且在设置此参数后，需要重新启动所有已升级的实例。

  GLOBAL_BENEFIT_SEGMENT_UNIQUE

  - 缺省值: 未设置，值: YES 或 NO
  - 仅当数据库的 **database_memory** 配置参数设为 AUTOMATIC 时，GLOBAL_BENEFIT_SEGMENT_UNIQUE 参数才会起作用。

    此参数指定每个已升级的实例（即，非下级实例）将使用其自己的特定于实例的 STMM 共享内存段。这意味着每个实例将调整属于它的任何数据库的整体数据库内存使用量，而与调整系统上属于其他实例的数据库的整体数据库内存使用量无关。

    对于系统上的所有实例，仅当 **instance_memory** 配置参数**未**设为 AUTOMATIC 时，才应该考虑将此参数设为 YES。

    此注册表变量在 DB2 实例启动期间读取一次。注意，需要在所有已升级的实例上设置此参数，并且在设置此参数后，需要重新启动所有已升级的实例。

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。

  此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。
- 对此变量的更改将在所有将来已编译 SQL 语句执行后立即生效。不需要重新启动实例或发出带有 **-immediate** 参数的 **db2set** 命令。

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）。

  如果运行 ALTOBJ 过程时未设置此路径，那么将在 ~sqllib/tmp 目录中创建一个临时消息文件。

  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_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
  ```

DB2_XSLT_ALLOWED_PATH

- 操作系统: 所有操作系统
- 缺省值: NULL 或 NONE，值: ALL 或有效 URI 的列表，由空格分隔
- 此注册表变量控制 DB2 实例如何引用在 XSLT 样式表中定义的外部实体。
  - NULL 或 NONE: 不允许任何 URI 引用，并且使用这类样式表的变换会失败。
  - ALL: 允许对 URI 的所有引用。

    注: 对外部 URI 的不受控制的引用可能是一个严重的安全问题。
  - URI 的列表: 仅允许对位于来自列表的 URI 的子目录中的 URI 的引用，如以下示例所示:

    ```
    db2set DB2_XSLT_ALLOWED_PATH ="http://some.website.com/test/dir
    /home/Joe/resource.txt"
    ```

第 24 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，此文件的缺省位置随 Windows 系列的操作系统版本而变化，要验证 Windows 上的缺省目录，请使用命令 **db2set DB2INSTPROF** 来检查 **DB2INSTPROF** 注册表变量的设置。还可通过更改 **DB2INSTPROF** 注册表变量来更改缺省实例目录。如果设置了 **DB2INSTPROF** 变量，那么该文件位于 **DB2INSTPROF** 变量指定的目录的 instance 子目录。

指定运行时数据文件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:

- **DB2CLIINIPATH**
- **diagpath**
- **spm_log_path**

所有数据库配置参数存储在名为 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。

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。

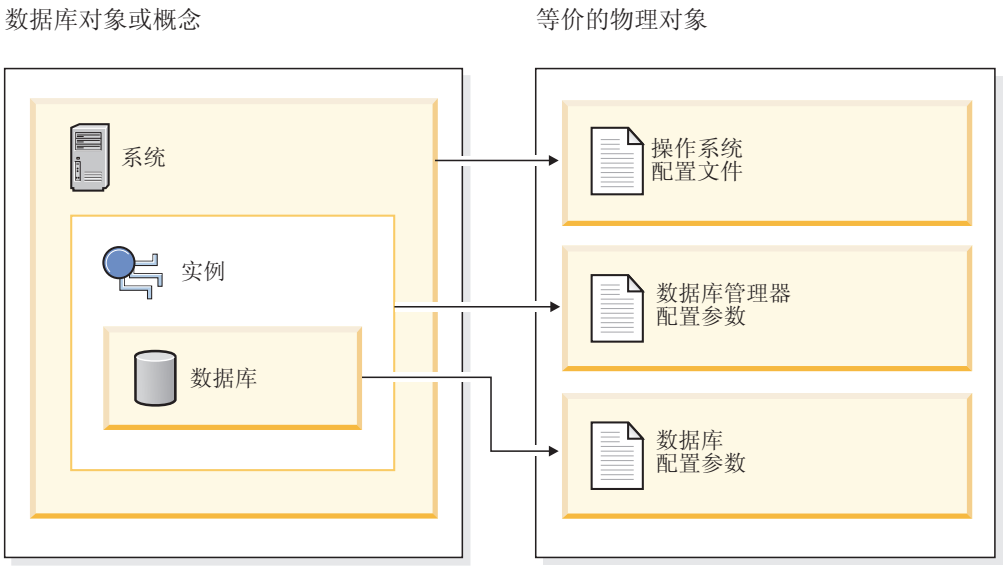


图 52. 数据库对象与配置文件之间的关系

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。

关于此任务

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。

调整配置最好是使用“配置顾问程序”或 **AUTOCONFIGURE** 命令开始。这些工具根据您对有关工作负载特征的提问的回答来生成参数值。

可以将一些配置参数设为 AUTOMATIC，从而允许数据库管理器自动管理这些参数来反映当前资源要求。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*数据库管理器参数重置为其缺省值
     - **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*数据库参数重置为其缺省值
     - **AUTOCONFIGURE**
   - 使用应用程序编程接口 (API):

     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
   - 使用公共 SQL 应用程序编程接口 (API) 过程来更新:

     可以从基于 SQL 的应用程序、DB2 命令行或命令脚本中调用公共 SQL API 过程。请通过调用下列过程来查看或更新配置参数:
     - GET_CONFIG - 获取数据库管理器配置参数或数据库配置参数
     - SET_CONFIG - 设置数据库管理器配置参数或数据库配置参数

- 通过使用 IBM Data Studio，右键单击实例以打开任务助手来更新数据库管理器配置参数。

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，那么必须先停用，然后再重新激活。）然后，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。

   要以联机方式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）。请注意，某些参数更改由于与分配空间相关的额外处理时间而可能花费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**: 在与数据库的新连接上更改的参数。例如，**dft_degree** 的新值对连接到数据库的新应用程序生效。

   虽然新参数值可能未立即生效，但是查看参数设置（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 优化器选择的存取方案。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 语句使用最佳的存取方案。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

选择一个新的存取方案。但是，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，**health_mon**）被描述为具有可接受的值 Yes 或 No，或者 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后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** 命令的缺省行为是立即应用该更改。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** 命令的 **AUTOMATIC** 关键字。

将一个参数更新为 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
```

要取消设置 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。将一个参数更新为 MANUAL 之后，该参数将不再是自动参数，并且将设为它的当前值（显示在 **GET DBM CFG SHOW DETAIL** 和 **GET DB CFG SHOW DETAIL** 命令所生成的 Current Value 列中）。

如果数据库由 **CREATE DATABASE** 或 sqlecrea API 创建，那么缺省情况下配置顾问程序会运行以使用自动计算的值更新数据库配置参数。如果数据库由添加了 AUTOCONFIGURE APPLY NONE 子句的 **CREATE DATABASE** 命令创建，或者 sqlecrea API 指定不运行配置顾问程序，那么配置参数设为缺省值。

“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** - 指示该参数可能对性能有重大影响。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** - 指示该参数可能对性能有一定程度的影响。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** - 指示该参数对性能具有不是很普遍或者不是太重要的影响。
- **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 124. 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 603 页的『agent_stack_sz -“代理程序堆栈大小”』
agentpri	否	否	高	SQLF_KTN_AGENTPRI	26	Sint16	第 605 页的『agentpri -“代理程序的优先级”』
alt_diagpath	是	否	无	SQLF_KTN_ALT_DIAGPATH SQLF_KTN_ALT_DIAGPATH_FULL	941	char [] (String)	第 606 页的『alt_diagpath -“备用诊断数据目录路径”』
alternate_auth_enc [6]	否	否	低	SQLF_KTN_ALTERNATE_AUTH_ENC	938	Uint16	第 608 页的『alternate_auth_enc -“服务器上用于传入连接的备用加密算法”配置参数』
aslheapsz	否	否	高	SQLF_KTN_ASLHEAPSZ	15	Uint32	第 608 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 610 页的『audit_buf_sz -“审计缓冲区大小”』
认证	否	否	低	SQLF_KTN_AUTHENTICATION	78	Uint16	第 610 页的『authentication -“认证类型”』
catalog_noauth	是	否	无	SQLF_KTN_CATALOG_NOAUTH	314	Uint16	第 615 页的『catalog_noauth -“没有权限时允许的编目”』
cf_diaglevel	否	否	无	SQLF_KTN_CF_DIAGLEVEL	968	Uint16	第 612 页的『cf_diaglevel - CF 的诊断错误捕获级别配置参数』
cf_diagpath	否	否	无	SQLF_KTN_CF_DIAGPATH SQLF_KTN_CF_DIAGPATH_FULL	969	char(215)	第 612 页的『cf_diagpath -“CF 的诊断数据目录路径”配置参数』
cf_mem_sz	否	是	高	SQLF_KTN_CF_MEM_SZ	960	Uint32	第 613 页的『cf_mem_sz -“CF 内存”配置参数』
cf_num_conns	是	是	高	SQLF_KTN_CF_NUM_CONNS	966	Uint32	第 614 页的『cf_num_conns -“每个 CF 的每个成员的 CF 连接数”配置参数』
cf_num_workers	否	是	高	SQLF_KTN_CF_NUM_WORKERS	961	Uint32	第 614 页的『cf_num_workers -“工作程序线程数”配置参数』
clnt_krb_plugin	否	否	无	SQLF_KTN_CLNT_KRB_PLUGIN	812	char(33)	第 616 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 616 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 617 页的『cluster_mgr -“集群管理器名称”』
comm_bandwidth	是	否	中	SQLF_KTN_COMM_BANDWIDTH	307	float	第 617 页的『comm_bandwidth -“通信带宽”』
comm_exit_list	否	否	低	SQLF_KTN_COMM_EXIT_LIST	10121	char(129)	第 618 页的『comm_exit_list -“通信缓冲区出口库列表”』
conn_elapse	是	否	中	SQLF_KTN_CONN_ELAPSE	508	Uint16	第 618 页的『conn_elapse -“连接耗用时间”』
cpuspeed	是	否	高	SQLF_KTN_CPUSPEED	42	float	第 619 页的『cpuspeed -“CPU 速度”』
dft_account_str	是	否	无	SQLF_KTN_DFT_ACCOUNT_STR	28	char(25)	第 620 页的『dft_account_str -“缺省对方付费帐户”』
dft_monswitches • **dft_mon_bufpool** • **dft_mon_lock** • **dft_mon_sort** • **dft_mon_stmt** • **dft_mon_table** • **dft_mon_ timestamp** • **dft_mon_uow**	是	否	中	SQLF_KTN_DFT_MONSWITCHES[2] • SQLF_KTN_DFT_MON_BUFPOOL • SQLF_KTN_DFT_MON_LOCK • SQLF_KTN_DFT_MON_SORT • SQLF_KTN_DFT_MON_STMT • SQLF_KTN_DFT_MON_TABLE • SQLF_KTN_DFT_MON_TIMESTAMP • SQLF_KTN_DFT_MON_UOW	29 • 33 • 34 • 35 • 31 • 32 • 36 • 30	Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16 • Uint16	第 621 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 622 页的『dftdbpath -“缺省数据库路径”』
diaglevel	是	否	低	SQLF_KTN_DIAGLEVEL	64	Uint16	第 622 页的『diaglevel -“诊断错误捕获级别”』
diagpath	是	否	无	SQLF_KTN_DIAGPATH SQLF_KTN_DIAGPATH_FULL	65	char(215)	第 623 页的『diagpath -“诊断数据目录路径”』
dir_cache	否	否	中	SQLF_KTN_DIR_CACHE	40	Uint16	第 628 页的『dir_cache -“目录高速缓存支持”』
discover[3]	否	否	中	SQLF_KTN_DISCOVER	304	Uint16	第 629 页的『discover -“发现方式”』
discover_inst	是	否	低	SQLF_KTN_DISCOVER_INST	308	Uint16	第 630 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 630 页的『fcm_num_buffers -“FCM 缓冲区数”』

表 124. 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
fcm_num_ channels	是	是	中	SQLF_KTN_FCM_NUM_ CHANNELS	902	Uint32	第 631 页的 『fcm_num_channels -“FCM 通道数”』
fcm_parallelism	否	否	高	SQLF_KTN_FCM_NUM_PARALLELISM	848	Sint32	第 632 页的 『fcm_parallelism -“节点间通信并行性”』
fed_noauth	是	否	无	SQLF_KTN_FED_NOAUTH	806	Uint16	第 633 页的 『fed_noauth -“绕过联合认证”』
联合	是	否	中	SQLF_KTN_FEDERATED	604	Sint16	第 633 页的 『federated -“联合数据库系统支持”』
federated_async	是	是	中	SQLF_KTN_FEDERATED_ ASYNC	849	Sint32	第 634 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 634 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 635 页的 『group_plugin -“组插件”』
health_mon	是	否	低	SQLF_KTN_HEALTH_MON	804	Uint16	第 636 页的 『health_mon -“运行状况监视器”』
indexrec[4]	是	否	中	SQLF_KTN_INDEXREC	20	Uint16	第 636 页的 『indexrec -“索引重新创建时间”』
instance_memory	是	是	中	SQLF_KTN_INSTANCE_ MEMORY	803	Uint64	第 638 页的 『instance_memory -“实例内存”』
intra_parallel	否	否	高	SQLF_KTN_INTRA_PARALLEL	306	Sint16	第 640 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 641 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 642 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 642 页的 『keepfenced -“保留受防护进程”』
local_gssplugin	否	否	无	SQLF_KTN_LOCAL_ GSSPLUGIN	816	char(33)	第 643 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 643 页的 『max_connections -“最大客户机连接数”』
max_connretries	是	否	中	SQLF_KTN_MAX_ CONNRETRIES	509	Uint16	第 644 页的 『max_connretries -“节点连接重试次数”』
max_coordagents	是	是	中	SQLF_KTN_MAX_ COORDAGENTS	501	Sint32	第 645 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 646 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 647 页的 『max_time_diff -“成员间的最大时差”』
mon_heap_sz	是	是	低	SQLF_KTN_MON_HEAP_SZ	79	Uint16	第 649 页的 『mon_heap_sz -“数据库系统监视器堆大小”』
notifylevel	是	否	低	SQLF_KTN_NOTIFYLEVEL	605	Sint16	第 650 页的 『notifylevel -“通知级别”』
num_initagents	否	否	中	SQLF_KTN_NUM_ INITAGENTS	500	Uint32	第 651 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 652 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 652 页的 『num_poolagents -“代理程序池大小”』
numdb	否	否	低	SQLF_KTN_NUMDB	6	Uint16	第 653 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 654 页的 『query_heap_sz -“查询堆大小”』
resync_interval	否	否	无	SQLF_KTN_RESYNC_ INTERVAL	68	Uint16	第 656 页的 『resync_interval -“事务再同步时间间隔”』
rqrioblk	否	否	高	SQLF_KTN_RQRIOBLK	1	Uint16	第 657 页的 『rqrioblk -“客户机 I/O 块大小”』
sheapthres	否	否	高	SQLF_KTN_SHEAPTHRES	21	Uint32	第 658 页的 『sheapthres -“排序堆阈值”』
spm_log_file_sz	否	否	低	SQLF_KTN_SPM_LOG_FILE_ SZ	90	Sint32	第 659 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 660 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 660 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 660 页的 『spm_name -“同步点管理器名”』
srvcon_auth	否	否	无	SQLF_KTN_SRVCON_AUTH	815	Uint16	第 661 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 661 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 662 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 662 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”』
ssl_svr_keydb	否	否	无	SQLF_KTN_SSL_SVR_KEYDB	930	char(1023)	第 664 页的 『ssl_svr_keydb -“服务器上用于传入 SSL 连接的 SSL 密钥文件路径”配置参数』

表 124. 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
ssl_svr_stash	否	否	无	SQLF_KTN_SSL_SVR_STASH	931	char(1023)	第 665 页的『ssl_svr_stash -"服务器上用于传入 SSL 连接的 SSL 隐藏文件路径"配置参数』
ssl_svr_label	否	否	无	SQLF_KTN_SSL_SVR_LABEL	932	char(1023)	第 665 页的『ssl_svr_label -"服务器上用于传入 SSL 连接的密钥文件中的标签"配置参数』
ssl_svcename	否	否	无	SQLF_KTN_SSL_SVCENAME	933	char(14)	第 667 页的『ssl_svcename -"SSL 服务名称"配置参数』
ssl_cipherspecs	否	否	无	SQLF_KTN_SSL_CIPHERSPECS	934	char(255)	第 663 页的『ssl_cipherspecs -"服务器上支持的密码规范"配置参数』
ssl_versions	否	否	无	SQLF_KTN_SSL_VERSIONS	935	char(255)	第 667 页的『ssl_versions -"服务器上支持的 SSL 版本"配置参数』
ssl_clnt_keydb	否	否	无	SQLF_KTN_SSL_CLNT_KEYDB	936	char(1023)	第 663 页的『ssl_clnt_keydb -"客户机上用于出站 SSL 连接的 SSL 密钥文件路径"配置参数』
ssl_clnt_stash	否	否	无	SQLF_KTN_SSL_CLNT_STASH	937	char(1023)	第 664 页的『ssl_clnt_stash -"客户机上用于出站 SSL 连接的 SSL 隐藏文件路径"配置参数』
start_stop_time	是	否	低	SQLF_KTN_START_STOP_ TIME	511	Uint16	第 666 页的『start_stop_time -"启动和停止超时"』
svcename	否	否	无	SQLF_KTN_SVCENAME	24	char(14)	第 668 页的『svcename -"TCP/IP 服务名称"』
sysadm_group	否	否	无	SQLF_KTN_SYSADM_ GROUP	39	char(128)	第 668 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 669 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 669 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 670 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 670 页的『tm_database -"事务管理器数据库名称"』
tp_mon_name	否	否	无	SQLF_KTN_TP_MON_NAME	66	char(19)	第 671 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 672 页的『trust_allclnts -"信赖所有客户机"』
trust_clntauth	否	否	无	SQLF_KTN_TRUST_ CLNTAUTH	302	Uint16	第 673 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 673 页的『util_impact_lim -"实例影响策略"』
wlm_dispatcher	是	否	中	SQLF_KTN_WLM_ DISPATCHER	976	Uint16	第 674 页的『wlm_dispatcher -"工作负载管理分派器"』
wlm_disp_concur	是	否	低	SQLF_KTN_WLM_DISP_ CONCUR	977	Sint16	第 675 页的『wlm_disp_concur -"工作负载管理器分派器线程并行度"』
wlm_disp_cpu_shares	是	否	低	SQLF_KTN_WLM_DISP_ CPU_SHARES	979	Uint16	第 676 页的『wlm_disp_cpu_shares -"工作负载管理器分派器 CPU 份额"』
wlm_disp_min_util	是	否	低	SQLF_KTN_WLM_DISP_ MIN_UTIL	978	Uint16	第 676 页的『wlm_disp_min_util -"工作负载管理器分派器最低 CPU 利用率"』

注:

请在头文件 sqlenv.h 和 sqlutil.h 中查找有效值和配置参数使用的定义。

1.

```
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
```

2. 有效值:

```
SQLF_DSCVR_KNOWN (1)
SQLF_DSCVR_SEARCH (2)
```

3. 有效值:

```
SQLF_INX_REC_SYSTEM (0)
SQLF_INX_REC_REFERENCE (1)
SQLF_INX_REC_RESTART (2)
SQLF_INX_REC_RESTART_NO_REDO (3)
SQLF_INX_REC_ACCESS_NO_REDO (4)
```

4. 有效值:

```
SQLF_TRUST_ALLCLNTS_NO (0)
SQLF_TRUST_ALLCLNTS_YES (1)
SQLF_TRUST_ALLCLNTS_DRDAONLY (2)
```

5. 有效值:

```
SQL_ALTERNATE_AUTH_ENC_AES (0)
SQL_ALTERNATE_AUTH_ENC_AES_CMP (1)
SQL_ALTERNATE_AUTH_ENC_NOTSPEC (255)
```

表 125. 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 650 页的『nodetype -“实例节点类型”』
release	SQLF_KTN_RELEASE	101	Uint16	第 655 页的『release -“配置文件发行版级别”』

注:

请在头文件 sqlenv.h 和 sqlutil.h 中查找有效值和配置参数使用的定义。

1. 有效值:

```
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『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。

列“成员配置”仅应用于 DB2 pureScale 环境。虽然所有数据库配置参数都可全局配置，但此列指示数据库配置参数能否针对每个成员配置。有关可针对每个成员配置的数据库配置参数的更多信息，请参阅 DB2 pureScale Feature 数据库配置参数。

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 V9.5 起，可以更新并重置一些或所有分区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重置每个分区。

如果数据库由 **CREATE DATABASE** 或 sqlecrea API 创建，那么缺省情况下配置顾问程序会运行以使用自动计算的值更新数据库配置参数。如果数据库由添加了 AUTOCONFIGURE APPLY NONE 子句的 **CREATE DATABASE** 命令创建，或者 sqlecrea API 指定不运行配置顾问程序，那么配置参数设为缺省值。

表 126. 可配置的数据库配置参数

参数	可联机配置	自动	成员配置	性能影响	标记	标记值	数据类型	其他信息
alt_collate	否	否	否	无	SQLF_DBTN_ALT_COLLATE	809	Uint32	第 677 页的『alt_collate -“备用整理顺序”』
applheapsz	是	是	是	中	SQLF_DBTN_APPLHEAPSZ	51	Uint16	第 680 页的『applheapsz -“应用程序堆大小”』
appl_memory	是	是	是	中	SQLF_DBTN_APPL_MEMORY	904	Uint64	第 680 页的『appl_memory -“应用程序内存”配置参数』

表 126. 可配置的数据库配置参数 (续)

参数	可联机配置	自动	成员配置	性能影响	标记	标记值	数据类型	其他信息
archretrydelay	是	否	否	无	SQLF_DBTN_ARCHRETRYDELAY	828	Uint16	第 681 页的『archretrydelay -“出错时的归档重试延迟”』
• **auto_maint** • **auto_db_backup** • **auto_tbl_maint** • **auto_runstats** • **auto_stats_prof** • **auto_stmt_stats** • **auto_stats_views** • **auto_prof_upd** • **auto_reorg**	是	否	否	中	• SQLF_DBTN_AUTO_MAINT • SQLF_DBTN_AUTO_DB_BACKUP • SQLF_DBTN_AUTO_TBL_MAINT • SQLF_DBTN_AUTO_RUNSTATS • SQLF_DBTN_AUTO_STATS_PROF • SQLF_DBTN_AUTO_STMT_STATS • SQLF_DBTN_AUTO_STATS_VIEWS • SQLF_DBTN_AUTO_PROF_UPD • SQLF_DBTN_AUTO_REORG	• 831 • 833 • 835 • 837 • 839 • 905 • 980 • 844 • 841	Uint32	第 682 页的『auto_maint -“自动维护”』
auto_del_rec_obj	是	否	否	中	SQLF_DBTN_AUTO_DEL_REC_OBJ	912	Uint16	第 682 页的『auto_del_rec_obj -“自动删除恢复对象”配置参数』
autorestart	是	否	否	低	SQLF_DBTN_AUTO_RESTART	25	Uint16	第 685 页的『autorestart -“允许自动重新启动”』
auto_reval	是	否	是	中	SQLF_DBTN_AUTO_REVAL	920	Uint16	第 684 页的『auto_reval -“自动重新验证和失效”配置参数』
avg_appls	是	是	是	高	SQLF_DBTN_AVG_APPLS	47	Uint16	第 686 页的『avg_appls -“平均活动应用程序数”』
blk_log_dsk_ful	是	否	是	无	SQLF_DBTN_BLK_LOG_DSK_FUL	804	Uint16	第 687 页的『blk_log_dsk_ful -“日志磁盘满时阻止进行日志记录”』
blocknonlogged	是	否	是	低	SQLF_DBTN_BLOCKNONLOGGED	940	Uint16	第 688 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	是	中	SQLF_DBTN_CATALOGCACHE_SZ	56	Uint32	第 692 页的『catalogcache_sz -“目录高速缓存大小”』
chngpgs_thresh	否	否	是	高	SQLF_DBTN_CHNGPGS_THRESH	38	Uint16	第 693 页的『chngpgs_thresh -“已更改页数阈值”』
connect_proc	是	否	否	无	SQLF_DBTN_CONNECT_PROC	954	char(257)	第 695 页的『connect_proc - 连接过程名称数据库配置参数』
cur_commit	否	否	是	中	SQLF_DBTN_CUR_COMMIT	917	Uint32	第 619 页的『cur_commit -“当前已落实”配置参数』
database_memory	是	是	是	中	SQLF_DBTN_DATABASE_MEMORY	803	Uint64	第 696 页的『database_memory -“数据库共享内存大小”』
dbheap	是	是	是	中	SQLF_DBTN_DB_HEAP	58	Uint64	第 698 页的『dbheap -“数据库堆”』
db_mem_thresh	是	否	是	低	SQLF_DBTN_DB_MEM_THRESH	849	Uint16	第 699 页的『db_mem_thresh -“数据库内存阈值”』
decflt_rounding	否	否	否	无	SQLF_DBTN_DECFLT_ROUNDING	913	Uint16	第 701 页的『decflt_rounding -“十进制浮点数舍入”配置参数』

表 126. 可配置的数据库配置参数 (续)

参数	可联机配置	自动	成员配置	性能影响	标记	标记值	数据类型	其他信息
dec_to_char_fmt	是	是	是	中	SQLF_DBTN_DEC_TO_CHAR_FMT	• 0: V95 • 1: NEW	Uint16	第 701 页的『dec_to_char_fmt -“小数到字符函数”配置参数』
dft_degree	是	否	是	高	SQLF_DBTN_DFT_DEGREE	301	Sint32	第 703 页的『dft_degree -“缺省级别”』
dft_extent_sz	是	否	否	中	SQLF_DBTN_DFT_EXTENT_SZ	54	Uint32	第 703 页的『dft_extent_sz -“表空间的缺省扩展数据块大小”』
dft_loadrec_ses	是	否	是	中	SQLF_DBTN_DFT_LOADREC_SES	42	Sint16	第 704 页的『dft_loadrec_ses -“缺省装入恢复会话数”』
dft_mttb_types	否	否	否	无	SQLF_DBTN_DFT_MTTB_TYPES	843	Uint32	第 704 页的『dft_mttb_types -“为优化缺省维护的表类型”』
dft_prefetch_sz	是	是	否	中	SQLF_DBTN_DFT_PREFETCH_SZ	40	Sint16	第 705 页的『dft_prefetch_sz -“缺省预取大小”』
dft_queryopt	是	否	是	中	SQLF_DBTN_DFT_QUERYOPT	57	Sint32	第 706 页的『dft_queryopt -“缺省查询优化类”』
dft_refresh_age	否	否	否	中	SQLF_DBTN_DFT_REFRESH_AGE	702	char(22)	第 707 页的『dft_refresh_age -“缺省刷新持续时间”』
dft_schemas_dcc	是	否	否	中	SQLF_DBTN_DFT_SCHEMAS_DCC	10115	Uint16	第 707 页的『dft_schemas_dcc -“针对新模式的缺省数据捕获”配置参数』
discover_db	是	否	否	中	SQLF_DBTN_DISCOVER	308	Uint16	第 708 页的『discover_db -“发现数据库”』
dlchktime	是	否	否	中	SQLF_DBTN_DLCHKTIME	9	Uint32	第 709 页的『dlchktime -“检查死锁的时间间隔”』
enable_xmlchar	是	否	否	无	SQLF_DBTN_ENABLE_XMLCHAR	853	Uint32	第 709 页的『enable_xmlchar -“启用以 XML 为目标的转换”配置参数』
failarchpath	是	否	否	无	SQLF_DBTN_FAILARCHPATH	826	char(243)	第 710 页的『failarchpath -“故障转移日志归档路径”』
hadr_local_host	否	否	无	无	SQLF_DBTN_HADR_LOCAL_HOST	811	char(256)	第 711 页的『hadr_local_host -“HADR 本地主机名”』
hadr_local_svc	否	否	无	无	SQLF_DBTN_HADR_LOCAL_SVC	812	char(41)	第 712 页的『hadr_local_svc -“HADR 本地服务名称”』
hadr_peer_窗口	否	否	无	低（请参阅注 3）	SQLF_DBTN_HADR_PEER_WINDOW	914	Uint32	第 712 页的『hadr_peer_window -“HADR 对等窗口”配置参数』
hadr_remote_host	否	否	无	无	SQLF_DBTN_HADR_REMOTE_HOST	813	char(256)	第 713 页的『hadr_remote_host -“HADR 远程主机名”』
hadr_remote_inst	否	否	无	无	SQLF_DBTN_HADR_REMOTE_INST	815	char(9)	第 714 页的『hadr_remote_inst -“远程服务器的 HADR 实例名”』
hadr_remote_svc	否	否	无	无	SQLF_DBTN_HADR_REMOTE_SVC	814	char(41)	第 714 页的『hadr_remote_svc -“HADR 远程服务名称”』
hadr_replay_delay	是	否	无	无	SQLF_DBTN_HADR_REPLAY_DELAY	10119	Sint32	第 714 页的『hadr_replay_delay -“HADR 重演延迟”配置参数』
hadr_spool_limit	是	否	无	无	SQLF_DBTN_HADR_SPOOL_LIMIT	10112	Sint32	第 715 页的『hadr_spool_limit -“HADR 日志假脱机限制”配置参数』
hadr_syncmode	否	否	无	无	SQLF_DBTN_HADR_SYNCMODE	817	Uint32	第 716 页的『hadr_syncmode -“处于对等状态的日志写操作的 HADR 同步方式”』
hadr_target_list	是	否	无	无	SQLF_DBTN_HADR_TARGET_LIST	10114	char(890)	第 717 页的『hadr_target_list -“HADR 目标列表”数据库配置参数』

表 126. 可配置的数据库配置参数 (续)

参数	可联机配置	自动	成员配置	性能影响	标记	标记值	数据类型	其他信息
hadr_timeout	否	否	无	无	SQLF_DBTN_HADR_TIMEOUT	816	Uint32	第 719 页的『hadr_timeout -“HADR 超时值”』
indexrec[2]	是	否	否	中	SQLF_DBTN_INDEXREC	30	Uint16	第 636 页的『indexrec -“索引重新创建时间”』
locklist	是	是	是	高（当它影响升级时）	SQLF_DBTN_LOCK_LIST	704	Uint64	第 721 页的『locklist -“锁定列表的最大存储量”』
locktimeout	否	否	是	中	SQLF_DBTN_LOCKTIMEOUT	34	Sint16	第 724 页的『locktimeout -“锁定超时”』
log_appl_info	否	否	无	低	SQLF_DBTN_LOG_APPL_INFO	10111	Uint32	第 724 页的『log_appl_info -“应用程序信息日志记录”数据库配置参数』
log_ddl_stmts	是	否			SQLF_DBTN_LOG_DDL_STMTS	10110	Uint32	第 725 页的『log_ddl_stmts -“日志数据定义语言 (DDL) 语句数”数据库配置参数』
logarchcompr1	是	否	否	无	SQLF_DBTN_ LOGARCHCOMPR1	10123	char(252)	第 725 页的『logarchcompr1 -“主归档日志文件压缩”配置参数』
logarchcompr2	是	否	否	无	SQLF_DBTN_ LOGARCHCOMPR2	10124	char(252)	第 726 页的『logarchcompr2 -“辅助归档日志文件压缩”配置参数』
logarchmeth1	是	否	否	无	SQLF_DBTN_ LOGARCHMETH1	822	char(252)	第 726 页的『logarchmeth1 -“主日志归档方法”』
logarchmeth2	是	否	否	无	SQLF_DBTN_ LOGARCHMETH2	823	char(252)	第 728 页的『logarchmeth2 -“辅助日志归档方法”』
logarchopt1	是	否	否	无	SQLF_DBTN_LOGARCHOPT1	824	char(243)	第 729 页的『logarchopt1 -“主日志归档选项”』
logarchopt2	是	否	否	无	SQLF_DBTN_LOGARCHOPT2	825	char(243)	第 729 页的『logarchopt2 -“辅助日志归档选项”』
logbufsz	否	否	是	高	SQLF_DBTN_LOGBUFSZ	33	Uint16	第 730 页的『logbufsz -“日志缓冲区大小”』
logfilsiz	否	否	否	中	SQLF_DBTN_LOGFIL_SIZ	92	Uint32	第 730 页的『logfilsiz -“日志文件大小”』
logindexbuild	是	否	是	无	SQLF_DBTN_ LOGINDEXBUILD	818	Uint32	第 732 页的『logindexbuild -“创建的日志索引页数”』
logprimary	否	否	否	中	SQLF_DBTN_LOGPRIMARY	16	Uint16	第 733 页的『logprimary -“主日志文件数”』
logsecond	是	否	否	中	SQLF_DBTN_LOGSECOND	17	Uint16	第 734 页的『logsecond -“辅助日志文件数”』
max_log	是	是	是		SQLF_DBTN_MAX_LOG	807	Uint16	第 735 页的『max_log -“每个事务的最大日志数”』
maxappls	是	是	是	中	SQLF_DBTN_MAXAPPLS	6	Uint16	第 736 页的『maxappls -“最大活动应用程序数”』
maxfilop	是	否	是	中	SQLF_DBTN_MAXFILOP	3	Uint16	第 737 页的『maxfilop - 每个数据库打开的最大数据库文件数』
maxlocks	是	是	是	高（当它影响升级时）	SQLF_DBTN_MAXLOCKS	15	Uint16	第 737 页的『maxlocks -“锁定升级前锁定列表的最大百分比”』
min_dec_div_3	否	否	否	高	SQLF_DBTN_MIN_DEC_DIV_3	605	Sint32	第 739 页的『min_dec_div_3 -“十进制除法，小数位为 3 的”』
mincommit	是	否	是	高	SQLF_DBTN_MINCOMMIT	32	Uint16	第 740 页的『mincommit -“要分组的落实数”』

表 126. 可配置的数据库配置参数 (续)

参数	可联机配置	自动	成员配置	性能影响	标记	标记值	数据类型	其他信息
mirrorlogpath	否	否	否	低	SQLF_DBTN_MIRRORLOGPATH	806	char(242)	第 741 页的『mirrorlogpath -"镜像日志路径"』
mon_act_metrics	是	否	是	中	SQLF_DBTN_MON_ACT_METRICS	931	Uint16	第 742 页的『mon_act_metrics -"监视活动度量值"配置参数』
mon_deadlock	是	否	是	中	SQLF_DBTN_MON_DEADLOCK	934	Uint16	第 743 页的『mon_deadlock -"监视死锁"配置参数』
mon_locktimeout	是	否	是	中	SQLF_DBTN_MON_LOCKTIMEOUT	933	Uint16	第 744 页的『mon_locktimeout -"监视锁定超时"配置参数』
mon_lockwait	是	否	是	中	SQLF_DBTN_MON_LOCKWAIT	935	Uint16	第 744 页的『mon_lockwait -"监视锁定等待"配置参数』
mon_lw_thresh	是	否	是	中	SQLF_DBTN_MON_LW_THRESH	936	Uint32	第 745 页的『mon_lw_thresh -"监视锁定等待阈值"配置参数』
mon_lck_msg_lvl	是	否	是	无	SQLF_DBTN_MON_LCK_MSG_LVL	951	Uint16	第 746 页的『mon_lck_msg_lvl -"监视锁定事件通知消息"配置参数』
mon_obj_metrics	是	否	是	中	SQLF_DBTN_MON_OBJ_METRICS	937	Uint16	第 746 页的『mon_obj_metrics -"监视对象度量值"配置参数』
mon_pkglist_sz	是	否	是	低	SQLF_DBTN_MON_PKGLIST_SZ	950	Uint32	第 748 页的『mon_pkglist_sz -"监视程序包列表大小"配置参数』
mon_req_metrics	是	否	是	中	SQLF_DBTN_MON_REQ_METRICS	930	Uint16	第 749 页的『mon_req_metrics -"监视请求度量值"配置参数』
mon_uow_data	是	否	是	中	SQLF_DBTN_MON_UOW_DATA	932	Uint16	第 750 页的『mon_uow_data -"监视工作单元事件"配置参数』
mon_uow_execlist	是	否	是	中	SQLF_DBTN_MON_UOW_EXECLIST	957	Uint16	第 751 页的『mon_uow_execlist -"监视带有可执行列表的工作单元事件"配置参数』
mon_uow_pkglist	是	否	是	中	SQLF_DBTN_MON_UOW_PKGLIST	956	Uint16	第 751 页的『mon_uow_pkglist -"监视带有包列表的工作单元事件"配置参数』
newlogpath	否	否	否	低	SQLF_DBTN_NEWLOGPATH	20	char(242)	第 752 页的『newlogpath -"更改数据库日志路径"』
num_db_backups	是	否	否	无	SQLF_DBTN_NUM_DB_BACKUPS	601	Uint16	第 753 页的『num_db_backups -"数据库备份数"』
num_freqvalues	是	否	否	低	SQLF_DBTN_NUM_FREQVALUES	36	Uint16	第 754 页的『num_freqvalues -"保留的高频值数目"』
num_iocleaners	否	是	是	高	SQLF_DBTN_NUM_IOCLEANERS	37	Uint16	第 755 页的『num_iocleaners -"异步页清除程序的数目"』
num_ioservers	否	是	是	高	SQLF_DBTN_NUM_IOSERVERS	39	Uint16	第 756 页的『num_ioservers -"I/O 服务器数"』
num_log_span	是	是	是		SQLF_DBTN_NUM_LOG_SPAN	808	Uint16	第 757 页的『num_log_span -"跨越的日志数"』
num_quantiles	是	否	是	低	SQLF_DBTN_NUM_QUANTILES	48	Uint16	第 757 页的『num_quantiles -"列的分位数"』
numarchretry	是	否	是	无	SQLF_DBTN_NUMARCHRETRY	827	Uint16	第 758 页的『numarchretry -"出错时重试次数"』
overflowlogpath	是	否	否	中	SQLF_DBTN_OVERFLOWLOGPATH	805	char(242)	第 759 页的『overflowlogpath -"溢出日志路径"』
pckcachesz	是	是	是	高	SQLF_DBTN_PCKCACHE_SZ	505	Uint32	第 760 页的『pckcachesz -"程序包高速缓存大小"』
rec_his_retentn	否	否	否	无	SQLF_DBTN_REC_HIS_RETENTN	43	Sint16	第 763 页的『rec_his_retentn -"恢复历史记录保留期"』

表 126. 可配置的数据库配置参数 (续)

参数	可联机配置	自动	成员配置	性能影响	标记	标记值	数据类型	其他信息
section_actuals	是	否	否	高	SQLF_DBTN_SECTION_ACTUALS	952	Uint64	第 764 页的『section_actuals -“部分实际值”配置参数』
self_tuning_mem	是	否	是	高	SQLF_DBTN_SELF_TUNING_MEM	848	Uint16	第 765 页的『self_tuning_mem -“自调整内存功能”』
seqdetect	是	否	否	高	SQLF_DBTN_SEQDETECT	41	Uint16	第 766 页的『seqdetect -“顺序检测和提前读标志”』
sheapthres_shr	是	是	是	高	SQLF_DBTN_SHEAPTHRES_SHR	802	Uint32	第 767 页的『sheapthres_shr -“共享排序的排序堆阈值”』
smtp_server	是	否	是	无	SQLF_DBTN_SMTP_SERVER	926	char [] (String)	第 768 页的『smtp_server -“SMTP 服务器”』
softmax	否	否	否	中	SQLF_DBTN_SOFTMAX	5	Uint16	第 768 页的『softmax -“恢复范围和软检查点时间间隔”』
sortheap	是	是	是	高	SQLF_DBTN_SORT_HEAP	52	Uint32	第 770 页的『sortheap -“排序堆大小”』
sql_ccflags	是	否	是	无	SQLF_DBTN_SQL_CCFLAGS	927	char (1023)	第 771 页的『sql_ccflags - 条件编译标志』
stat_heap_sz	是	是	是	低	SQLF_DBTN_STAT_HEAP_SZ	45	Uint32	第 772 页的『stat_heap_sz -“统计信息堆大小”』
stmt_conc	是	否	是	中	SQLF_DBTN_STMT_CONC	919	Uint32	第 772 页的『stmt_conc -“语句集中器”配置参数』
stmtheap	是	是	是	中	SQLF_DBTN_STMT_HEAP	821	Uint32	第 773 页的『stmtheap -“语句堆大小”』
systime_period_adj	是	否	否	无	SQLF_DBTN_SYSTIME_PERIOD_ADJ	955	Uint16	第 775 页的『systime_period_adj -“调整临时 SYSTEM_TIME 时间段”数据库配置参数』
trackmod	否	否	否	低	SQLF_DBTN_TRACKMOD	703	Uint16	第 776 页的『trackmod -“启用跟踪已修改页”』
tsm_mgmtclass	是	否	是	无	SQLF_DBTN_TSM_MGMTCLASS	307	char(30)	第 776 页的『tsm_mgmtclass -“Tivoli Storage Manager 管理类”』
tsm_nodename	是	否	是	无	SQLF_DBTN_TSM_NODENAME	306	char(64)	第 777 页的『tsm_nodename -“Tivoli Storage Manager 节点名』
tsm_owner	是	否	是	无	SQLF_DBTN_TSM_OWNER	305	char(64)	第 777 页的『tsm_owner -“Tivoli Storage Manager 所有者名称』
tsm_password	是	否	是	无	SQLF_DBTN_TSM_PASSWORD	501	char(64)	第 777 页的『tsm_password -“Tivoli Storage Manager 密码』
util_heap_sz	是	否	是	低	SQLF_DBTN_UTIL_HEAP_SZ	55	Uint32	第 778 页的『util_heap_sz -“实用程序堆大小”』
vendoropt	是	否	否	无	SQLF_DBTN_VENDOROPT	829	char(242)	第 779 页的『vendoropt -“供应商选项”』<
wlm_collect_int	是	否	是	低	SQLF_DBTN_WLM_COLLECT_INT	907	Sint32	第 780 页的『wlm_collect_int -“工作负载管理收集时间间隔”配置参数』

表 126. 可配置的数据库配置参数 (续)

参数	可联机配置	自动	成员配置	性能影响	标记	标记值	数据类型	其他信息

注:

请在头文件 sqlenv.h 和 sqlutil.h 中查找有效值和配置参数使用的定义。

1. 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:

```
Default => Bit 1 on (xxxx xxxx xxxx xxx1): auto_maint
Bit 2 off (xxxx xxxx xxxx xx0x): auto_db_backup
Bit 3 on  (xxxx xxxx xxxx x1xx): auto_tbl_maint
Bit 4 on  (xxxx xxxx xxxx 1xxx): auto_runstats
Bit 5 off (xxxx xxxx xxx0 xxxx): auto_stats_prof
Bit 6 off (xxxx xxxx xx0x xxxx): auto_prof_upd
Bit 7 off (xxxx xxxx x0xx xxxx): auto_reorg
Bit 8 off (xxxx xxxx 0xxx xxxx): auto_storage
Bit 9 off (xxxx xxx0 xxxx xxxx): auto_stmt_stats
0 0 0 D

Maximum => Bit 1 on (xxxx xxxx xxxx xxx1): auto_maint
Bit 2 off (xxxx xxxx xxxx xx1x): auto_db_backup
Bit 3 on  (xxxx xxxx xxxx x1xx): auto_tbl_maint
Bit 4 on  (xxxx xxxx xxxx 1xxx): auto_runstats
Bit 5 off (xxxx xxxx xxx1 xxxx): auto_stats_prof
Bit 6 off (xxxx xxxx xx1x xxxx): auto_prof_upd
Bit 7 off (xxxx xxxx x1xx xxxx): auto_reorg
Bit 8 off (xxxx xxxx 1xxx xxxx): auto_storage
Bit 9 off (xxxx xxx1 xxxx xxxx): auto_stmt_stats
0 1 F F
```

2. 有效值:

```
SQLF_INX_REC_SYSTEM (0)
SQLF_INX_REC_REFERENCE (1)
SQLF_INX_REC_RESTART (2)
SQLF_INX_REC_RESTART_NO_REDO (3)
SQLF_INX_REC_ACCESS_NO_REDO (4)
```

3. 如果将 **hadr_peer_window** 参数设为一个非零时间值，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 127. 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 687 页的『backup_pending -“备份暂挂指示器”』
codepage	SQLF_DBTN_CODEPAGE	101	Uint16	第 694 页的『codepage -“用于数据库的代码页”』
codeset	SQLF_DBTN_CODESET	120	char(9)[1]	第 694 页的『codeset -“用于数据库的代码集”』
collate_info	SQLF_DBTN_COLLATE_INFO	44	char(260)	第 694 页的『collate_info -“整理信息”』
country/region	SQLF_DBTN_COUNTRY	100	Uint16	第 696 页的『country/region -“数据库地域代码”』
database_consistent	SQLF_DBTN_CONSISTENT	111	Uint16	第 696 页的『database_consistent -“数据库处于一致状态”』
database_level	SQLF_DBTN_DATABASE_LEVEL	124	Uint16	第 696 页的『database_level -“数据库发行版级别”』
hadr_db_role	SQLF_DBTN_HADR_DB_ROLE	810	Uint32	第 711 页的『hadr_db_role -“HADR 数据库角色”』
log_retain_status	SQLF_DBTN_LOG_RETAIN_STATUS	114	Uint16	第 725 页的『log_retain_status -“日志保留状态指示器”』
loghead	SQLF_DBTN_LOGHEAD	105	char(12)	第 732 页的『loghead -“第一个活动日志文件”』
logpath	SQLF_DBTN_LOGPATH	103	char(242)	第 732 页的『logpath -“日志文件的位置”』
multipage_alloc	SQLF_DBTN_MULTIPAGE_ALLOC	506	Uint16	第 752 页的『multipage_alloc -“启用多页文件分配”』

表 127. 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 759 页的『numsegs -“缺省 SMS 容器数”』
pagesize	SQLF_DBTN_PAGESIZE	846	Uint32	第 760 页的『pagesize - 数据库缺省页大小』
release	SQLF_DBTN_RELEASE	102	Uint16	第 655 页的『release -“配置文件发行版级别”』
restore_pending	SQLF_DBTN_RESTORE_PENDING	503	Uint16	第 763 页的『restore_pending -“复原暂挂”』
restrict_access	SQLF_DBTN_RESTRICT_ACCESS	852	Sint32	第 764 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 764 页的『rollfwd_pending -“前滚暂挂指示器”』
suspend_io	SQLF_DBTN_SUSPEND_IO	953	Uint16	第 774 页的『suspend_io -“数据库 I/O 操作状态”配置参数』
territory	SQLF_DBTN_TERRITORY	121	char(5)[2]	第 776 页的『territory -“数据库地域”』
user_exit_status	SQLF_DBTN_USER_EXIT_STATUS	115	Uint16	第 778 页的『user_exit_status -“用户出口状态指示器”』

注:

1. 在 HP-UX、Linux 和 Solaris 操作系统上为 char(17)。
2. 在 HP-UX、Linux 和 Solaris 操作系统上为 char(33)。

DB2 管理服务器 (DAS) 配置参数摘要

要点: V9.7 中已经不推荐使用“DB2 管理服务器 (DAS)”，在以后的发行版中可能会将其除去。DAS 在 DB2 pureScale 环境中不受支持。通过使用安全 Shell 协议的软件程序进行远程管理。有关更多信息，请参阅『不推荐使用DB2 管理服务器 (DAS)』，网址为: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html。

表 128. DAS 配置参数

参数	参数类型	其他信息
认证	可配置	第 780 页的『authentication -“认证类型 DAS”』
contact_host	可联机配置	第 781 页的『contact_host -“联系人列表的位置”』
das_codepage	可联机配置	第 781 页的『das_codepage -“DAS 代码页”』
das_territory	可联机配置	第 782 页的『das_territory -“DAS 地域”』
dasadm_group	可配置	第 782 页的『dasadm_group -“DAS 管理权限组名”』
db2system	可联机配置	第 783 页的『db2system -“DB2 服务器系统的名称”』
discover	可联机配置	第 784 页的『discover -“DAS 发现方式”』
exec_exp_task	可配置	第 785 页的『exec_exp_task -“执行已到期任务”』
jdk_64_path	可联机配置	第 721 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 785 页的『jdk_path -“Java 软件开发者工具箱安装路径 DAS”』
sched_enable	可配置	第 786 页的『sched_enable -“调度程序方式”』
sched_userid	参考	第 786 页的『sched_userid -“调度程序用户标识”』
smtp_server	可联机配置	第 786 页的『smtp_server -“SMTP 服务器”』
toolscat_db	可配置	第 787 页的『toolscat_db -“工具目录数据库”』
toolscat_inst	可配置	第 787 页的『toolscat_inst -“工具目录数据库实例”』
toolscat_schema	可配置	第 787 页的『toolscat_schema -“工具目录数据库模式”』

ingest 实用程序配置参数摘要

表 129. ingest 实用程序配置参数

参数	参数类型	其他信息
commit_count	可配置	第 796 页的『commit_count -“落实计数”配置参数』
commit_period	可配置	第 797 页的『commit_period -“落实时间段”配置参数』
num_flushers _per_partition	可配置	第 798 页的『num_flushers_per_partition -“每个数据库分区的清仓程序数”配置参数』
num_formatters	可配置	第 798 页的『num_formatters -“格式化程序数”配置参数』
pipe_timeout	可配置	第 799 页的『pipe_timeout -“管道超时”配置参数』
retry_count	可配置	第 799 页的『retry_count -“重试计数”配置参数』
retry_period	可配置	第 799 页的『retry_period -“重试时间段”配置参数』
shm_max_size	可配置	第 800 页的『shm_max_size -“共享内存的最大大小”配置参数』

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 130. 配置参数节标题描述

节标题	描述和可能的值
配置类型	可能的值包括: • 数据库管理器 • 数据库 • 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: • 客户机 • 带有本地客户机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 • DB2 管理服务器 • OLAP 函数 • 带有本地客户机和远程客户机的分区数据库服务器 • 启用联合时带有本地客户机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	可能的值包括: • 可配置（必须重新启动数据库管理器才能使更改生效） • 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） • 供参考（值仅供参考，不能更新） • 可由 DB2 pureScale 环境中的成员配置
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。

表 130. 配置参数节标题描述 (续)

节标题	描述和可能的值
计量单位	如果适用，那么将列示计量单位。可能的值包括: • 字节 • 计数器 • 每秒兆字节 • 毫秒 • 分钟 • 页 (4 KB) • 百分比 • 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: • 立即 • 语句边界 • 事务边界 • 连接
分配时间	如果适用，那么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，那么它指示数据库管理器释放配置参数的时间。
限制	如果适用，那么它列示适用于配置参数的所有限制。
局限性	如果适用，那么它列示适用于配置参数的所有局限性。
建议	如果适用，那么它列示适用于配置参数的所有建议。
使用说明	如果适用，那么它列示适用于配置参数的所有用法说明。

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**): 如果启用了连接集中器，那么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 编译器选择的存取方案。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存取方案而不立即读取已更改的参数值。要重置程序包高速缓存，请执行 FLUSH PACKAGE CACHE 语句。

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，这些数据源是数据库管理系统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存取方案时将考虑其他存取方案备用。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 编译器选择的存取方案:

- 当创建或更改缓冲池时指定的缓冲池大小

  当优化器选择存取方案时，优化器要考虑将页从磁盘访存至缓冲池的 I/O 成本并估算满足查询所需要的 I/O 次数。估算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
- 缺省级别 (**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 (1) 表示无分区内并行性。值负 1 (-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 (**dft_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** 参数来帮助估算运行时期间可用于所选存取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存取方案。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。如果某个排序不需要系统临时表来存储数据列表，那么该排序始终会提高性能，并且会在可能的情况下使用该排序。

当选择存取方案时，优化器将通过以下操作来估算排序操作的成本，包括评估是否可以在单个顺序访问中读取排序:

– 估算要排序的数据量

– 查看 **sortheap** 参数，以确定是否有足够的空间在单个顺序访问中读取排序。

- 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，那么它为存取方案选择一个表级别锁定，这样就不会在查询执行期间因锁定升级而需要额外的处理时间。

- CPU 速度（**cpuspeed**）

  优化器使用 CPU 速度来估算执行特定操作的成本。CPU 成本估算和各种 I/O 成本估算有助于选择查询的最佳存取方案。

  一台机器的 CPU 速度可显著影响所选的存取方案。在安装或升级数据库时，此配置参数将自动设为适当的值。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存取方案。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算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。

影响 DB2 pureScale Feature 的配置参数

在 DB2 pureScale 环境中，数据库配置参数被指定为*全局*数据库配置参数或*每成员*数据库配置参数。此差别允许 DB2 pureScale Feature 实例受益于全局数据库配置一致性（即使容许成员间存在差别）。

全局数据库配置参数

对于 DB2 实例中的所有成员，全局数据库配置参数共用单个一致值。不管哪个成员发出对全局参数的值的更新，此更新都会在所有成员中全局应用。全局数据库配置一致性确保 DB2 实例中的所有成员以同一方式访问和处理同一组数据。

每成员数据库配置参数

每成员数据库配置参数在 DB2 实例中的每个成员间可具有不同值。缺省情况下，除非对每成员参数的值的更新仅针对特定成员指定，否则此更新将在所有成员中全局应用。DB2 实例中的任何成员都可发出更新。DB2 pureScale Feature 中的每成员数据库配置参数适合在需要非同质环境时使用。

对于每个成员上的可用资源不同的数据库，在每个成员上定制数据库配置可优化数据库性能。例如，每个成员上的可用内存的差别可受益于每个成员上的独有数据库配置。

成员映射至不同功能的应用程序的数据库可受益于每个成员上的定制数据库配置。

在 DB2 pureScale 环境中更新数据库配置参数

可使用 DB2 命令行处理器 (CLP) 或使用 DB2 配置 API 在 DB2 pureScale 实例中更新数据库配置参数。可选择使用 **MEMBER** 子句，此子句仅适合与每成员数据库配置参数配合使用。

更新全局数据库配置参数

全局数据库配置参数存储在单个数据库配置文件中。更新成员负责更新数据库配置文件。仅当更新成员无法写至全局配置文件时，对全局参数的更新才会失败。

注: 某个成员无法向配置文件写入内容时，不需要任何回滚，因为 DB2 pureScale 实例中的所有成员处于一致状态。

如果尝试使用 **MEMBER** 子句更新全局数据库配置参数，那么您会接收到错误 (SQL5125N)。

更新每成员数据库配置参数

只能使用同一配置参数的不同值来配置 DB2 pureScale 实例内的成员。使用 **MEMBER** 子句来指定对实例内的单个成员的更新。如果省略 **MEMBER** 子句，那么会在实例内的所有成员中应用此更改。

以下命令将 MEMBER 2 的 **util_heap_sz** 参数更新为值 5000:

```
UPDATE DATABASE CONFIGURATION FOR WSDB MEMBER 2 USING UTIL_HEAP_SZ 5000
```

只有 MEMBER 2 的 **util_heap_sz** 值更新为 5000。如果省略 **MEMBER** 子句，那么 **util_heap_sz** 的值对于 DB2 pureScale 实例中的所有成员都更新为 5000。

此命令可由 DB2 pureScale 实例中的任何成员发出。如果 MEMBER 2 关闭或变为不活动，那么配置更新可能失败也可能未失败。DB2 pureScale 实例中的每个成员可写至另一成员的配置文件，所以仅当更新成员无法成功写至 MEMBER 2 的配置文件时更新失败。

注: 如果未指定 **MEMBER** 子句并且更新成员尝试更新 MEMBER 2 的配置文件但失败，那么会在整个 DB2 pureScale 实例中回滚此更新。如果回滚也失败，那么配置文件可能处于不一致状态。

对于可在数据库处于联机状态时更新的数据库配置参数，不需要采取其他措施就可确保高速缓存的和内存中的值在 DB2 pureScale 实例中保持一致。

注: 全局配置参数的值存储在分区全局目录的全局配置文件中，每成员配置参数的值存储在每个成员的特定于成员的目录中。

DB2 pureScale Feature 配置参数

对于 IBM DB2 pureScale Feature，有一些支持成员和集群高速缓存设施（又称为 CF）的配置参数。这些数据库配置参数分为*全局*和*每成员*参数。

DB2 pureScale Feature数据库管理器配置参数

下表列示集群高速缓存设施的数据库管理器配置参数。数据库管理器配置参数适用于实例级别配置。

表 131. DB2 pureScale Feature数据库管理器配置参数的摘要

参数名称	详细信息
cf_diaglevel	指定 cfdiag.log 文件中记录的诊断错误的级别。
cf_diagpath	指定包含 CF 诊断日志文件的目录，此文件是根据 **cf_diaglevel** 参数的值生成的。
cf_mem_size	确定 CF 使用的内存总计。此值必须小于 CF 主机上的物理内存量。
cf_num_conns	确定 CF 连接池的初始大小。
cf_num_workers	确定 CF 服务器启动的工作程序线程数。通常此值应小于或等于 CF 服务器上的 CPU 数。

DB2 pureScale Feature数据库配置参数

下表列示集群高速缓存设施的数据库配置参数。这些数据库配置参数应用于数据库级别配置。

注: CF 数据库配置参数 cf_db_mem_sz 应用于数据库范围级别。余下 CF 数据库配置参数应用于结构级别并且具有由 cf_db_mem_sz 确定的上限。有关数据库配置参数之间的关系的更多信息，请参阅“为数据库配置集群高速缓存设施内存”主题。

表 132. DB2 pureScale Feature 全局数据库配置参数的摘要

参数名称	详细信息
cf_catchup_trgt	确定用于完成追赶操作以使新添加或新重新启动的 CF 处于与现有主 CF 对等的状态的目标时间（以分钟计）。
cf_db_mem_sz	确定当前数据库的总 CF 内存限制。
cf_gbp_sz	确定 CF 为当前数据库的组缓冲池分配的总内存大小。
cf_lock_sz	确定 CF 为当前数据库的锁定结构分配的总内存大小。
cf_sca_sz	确定 CF 为当前数据库的共享通信区分配的总内存大小。

以下 DB2 数据库配置参数被指定为 DB2 pureScale Feature 的*全局*数据库配置参数。有关全局数据库配置参数的更多信息，请参阅“DB2 pureScale Feature数据库配置参数”主题。

表 133. 可配置的全局数据库配置参数

参数	其他信息
alt_collate	**alt_collate -"备用整理顺序"配置参数**
archretrydelay	**archretrydelay -"出错时的归档重试延迟"配置参数**
auto_del_rec_obj	auto_del_rec_obj -“自动删除恢复对象”配置参数

表 133. 可配置的全局数据库配置参数 (续)

参数	其他信息
• auto_maint • auto_db_backup • auto_tbl_maint • auto_runstats • auto_stats_prof • auto_stmt_stats • auto_prof_upd • auto_reorg	auto_maint -"自动维护"配置参数
autorestart	autorestart -"允许自动重新启动"配置参数
decflt_rounding	decflt_rounding -"十进制浮点数舍入"配置参数
dft_extent_sz	dft_extent_sz -"表空间的缺省扩展数据块大小"配置参数
dft_mttb_types	dft_mttb_types -"为优化缺省维护的表类型"配置参数
dft_prefetch_sz	dft_prefetch_sz -"缺省预取大小"配置参数
dft_refresh_age	dft_refresh_age -"缺省刷新持续时间"配置参数
dft_sqlmathwarn	dft_sqlmathwarn -"出现算术异常时继续"配置参数
discover_db	discover_db -"发现数据库"配置参数
dlchktime	dlchktime -"检查死锁的时间间隔"配置参数
enable_xmlchar	enable_xmlchar -"启用以 XML 为目标的转换"配置参数
failarchpath	failarchpath -"故障转移日志归档路径"配置参数
indexrec	indexrec -"索引重新创建时间"配置参数
locktimeout	locktimeout -"锁定超时"配置参数
logarchmeth1	logarchmeth1 -"主日志归档方法"配置参数
logarchmeth2	logarchmeth2 -"辅助日志归档方法"配置参数
logarchopt1	logarchopt1 -"主日志归档选项"配置参数
logarchopt2	logarchopt2 -"辅助日志归档选项"配置参数
logfilsiz	logfilsiz -"日志文件大小"配置参数
logprimary	logprimary -"主日志文件数"配置参数
logsecond	logsecond -"辅助日志文件数"配置参数
min_dec_div_3	min_dec_div_3 -"十进制除法，小数位为 3"配置参数
mirrorlogpath	mirrorlogpath -"镜像日志路径"配置参数
mon_act_metrics	mon_act_metrics -"监视活动度量值"配置参数
mon_deadlock	mon_deadlock -"监视死锁"配置参数
mon_locktimeout	mon_locktimeout -"监视锁定超时"配置参数
mon_lockwait	mon_lockwait -"监视锁定等待"配置参数
mon_lw_thresh	mon_lw_thresh -"监视锁定等待阈值"配置参数
mon_obj_metrics	mon_obj_metrics -"监视对象度量值"配置参数
mon_req_metrics	mon_req_metrics -"监视请求度量值"配置参数
mon_uow_data	mon_uow_data -"监视工作单元事件"配置参数
newlogpath	newlogpath -"更改数据库日志路径"配置参数
num_db_backups	num_db_backups -"数据库备份数"配置参数
num_freqvalues	num_freqvalues -"保留的高频值数目"配置参数
numarchretry	numarchretry -"出错时的重试次数"配置参数
overflowlogpath	overflowlogpath -"溢出日志路径"配置参数
rec_his_retentn	rec_his_retentn -"恢复历史记录保留期"配置参数
seqdetect	seqdetect -"顺序检测标志"配置参数
softmax	softmax -"恢复范围和软检查点时间间隔"配置参数
trackmod	trackmod -"启用跟踪已修改页"配置参数
vendoropt	vendoropt -"供应商选项"配置参数

以下 DB2 数据库配置参数被指定为 DB2 pureScale Feature 的*每成员*数据库配置参数。有关每成员数据库配置参数的更多信息，请参阅“DB2 pureScale Feature数据库配置参数”主题。

表 134. 可配置的每成员数据库配置参数

参数	其他信息
applheapsz	applheapsz -“应用程序堆大小”配置参数
appl_memory	appl_memory -"应用程序内存"配置参数
auto_reval	
avg_appls	avg_appls -"平均活动应用程序数"配置参数
blk_log_dsk_ful	blk_log_dsk_ful -"日志磁盘变满时阻止进行日志记录"配置参数
blocknonlogged	blocknonlogged -"禁止创建允许不记录日志的活动的表"配置参数
catalogcache_sz	catalogcache_sz -"目录高速缓存大小"配置参数
chngpgs_thresh	chngpgs_thresh -"已更改页数阈值"配置参数
connect_proc	connect_proc -"连接过程名称"数据库配置参数
cur_commit	cur_commit -"当前已落实"配置参数
database_memory	database_memory -"数据库共享内存大小"配置参数
dbheap	dbheap -"数据库堆"配置参数
db_mem_thresh	db_mem_thresh -"数据库内存阈值"配置参数
dec_to_char_fmt	dec_to_char_fmt -"小数到字符函数"配置参数
dft_degree	dft_degree -"缺省级别"配置参数
dft_loadrec_ses	dft_loadrec_ses -"缺省装入恢复会话数"配置参数
dft_queryopt	dft_queryopt -"缺省查询优化类"配置参数
discover_db	discover_db -"发现数据库"配置参数
hadr_local_host	hadr_local_host -"HADR 本地主机名"配置参数
hadr_local_svc	hadr_local_svc -"HADR 本地服务名称"配置参数
hadr_peer_ window	hadr_peer_window -"HADR 对等窗口"配置参数
hadr_remote_host	hadr_remote_host -"HADR 远程主机名"配置参数
hadr_remote_inst	hadr_remote_inst -"远程服务器的 HADR 实例名"配置参数
hadr_remote_svc	hadr_remote_svc -"HADR 远程服务名称"配置参数
hadr_syncmode	hadr_syncmode -"处于对等状态的日志写操作的 HADR 同步方式"配置参数
hadr_timeout	hadr_timeout -"HADR 超时值"配置参数
locklist	locklist -"锁定列表的最大存储量"配置参数
locktimeout	locktimeout -"锁定超时"配置参数
logbufsz	logbufsz -"日志缓冲区大小"配置参数
logindexbuild	logindexbuild -"创建的日志索引页数"配置参数
max_log	max_log -"每个事务的最大日志数"配置参数
maxappls	maxappls -"最大活动应用程序数"配置参数
maxfilop	maxfilop -"对每个应用程序打开的最大数据库文件数"配置参数
maxlocks	maxlocks -"升级前锁定列表的最大百分比"配置参数
mincommit	mincommit -"要分组的落实数"配置参数
mon_act_metrics	mon_act_metrics -"监视活动度量值"配置参数
mon_deadlock	mon_deadlock -"监视死锁"配置参数
mon_locktimeout	mon_locktimeout -"监视锁定超时"配置参数
mon_lockwait	mon_lockwait -"监视锁定等待"配置参数
mon_lw_thresh	mon_lw_thresh -"监视锁定等待阈值"配置参数
mon_obj_metrics	mon_obj_metrics -"监视对象度量值"配置参数
mon_req_metrics	mon_req_metrics -"监视请求度量值"配置参数
mon_uow_data	mon_uow_data -"监视工作单元事件"配置参数
num_iocleaners	num_iocleaners -"异步页清除程序数"配置参数
num_ioservers	num_ioservers -"I/O 服务器数"配置参数
num_log_span	num_log_span -"跨越的日志数"配置参数

表 134. 可配置的每成员数据库配置参数 (续)

参数	其他信息
num_quantiles	num_quantiles -"列的分位数"配置参数
numarchretry	numarchretry -"出错时的重试次数"配置参数
pckcachesz	pckcachesz -"程序包高速缓存大小"配置参数
self_tuning_mem	self_tuning_mem -"自调整内存"配置参数
sheapthres_shr	sheapthres_shr -"共享排序的排序堆阈值"配置参数
sortheap	sortheap -"排序堆大小"配置参数
stat_heap_sz	stat_heap_sz -"统计信息堆大小"配置参数
stmt_conc	stmt_conc -"语句集中器"配置参数
stmtheap	stmtheap -"语句堆大小"配置参数
tsm_mgmtclass	tsm_mgmtclass -"Tivoli Storage Manager 管理类"配置参数
tsm_nodename	tsm_nodename -"Tivoli Storage Manager 节点名"配置参数
tsm_owner	tsm_owner -"Tivoli Storage Manager 所有者名称"配置参数
tsm_password	tsm_password -"Tivoli Storage Manager 密码"配置参数
util_heap_sz	util_heap_sz -"实用程序堆大小"配置参数
wlm_collect_int	wlm_collect_int -"工作负载管理收集时间间隔"配置参数

配置更改后重新编译查询

为观察影响查询优化的配置更改的效果，可能需要让查询优化器重新编译已高速缓存的语句。

过程

可通过执行下列任何操作以便让查询优化器重新编译语句:

- 使用 **RUNSTATS** 命令来使高速缓存动态语句对特定表失效:

  ```
  RUNSTATS ON TABLE <tableschema>.<tablename>
     WITH DISTRIBUTION AND SAMPLED DETAILED INDEXES ALL
  ```

  注: 这会刷新表统计信息，后续查询编译将使用新统计信息和新配置设置。
- 除去程序包高速缓存中当前的所有高速缓存动态 SQL 语句:

  ```
  FLUSH PACKAGE CACHE DYNAMIC
  ```

配置 max_coordagents 和 max_connections 时的限制和行为

在 V9.5 中，**max_coordagents** 和 **max_connections** 参数的缺省值将为 AUTOMATIC，并且 **max_coordagents** 设为 200 且 **max_connections** 设为 -1（也就是说，设为 **max_coordagents** 的值）。这些设置将集中器设为 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。具体而言，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 **START DBM** 时 **max_coordagents** 小于 **max_connections**（集中器处于打开状态），那么对这两个参数执行的所有联机更新都必须保持 **max_coordagents < max_connections** 这种关系。同样，如果在 **START DBM** 时 **max_coordagents** 大于或等于 **max_connections**（集中器处于关闭状态），那么执行的所有联机更新都必须保持此关系。

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: 当集中器处于打开状态时，即使这两个配置参数设为 AUTOMATIC，指定给它们的值也很重要。
- 如果两个参数都设为 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. 当集中器处于打开状态时，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行连接，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** 设为 AUTOMATIC 并且值大于 **max_coordagents**（即集中器处于打开状态），并且 **max_coordagents** 未设为 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. 这两个参数都设为 AUTOMATIC
2. 集中器已启用并且 **max_connections** 设为 AUTOMATIC，而 **max_coordagents** 未设为此值。

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 为每个代理程序线程堆栈分配的内存。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

计量单位

页 (4 KB)

分配时间

Linux 和 UNIX: 创建线程时，堆栈空间（进程虚拟内存）根据需要分配或在主 DB2 服务器进程中重复使用。根据需要使用/落实堆栈内存。

Windows: 创建线程时，AGENT_STACK_SZ 表示最初落实的堆栈内存。根据需要使用/落实其他堆栈内存。

释放时间

Linux 和 UNIX: 线程终止时，堆栈空间（进程虚拟内存）保留供重复使用，DB2 服务器关闭时，会释放该空间。

Windows: 线程终止时，会释放堆栈空间和内存。

在 UNIX 和 Linux 上，**agent_stack_sz** 将向上舍入为下一个更大的 2 次幂值。对于 Linux 和 UNIX 系统上的大多数工作负载来说，这些缺省设置应该足够使用。

在 Windows 环境中，此参数用于对每个代理程序设置最初落实的堆栈大小。不管此设置如何，每个代理程序堆栈最多可增长至最小保留堆栈大小（在 32 位版本的 Windows 上为 256 KB，在 64 位版本的 Windows 上为 2 MB）。如果超过此大小，那么代理程序堆栈可能会耗尽空间并返回错误。

Windows 使用“保留”堆栈的概念（即堆栈可增长至的最大值）和“已落实”堆栈（即创建时落实至堆栈的内存量）。此外，还将保护页添加至指定的已落实堆栈大小，以确定必需的最小保留堆栈空间。例如，如果 **agent_stack_sz**（已落实堆栈）设为值 16，那么将添加 1 个保护页，因此保留堆栈大小必须至少为 17 页。通过将 **agent_stack_sz**（即，已落实堆栈）设为导致最小保留堆栈大小大于缺省保留堆栈大小（64 页）的值，可增大最大（即，保留）代理程序堆栈大小。请注意，Windows 使用 1 MB 的倍数将保留堆栈大小设置为超过 256 KB。例如，在 32 位 Windows 上，将代理程序堆栈大小设为 [64 - 255] 4-KB 页范围内的值会导致最大堆栈为 1 MB，也就是 64*4 = 244 KB 和 255*4 = 1020 KB（向上舍入为 1 MB）。将 **agent_stack_sz** 的值设为小于缺省保留堆栈大小的值不会对最大限制产生影响，因为在必要时，堆栈仍然会增大，直至缺省保留堆栈大小。

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。使用 db2hdr 实用程序更改缺省保留堆栈大小的优点是它提供了更细粒度，因此允许将堆栈大小设为最低必需堆栈大小。在 32 位 Windows 上，这将节省虚拟地址空间。但是，您必须停止然后重新启动 DB2，对 db2syscs.exe 进程的更改才能生效，并且进行任何修订包升级后都必须再次执行此方法。

建议:

如果您要在 32 位 Windows 环境中使用大量 XML 数据或复杂 XML 数据，那么应该将 **agent_stack_sz** 的值更新为至少 64 个 4 KB 页。在模式注册或 XML 文档验证期间，非常复杂的 XML 模式可能要求 **agent_stack_sz** 的值为大得多的值。

此限制对于大部分数据库操作已足够。

注意

- 代理程序堆栈内存不会将实例内存使用量计算在内。
- 虽然可在配置 **agent_stack_sz** 时指定较高堆栈空间分配来获得最大使用率，但平均下来，线程只使用少量已分配堆栈空间。只有此较少量内存需要系统内存。

- 在 HP-UX 平台上，线程堆栈空间需要交换预留。大致总交换需求将等于线程数峰值 * (**agent_stack_sz**)。此计算中 **agent_stack_sz** 的值向上舍入为下一个更大的 2 次幂值。

agentpri -“代理程序的优先级”

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。

要点: 从 V9.5 开始，不推荐使用 **agentpri** 数据库管理器配置。在以前的 V9.5 数据服务器和客户机中仍可以使用此参数。此外，V10.1 中已经不推荐使用 WLM 服务类的代理程序优先级，在以后的发行版中可能会将其除去。请开始改用 WLM 分派器功能或代理程序优先级。有关更多信息，请参阅《*DB2 V10.1 新增内容*》中的『已经不推荐使用服务类的代理程序优先级』。

注: 下列信息仅适用于 V9.5 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

- 带有本地客户机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
- 带有本地客户机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]

Linux -1 (system) [1 - 99]

HP-UX

-1 (system) [0 - 31]

将此参数设为 -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。将此参数设为除 -1 或 system 以外的值时，数据库管理器将在静态优先级设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 平台上将此参数设为非缺省值，那么不能使用控制器来更改代理程序的优先级。
- 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。

alt_diagpath -“备用诊断数据目录路径”

此参数允许您指定在主诊断数据路径 **diagpath** 不可用时所使用的 DB2 诊断信息的标准备用路径。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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，，'"*pathname* $h"'，'"*pathname* $h/*trailing-dir*"',,'"*pathname* $n"'，[2] '"*pathname* $n/*trailing-dir*"'，'"*pathname* $m"'，'"*pathname* $m/*trailing-dir*"'，'"*pathname* hn"',[3] '"*pathname* hn/*trailing-dir*"','"*pathname* hm"'，or '"*pathname* hm/*trailing-dir*"']

符号

pathname

在主诊断数据目录路径不可用时使用的目录路径

$h 解析为 HOST_*hostname*

2. 建议不要使用 $n，将来的发行版可能会将其除去。

3. 建议不要使用 hn，将来的发行版可能会将其除去。

注: 从 V10 开始，在 DB2 pureScale 环境 中，$h 是指成员的原始主机。

$n 解析为 NODE*number*

$m 解析为 DIAG*_number*。请注意，不管 DIAG*_number* 是指数据库分区、CF 还是成员，都会使用该设置。

/trailing-dir

单个目录，或者是跟在 $h 或 $n 后面的目录和子目录

有下列值可用:

- '"*pathname* $h"'
- '"*pathname* $h/*trailing-dir*"'
- '"*pathname* $n"'
- '"*pathname* $n/*trailing-dir*"'
- '"*pathname* $m"'
- '"*pathname* $m/*trailing-dir*"'
- '"*pathname* hn"'
- '"*pathname* hn/*trailing-dir*"'
- '"*pathname* hm"'
- '"*pathname* hm/*trailing-dir*"'

备用诊断数据目录可与使用 **diagpath** 参数设置的主诊断数据目录包含相同的诊断数据。如果设置了 **alt_diagpath** 并且主诊断数据目录变得不可用，那么诊断日志记录将在所指定的备用诊断数据目录路径中继续，然后在主诊断路径再次变为可用时在其原始位置继续。如果此参数为空，并且由 **diagpath** 参数所指定的主诊断数据目录不可用，那么将不再写入诊断信息，直到主诊断路径再次变得可用为止。为了提高弹性，请将备用诊断数据目录设为与主诊断数据目录指向不同的文件系统。

从 V10.1 开始，缺省情况下，备用诊断数据目录路径会写至每个成员和 CF 的专用 db2diag.log。要复原至前发行版的行为（在这些发行版中，诊断数据写至同一目录），请指定带有 *pathname* 且不带标记（$h、$n 或 $m）的 **alt_diagpath**。

注:

- 在某些 Linux 和 UNIX 系统上，为了避免操作系统 Shell 解释 $ 符号，必须在双引号外部再添加一个单引号，如上面的语法中所示。
- 在 CLP 交互方式下，或者在命令是从输入文件中读取并执行的情况下，不需要添加双引号。
- $h、$m 和 $n 不区分大小写。
- **alt_diagpath** 的动态行为未扩展至所有进程。
- DB2 服务器进程 **db2sysc** 可以检测动态更改，例如，当您对实例附件发出 **UPDATE DATABASE MANAGER CONFIGURATION** 命令时。
- 当 DB2 客户机和应用程序进程启动时，它们将使用 **alt_diagpath** 配置参数设置，且不会检测任何动态更改。
- 在 UNIX 系统上，如果 **diagpath** 和 **alt_diagpath** 都不可用，那么会将 db2 诊断消息转储到 syslog 文件中。
- 没有用于 **alt_diagpath** 配置参数的缺省目录。

- **alt_diagpath** 与 **diagpath** 配置参数互斥。不能将它们设为同一个目录路径。
- 如果 **alt_diagpath**（或者 **diagpath**）不可用，那么这意味着诊断数据转储由于发生了错误（例如：删除了目录、磁盘错误、丢失了磁盘、网络问题、文件许可权错误或者磁盘已满）而失败。

alternate_auth_enc -“服务器上用于传入连接的备用加密算法”配置参数

此配置参数指定用于对提交给 DB2 服务器服务器以进行认证的用户标识和密码进行加密的备用加密算法。具体而言，此参数影响 DB2 客户机与 DB2 数据库服务器之间协商确定的认证方法是 SERVER_ENCRYPT 时使用的加密算法。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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]

NOT_SPECIFIED [AES_CMP; AES_ONLY]

如果 DB2 客户机与 DB2 服务器之间协商确定的认证方法是 SERVER_ENCRYPT，那么将对所提交的用于在 DB2 数据库服务器上进行认证的用户标识和密码进行加密。协商确定的认证方法取决于服务器上的认证类型设置以及客户机所请求使用的认证类型。用于对用户标识和密码进行加密的所选加密算法取决于 **alternate_auth_enc** 数据库管理器配置参数的设置。根据此设置的不同，此算法可以是 DES 或 AES。

使用缺省值（NOT_SPECIFIED）时，数据库服务器将接受客户机所建议的加密算法。

如果 **alternate_auth_enc** 设为 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 **alternate_auth_enc** 设为 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。

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

- 带有本地客户机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
- 带有本地客户机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]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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:

- 它用来确定打开分块游标时使用的 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** 进程之间的通信量。（**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（用于本地客户机和远程客户机）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。

较大记录块还可能产生比应用程序执行块访存所需更多的数据。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

- 带有本地客户机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
- 带有本地客户机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

此参数的缺省值为零 (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，以用于放置审计工具所生成的审计记录。该值乘以 4KB 页就是为审计缓冲区分配的空间容量。**db2auditd** 审计守护进程将按固定的时间间隔或者在审计缓冲区已满时将审计缓冲区清空并将其内容保存到磁盘中。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，该审计工具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，审计工具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”

此参数指定并确定用户的认证如何进行以及在何处进行。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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** 设为 SERVER，那么会将用户标识和密码从客户机发送至服务器，以便可以在服务器上执行认证。值 SERVER_ENCRYPT 提供与 SERVER 相同的行为，但是将对通过网络发送的任何用户标识和密码进行加密。

值 DATA_ENCRYPT 表示服务器接受加密的 SERVER 认证方案和用户数据的加密。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 和 XQuery 语句
- SQL 程序变量数据
- 来自处理 SQL 或 XQuery 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

值 DATA_ENCRYPT_CMP 表示服务器接受加密的 SERVER 认证方案和用户数据的加密。另外，此认证类型允许与不支持 DATA_ENCRYPT 认证类型的较早产品兼容。这些产品允许使用 SERVER_ENCRYPT 认证类型来进行连接，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，在 **CATALOG DATABASE** 命令上使用该认证类型无效。

注: 为了符合标准（在『符合标准』主题中定义），配置 SERVER 是唯一受支持的值。

值 CLIENT 指示所有认证都在客户机上进行。不需要在服务器上进行认证。

值 KERBEROS 意味着在 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。

值 GSSPLUGIN 意味着使用外部的基于 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的认证类型为 GSS_SERVER_ENCRYPT，那么有效系统认证类型是 GSSPLUGIN（即，如果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。如果远程客户机正在连接至数据库服务器，那么建议您使用 SERVER_ENCRYPT 值来保护用户标识和密码。

cf_diaglevel - CF 的诊断错误捕获级别配置参数

此参数指定将记录在 cfdiag*.log 文件中的诊断错误类型

配置类型

数据库管理器

适用于

- DB2 pureScale

参数类型

可脱机配置

传播类 立即

缺省值 [范围]

2 [1 - 4]

此参数的有效值为:

- 0 - 未捕获到任何诊断数据
- 1 - 仅严重错误
- 2 - 所有错误
- 3 - 所有错误和警告
- 4 - 所有错误、警告和参考消息

cf_diagpath 配置参数用于指定将包含将根据 **cf_diaglevel** 参数的值生成的诊断消息文件的目录

CF 日志文件位置将按以下顺序确定:

- **cf_diagpath**（DBM 配置参数）
- **diagpath**（DBM 配置参数）
- **DB2PATH**/db2dump

cf_diagpath -"CF 的诊断数据目录路径"配置参数

此参数允许您指定 CF 的诊断信息文件的标准路径

配置类型

数据库管理器

适用于

- DB2 pureScale

参数类型

可脱机配置

传播类 立即

缺省值 [范围]

INSTHOME/sqllib/db2dump/ $m [any valid path name]

从 V10.1 开始，缺省情况下 cf 诊断数据目录路径写至每个 CF 的专用 db2diag.log。要复原至以前发行版的行为（在这些发行版中，将 CF 的诊断数据写至同一目录），请指定带有 *pathname* 且不带标记的 **cf_diagpath**。

每个 CF 日志文件名称具有以下格式:

```
cfdiag-YYYYMMDDhhmmssuuuuuu.<cf#>.log
```

但是，还有一个静态 CF 诊断日志名称，它始终指向最新的 CF 诊断日志文件并具有以下格式:

```
cfdiag.<cf#>.log
```

例如，如果已列示所有 CF 日志文件，那么您将见到类似如下的内容:

```
$ ls -la cfdiag*
lrwxrwxrwx 1 db2inst1 pdxdb2   35 2011-02-09 15:07 cfdiag.128.log ->
cfdiag-20110209150733000049.128.log
lrwxrwxrwx 1 db2inst1 pdxdb2   35 2011-02-09 15:10 cfdiag.129.log ->
cfdiag-20110209151021000040.129.log
-rw-r-xr-- 1 db2inst1 pdxdb2 1271 2011-02-09 15:07 cfdiag-20110209150733000049.128.log
-rw-r-xr-- 1 db2inst1 pdxdb2 1271 2011-02-09 15:07 cfdiag-20110209150740000082.129.log
-rw-r-xr-- 1 db2inst1 pdxdb2 1274 2011-02-09 15:10 cfdiag-20110209151021000040.129.log
```

在此示例中，`cfdiag.128.log` 和 `cfdiag.129.log` 是本来时间戳记日志文件的最新版本的符号链接。

CF 日志文件类似 `db2diag.log file`。

cf_mem_sz -“CF 内存”配置参数

此参数控制集群高速缓存设施（又称为 CF）使用的总内存。

配置类型

数据库管理器

适用于 仅适用于 DB2 pureScale 实例。

- 带有本地客户机和远程客户机的数据库服务器

参数类型

可脱机配置

缺省值 [范围]

AUTOMATIC [32768 - 4 294 967 295]

计量单位

页 (4 KB)

分配时间

CF 启动时

释放时间

CF 停止时

如果应用缺省设置 AUTOMATIC，那么集群高速缓存设施内存量通过查询 CF 服务器上可用的总内存确定。 然后此参数设为 CF 服务器上的总内存的适当百分比或机器上的可用内存量（取两者中的较小值）。适当百分比通常为 CF 上可用总内存的 70% 到 90%。 AUTOMATIC 计算期间需要考虑的因素还包括:

- CF 和任何 DB2 成员共存时
- 同一主机上是否有多个实例

cf_num_conns -“每个 CF 的每个成员的 CF 连接数”配置参数

此参数控制集群高速缓存设施 (CF) 连接池的初始大小。

配置类型

数据库管理器

参数类型

可联机配置

缺省值 [范围]

AUTOMATIC [4 - 256]

分配时间

当启动 DB2 时

释放时间

当停止 DB2 时

如果将 **cf_num_conns** 参数设为 AUTOMATIC（缺省值），那么对于每个 CF，DB2 数据库管理器会在每个成员启动时为该成员创建初始数目的 CF 连接。此初始数目基于工作程序线程数、（请参阅『cf_num_workers -“工作程序线程数”配置参数』）每个工作程序线程的连接数和集群中的成员数。**cf_num_conns** 参数的实际最大值由 DB2 在成员启动时根据这些参数自动计算。

如果 **cf_num_conns** 参数设为 AUTOMATIC 并且随后升至高于当前值的值，那么对于实例连接，会创建新的 CF 连接。但是，如果将 **cf_num_conns** 参数设为低于当前值的值，那么 CF 连接不会关闭。

如果 **cf_num_conns** 参数设为 AUTOMATIC，那么会在余下端口上平均地重新建立由端口故障（或其他网络问题）导致的任何被删除连接。失效端口恢复联机时，连接会根据需要重新平衡。

如果将 **cf_num_conns** 参数设为固定数字值，那么对于每个 CF，DB2 数据库管理器会在每个成员启动时为该成员创建正好该数目的 CF 连接。DB2 数据库管理器不会进行任何自动增长或缩减。

如果 **cf_num_conns** 参数设为固定值，并且随后该值增加或减少，那么对于实例连接，CF 连接数会根据该新值立即增加或减少。

如果 **cf_num_conns** 参数设为固定值，那么端口故障不会增加余下端口上的连接数。对每个端口建立的连接数仍为将 **cf_num_conns** 参数除以端口数得出的值。即使某些端口脱机，该数也不会更改。

cf_num_workers -“工作程序线程数”配置参数

cf_num_workers 参数指定集群高速缓存设施 (CF) 上的工作程序线程总数。工作程序线程分布在多个通信适配器端口之间以平衡每个接口上用于处理请求的工作程序线程数。

配置类型

数据库管理器

适用于 仅适用于 DB2 pureScale 实例。

参数类型

可脱机配置

缺省值 [范围]

AUTOMATIC [1 - 31]

如果设为缺省值 (AUTOMATIC)，那么参数值配置为 CF 上的可用处理器数减 1。可用 CPU 在实例间平均划分，然后从针对每个实例生成的值减去一个处理器。如果 CF 主机上有共存成员，那么 CF 上的工作程序线程数进一步除以主机上的 CF 和成员的总数。

如果 CF 服务器上只有一个处理器，那么此值设为 1。

CF 工作程序线程总数显示在 cfdiag.log 文件中。

要计算指定给每个集群互连接口的工作程序线程数，请将 CF 工作程序线程总数除以对 CF 定义的通信适配器端口数。

```
指定给每个接口的工作程序数 = (cf_num_workers) / (接口数)
```

如果 **cf_num_workers** 参数设为 AUTOMATIC，那么数据库管理器会配置工作程序线程数以便它可由为 CF 配置的通信适配器端口数平均划分。

例如，在带有 7 个可用处理器和 2 个通信适配器端口的 CF 服务器上， AUTOMATIC 设置的值为:

```
CF 工作程序线程总数 = (7 个处理器) - (1 个处理器，以避免使用所有处理器) = 6
这样，连接至每个集群互连接口的工作程序数 =      6 / 2 = 3
```

如果手动设置 **cf_num_workers** 参数，请将该值设置为等于或大于通信适配器端口数以便每个接口至少有一个工作程序线程。如果工作程序线程数不足以覆盖所有接口，那么系统会针对启动失败的 CF 记录警报。该参数值必须更改才能解决此问题。

限制: 如果您正运行 InfiniBand 或 uDAPL，请不要将此值设为大于 CF 服务器上的处理器数。每个工作程序线程在一个处理器上运行以等待 uDAPL 通信。如果工作程序线程数超过处理器数，那么性能会受影响。

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

- 带有本地客户机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
- 带有本地客户机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]

带有本地客户机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

- 客户机
- 带有本地客户机和远程客户机的数据库服务器
- 仅带有本地客户机的数据库服务器
- 带有本地客户机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]
在 Linux 和 UNIX 操作系统上为 NULL，在 Windows 操作系统上为 IBMkrb5
[任何有效字符串]

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数据库管理器配置中的认证类型设为 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

- 带有本地客户机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
- 带有本地客户机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。使用 CLIENT 认证对客户机进行认证时，或执行本地认证并且 DBM CFG 中的 **authentication** 为 CLIENT、SERVER、SERVER_ENCRYPT、DATA_ENCRYPT 或 DATA_ENCRYPT_CMP 时，会使用该

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 数据库产品。

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

- 带有本地客户机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
- 带有本地客户机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

- 在 DB2 pureScale 环境中：TSA；否则，没有缺省值

有效值

- TSA

当您安装 DB2 pureScale Feature 或您要使用 DB2 高可用性实例配置实用程序 (**db2haicu**) 来配置集群以获取高可用性时，会自动设置此参数。

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 带有本地客户机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, 0.1 - 100000]

值 -1 导致将该参数值重置为缺省值。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算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）。

comm_exit_list -“通信缓冲区出口库列表”

此参数指定 DB2 将使用的通信缓冲区出口库的列表。通信缓冲区出口库是以动态方式装入的库，供应商应用程序可使用此库获取对用于与客户机应用程序通信的 DB2 通信缓冲区的访问并检查这些通信缓冲区。

配置类型
数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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。如果值并非 null，那么应该采用 DB2 要使用的通信缓冲区出口库的逗号分隔列表。每个库必须用逗号分隔，逗号之前或之后不应有空格。

每个名称最长可有 32 个字节。该名称不应包括扩展名（例如，.so 或 .a）。参数的总长度不得超过 128 个字节。

conn_elapse -“连接耗用时间”

此参数指定在两个 DB2 成员间建立网络连接所用的秒数。

配置类型
数据库管理器

适用于 DB2 pureScale服务器（带有多个 DB2 成员）

带有本地客户机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。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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] 值 -1 将导致参数值根据度量程序的运行来重置。

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）。

cur_commit -“当前已落实”配置参数

此参数控制游标稳定性 (CS) 扫描的行为。

配置类型

数据库

参数类型

- 可配置
- 可由 DB2 pureScale 环境中的成员配置

缺省值 [范围]

ON [ON, AVAILABLE, DISABLED]

对于新数据库，缺省值设为 ON。如果缺省值设为 ON，那么查询将返回提交该查询时数据的当前已落实值。

从 V9.5 或之前版本升级数据库期间，**cur_commit** 配置参数将设为 DISABLED，以便保持前发行版的行为。如果要将“当前已落实”行为用于游标稳定性扫描，那么必须在升级后将 **cur_commit** 配置参数设为 ON。

您可以显式地将 **cur_commit** 配置参数设为 AVAILABLE。设置此参数之后，必须显式地请求“当前已落实”行为才能查看当前已落实的结果。

注: 使用游标稳定性隔离级别时，注册表变量 **DB2_EVALUNCOMMITTED**、**DB2_SKIPDELETED** 和 **DB2_SKIPINSERTED** 受“当前已落实”行为影响。如果在 **BIND** 上或在语句准备期间显式地指定了 **USE CURRENTLY COMMITTED** 或 **WAIT FOR OUTCOME**，那么这些注册表变量将加以忽略。

注: 如果使用“当前已落实”时数据库中存在大量的锁定冲突，那么性能注意事项可能适用。行的落实版本将从日志中检索且执行效果更好，并避免当日志记录仍在日志缓冲区时发生日志磁盘活动。因此，为提高检索先前已落实的数据时的性能，您可能要考虑增加 **logbufsz** 参数的值。

date_compat -“日期兼容性”数据库配置参数

此参数指示与 TIMESTAMP(0) 数据类型相关联的 DATE 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 DATE 数据类型的 **DB2_COMPATIBILITY_VECTOR** 注册表变量的设置。值不可更改。

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

- 带有本地客户机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
- 带有本地客户机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（V2 之前的任何版本），此参数尤其有用。

建议：使用下列可能的值设置此记帐字符串：

- 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

- 带有本地客户机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
- 带有本地客户机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处理时间，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

- 带有本地客户机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
- 带有本地客户机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 数据库系统的驱动器 [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 操作系统上）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"

此参数指定将在 db2diag 日志文件中记录的诊断错误的类型。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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** – 从 FP1 开始，在客户端只捕获管理通知消息。在服务器端仍捕获紧急错误、事件消息和管理通知消息。
- **1** – 只捕获严重错误、紧急错误、事件消息和管理通知消息。
- **2** – 捕获所有错误、事件消息和管理通知消息。
- **3** – 捕获所有错误、警告、事件消息和管理通知消息。
- **4** – 捕获所有错误、警告、参考消息、事件消息和管理通知消息。

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。
- DB2 服务器进程 **db2sysc** 可以检测动态更改，例如，当您对实例附件发出 **UPDATE DATABASE MANAGER CONFIGURATION** 命令时。
- 当 DB2 客户机和应用程序进程启动时，它们将使用 **diaglevel** 配置参数设置，且不会检测任何动态更改。
- 为了帮助解决问题，您可以增大此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 诊断信息的标准主路径。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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]

$*INSTHOME*/sqllib/db2dump/ $m [任何有效路径名（有关解析变量选项，请参阅下文）]

符号

pathname

要使用的目录路径，而不是缺省诊断数据目录

$h 解析为 HOST_*hostname*。

注: 从 V10 开始，在 DB2 pureScale 环境 中，$h 是指成员的原始主机。

$n 解析为 NODE*number*

$m 解析为 DIAG*number*。请注意，不管 DIAG*number* 是指数据库分区、CF 还是成员，都会使用该设置。

/trailing-dir

单个目录，或者是跟在 $h 或 $n 后面的目录和子目录

有下列值可用:

- '"$h"'
- '"$h/*trailing-dir*"'
- '"*pathname* $h"'
- '"*pathname* $h/*trailing-dir*"'
- '"$n"'[4]
- '"$n/*trailing-dir*"'
- '"*pathname* $n"'
- '"*pathname* $n/*trailing-dir*"'
- '"$m"'
- '"$m/*trailing-dir*"'
- '"*pathname* $m"'
- '"*pathname* $m/*trailing-dir*"'
- '"hn"'[5]
- '"hn/*trailing-dir*"'
- '"*pathname* hn"'
- '"*pathname* hn/*trailing-dir*"'
- '"hm"'
- '"hm/*trailing-dir*"'
- '"*pathname* hm"'

4. 建议不要使用 $n，将来的发行版可能会将其除去。

5. 建议不要使用 hn，将来的发行版可能会将其除去。

- '"*pathname* hm/*trailing-dir*"'

根据您使用的平台不同，主诊断数据目录可包含转储文件、陷阱文件、错误日志、通知文件、警报日志文件和“首次出现数据收集”(FODC) 程序包。

如果此参数为空，那么诊断信息将写入下列其中一个目录或文件夹中的缺省诊断路径目录字符串:

- 在 Windows 环境中:
  - 用户数据文件（例如，实例目录下的文件）的缺省位置因为 Windows 系列操作系统的版本不同而变化。使用 **DB2SET DB2INSTPROF** 命令来获取实例目录的位置。该文件位于 **DB2INSTPROF** 注册表变量指定的目录的 instance 子目录中。
- 在 Linux 和 UNIX 环境中: 信息将写入 *INSTHOME*/sqllib/db2dump/$m，其中 *INSTHOME* 是实例的主目录。

从 V10.1 开始，缺省情况下，诊断数据目录路径写至每个成员和 CF 的专用 db2diag 日志文件。要复原至前发行版的行为（在这些发行版中，诊断数据写至同一目录），请指定带有 *pathname* 且不带标记（$h、$n 或 $m）的 **diagpath**。

要分割诊断数据目录路径以按物理主机收集诊断信息，请将此参数设为下列其中一个值:

- 分割缺省诊断数据目录路径:

  ```
  db2 update dbm cfg using diagpath '"$h"'
  ```

  此命令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*
- 使用结尾目录来分割缺省诊断数据目录路径:

  ```
  db2 update dbm cfg using diagpath '"$h/trailing-dir"'
  ```

  此命令将在缺省诊断数据目录下使用主机名和结尾目录创建一个子目录，如下所示:

  Default_diagpath/HOST_*hostname*/*trailing-dir*
- 分割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"'
  ```

  此命令将在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*
- 使用结尾目录分割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/trailing-dir"'
  ```

  此命令将在您自己指定的诊断数据目录下使用主机名和结尾目录创建一个子目录，如下所示:

  pathname/HOST_*hostname*/*trailing-dir*

要分割诊断数据目录路径以按物理主机和按物理主机的数据库分区收集诊断信息，请将此参数设为下列其中一个值:

- 分割缺省诊断数据目录路径:

  ```
  db2 update dbm cfg using diagpath '"$h$n"'
  ```

此命令将在缺省诊断数据目录下使用主机名和分区号为主机中的每个逻辑分区创建一个子目录，如下所示:

Default_diagpath/HOST_*hostname*/NODE*number*

- 使用结尾目录来分割缺省诊断数据目录路径:

```
db2 update dbm cfg using diagpath '"$h$n/trailing-dir"'
```

此命令将在缺省诊断数据目录下使用主机名、分区号和结尾目录为主机中的每个逻辑分区创建一个子目录，如下所示:

Default_diagpath/HOST_*hostname*/NODE*number*/*trailing-dir*

- 分割您自己指定的诊断数据目录路径（*pathname* 与 hn 之间有个空格）:

```
db2 update dbm cfg using diagpath '"pathname $h$n"'
```

此命令将在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中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，名为 boson 的 AIX 主机有 3 个数据库分区，每个数据库分区的节点号分别为 0、1 和 2。该目录的列表输出示例类似如下内容:

```
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
```

- 使用结尾目录分割您自己指定的诊断数据目录路径（*pathname* 与 hn 之间有个空格）:

```
db2 update dbm cfg using diagpath '"pathname $h$n/trailing-dir"'
```

此命令将在您自己指定的诊断数据目录下使用主机名、分区号和结尾目录为主机中的每个逻辑分区创建一个子目录，如下所示:

pathname/HOST_*hostname*/NODE*number*/*trailing-dir*

注:

- 在某些 Linux 和 UNIX 系统上，为了避免操作系统 Shell 解释 $ 符号，必须在双引号外部再添加一个单引号，如上面的语法中所示。
- 在 CLP 交互方式下，或者在命令是从输入文件中读取并执行的情况下，不需要添加双引号。
- $h、$n 和 $m 都不区分大小写。

- **diagpath** 的动态行为不会扩展到所有进程
- DB2 服务器进程 **db2sysc** 可以检测动态更改，例如，当您对实例附件发出 **UPDATE DATABASE MANAGER CONFIGURATION** 命令时。
- 当 DB2 客户机和应用程序进程启动时，它们将使用 **diagpath** 配置参数设置，且不会检测任何动态更改。

在 V9.5 及更高版本中，全局级 **DB2INSTPROF** 的缺省值存储在上面显示的新位置。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**

注: 在 DB2 V9.7 FP4 及更高版本的修订包中，可以通过设置备用诊断路径并与 **diagpath** 参数结合起来使诊断日志记录更具有弹性。如果设置了 **alt_diagpath** 并且由 **diagpath** 指定的路径变得不可用，那么诊断日志记录将在所指定的备用诊断数据目录路径中继续，然后在主诊断路径再次变为可用时在其原始位置继续。

diagsize -“轮换诊断日志和管理通知日志”配置参数

此参数帮助控制诊断日志文件和管理通知日志文件的最大大小。

配置类型
数据库管理器

参数类型
不可联机配置

缺省值 0

用于指定轮换日志大小的最小值:
2

用于指定轮换日志大小的最大值:
diagpath 所指定的目录中的可用空间量

计量单位
兆字节

如果此参数的值是 0（缺省值），那么只有一个诊断日志文件（名为 db2diag.log）。并且，只有一个管理通知日志文件（名为 *<instance>*.nfy），它仅用于 Linux 和 UNIX 操作系统。这些文件的大小可以无限地增大。

如果将此参数设为非零值并重新启动 <instance>，那么将使用一系列轮换诊断日志文件和一系列轮换管理通知日志文件。这些文件名为 db2diag.*n*.log 和 *<instance>*.*n*.nfy，其中，*n* 是整数；*<instance>*.*n*.nfy 文件仅适用于 Linux 和 UNIX 操作系统。db2diag.*n*.log 文件和 *<instance>*.*n*.nfy 文件的数目都不能超过 10。第 10 个文件的大小达到限制后，将删除最旧的文件并创建新文件。

例如，在 Linux 和 UNIX 操作系统上，**diagpath** 中的轮换日志文件可能是以下输出效果:

```
db2diag.14.log, db2diag.15.log, ... , db2diag.22.log, db2diag.23.log
<instance>.0.nfy, <instance>.1.nfy...,<instance>.8.nfy,<instance>.9.nfy
```

如果 db2diag.23.log 已满，那么将删除 db2diag.14.log，并且将创建 db2diag.24.log 以便进行 db2diag 日志记录

如果 <instance>.9.nfy 已满，那么将删除 <instance>.0.nfy，并且将创建 <instance>.10.nfy 以便进行管理通知日志记录。

注意，消息始终被记录到索引号最大的轮换日志文件 db2diag.*largest n*.log 和 <*instance*>.*largest n*.nfy

db2diag.*n*.log 和 <*instance*>.*n*.nfy 文件的总大小由 **diagsize** 配置参数的值确定。缺省情况下，除了在 Windows 操作系统上以外，将把 **diagsize** 值的 90% 分配给 db2diag.*n*.log 文件，并将 **diagsize** 值的 10% 分配给 <*instance*>.*n*.nfy 文件。例如，在 Linux 或 UNIX 操作系统上，如果将 **diagsize** 设为 1024，那么全部 db2diag.*n*.log 文件的总大小不能超过 921.6 MB，并且全部 <*instance*>.*n*.nfy 文件的总大小不能超过 102.4 MB。在 Windows 操作系统上，**diagsize** 的整个值都将分配给 db2diag.*n*.log 文件。每个日志文件的大小由分配给每类日志文件的空间总量除以 10 确定。例如，如果全部 db2diag.*n*.log 文件的总大小不能超过 921.6 MB，那么每个 db2diag.*n*.log 文件的大小将是 92.16 MB。

对 **diagsize** 配置参数指定的最大值不能超过 **diagpath** 配置参数所指定目录中的可用空间量。诊断日志文件和管理通知日志文件都存储在此目录中。为了避免因文件轮换（删除最旧的日志文件）而过早丢失信息，请将 **diagsize** 设为大于 50 MB 的值，但不应超过 **diagpath** 所指定目录中可用空间量的 80%。

例如，

- 要将 **diagsize** 设为 1024 MB，这将在 DB2 重新启动时切换到轮换日志记录行为，请使用以下命令：

  ```
  db2 update dbm cfg using diagsize 1024
  ```

- 要将 **diagsize** 设为 0，这将在 DB2 重新启动时切换到缺省日志记录行为，请使用以下命令：

  ```
  db2 update dbm cfg using diagsize 0
  ```

注：从 DB2 V9.7 FP1 开始，如果 **diagsize** 配置参数设为一个非零值，并且设置了 **diagpath** 配置参数以将诊断数据分配到不同的目录中，那么 **diagsize** 配置参数的非零值将指定下列文件组合的总大小：所有旋转管理通知日志文件与所给定的分配诊断数据目录中包含的所有旋转诊断日志文件的组合。例如，如果有 4 个数据库分区的系统将 **diagsize** 设为 1 GB，并且 **diagpath** 设为 "$n"（按数据库分区分配诊断数据），那么组合在一起的通知和诊断日志的最大总大小可以达到 4 GB (4 x 1 GB)。

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

- 带有本地客户机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

- 带有本地客户机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。通常，在实例运行期间，从不刷新此服务器目录高速缓存。但是，进行脱机备份时会将服务器目录高速缓存标记为无效，即使对于正在运行的实例，也会刷新高速缓存。

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

- 带有本地客户机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
- 带有本地客户机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，可以将有关 DB2 系统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

- 带有本地客户机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
- 带有本地客户机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

- 带有本地客户机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
- 带有本地客户机和远程客户机的分区数据库服务器或 DB2 pureScale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 [895 - 65300]

64 位平台

Automatic [895 - 524288]

- 带有本地客户机和远程客户机的数据库服务器: 1024
- 带有本地客户机的数据库服务器: 895
- 带有本地客户机和远程客户机的分区数据库服务器或 DB2 pureScale 数据库服务器: 4096

缺省情况下，快速通信管理器 (FCM) 缓冲区用于成员内和成员间通信。

要点: 最初创建数据库后，DB2 配置顾问程序可更改 **fcm_num_buffers** 参数的缺省值。

可为 **fcm_num_buffers** 配置参数设置初始值和 AUTOMATIC 值。此参数设为 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。资源增加或减少的量取决于操作系统。在 Linux 操作系统上，缓冲区数只能增加至比起始值高 25%。如果数据库管理器尝试启动实例并且无法分配指定数目的缓冲区，那么它会降低此数目直到能够启动该实例。

如果要将 **fcm_num_buffers** 参数设为特定值和 AUTOMATIC 并且您不希望系统控制器线程将资源量调整至指定值之下，请将 **DB2_FCM_SETTINGS** 注册表变量的 FCM_CFG_BASE_AS_FLOOR 选项设为 YES 或 TRUE。**DB2_FCM_SETTINGS** 注册表变量值将以动态方式调整。

如果您正在使用多个逻辑节点，那么同一机器上所有逻辑节点将共享一个缓冲区数为 **fcm_num_buffers** 的池。通过将 **fcm_num_buffers** 参数的值与物理机器上的逻辑节点数相乘来确定池大小。检查您正在使用的值；考虑具有多个逻辑节点的机器上分配的 FCM 缓冲区数。如果同一机器上有多个逻辑节点，那么您可能必须增加 **fcm_num_buffers** 参数的值。系统上的用户太多、系统上的数据库分区服务器太多或应用程序太复杂可能导致系统消息缓冲区不足。

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

- 带有本地客户机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
- 带有本地客户机和远程客户机的分区数据库服务器或 DB2 pureScale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 或 2048 [128 - 120000]

UNIX 64 位平台

Automatic，起始值为 256、512 或 2048 [128 - 524288]

Windows 32 位

Automatic，起始值为 10000 [128 - 120000]

Windows 64 位

Automatic，起始值为 256、512 或 2048 [128 - 524288]

不同类型的服务器的缺省起始值如下所示:

- 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。
- 对于带有本地客户机和远程客户机的分区数据库环境服务器，起始值为 2048。

缺省情况下，快速通信管理器 (FCM) 缓冲区用于成员内和成员间通信。为允许非集群数据库系统使用 FCM 子系统和 **fcm_num_channels** 参数，必须将 **intra_parallel** 参数设为 YES

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）都依靠通道在各个成员之间传送数据。

如果设为 AUTOMATIC，那么 FCM 会监视通道使用情况，并且会随要求的更改而逐渐分配和释放资源。

fcm_parallelism -“节点间通信并行性”

此参数指定用于 DB2 实例内的成员间通信（控制消息和数据流）的并行度。

此参数确定发送方和接收方快速通信管理器同步管道对的数目。缺省情况下，一个实例只有一对: 一个发送方和一个接收方，它们处理与实例中的其他成员的所有通信。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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 [1 - 8]

如果此配置参数设为 1，那么不会使用并行性。只有在您关闭实例中的所有成员然后将其重新启动后，对此参数的更新才会生效。

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

- 带有本地客户机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
- 带有本地客户机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** 设为 yes、**authentication** 设为 server 或 server_encrypt，并且 **federated** 设为 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

- 带有本地客户机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
- 带有本地客户机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 和 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存取方案中的最大异步表队列（ATQ）数。ATQ 的机制在 DB2 pureScale 环境中不起作用。

配置类型
数据库管理器

适用于

- 启用联合时带有本地客户机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），-1, ANY]

如果指定了 ANY 或 -1，那么优化器将确定用于存取方案的 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存取方案中的最大 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

- 带有本地客户机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

- 带有本地客户机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行存储过程和 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，那么以下过程在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

- 带有本地客户机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
- 带有本地客户机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。

要点: 在 V10.1 中不推荐使用此参数，在将来的发行版中可能会将其除去。此参数仍可在 V10.1 之前的发行版中使用。

不能在 DB2 pureScale 环境中使用此参数。

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]
Off [On; Off]

相关参数

如果打开了 **health_mon**，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。

可使用 CLP 来选择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，以及在前滚期间或在备用数据库上重放高可用性灾难恢复 (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

- 带有本地客户机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

- 带有本地客户机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

使用数据库管理器配置文件中指定的系统设置来决定何时重建无效的索引，以及在前滚或 HADR 日志重放期间是否将重做任何索引构建日志记录。

注: 此设置仅对数据库配置有效。

ACCESS

第一次访问基础表时将重建无效的索引。在前滚或 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。对新主存储器上基础表的访问会导致索引重建，这会导致写入日志记录，然后将其发送到新的备用存储器，进而导致索引在备用存储器上无效。

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: 在 DB2 pureScale 环境中，仅在组崩溃恢复期间（而不会在成员崩溃恢复中）重建索引。

注: 如果应用程序连接至数据库时该数据库异常终止，并且已启用 autorestart 参数，那么应用程序连接至数据库时会隐式发出 **RESTART DATABASE** 命令。如果未发出该命令，那么下次访问底层表时会重建无效索引。

RESTART_NO_REDO

将在显式或隐式发出 **RESTART DATABASE** 命令时重建无效的索引。在前滚或 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。接管会导致在新主存储器的基础表上重建索引，这会导致写入日志记录，然后将其发送到新的备用存储器，进而导致索引在备用存储器上无效。

如果应用程序连接至数据库时该数据库异常终止，并且已启用 autorestart 参数，那么应用程序连接至数据库时会隐式发出 **RESTART DATABASE** 命令。如果未发出该命令，那么下次访问底层表时会重建无效索引。

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，如果索引发生此问题，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下降。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。

如果将此参数设为 RESTART，那么重新启动数据库所耗用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（例如，**CREATE INDEX** 和 **REORG INDEX** 操作）或对索引重建操作激活完全日志记录时，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更改表时启用 LOG INDEX BUILD 来激活日志记录。通过将 **indexrec** 设为 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

- 带有本地客户机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
- 带有本地客户机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 - 1 048 576]

DB2 Workgroup Server Edition

Automatic [0 - 4 194 304]

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 V9.7 FP1 和 V9.5 FP5 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施加限制。对于 V9.7 和 V9.5 FP4 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。

对分区数据库环境中的实例的限制

您不应对分区数据库环境中的 **instance_memory** 使用特定值。在分区数据库环

境中，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_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 实例耗用的实例内存总量。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查询并行性。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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 导致该参数值设为 YES 或 NO，这取决于正在运行数据库管理器的硬件。

注：可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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。
- 可通过调用 ADMIN_SET_INTRA_PARALLEL 过程来覆盖应用程序中的 **intra_parallel** 设置。可通过在工作负载定义中设置 MAXIMUM DEGREE 属性在工作负载中覆盖 **intra_parallel** 设置和 ADMIN_SET_INTRA_PARALLEL 过程在应用程序中设置的值。

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

- 带有本地客户机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
- 带有本地客户机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

- 带有本地客户机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
- 带有本地客户机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，如果重置数据库管理器（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

- 带有本地客户机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
- 带有本地客户机和远程客户机的分区数据库服务器

s

参数类型

可配置

缺省值 [范围]

Yes [Yes; No]

如果将 **keepfenced** 设为 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，那么可将此参数设为 yes。这将通过避免初始设防方式进程创建处理时间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。如果将此参数设为 no，每个事务都会产生创建新的设防方式进程的额外处理时间，这将使性能显著降低。但是，如果此参数设为 yes，那么每个事务都将尝试使用现有设防方式进程，这将避免此额外处理时间。

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

- 带有本地客户机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
- 带有本地客户机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”

此参数指示每个成员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

- 带有本地客户机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
- 带有本地客户机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 601 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成员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。将此参数设为大于缺省值的值时，会激活集中器功能部件。

用法

如果用户保留下列其中一项权限，那么客户机连接的最大数量可超过由 **max_connections** 配置参数设置的值:

- SYSADM
- SYSCTRL
- SYSMAIN

注: 由于很多原因，跳过由 **max_connections** 配置参数设置的客户机连接的最大数量非常有用，包括对下列任务也很有用:

- 管理任务
- 捕获快照信息
- 关键故障诊断任务
- 维护任务（例如，强制用户脱离该系统）

max_connretries -“节点连接重试次数”

此参数指定尝试在两个 DB2 成员之间建立网络连接的最大次数。

配置类型

数据库管理器

适用于 带有本地客户机和远程客户机的分区数据库服务器

DB2 pureScale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 DB2 成员之间建立通信失败（例如，达到 **conn_elapse** 参数指定的值），那么 **max_connretries** 指定可对 DB2 成员执行连接重试的次数。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

- 带有本地客户机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
- 带有本地客户机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 601 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。

当集中器处于打开状态时（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。

用法

如果用户保留下列其中一项权限，那么协调代理程序的最大数量可超过由 **max_coordagents** 配置参数设置的值:

- SYSADM
- SYSCTRL
- SYSMAIN

注: 由于很多原因，跳过由 **max_coordagents** 配置参数设置的协调代理程序的最大数量非常有用，包括对下列任务也很有用:

- 管理任务
- 捕获快照信息
- 关键故障诊断任务
- 维护任务（例如，强制用户脱离该系统）

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

- 带有本地客户机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
- 带有本地客户机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 - 32 767]（ANY 表示由系统确定）

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

必须将 **intra_parallel** 配置参数设为 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: 可使用 CURRENT DEGREE 专用寄存器或 **DEGREE** 绑定选项在编译语句时指定 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 -“成员间的最大时差”

此参数指定节点配置文件中列示的 DB2 pureScale 环境中成员间允许的最大时差。

配置类型
数据库管理器

适用于 带有本地客户机和远程客户机的成员

参数类型
可配置

缺省值 [范围]

在 DB2 pureScale 环境中
1 [1 - 1 440]

在 DB2 pureScale 环境外部
60 [1 - 1 440]

计量单位
分钟

每个成员都有自己的系统时钟。系统会定期检查两个或更多成员系统时钟之间的时差。如果系统时钟之间的时差大于 **max_time_diff** 参数指定的量，那么 db2diag 日志文件中将记录警告。

在 DB2 pureScale 环境中，为确保成员相互同步，需要网络时间协议 (NTP) 设置并且，系统会定期针对每个成员验证此设置。如果未检测到 NTP 守护程序，那么 db2diag 日志文件中会记录警告。

分区数据库环境中返回了 SQL1473N 错误消息，其中会将系统时钟与 SQLOGCTL.LFH 日志控制文件中保存的虚拟时间戳记 (VTS) 进行比较。如果 .LFH 日志控制文件中的时间戳记小于系统时间，那么数据库日志中的时间会设为 VTS，直到系统时钟与 VTS 匹配。

DB2 数据库管理器使用全球标准时间 (UTC)，所以设置 **max_time_diff** 参数时不用考虑时区差异。UTC 与格林威治标准时间相同。

maxagents -“最大代理程序数”

从 V9.5 起建议不要使用此参数，但是 V9.5 之前的数据服务器和客户机仍然可以使用此参数。DB2 V9.5 或更高发行版中的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 V9.5 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

- 带有本地客户机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
- 带有本地客户机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]

在带有本地客户机和远程客户机的分区数据库服务器上: 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额外的处理资源。

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 -“最大并行代理程序数”

从 V9.5 起建议不要使用此参数，但是 V9.5 之前的数据服务器和客户机仍然可以使用此参数。DB2 V9.5 或更高发行版中的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 V9.5 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

- 带有本地客户机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
- 带有本地客户机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）。执行启用监视开关、重置监视数据、激活事件监视器或将监视事件发送至处于活动状态的事件监视器之类的数据库监视活动时，系统会从监视器堆分配内存。

对于 V9.5，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

- 带有本地客户机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
- 带有本地客户机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，将会发生以下事件中的一件:

- 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。
- 如果应用程序需要发送事件监视器记录但不能从监视器堆分配记录，那么该应用程序可能会受阻直到分配记录。

nodetype -“实例节点类型”

此参数指定机器上安装的 DB2 产品创建的实例类型（数据库管理器类型）。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机和远程客户机的分区数据库服务器

参数类型

参考

以下列表中显示了此参数返回的可能值以及与该节点类型相关的产品:

- **带有本地客户机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。
- **带有本地客户机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

- 带有本地客户机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
- 带有本地客户机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 操作系统上，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。（建议不要使用此设置。）
- 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。将在此级别显示参考消息。
- 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。

使用说明

- 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。
- 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。
- 在特定环境下，为显示很重要的消息，DB2 将覆盖 **notifylevel** 设置

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

- 带有本地客户机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
- 带有本地客户机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]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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 **START DBM** 时在 **db2fmp** 池中创建的非线程化空闲 **db2fmp** 进程的初始数目。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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。

将 **fenced_pool** 设为适合于系统的大小要比在 **START DBM** 时启动许多 **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

- 带有本地客户机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
- 带有本地客户机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 或 AUTOMATIC [-1, 0 - 64000]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

此配置参数设为 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 100 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。

此参数超负荷。

numdb 数据库管理器配置参数的效果取决于使用该参数的环境。

- 对于 DB2 Connect 环境，可使用 **numdb** 数据库管理器配置参数指定可针对 DB2 Connect 服务器编目的数据库别名的最大数目。
- 在 DB2 Connect 环境外部，可使用 **numdb** 数据库管理器配置参数指定可同时处于活动状态的本地数据库数目。

配置类型

数据库管理器

适用于

- 带有本地客户机和远程客户机的数据库服务器
- 仅带有本地客户机的数据库服务器
- 带有本地客户机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 DB2 pureScale 环境

32 [1 - 200]

带有本地客户机和远程客户机的 UNIX 数据库服务器

32 [1 - 256]

仅带有本地客户机的 UNIX 数据库服务器

8 [1 - 256]

带有本地客户机和远程客户机的 Windows 数据库服务器

32 [1 - 256]

仅带有本地客户机的 Windows 数据库服务器

3 [1 - 256]

计量单位

计数器

使用说明

- 设置或更新此参数时应记住，每个数据库都会占用存储空间，并且每个处于活动状态的数据库都需要额外的新共享内存段。
- 如果正在实例上运行的数据库数目小于缺省数目，请将 **numdb** 设为缺省值。
- **numdb** 配置参数必须与 **db2_database_cf_memory** 和 **cf_db_mem_sz** 配置参数协调使用。
- 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。规划与数据库或连接至该数据库的应用程序的内存分配相关的所有参数的配置，并同时更改全部参数。
- 除了 DB2 pureScale 环境中的 **numdb** 限制外，还对实例中所有成员中的数据库激活最大数有所限制。此数据库激活数最大为 512。例如，如果由 4 个成员组成的 DB2 pureScale 环境中的每个成员激活了 200 个数据库，那么数据库成员激活总数为 800。因为此数据库激活数 (800) 超过最大上限，所以系统会返回错误。
- 在多个数据库环境中，如果需要成员崩溃恢复 (MCR)，那么在每个成员上并行恢复的数据库数由 **numdb** 配置参数或 **DB2_MCR_RECOVERY_PARALLELISM_CAP** 注册表变量的值（两者中的较小者）设置。

query_heap_sz -“查询堆大小”

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。

要点: 在 V9.5 中建议不要使用此参数，在将来的发行版中可能会将其除去。在 V9.5 之前的数据服务器和客户机中仍然可以使用此参数。在 V9.5 和更高发行版中，将忽略对此配置参数指定的值。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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

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

- 带有本地客户机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
- 带有本地客户机和远程客户机的分区数据库服务器

参数类型

参考

rstrt_light_mem -“轻量级重新启动内存”配置参数

此参数指定在主机上分配并保留以用于轻量级重新启动恢复的最大内存量。此内存量为 **instance_memory** 配置参数的百分比。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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]

AUTOMATIC [1 - 10: 对于 AUTOMATIC; 用户定义的范围为 1 - 50]

计量单位

实例内存百分比

分配时间

该实例启动时

释放时间

该实例停止时

将 **rstrt_light_mem** 设为 AUTOMATIC 意味着该实例启动时，DB2 数据库管理器自动计算要预先分配并保留以用于轻量级重新启动恢复的内存量的固定上限。指定的内存量被保留以容纳失败成员，它们需要在其原始主机以外的主机上重新启动（轻量级重新启动方式）。DB2 数据库管理器根据 **instance_memory** 和 **numdb**（当前处于活动状态的数据库的最大数目）的当前设置及主机上的成员数来计算适当值。自动计算的值范围在实例内存限制的 1% 到 10%，并且包括在实例内存的总量中。用户还可将 **rstrt_light_mem** 显式设为范围为实例内存限制的 1% 到 50% 的值。附加内存可缩短恢复时间，特别是在有多个数据库需要恢复时。附加内存还会减少可供正常工作使用的内存，从而减少成员可处理的吞吐量。

该参数是可配置参数但未联机。一旦启动 DB2 实例时分配了保留恢复内存，那么内存量就是固定的并且不会更改，除非配置值更改并且 DB2 实例全局停止并再次启动。

要显示有关在主机上分配的总内存量的信息，请将 **db2pd** 与新 **-totalmem** 选项配合使用。此信息还包括主机上分配的轻量级重新启动内存量。仅返回正访问的当前主机上的信息，但 **db2pd** 可在不同主机上并行运行。

resync_interval -“事务再同步时间间隔”

此参数指定事务管理器 (RM)、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

- 带有本地客户机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
- 带有本地客户机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”

在数据服务器运行时客户机上打开分块游标时，此参数指定块大小。

配置类型

数据库管理器

适用于

- 带有远程客户机的数据库服务器
- 客户机
- 带有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

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。

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

- 带有本地客户机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
- 带有本地客户机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]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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、动态位图（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行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

- 带有本地客户机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
- 带有本地客户机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** 命令来确定没有不确定事务。
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

- 带有本地客户机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
- 带有本地客户机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 [任何有效路径或设备]

如果此参数为空，那么在缺省情况下会将这些日志写入 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。

注: 如果启用了 SPM，并且尚不存在缺省目录，那么将创建缺省目录。要启用 SPM，必须设置配置参数 **spm_name**。

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

- 带有本地客户机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
- 带有本地客户机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

- 带有本地客户机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
- 带有本地客户机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 派生自 TCP/IP 主机名

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

- 带有本地客户机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
- 带有本地客户机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]

NOT_SPECIFIED [NOT_SPECIFIED; CLIENT; SERVER; SERVER_ENCRYPT; DATA_ENCRYPT; DATA_ENCRYPT_CMP; 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 610 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。如果 **srvcon_auth** 参数指定为 KERBEROS、KRB_SERVER_ENCRYPT、GSSPLUGIN 或 GSS_SERVER_ENCRYPT 或未指定 **srvcon_auth** 且 **authentication** 指定为 KERBEROS、KRB_SERVER_ENCRYPT、GSSPLUGIN 或 GSS_SERVER_ENCRYPT，那么它会处理服务器上的入局连接。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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。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。

如果 **srvcon_auth** 参数指定为 CLIENT、SERVER、SERVER_ENCRYPT、DATA_ENCRYPT 或 DATA_ENCRYPT_CMP，或者未指定 **srvcon_auth** 并且 **authentication** 指定为 CLIENT、SERVER、SERVER_ENCRYPT、DATA_ENCRYPT 或 DATA_ENCRYPT_CMP，那么它处理服务器上的入局连接。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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，该值为空，并且会使用随 DB2 数据库提供的用户标识密码插件库。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 数据库产品。

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

- 带有本地客户机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
- 带有本地客户机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

ssl_cipherspecs -“服务器上支持的密码规范”配置参数

此配置参数指定使用 SSL 协议时，服务器允许传入连接请求使用的密码套件。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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 [TLS_RSA_WITH_AES_256_CBC_SHA; TLS_RSA_WITH_AES_128_CBC_SHA; TLS_RSA_WITH_3DES_EDE_CBC_SHA]

可指定多个密码规范，例如，TLS_RSA_WITH_AES_256_CBC_SHA 或 TLS_RSA_WITH_AES_128_CBC_SHA 或 TLS_RSA_WITH_3DES_EDE_CBC_SHA。必须用逗号 (,) 将它们隔开，并且逗号前后不能有空格。

如果指定了 NULL 或多个值，那么在 SSL 握手期间，客户机与服务器将进行协商并确定要使用的最安全密码套件。如果找不到兼容的密码套件，那么连接将失败。您无法通过调整指定密码套件时采用的顺序来指定它们的优先级。

ssl_clnt_keydb -“客户机上用于出站 SSL 连接的 SSL 密钥文件路径”配置参数

此配置参数指定客户机端用于 SSL 连接的密钥文件。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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 [任何有效路径；GSK_MS_CERTIFICATE_STORE]

此参数指定密钥文件的标准文件路径；或者（仅在 Windows 上）指定关键字 GSK_MS_CERTIFICATE_STORE 以使用 Microsoft Windows 证书库。

缺省情况下，SSL 密钥文件的扩展名为 .kbd，它用于存储服务器个人证书中的签署者证书。对于自签署服务器个人证书而言，签署者证书是公用密钥。对于认证中心签署的服务器个人证书而言，签署者证书是根 CA 证书。在 SSL 握手期间，客户机将访问密钥文件以验证服务器个人证书。

缺省情况下，值为 NULL。根据应用程序类型的不同，您应该通过数据库管理器配置参数 **ssl_clnt_keydb**、连接字符串 **ssl_clnt_keydb** 或者 db2cli.ini 和 db2dsdriver.cfg 关键字 **SSLClientKeystoredb** 对 SSL 连接请求指定客户机 SSL 密钥文件路径 。如果未指定任何这些内容，那么 SSL 连接将失败。

ssl_clnt_stash -“客户机上用于出站 SSL 连接的 SSL 隐藏文件路径”配置参数

此配置参数指定客户机端用于 SSL 连接的隐藏文件的标准文件路径。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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 [任何有效路径]

缺省情况下，SSL 隐藏文件的扩展名为 .sth，它用于存储密钥数据库密码的加密版本。隐藏文件中存放的密码用于在 SSL 连接请求期间访问 SSL 密钥文件。

在 Windows 平台上，如果 **ssl_clnt_keydb** 设为关键字 GSK_MS_CERTIFICATE_STORE，那么不需要 **ssl_clnt_stash**。

缺省情况下，值为 NULL。根据应用程序类型的不同，您可以通过数据库管理器配置参数 **ssl_clnt_stash**、连接字符串 **ssl_clnt_stash** 或者 db2cli.ini 关键字 **ssl_clnt_stash** 对 SSL 连接请求指定客户机 SSL 隐藏文件路径。如果未指定任何这些内容，那么 SSL 连接将失败。

ssl_svr_keydb -“服务器上用于传入 SSL 连接的 SSL 密钥文件路径”配置参数

此配置参数指定服务器端用于执行 SSL 设置的密钥文件。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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 [任何有效路径；GSK_MS_CERTIFICATE_STORE]

此参数指定密钥文件的标准文件路径或（仅在 Windows 上）关键字 GSK_MS_CERTIFICATE_STORE，该关键字指示使用 Microsoft Windows 证书库。

缺省情况下，SSL 密钥文件的扩展名为 .kdb，它用于存储个人证书、个人证书请求和签署者证书。在实例启动期间将访问此密钥文件，并且，在 SSL 握手期间，会将服务器个人证书发送到客户机以便执行服务器认证。

缺省情况下，值为 NULL。在实例启动期间，您必须定义 **DB2COMM** 注册表变量是否包含 SSL。否则，实例将在不支持 SSL 协议的情况下启动。

ssl_svr_label -“服务器上用于传入 SSL 连接的密钥文件中的标签”配置参数

此配置参数指定密钥数据库中服务器的个人证书的标签。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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

缺省情况下，值为 NULL。建立 SSL 连接时，会将此配置参数指定的服务器证书发送到客户机以便执行服务器认证。如果此值为 NULL，那么将使用密钥文件中定义的缺省证书。如果不存在缺省证书，那么连接将失败。

ssl_svr_stash -“服务器上用于传入 SSL 连接的 SSL 隐藏文件路径”配置参数

此配置参数指定服务器端用于设置 SSL 的隐藏文件的标准文件路径。

配置类型

数据库管理器

适用于

- 带有本地客户机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

- 带有本地客户机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 [任何有效路径]

缺省情况下，SSL 隐藏文件的扩展名为 .sth，它用于存储密钥数据库密码的加密版本。隐藏文件中存放的密码用于在实例启动期间访问 SSL 密钥文件。

缺省情况下，值为 NULL。在实例启动期间，您必须定义 **DB2COMM** 注册表变量是否包含 SSL。否则，实例将在不支持 SSL 协议的情况下启动。

在 Windows 平台上，如果 **ssl_svr_keydb** 设为关键字 GSK_MS_CERTIFICATE_STORE，那么不需要 **ssl_svr_stash**。

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都必须响应 **START DBM** 或 **STOP DBM** 命令。在 **ADD DBPARTITIONNUM** 和 **DROP 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 带有本地客户机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

在指定时间内未响应 **db2start** 或 **db2stop** 命令的成员或节点将由多成员/节点实例中的 **db2start** 或 **db2stop** 自动终止并清除。诊断消息会记录至数据库管理器配置中定义的 **diagpath** 或其缺省位置（例如，UNIX 操作系统上的 sqllib/db2dump/ $m）。

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被自动终止并清除。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。

使用 **db2start**、**START DATABASE MANAGER** 或 **ADD DBPARTITIONNUM** 命令的其中一个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器路径定义。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。

ssl_svcename -“SSL 服务名称”配置参数

此配置参数指定数据库服务器通过 SSL 协议等待来自远程客户机节点的通信时使用的端口名。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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

此配置参数指定数据库服务器通过 SSL 协议等待来自远程客户机节点的通信时使用的端口。此服务名称必须保留给数据库管理器使用。在实例启动期间，您必须定义 **DB2COMM** 注册表变量是否包含 SSL。否则，实例将在不支持 SSL 协议的情况下启动。

如果 **DB2COMM** 同时包含 TCPIP 和 SSL，那么 **ssl_svcename** 不能与 **svcename** 指定相同的端口。否则，实例将在不支持 SSL 或 TCP/IP 协议的情况下启动。

在 UNIX 操作系统上，services 文件在 /etc/services 中。

您必须在数据库客户机上的 services 文件中定义数据库服务器 SSL 端口（端口号为 *n*）及其服务名称。

ssl_versions -“服务器上支持的 SSL 版本”配置参数

此配置参数指定服务器对于传入连接请求所支持的安全套接字层（SSL）和传输层安全性（TLS）版本。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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 [TLSv1]

如果将此参数设为 null 或 TLSv1，那么此参数将启用对 TLS V1.0 (RFC2246) 和 TLS V1.1 (RFC4346) 的支持。

在 SSL 握手期间，客户机与服务器将进行协商并确定最安全的版本以使用 TLS V1.0 或 TLS V1.1。如果客户机与服务器之间没有兼容的版本，那么连接将失败。如果客户机支持 TLS V1.0 和 TLS V1.1，但服务器只支持 TLS V1.0，那么将使用 TLS V1.0。

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

- 带有本地客户机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
- 带有本地客户机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** 参数设为与主连接端口相关的端口号或服务名称，以便当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

- 带有本地客户机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
- 带有本地客户机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`

SYSADM 权限级别是实例级的最高级别管理权限。具有 SYSADM 权限的用户能够在实例中运行某些实用程序以及发出某些数据库和数据库管理器命令。

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 操作系统而言，可以将此参数设为本地组或域组。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 **sysadm_group** 数据库管理器配置参数指定“NULL”，那么下列用户具有 SYSADM 权限:
  - 本地 Administrators 组的成员
  - 域控制器上 Administrators 组的成员（如果未设置 **DB2_GRP_LOOKUP** 或者将其设为 DOMAIN）
  - DB2ADMNS 组的成员（如果已启用“扩展安全性”功能）。DB2ADMNS 组的位置在安装期间确定
  - LocalSystem 帐户
- 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。

要将此参数复原为缺省值 (NULL)，请使用 **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

- 带有本地客户机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
- 带有本地客户机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

- 带有本地客户机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
- 带有本地客户机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。

要将此参数复原为缺省值（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

- 带有本地客户机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
- 带有本地客户机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。有关 SYSMON 权限支持的完整命令列表，请参阅系统监视器权限 (SYSMON)。

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《*数据库管理概念和配置参考*》的『SQL 和 XML 限制』中指定的长度限制，就接受这些组名。

要将此参数复原为缺省值（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

- 带有本地客户机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
- 带有本地客户机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 for OS/390 V5 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。

可以将此参数设为 **1ST_CONN**，这样将把 TM 数据库设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

- 带有本地客户机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
- 带有本地客户机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，那么此参数应设为“CB”
- 如果应用程序在 IBM MQSeries® 环境中运行，那么此参数应设为“MQ”
- 如果应用程序在 BEA Tuxedo 环境中运行，那么此参数应设为“TUXEDO”
- 如果应用程序在 IBM San Francisco 环境中运行，那么此参数应设为“SF”。

IBM WebSphere EJB 和 **Microsoft Transaction Server** 用户不需要对此参数配置任何值。

如果未在使用上述产品中的任何一个，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。将此参数配置为“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 V9.1 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

- 带有本地客户机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
- 带有本地客户机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 **authentication** 参数设为 CLIENT 时，此参数才处于活动状态。

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** 参数设为 CLIENT 时，才能将此参数更改为 NO。如果此参数设为 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。

将此参数设为 DRDAONLY 可针对所有客户机进行保护，DB2 for OS/390 and z/OS、DB2 for VM and VSE 和 DB2 for OS/400® 中的客户机除外。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。

如果 **trust_allclnts** 设为 DRDAONLY，那么 **trust_clntauth** 参数用于确定客户机的认证位置。如果 **trust_clntauth** 设为 CLIENT，那么认证会在客户机上进行。如果 **trust_clntauth** 设为 SERVER，那么认证会在客户机（如果未提供任何密码）和服务器（如果提供了密码）上进行。

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

- 带有本地客户机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
- 带有本地客户机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。如果此参数设为 SERVER，那么将在服务器上验证用户标识和密码。

如果使用 db2CfgSet API 来设置该数据库管理器配置参数，那么 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下降。

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
- 带有本地客户机和远程客户机的数据库服务器
- 带有本地客户机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

如果性能下降受限制，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耗用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** 设为 100 来整体禁用调速。

wlm_dispatcher -“工作负载管理分派器”

此参数用于启用 (YES) 或禁用 (NO) DB2 工作负载管理分派器。缺省情况下，已启用的分派器允许设置 CPU 限制。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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 [NO; YES]

升级 DB2 数据库管理器时，**wlm_dispatcher** 数据库管理器配置参数的值设为 NO。

工作负载管理分派器使用基于份额的 CPU 资源和/或 CPU 限制分配在 DB2 数据库管理器中提供服务类级别的 CPU 调度功能。

启用工作负载管理分派器后，在用户服务类和维护服务类中运行的所有工作都由此分派器控制。启用该分派器后，该分派器强制将 CPU 限额设置作为缺省设置。为使用基于份额的 CPU 资源分配，必须启用 **wlm_disp_cpu_shares** 数据库管理器配置参数。

如果 **wlm_dispatcher** 配置参数设为 YES，那么存在以下情况:

- 如果任何服务类的代理程序优先级设为缺省值以外的任何值，那么系统会在数据库激活时向 db2diag 日志和管理通知日志写入警告消息。
- 如果尝试创建或改变服务类以将代理程序优先级设为缺省值以外的值，那么系统会向发出用于创建或改变服务类的语句的应用程序返回警告。

wlm_disp_concur -“工作负载管理器分派器线程并行度”

此参数指定 DB2 工作负载管理器 (WLM) 分派器如何设置线程并行度级别。还可将此线程并行度级别手动设为固定值。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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]

COMPUTED [COMPUTED; *manually_set_value*]

升级 DB2 数据库管理器时，**wlm_disp_concur** 数据库管理器配置参数的值为 COMPUTED。

COMPUTED

DB2 数据库管理器根据可供 DB2 数据库管理器使用的逻辑 CPU 数的 4 倍值计算固定线程并行度级别。

manually_set_value

可将线程并行度级别手动设为固定值 (1 - 32767)。最佳值取决于所使用特定硬件和操作系统级别；通常，在主机或 LPAR 上逻辑 CPU 数的 2 倍到 4 倍之间。

计量单位

并行线程数

此数据库管理器配置参数的设置控制 WLM 分派器允许分派至并行操作系统运行队列的线程数。此值设为可供 DB2 数据库管理器使用的逻辑 CPU 数的较低倍数。通常，可将此值设为可用逻辑 CPU 数的 4 倍以考虑可能的调度等待时间（因为线程进入/退出活动状态而导致）。最佳值的大小应该刚好足够确保 DB2 数据库管理器有足够的线程数来完全使用主机或 LPAR 上的 CPU。此最佳值确保最高效率并允许 DB2 WLM 分派器最大程度地控制 CPU 分配。

wlm_disp_cpu_shares -“工作负载管理器分派器 CPU 份额”

此参数会启用 (YES) 或禁用 (NO) 对 DB2 工作负载管理器 (WLM) 分派器的 CPU 份额的控制。缺省情况下，已启用的 WLM 分派器仅控制 CPU 限额。

配置类型

数据库管理器

适用于

- 带有本地客户机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
- 带有本地客户机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 [NO; YES]

升级 DB2 数据库管理器时，**wlm_disp_cpu_shares** 据库管理器配置参数的值为 COMPUTED NO。

如果 **wlm_dispatcher** 数据库管理器配置参数的值设为 YES 并且 **wlm_disp_cpu_shares** 数据库管理器配置参数的值设为 NO，那么 WLM 分派器仅对服务类的管理应用 CPU 限额。

如果 **wlm_dispatcher** 数据库管理器配置参数的值设为 YES 并且 **wlm_disp_cpu_shares** 数据库管理器配置参数的值设为 YES，那么 WLM 分派器对服务类的管理应用 CPU 限额和 CPU 份额。缺省情况下，所有服务类被分配 1000 个硬 CPU 份额以确保相等的 CPU 资源划分。

表 135. DB2 WLM 分派器管理服务类时必需的数据库管理器配置参数设置的摘要

服务类管理	wlm_dispatcher 设置	wlm_disp_cpu_shares 设置
无	NO	NO
CPU 限额	YES	NO
CPU 限额 + CPU 份额	YES	YES

wlm_disp_min_util -“工作负载管理器分派器最低 CPU 利用率”

此参数指定要包括在 DB2 WLM 管理的 CPU 资源份额中的服务类所需的最低 CPU 利用率。

配置类型

数据库管理器

适用于

- 带有本地客户机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成员数据库

参数类型

可联机配置

传播类 立即

缺省值 [范围]

5 [0 至 100]

升级 DB2 数据库管理器时，**wlm_disp_min_util** 数据库管理器配置参数的值为 5。

计量单位

百分比

为使用示例说明此数据库管理器配置参数的用法，假定有 3 个服务类 A、B 和 C，每个服务类有 100 份 CPU 资源。在此示例中，不管这些服务类份额是硬 CPU 份额还是软 CPU 份额，都会获得相同结果。服务类 A 和 B 的 CPU 利用率都大于或等于为 **wlm_disp_min_util** 配置参数设置的值 8%。服务类 C 的 CPU 利用率为 3%，它小于为 **wlm_disp_min_util** 配置参数设置的值 8%。在 CPU 份额计算中，服务类 C 被认为没有任何执行工作。因此，只有服务类 A 和 B 使用的 CPU 资源份额相等，每个服务类分到 50% 的份额。如果服务类 C 开始执行的工作达到的 CPU 利用率大于或等于为 **wlm_disp_min_util** 配置参数设置的值 8%，那么此时服务类 A、B 和 C 现在被视为其 CPU 资源份额相等，每个服务类分到 33.3% 的份额。

在多成员数据库环境中，系统会将主机或 LPAR 上所有成员的 CPU 利用率的聚集与 **wlm_disp_min_util** 配置参数相比以确定该主机或 LPAR 是否包括在 WLM 管理的 CPU 资源份额中。

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

- 带有本地客户机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
- 带有本地客户机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，就不能更改或将它重置。

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”

从 V9.5 起建议不要使用此参数，但是 V9.5 之前的数据服务器和客户机仍然可以使用此参数。DB2 V9.5 或更高发行版中的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 V9.5 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]

带有本地客户机和远程客户机的数据库服务器

- 128 [1 - 64 000]（未启用 **intra_parallel** 时）
- 512 [1 - 64 000]（已启用 **intra_parallel** 时）

带有本地客户机的数据库服务器

- 64 [1 - 64 000]（适用于非 UNIX 平台: 未启用 **intra_parallel** 时）
- 512 [1 - 64 000]（适用于非 UNIX 平台: 已启用 **intra_parallel** 时）
- 128 [1 - 64 000]（适用于 Linux 和 UNIX 平台: 未启用 **intra_parallel** 时）
- 512 [1 - 64 000]（适用于 Linux 和 UNIX 平台: 已启用 **intra_parallel** 时）

带有本地客户机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。所需的内存量随并行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”

从 V9.5 起建议不要使用此参数，但是 V9.5 之前的数据服务器和客户机仍然可以使用此参数。DB2 V9.5 或更高发行版中的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 V9.5 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
- 带有本地客户机和远程客户机的数据库服务器
- 带有本地客户机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

- 带有本地客户机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
- 带有本地客户机和远程客户机的分区数据库服务器

参数类型

- 可联机配置
- 可由 DB2 pureScale 环境中的成员配置

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更改。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”

applheapsz 配置参数指的是整个应用程序可以消耗的应用程序内存总量。

在 DB2 V9.5 之前的版本中，**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 V9.5，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。对于分区数据库环境、集中器或 SMP 配置，这意味着除非使用 AUTOMATIC 设置，否则在相似工作负载下，先前发行版中使用的 **applheapsz** 值可能需要增加。

配置类型

数据库

参数类型

- 可联机配置
- 可由 DB2 pureScale 环境中的成员配置

缺省值 [范围]

Automatic [16 - 60 000]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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

- 带有本地客户机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
- 带有本地客户机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

此参数指定在修改数据库日志文件、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 修改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改恢复历史记录文件。如果将 **auto_del_rec_obj** 数据库配置参数设为 ON，那么数据库管理器在修改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。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，数据库管理器仅能删除数据库日志、备份映像和装入副本映像。如果 **logarchmeth1** 参数设为 LOGRETAIN，并且发出了 **ARCHIVE LOG** 命令，那么即使条目出现在历史记录文件中并且 **auto_del_rec_obj** 设为 ON，也不会通过修改来删除日志文件。

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_stats_views**、**auto_prof_upd**、**auto_reorg** 和 **auto_sampling**）的父参数。

配置类型

数据库

适用于

- 带有本地客户机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
- 带有本地客户机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，将此参数设为 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，将此参数设为 OFF。要启用此参数，必须将它设为 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，将此参数设为 ON。

auto_runstats

此自动维护表参数将启用或禁用数据库的自动表 **RUNSTATS** 操作。**RUNSTATS** 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。要启用此参数，必须将它设为 ON，而且还必须启用其父参数。

缺省情况下，将此参数设为 ON。

auto_stats_prof

要点: V10.1 中不推荐使用自动统计信息概要分析功能，在以后的发行版中可能会将其除去。有关更多信息，请参阅《*DB2 V10.1 新增内容*》中的『不推荐使用自动统计信息概要分析功能』。

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，必须将它设为 ON，而且还必须启用其父参数。

缺省情况下，将此参数设为 OFF。

在分区数据库环境、某些联合数据库环境、DB2 pureScale 环境或者分区内并行性处于启用状态的环境中，不支持自动生成统计信息概要文件。如果启用了 **section_actuals** 数据库配置参数，那么无法启用自动统计信息概要文件生成 (SQLCODE -5153)。

auto_stmt_stats

此参数用于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“自动运行统计”功能仍将为 OFF。

假定同时启用了“自动运行统计”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
          自动运行统计                   (AUTO_RUNSTATS) = ON
            实时统计信息                   (AUTO_STMT_STATS) = ON
            统计视图                     (AUTO_STATS_VIEWS) = OFF
            自动抽样                       (AUTO_SAMPLING) = OFF
自动进行统计信息概要分析     (AUTO_STATS_PROF) = OFF
            统计信息概要文件更新         (AUTO_PROF_UPD) = OFF
          自动重组                      (AUTO_REORG) = ON
```

通过将 **auto_tbl_maint** 设为 OFF，可以临时禁用“自动运行统计”和“自动重组”功能。以后可以通过将 **auto_tbl_maint** 重置为 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，将此参数设为 ON。

auto_stats_views

此参数用于启用和禁用对统计视图的自动收集统计信息。如果此参数设为 ON，那么可自动维护统计视图的统计信息。

缺省情况下，将此参数设为 OFF。

auto_prof_upd

要点: V10.1 中不推荐使用自动统计信息概要分析功能，在以后的发行版中可能会将其除去。有关更多信息，请参阅《*DB2 V10.1 新增内容*》中的『不推荐使用自动统计信息概要分析功能』。

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，必须将它设为 ON，而且还必须启用其父参数。

缺省情况下，将此参数设为 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，必须将它设为 ON，而且还必须启用其父参数。

缺省情况下，将此参数设为 OFF。

auto_sampling

此参数控制收集大型表的统计信息时自动收集统计信息是否使用抽样。要启用自动抽样，请将 **auto_maint**、**auto_tbl_maint**、**auto_runstats** 和 **auto_sampling** 设为 ON。如果启用了自动收集统计信息，那么系统会使用页级别抽样并自动确定抽样率。系统会对基本表的数据页和索引页进行抽样（相对于统计视图，统计视图没有索引）。

缺省情况下，将此参数设为 OFF。

auto_reval -“自动重新验证和失效”配置参数

此配置参数控制重新验证和失效语义。

配置类型

数据库

参数类型

- 可联机配置
- 可由 DB2 pureScale 环境中的成员配置

缺省值 [范围]

DEFERRED [IMMEDIATE, DISABLED, DEFERRED, DEFERRED_FORCE]

缺省情况下，在创建新数据库时，此配置参数将设为 DEFERRED。

如果从 V9.5 或更低版本升级数据库，那么 **auto_reval** 将设为 DISABLED。重新生效行为与前发行版相同。

如果将此参数设为 IMMEDIATE，那么意味着所有从属对象在失效后都将立即重新生效。这适用于某些 DDL 语句，例如 ALTER TABLE、ALTER COLUMN 或者 CREATE OR REPLACE。从属对象的成功重新生效并不依赖于任何其他 DDL 更改；因此，重新生效操作可以立即完成。

如果将此参数设为 DEFERRED，那么意味着所有从属对象都将在它们下次被访问时重新生效。

注意，如果将此参数设为 IMMEDIATE 或 DEFERRED，并且任何重新生效操作失败，那么无效的从属对象在下次被访问之前将保持处于无效状态。

如果将此参数设为 DEFERRED_FORCE，其效果相当于将它设为 DEFERRED 并执行另一条启用了出错功能的 CREATE 命令。

在某些情况下，您显式指定的语法可能会覆盖 **auto_reval** 的设置。例如，如果使用 ALTER TABLE 语句的 DROP COLUMN 子句，但未指定 CASCADE 或 RESTRICT，那么语义将由 **auto_reval** 控制。但是，如果指定了 CASCADE 或 RESTRICT，那么将使用先前的级联或限制语义，从而覆盖 **auto_reval** 指定的新语义。

此配置参数是动态的，这意味着对它的值所作的更改将立即生效。您不必重新连接到数据库即可使更改生效。

autorestart -“允许自动重新启动”

autorestart 参数确定用户连接至先前异常终止的数据库时，数据库管理器是否自动启动崩溃恢复。如果未设置 **autorestart** 参数，那么用户需要先发出显式重新启动数据库命令才能连接至该数据库。

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 **autorestart** 参数设为 ON，那么必要时下一次数据库连接尝试会自动执行崩溃恢复。

在 DB2 pureScale 环境中，自动重新启动行为发生了更改。如果数据库成员失败，那么系统会对该成员自动启动成员崩溃恢复。此外，如果两个集群高速缓存设施同时失败，那么系统会自动启动组崩溃恢复。在成员崩溃恢复期间，不允许通过该成员对数据库进行任何连接。如果尝试通过需要崩溃恢复的成员连接至数据库，那么系统会返回 SQL1015N 错误。如果崩溃恢复失败，那么系统会在进行崩溃恢复时再次尝试。如果在进行崩溃恢复时第二次尝试连接失败，那么会对主机上的恢复失败的成员设置 ALERT，并在另一主机上对该成员执行轻量级重新启动。

如果 **autorestart** 参数设为 OFF，那么自动成员崩溃恢复和组崩溃恢复进程处于不活动状态，并且必须手动执行（如果需要）。可使用 **RESTART DATABASE** 命令启动崩溃恢复进程。

在 DB2 pureScale 环境中，如果成员失败并且无法在其原始主机上重新启动，那么 DB2 集群服务会在 DB2 pureScale集群内其他可用成员主机的其中一个上重新启动该成员。这称为轻量级重新启动处理。如果成员处于轻量级重新启动方式，那么您不能直接连接至该成员，因此不能进行手动重新启动。必须先将 **autorestart** 参数设置回 ON。然后轻量级重新启动成员会自动检测数据库配置参数中的更改并在必要时启动崩溃恢复。

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算在运行时将有多少缓冲池空间可用于选择的存取方案。

配置类型
数据库

参数类型

- 可联机配置
- 可由 DB2 pureScale 环境中的成员配置

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 数据库产品时，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，以便该优化器在判断缓冲池的可用性时应该更保守。

当此参数设为 AUTOMATIC 时，在创建数据库配置文件或者重置数据库配置文件时，会立即将此参数更新为适当的值。

设置此参数可以帮助优化器更准确地模拟缓冲池的使用量。如果您手动设置此参数，无论您运行的应用程序的平均数是多少，值都是从 2 开始。在您采用此设置来评估优化器的行为和测试应用程序的性能之后，可以按较小的增量来增大此参数的值。每当您

增大此参数的值时，请继续评估优化器的行为并测试应用程序的性能。将此参数的值设的太大可能会导致优化器低估可用于查询的缓冲池空间量。

更改此参数的值之后，请考虑使用 **REBIND PACKAGE** 命令重新绑定应用程序。

backup_pending -“备份暂挂指示器”

backup_pending 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

仅当将数据库配置更改为数据库从不可恢复变为可恢复时，此参数才设为 ON。即，最初 **logarchmeth1** 和 **logarchmeth2** 参数设为 OFF，然后设置这两个参数中的一个或全部，并接受对数据库配置的更新。

blk_log_dsk_ful -“日志磁盘满时阻止进行日志记录”

可以设置此参数以防止 DB2 数据库系统不能在活动日志路径中创建新日志文件时出现的“磁盘已满”错误。

配置类型

数据库

参数类型

- 可联机配置
- 可由 DB2 pureScale 环境中的成员配置

传播类 立即

缺省值 [范围]

No [Yes; No]

DB2 数据库系统将尝试以每 5 分钟一次的频率创建日志文件直到成功，而不是生成“磁盘已满”错误。在每次尝试之后，DB2 数据库系统会将消息写入“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** 设为 yes 会导致 DB2 数据库系统遇到“日志磁盘已满”错误时应用程序挂起，从而允许您解决该错误并允许事务完成。可通过增加文件系统容量或除去同一文件系统上大量文件来解决日志磁盘已满情况。

如果 **blk_log_dsk_ful** 设为 no，那么接收“日志磁盘已满”错误的事务将失败并回滚。

使用无限日志记录时（即，数据库配置参数 **logsecond** 设为 -1 时）将 **blk_log_dsk_ful** 设为 yes。对于无限日志记录，如果事务导致“日志磁盘已满”错误，那么数据库可在某些情况下脱机。

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允许对表激活 NOT LOGGED 或 NOT LOGGED INITIALLY 属性。

配置类型

数据库

参数类型

可联机配置

可由 DB2 pureScale 环境中的成员配置

缺省值 [范围]

No [Yes, No]

缺省情况下，**blocknonlogged** 设为 NO: 允许执行不进行日志记录的操作，这将减少日志记录工作，从而提高性能。但是，此配置有一些相关联的潜在缺点，在高可用性灾难恢复 (HADR) 数据库环境中尤其如此。DB2 HADR 数据库环境使用数据库日志将数据从主数据库复制到辅助数据库。主数据库允许不进行日志记录的操作，但不会将此类操作复制到备用数据库。如果对主数据库执行任何非日志记录操作，那么备用数据库必须重新初始化。例如，在执行不进行日志记录的操作之后，可以使用联机分割镜像或暂挂 I/O 支持对辅助数据库进行再同步。

使用说明

- 如果将 **blocknonlogged** 设为 YES，那么出现下列其中一个条件时 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。
  - CLOB、DBCLOB 或 BLOB 列按未记录方式定义。
- 如果 **blocknonlogged** 设为 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。

cf_catchup_trgt -“辅助集群高速缓存设施的同步复制目标时间”配置参数

此配置参数确定用于完成同步复制操作以使新添加或新重新启动的集群高速缓存设施处于与现有主集群高速缓存设施对等的状态的目标时间（以分钟计）。

配置类型

数据库

参数类型

可联机配置

缺省值 [范围]

15 [1 - 600]

计量单位

自辅助集群高速缓存设施启动以来的分钟数

此参数设置成员使新添加的辅助集群高速缓存设施进入与主集群高速缓存设施对等状态的目标时间。辅助集群高速缓存设施处于对等状态后，它能够在集群高速缓存设施失效或因为维护而关闭时接管主集群高速缓存设施的角色。如果将 **cf_catchup_trgt** 设置得较低，那么会导致成员积极执行同步复制活动，从而将在辅助集群高速缓存设施未准备好接管主项角色的情况下运行 DB2 pureScale实例的时间窗缩至最短。设置得越低，对数据库事务和系统整个性能的影响越大，因为会有更多的系统资源（例如，I/O 带宽）用于同步复制活动。

注: 尽管 **cf_catchup_trgt** 的缺省设置会在系统性能与高可用性之间取得平衡，但可使用以下基本原则来调整此参数:

- 如果可用性比性能重要，请对 **cf_catchup_trgt** 使用较低设置。
- 如果性能比可用性重要，请对 **cf_catchup_trgt** 使用较高设置。

cf_db_mem_sz -“数据库内存”配置参数

此参数控制此数据库的集群高速缓存设施（又称为 CF）的总内存限制。

配置类型

数据库

适用于 仅适用于 DB2 pureScale 实例。

- 带有本地客户机和远程客户机的数据库服务器

参数类型

可联机配置

缺省值 [范围]

AUTOMATIC [32768 - 4 294 967 295]

计量单位

页 (4 KB)

cf_gbp_sz、**cf_lock_sz** 和 **cf_sca_sz** 参数的 集群高速缓存设施结构内存限制之和必须小于 **cf_db_mem_sz** 参数的 CF 结构内存限制。在 **cf_db_mem_sz** 参数绑定的同一数据库内的 CF 资源（例如，组缓冲池 (GBP)、共享通信区 (SCA) 和锁定结构）之间自动调整内存。

不支持在数据库之间自我调整 CF 内存。

使用说明

- 将 **numdb** 配置参数的值设为高于或等于该实例上的数据库总数。此外，将 **DB2_DATABASE_CF_MEMORY** 的值设为允许此实例上的每个数据库（不管是通常处于不活动状态还是处于活动状态）同时处于活动状态的值。
- 如果手动设置 **cf_db_mem_sz** 的值，那么该值必须小于总 CF 服务器内存。总 CF 服务器内存由 **cf_mem_sz** 数据库管理器配置参数控制。
- 如果 **cf_db_mem_sz** 参数的先前值设为 AUTOMATIC，那么可使用 **GET DATABASE CONFIGURATION** 命令的 **SHOW DETAIL** 选项来确定当前值。
- 必须将 **cf_db_mem_sz** 配置参数与 **DB2_DATABASE_CF_MEMORY** 注册表变量和 **numdb** 配置参数协调使用。

限制

- 有附加 3840 个 4K 页（取自 **cf_mem_sz** 参数）将由 CF 内部使用。此外，**cf_mem_sz** 参数中会额外保留 256 个 4K 页以供实例中的每个活动数据库使用。仅需要在手动设置 **cf_mem_sz** 时考虑这些附加页。
- 如果将此参数的固定值改变为低于当前值的新值，那么该新值必须大于 **cf_gbp_sz**、**cf_lock_sz** 和 **cf_sca_sz** 结构参数的内存大小之和，否则该操作将失败。为避免此问题，请在尝试降低此参数之前降低个别结构内存大小。
- 如果将此参数的固定值改变为高于当前值的新值，那么除了 **cf_mem_sz** 参数定义的总 CF 服务器内存外，还会施加上限限制。通常，此上限由 **cf_db_mem_sz** 与 **cf_mem_sz** 参数值之间需要的 3600 页到 5000 页的内存缓冲区定义。**cf_db_mem_sz** 参数的值不应超过 **cf_mem_sz** 与此内存缓冲区之差。

cf_gbp_sz -“组缓冲池”配置参数

此参数确定集群高速缓存设施（又称为 CF）用于此数据库的组缓冲池 (GBP) 的内存大小。

配置类型
数据库

适用于 仅适用于 DB2 pureScale 实例。

- 带有本地客户机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

参数类型
可联机配置

缺省值 [范围]
AUTOMATIC [AUTOMATIC, 4096 - 4294967296]

计量单位
页 (4 KB)

分配时间
第一次对任何 DB2 成员连接或激活数据库时

释放时间
对最后一个 DB2 成员取消激活数据库时

GBP 中有两种主要类型的资源: 目录条目和数据元素。目录条目存储与页有关的元数据信息。数据元素存储实际页数据。

DB2 成员仍将页高速缓存到它们自己的本地缓冲池中。GBP 仅被本地缓冲区管理器用于维护系统间缓冲区相干性，不能直接被正在任何 DB2 成员上运行的 DB2 EDU 直接引用。

cf_lock_sz -“CF 锁定管理器”配置参数

此参数确定 CF 用于此数据库的锁定的内存大小。

配置类型
数据库

适用于 仅适用于 DB2 pureScale 实例。

- 带有本地客户机和远程客户机的数据库服务器

参数类型

可联机配置

缺省值 [范围]

AUTOMATIC [AUTOMATIC, 4096 - 1073741824]

计量单位

页 (4 KB)

分配时间

第一次对任何成员 激活数据库时。

注: 系统会在 CF 上而不是在任何成员上分配内存。如果有多个 CF，那么每个 CF 分配相同内存量。

释放时间

在所有成员上取消激活数据库时。

这是全局配置参数，意味着对 DB2 pureScale 实例中的每个成员使用相同值。

请注意，CF 上的内存仅分配一次；在任何成员上第一次激活数据库时。可使用 **current_cf_lock_size** 监视元素来监视 CF 锁定内存的消耗。

cf_sca_sz -“共享通信区”配置参数

此共享通信区 (SCA) 配置参数确定集群高速缓存设施 (CF) 中的 SCA 使用的内存大小。该 SCA 对应每个数据库实体，并且包含表、索引、表空间和目录的数据库范围控制块信息。

配置类型

数据库

适用于 DB2 pureScale 环境

参数类型

可联机配置

缺省值 [范围]

AUTOMATIC [AUTOMATIC, 2048 - 1073741824]

计量单位

页 (4 KB)

分配时间

第一次对任何 DB2 成员激活数据库的时间

释放时间

删除该数据库或停止 CF 服务器的时间

如果 **cf_sca_sz** 参数设为 AUTOMATIC（缺省值），那么 DB2 数据库管理器会在对任何成员第一次激活数据库时计算内存值，此值应足以供基本数据库操作使用。如果要为该 SCA 配置准确的内存大小，那么可将 **cf_sca_sz** 参数设为固定数字值。

可通过运行 **GET DB CFG SHOW DETAIL** 命令来获取 SCA 大小的当前值。如果 **cf_sca_sz** 参数设为 AUTOMATIC，那么 SHOW DETAIL 子句会显示 SCA 的计算值和分配值。

如果 SCA 内存已被数据库操作用光，那么会返回内存不足错误消息。在这种情况下，可通过将 **cf_sca_sz** 参数设为较高值来增加 SCA 的大小。

有关更多详细信息，请参阅『为数据库配置集群高速缓存设施内存』。

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
- 可由 DB2 pureScale 环境中的成员配置

传播类 立即

缺省值 [范围]

-1 [**maxappls***5, 8 - 524 288]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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）来获取先前检索的信息，从而降低其处理时间。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。在这种情况下，缺省值 -1 会将 **catalogcache_sz** 设为 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 -“已更改页数阈值”

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
- 可由 DB2 pureScale 环境中的成员配置

缺省值 [范围]

60 [5 - 99]

注：可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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。

如果设置了 **DB2_USE_ALTERNATE_PAGE_CLEANING** 注册表变量（即，使用了进行页清理的另一种方法），那么 **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。对于语言感知整理或区分语言环境的 UCA 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 4.0.0 的 Unicode 整理算法 (UCA)，并且规范化隐式设为 ON。
- **X’8002’** - 顺序为 UCA400_LTH，它实施基于 Unicode 标准版本 4.0.0 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 4.0.0 的 Unicode 整理算法 (UCA)，并对所有斯洛伐克语字符作适当排序。

注:

- 对于语言感知整理或区分语言环境的 UCA 整理，前 256 个字节包含整理名的字符串表示。
-

  要点: V10.1 中已经不推荐使用基于 Unicode 标准 V4.0.0 的 Unicode 整理算法的整理，在以后的发行版中可能会将它们除去。有关更多信息，请参阅《*DB2 V10.1 新增内容*》中的『不推荐使用基于 Unicode 标准 V4.0.0 的 Unicode 整理算法的整理』。

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。

connect_proc - 连接过程名称数据库配置参数

此数据库配置参数使您可以输入或更新一个由两部分组成的连接过程名称，每次应用程序连接到数据库时都将执行该连接过程。

配置类型

数据库

参数类型

可联机配置

可由 DB2 pureScale 环境中的成员配置

缺省值 NULL

必须遵循以下连接过程约定，否则将返回错误。

- 非零长度字符串必须指定由两个部分组成的过程名称（即，[schema name].[procedure name]）
- 连接过程名称（模式名称及过程名称）只能包含以下字符:
  - A-Z
  - a-z
  - _（下划线）
  - 0-9
- 此外，模式名称和过程名称需要遵循普通标识符的规则。

一旦将 **connect_proc** 参数配置为一个长度不为零的值之后，服务器将隐式执行在每个新连接上指定的过程。

使用说明

- 更新此参数时，需要连接到数据库。但是，如果已取消激活数据库，那么取消设置此参数并不需要连接。
- 只能使用 **UPDATE DATABASE CONFIGURATION** 命令的 **IMMEDIATE** 选项来设置 **connect_proc** 参数。设置 **connect_proc** 参数时，不能使用 **DEFERRED** 选项。

- 只有参数个数正好为零的过程才能用作连接过程。只要设置了 **connect_proc** 参数，数据库中就不能存在任何其他具有由两部分组成的相同名称的过程。
- 在更新 **connect_proc** 参数之前，数据库中必须存在连接过程。如果数据库中不存在带有零个参数的连接过程，或者由多个具有相同名称的连接过程，那么 **UPDATE DATABASE CONFIGURATION** 命令将失败，且带有错误。
- 请在数据分区环境的所有分区上使用同一连接过程。

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

在数据库升级未完成或失败的情况下，此参数将反映该数据库在升级前所处的发行版级别，并且可能与 **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

- 带有本地客户机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
- 带有本地客户机和远程客户机的分区数据库服务器

参数类型

- 可联机配置
- 可由 DB2 pureScale 环境中的成员配置

  注: 如果此参数在 DB2 pureScale 环境中设为不同值，那么不应启用 **self_tuning_mem**。

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

取消激活数据库时

将此参数设为 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V9.7 中，将 **database_memory** 配置参数设为 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）。
- 在 Solaris 操作系统上，如果在 DB2 V9.7 中将 **database_memory** 设为 AUTOMATIC，那么数据库管理器将使用可分页内存作为数据库共享内存。在基于 UltraSPARC 的 Solaris 操作系统中，如果有 64 KB 内存页可用，那么数据库管理器将尝试使用这些内存页。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，如果在 DB2 V9.7 中将 **database_memory** 设为 COMPUTED 或者设为一个数值，那么数据库管理器将使用私有的共享内存（ISM）和大页作为数据库共享内存。

控制 DB2 内存消耗:

instance_memory 设为 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。启用自调整内存管理器以执行 **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。

一些 Linux[1] 内核上的限制:

由于一些 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** 设为 AUTOMATIC。但是目前，只有在 **instance_memory** 设为特定值而不是 AUTOMATIC 时，才允许在这些内核上使用此设置。如果 **database_memory** 设为 AUTOMATIC，并且稍后将 **instance_memory** 设置回为 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，不管 **instance_memory** 参数的设置如何都是如此。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。

dbheap -“数据库堆”

此参数确定数据库堆使用的最大内存。

对于 V9.5，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
- 可由 DB2 pureScale 环境中的成员配置

传播类 立即

缺省值 [范围]

Automatic [32 - 524 288]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

计量单位

页 (4 KB)

分配时间

当数据库激活时

释放时间

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 per tablespace + 4K per table + (1K + 4*extents used),
  per range clustered table (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。在这种情况下，建议您将数据库堆配置设为 AUTOMATIC，以便数据库管理器将自动管理数据库堆大小。

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
- 可由 DB2 pureScale 环境中的成员配置

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要求后，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** 注册表变量为大型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。

date_compat -“日期兼容性”数据库配置参数

此参数指示与 TIMESTAMP(0) 数据类型相关联的 DATE 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 DATE 数据类型的 **DB2_COMPATIBILITY_VECTOR** 注册表变量的设置。值不可更改。

dec_to_char_fmt -“小数到字符函数”配置参数

此参数控制用于将小数值转换为字符值的 CHAR 标量函数和 CAST 规范的结果。

配置类型

数据库

参数类型

- 可联机配置
- 可由 DB2 pureScale 环境中的成员配置

请参阅更改 dec_to_char_fmt 的效果。

缺省值 [范围]

NEW [NEW, V95]

此参数的设置确定 CHAR 函数的结果是否包括前导零和尾部小数点字符。如果将此参数设为 NEW，那么将不包括前导零和尾部小数点字符；如果将此参数设为 V95，那么将包括前导零和尾部小数点字符。

CHAR_OLD 标量函数的结果也将包括前导零和尾部小数点字符，该函数的语法与 CHAR 函数相同。

升级过程中，对于在 V9 之前创建并已升级到 V9.7 或更高版本的数据库，缺省情况下 **dec_to_char_fmt** 参数设为 *V95*。

更改 dec_to_char_fmt 的值的效果

- 您在 V9.7 以前创建的具体化查询表 (MQT) 中包含的结果可能与您使用 NEW 设置创建的 MQT 不同。要确保先前创建的 MQT 只包含符合新格式的数据，请使用 REFRESH TABLE 语句来刷新这些 MQT。
- 触发器的结果可能会受更改后的格式影响。将此参数的值设为 NEW 以更改格式不会影响已写入的数据。
- 如果进行重新评估，那么允许将数据插入到表中的约束可能会拒绝同一数据。同样，如果进行重新评估，那么不允许将数据插入到表中的约束可能会接受同一数据。请使用 SET INTEGRITY 语句来检查并更正表中可能不再满足约束的数据。
- 对于结果受 **dec_to_char_fmt** 值的更改影响的生成列，在更改 **dec_to_char_fmt** 的值之后，请重新编译所有依赖于该列中的值的静态 SQL 程序包。要确定哪些静态 SQL 程序包受影响，必须编译所有程序包，然后使用 **db2rbind** 命令重新绑定这些程序包。

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和 **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

请参阅第 702 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 数据库系统支持五种符合 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更改；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 数据库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 136 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 136. 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 -“缺省级别”

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
- 可由 DB2 pureScale 环境中的成员配置

传播类 连接

缺省值 [范围]

1 [-1(ANY), 1 - 32 767]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

缺省值为 1。

值为 1 意味着没有分区内并行性。值为 -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]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

计量单位
页

在创建表空间时，可选择指定 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。

在 DB2 pureScale 环境中，应使用至少为缺省值（32 页）的扩展数据块大小。为表或索引添加扩展数据块时，此最小扩展数据块大小减少了 DB2 pureScale 环境中的内部消息流量。频繁分配新扩展数据块和较大扩展数据块大小有益的示例包括装入和导入实用程序、CREATE INDEX 语句和从应用程序中成批插入操作。

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

应将此参数的值设为使用原始 **LOAD** 命令中的 COPY YES 选项指定的 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 - 32 767]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

计量单位
页

创建表空间时，可选择将 PREFETCHSIZE 的值指定为 AUTOMATIC 或者 *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。

如果在预取大小设为 AUTOMATIC 的情况下创建表空间，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，那么可将 **dft_prefetch_sz** 设为 16（32 的分数）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 **BIND** 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
- 可由 DB2 pureScale 环境中的成员配置

传播类 连接

缺省值 [范围]
5 [0 - 9]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

计量单位
查询优化类（请参阅以下列表）

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，可限制选择存取方案所消耗的成本。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)]

计量单位
秒

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)，并且 QUERY OPTIMIZATION 类的值为 2、5 或更高，那么系统会考虑使用 REFRESH DEFERRED 具体化查询表来优化动态查询的处理。

dft_schemas_dcc -“针对新模式的缺省数据捕获”配置参数

此参数允许控制针对新创建的模式的 DATA CAPTURE CHANGES 的缺省设置以用于复制。

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]
No [Yes; No]

缺省情况下，**dft_schemas_dcc** 设为 NO。如果设为 YES，那么所有新创建的模式在缺省情况下将具有 DATA CAPTURE CHANGES 子句。

dft_sqlmathwarn -“出现算术异常时继续”

此参数设置缺省值，在 SQL 语句执行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。该插入操作失败，因为 DB2 数据库管理器不能检查约束。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 数据库管理器评估该约束。当 **dft_sqlmathwarn** 为 Yes 时使用 B=0 设置插入的行将保留在该表中且可供查询。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：除非您专门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更多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耗用的时间。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。

enable_xmlchar -“启用以 XML 为目标的转换”配置参数

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”

此参数指定一个路径，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 数据库系统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

- 带有本地客户机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
- 带有本地客户机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”

从 V9.5 起建议不要使用此参数，但是 V9.5 之前的数据服务器和客户机仍然可以使用此参数。DB2 V9.5 或更高发行版中的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 V9.5 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
- 带有本地客户机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
- 带有本地客户机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[6]

缺省值 Null

6. 对此参数的更改在数据库激活时生效。如果数据库已联机，那么可通过在主数据库上停止然后重新启动 HADR 来使更改生效。

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”。

使用说明

- 如果主项和辅助项启动但未相互连接，并且 DB2 诊断日志中未出现任何错误指示，那么可能是因为主机名解析时出现小问题。如果使用主机名配置了 HADR，请改用 IP 地址重试。
- 如果主项或备用项驻留在 NAT（网络地址转换）设备后的专用网络中，那么您可能需要将注册表变量 **DB2_HADR_NO_IP_CHECK** 设为 ON。请参阅 HADR 和网络地址转换 (NAT) 支持

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

- 带有本地客户机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[7]

缺省值 Null

主数据库系统或备用数据库系统上的 **hadr_local_svc** 值不能与它们相应主机上的 **svcename** 或 **svcename** +1 值相等。

如果正在使用 SSL，那么不要在主数据库系统或备用数据库系统上将 **hadr_local_svc** 与 **ssl_svcename** 设为相同的值。

hadr_peer_window -“HADR 对等窗口”配置参数

如果将 **hadr_peer_window** 设为一个非 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[8]

7. 对此参数的更改在数据库激活时生效。如果数据库已联机，那么可通过在主数据库上停止然后重新启动 HADR 来使更改生效。

8. 对此参数的更改在数据库激活时生效。如果数据库已联机，那么可通过在主数据库上停止然后重新启动 HADR 来使更改生效。

缺省值 [范围]

0 [0 – 4 294 967 295]

计量单位

秒

用法说明:

- 该值在主数据库和备用数据库上需要相同，除非使用 HADR 多个备用方式。使用多个备用方式，主体备用数据库会使用主数据库的设置 **hadr_peer_window**，且辅助备用数据库上的任何设置 **hadr_peer_window** 都会加以忽略。
- 建议最小值为 120 秒。
- 当 **hadr_syncmode** 值设为 ASYNC 或 SUPERASYNC 时，将忽略 **hadr_peer_window** 值。
- 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停用备用数据库，就不会调用对等窗口。
- 仅当 HADR 备用数据库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 秒钟之内未实现互相同步，那么使用 **hadr_peer_window** 参数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。
- 在备用数据库上，对等窗口结束时间是备用数据库从主数据库接收到的最后一个脉动信号消息中指定的时间，但与备用数据库检测连接丢失的时间没有直接关系。

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

- 带有本地客户机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[9]

缺省值 Null

与 **hadr_local_host** 相似，此参数必须只映射至一个 IP 地址。

9. 对此参数的更改在数据库激活时生效。如果数据库已联机，那么可通过在主数据库上停止然后重新启动 HADR 来使更改生效。

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。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

- 带有本地客户机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[10]

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

- 带有本地客户机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[11]

缺省值 Null

hadr_replay_delay -“HADR 重演延迟”配置参数

此参数指定从事务在主数据库上落实到事务在备用数据库上落实之间必须经历的秒数。

配置类型

数据库

适用于

- 带有本地客户机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]

0 [0 至 2147483647]

---

10. 对此参数的更改在数据库激活时生效。如果数据库已联机，那么可通过在主数据库上停止然后重新启动 HADR 来使更改生效。

11. 对此参数的更改在数据库激活时生效。如果数据库已联机，那么可通过在主数据库上停止然后重新启动 HADR 来使更改生效。

计量单位

秒

hadr_replay_delay 配置参数对 HADR 备用数据库启用*延迟重演*，这意味着备用数据库有意拖延 HADR 主数据库，因为来自主数据库中的日志重演。备用数据库必须处于 SUPERASYNC 方式，**hadr_replay_delay** 才能设为非零值。不能在主数据库上设置此参数。必须先在备用数据库上禁用延迟重演，备用数据库才能作为主数据库接管。

如果启用延迟重演，那么还建议通过设置 **hadr_spool_limit** 数据库配置参数来启用日志假脱机。因为有意延迟，那么重演位置可能远远落后于备用数据库上的日志接收位置。如果没有假脱机，那么日志接收超过重演的部分只能是接收缓存量。如果有假脱机，那么备用数据库可接收的日志量大大超过重演位置，从而提供更多保护以避免主数据库发生故障时数据丢失。请注意，在任一情况下，因为强制 SUPERASYNC 方式，所以主数据库不会因为延迟重演而受阻。

对于延迟重演，如果主数据库上有错误事务并且在备用数据库上重演之前进行了通知，那么可将数据库前滚至刚好在落实错误事务之前，然后停止前滚以检索主数据库上丢失的数据。

注: 如果已在备用数据库上启用重演，那么直到您禁用延迟重演（在该备用数据库上将 **hadr_replay_delay** 参数设为 0），备用数据库才能作为新主数据库接管。

hadr_spool_limit -“HADR 日志假脱机限制”配置参数

此参数确定允许假脱机至 HADR 备用数据库上的磁盘的最大日志数据量。

配置类型

数据库

适用于

- 带有本地客户机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[12]

缺省值 [范围]

0 [-1 - 2 147 483 647]

计量单位

页 (4 KB)

hadr_spool_limit 配置参数允许在 HADR 备用数据库上进行日志假脱机。日志假脱机允许 HADR 主数据库上的事务继续进行而不必等待 HADR 备用数据库上的日志重演。于是主项发送的日志数据会写至或*假脱机*至备用项上的磁盘（如果它在日志重演中落后）。备用项稍后可从磁盘读取日志数据。这允许系统更能容许主项上的事务时间峰值或备用项上的日志重演变慢（因为特定类型的日志记录重演）。

使用日志假脱机时，确保已对备用数据库的活动日志路径提供足够的磁盘空间。必须有足够磁盘空间来容纳活动日志，磁盘空间由 **logprimary**、**logsecond**、**logfilsiz** 和 **hadr_spool_limit** 配置参数确定。

12. 对此参数的更改在数据库激活时生效。如果数据库已联机，那么可通过在主数据库上停止然后重新启动 HADR 来使更改生效。

缺省值 0 意味着没有假脱机。这意味着备用项可落后于主项（最多日志接收缓冲区的大小）。如果缓冲区变满，那么主项上的新事务可能被阻止，因为主系统不能再向备用系统发送任何日志数据。

值 -1 意味着无限假脱机（可用磁盘空间支持的任意量）。如果您对 **hadr_spool_limit** 使用较高值，那么应考虑主项的日志位置与备用项的日志重演之间存在较大间隔的情况，此时可能导致接管时间较长（因为在假脱机日志的重演完成之前，该备用项不能充当新备用项的角色）。

请注意，使用日志假脱机不会损害 HADR 功能提供的 HADR 保护。通过使用指定同步方式，主项中的数据仍以日志形式复制到备用项；可能要花一点时间（通过日志重演）将该数据应用至表空间。

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

- 带有本地客户机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[13]

缺省值 [范围]

NEARSYNC [ASYNC, SUPERASYNC, 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，并且备用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

与 SYNC 和 NEARSYNC 方式相比，ASYNC 方式的事务响应时间更短，但如果主数据库发生故障，那么事务失败的可能性更大。

13. 对此参数的更改在数据库激活时生效。如果数据库已联机，那么可通过在主数据库上停止然后重新启动 HADR 来使更改生效。

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。

SUPERASYNC

此方式具有最短的事务响应时间，但如果主系统发生故障，那么事务失败的可能性也最大。当您不希望事务由于网络中断或拥塞而受到阻塞或经历较长的响应时间时，此方式很有用。

在此方式中，HADR 对永远不会处于对等状态或断开对等状态。仅当日志记录已写入主数据库上的日志文件时，才会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。

图 53 显示了根据所选同步方式何时认为事务的日志是成功的:

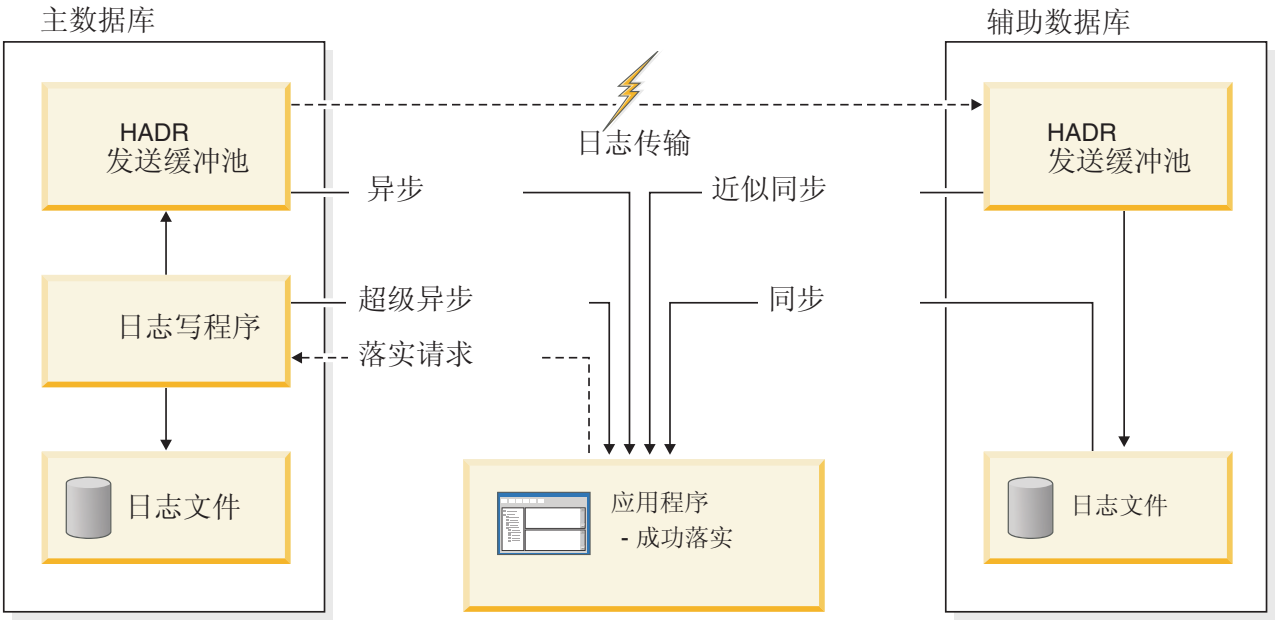


图 53. 高可用性和灾难恢复（HADR）的同步方式

hadr_target_list -“HADR 目标列表”数据库配置参数

此参数（允许 HADR 在多备用方式下运行）指定包含最多三个充当 HADR 备用数据库的目标 *host*:*port* 对的列表。

配置类型

数据库

适用于

- 带有本地客户机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

可联机配置

缺省值 NULL

为了让 HADR 以多备用方式运行，必须在所有 HADR 数据库上设置 **hadr_target_list** 配置参数。不能通过设置此参数而不关闭主数据库来从单备用方式切换至多备用方式。您对主数据库的 **hadr_target_list** 配置参数指定的 *host*:*port* 对将确定哪些主机充当该主数据库的备用数据库。如果备用数据库作为新 HADR 主数据库接管，那么此备用数据库的目标列表中的 *host*:*port* 对将标识要使用的备用主机。

缺省设置 NULL 指示 HADR 将在单备用方式下运行。

与 **hadr_remote_host** 和 **hadr_local_host** 数据库配置参数一样，您对 **hadr_target_list** 配置参数指定的主机可以是主机名或 IP 地址。主机名只能包含字母数字、短杠和下划线。与 **hadr_remote_svc** 和 **hadr_local_svc** 配置参数一样，您对 **hadr_target_list** 配置参数指定的端口可以是端口号或服务名称。服务名称可由任意字符组成。主机名将映射至 IP 地址，服务名称将映射至端口号。您对 **hadr_target_list** 配置参数指定的值可包含主机名、主机 IP 地址、服务名称和端口值的组合。按以下格式指定 *host*:*port* 对:

host1:*port1*|*host2*:*port2*|*host3*:*port3*

必须将数字因特网协议 V6 (IPv6) 地址括在方括号 ([]) 中以区分 IP 部分与端口部分，如以下示例所示:

```
[FEDC:BA98:7654:3210:FEDC:BA98:7654:3210]:4000
```

HADR 数据库的目标列表中的第一个条目被指定为*主体 HADR 备用数据库*，任何其他条目称为*辅助 HADR 备用数据库*。可配置主体备用数据库和主数据库以使用任何同步方式；但是，主体备用数据库的同步方式是通过主数据库上 **hadr_syncmode** 配置参数的设置确定的。辅助备用数据库目标的同步方式始终为 SUPERASYNC。在主数据库上启动 HADR 时，**hadr_remote_host** 和 **hadr_remote_svc** 配置参数的值自动设为主体备用数据库的对应值，除非您将 **DB2_HADR_NO_IP_CHECK** 注册表变量设为 ON。

不需要对称或互补设置。例如，数据库 A 可将数据库 B 指定为其主体备用数据库，数据库 B 可将数据库 C 指定为其主体备用数据库。但是，必须能够在任何数据库对之间进行角色切换。例如，如果数据库 B 列示在数据库 A 的目标列表中，那么数据库 A 必须列示在数据库 B 的目标列表中。

尽管 **hadr_target_list** 参数可更新并且在数据库联机时生效，但 HADR 处于活动状态仍有一些限制:

- 必须先在主数据库上停止 HADR，才能更改主数据库的主体备用数据库。
- 如果备用数据库连接至主数据库，那么不能从列表中除去该备用数据库。要断开备用数据库的连接，只需要将其取消激活。然后可从主数据库的目标列表中将其除去。
- 不能动态更新备用数据库的 **hadr_target_list** 配置参数，除非您已启用 HADR 的“读取备用项”功能。
- 如果备用数据库已连接至主数据库，那么不能从备用数据库的目标列表中除去主数据库。
- 目标列表必须包含 IPv4 或 IPv6 格式的 IP 地址，但不能同时包含两者。

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

- 带有本地客户机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[14]

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，以及在前滚期间或在备用数据库上重放高可用性灾难恢复 (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

- 带有本地客户机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
- 带有本地客户机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

使用数据库管理器配置文件中指定的系统设置来决定何时重建无效的索引，以及在前滚或 HADR 日志重放期间是否将重做任何索引构建日志记录。

注: 此设置仅对数据库配置有效。

14. 对此参数的更改在数据库激活时生效。如果数据库已联机，那么可通过在主数据库上停止然后重新启动 HADR 来使更改生效。

ACCESS

第一次访问基础表时将重建无效的索引。在前滚或 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。对新主存储器上基础表的访问会导致索引重建，这会导致写入日志记录，然后将其发送到新的备用存储器，进而导致索引在备用存储器上无效。

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: 在 DB2 pureScale 环境中，仅在组崩溃恢复期间（而不会在成员崩溃恢复中）重建索引。

注: 如果应用程序连接至数据库时该数据库异常终止，并且已启用 autorestart 参数，那么应用程序连接至数据库时会隐式发出 **RESTART DATABASE** 命令。如果未发出该命令，那么下次访问底层表时会重建无效索引。

RESTART_NO_REDO

将在显式或隐式发出 **RESTART DATABASE** 命令时重建无效的索引。在前滚或 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。接管会导致在新主存储器的基础表上重建索引，这会导致写入日志记录，然后将其发送到新的备用存储器，进而导致索引在备用存储器上无效。

如果应用程序连接至数据库时该数据库异常终止，并且已启用 autorestart 参数，那么应用程序连接至数据库时会隐式发出 **RESTART DATABASE** 命令。如果未发出该命令，那么下次访问底层表时会重建无效索引。

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，如果索引发生此问题，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下降。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。

如果将此参数设为 RESTART，那么重新启动数据库所耗用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（例如，**CREATE INDEX** 和 **REORG INDEX** 操作）或对索引重建操作激活完全日志记录时，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更改表时启用 LOG INDEX BUILD 来激活日志记录。通过将 **indexrec** 设为 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 解释器使用的环境变量是根据此参数的值计算出来的。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
- x64 和 IPF 上的 64 位 Windows
- AMD64 和 Intel EM64T 系统 (x64)、POWER 和 zSeries 上的 64 位 Linux 内核。

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 V8 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
- 可由 DB2 pureScale 环境中的成员配置

传播类 立即

缺省值 [范围]

Automatic [4 - 134217728]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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行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。

尽管 **locklist** 的值可与 **maxlocks** 参数一起调整，但禁止自调整 **locklist** 参数不会自动禁止自调整 **maxlocks** 参数。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。

仅当对数据库启用了自调整内存功能（**self_tuning_mem** 配置参数设为 ON）时，才会自动调整此配置参数。

在所有平台上，每个锁定需要 128 或 256 字节的锁定列表，这取决于是否对该对象挂起了其他锁定:

- 对于未挂起其他锁定的对象，挂起一个锁定需要 256 个字节。
- 对于存在一个挂起的锁定的对象，记录一个锁定需要 128 个字节。

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。此计算结果是假定仅存在共享锁定时的近似值。通过将应用程序挂起的锁定数乘以在挂起其他锁定的对象上挂起一个锁定所需的值，可以算出所使用的锁定列表的百分比。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。请参阅 **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 * 128 * maxappls) / 4096
   ```
   b. 启用集中器:
   ```
   (512 * 128 * max_coordagents) / 4096
   ```
   c. 在启用集中器的分区数据库中:
   ```
   (512 * 128 * 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算值，而 128 是针对具有现有锁定的对象的每个锁定所需要的字节数。
2. 计算锁定列表大小的上限:
   ```
   (512 * 256 * maxappls) / 4096
   ```
   其中 256 是针对对象的第一个锁定所需的字节数。

   注：而在 DB2 pureScale 环境中，将 **locklist** 配置参数设为等于此步骤中计算的值的上限，并且是存在于当前连接的数据库中所有缓冲池总页数的 3%。
3. 估算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. 按下文所述，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** 设为 AUTOMATIC，那么也应该将 **locklist** 设为 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。请参阅 **locks_held_top**（最大挂起锁定数）监视元素。

此信息可帮助您验证或调整每个应用程序的估算锁定数。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**。

在更改此参数之后，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

如果将此参数设为 0，那么不会等待锁定。在这种情况下，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。

如果将此参数设为 -1，那么锁定超时检测关闭。在这种情况下，将等待锁定（如果请求时无可用锁定），直到发生下列事件中的一件:

- 授予锁定
- 发生死锁。

注: 为此配置参数指定的值不用于控制事件监视器目标表的锁定超时。事件监视器使用一个单独的非可配置设置，该设置将导致事件监视器表上的锁定超时。

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_appl_info -“应用程序信息日志记录”数据库配置参数

此参数指定应用程序信息日志记录在每个更新事务启动时写入。

配置类型

数据库

参数类型

可配置

缺省值 [范围]

No [Yes; No]

设为 YES 时，会在每个更新事务启动时添加额外的日志记录。该应用程序信息日志记录用于事务（请参阅日志记录头）。

如果有使用 DATA CAPTURE CHANGES 选项创建的用于复制（或其他用途，例如，审计）的表，请启用 **log_appl_info** 配置参数。

log_ddl_stmts -“日志数据定义语言 (DDL) 语句数”数据库配置参数

此参数指定有关数据定义语言 (DDL) 语句的额外信息将写入日志。

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]

log_ddl_stmts 配置参数的缺省设置可避免为 DDL 操作编写额外日志记录的开销。如果您需要复制 DDL 操作，且使用复制捕获程序从该日志捕获更改，那么将此参数设为 YES。

log_retain_status -“日志保留状态指示器”

如果 **logarchmeth1** 数据库配置参数设为 **logretain**，那么日志保留状态参数将显示值 RECOVERY，否则它将显示值 NO。

配置类型

数据库

参数类型

参考

logarchcompr1 -“主归档日志文件压缩”配置参数

此参数指定是否压缩写至日志的主归档目标的日志文件。

配置类型

数据库

适用于

- 带有本地客户机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
- 带有本地客户机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 指定不压缩写至日志的主归档目标的日志文件。

ON 指定压缩写至日志的主归档目标的日志文件。如果以动态方式设置此参数，那么不会压缩已归档的日志文件。

注意:

- 如果将 **logarchmeth1** 配置参数设为 DISK、TSM 或 VENDOR 以外的值，那么日志归档压缩不起作用，不管 **logarchcompr1** 配置参数设置如何都是如此。

logarchcompr2 -"辅助归档日志文件压缩"配置参数

此参数指定是否压缩写至日志的辅助归档目标的日志文件。

配置类型

数据库

适用于

- 带有本地客户机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
- 带有本地客户机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 指定不压缩写至日志的辅助归档目标的日志文件。

ON 指定压缩写至日志的辅助归档目标的日志文件。如果以动态方式设置此参数，那么不会压缩已归档的日志文件。

注意:

- 如果将 **logarchmeth2** 配置参数设为 DISK、TSM 或 VENDOR 以外的值，那么日志归档压缩不起作用，不管 **logarchcompr2** 配置参数设置如何都是如此。

logarchmeth1 -"主日志归档方法"

此参数指定从当前日志路径归档的日志的主目标的介质类型。

配置类型

数据库

适用于

- 带有本地客户机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

- 带有本地客户机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。如果将 **logarchmeth1** 和 **logarchmeth2** 配置参数都设为 OFF（这是缺省值），那么数据库被认为正在使用循环日志记录，且不可前滚恢复。

LOGRETAIN

指定活动日志文件将保留并且变为联机归档日志文件以在前滚恢复中使用。

USEREXIT

指定执行日志保留时间记录并应使用用户出口程序对日志文件进行归档和检索。日志文件是在变满时进行归档的。日志文件是在 **ROLLFORWARD** 实用程序必须使用日志文件复原数据库时进行检索的。

DISK 此值必须后跟冒号 (:)，然后是将在其中对日志文件归档的现有路径的标准名称。例如，如果您将 **logarchmeth1** 配置参数设为 DISK:/u/dbuser/archived_logs，那么归档日志文件将放置在 /u/dbuser/archived_logs/*instance*/*dbname*/*nodename*/*logstream*/*chainid*/ 目录中。

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 指示应使用缺省管理类在本地 TSM 服务器上对日志文件归档（如果指定时不带任何其他配置参数）。如果此选项后跟冒号 (:) 和 TSM 管理类，那么应使用指定的管理类对日志文件进行归档。

如果使用 TSM 对日志进行归档，那么在使用数据库配置参数指定的管理类之前，TSM 会尝试将该对象绑定至您在 TSM 客户机选项文件中的 INCLUDE-EXCLUDE 列表中指定的管理类。如果找不到匹配项，那么将使用您在 TSM 服务器上指定的缺省 TSM 管理类。然后，TSM 将该对象重新绑定至您对数据库配置参数指定的管理类。因此，缺省管理类和您对数据库配置参数指定的管理类必须包含归档副本组，否则归档操作会失败。TSM 条目的示例包括:

- 如果指定了管理类: `db2 update db cfg for mydb using logarchmeth1 TSM:DB2_LOGS`
- 如果未指定管理类: `db2 update db cfg for mydb using logarchmeth1 TSM`

VENDOR 指定使用供应商库对日志文件进行归档。此值必须后跟冒号 (:) 和库名。库中的 API 必须对供应商产品使用备份和复原 API。供应商条目的示例为: `db2 update db cfg for mydb using logarchmeth1 VENDOR:/home/dbuser/vendorLib/<library name>`

Notes:

- 如果将 **logarchmeth1** 或 **logarchmeth2** 配置参数设为 OFF 以外的值，那么配置数据库以进行前滚恢复。

- 如果对 **logarchmeth1** 配置参数使用 userexit 或 logretain 选项，那么必须将 **logarchmeth2** 配置参数设为 OFF。
- 要指定包含空格的归档路径，请使用 **db2CfgSet** API。

logarchmeth2 -“辅助日志归档方法”

此参数指定从当前日志路径或镜像日志路径归档的日志的辅助目标的介质类型。

配置类型

数据库

适用于

- 带有本地客户机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
- 带有本地客户机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 [DISK, TSM, VENDOR]

OFF 指定不使用日志归档方法。如果将 **logarchmeth1** 和 **logarchmeth2** 配置参数都设为 OFF，那么数据库被认为正在使用循环日志记录，且不可前滚恢复。这是缺省情况。

DISK 此值必须后跟冒号 (:)，然后是将在其中对日志文件归档的现有路径的标准名称。例如，如果您将 **logarchmeth1** 配置参数设为 DISK:/u/dbuser/archived_logs，那么归档日志文件将放在 /u/dbuser/archived_logs/*instance*/*dbname*/*nodename*/*chainid*/ 目录中。

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 指示应使用缺省管理类在本地 TSM 服务器上对日志文件归档（如果指定时不带任何其他配置参数）。如果此选项后跟冒号 (:) 和 TSM 管理类，那么应使用指定的管理类对日志文件进行归档。

如果使用 TSM 对日志进行归档，那么在使用数据库配置参数指定的管理类之前，TSM 会尝试将该对象绑定至您在 TSM 客户机选项文件中的 INCLUDE-EXCLUDE 列表中指定的管理类。如果找不到匹配项，那么将使用您在 TSM 服务器上指定的缺省 TSM 管理类。然后，TSM 将该对象重新绑定至您对数据库配置参数指定的管理类。因此，缺省管理类和您对数据库配置参数指定的管理类必须包含归档副本组，否则归档操作会失败。

VENDOR

指定使用供应商库对日志文件进行归档。此值必须后跟冒号 (:) 和库名。库中的 API 必须对供应商产品使用备份和复原 API。

Notes:

1. 如果将 **logarchmeth1** 或 **logarchmeth2** 配置参数设为 OFF 以外的值，那么配置数据库以进行前滚恢复。

2. 如果对 **logarchmeth1** 配置参数使用 **userexit** 或 **logretain** 选项，那么必须将 **logarchmeth2** 配置参数设为 OFF。

如果使用 **logarchmeth2** 配置参数指定路径，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。**logarchmeth2** 配置参数归档的日志文件取决于您是否还设置了 **mirrorlogpath** 数据库配置参数的值:

- 如果未设置 **mirrorlogpath** 配置参数的值，那么 **logarchmeth2** 配置参数会对 **logpath** 配置参数指定的当前日志路径中的日志文件进行归档。
- 如果设置了 **mirrorlogpath** 配置参数的值，那么 **logarchmeth2** 配置参数对镜像日志路径中的日志文件进行归档。

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

- 带有本地客户机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
- 带有本地客户机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 在配置为支持代理节点的 TSM 环境中，″-fromnode=nodename″ 选项和 ″-fromowner=ownername″ 选项与 ″-asnodename=nodename″ 选项不兼容，并且不能一起使用。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

- 带有本地客户机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
- 带有本地客户机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 在配置为支持代理节点的 TSM 环境中，″-fromnode=nodename″ 选项和 ″-fromowner=ownername″ 选项与 ″-asnodename=nodename″ 选项不兼容，并且不能一起使用。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
- 可由 DB2 pureScale 环境中的成员配置

缺省值 [范围]

32 位平台

256 [4 - 4 096]

64 位平台

256 [4 - 131 070]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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
- 发生了其他某些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日志缓冲区变满（这要求它先将数据写至磁盘，才能写入新日志记录）的频率。请参阅 **num_log_buffer_full**“日志缓冲区变满次数”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 - 1 048 572]

Windows

1000 [4 - 1 048 572]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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。

- 日志保留或归档日志记录

  如果日志文件已满，那么它会关闭并且不可被覆盖。如果配置了归档日志记录，那么随后该日志文件将归档。

建议: 必须使日志文件的大小与日志文件数平衡:

- 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的上限（**logprimary** + **logsecond**）共同确定活动日志空间的上限 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处理时间而影响系统性能。

- 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。

  而且，日志文件的大小可影响对日志文件进行归档的频率以及对个别日志文件进行归档时所耗用的时间。这些因素将影响灾难恢复方案（如果主服务器发生问题）的归档位置中日志文件的可用性。**ARCHIVE LOG FOR DATABASE** 命令可用于更频繁地截断和归档日志文件（如果需要）。

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，如果了解您的并行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

- 带有本地客户机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
- 带有本地客户机和远程客户机的分区数据库服务器

参数类型

- 可联机配置
- 可由 DB2 pureScale 环境中的成员配置

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

对 **newlogpath** 参数的更改生效后，您不能直接更改此参数，因为它由数据库管理器设置。

成员的缺省日志路径是全局数据库目录内的某个目录。例如，给定实例名称 dbinst、数据库名称 dbname 和用户名 dbuser，成员的缺省日志路径可能为 /home/dbuser/dbinst/NODE0000/SQL00001/LOGSTREAM0000。在 DB2 pureScale 环境中，每个成员都有一个 LOGSTREAM*xxxx* 目录。

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]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

计量单位

计数器

分配时间

- 创建数据库时
- 将日志移至另一位置时（当更新 **newlogpath** 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
- 日志文件变为不活动状态，并且新日志文件在主日志路径中至少需要保留 **logprimary** 个日志（**logarchmeth1** 或 **logarchmeth2** 参数不得设为 OFF 以外的值）。
- 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** <= 256。
- 如果 **logsecond** 的值不为 -1，那么 **logprimary** + **logsecond**) <= 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要求，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所产生额外处理时间。

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要求。

- **sec_log_used_top**（使用的最大辅助日志空间）
- **tot_log_used_top**（使用的最大总日志空间）
- **sec_logs_allocated**（当前分配的辅助日志）

logsecond -“辅助日志文件数”

该参数指定创建并用于恢复日志文件的辅助日志文件的数量。仅在需要时才创建辅助日志文件。

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]
10 [-1; 0 - 254]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

计量单位
计数器

分配时间
logprimary 不足时需要。有关分配的更多详细信息，请参阅随后的详细信息。

释放时间
随着时间推移，数据库管理器确定不再需要哪些辅助日志文件。

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目，那么会将一个错误代码返回至应用程序。

如果将 **logsecond** 设为 -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。

注: 不能在 DB2 pureScale 环境中配置无限活动日志空间。

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** 设为 -1。

通过将 **logsecond** 设为 -1，将对工作单元的大小或并行工作单元的数目没有限制。但是，由于需要从归档检索日志文件，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** 设为 -1，必须将 **logarchmeth1** 配置参数设为除 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。因为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。

启用无限记录功能（将 **logsecond** 设为 -1）之后，数据库管理器不会为需要回滚和写日志记录的事务保留活动日志空间。在回滚处理期间，如果活动日志路径和归档目标都已满（或者，如果归档目标不可访问），那么 **blk_log_dsk_ful**（“日志磁盘满时阻止进行日志记录”数据库配置参数）也应该设为 ENABLED 以避免发生数据库故障。

max_log -“每个事务的最大日志数”

此参数指定是否对一个事务可以消耗的主日志空间的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
- 可由 DB2 pureScale 环境中的成员配置

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。

如果该值设为 0，那么对事务可以使用的主日志总空间的百分比没有限制。

如果应用程序违反 **max_log** 配置，那么会强制该应用程序与数据库断开连接并且将回滚事务。

可以通过将 *DB2_FORCE_APP_ON_MAX_LOG* 注册表变量设为“FALSE”来覆盖此行为。这样会导致违反 **max_log** 配置的事务失败；然而，应用程序仍然可以落实工作单元中先前语句已完成的工作，它也可以回滚已完成的工作以撤销该工作单元。

启用无限制活动日志空间时，此参数以及 **num_log_span** 配置参数很有用。如果打开了无限制日志记录（即，如果将 *logsecond* 设为 -1），那么事务不会受日志文件数上限

(*logprimary* + *logsecond*) 的限制。当达到 *logprimary* 的值时，DB2 将开始归档活动日志，而不是使事务失败。这可能会导致问题，例如，如果应用程序包含长时间运行且尚未落实的事务。如果出现这种情况，那么活动日志空间将继续增长，这有可能导致崩溃恢复性能下降。要防止出现这种情况，可以为 **max_log** 和/或 **num_log_span** 配置参数指定值。

注: 下列 DB2 命令不受 **max_log** 配置参数的限制: ARCHIVE LOG、BACKUP DATABASE、LOAD、REORG、RESTORE DATABASE 和 ROLLFORWARD DATABASE。

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）的并行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，所以允许大量并行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
- 可由 DB2 pureScale 环境中的成员配置

传播类 立即

缺省值 [范围]
Automatic [1 - 60000]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

计量单位
计数器

将 **maxappls** 设为 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 - 每个数据库打开的最大数据库文件数

此参数指定每个应用程序都可以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
- 可由 DB2 pureScale 环境中的成员配置

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，打开和关闭文件的额外处理时间将变得极大，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

- 可由 DB2 pureScale 环境中的成员配置

传播类 立即

缺省值 [范围]

Automatic [1 - 100]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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。

尽管 **locklist** 的值可与 **maxlocks** 参数一起调整，但禁止自调整 **locklist** 参数不会自动禁止自调整 **maxlocks** 参数。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** 配置参数设为 ON）时，才会自动调整此配置参数。

在所有平台上，每个锁定需要 128 或 256 字节的锁定列表，这取决于是否对该对象挂起了其他锁定:

- 对于未挂起其他锁定的对象，挂起一个锁定需要 256 字节。
- 对于存在一个挂起的锁定的对象，记录一个锁定需要 128 个字节。

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。如果您仅有几个并行运行的应用程序，那么可以将下列公式当作第一个公式的替代公式:

```
maxlocks = 2 * 100 / (并行运行的应用程序的
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 * 128)`

其中 4096 是一页中的字节数，100 是 **maxlocks** 允许的最大百分比值，128 是每个锁定的字节数。 如果您知道其中一个应用程序需要 1000 个锁定，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 个锁定。

如果将 **maxlocks** 设置的太低，那么当对其他并行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。在这种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。它可设为“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。以下列表显示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。

- 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，或者已将它设为“No”。如果该语句返回 1.000，那么 **min_dec_div_3** 设为“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，这样有助于减少写入日志记录时数据库管理器所需的处理时间。

要点: 在 V10.1 中不推荐使用此参数，在将来的发行版中可能会将其除去。此参数仍可在 V10.1 之前的发行版中使用。在 V10.1 及更高发行版中，将忽略对此配置参数指定的值。

配置类型

数据库

参数类型

- 可联机配置
- 可由 DB2 pureScale 环境中的成员配置

传播类 立即

缺省值 [范围]

1 [1 - 25]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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。将此参数设置得太高可能会对应用程序响应时间有负面影响。

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: 建议将此参数参数设为缺省值 1。

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处理时间减至最小。

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。在这种情况下，**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”

该参数为镜像日志路径指定最多为 242 字节的字符串。该字符串必须指向标准路径名。

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]

在 DB2 pureScale 环境和 DB2 Enterprise Server Edition 环境中，数据库分区号和日志流标识是自动追加至路径（例如，/home/dbuser/dblogs/NODE0000/LOGSTREAM0000/）的。

如果设置了 **mirrorlogpath** 配置参数的值，那么 DB2 数据库系统将在日志路径和镜像日志路径中创建活动日志文件。所有日志数据将写至这两个路径。镜像日志路径有一组重复的活动日志文件，因此如果存在破坏其中一个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。

在 DB2 pureScale 环境中，连接至数据库或激活数据库的第一个成员会处理对此日志路径参数值的更改。DB2 数据库管理器验证该路径是否存在以及它是否对该路径具有读访问权和写访问权。数据库管理器还会尝试为这些日志文件创建特定于成员的子目录。如果其中任一操作失败，那么 DB2 数据库管理器会拒绝指定路径并使用旧路径来使数据库联机。如果数据库管理器接受指定路径，那么新值将传播至每个成员。如果尝试切换至新路径时成员失败，那么激活该成员或连接至该成员的后续尝试将失败，并生成 SQL5099N。所有成员都必须使用同一日志路径。

如果更改镜像日志路径，那么一些日志文件可能会保留在旧镜像日志路径中。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，如果正对相关数据库运行复制，那么复制可能仍需要日志路径更改之前的日志文件。如果将数据库配置为使用日志归档并且 DB2 数据库系统自动对所有日志文件进行归档或您手动对其归档，那么 DB2 数据库系统可检索这些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** 配置参数将原始设备指定为存储日志文件的位置，那么不允许执行 **mirrorlogpath** 配置参数指示的镜像日志记录。如果 **logpath** 或 **newlogpath** 配置参数指定文件路径作为存储日志文件的位置，那么允许执行镜像日志记录，并且 **mirrorlogpath** 配置参数必须指定文件路径。

如果您还设置了 **logarchmeth2** 配置参数，那么 **mirrorlogpath** 配置参数会影响日志归档行为。如果设置了 **mirrorlogpath** 和 **logarchmeth2** 配置参数的值，那么 **logarchmeth2** 配置参数会对镜像日志路径中的日志文件进行归档，而不对活动日志路径中的日志文件的其他副本进行归档。可使用此日志归档行为来改进前滚恢复期间的弹性，因为镜像日志路径中的可用已归档日志文件可能仍可用于继续数据库恢复操作，即使主日志文件在归档前已损坏。

建议:

- 就像日志文件一样，镜像日志文件应放在 I/O 不高的物理磁盘上。
- 强烈建议将镜像日志路径和主日志路径放在不同设备上。

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。

log_reads
读取的日志页数。

log_writes
写入的日志页数。

可使用操作系统监视工具来收集有关其他磁盘 I/O 活动的信息，然后比较两种类型的 I/O 活动。

mon_act_metrics -“监视活动度量值”配置参数

此参数控制对整个数据库收集活动度量值的方式，并且将影响与任何 DB2 工作负载定义相关联的连接所提交的活动。

配置类型
数据库

参数类型

- 可联机配置

缺省值 [范围]
BASE [NONE, BASE, EXTENDED]

升级注意事项
在 V9.7 之前创建并随后升级到 V9.7 的数据库上，缺省情况下将 **mon_act_metrics** 参数设为 NONE。

如果将此配置参数设为 BASE，那么将对数据服务器上执行的所有活动收集所有通过下列接口报告的所有度量值，而不考虑与提交该活动的连接相关联的 DB2 工作负载:

- MON_GET_PKG_CACHE_STMT_DETAILS
- MON_GET_ACTIVITY_DETAILS
- MON_GET_PKG_CACHE_STMT
- 程序包高速缓存事件监视器
- 活动事件监视器

如果将此配置参数设为 EXTENDED，那么收集的度量值与 BASE 设置下的相同。此外，会以更大的详细程度确定为以下监视元素报告的值:

- **total_section_time**

- **total_section_proc_time**
- **total_routine_user_code_time**
- **total_routine_user_code_proc_time**
- **total_routine_time**

有关 EXTENDED 设置如何影响这些监视元素的信息，请参阅详细的监视元素描述。

如果将此配置参数设为 NONE，那么仅对于 COLLECT ACTIVITY METRICS 子句设为 BASE 的 DB2 工作负载的相关联连接所提交的活动子集，才会收集通过上面列示的接口报告的度量值。

mon_deadlock -“监视死锁”配置参数

此参数为锁定事件监视器控制数据库级别的死锁事件生成。

配置类型

数据库

参数类型

- 可联机配置

缺省值 [范围]

WITHOUT_HIST [NONE, WITHOUT_HIST, HISTORY, HIST_AND_VALUES]

如果将此参数设为 NONE，那么不会生成死锁事件，除非使用 COLLECT DEADLOCK DATA 子句对 DB2 工作负载对象启用死锁事件收集。

如果将此参数设为 WITHOUT_HIST，那么死锁事件发生时，有关死锁事件的数据会发送至任何处于活动状态的锁定事件监视器。为死锁等待事件收集的数据将不包括等待死锁的应用程序执行的活动的历史记录。

如果将此参数设为 HISTORY，那么锁定等待事件将包括等待锁定的应用程序执行的活动的历史记录。该历史记录仅包括该应用程序的当前事务中执行的活动，并且仅在达到最大大小时才报告所执行的最新活动。缺省历史记录大小为 250。可使用 **DB2_MAX_INACT_STMTS** 注册表变量来配置历史记录大小。

如果将此参数值设为 HIST_AND_VALUES，那么随锁定等待事件收集的活动历史记录将包括具有这些值的活动的输入数据值。这些数据值不包括 LOB 数据、更改开始 LONG VARCHAR 数据、LONG VARGRAPHIC 数据、更改结束结构化类型数据或 XML 数据。

此参数为锁定事件监视器控制数据库级别的死锁事件收集。**mon_deadlock** 参数确定发生死锁时锁定事件监视器是否收集死锁等待事件。

mon_deadlock 参数值表示对所有 DB2 应用程序启用的最低级别的收集。如果个别 DB2 工作负载指定高于配置参数的收集级别，那么会使用 DB2 工作负载设置而不是配置参数值。请注意，系统应用程序不会在工作负载中运行，因此 **mon_deadlock** 参数是系统应用程序收集死锁数据的唯一方法。

由于锁定等待线程或锁定持有者可能跨工作负载，因此，要通过锁定事件监视器来捕获死锁，请在数据库级别启用死锁收集功能。您可以在工作负载级别单独地控制死锁数据收集级别，也可以通过此参数在数据库级别设置收集级别。

mon_locktimeout -“监视锁定超时”配置参数

此参数为锁定事件监视器控制数据库级别的锁定超时事件生成，并且将影响所有 DB2 工作负载定义。

配置类型

数据库

参数类型

- 可联机配置

缺省值 [对数据库的所有工作负载或服务类启用的最低收集级别]

NONE [NONE, WITHOUT_HIST, HISTORY, HIST_AND_VALUES]

如果将此参数设为 NONE，那么不会生成任何锁定超时事件，除非使用 COLLECT LOCK TIMEOUT DATA 子句对 DB2 工作负载对象启用了锁定超时事件收集。

如果将此参数设为 WITHOUT_HIST，那么锁定事件发生时，有关锁定事件的数据会发送至任何处于活动状态的锁定事件监视器。为锁定超时事件收集的数据将不包括等待锁定的应用程序执行的活动的历史记录。

如果将此参数设为 HISTORY，那么锁定超时事件将包括等待锁定的应用程序执行的活动的历史记录。该历史记录仅包括该应用程序的当前事务中执行的活动，并且仅在达到最大大小时才报告所执行的最新活动。缺省历史记录大小为 250。可使用 **DB2_MAX_INACT_STMTS** 注册表变量来配置历史记录大小。

如果将此参数值设为 HIST_AND_VALUES，那么随锁定超时事件收集的活动历史记录将包括具有这些值的活动的输入数据值。这些数据值不包括 LOB 数据、更改开始 LONG VARCHAR 数据、LONG VARGRAPHIC 数据、更改结束结构化类型数据或 XML 数据。

此参数为锁定事件监视器控制数据库级别的锁定超时事件收集。**mon_locktimeout** 参数确定在应用程序等待锁定期间超时的情况下锁定事件监视器是否收集锁定超时事件。

mon_locktimeout 参数值表示对所有 DB2 应用程序启用的最低收集级别。可通过对 DB2 工作负载对象使用 COLLECT LOCK TIMEOUT DATA 子句而不是 **mon_locktimeout** 参数来对一部分 DB2 应用程序启用锁定超时事件收集。如果个别 DB2 工作负载指定高于配置参数的收集级别，那么会使用 DB2 工作负载设置而不是配置参数值。请注意，系统应用程序不会在工作负载中运行，因此 **mon_locktimeout** 参数是系统应用程序收集锁定超时数据的唯一方法。

mon_lockwait -“监视锁定等待”配置参数

此参数为锁定事件监视器控制数据库级别的锁定等待事件生成。

配置类型

数据库

参数类型

- 可联机配置

缺省值 [范围]

NONE [NONE, WITHOUT_HIST, HISTORY, HIST_AND_VALUES]

如果将此参数设为 NONE，那么不会生成任何锁定等待事件，除非使用 COLLECT LOCK WAIT DATA 子句对 DB2 工作负载对象启用了锁定等待事件收集。

如果将此参数设为 WITHOUT_HIST，那么锁定事件发生时，有关锁定事件的数据会发送至任何处于活动状态的锁定事件监视器。为锁定等待事件收集的数据将不包括等待锁定的应用程序执行的活动的历史记录。

如果将此参数设为 HISTORY，那么锁定等待事件将包括等待锁定的应用程序执行的活动的历史记录。该历史记录仅包括该应用程序的当前事务中执行的活动，并且仅在达到最大大小时才报告所执行的最新活动。缺省历史记录大小为 250。可使用 **DB2_MAX_INACT_STMTS** 注册表变量来配置历史记录大小。

如果将此参数值设为 HIST_AND_VALUES，那么随锁定等待事件收集的活动历史记录将包括具有这些值的活动的输入数据值。这些数据值不包括 LOB 数据、更改开始 LONG VARCHAR 数据、LONG VARGRAPHIC 数据、更改结束结构化类型数据或 XML 数据。

mon_lockwait 参数值表示对所有 DB2 应用程序启用的最低收集级别。可通过对 DB2 工作负载对象使用 COLLECT LOCK WAIT DATA 子句而不是 **mon_lockwait** 参数来对一部分 DB2 应用程序启用锁定等待事件收集。如果个别 DB2 工作负载指定高于配置参数的收集级别，那么会使用 DB2 工作负载设置而不是配置参数值。请注意，系统应用程序不会在工作负载中运行，因此 **mon_lockwait** 参数是系统应用程序收集锁定等待数据的唯一方法。

此参数为锁定事件监视器控制数据库级别的锁定等待事件收集。**mon_lockwait** 配置参数将与 **mon_lw_thresh** 配置参数配合使用。**mon_lockwait** 参数确定应用程序等待锁定的时间超过 **mon_lw_thresh** 微秒时锁定等待事件监视器是否收集锁定等待事件。

mon_lw_thresh -“监视锁定等待阈值”配置参数

此参数控制在生成 **mon_lockwait** 的事件之前等待锁定时耗用的时间。

配置类型

数据库

参数类型

- 可联机配置

缺省值 [范围]

5000000 [1000 ... MAX_INT]

升级注意事项

在 V9.7（然后升级到 V9.7 或更高版本）之前创建的数据库上，缺省情况下将 **mon_lw_thresh** 参数设为 4294967295。

计量单位

微秒

如果在数据库级别和工作负载级别都设置了此参数，那么给定工作负载将采用所配置的这两个时间中的较短者。

mon_lck_msg_lvl -“监视锁定事件通知消息”配置参数

此参数用于控制在发生锁定超时、死锁和锁定升级事件时将消息记录到管理通知日志中。

配置类型

数据库

参数类型

可联机配置

缺省值 [范围]

1 [0 - 3]

在发生锁定超时、死锁和锁定升级事件的情况下，可以通过将此数据库配置参数设为一个适合于您需要的通知级别的值，来将消息记录到管理通知日志中。以下列表概述了可以设置的通知级别:

0 级别 0: 不提供关于锁定升级、死锁和锁定超时的通知

1 级别 1: 关于锁定升级的通知

2 级别 2: 关于锁定升级和死锁的通知

3 级别 3: 关于锁定升级、死锁和锁定超时的通知

此数据库配置参数的缺省通知级别设为 1。

mon_obj_metrics -“监视对象度量值”配置参数

此参数控制对整个数据库收集数据对象度量值的方式。

配置类型

数据库

参数类型

- 可联机配置

缺省值 [范围]

BASE [NONE, BASE, EXTENDED]

升级注意事项

在 V9.7 之前创建并随后升级到 V9.7 或更高版本的数据库上，缺省情况下将 **mon_obj_metrics** 参数设为 NONE。

将此配置参数设为 BASE 以收集通过下列表函数报告的所有度量值:

- MON_GET_BUFFERPOOL
- MON_GET_CONTAINER
- MON_GET_TABLESPACE

将此配置参数设为 EXTENDED 以收集通过下列表函数报告的额外度量值:

表 137. 使用 EXTENDED 选项返回的度量值

表函数	报告的度量值
MON_GET_INDEX	object_index_gbp_indep_pages_found_in_lbp object_index_gbp_invalid_pages object_index_gbp_l_reads object_index_gbp_p_reads object_index_l_reads object_index_lbp_pages_found object_index_p_reads
MON_GET_INDEX_USAGE_LIST	object_index_gbp_indep_pages_found_in_lbp object_index_gbp_invalid_pages object_index_gbp_l_reads object_index_gbp_p_reads object_index_l_reads object_index_lbp_pages_found object_index_p_reads
MON_GET_TABLE	direct_read_reqs direct_reads direct_write_reqs direct_writes lock_escals lock_escals_global lock_wait_time lock_wait_time_global lock_waits lock_waits_global object_data_gbp_indep_pages_found_in_lbp object_data_gbp_invalid_pages object_data_gbp_l_reads object_data_gbp_p_reads object_data_l_reads object_data_lbp_pages_found object_data_p_reads object_xda_gbp_indep_pages_found_in_lbp object_xda_gbp_invalid_pages object_xda_gbp_l_reads object_xda_gbp_p_reads object_xda_l_reads object_xda_lbp_pages_found object_xda_p_reads

表 137. 使用 EXTENDED 选项返回的度量值 (续)

表函数	报告的度量值
MON_GET_TABLE_USAGE_LIST	direct_read_reqs direct_reads direct_write_reqs direct_writes lock_escals lock_escals_global lock_wait_time lock_wait_time_global lock_waits lock_waits_global object_data_gbp_indep_pages_found_in_lbp object_data_gbp_invalid_pages object_data_gbp_l_reads object_data_gbp_p_reads object_data_l_reads object_data_lbp_pages_found object_data_p_reads object_xda_gbp_indep_pages_found_in_lbp object_xda_gbp_invalid_pages object_xda_gbp_l_reads object_xda_gbp_p_reads object_xda_l_reads object_xda_lbp_pages_found object_xda_p_reads overflow_accesses overflow_creates rows_deleted rows_inserted rows_read rows_updated

如果将此配置参数设为 NONE，那么不会收集通过上述表函数报告的度量值。

mon_pkglist_sz -“监视程序包列表大小”配置参数

此参数控制由工作单元事件监视器捕获的每个工作单元在程序包列表中可以显示的最大条目数。

配置类型

数据库

参数类型

可联机配置

传播子句

下一个工作单元

缺省值 [范围]

32 [0 - 1024]

计量单位

程序包列表中的条目数

程序包列表将具有由此数据库配置参数值指定的最大大小。程序包列表的大小是工作单元启动时确定的。进入下一个工作单元时才会反映对程序包列表大小所作的更改。程序包列表的缺省大小为包含 32 个条目。

mon_req_metrics -"监视请求度量值"配置参数

此参数控制对整个数据库收集请求度量值的方式，并且将影响任何 DB2 服务类中执行的请求。

配置类型

数据库

参数类型

- 可联机配置

缺省值 [范围]

BASE [NONE, BASE, EXTENDED]

升级注意事项

在 V9.7 之前创建并随后升级到 V9.7 或更高版本的数据库上，缺省情况下将 **mon_req_metrics** 参数设为 NONE。

如果将此配置参数设为 BASE，那么将对数据服务器上执行的所有请求收集所有通过下列接口报告的所有度量值，而不考虑在哪个 DB2 服务类中运行该请求:

- MON_GET_UNIT_OF_WORK
- MON_GET_UNIT_OF_WORK_DETAILS
- MON_GET_CONNECTION
- MON_GET_CONNECTION_DETAILS
- MON_GET_SERVICE_SUBCLASS
- MON_GET_SERVICE_SUBCLASS_DETAILS
- MON_GET_WORKLOAD
- MON_GET_WORKLOAD_DETAILS
- 统计信息事件监视器（event_wlstats 和 event_scstats 逻辑数据组中的 DETAILS_XML 监视元素）
- 工作单元事件监视器

如果将此配置参数设为 EXTENDED，那么收集的度量值与 BASE 设置下的相同。此外，会以更大的详细程度确定为以下监视元素报告的值:

- **total_section_time**
- **total_section_proc_time**
- **total_routine_user_code_time**
- **total_routine_user_code_proc_time**
- **total_routine_time**

有关 EXTENDED 设置如何影响这些监视元素的信息，请参阅详细的监视元素描述。

如果将此配置参数设为 NONE，那么将仅对以下 DB2 服务类中运行的请求子集收集通过上面列示的接口报告的度量值（该服务类的服务超类已将 COLLECT REQUEST METRICS 子句设为 BASE）。

mon_uow_data -“监视工作单元事件”配置参数

此参数控制在数据库级别为工作单元事件监视器生成工作单元事件的方式，并且将影响数据服务器上的工作单元。它是 **mon_uow_execlist** 和 **mon_uow_pkglist** 配置参数的父参数。

配置类型

数据库

适用于

- 带有本地客户机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
- 带有本地客户机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 下一个工作单元

缺省值 [范围]

NONE [NONE, BASE]

此参数的有效值如下所示:

NONE 禁用 UOW 事件监视器的所有数据收集。

BASE 启用 UOW 事件监视器的基本数据收集。

此参数指定工作单元完成时是否应将该工作单元（又称为事务）的信息发送至处于活动状态的工作单元事件监视器。

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，可全局启用或禁用数据收集。

如果此参数设为 BASE，那么在数据服务器上执行的所有工作单元完成时，系统会将有关工作单元的信息发送至处于活动状态的工作单元事件监视器。如果此参数设为 NONE，那么只会将在 COLLECT UNIT OF WORK DATA 子句设为 BASE 的 DB2 工作负载下执行的工作单元的信息发送至工作单元事件监视器。缺省设为 NONE。

在工作单元结束时收集的信息包括该工作单元的系统级别请求度量值，例如，执行该工作单元期间使用的 CPU 量。这些请求度量值的收集与工作单元数据的收集分开控制。通过对 DB2 服务超类使用 COLLECT REQUEST METRICS 子句或将 **mon_req_metrics** 数据库配置参数设为值 BASE 来收集工作单元数据。如果未启用请求度量值收集，那么作为工作单元数据的一部分收集的所有请求度量值均为零。

mon_uow_execlist -“监视带有可执行列表的工作单元事件”配置参数

此参数控制如何生成包括可执行标识列表信息的工作单元事件。对于工作单元事件监视器，这是在数据库级别完成的。**mon_uow_execlist** 数据库配置参数是 **mon_uow_data** 数据库配置参数的子参数。

配置类型
数据库

适用于

- 带有本地客户机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
- 带有本地客户机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 下一个工作单元

缺省值 [范围]
OFF [OFF, ON]

此参数的有效值如下所示:

OFF 对工作单元事件监视器禁用可执行标识列表收集。

ON 对工作单元事件监视器启用可执行标识列表收集。

mon_uow_pkglist -“监视带有包列表的工作单元事件”配置参数

此参数控制包括包列表信息的工作单元事件生成。这是在工作单元事件监视器的数据库级别完成的。**mon_uow_pkglist** 数据库配置参数是 **mon_uow_data** 数据库配置参数的子参数。

配置类型
数据库

适用于

- 带有本地客户机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
- 带有本地客户机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 下一个工作单元

缺省值 [范围]
OFF [OFF, ON]

此参数的有效值如下所示:

OFF 对工作单元事件监视器禁用包列表收集。

ON 对工作单元事件监视器启用包列表收集。

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。

要点: 从 V9.1 开始，建议不要使用原始设备来进行数据库日志记录，此操作在 DB2 pureScale 环境中不受支持。（如果受支持）数据库管理器对活动日志文件使用自动非缓冲 I/O，并且所有其他日志文件访问将被缓冲。

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 DB2 pureScale 环境和 DB2 Enterprise Server Edition 环境中，数据库分区号和日志流标识是自动追加至路径（例如，`/home/dbuser/dblogs/NODE0000/LOGSTREAM0000/`）的。

在 DB2 pureScale 环境中，连接至数据库或激活数据库的第一个成员会处理对此日志路径参数的配置更改。DB2 数据库管理器验证该路径是否存在以及它是否对该路径具有读访问权和写访问权。它还会为日志文件创建特定于成员的子目录。如果其中任一操作失败，那么 DB2 数据库管理器会拒绝指定路径并使用旧路径来使数据库联机。如果接受指定路径，那么会向每个成员传播新值。如果尝试切换至新路径时成员失败，那么激活该成员或连接至该成员的后续尝试将失败 (SQL5099N)。所有成员都必须使用同一日志路径。

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。

在下列两个事件发生之前，新设置不会成为 **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。如果数据库配置为使用日志归档并且所有日志文件已由 DB2 数据库系统自动归档或由您手动归档，那么 DB2 数据库系统将能够检索日志文件来完成复制过程。否则，可以将这些文件从旧日志路径复制到新日志路径。

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减少花费的额外处理时间（例如，等待 I/O 时）。

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其相应的日志副本。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改到期的条目。

应将 **rec_his_retentn** 配置参数设为与 **num_db_backups** 的值兼容的值。例如，如果 **num_db_backups** 设为一个较大的值，那么 **rec_his_retentn** 的值应当足够大，以支持设为 **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** 命令中指定 **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 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** 命令参数在表或列级别指定作为 **RUNSTATS** 命令的一部分收集的高频值的数目。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算。选择性计算越精确，选择的存取方案就越有效。

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为 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
- 可由 DB2 pureScale 环境中的成员配置

缺省值 [范围]

Automatic [0 - 255]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

计量单位

计数器

释放引擎数等于为 **num_iocleaners** 配置参数设置的值。但是，最大释放引擎数为 128。

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。

页清除程序还将减少从软故障（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。

如果将此参数设为 AUTOMATIC，那么启动的页清除程序数取决于当前机器上配置的物理 CPU 核心数以及分区数据库环境中的本地逻辑数据库分区数。当此参数设为 AUTOMATIC 时，始终至少启动一个页清除程序。

当此参数设为 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物理 CPU 核心数。

注: 在 HP-UX 平台上，我们将在计算中使用逻辑 CPU 数而不是物理 CPU 核心数。

建议: 当设置此参数的值时，请考虑下列因素:

- 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。这仅在 **DB2_USE_ALTERNATE_PAGE_CLEANING** 设为 OFF（这也是缺省值）时适用。
- 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。这仅在 **DB2_USE_ALTERNATE_PAGE_CLEANING** 设为 OFF（这也是缺省值） 时适用。

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**。

    注: 建议不要降低至通过 AUTOMATIC 计算出的较低值。
- 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
- 可由 DB2 pureScale 环境中的成员配置

缺省值 [范围]

Automatic [1 - 255]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

计量单位

计数器

分配时间

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 parallelism settings 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 服务器都有关联的一些额外处理时间，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
- 可由 DB2 pureScale 环境中的成员配置

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。

如果该值设为 0，那么对单个事务可以跨越的日志文件数没有限制。

如果应用程序违反 **num_log_span** 配置，那么会强制该应用程序与数据库断开连接并且将回滚事务。

启用无限制活动日志空间时，此参数以及 **max_log** 配置参数很有用。如果打开了无限制日志记录（即，如果将 *logsecond* 设为 -1），那么事务不会受日志文件数上限 (*logprimary* + *logsecond*) 的限制。当达到 *logprimary* 的值时，DB2 将开始归档活动日志，而不是使事务失败。这可能会导致问题，例如，如果应用程序包含长时间运行且尚未落实的事务。如果出现这种情况，那么活动日志空间将继续增长，这有可能导致崩溃恢复性能下降。要防止出现这种情况，可以为 **max_log** 和/或 **num_log_span** 配置参数指定值。

注: 下列 DB2 命令不受 **num_log_span** 配置参数的限制: ARCHIVE LOG、BACKUP DATABASE、LOAD、REORG、RESTORE DATABASE 和 ROLLFORWARD DATABASE。

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
- 可由 DB2 pureScale 环境中的成员配置

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。如果指定 0 或 1，那么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** 命令参数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算。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：在大多数情况下，此参数的缺省值提供合理的准确估算。如果观察到下列两者之间存在重大和一致的差别，那么可考虑增加此值:

- 从说明输出的选择性估算；和
- 非统一分布式列数据的范围谓词的实际选择性。

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 数据库系统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

- 带有本地客户机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
- 带有本地客户机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 数据库系统将继续重试归档至主日志路径或辅助日志路径。

numsegs -“缺省 SMS 容器数”

从 V9.5 起建议不要使用此参数，但是 V9.5 之前的数据服务器和客户机仍然可以使用此参数。DB2 V9.5 或更高发行版中的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 V9.5 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”

overflowlogpath 参数指定一个位置，DB2 数据库将在该位置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。

- 使用 **overflowlogpath** 参数指定一个位置，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** 命令中都指定 **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** 设为 **-1**，那么可使用 **overflowlogpath** 参数对 DB2 指定用于存储从归档检索的活动日志文件的目录。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。如果未设置 **overflowlogpath** 参数，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** 参数为 DB2 数据库提供其他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。
- 如果需要使用 db2ReadLog API 进行复制，例如，使用 **overflowlogpath** 指定 DB2 数据库搜索此 API 所需日志文件的位置。在 DB2 V8 之前，db2ReadLog 称为 sqlurlog。 如果（在活动日志路径或溢出日志路径中）找不到日志文件并且通过使用 **logarchmeth1** 或 **logarchmeth2** 参数配置数据库以归档日志，那么 DB2 会检索该日志文件。还应使用 **overflowlogpath** 参数以对 DB2 数据库指定用于存储所检索日志文件的目录。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。
- 如果已为活动日志路径配置原始设备，并且要将 **logsecond** 参数设为 **-1** 或要使用 db2ReadLog API，那么必须配置 **overflowlogpath** 参数。
- 您可以为 DB2 数据库指定一个位置，以检索 **BACKUP DATABASE INCLUDE LOGS** 操作所需要的日志文件。

要设置 **overflowlogpath**，指定一个最长 242 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 DB2 pureScale 环境和 DB2 Enterprise Server Edition 环境中，数据库分区号和日志流标识是自动追加至路径（例如，`/home/dbuser/dblogs/NODE0000/LOGSTREAM0000/`）的。

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
- 可由 DB2 pureScale 环境中的成员配置

传播类 立即

缺省值 [范围]

32 位操作系统

Automatic [-1, 32 - 128 000]

64 位操作系统

Automatic [-1, 32 - 2 147 483 646]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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，从而减少其内部处理时间。将这些段保存在程序包高速缓存中，直到发生下列其中一个事件:

- 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。

当此参数设为 AUTOMATIC 时，就启用了自调整功能。当 **self_tuning_mem** 设为 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。

仅当对数据库启用了自调整内存功能（**self_tuning_mem** 配置参数设为 ON）时，才会自动调整此配置参数。

当此参数设为 -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，如果有必要的话，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”

从 V9.5 起建议不要使用此参数，但是 V9.5 之前的数据服务器和客户机仍然可以使用此参数。DB2 V9.5 或更高发行版中的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 V9.5 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

- 带有本地客户机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
- 带有本地客户机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]

20 000 [-1; 32 - 112 000]

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占用的处理时间。

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为一个较小的数。

如果 **rec_his_retentn** 设为 -1 并且 **auto_del_rec_obj** 设为 OFF，那么用于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 显式修改恢复历史记录文件中的其他条目。如果 **rec_his_retentn** 设为 -1 并且 **auto_del_rec_obj** 设为 ON，那么不会自动修改历史记录文件，也不会删除恢复对象。

如果 **rec_his_retentn** 设为 0 并且 **auto_del_rec_obj** 设为 OFF，那么将修改历史记录文件中的所有条目（最后一次完全备份除外）。如果 **auto_del_rec_obj** 设为 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改历史记录文件和删除恢复对象。

不管保留期多短，将始终保留最近的完整数据库备份及其复原集，除非您使用带有 **FORCE** 选项的 **PRUNE HISTORY** 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，未在 **CREATE DATABASE** 命令中使用 **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**）。

section_actuals -“部分实际值”配置参数

此参数允许收集部分实际值（部分执行期间度量的运行时统计信息），以便以后创建事件监视器时可查看该统计信息。

配置类型
数据库

适用于

- 带有本地客户机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
- 带有本地客户机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 工作单元边界

缺省值 [范围]
NONE [NONE, BASE]

以下值对于该参数是有效的:

NONE 指定禁用部分实际值收集。

BASE 指定启用该度量并且收集以下统计信息:

- 基本运算符基数计数
- 部分执行期间引用的每个对象的统计信息（仅 DML 语句）

启用部分实际值后，通过使用活动事件监视器捕获部分实际值，并通过使用 EXPLAIN_FROM_ACTIVITY 过程执行的部分说明进行查看。

如果启用 **auto_stats_prof** 数据库配置参数，那么不能启用此参数 (SQLCODE -5153)。

self_tuning_mem -“自调整内存功能”

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
- 可由 DB2 pureScale 环境中的成员配置

传播类 立即

缺省值 [范围]

单个数据库分区环境
ON [ON; OFF]

多数据库分区环境
OFF [ON; OFF]

在从先前版本升级的数据库中，**self_tuning_mem** 将保留旧值。

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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。如果 **self_tuning_mem** 设为 ON，但启用了自调整的内存使用者不足两个，那么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。）

在单数据库分区环境中，缺省情况下，此参数为 ON。在多数据库分区环境中，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:

- 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）
- 缓冲池（由 ALTER BUFFERPOOL 和 CREATE BUFFERPOOL 语句的大小参数控制）

  注: 在 DB2 pureScale 环境中，只有本地缓冲池 (LBP) 可使用自调整内存功能进行管理。组缓冲池 (GBP) 的内存通过使用 **cf_gbp_sz** 配置参数进行分配和控制。

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) - 尽管参数设为 ON，但因为启用自调整的内存使用者不足两个或者该数据库或实例处于停顿方式，所以未执行自调整。
- 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。

如果您已从先前版本的 DB2 升级到 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 -“顺序检测和提前读标志”

此参数控制是否允许数据库管理器在 I/O 活动期间执行顺序检测或提前读预取。

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 和 No]

数据库管理器可在索引扫描期间监视 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*。但是，在运行时，检测到顺序检测预取的工作表现不够好时，预取类型可能从顺序检测预取切换至提前读预取。

如果此参数设为 No，那么系统会在查询执行期间对带有 **INDEXDATA** 选项的 **INSPECT** 命令、**RUNSTATS** 命令和所有索引扫描禁用顺序检测预取和提前读预取。但是，对于针对索引的带有 **CLEANUP** 选项的 **REORG** 命令，将 **seqdetect** 设为 No 仅对索引禁用顺序检测预取。

建议：在大多数情况下，应使用此参数的缺省值。仅当其他调整无法更正严重查询性能问题时，才应尝试禁用 **seqdetect**。

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
- 可由 DB2 pureScale 环境中的成员配置

传播类 立即

缺省值 [范围]

32 位平台

```
Automatic [250 - 524 288]
```

64 位平台

```
Automatic [250 - 2 147 483 647]
```

注：可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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**、缓冲池（每个缓冲池计为一个）、**locklist** 和 **database_memory**。

仅当数据库管理器配置参数 **sheapthres** 设为 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:

- 装入到 XML 表中需要共享排序内存。在这种情况下，我们需要 **sheapthres_shr** 为非零值，否则将返回错误，指示无法为此实用程序分配共享排序内存。
- 当 **intra_parallel** 数据库管理器配置参数设为 yes 时，这是因为当 **intra_parallel** 设为 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。

smtp_server -“SMTP 服务器”

此参数标识一个简单电子邮件传输协议 (SMTP) 服务器。此 SMTP 服务器将传输 UTL_MAIL 内置模块所发送的电子邮件。

此参数还接受逗号定界的 SMTP 服务器列表。UTL_MAIL 尝试通过该列表中的第一个 SMTP 服务器发送电子邮件。如果该 SMTP 服务器不可用，那么将使用该列表中的第二个服务器。如果逗号定界列表中的所有服务器都不可用，那么将返回错误。

配置类型
数据库

适用于 带有本地客户机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
可联机配置

传播类 立即

缺省值 [范围]
Null [逗号定界的有效 SMTP 服务器 TCP/IP 主机名列表]

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]

DB2 pureScale 环境

100 [1 - 65 535]

在 DB2 pureScale 环境外部

100 [1 - 100 * **logprimary**]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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文件数。例如，如果使用缺省值 100，那么数据库管理器将尝试把需要恢复的日志文件数保持为 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文件数保持为 3。

  要影响进行崩溃恢复所需要的日志文件数，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。这是将信息写至日志控制文件的过程。此信息用于确定日志中的起点以防需要重新启动数据库。

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额外的处理时间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控制软检查点的频率。

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处理时间，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”

此参数定义需要排序堆内存的操作将使用的最大专用或共享内存页数。

配置类型

数据库

参数类型

- 可联机配置
- 可由 DB2 pureScale 环境中的成员配置

传播类 立即

缺省值 [范围]

32 位平台

```
Automatic [16 - 524288]
```

64 位平台

```
Automatic [16 - 4194303]
```

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

计量单位

页 (4 KB)

分配时间

需要执行那些需要排序内存的操作时

释放时间

需要排序内存的操作完成时

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。

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** 参数未设为 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** 设为 0 时，才允许自动调整 **sortheap**。

仅当对数据库启用了自调整内存功能（**self_tuning_mem** 配置参数设为 ON）时，才会自动调整此配置参数。

建议：当使用排序堆时，应该考虑下列因素：

- 适当的索引可使排序堆的使用减至最小程度。
- 以下各项使用排序堆内存。使用下列任何技巧时增加此参数的大小：
  - 散列连接缓冲区
  - 块索引 AND 运算
  - 合并连接
  - 内存中的表
  - 动态位图（用于索引 AND 运算和星型连接）
  - 表队列（有多个数据库代理程序在处理一个查询时）
  - 部分提前相异操作
  - 部分提前聚集操作
- 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，应考虑重新绑定应用程序（使用 **REBIND** 命令）。

更新 **sortheap** 值时，数据库管理器将立即开始将这个新值用于任何当前排序或新排序。

sql_ccflags - 条件编译标志

此参数包含对所选 SQL 语句进行条件编译时使用的条件编译值的列表。

配置类型

数据库

参数类型

可联机配置

sql_ccflags 的值必须包含一个或多个名称/值对，而名称与值之间要用冒号字符分隔。每个名称/值对与前一个名称/值对之间必须用逗号分隔。名称必须是有效的普通 SQL 标识，值必须是 SQL BOOLEAN 常量、SQL INTEGER 常量或 NULL 关键字。字符串的最大长度为 1023 个字节。

更新 **sql_ccflags** 的值时，并不会立即检查该值以确保它有效。要在首次使用该值来初始化 CURRENT SQL_CCFLAGS 专用寄存器时才会检查该值；也就是说，与数据库之间的连接首次引用 CURRENT SQL_CCFLAGS 时或者在 SQL 语句中遇到查询伪指令时才会检查该值。更新 **sql_ccflags** 的值之后，请使用以下语句来连接至数据库和查询专用寄存器：VALUES CURRENT SQL_CCFLAGS。

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。

此参数设置的约束将应用于每个 **RUNSTATS** 操作。

对于 V9.5，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
- 可由 DB2 pureScale 环境中的成员配置

缺省值 [范围]

Automatic [1 096 - 524 288]

注：可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

计量单位

页 (4 KB)

分配时间

当启动 **RUNSTATS** 实用程序时

释放时间

当完成 **RUNSTATS** 实用程序时

建议：**RUNSTATS** 内存需求取决于多种因素。更多统计信息选项需要更多内存，例如，是否正在收集 LIKE 统计信息或 DETAILED 索引统计信息。收集了列统计信息之后，收集更多列数的统计信息将需要更多内存。收集分布统计信息时，收集更多高频值和/或分位数值将需要更多内存。建议使用缺省设置 AUTOMATIC。

stmt_conc -“语句集中器”配置参数

此配置参数用于设置缺省的语句集中器行为。

配置类型

数据库

参数类型

- 可联机配置
- 可由 DB2 pureScale 环境中的成员配置

传播类 语句边界

缺省值 [范围]

OFF [OFF, LITERALS]

此配置参数用于对动态语句启用语句集中功能。仅当客户机未显式启用或禁用语句集中器时，才会使用数据库配置中的设置。

启用语句集中器之后，该集中器将修改动态语句，以允许增加程序包高速缓存条目的共享。

如果此配置参数设为 OFF，那么会禁用语句集中器。如果此配置参数设为 LITERALS，那么会启用语句集中器。如果启用了语句集中器，那么除字面值以外其他部分完全相同的 SQL 语句可共享程序包高速缓存条目。

例如，如果 **stmt_conc** 设为 LITERALS，那么以下语句共享程序包高速缓存中的条目

```
SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO='000020' SELECT FIRSTNME,
LASTNAME FROM EMPLOYEE WHERE EMPNO='000070'
```

程序包高速缓存中的条目使用以下语句:

```
SELECT FIRSTNME, LASTNAME FROM EMPLOYEE WHERE EMPNO=:L0
```

DB2 数据库系统根据原始语句中使用的字面值来提供 :L0 的值:

```
:L0(either '000020' or '000070')
```

此参数可能会对存取方案的选择产生重大影响，因为它会改变语句文本。仅当程序包高速缓存中的类似语句具有类似的方案时，才必须使用语句集中器。例如，如果一个语句中的不同字面值导致方案不同，那么不得将语句集中器设为 LITERALS。

stmt_conc 配置参数可能导致 VARCHAR 和 VARGRAPHIC 字符串字面值的长度属性大于该字符串字面值的长度。

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

可由 DB2 pureScale 环境中的成员配置

传播类 语句边界

缺省值 [范围]

对于 32 位平台

AUTOMATIC [128 - 524288]

- 带有本地客户机和远程客户机的数据库服务器: 缺省值为 AUTOMATIC（底层值为 2048）。
- 还可将该参数仅设为某个固定值（不带 AUTOMATIC 属性）。

对于 64 位平台

AUTOMATIC [128 - 524288]

- 带有本地客户机和远程客户机的数据库服务器: 缺省值为 AUTOMATIC（底层值为 8192）。

- 还可将该参数仅设为某个固定值（不带 AUTOMATIC 属性）。

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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** 参数设为 AUTOMATIC。该参数设为 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 错误。

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。

如果查询的运行时性能不足，请考虑增加 **stmtheap** 配置参数值（基于 AUTOMATIC 的值或固定值），从而确保动态编程连接枚举能够成功。如果更新 **stmtheap** 配置参数以改进查询性能，请重新编译该语句以便查询优化器可创建新存取方案来利用语句堆的已更改量。

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。

suspend_io -“数据库 I/O 操作状态”配置参数

此参数显示数据库的 I/O 写操作是已暂挂还是正在暂挂。

配置类型

数据库

参数类型

参考

值包括 YES、IN_PROGRESS 和 NO。

systime_period_adj -“调整临时 SYSTEM_TIME 时间段”数据库配置参数

此数据库配置参数指定生成系统时间段临时表的历史记录行（其结束时间戳记小于开始时间戳记）时要执行的操作。

如果两个不同事务尝试更新系统时间段临时表中的同一行时发生冲突，那么会发生此情况。调整系统时钟时也会发生此情况；例如，系统时钟在夏令时结束时往回调 1 小时。

配置类型

数据库

适用于

- 带有本地客户机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
- 带有本地客户机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 [NO, YES]

在系统时间段临时表中更新一行时，会生成带有 SYSTEM_TIME 时间段的历史记录行，该时间段指示该历史记录行中的数据为最新版本的时间范围。行开始列中的值指示该历史记录行中的数据何时变为最新版本。行结束列中的值指示该历史记录行何时变为历史数据。

下面是一个示例，说明两个冲突事务有可能生成其行结束时间戳记小于行开始时间戳记的历史记录行。

1. 事务 TRA 已在系统时间段临时表中对它在时间戳记 T1 时更新的一行生成行开始值。
2. 事务 TRB 已对它在时间戳记 T2 时（其中 T1 < T2）更新的同一行生成行开始值。
3. 事务 TRB 生成历史记录行并落实。
4. 事务 TRA 生成其历史记录行并落实。

完成此事件序列后，为事务 TRA 生成的历史记录行的结束时间戳记应小于其开始时间戳记。

通过允许存在冲突时调整时间戳记或回滚涉及的其中一个事务，数据库管理器可确保已生成历史记录行的结束时间戳记始终大于开始时间戳记。

NO 要插入的历史记录行的结束时间戳记小于开始时间戳记时，不会进行任何时间戳记值调整。反而，尝试插入该历史记录行的事务失败，并且返回错误 (SQLSTATE 57062, SQLCODE SQL20528N)。不允许调整可确保事务期间生成的所有历史记录行具有相同结束时间戳记并且可使用该结束时间戳记轻松标识。

YES 对系统时间段临时表的行开始列值的时间戳记值以及存在时间戳记冲突时生成

的历史记录行的结束时间戳记值进行调整。此调整将结束时间戳记修改为比开始时间戳记大 1 微秒。这将确保该历史记录行的结束时间戳记大于开始时间戳记。系统返回一条消息，指示进行了调整 (SQLSTATE 01695，SQLCODE SQL5191W)。

不需要任何时间戳记调整时，返回 SQLCODE DB20000I。

应用程序员可考虑使用 SQLCODE 或 SQLSTATE 值来处理来自 SQL 语句的与这些时间戳记值调整相关的返回码。

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]

在将此参数设为“YES”之后，必须执行完整数据库备份，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

- 可联机配置
- 可由 DB2 pureScale 环境中的成员配置

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。

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与 TSM 客户机选项文件中的 INCLUDE-EXCLUDE 列表所指定的管理类绑定。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 将备份对象与数据库配置参数指定的管理类重新绑定。

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
- 可由 DB2 pureScale 环境中的成员配置

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
- 可由 DB2 pureScale 环境中的成员配置

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
- 可由 DB2 pureScale 环境中的成员配置

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”

如果设为 YES，那么 **user_exit_status** 参数指示已对数据库管理器启用前滚恢复，并且数据库会根据 **logarchmeth1** 参数或 **logarchmeth2** 参数设置的值来对日志文件进行归档和检索。

配置类型
数据库

参数类型
参考

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
- 可由 DB2 pureScale 环境中的成员配置

传播类 立即

缺省值 [范围]
5000 [16 - 524 288]

注: 可以在最初创建数据库后通过 DB2 配置顾问程序来更改缺省值。

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为较大的值。

varchar2_compat -“Varchar2 兼容性”数据库配置参数

此参数指示是否将与 VARCHAR2 和 NVARCHAR2 数据类型相关联的兼容性语义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。

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

- 带有本地客户机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
- 带有本地客户机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。必须改用 backup 或 restore 实用程序的 **OPTIONS** 参数。

在配置为支持代理节点的 TSM 环境中，″-fromnode=nodename″ 选项和 ″-fromowner=ownername″ 选项与 ″-asnodename=nodename″ 选项不兼容，并且不能一起使用。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* 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的统计信息事件监视器，然后重置统计信息。如果存在活动的统计信息事件监视器，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、管道或表。如果它不存在，那么将只复位统计信息，而不进行收集。

将按照所指定的时间间隔（相对于星期天 00:00:00 开始测量）进行收集。当目录成员处于活动状态时，将在预定的下一个时间间隔开始时（相对于此固定时间）进行下一次收集。预定的时间间隔并不是相对于目录成员激活时间来计算的。如果成员在收集时并未处于活动状态，那么并不会为此成员收集统计信息。例如，如果时间间隔值设为 60，而目录成员是在星期天上午 9:24 激活的，那么将安排在每个整点时进行一次收集。因此，将在上午 10:00 进行下一次收集。如果成员在上午 10:00 并未处于活动状态，那么将不会为此成员收集统计信息。

收集和重置进程是从目录成员启动的。必须在目录成员上指定 **wlm_collect_int** 参数。它不在其他成员上使用。

配置类型

数据库

参数类型

- 可联机配置

缺省值 [范围]

`0 [0 (no collection performed), 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处理时间变长。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处理时间。

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重置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重置统计信息。

DB2 管理服务器（DAS）配置参数

authentication -“认证类型 DAS”

此参数确定如何进行以及在何处进行用户认证。

要点: V9.7 中已经不推荐使用“DB2 管理服务器 (DAS)”，在以后的发行版中可能会将其除去。DAS 在 DB2 pureScale 环境中不受支持。通过使用安全 Shell 协议的软件程

序进行远程管理。有关更多信息，请参阅『不推荐使用DB2 管理服务器 (DAS)』，网址为: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html。

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** 设为 SERVER_ENCRYPT，那么会将用户标识和密码从客户机发送至服务器，以便可以在服务器上执行认证。将对通过网络发送的用户标识和密码进行加密。

值 KERBEROS_ENCRYPT 意味着认证在 Kerberos 服务器上通过对认证使用 Kerberos 安全协议来执行。

注: 仅在运行 Windows 的服务器上支持 KERBEROS_ENCRYPT 认证类型。

只能从 V9 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 V8 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。

要点: V9.7 中已经不推荐使用“DB2 管理服务器 (DAS)”，在以后的发行版中可能会将其除去。DAS 在 DB2 pureScale 环境中不受支持。通过使用安全 Shell 协议的软件程

序进行远程管理。有关更多信息，请参阅『不推荐使用DB2 管理服务器 (DAS)』，网址为: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html。

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 V8 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。

要点: V9.7 中已经不推荐使用“DB2 管理服务器 (DAS)”，在以后的发行版中可能会将其除去。DAS 在 DB2 pureScale 环境中不受支持。通过使用安全 Shell 协议的软件程序进行远程管理。有关更多信息，请参阅『不推荐使用DB2 管理服务器 (DAS)』，网址为: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html。

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 V8 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。

要点: V9.7 中已经不推荐使用“DB2 管理服务器 (DAS)”，在以后的发行版中可能会将其除去。DAS 在 DB2 pureScale 环境中不受支持。通过使用安全 Shell 协议的软件程

序进行远程管理。有关更多信息，请参阅『不推荐使用DB2 管理服务器 (DAS)』，网址为: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html。

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。

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，**setup** 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

- 带有本地客户机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
- 带有本地客户机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 - 未捕获到任何诊断数据
- 1 - 仅严重错误
- 2 - 所有错误
- 3 - 所有错误和警告
- 4 - 所有错误、警告和参考消息

使用说明

- **diaglevel** 的动态行为不会扩展至所有进程。
- DB2 服务器进程 **db2sysc** 可以检测动态更改，例如，当您对实例附件发出 **UPDATE DATABASE MANAGER CONFIGURATION** 命令时。
- 当 DB2 客户机和应用程序进程启动时，它们将使用 **diaglevel** 配置参数设置，且不会检测任何动态更改。
- 为了帮助解决问题，您可以增大此参数的值以收集更多问题确定数据。
- 在特定环境下，为显示很重要的消息，DB2 将覆盖 **diaglevel** 配置参数设置。

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。

要点: V9.7 中已经不推荐使用“DB2 管理服务器 (DAS)”，在以后的发行版中可能会将其除去。DAS 在 DB2 pureScale 环境中不受支持。通过使用安全 Shell 协议的软件程序进行远程管理。有关更多信息，请参阅『不推荐使用DB2 管理服务器 (DAS)』，网址为: http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/topic/com.ibm.db2.luw.wn.doc/doc/i0059276.html。

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。如果 **discover** = SEARCH，那么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。

缺省情况下启用发现方式。

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** 设为 Yes，那么当“调度程序”重新启动时，将运行星期六作业。

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 解释器使用的环境变量是根据此参数的值计算出来的。

在 Windows 操作系统上，在 DB2 安装期间，Java 文件（如果需要的话）放置在 sqllib 目录（在 java\jdk 中）下面: 然后将 **jdk_path** 配置参数设为 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 V8 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，Data Studio）来在管理服务器上调度和执行任务。

此参数只能从 V8 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”开启时，该参数识别“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 V8 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。

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。

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 V8 命令行处理器 (CLP) 更新。

cf_sca_sz -“共享通信区”配置参数

此共享通信区 (SCA) 配置参数确定集群高速缓存设施 (CF) 中的 SCA 使用的内存大小。该 SCA 对应每个数据库实体，并且包含表、索引、表空间和目录的数据库范围控制块信息。

配置类型
数据库

适用于 DB2 pureScale 环境

参数类型
可联机配置

缺省值 [范围]
AUTOMATIC [AUTOMATIC, 2048 - 1073741824]

计量单位
页 (4 KB)

分配时间
第一次对任何 DB2 成员激活数据库的时间

释放时间
删除该数据库或停止 CF 服务器的时间

如果 **cf_sca_sz** 参数设为 AUTOMATIC（缺省值），那么 DB2 数据库管理器会在对任何成员第一次激活数据库时计算内存值，此值应足以供基本数据库操作使用。如果要为该 SCA 配置准确的内存大小，那么可将 **cf_sca_sz** 参数设为固定数字值。

可通过运行 **GET DB CFG SHOW DETAIL** 命令来获取 SCA 大小的当前值。如果 **cf_sca_sz** 参数设为 AUTOMATIC，那么 SHOW DETAIL 子句会显示 SCA 的计算值和分配值。

如果 SCA 内存已被数据库操作用光，那么会返回内存不足错误消息。在这种情况下，可通过将 **cf_sca_sz** 参数设为较高值来增加 SCA 的大小。

有关更多详细信息，请参阅『为数据库配置集群高速缓存设施内存』。

DB2 pureScale Feature 集群高速缓存设施配置

在 DB2 pureScale 环境中，集群高速缓存设施（又称为 CF）用于顺利执行全局锁定和缓冲池管理。CF 配置会集成到 DB2 基础结构内并由 DB2 配置界面管理。

CF 配置具有一组独有的配置参数，用于优化 DB2 pureScale 环境。

CF 的配置参数分类为 CF 服务器配置参数或 CF 结构配置参数。CF 服务器参数配置为 DB2 数据库管理器配置参数。CF 结构参数配置为 DB2 数据库配置参数。创建 DB2 pureScale Feature 实例或数据库时，会对配置文件应用缺省设置。

集群高速缓存设施配置

CF 服务器配置信息作为 DB2 数据库管理器配置信息的一部分保留。因此，CF 服务器会按使用 DB2 配置界面来配置数据库管理器的方式进行配置。DB2 命令行处理器 (CLP) 或配置 API 可用于显示和更新可配置 CF 服务器信息。

以下配置参数支持 CF 服务器:

- **cf_diaglevel**
- **cf_diagpath**
- **cf_mem_sz**
- **cf_num_conns**
- **cf_num_workers**

集群高速缓存设施结构配置

CF 结构内存作为 DB2 数据库配置信息的一部分保留。CF 结构参数按使用 DB2 CLP 或配置 API 来配置 DB2 数据库参数方式进行配置，以显示和更新可配置 CF 结构。

添加了以下配置参数以支持 CF 结构:

- **cf_db_mem_sz**
- **cf_gbp_sz**
- **cf_lock_sz**
- **cf_sca_sz**

在 CF 服务器中为每个数据库创建了 CF 结构。在调用 **CREATE DATABASE** 命令期间，DB2 配置顾问程序还会将 CF 结构配置参数设为其缺省值 AUTOMATIC 并计算 DB2 pureScale实例的适当启动值。

适当的 CF 结构配置对在 DB2 pureScale 环境中实现最佳性能很重要。CF 结构的所计算缺省值可能无法实现 DB2 pureScale 环境的最佳性能。在性能基准环境中，手动调整 CF structure 大小以实现必需的性能级别。还应注意，在 DB2 pureScale 下使用表分区会导致额外内存用量。

配置顾问程序

创建数据库时会调用 DB2 配置顾问程序，以根据主机环境（例如，内存量和处理器数）为数据库配置设置一组合理的缺省值。

DB2 配置顾问程序借助 DB2 pureScale Feature 得到了增强，能够计算 CF 结构配置参数的启动值。参数 **cf_db_mem_sz**、**cf_gbp_sz**、**cf_lock_sz** 和 **cf_sca_sz** 是根据 **cf_mem_sz** 定义的总 CF 内存大小计算的，以确保在这些结构中实现最佳内存分布，从而实现 DB2 pureScale实例的合理性能标准。

具有两个集群高速缓存设施的高可用性

在高可用 (HA) CF 环境中运行时，必须设置两个 CF 服务器。一个 CF 充当主服务器，另一个充当备份或辅助 CF 服务器。在此环境中，主 CF 服务器和辅助 CF 服务器共享同一主机配置。

有关高可用性环境中的结构双工支持的更多信息，请参阅辅助集群高速缓存设施的结构双工支持行为。

在有两个 CF 服务器的 DB2 pureScale 环境中，可使用卷动升级技巧来替换这两个集群高速缓存设施以维护高可用环境。

有关在 CF 替换期间维护高可用性环境的更多信息，请参阅替换两个集群高速缓存设施。

配置集群高速缓存设施

使用 DB2 命令行处理器 (CLP) 或 DB2 配置 API 为IBM DB2 pureScale Feature配置集群高速缓存设施 (CF) 参数。

关于此任务

集群高速缓存设施配置信息作为 DB2 数据库管理器配置信息的一部分保留。

集群高速缓存设施启动时，会从数据库管理器配置文件读取 CF 服务器配置参数以初始化 CF 服务器。CF 服务器会创建并侦听管理端口以接收 CF 服务器配置信息。CF 服务器或 CF 结构执行的 **db2start** 命令或联机操作完成后，CF 服务器会接收服务器配置参数，并且完全初始化。

如果 CF 服务器配置参数设为 AUTOMATIC，那么它们的值是在启动 CF 服务器期间自动确定的。在此启动期间，系统会查询 CF 服务器主机以了解物理 RAM 量以及 CPU 数目，这可帮助为 **cf_mem_sz** 和 **cf_num_workers** 配置参数配置适当的值。

可通过两种方法来配置 CF 服务器:

- 使用 CLP 来更新配置参数
- 使用 API 来更新配置参数

过程

- 要在 CLP 中查看 CF 服务器配置参数的值，请发出 **GET DATABASE MANAGER CONFIGURATION** 命令。

  例如，在 DB2 pureScale实例中发出此命令时，会显示以下 CF 服务器配置信息:

```
CF 服务器配置:
   内存大小 (4KB)                          (CF_MEM_SZ) = AUTOMATIC
   工作程序线程数                  (CF_NUM_WORKERS) = AUTOMATIC
   连接数                         (CF_NUM_CONNS) = AUTOMATIC
   诊断错误捕获级别                (CF_DIAGLEVEL) = 2
   诊断数据目录路径                (CF_DIAGPATH)
```

  有关更多详细输出，请参阅 **GET DATABASE MANAGER CONFIGURATION** 命令。

- 要在 CLP 中更新 CF 服务器配置参数的值，请发出 **UPDATE DATABASE MANAGER CONFIGURATION** 命令。

  例如，发出以下命令以更新 CF 内存大小 (**cf_mem_sz**) 时，会显示以下信息:

```
UPDATE DBM CFG USING CF_MEM_SZ 27152

DB20000I  The UPDATE DATABASE MANAGER CONFIGURATION 命令已成功完成。

SQL1362W  未动态更改针对即时修改提交的一个或多个参数。客户机更改直到下次
应用程序启动或发出 TERMINATE 命令时才会生效。服务器更改
直到下一次执行 DB2START 命令时才能生效。
```

  CF 服务器配置更改不会立即生效。所有更新值会保存下来，并在下一次 CF 服务器启动时应用。

- 要在 CLP 中重置 CF 服务器配置参数的值，请发出 **RESET DATABASE MANAGER CONFIGURATION**（或 **RESET DBM CFG**）命令。DB2 数据库管理器配置参数重置为缺省值。
- 要使用应用程序编程接口 (API) 来更新 CF 服务器配置参数的值，请调用 db2CfgSet API。要查看这些值，请发出 db2CfgGet API。

调用 db2CfgGet 或 db2CfgSet API 时使用的 **db2Cfg** 数据结构中的标记元素的条目支持 CF 服务器配置参数的获取和设置。

*表 138. CF 服务器配置参数的获取和设置中的受支持 **db2Cfg** 标记*

标记	数据类型
SQLF_KTN_CF_MEM_SZ	Uint32
SQLF_KTN_CF_NUM_WORKERS	Uint32
SQLF_KTN_CF_NUM_CONNS	Uint32
SQLF_KTN_CF_DIAGLEVEL	Uint16
SQLF_KTN_CF_DIAGPATH SQLF_KTN_CF_DIAGPATH_FULL	char(215)

为数据库配置集群高速缓存设施内存

使用 DB2 命令行处理器 (CLP) 或 DB2 配置 API 来显示并更新数据库的集群高速缓存设施内存配置参数。

关于此任务

集群高速缓存设施结构内存配置信息作为 DB2 数据库配置信息的一部分保留。

在任何成员上第一次激活数据库期间为 集群高速缓存设施 分配了用于组缓冲池 (GBP)、锁定使用以及共享通信区 (SCA) 的 集群高速缓存设施 结构内存，此内存保持被分配状态直到在上一个成员被取消激活。这些参数的缺省值设为 AUTOMATIC。设为 AUTOMATIC 时，DB2 会在数据库激活期间计算这些参数的适当大小。因为这些值之间紧密相关并相互依赖，所以手动设置其中至少一个参数会导致激活期间无法计算所有参数，即使某些参数仍设为 AUTOMATIC 都是如此。它们的值显示为最近自动计算的值，并且可使用 **GET DB CFG SHOW DETAIL** 命令查看。

结构参数也支持 ONLINE 选项。对 CF 内存参数的所有更新会立即应用。更新请求是同时发出的，并且直到 CF 服务器设置了新值才会返回。

可通过两种方法来为数据库配置集群高速缓存设施内存:

- 使用 CLP 来更新配置参数
- 使用 API 来更新配置参数

过程

- 要在 CLP 中查看 CF 结构配置参数的值，请发出 **GET DATABASE CONFIGURATION** 命令。

  例如，在 DB2 pureScale实例中发出此命令时，会显示以下 CF 结构配置信息:

```
CF Resource Configuration:
     CF database memory size (4KB)           (CF_DB_MEM_SZ) = AUTOMATIC
     Group buffer pool size (4KB)            (CF_GBP_SZ)    = AUTOMATIC
     Global lock memory size (4KB)           (CF_LOCK_SZ)   = AUTOMATIC
     Shared communication area size (4KB)    (CF_SCA_SZ)    = AUTOMATIC
```

  有关详细输出，请参阅 **GET DATABASE CONFIGURATION** 命令。

  注: 组缓冲池 (GBP) 参数在 CACHE 结构中需要足够的目录条目来处理读取并注册 (RAR) 请求。DB2 会针对每个 RAR 请求监视可用目录条目和数据元素计数并根据

需要自动调整这些值。此调整以动态方式完成或通过联机更新完成，不管 **cf_gbp_sz** 参数是否配置为 AUTOMATIC 都是如此。有关新 CF 结构配置参数的更多信息，请参阅“DB2 pureScale Feature 的配置参数摘要”主题。

使用 **SHOW DETAIL** 选项来查看结构配置参数的当前值时，会显示以下信息:

```
CF Resource Configuration:
  Group buffer pool size (4KB)
              (CF_GBP_SZ)  = AUTOMATIC(32768)  AUTOMATIC(32768)
  Global lock memory size (4KB)
              (CF_LOCK_SZ) = AUTOMATIC(17000)  AUTOMATIC(17000)
  Shared communication area size (4KB)
              (CF_SCA_SZ)  = AUTOMATIC(1500)   AUTOMATIC(1500)
```

有关详细输出，请参阅 **GET DATABASE CONFIGURATION** 命令。

针对每个结构参数显示的结果有两个相应值。如果圆括号中的值不同，那么表示存在对结构参数值的暂挂更新。括在第一组圆括号中的值是结构参数的当前值。括在第二组圆括号中的值是结构参数的暂挂值。

同时存在主 CF 服务器和辅助 CF 服务器时，只会显示主 CF 服务器的配置参数。同一值将应用于这两个 CF 服务器。

- 要在 CLP 中更新 CF 结构配置参数的值，请发出 **UPDATE DATABASE CONFIGURATION**（或 **UPDATE DB CFG**）命令。将结构参数更新为缺省设置 (AUTOMATIC) 以外的任何值会导致 CF 结构内存大小被修正。

  例如，发出以下命令以将全局锁定管理器内存的大小设为 20,000 页:

  ```
  UPDATE DATABASE CONFIGURATION FOR WSDB USING CF_LOCK_SZ 20000
  ```

  注: 如果没有足够的未使用内存来容纳此请求，那么用于降低结构参数的内存量的更新不会立即生效。要了解是否有任何针对结构参数的更新暂挂，请从 CLP 发出 **GET DB CFG** 命令。
- 要在 CLP 中重置 CF 结构配置参数的值，请发出 **RESET DATABASE CONFIGURATION**（或 **RESET DB CFG**）命令。CF 结构配置参数会重置为缺省值 AUTOMATIC。
- 要使用应用程序编程接口 (API) 来更新 CF 结构配置参数的值，请调用 db2CfgSet API。要查看这些值，请发出 db2CfgGet API。

  调用 db2CfgGet 或 db2CfgSet API 时使用的 **db2Cfg** 数据结构中的标记元素的条目，该条目支持获取和设置 CF 结构配置参数。

*表 139. 获取和设置 CF 结构配置参数时受支持的 **db2Cfg** 标记*

标记	数据类型
SQLF_DBTN_CF_GBP_SZ	Uint32
SQLF_DBTN_CF_SCA_SZ	Uint32
SQLF_DBTN_CF_LOCK_SZ	Uint32
SQLF_DBTN_CF_DB_MEM_SZ	Uint32

DB2 pureScale CF 内存参数配置

在 DB2 pureScale实例中，一些配置参数和注册表变量配合工作以控制集群高速缓存设施（又称为 CF）的内存分配。

参数

以下配置参数和注册表变量影响 CF 内存分配和管理。这些参数和变量只有在成员启动或重新启动后才生效，描述中另行声明时除外。

- **cf_mem_sz**: 控制 CF 使用的总内存。缺省设为 AUTOMATIC。
- **cf_db_mem_sz**: 控制 CF 用于此特定数据库的总内存限制。缺省设为 AUTOMATIC。
- **cf_gbp_sz**: 确定 CF 用于此特定数据库的组缓冲池的内存。缺省设为 AUTOMATIC。
- **cf_lock_sz**: 确定 CF 用于此特定数据库的锁定的内存。缺省设为 AUTOMATIC。
- **cf_sca_sz**: 确定 CF 用于此特定数据库的共享通信区 (SCA) 的内存。缺省设为 AUTOMATIC。
- **numdb**: 指定可同时处于活动状态的本地数据库数。活动数据库的缺省数目为 32。此参数只有在组重新启动后才生效。
- **DB2_DATABASE_CF_MEMORY**: 此注册表变量确定总 CF 内存中指定给每个数据库的百分比，其中 **cf_db_mem_sz** 值设为 AUTOMATIC。缺省百分比为 100。此变量只有在组重新启动后才生效。

设为 AUTOMATIC 时，CF 内存配置参数的值将根据 DB2 pureScale 环境的特性和属性动态生成。

参数关系

注: 将 **numdb** 的值设为高于或等于实例上的数据库总数。此值包括处于活动和不活动状态的数据库。此外，还可设置 **DB2_DATABASE_CF_MEMORY** 的值以使此实例上的每个数据库（不管通常是处于不活动状态还是处于活动状态）可同时处于活动状态。

同时在实例级别和数据库级别定义了集群高速缓存设施内存参数:

- 在实例级别，为 CF 保留的内存总量由 **cf_mem_sz** 配置参数确定。
- 在数据库级别，数据库的所有 CF 结构使用的内存总量由 **cf_db_mem_sz** 配置参数控制。

数据库级别 **cf_db_mem_sz** 参数定义的内存限制由实例级别 **cf_mem_sz** 参数约束。因此，上限是在 **cf_db_mem_sz** 参数上设置的: 它不能超过此内存阈值来满足 CF 数据库结构的内存需求。所有活动数据库的 **cf_db_mem_sz** 参数值的总和不能超过实例的 **cf_mem_sz** 的值。

同样，尽管 **cf_db_mem_sz** 参数控制数据库的所有 CF 结构可使用的内存总量，但数据库管理器决不会保留等于 **cf_db_mem_sz** 值的内存块。

用于控制数据库的每个 CF 结构的内存量的配置参数包括:

- **cf_gbp_sz**
- **cf_lock_sz**
- **cf_sca_sz**

CF 在内部使用 3840 个 4K 页；这些页取自 **cf_mem_sz** 参数指定的内存量。此外，还将为实例上的每个活动数据库保留 256 个 4K 页（取自 **cf_mem_sz**）。仅当手动设置 **cf_mem_sz** 参数时，才包括这些附加内存量。

自动配置

cf_mem_sz 参数的缺省设为 AUTOMATIC。这意味着可供 CF 使用的内存量是使用 CF 主机上可用的内存总量计算的。该参数动态设为:

- 适当百分比通常为 CF 主机上可用总内存的 70% 到 90%。
- CF 主机上的可用内存量。

（取两者中的较低值）。

cf_db_mem_sz 参数的缺省设为 AUTOMATIC。实际值根据第一次对任何成员激活数据库时生效的 **DB2_DATABASE_CF_MEMORY**、**cf_mem_sz** 和 **numdb** 参数值确定。

AUTOMATIC 计算期间考虑的另一个因素是 CF 是否与任何 DB2 成员共存。

联机配置

cf_mem_sz 参数不可联机配置。CF 服务器必须重新启动，**cf_mem_sz** 的新值才能生效。

cf_db_mem_sz 参数可联机配置，但有一些约束需要注意:

- **cf_db_mem_sz** 的值必须始终小于 **cf_mem_sz** 的当前值。如果要将 **cf_db_mem_sz** 提高至大于 **cf_mem_sz** 的当前值的值，那么必须先提高 **cf_mem_sz** 的值。
- 即使 **cf_db_mem_sz** 参数提高至小于 **cf_mem_sz** 参数的值，出现以下情况时也会导致错误:
  - 请求联机提高 CF 结构内存参数，其中 CF 结构内存参数之和*未超过* **cf_db_mem_sz** 的值，但*超过* **cf_mem_sz** 参数设置的上限阈值限制。CF 中没有足够可用内存，无法满足 CF 结构内存请求，因此生成了错误。
- **cf_db_mem_sz** 的值必须始终大于 **cf_gbp_sz**、**cf_lock_sz** 与 **cf_sca_sz** 参数之和。如果 **cf_db_mem_sz** 的值降低，那么这三个 CF 结构参数的值也必须降低以符合 **cf_db_mem_sz** 参数设置的已更改内存限制。如果未能这样做，那么会导致错误，因为 **cf_db_mem_sz** 参数提供的内存不足。

cf_gbp_sz、**cf_lock_sz** 和 **cf_sca_sz** 参数可联机配置，但有一些约束需要注意:

- 如果系统很忙或者提交请求时没有足够的可用 CF 内存，那么尝试更改这些参数可能导致超时错误。

  注: 尽管提交请求时内存可能不可用，但系统会继续尝试更新以防有足够内存变为可用从而满足此请求。更新操作会等待处理完成。
- 在 CF 结构内存参数的更新操作期间，如果启动了更新同一结构内存参数的另一请求，那么会触发错误。

多个活动数据库

在运行多个活动数据库的 DB2 pureScale 环境中，规划每个活动数据库的内存配置需求时有一些其他注意事项。此情况下的关键配置参数为 **DB2_DATABASE_CF_MEMORY**。此参数与其他 CF 配置参数配合使用以指示 CF 内存总量中将分配给每个数据库的百分比。**DB2_DATABASE_CF_MEMORY** 参数表示为浮点数据类型，并且如果其:

值为 -1
每个活动数据库的 **cf_db_mem_sz** 参数的动态生成值等于以下计算的结果: **cf_mem_sz** 除以 **numdb**:

$$cf\_db\_mem\_sz = \frac{cf\_mem\_sz}{numdb}$$

这意味着处于活动状态的每个数据库从 **cf_mem_sz** 接收到相等份额的内存（如果该数据库的 **cf_db_mem_sz** 参数设为 AUTOMATIC）。

注：如果 **cf_db_mem_sz** 的计算值小于 **cf_mem_sz** 参数的最小值，那么 **cf_db_mem_sz** 值自动设为 **cf_db_mem_sz** 的最小允许值。

值为 0

DB2 pureScale 环境中的每个数据库都必须手动设置 **cf_db_mem_sz** 值。或者，如果 **cf_db_mem_sz** 的值设为 AUTOMATIC，那么该数据库将接收到最小允许值 32,768。

值介于 0 到 100 之间

cf_db_mem_sz 参数的值等于以下计算的结果:

$$cf\_db\_mem\_sz = cf\_mem\_sz * \frac{db2\_database\_cf\_memory}{100}$$

注：如果 **cf_db_mem_sz** 的计算值小于 **cf_mem_sz** 参数的最小值，那么 **cf_db_mem_sz** 值自动设为 **cf_db_mem_sz** 的最小允许值。

值为 100

只有一个数据库能够在 DB2 pureScale 环境中激活，并且它将接收 **cf_mem_sz** 参数定义的所有 CF 内存。

在 **DB2_DATABASE_CF_MEMORY** 用于动态计算 **cf_db_mem_sz** 参数值的配置中，**cf_db_mem_sz** 的值决不能小于其最小允许值，不管 **DB2_DATABASE_CF_MEMORY** 参数的设置如何都是如此。

限制

不支持在数据库之间自我调整 CF 内存。

示例

示例 1: 在此示例中，**DB2_DATABASE_CF_MEMORY** 注册表变量未更改其缺省设置“100”。因此，只有一个数据库接收到所有 CF 内存（此内存量等于 **cf_mem_sz** 参数定义的值）。如果创建或启动另一数据库，那么该命令将失败，并产生内存不足错误。为避免此情况以便可创建具有相等内存份额的另一数据库，必须修改 CF 内存参数（这假定 **cf_db_mem_sz** 设为 AUTOMATIC）:

```
db2stop
db2 update dbm cfg using numdb 2
db2set db2_database_cf_memory=50
db2 start
```

示例 2: 在此示例中，**DB2_DATABASE_CF_MEMORY** 注册表变量更改为给予所有数据库相等份额的 CF 内存（根据有两个活动数据库的情况）:

```
db2stop
db2 update dbm cfg using numdb 2
db2set db2_database_cf_memory=-1
db2 start
db2 connect to INVOICES      # success
db2 connect to HRDEPT        # success
```

示例 3: 在此示例中，**DB2_DATABASE_CF_MEMORY** 注册表变量更改为给予所有 200 个数据库相等份额的 CF 内存（根据有 200 个处于活动状态的数据库的情况）：

```
db2stop
db2 update dbm cfg using numdb 200
db2set db2_database_cf_memory=0.5
db2 start
```

辅助集群高速缓存设施的结构双工支持行为

在高可用环境中运行时，必须使用多个集群高速缓存设施以在集群高速缓存设施（又称为 CF）当机的情况下提供持续支持。

集群高速缓存设施支持以下结构:

- 组缓冲池 (GBP)
- CF 锁管理器 (LOCK)
- 共享通信区 (SCA)。

主 CF 服务器和辅助 CF 服务器在高可用环境中共用同一主机配置。建议对这两个 CF 服务器使用相同的主机规格，例如，CPU 速度、CPU 数以及随机存取存储器 (RAM) 量。这两个 CF 服务器共用一组配置参数并访问相同端口号。

CF 具有可配置结构: GBP、LOCK 和 SCA。每个结构都支持联机更新。增加这些结构的已分配内存（又称为增大结构）时，会先向辅助 CF 服务器发送请求，然后向主 CF 服务器发送请求。降低这些结构的已分配内存（又称为缩小结构）时相反。缩小请求先发送至主 CF 服务器，然后发送至辅助 CF 服务器。辅助 CF 中的结构始终与主 CF 中的结构一样大或比后者大，以确保 CF 故障转移后没有存储问题。

如果主 CF 或辅助 CF 未能更新 CF 配置参数，那么另一 CF 会回滚至该 CF 配置参数的先前设置。此回滚确保这两个 CF 上有足够的结构内存可用。

缺省情况下，调整大小请求会等待至调整大小操作完成才返回至客户机。如果使用用户控制操作（例如 **Ctrl+C**）来中断调整大小请求，那么调整大小操作会在这两个 CF 上取消。然后，辅助 CF 上的结构的大小会调整以匹配主 CF 上的值。可通过运行 `DB2 GET DB CFG SHOW DETAIL` 命令来监视调整大小请求。

ingest 实用程序配置参数

commit_count -“落实计数”配置参数

此参数指定发出落实之前每个清仓程序在单个事务中写入的行数。

配置类型

ingest 实用程序

适用于 ingest 实用程序

参数类型

可配置

缺省值 [范围]

0 [0 至最大 32 位带符号整数]

如果 **commit_count** 不是 1,000 的倍数，那么它将舍入为最接近的倍数并且 ingest 实用程序会发出警告 (SQL2903W)。

commit_count 设为 0（缺省值）时，会使用 **commit_period**，意味着缺省情况下，ingest 实用程序仅根据耗用时间落实事务。如果要仅根据行数落实事务，那么必须将 **commit_count** 设为非零值，并将 **commit_period** 设为 0。

如果既未指定 **commit_count** 也未指定 **commit_period**，那么会使用 **commit_period** 缺省设置 1 秒。

如果指定了 **commit_count** 和 **commit_period**，那么 ingest 实用程序会接受这两个参数的设置；即，如果它已写入指定行数或指定秒数内没有落实操作，那么它会发出落实。

建议 如果行大小很小，请将 **commit_count** 设为较大值。如果行大小很大，请将 **commit_count** 设为较小值。

如果没有其他应用程序在运行，那么可使用以下公式极粗略地估算绝对最大落实计数:

```
(logfilsiz * (logprimary + logsecond) * 4KB) divided by (estimated row size + overhead)
   divided by (total number of flushers)
```

如果其他应用程序在运行，那么最大落实计数较小。

如果 **commit_count** 的值设置得太大，那么锁定列表或事务日志将变满。在这种情况下，INGEST 命令会终止，并生成错误 SQL0912N 或 SQL0964C。如果接收到 SQL0912N 或 SQL0964C，并且已设置 **retry_count** 配置参数，那么 ingest 实用程序会执行下列两个操作

- 调整 **commit_count** 和/或 **commit_period** 的设置并发出警告消息
- 发出落实并重试该操作

commit_period -“落实时间段”配置参数

指定落实事务之间的秒数。

配置类型
ingest 实用程序

适用于 ingest 实用程序

参数类型
可配置

缺省值 [范围]
1 [0 至 2,678,400（31 天）]

计量单位
秒

每次清仓时，ingest 实用程序会检查自上次落实以来经历的秒数。如果它大于或等于 **commit_period** 的设置，那么 ingest 实用程序会发出落实。

如果 **commit_period** 设为 0，那么会使用 **commit_count** 配置参数，意味着将根据行数落实事务。

如果既未指定 **commit_count** 也未指定 **commit_period**，那么会使用 **commit_period** 缺省设置 1 秒。

如果指定了 **commit_count** 和 **commit_period**，那么 ingest 实用程序会接受这两个参数的设置；即，如果它已写入指定行数或指定秒数内没有落实操作，那么它会发出落实。

建议 如果行大小很小，请将 **commit_period** 设为较大值。如果行大小很大，请将 **commit_period** 设为较小值。

如果 **commit_period** 的值设置得太大，那么锁定列表或事务日志将变满。在这种情况下，INGEST 命令会终止，并生成错误 SQL0912N 或 SQL0964C。如果接收到 SQL0912N 或 SQL0964C，并且已设置 **retry_count** 配置参数，那么 ingest 实用程序会执行下列两个操作

- 调整 **commit_count** 和/或 **commit_period** 的设置并发出警告消息
- 发出落实并重试该操作

num_flushers_per_partition -“每个数据库分区的清仓程序数”配置参数

指定要为每个数据库分区分配的清仓程序数。

配置类型
ingest 实用程序

适用于 ingest 实用程序

参数类型
可配置

缺省值 [范围]
max(1, ((逻辑 CPU 数)/2)/(分区数)) [0 至系统资源数]

如果 **num_flushers_per_partition** 设为 0。那么对所有分区使用一个清仓程序。

注: 如果操作为 DELETE、MERGE 或 UPDATE，那么该实用程序有时会将此参数降至 0 或 1 以降低发生死锁的机率 (SQL2903W)。

num_formatters -“格式化程序数”配置参数

指定要分配的格式化程序的数目。

配置类型
ingest 实用程序

适用于 ingest 实用程序

参数类型
可配置

缺省值 [范围]
max(1, ((逻辑 CPU 数)/2)) [1 至系统资源数]

尽管缺省值为最佳设置，但如果您在 **INGEST** 命令上指定 DUMPFILE（或 BADFILE）参数并且您希望 ingest 实用程序按无效记录出现在输入文件中的次序写下这些记录，那么必须将 **num_formatters** 设为 1。

pipe_timeout -“管道超时”配置参数

此参数指定输入源为管道时等待数据的最大秒数。

配置类型

ingest 命令

适用于 ingest 实用程序

参数类型

可配置

缺省值 [范围]

600 秒（10 分钟）[0 至 2,678,400（31 天）]

计量单位

秒

如果 **pipe_timeout** 设为 0，那么 ingest 实用程序会无限期等待。如果指定时间段内没任何数据到达，那么 **INGEST** 命令会结束并返回错误。

retry_count -“重试计数”配置参数

指定重试已失败但可恢复的事务的次数。

配置类型

ingest 实用程序

适用于 ingest 实用程序

参数类型

可配置

缺省值 [范围]

0 [0 至 1,000]

ingest 实用程序仅重试因为下列其中一个原因失败的事务:

- 连接失败但已重新建立。
- 发生了自动回滚的死锁或超时。
- 系统错误导致工作单元回滚。
- 虚拟存储器或数据库资源不可用。

retry_period -“重试时间段”配置参数

指定重试已失败但可恢复的事务之前等待的秒数。

配置类型

ingest 实用程序

适用于 ingest 实用程序

参数类型

可配置

缺省值 [范围]

0 [0 至 2,678,400（31 天）]

计量单位

秒

ingest 实用程序仅重试因为下列其中一个原因失败的事务:

- 连接失败但已重新建立。
- 发生了自动回滚的死锁或超时。
- 系统错误导致工作单元回滚。
- 虚拟存储器或数据库资源不可用。

shm_max_size -“共享内存的最大大小”配置参数

指定进程间通信 (IPC) 共享内存的最大大小（以字节计）。因为 ingest 实用程序在客户机上运行，所以此内存是在客户机上分配的。

配置类型

ingest 命令

适用于 ingest 实用程序

参数类型

可配置

缺省值 [范围]

1 GB (1,073,741,824) [*n* 至可用内存]

n 是按如下所示计算的:

```
11000 +
(numTransporters x 500) +
(NUM_FORMATTERS x 500) +
(numUsedPartitions x 50) +
(totalNumFlushers x 4000) +
(MSG_BUF_COUNT x (100 + MSG_BUF_SIZE)) +
(numFields x 66300) +
(1.5 x NUM_FORMATTERS x sumOfAllFieldLengths)
```

其中:

- 如果操作为 INSERT 或 REPLACE，那么 *numTransporters* 是输入源数，否则为 1。
- *numUsedPartitions* 是目标表使用的数据库分区数。
- *totalNumFlushers* 为 **num_flushers_per_partition** x *numUsedPartitions*。
- *sumOfAllFieldLengths* 是所有字段定义中的总字节数。
- *numFields* 是字段定义数。

计量单位

字节

第 5 部分 附录

附录 A. DB2 技术信息概述

DB2 技术信息以多种可以通过多种方法访问的格式提供。

您可以通过下列工具和方法获得 DB2 技术信息:

- DB2 信息中心
  - 主题（任务、概念和参考主题）
  - 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。

您可以在线访问 ibm.com 上的其他 DB2 技术信息，例如技术说明、白皮书和 IBM Redbooks® 出版物。请访问以下网址处的 DB2 信息管理软件资料库站点: 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，请向 db2docs@ca.ibm.com 发送电子邮件。DB2 文档小组将阅读您的所有反馈，但无法直接给您答复。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。

请不要使用以上电子邮件地址与 DB2 客户支持机构联系。如果您遇到文档无法解决的 DB2 技术问题，请与您当地的 IBM 服务中心联系以获得帮助。

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?rs=71&uid=swg2700947 下载 PDF 格式的 DB2 V10.1 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。

每次更新手册时，表单号都会递增。确保您正在阅读下面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 140. 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	SC27-3864-00	是	2012 年 4 月
Administrative Routines and Views	SC27-3865-01	否	2013 年 1 月
Call Level Interface Guide and Reference Volume 1	SC27-3866-01	是	2013 年 1 月
Call Level Interface Guide and Reference Volume 2	SC27-3867-01	是	2013 年 1 月
Command Reference	SC27-3868-01	是	2013 年 1 月
数据库管理概念和配置参考	S151-1758-01	是	2013 年 1 月
数据移动实用程序指南和参考	S151-1756-01	是	2013 年 1 月
数据库监视指南和参考	S151-1759-01	是	2013 年 1 月
数据恢复及高可用性指南与参考	S151-1755-01	是	2013 年 1 月
数据库安全性指南	S151-1753-02	是	2013 年 1 月
DB2 Workload Management Guide and Reference	SC27-3891-01	是	2013 年 1 月
开发 ADO.NET 和 OLE DB 应用程序	S151-1765-01	是	2013 年 1 月
开发嵌入式 SQL 应用程序	S151-1763-01	是	2013 年 1 月
Developing Java Applications	SC27-3875-01	是	2013 年 1 月
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-3876-00	否	2012 年 4 月
Developing RDF Applications for IBM Data Servers	SC27-4462-00	是	2013 年 1 月
开发用户定义的例程（SQL 和外部例程）	S151-1761-01	是	2013 年 1 月
数据库应用程序开发入门	G151-1764-01	是	2013 年 1 月
Linux 和 Windows 上的 DB2 安装和管理入门	G151-1769-00	是	2012 年 4 月
全球化指南	S151-1757-00	是	2012 年 4 月
安装 DB2 服务器	G151-1768-01	是	2013 年 1 月
安装 IBM Data Server Client	G151-1751-00	否	2012 年 4 月
消息参考第 1 卷	S151-1767-01	否	2013 年 1 月

表 140. 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
消息参考第 2 卷	S151-1766-01	否	2013 年 1 月
Net Search Extender 管理和用户指南	S151-1905-01	否	2013 年 1 月
分区和集群指南	S151-1754-01	是	2013 年 1 月
Preparation Guide for DB2 10.1 Fundamentals Exam 610	SC27-4540-00	否	2013 年 1 月
Preparation Guide for DB2 10.1 DBA for Linux, UNIX, and Windows Exam 611	SC27-4541-00	否	2013 年 1 月
pureXML 指南	S151-1775-01	是	2013 年 1 月
Spatial Extender User's Guide and Reference	SC27-3894-00	否	2012 年 4 月
SQL 过程语言: 应用程序启用和支持	S151-1762-01	是	2013 年 1 月
SQL Reference Volume 1	SC27-3885-01	是	2013 年 1 月
SQL Reference Volume 2	SC27-3886-01	是	2013 年 1 月
Text Search Guide	SC27-3888-01	是	2013 年 1 月
故障诊断和调整数据库性能	S151-1760-01	是	2013 年 1 月
升级到 DB2 V10.1	S151-1770-01	是	2013 年 1 月
DB2 V10.1 新增内容	S151-1752-01	是	2013 年 1 月
XQuery 参考	S151-1774-01	否	2013 年 1 月

表 141. 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 安装和配置 DB2 Connect Personal Edition	S151-1773-00	是	2012 年 4 月
DB2 Connect 安装和配置 DB2 Connect 服务器	S151-1772-01	是	2013 年 1 月
DB2 Connect 用户指南	S151-1771-01	是	2013 年 1 月

从命令行处理器显示 SQL 状态帮助

DB2 产品针对可能充当 SQL 语句结果的条件返回 SQLSTATE 值。SQLSTATE 帮助说明 SQL 状态和 SQL 状态类代码的含义。

过程

要启动 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

您可以在 ibm.com® 上的不同信息中心中找到其他版本 DB2 产品的文档。

关于此任务

对于 DB2 V10.1 主题，*DB2 信息中心* URL 是 http://publib.boulder.ibm.com/infocenter/db2luw/v10r1。

对于 DB2 V9.8 主题，*DB2 信息中心* URL 是 http://publib.boulder.ibm.com/infocenter/db2luw/v9r8/。

对于 DB2 V9.7 主题，*DB2 信息中心* URL 是 http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/。

对于 DB2 V9.5 主题，*DB2 信息中心* URL 是 http://publib.boulder.ibm.com/infocenter/db2luw/v9r5。

对于 DB2 V9.1 主题，*DB2 信息中心* URL 是 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/。

更新安装在计算机或内部网服务器上的 DB2 信息中心

安装在本地的 DB2 信息中心必须定期进行更新。

开始之前

必须已安装 DB2 V10.1 信息中心。有关详细信息，请参阅安装 DB2 服务器中的“使用 DB2 安装向导来安装 DB2 信息中心”主题。所有适用于安装信息中心的先决条件和限制同样适用于更新信息中心。

关于此任务

可以自动或手动更新现有的 DB2 信息中心:

- 自动更新将更新现有的信息中心功能部件和语言。自动更新的一个优点是，与手动更新相比，信息中心的不可用时间较短。另外，自动更新可设置为作为定期运行的其他批处理作业的一部分运行。
- 可以使用手动更新方法来更新现有的信息中心功能部件和语言。自动更新可以缩短更新过程中的停机时间，但如果您想添加功能部件或语言，那么必须执行手动过程。例如，如果本地信息中心最初安装的是英语和法语版，而现在还要安装德语版；那么手动更新将安装德语版，并更新现有信息中心的功能和语言。但是，手动更新要求您手动停止、更新和重新启动信息中心。在整个更新过程期间信息中心不可用。在自动更新过程中，信息中心仅在更新完成后停止工作以重新启动信息中心。

此主题详细说明了自动更新的过程。有关手动更新的指示信息，请参阅“手动更新安装在您的计算机或内部网服务器上的 DB2 信息中心”主题。

过程

要自动更新安装在计算机或内部网服务器上的 DB2 信息中心:

1. 在 Linux 操作系统上,
   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/V10.1 目录中。
   b. 从安装目录浏览至 doc/bin 目录。
   c. 运行 update-ic 脚本:

      ```
      update-ic
      ```
2. 在 Windows 操作系统上,
   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\V10.1 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 update-ic.bat 文件:

      ```
      update-ic.bat
      ```

结果

DB2 信息中心将自动重新启动。如果更新可用，那么信息中心会显示新的以及更新后的主题。如果信息中心更新不可用，那么会在日志中添加消息。日志文件位于 doc\eclipse\configuration 目录中。日志文件名称是随机生成的编号。例如，1239053440785.log。

手动更新安装在计算机或内部网服务器上的 DB2 信息中心

如果您已在本地安装 DB2 信息中心，那么可从 IBM 获取文档更新并进行安装。

关于此任务

手动更新安装在本地的 *DB2 信息中心*要求您:

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。DB2 信息中心的工作站版本总是以独立方式运行。
2. 使用“更新”功能部件来查看可用的更新。如果有您必须安装的更新，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，请使用一台已连接至因特网并已安装 *DB2 信息中心*的机器将更新站点镜像至本地文件系统。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，只有在单机方式下才能使用“更新”功能部件。

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 2008、Windows Vista 和更高版本上，稍后列示在此部分的命令必须作为管理员运行。要打开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**。

过程

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/db2icdv10 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\V10.1 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/V10.1 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 浏览器将打开以显示独立信息中心。
3. 单击**更新**按钮（  ）。（必须在浏览器中启用 JavaScript。）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. 要启动安装过程，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，浏览至安装目录中的 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停止使用 help_start 批处理文件启动的进程所需的命令。不要使用 Ctrl-C 或任何其他方法来停止 help_start.bat。
   - 在 Linux 上，浏览至安装目录中的 doc/bin 目录并运行 help_end 脚本:

     ```
     help_end
     ```

注: help_end 脚本包含安全地停止使用 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停止 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/db2icdv10 start
     ```

结果

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 数据库产品的各个方面。这些课程提供了逐步指示信息。

开始之前

您可以在信息中心中查看 XHTML 版的教程: http://publib.boulder.ibm.com/infocenter/db2luw/v10r1/。

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 数据存储器执行基本操作。

DB2 故障诊断信息

我们提供了各种各样的故障诊断和问题确定信息来帮助您使用 DB2 数据库产品。

DB2 文档
: 您可以在《*故障诊断和调整数据库性能*》或者 *DB2 信息中心*的“数据库基础”部分中找到故障诊断信息，这些信息包含以下内容:
  - 有关如何使用 DB2 诊断工具和实用程序来隔离和确定问题的信息。
  - 一些最常见问题的解决方案。
  - 旨在帮助您解决 DB2 数据库产品使用过程中可能会遇到的其他问题的建议。

IBM Support Portal
: 如果您遇到问题并且希望得到帮助以查找可能的原因和解决方案，请访问 IBM Support Portal 。这个技术支持站点提供了指向最新 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。

  访问 IBM 支持门户网站网址为: http://www.ibm.com/support/entry/portal/Overview/Software/Information_Management/DB2_for_Linux,_UNIX_and_Windows

信息中心条款和条件

如果符合以下条款和条件，那么授予您使用这些出版物的许可权。

适用性: 用户需要遵循 IBM Web 站点的使用条款及以下条款和条件。

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。

权利: 除非本许可权中明确授予，否则不得授予对这些出版物或其中包含的任何信息、数据、软件或其他知识产权的任何许可权、许可证或权利，无论是明示的还是暗含的。

IBM 保留根据自身的判断，认为对出版物的使用损害了 IBM 的权益（由 IBM 自身确定）或未正确遵循以上指示信息时，撤回此处所授予权限的权利。

只有您完全遵循所有适用的法律和法规，包括所有的美国出口法律和法规，您才可以下载、出口或再出口该信息。

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。

IBM 商标: IBM、 IBM 徽标和 ibm.com 是 International Business Machines Corp., 在全球许多管辖区域注册的商标或注册商标。其他产品和服务名称可能是 IBM 或其他公司的商标。当前的 IBM 商标列表，可从 Web 站点 www.ibm.com/legal/copytrade.shtml获得

附录 B. 声明

本信息是为在美国提供的产品和服务编写的。有关非 IBM 产品的信息是基于首次出版此文档时的可获信息且会随时更新。

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，则由用户自行负责。

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.

有关双字节字符集 (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:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
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。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是此 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

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。此样本程序“按现状”提供，且不附有任何种类的保证。对于使用此样本程序所引起的任何损坏，IBM 将不承担责任。

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:

© (*your company name*) (*year*). 此部分代码是根据 IBM 公司的样本程序衍生出来的。
© Copyright IBM Corp. *_enter the year or years_*. All rights reserved.

商标

IBM 商标: IBM、 IBM 徽标和 ibm.com 是 International Business Machines Corp., 在全球许多管辖区域注册的商标或注册商标。其他产品和服务名称可能是 IBM 或其他公

司的商标。当前的 IBM 商标列表，可从 Web 站点 www.ibm.com/legal/copytrade.shtml 上“版权和商标信息”部分获取。

下列各项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 的商标和徽标是 Oracle 和/或其子公司的商标或注册商标。
- 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、Celeron、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]

[B]

[C]

[D]

[F]

[G]

[H]

[J]

[K]

[L]

[M]

[N]

[P]

[Q]

[R]

[S]

[T]

[W]

[X]

[Y]

[Z]

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

[特别字符]

Printed in China

S151-1758-01

Spine information:

IBM DB2 10.1 for Linux, UNIX, and Windows

数据库管理概念和配置参考