IBM DB2 10.5
for Linux, UNIX, and Windows

分区和集群指南

S151-1996-00

IBM DB2 10.5
for Linux, UNIX, and Windows

分区和集群指南

S151-1996-00

注意

使用此信息及其支持的产品前，请先阅读第 403 页的附录 E, 『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

DB2® 关系数据库管理系统的功能在很大程度上受分区和集群功能影响，这两种功能允许管理员和系统操作员有效地增强数据库性能并将许多数据库对象分布在硬件资源中。更快的数据检索速度、将对象分布在不断增多的硬件资源中的能力以及利用并行性和存储器容量，这些最终会极大地提高效率。本书包含 DB2 数据库库中的一个有组织的主题集，它形成一个丰富的信息源，只重点讨论规划、设计、实施、使用以及数据库分区、表分区、表集群、表范围集群、多维集群表和并行性的维护。

本书适用对象

本书主要面向需要设计、实现或维护将由本地和远程客户机访问的分区或集群数据库的数据库管理员、系统管理员、安全管理员和系统操作员。应用程序开发者和其他用户也可通过本书获得丰富信息源，并从中了解 DB2 关系数据库管理系统的分区、集群和并行性功能的管理和操作方式。对于那些关注我们此处讨论的任何或所有主要功能的将来实施的人来说，本书是难得的参考资料。

本书的结构

DB2 库中的此主题集提供一个丰富的信息源，它只重点讨论 DB2 分区、集群和并行性功能。为了使您能够方便有效地使用本书，它分为六个主要部分，前五个部分表示管理员、系统操作员和应用程序开发者所关心的主要管理主题。本书主要部分中的主题可映射至表示 DB2 库中另一本书的内容的主题，从而使您能够方便地交叉引用更多一般信息，因为它涉及大量其他 DB2 功能和对象。例如，在阅读第 4 部分第 20 章中关于多维集群表的优化策略如何改善性能的主题后，您可能希望通过阅读该特定示例主题映射至的《*调整数据库性能*》一书，来检查常规表上可配置的其他一般性能增强功能。在下面的表 1 中，您将发现本书中映射至其他书籍的主要部分，通过参阅这些书籍，您可以了解关于类似主题包含的其他 DB2 对象和功能的更多信息。

表 1. 本书各部分至 DB2 库中的其他书籍的映射

《分区和集群指南》中的各部分	映射至 DB2 库中的书籍
第 1 部分 规划和设计注意事项	*数据库管理概念和配置参考* *数据库安全性指南*
第 2 部分 安装注意事项	*数据库管理概念和配置参考* *安装 DB2 服务器*
第 3 部分 实施和维护	*数据移动实用程序指南和参考* *数据恢复和高可用性指南与参考* *数据库管理概念和配置参考* *升级到 DB2 V10.5* *数据库监视指南和参考* *XQuery 参考*

表 1. 本书各部分至 DB2 库中的其他书籍的映射 (续)

《分区和集群指南》中的各部分	映射至 DB2 库中的书籍
第 4 部分 性能问题	*数据库管理概念和配置参考* *故障诊断和调整数据库性能*
第 5 部分 管理 API、命令和 SQL 语句	*Administrative API Reference* *Administrative Routines and Views* *Command Reference* *开发 ADO.NET 和 OLE DB 应用程序* *开发嵌入式 SQL 应用程序* *Developing Java Applications* *Developing Perl, PHP, Python, and Ruby on Rails Applications* *开发用户定义的例程（SQL 和外部例程）* *数据库应用程序开发入门* *SQL Reference Volume 1* *SQL Reference Volume 2*
第 6 部分 附录	*数据恢复和高可用性指南与参考* *安装 DB2 服务器* *SQL Reference Volume 1*

本书的各章节中讨论了下列主要主题领域:

第 1 部分 规划和设计注意事项

下列所有章节都包含与规划和设计数据库/表相关的概念性信息，将对这些数据库/表进行分区或集群，或者将它们用于并行数据库系统中。

- 第 1 章『分区数据库和表』引入了关于对数据库和表进行分区的功能和好处的相关概念。
- 第 2 章『范围集群表』提供了关于使用范围集群表的功能和优点的一般概念性信息。
- 第 3 章『多维集群 (MDC) 表』描述了使用多维集群作为对表中数据进行集群的极佳方法。
- 第 4 章『并行数据库系统』描述了如何利用并行性极大地提高性能。

第 2 部分 安装注意事项

下列章节提供了关于在准备数据库分区时需要执行的预安装和安装任务的信息。

- 第 5 章『安装先决条件』描述了与准备分区数据库环境中涉及的 DB2 服务器相关的先决条件和限制。
- 第 6 章『安装之前』讨论了 UNIX 和 Linux 操作系统中的其他预安装任务和注意事项。

- 第 7 章『安装 DB2 服务器产品』描述了如何安装数据库分区服务器和设置分区数据库环境。
- 第 8 章『安装之后』描述了如何验证 Windows、UNIX 和 Linux 操作系统上的安装。

第 3 部分 实施和维护

完成规划、设计和安装步骤后，下列章节讨论了如何实施和维护先前准备工作生成的功能和/或对象。

- 第 9 章『创建数据库之前』描述了在创建数据库之前应考虑的事项，如启用并行性、在创建分区数据库环境、创建和配置数据库分区及建立数据库分区之间的通信之前应考虑的事项。
- 第 10 章『创建和管理分区数据库环境』描述了如何创建和管理数据库分区及分区组。
- 第 11 章『创建表和其他相关表对象』提供了关于如何创建和设置分区表、范围集群表及 MDC 表的信息。
- 第 12 章『改变数据库』描述了如何改变实例和/或数据库。
- 第 13 章『改变表和其他相关表对象』提供了关于如何修改分区表的信息。
- 第 14 章『装入』讨论了在并行性、多维集群和分区表情况下的装入注意事项。
- 第 15 章『在分区数据库环境中装入数据』描述了如何在分区数据库环境中插入、继续、重新启动或终止数据装入操作。
- 第 16 章『分区数据库迁移环境』简要概述了有关的迁移分区数据库和对更多详细信息的引用。
- 第 17 章『使用快照和事件监视器』除了描述如何使用 CREATE EVENT MONITOR 语句外，还提供了关于如何使用快照监视器结果来监视表重组过程或访问分区数据库系统的全局状态的相关信息。
- 第 18 章『开发好的备份和恢复策略』描述了分区数据库环境中的崩溃恢复相关概念，这有助于在出现故障前开发备份和恢复策略。
- 第 19 章『故障诊断』简要概述了故障诊断和有用信息，此信息是关于如何在实例中的所有计算机上或所有数据库分区服务器上发出有助于进行故障诊断的命令（如 **db2trc**）。

第 4 部分 性能问题

下列章节包含允许您提高分区和/或集群环境的性能的相关信息。

- 第 20 章『数据库设计中的性能问题』描述了表分区和多维集群的性能增强功能，包括表分区和多维集群的优化策略。
- 第 21 章『索引』提供有助于了解分区表索引的概念性信息。
- 第 22 章『设计顾问程序』描述了如何使用设计顾问程序来获取关于从单一分区迁移至多分区数据库的信息，以及关于分发数据和创建新索引、具体化查询表及多维集群表的建议。
- 第 23 章『管理并行性』提供了关于锁定方式的信息。
- 第 24 章『代理程序管理』描述了如何优化用于处理应用程序请求的数据库代理程序。

- 第 25 章『优化存取方案』描述了如何改善存取方案以及优化器如何使用通过各种扫描获得的信息来优化数据访问策略，并且包括关于连接策略的信息，所有这些都是为了提高使用并行性的分区数据库环境、集群表和/或系统的性能。
- 第 26 章『数据重新分发』帮助您确定是否应执行数据重新分发，如果应执行，描述如何在数据库分区之间重新分发数据。
- 第 27 章『配置自调整内存』讨论了分区数据库环境中自调整内存功能的使用，并提供了配置建议。
- 第 28 章『DB2 配置参数和变量』提供了关于如何在多个数据库分区间设置数据库配置参数和环境变量的信息，并列示了与分区数据库环境和并行性功能相关的参数和变量。

第 5 部分 管理 API、命令和 SQL 语句

下列章节将与分区数据库环境相关的管理 API、命令和 SQL 元素的信息全部合并在一起。

- 第 29 章『管理 API』提供了仅与分区数据库环境相关的 API 的信息。
- 第 30 章『命令』提供了仅与分区数据库环境相关的命令的信息。
- 第 31 章『SQL 语言元素』提供了与数据库分区兼容的数据类型和专用寄存器。
- 第 32 章『SQL 函数』描述了仅与分区数据库环境相关的 SQL 函数。
- 第 33 章『SQL 语句』描述了仅与分区数据库环境相关的 SQL 语句。
- 第 34 章『受支持的管理 SQL 例程和视图』描述了仅与分区数据库环境相关的 SQL 例程和视图。

第 6 部分 附录

- 附录 A『作为非 root 用户安装』描述了作为非 root 用户在 UNIX 和 Linux 操作系统上安装 DB2 数据库产品的过程。
- 附录 B『使用备份』描述了如何使用 **BACKUP DATABASE** 命令。
- 附录 C『分区数据库环境目录视图』列示了分区数据库环境的特殊目录视图。

突出显示约定

本书中使用了下列突出显示约定。

粗体	指示命令、关键字和其他由系统预定义名称的项。
斜体	指示下列其中一种情况: • 必须由用户提供的名称或值（变量） • 一般强调 • 引入新术语 • 引用其他信息源

等宽字体	指示下列其中一种情况: • 文件和目录 • 指示您应在命令提示符处或窗口中输入的信息 • 特定数据值的示例 • 类似于系统可能显示的文本的文本示例 • 系统消息示例 • 程序代码示例

第 1 部分 规划和设计注意事项

第 1 章 分区数据库和表

设置分区数据库环境

必须在创建数据库之前决定创建多分区数据库。在作出数据库设计决定时，必须确定是否应利用数据库分区可以提供的性能提高。

关于此任务

在分区数据库环境中，仍然使用 **CREATE DATABASE** 命令或 sqlecrea() 函数来创建数据库。无论使用哪种方法，都可以通过 db2nodes.cfg 文件中列示的任何分区来发出请求。db2nodes.cfg 文件是数据库分区服务器配置文件。

除了在 Windows 操作系统环境上之外，可以使用任何编辑器来查看和更新数据库分区服务器配置文件 (db2nodes.cfg) 的内容。在 Windows 操作系统环境上，请使用 **db2ncrt** 和 **db2nchg** 命令来创建和更改数据库分区服务器配置文件

在创建多分区数据库之前，必须选择将作为数据库的目录分区的数据库分区。然后，可以直接从该数据库分区创建数据库，也可以从连接至该数据库分区的远程客户机创建数据库。您要连接并对其执行 **CREATE DATABASE** 命令的数据库分区成为该特定数据库的*目录分区*。

目录分区是用于存储所有系统目录表的数据库分区。对系统表的所有访问都必须通过此数据库分区进行。所有联合数据库对象（例如，包装器、服务器和昵称）都存储在此数据库分区上的系统目录表中。

若可能，应该在独立的实例中创建每个数据库。若不可能做到此点（即，必须在每个实例中创建多个数据库），应该将目录分区分布至可用的数据库分区中。这样做可以减少在单个数据库分区中对目录信息的争用。

注：应该定期备份目录分区，同时，因为其他数据会增加备份所需的时间，所以要避免将用户数据置于该节点上（任何可能的时候）。

当创建数据库时，它会在 db2nodes.cfg 文件中定义的所有数据库分区之间自动创建。

创建系统中的第一个数据库时，就会形成一个系统数据库目录。并追加有关您创建的任何其他数据库的信息。在 UNIX 上工作时，系统数据库目录是 sqldbdir，位于主目录下的 sqllib 目录中或安装 DB2 数据库的目录下面。在 UNIX 上工作时，由于组成分区数据库环境的所有数据库分区只有一个系统数据库目录，所以此目录必须位于共享文件系统（例如，UNIX 平台上的 NFS）上。在 Windows 上工作时，系统数据库目录位于实例目录中。

位于 sqldbdir 目录中的还有系统意向文件。它称为 sqldbins，用于确保数据库分区保持同步。该文件也必须位于共享文件系统中，因为所有数据库分区中只有一个目录。该文件由组成数据库的所有数据库分区共享。

必须修改配置参数，才能利用数据库分区。使用 **GET DATABASE CONFIGURATION** 和 **GET DATABASE MANAGER CONFIGURATION** 命令以了解特定数据库或数据库管理器配置文件中的

个别条目的值。要修改特定数据库或数据库管理器配置文件中的个别条目，可分别使用 **UPDATE DATABASE CONFIGURATION** 和 **UPDATE DATABASE MANAGER CONFIGURATION** 命令。

影响分区数据库环境的数据库管理器配置参数包括 **conn_elapse**、**fcm_num_buffers**、**fcm_num_channels**、**max_connretries**、**max_coordagents**、**max_time_diff**、**num_poolagents** 和 **start_stop_time**。

跨多个数据库分区对数据库进行分区

在将数据分布在分区数据库的多个数据库分区上时，数据库管理器提供了很大的灵活性。

用户可以通过声明分布键来选择分发数据的方式，并且可以通过选择要在其中存储数据的数据库分区组和表空间来确定其表数据可以分发在哪些数据库分区上以及分发在多少个数据库分区上。

此外，可更新的分发映射指定分布键值至数据库分区的映射。这使得可以灵活地将大型表的工作负载并行分布在分区数据库上，并同时允许将较小的表存储在一个或少数数据库分区中（如果应用程序设计者选择这样做）。每个本地数据库分区都可以有它所存储数据的本地索引，以便提供高性能的本地数据访问。

在分区数据库中，分布键用于将表数据分布到一组数据库分区中。也会分配索引数据及其相应的表，并将它们本地存储在每个数据库分区中。

在使用数据库分区存储数据之前，必须对数据库管理器定义这些数据库分区。在名为 `db2nodes.cfg` 的文件中定义数据库分区。

在 CREATE TABLE 语句或 ALTER TABLE 语句中为分区数据库分区组中的表空间中的表指定分布键。如果没有为表指定分布键，那么缺省情况下将根据主键的第一列创建分布键。如果未定义主键，那么缺省分布键是该表中定义的第一个非长型或 LOB 数据类型的列。分区数据库中的表必须至少有一列既非长型也非 LOB 数据类型。只有在显式指定后，单一分区数据库分区组中的表空间中的表才具有分布键。

按如下所示将行放置在数据库分区中:

1. 对所有分布键列应用散列算法（数据库分区函数），这将生成分发映射索引值。
2. 分发映射中该索引值处的数据库分区号标识将存储行的数据库分区。

数据库管理器支持*部分分区*，这意味着可以将表分布到系统中的数据库分区子集上（即，数据库分区组）。不必将表分布在系统中的所有数据库分区上。

在访问用于连接或子查询的数据时，数据库管理器能够识别这些数据是否位于同一数据库分区组中的同一数据库分区上。这称为*表并置*。在并置的表中具有相同分布键值的那些行位于同一数据库分区上。数据库管理器可以选择在存储该数据的数据库分区中执行连接或子查询处理。这样做的优点是可以显著改善性能。

并置的表必须:

- 位于将不重新分发的同一数据库分区组中。（重新分发期间，数据库分区组中的表可能正使用不同的分发映射 - 它们不是并置的。）
- 具有与列数相同的分布键数。

- 具有与数据库分区兼容的相应分布键列。
- 位于在同一数据库分区中定义的单一分区数据库分区组中。

分区数据库认证注意事项

在一个分区数据库中，必须为数据库的每个分区定义同一组用户和组。如果这些定义不相同，那么用户也许是被授权在不同的分区上做不同的事情。

建议所有分区保持一致。

数据库分区组

数据库分区组是已命名的集合，它包含属于某个数据库的一个或多个数据库分区。

包含多个数据库分区的数据库分区组称为*多分区数据库分区组*。多分区数据库分区组只能使用属于相同实例的数据库分区来定义。

图 1 显示了一个具有五个数据库分区的数据库的示例。

- 数据库分区组 1 包含除一个数据库分区外的所有其他分区。
- 数据库分区组 2 包含一个数据库分区。
- 数据库分区组 3 包含两个数据库分区。
- 组 2 中的数据库分区与组 1 共享（并且重叠）。
- 组 3 中的单个数据库分区与组 1 共享（并且重叠）。

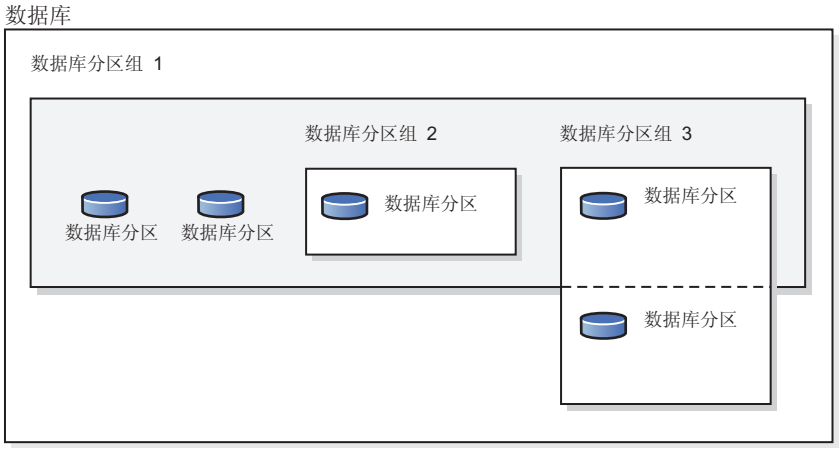


图 1. 数据库中的数据库分区组

创建数据库时，还将创建名为 db2nodes.cfg 的*数据库分区配置文件*中指定的所有数据库分区。可以分别使用 **ADD DBPARTITIONNUM** 或 **DROP DBPARTITIONNUM VERIFY** 命令来添加或除去其他数据库分区。数据将分布在数据库分区组中的所有数据库分区上。

创建数据库分区组时，将使一个*分发映射*与该组相关联。数据库管理器使用分发映射以及*分布键*和散列法算法来确定数据库分区组中的哪个数据库分区将存储给定的数据行。

缺省数据库分区组

创建数据库时将自动定义三个数据库分区组:

- 用于容纳 SYSCATSPACE 表空间的 IBMCATGROUP，它保存系统目录表
- 用于容纳 TEMPSPACE1 表空间的 IBMTEMPGROUP，它保存数据库处理期间创建的临时表
- 用于容纳 USERSPACE1 表空间的 IBMDEFAULTGROUP，它保存用户表和索引。可以在 IBMDEFAULTGROUP 或用户创建的任何数据库分区组中创建已声明或已创建的临时表的用户临时表空间，但不能在 IBMTEMPGROUP 中创建这些表空间。

数据库分区组中的表空间

当表空间与多分区数据库分区组相关联时（在执行 CREATE TABLESPACE 语句期间），将在此数据库分区组中的每个数据库分区上将该表空间内的所有表进行分区。以后不能使与特定数据库分区组相关联的表空间与其他数据库分区组相关联。

创建数据库分区组

使用 CREATE DATABASE PARTITION GROUP 语句来创建数据库分区组。此语句指定将用来存放表空间容器和表数据的一组数据库分区。此语句还会执行下列操作:

- 为数据库分区组创建分发映射。
- 生成分发映射标识。
- 将记录插入下列目录视图中:
  - SYSCAT.DBPARTITIONGROUPDEF
  - SYSCAT.DBPARTITIONGROUPS
  - SYSCAT.PARTITIONMAPS

改变数据库分区组

使用 ALTER DATABASE PARTITION GROUP 语句来将数据库分区添加到数据库分区（或从数据库分区组中删除数据库分区）。在添加或删除数据库分区之后，请使用 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令来在数据库分区组中的一组新数据库分区之间重新分发数据。

数据库分区组设计注意事项

除非要与一个更大的表并置，否则请将小表放在单一分区数据库分区组中。*并置*是将不同表中包含相关数据的行放置在同一个数据库分区中。并置的表有助于数据库管理器使用更高效的连接策略。这样的表可以存在于单一分区数据库分区组中。如果表位于多分区数据库分区组中、在分布键中具有相同数目的列并且对应列的数据类型兼容，那么这些表将视为已并置。在并置的表中具有相同分布键值的行被放置在同一个数据库分区上。这些表可以位于相同数据库分区组中的单独表空间中，且仍被视为是并置的。

请避免将中等大小的表扩展到太多数据库分区上。例如，100 MB 的表在具有 16 个分区的数据库分区组中的性能可能高于在具有 32 个分区的数据库分区组中的性能。

可以使用数据库分区组将联机事务处理 (OLTP) 表与决策支持 (DSS) 表分开。这样将有助于确保不会对 OLTP 事务的性能产生负面影响。

如果您正在使用多分区数据库分区组，请考虑以下几点:

- 在多分区数据库分区组中，如果唯一索引是分布键的超集，那么只能创建唯一索引。
- 必须给每个数据库分区指定唯一的编号，这是因为在一个或多个数据库分区组中，可能会发现相同的数据库分区。
- 要确保快速恢复包含系统目录表的数据库分区，应避免将用户表放置在同一个数据库分区上。请将用户表放置在不包括 IBMCATGROUP 数据库分区组中的那些数据库分区的数据库分区组中。

分发映射

在分区数据库环境中，数据库管理器必须知道到哪里去查找它需要的数据。数据库管理器使用一个称为*分发映射*的映射来查找数据。

分发映射是一个内部生成的数组，对于多分区数据库分区组，它包含 32 768 个条目，对于单一分区数据库分区组，它只包含一个条目。对于单一分区数据库分区组，分发映射只有一个条目，该条目包含存储数据库表的所有行的数据库分区的编号。对于多分区数据库分区组，数据库分区组的编号指定方式使得能够一个接一个地使用每个数据库分区，以确保整个映射分发均匀。正如使用网格将城市地图划分为区一样，数据库管理器使用*分布键*来确定存储数据的位置（数据库分区）。

例如，假设您有一个位于四个数据库分区（编号为 0 到 3）上的数据库。此数据库的 IBMDEFAULTGROUP 数据库分区组的分发映射是:

```
0 1 2 3 0 1 2 ...
```

如果已使用数据库分区 1 和 2 在该数据库中创建了一个数据库分区组，那么该数据库分区组的分发映射是:

```
1 2 1 2 1 2 1 ...
```

如果要装入到数据库的表的分布键是可能值在 1 与 500 000 之间的整数，那么会将该分布键散列为 0 与 32 767 之间的编号。将该编号用作分发映射中的索引，以选择用于该行的数据库分区。

第 8 页的图 2 显示如何将具有分布键值 (c1, c2, c3) 的行映射至编号 2，然后引用数据库分区 n5。

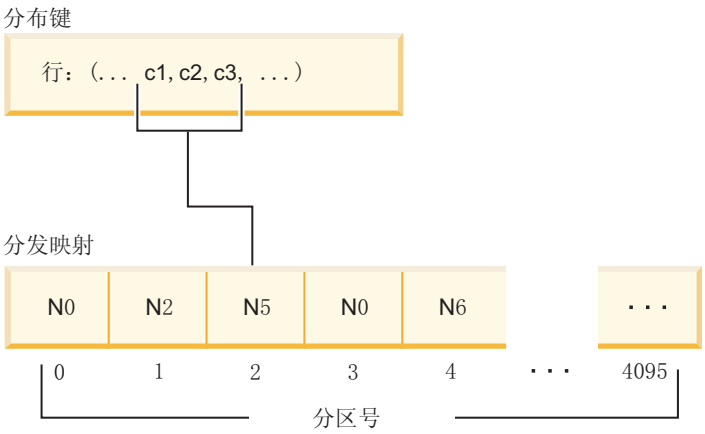


图 2. 使用分发映射的数据分布

分发映射可以灵活地控制将数据存储在多分区数据库中的哪个位置。如果必须更改数据库中各数据库分区上的数据分布，那么可以使用数据重新分发实用程序。此实用程序允许重新平衡或调整数据分布的偏差。

可以使用 db2GetDistMap API 来获取可查看的分发映射的副本。如果继续使用 sqlugtpi API 来获取分发信息，那么此 API 可能返回错误消息 SQL2768N，因为它只能检索到包含 4096 个条目的分发映射。

分布键

*分布键*是一列（或一组列），用于确定存储特定数据行的数据库分区。

分布键是使用 CREATE TABLE 语句在表上定义的。如果没有为分布在数据库分区组中的多个数据库分区中的表空间中的表定义分布键，在缺省情况下将会根据主键的第一列创建分布键。

如果未指定主键，那么缺省分布键是在该表中定义的第一个非长字段列。（*长型*包括所有长型数据类型和所有大对象 (LOB) 数据类型）。如果正在与单一分区数据库分区组关联的表空间中创建表，且您希望有分布键，那么必须显式定义该分布键。缺省情况下，不会创建该分布键。

如果没有列满足缺省分布键的要求，那么会创建不带分布键的表。仅允许没有分布键的表存在于单一分区数据库分区组中。以后可以使用 ALTER TABLE 语句来添加或删除分布键。仅可对其表空间与单一分区数据库分区组相关的表改变分布键。

选择好的分布键很重要。请考虑下列各项:

- 如何访问表
- 查询工作负载的性质
- 数据库系统所使用的连接策略

如果并置不是主要的注意事项，那么好的表分布键就是将数据均匀分布在数据库分区组中的所有数据库分区上的分布键。与数据库分区组相关联的表空间中每个表的分布键确定这些表是否是并置的。下列情况中，表被认为是并置的:

- 表被放置在同一数据库分区组中的表空间内
- 每个表中的分布键具有相同数量的列

- 对应列的数据类型是分区兼容的

这些特征确保并置的表中具有相同分布键值的那些行位于同一个数据库分区上。

不适当的分布键会导致数据分布不均匀。请勿选择数据分布不均匀的列或含有少数单值的列作为分布键。单值的数目必须足够大，才能确保将行均匀分布在数据库分区组中的所有数据库分区上。应用分发算法的成本与分布键的大小是成正比的。分布键不能超过 16 列，而且列越少，性能越好。请勿在分布键中包括不必要的列。

定义分布键时，请考虑以下事项:

- 不支持创建只包含 BLOB、CLOB、DBCLOB、LONG VARCHAR、LONG VARGRAPHIC、XML 或结构化数据类型的多分区表。
- 不能改变分布键定义。
- 在分布键中包括连接最频繁的列。
- 在分布键中包括经常用于 GROUP BY 子句的列。
- 任何唯一键或主键必须包含所有分布键列。
- 在联机事务处理 (OLTP) 环境中，确保分布键中的所有列都通过等价谓词参与事务。例如，假定有一个在事务中经常使用的职员号列 EMP_NO，如:

```
UPDATE emp_table SET ... WHERE
emp_no = host-variable
```

  在此情况下，EMP_NO 列对于 EMP_TABLE 而言是一个不错的单列分布键。

*数据库分区*是确定表中每一行的位置的方法。该方法的原理如下:

1. 将散列算法应用于分布键的值，并生成一个介于零（0）与 32767 之间的编号。
2. 创建数据库分区组时将创建分发映射。每个编号依次以循环方式重复，以填写该分发映射。
3. 将该编号用作分发映射中的索引。分发映射中该位置处的编号是存储该行的数据库分区的编号。

表并置

如果两个或多个表在对特定查询的响应中频繁地提供数据，那么您将希望这些表中的相关数据的物理位置尽可能地接近。在分区数据库环境中，这个进程被称为*表并置*。

当表存储在同一个数据库分区组中且它们的分布键兼容时，这些表就是并置的。将两个表置于同一个数据库分区组中以确保存在公共的分发映射。这些表可能位于不同的表空间，但是这些表空间必须与同一数据库分区组相关联。每个分布键中对应列的数据类型必须是*分区兼容*的。

当访问用于连接或子查询的多个表时，数据库管理器会确定要连接的数据是否位于同一数据库分区上。当发生这种情况时，将在存储数据的数据库分区上执行连接或子查询，而不必在数据库分区之间移动数据。此功能的优点是可以显著改善性能。

分区兼容性

对分布键的对应列的基本数据类型进行比较，并可将它们声明为*分区兼容*。分区兼容的数据类型具有如下属性: 具有相同值但不同类型的两个变量会按相同的分区算法映射至同一个编号。

分区兼容性具有下列特征:

- 基本数据类型与另一个相同的基本数据类型兼容。
- 内部格式用于 DATE、TIME 和 TIMESTAMP 数据类型。它们彼此都不兼容，且与字符或图形数据类型不兼容。
- 分区兼容性不受列的可空性影响。
- 分区兼容性受整理影响。除了忽略整理的强度（S）属性，区分语言环境的基于 UCA 的整理在整理时需要完全匹配。为确定分区兼容性，所有其他整理被视为等价。
- 仅当使用的整理并非区分语言环境的基于 UCA 的整理时，使用 FOR BIT DATA 定义的字符列才与未使用 FOR BIT DATA 定义的字符列兼容。
- 对兼容性数据类型的 NULL 值的处理是完全相同的；对非兼容性数据类型的 NULL 值的处理可能不相同。
- 用户定义的类型的基本数据类型用于分析分区兼容性。
- 对分布键中值相同的小数的处理是完全相同的，即使它们的小数位和精度不同也是如此。
- 字符串中（CHAR、VARCHAR GRAPHIC 或 VARGRAPHIC）的尾部空格会被散列算法忽略。
- BIGINT、SMALLINT 和 INTEGER 是兼容的数据类型。
- 使用区分语言环境的基于 UCA 的整理时，CHAR、VARCHAR、GRAPHIC 和 VARGRAPHIC 是兼容的数据类型。使用另一整理时，不同长度的 CHAR 和 VARCHAR 是兼容类型，而 GRAPHIC 和 VARGRAPHIC 是兼容类型，但 CHAR 和 VARCHAR 与 GRAPHIC 和 VARGRAPHIC 不兼容。
- 分区兼容性不适用于 LONG VARCHAR、LONG VARGRAPHIC、CLOB、DBCLOB 和 BLOB 数据类型，因为不支持它们作为分布键。

分区表

分区表使用了数据组织方案，在该数据组织方案中，根据该表中一个或多个表分区键列中的值将表数据分布到多个存储对象（称为*数据分区*或*范围*）中。

数据分区或范围是一个表的一部分，它包含该表的一部分行，并且与其他行集分开存储。根据 CREATE TABLE 语句的 PARTITION BY 子句中指定的内容，给定表的数据被划分到多个数据分区或范围中。这些数据分区或范围可位于不同表空间和/或同一表空间中。如果一个表是使用 PARTITION BY 子句创建的，那么该表是分区表。

指定的所有表空间都必须具有相同的页大小、扩展数据块大小、存储机制（DMS 或 SMS）和类型（REGULAR 或 LARGE），并且所有表空间都必须位于同一数据库分区组中。

分区表能够简化表数据的转入和转出，并且，与普通的表相比，分区表包含的数据可以多得多。分区表最多可以有 32,767 个数据分区。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。

分区表的索引可以是分区或非分区索引。在单个分区表上，非分区索引和分区索引可以同时存在。

限制

不支持在分区表中使用分区的分层表或临时表、范围集群表和分区视图。

表分区

表分区是一种数据组织方案，在此方案中，表数据根据一个或多个表列中的值划分到多个称为*数据分区*的存储对象中。每个数据分区都是单独存储的。这些存储对象可以在不同的表空间中，也可以在相同表空间中。

存储对象的行为与单个表的行为非常类似，它通过使用 ALTER TABLE ... ATTACH 语句将现有表合并到分区表中，可以很容易实现快速转入。同样，使用 ALTER TABLE ... DETACH 语句很容易实现转出。查询处理同样可以利用分离的数据来避免扫描不相关数据，从而使许多数据仓库样式查询具有更好地查询性能。

按照 CREATE TABLE 语句的 PARTITION BY 子句中指定那样将表数据分区。此定义中使用的列被称为表分区键列。

这种组织方案可以单独地使用，也可与其他组织方案结合使用。通过组合使用 CREATE TABLE 语句的 DISTRIBUTE BY 和 PARTITION BY 子句，可以将数据分布到跨多个表空间的数据库分区。该组织方案包括:

- DISTRIBUTE BY HASH
- PARTITION BY RANGE
- ORGANIZE BY DIMENSIONS

表分区可能用于 DB2 V9.1 Enterprise Server Edition for Linux, UNIX, and Windows 和更高版本。

表分区的优点

如果下列任何情况适用于您和您的组织，请考虑表分区的许多优点:

- 数据仓库因为更容易转入和转出表数据而受益。
- 数据仓库包括大型表。
- 您将考虑从先前发行版或某个具竞争力的数据库产品迁移至 V9.1 数据库
- 您想要更有效地使用分层存储管理 (HSM) 解决方案。

表分区简化了表数据转入和转出以及管理工作，并且提高了索引位置的灵活性和查询处理效率。

有效转入和转出

表分区功能提高了表数据的转入和转出效率。可通过使用 ALTER TABLE 语句的 ATTACH PARTITION 和 DETACH PARTITION 子句来实现此效率。通过转入分区表数据，可以方便地将新范围作为附加数据分区合并到分区表中。通过转出分区表数据，可轻松地从分区表中分离出某些范围的数据，以进行后续清除或归档处理。

借助 DB2 V9.7 FP1 和更高发行版，在使用带 DETACH PARTITION 子句的 ALTER TABLE 语句从分区表拆离数据分区时，源分区表仍然可供在 RS、CS 或 UR 隔离级别下运行的动态查询访问。同样，使用带有 ATTACH PARTITION 子句的 ALTER TABLE 语句将数据分区连接到分区表时，目标分区表仍然可供在 RS、CS 或 UR 隔离级别下运行的动态查询访问。

更容易管理大型表

由于您可以在各个数据分区上执行管理任务，因此表级别管理更灵活。这些任务包括：拆离和重新连接数据分区、备份和复原各个数据分区以及重组各个索引。通过将花费较长时间的维护操作分解成一系列较小的操作，可以缩短这种维护操作的执行时间。例如，将数据分区放置在单独的表空间中后，备份操作可以逐个处理数据分区。因此，可以一次备份分区表的一个数据分区。

灵活的索引位置

现在，可以将索引放置在不同的表空间中，从而允许对索引位置进行更精细地控制。此设计具有以下一些好处：

- 删除索引和创建联机索引时改进性能。
- 能够针对每个表索引之间的任何表空间特征使用不同的值（例如，为了确保更好的空间利用率，对每个索引使用不同的页大小可能更合适）。
- 减少 I/O 争用，这有助于更高效地对表索引数据进行并行访问。
- 删除各个索引时，空间立即对系统可用，不需要索引重组。
- 如果您选择执行索引重组，可以重组单个索引。

DMS 和 SMS 表空间都支持在不同于表的另一个位置使用索引。

提高了商业智能样式查询的性能

增强了查询处理功能，能够根据查询谓词自动消除某些数据分区。此查询处理称为*数据分区消除*，可为许多决策支持查询带来好处。

下列示例创建了一个名为 CUSTOMER 的表，其中包含 l_shipdate >= '01/01/2006' 和 l_shipdate <= '03/31/2006' 的行存储在表空间 TS1 中，包含 l_shipdate >= '04/01/2006' 和 l_shipdate <= '06/30/2006' 的行存储在表空间 TS2 中，等等。

```
CREATE TABLE customer (l_shipdate DATE, l_name CHAR(30))
IN ts1, ts2, ts3, ts4, ts5
PARTITION BY RANGE(l_shipdate) (STARTING FROM ('01/01/2006')
ENDING AT ('12/31/2006') EVERY (3 MONTHS))
```

数据分区和范围

分区表使用数据组织方案，在此方案中，表数据根据表的一个或多个表分区键列值划分到多个称为*数据分区*的存储对象中。可以自动生成对每个数据分区指定的范围，也可以在创建表时手动生成这些范围。

数据分区在整个 DB2 库中以多种方式被引用。下表列示了最常见的引用方式：

- DATAPARTITIONNAME 是创建给定的表时对数据分区指定的永久名称。此列值存储在 SYSCAT.DATAPARTITIONS 目录视图中。连接或拆离操作不保留此名称。
- DATAPARTITIONID 是创建给定的表时对数据分区指定的永久标识。该标识用于唯一地标识给定表中的特定数据分区。连接或拆离操作不保留此标识。此值由系统生成，可能出现在各个实用程序的输出中。
- SEQNO 指示特定数据分区范围相对于表中其他数据分区范围的顺序，其中拆离的数据分区排在所有可视的数据分区和连接的数据分区之后。

数据组织方案

引入表分区之后，DB2 数据库提供了一种三级数据组织方案。CREATE TABLE 语句中有三个子句，这些子句包含指示要如何组织数据的算法。

下列三个子句演示了可以任意组合在一起使用的数据组织级别:

- DISTRIBUTE BY 用于将数据均匀地分布在数据库分区上（以启用查询内并行性和平衡每个数据库分区上的负载）（数据库分区）
- PARTITION BY 用于对同一个数据分区中具有类似单维值的行进行分组（表分区）
- ORGANIZE BY 用于对同一表扩展数据块中多个维上具有类似值的行进行分组（多维集群），或根据插入操作的时间对行进行分组（插入时间集群表）。

此语法允许子句之间保持一致并允许进一步的数据组织算法。每个子句可以单独地使用，也可与其他子句结合使用。通过组合使用 CREATE TABLE 语句的 DISTRIBUTE BY 和 PARTITION BY 子句，可以在跨多个表空间的数据库分区之间分发数据。此方法允许类似于 Informix® Dynamic Server 和 Informix Extended Parallel Server 混合功能的行为。

在一个表中，可以组合每个数据组织方案中使用的子句来创建更先进的分区方案。例如，分区数据库环境不仅兼容，而且是对表分区的补充。

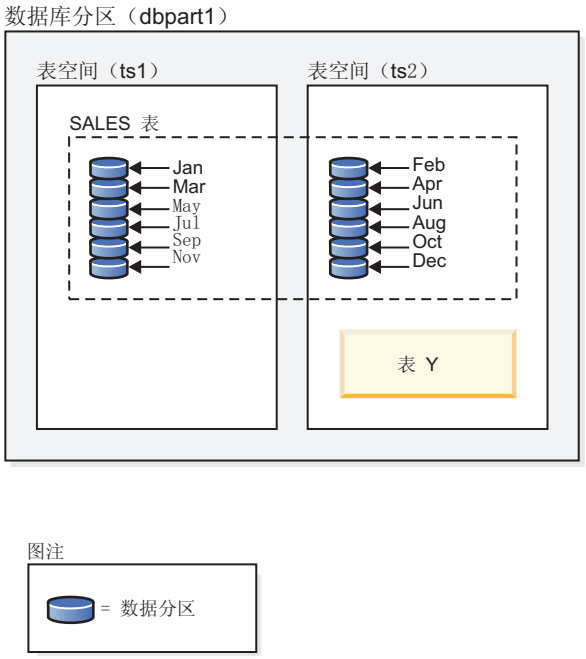


图 3. 演示表分区组织方案，其中表示月销售数据的表划分到多个数据分区中。该表还跨越两个表空间（ts1 和 ts2）。

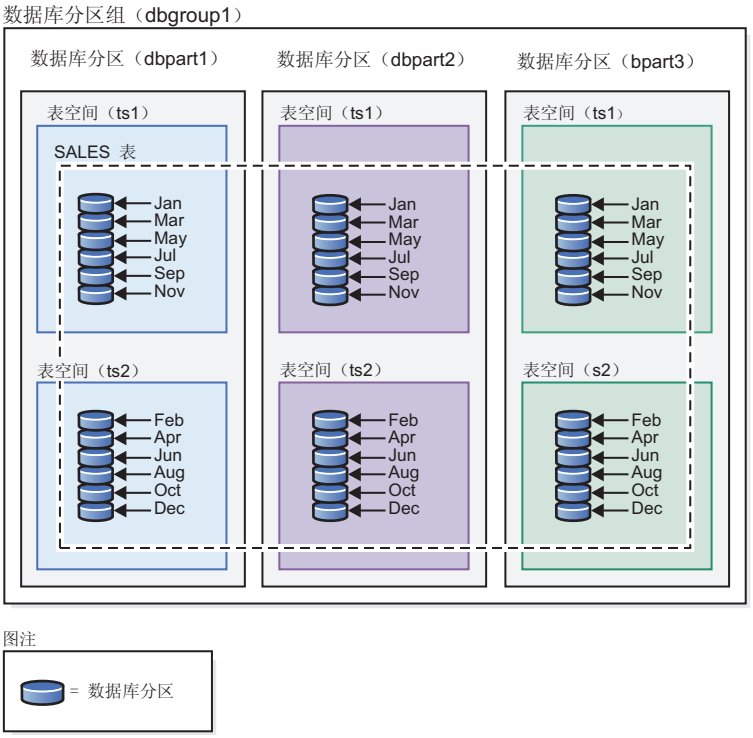


图 4. 演示数据库分区与表分区相互补充的组织方案。表示月销售数据的表划分到多个数据分区中并且跨两个表空间（ts1 和 ts2），而这两个表空间又分布在数据库分区组 (dbgroup1) 的多个数据库分区（dbpart1、dbpart2 和 dbpart3）中。

多维集群 (MDC) 与表分区之间的显著区别是多维和单维。MDC 适合于立方体（即，具有多维的表），而当有一个维是数据库设计的中心时（例如，DATE 列），表分区就能很好地起作用。当同时符合这两个条件时，MDC 和表分区互补。这在第 15 页的图 5 中进行了演示。

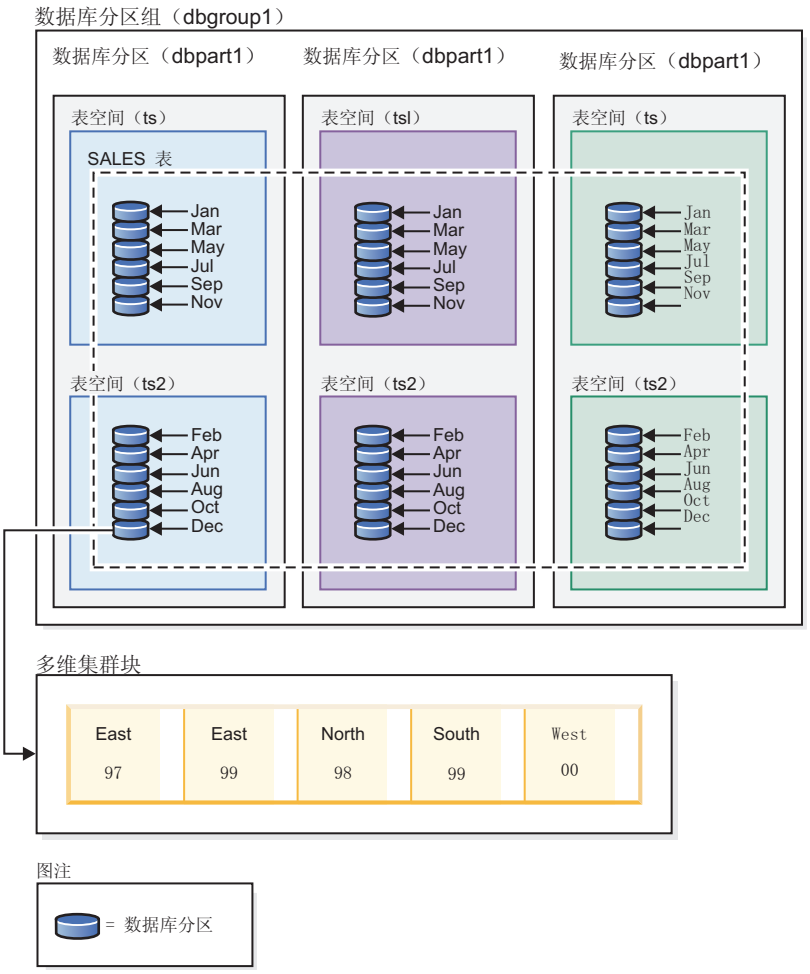


图 5. 数据库分区、表分区和多维组织方案的表示法，其中 SALES 表中的数据不仅分布在多个数据库分区中、划分到表空间 ts1 和 ts2 上，而且还对 date 和 region 维上具有类似值的行进行分组。

还有另一种数据组织方案，它不能与先前列示的任何方案配合使用。此方案是 ORGANIZE BY KEY SEQUENCE。它用于将每个记录插入到在创建表（范围集群的表）时为该记录保留的行中。

数据组织术语

数据库分区 (Database partitioning)
一种数据组织方案，即，表数据根据该表中的一个或多个分布键列中的散列值以及使用的数据库分区的分发映射分布到多个数据库分区中。给定表的数据根据 CREATE TABLE 语句的 DISTRIBUTE BY HASH 子句中指定的内容进行分布。

数据库分区 (Database partition)
数据库分区服务器上的一部分数据库，它由自己的用户数据、索引、配置文件和事务日志组成。数据库分区可以是逻辑或物理的。

表分区 一种数据组织方案，即，表数据根据该表中一个或多个分区列中的值分布到多个数据分区中。根据 CREATE TABLE 语句的 PARTITION BY 子句中指定的内容，给定表的数据被划分到多个存储对象中。这些存储对象可以在不同的表空间中。

数据分区 (Data partition)

表的一部分行，这些行不与其他部分的行存储在一起，并且按照 CREATE TABLE 语句的 PARTITION BY RANGE 子句中指定的内容分组。

多维集群 (MDC)

一个表，其数据按 ORGANIZE BY DIMENSIONS 子句中指定的一个或多个维或者集群键以物理方式组织成块。

插入时间集群 (ITC)

一个表，其数据将根据由 ORGANIZE BY INSERT 子句所指定的行插入时间来实际进行集群。

每种数据组织方案的优势

了解每种数据组织方案的优势可以帮助您在规划、设计或重新评估数据库系统要求时确定最佳方法。表 2 提供了普通客户要求的高级视图，并显示了各种数据组织方案可以如何帮助您满足这些要求。

表 2. 将表分区与数据库分区功能配合使用

问题	建议的方案	说明
数据转出	表分区	使用拆离来转出大量数据，并且只出现最少中断。
并行查询执行（查询性能）	数据库分区功能	提供查询并行性以改善查询性能
数据分区消除（查询性能）	表分区	提供数据分区消除以改善查询性能
获得最佳查询性能	两者	一起使用时可以获得最佳查询性能：查询并行性和数据分区消除互补
管理员的工作负载较重	数据库分区功能	为每个数据库分区执行许多任务

表 3. 将表分区与 MDC 表配合使用

问题	建议的方案	说明
转出期间的数据可用性	表分区	使用 DETACH PARTITION 子句来转出大量数据，并且只出现最少中断。
查询性能	两者	MDC 最适合用来查询多个维。表分区通过数据分区消除提高性能。
最少重组	MDC	MDC 维护集群，从而减少进行重组的必要性。

注: 对于 UNION ALL 视图，目前建议使用表分区。

DB2 和 Informix 数据库中的数据组织方案

表分区是一种数据组织方案，在此方案中，表数据根据一个或多个表列中的值划分到多个称为*数据分区*的存储对象中。每个数据分区都是单独存储的。这些存储对象可以在不同的表空间中，也可以在相同表空间中。

按照 CREATE TABLE 语句的 PARTITION BY 子句中指定那样将表数据分区。此定义中使用的列被称为表分区键列。DB2 表分区功能与 Informix Dynamic Server 和 Informix Extended Parallel Server 提供的数据分段组织方法相对应。

Informix 方法

Informix 支持若干种数据组织方案，在 Informix 产品中，这些方案称为*分段*。其中一种较常使用的分段类型是 FRAGMENT BY EXPRESSION。这种类型的分段的工作方式与 CASE 语句非常相似，其中有一个与表的每个分段相关联的表达式。检查这些表达式以便确定行的放置位置。

Informix 与 DB2 数据库系统的比较

DB2 数据库提供了丰富的补充功能，这些功能与 Informix 数据组织方案直接对应，这使客户能够相对容易地从 Informix 语法转换为 DB2 语法。DB2 数据库管理器将生成列与 CREATE TABLE 语句的 PARTITION BY RANGE 子句配合使用，以处理复杂的 Informix 方案。表 4 对 Informix 和 DB2 数据库产品中使用的数据组织方案作了比较。

表 4. 所有 Informix 与 DB2 数据组织方案的映射

数据组织方案	Informix 语法	DB2 V9.1 语法
• Informix: 基于表达式 • DB2: 表分区	FRAGMENT BY EXPRESSION	PARTITION BY RANGE
• Informix: 循环法 • DB2: 缺省	FRAGMENT BY ROUND ROBIN	没有语法: DB2 数据库管理器自动在容器之间分发数据
• Informix: 范围分布 • DB2: 表分区	FRAGMENT BY RANGE	PARTITION BY RANGE
• Informix: 系统定义的散列 • DB2: 数据库分区	FRAGMENT BY HASH	DISTRIBUTE BY HASH
• Informix: HYBRID • DB2: 在进行表分区的情况下进行数据库分区	FRAGMENT BY HYBRID	DISTRIBUTE BY HASH 和 PARTITION BY RANGE
• Informix: 不适用 • DB2: 多维集群	不适用	ORGANIZE BY DIMENSIONS

示例

下列示例详细说明了在 DB2 数据库中如何实现与任何使用表达式的 Informix 分段方案等同的结果。

示例 1: 下面这个基本的 Create Table 语句显示了 Informix 分段以及等同的 DB2 数据库系统表分区语法:

Informix 语法:

```
CREATE TABLE demo(a INT) FRAGMENT BY EXPRESSION
 a = 1 IN db1,
 a = 2 IN db2,
 a = 3 IN db3;
```

DB2 语法:

```
CREATE TABLE demo(a INT) PARTITION BY RANGE(a)
 (STARTING(1) IN db1,
 STARTING(2) IN db2,
 STARTING(3) ENDING(3) IN db3);
```

Informix XPS 支持称为 hybrid 的两层分段方案，在此方案中，使用一个表达式来在协作服务器之间分发数据，并使用第二个表达式来在协作服务器内分发数据。这使所有协作服务器都能够参与查询（即，在所有协作服务器上都有数据），并使查询能够利用数据分区消除功能的优势。

通过结合使用 CREATE TABLE 语句的 DISTRIBUTE BY 和 PARTITION BY 子句，DB2 数据库系统实现了与 Informix hybrid 等同的组织方案。

示例 2: 以下示例显示了组合子句的语法:

Informix 语法

```
CREATE TABLE demo(a INT, b INT) FRAGMENT BY HYBRID HASH(a)
 EXPRESSION b = 1 IN dbsl1,
   b = 2 IN dbsl2;
```

DB2 语法

```
CREATE TABLE demo(a INT, b INT) IN dbsl1, dbsl2
 DISTRIBUTE BY HASH(a),
 PARTITION BY RANGE(b) (STARTING 1 ENDING 2 EVERY 1);
```

此外，可以使用多维集群来进一步组织数据:

```
CREATE TABLE demo(a INT, b INT, c INT) IN dbsl1, dbsl2
 DISTRIBUTE BY HASH(a),
 PARTITION BY RANGE(b) (STARTING 1 ENDING 2 EVERY 1)
 ORGANIZE BY DIMENSIONS(c);
```

这样，列 **a** 值相同的所有行都在同一个数据库分区中。所有列 **b** 值相同的行都在同一个表空间中。对于具有给定的 **a** 和 **b** 值的行，会将再具有相同 **c** 值的所有行集中到一个磁盘上。此方法非常适合于 OLAP 类型的下寻操作，这是因为，仅需扫描单个数据库分区上单个表空间中的一个或数个扩展数据块就可以满足此类查询。

应用表分区以解决常见的应用程序问题

下列各节讨论如何应用各种 DB2 表分区功能以解决常见的应用程序问题。每一节都特别侧重于采取最佳措施来将各种 Informix 分段方案映射到等同的 DB2 表分区方案。

创建简单数据分区范围时的注意事项

其中一种最常见的表分区应用是根据日期键对大型事实表进行分区。如果需要创建大小统一的日期范围，请考虑使用自动生成的 CREATE TABLE 语法格式。

示例

示例 1: 以下示例显示自动生成的语法格式:

```
CREATE TABLE orders
(
 l_orderkey  DECIMAL(10,0) NOT NULL,
 l_partkey  INTEGER,
 l_suppkey  INTEGER,
```

```
 l_linenumber  INTEGER,
 l_quantity  DECIMAL(12,2),
 l_extendedprice  DECIMAL(12,2),
 l_discount  DECIMAL(12,2),
 l_tax   DECIMAL(12,2),
 l_returnflag  CHAR(1),
 l_linestatus  CHAR(1),
 l_shipdate  DATE,
 l_commitdate  DATE,
 l_receiptdate  DATE,
 l_shipinstruct  CHAR(25),
 l_shipmode  CHAR(10),
 l_comment  VARCHAR(44))
  PARTITION BY RANGE(l_shipdate)
  (STARTING '1/1/1992' ENDING '12/31/1993' EVERY 1 MONTH);
```

这将创建 24 个范围，即对 1992-1993 的每个月创建一个范围。尝试插入 l_shipdate 超出该范围的行将导致错误。

示例 2: 将上一示例与以下 Informix 语法作比较:

```
create table orders
(
 l_orderkey  decimal(10,0) not null,
 l_partkey  integer,
 l_suppkey  integer,
 l_linenumber integer,
 l_quantity  decimal(12,2),
 l_extendedprice decimal(12,2),
 l_discount  decimal(12,2),
 l_tax   decimal(12,2),
 l_returnflag char(1),
 l_linestatus char(1),
 l_shipdate  date,
 l_commitdate date,
 l_receiptdate date,
 l_shipinstruct char(25),
 l_shipmode  char(10),
 l_comment  varchar(44)
) fragment by expression
l_shipdate < '1992-02-01' in ldbs1,
l_shipdate >= '1992-02-01' and l_shipdate < '1992-03-01' in ldbs2,
l_shipdate >= '1992-03-01' and l_shipdate < '1992-04-01' in ldbs3,
l_shipdate >= '1992-04-01' and l_shipdate < '1992-05-01' in ldbs4,
l_shipdate >= '1992-05-01' and l_shipdate < '1992-06-01' in ldbs5,
l_shipdate >= '1992-06-01' and l_shipdate < '1992-07-01' in ldbs6,
l_shipdate >= '1992-07-01' and l_shipdate < '1992-08-01' in ldbs7,
l_shipdate >= '1992-08-01' and l_shipdate < '1992-09-01' in ldbs8,
l_shipdate >= '1992-09-01' and l_shipdate < '1992-10-01' in ldbs9,
l_shipdate >= '1992-10-01' and l_shipdate < '1992-11-01' in ldbs10,
l_shipdate >= '1992-11-01' and l_shipdate < '1992-12-01' in ldbs11,
l_shipdate >= '1992-12-01' and l_shipdate < '1993-01-01' in ldbs12,
l_shipdate >= '1993-01-01' and l_shipdate < '1993-02-01' in ldbs13,
l_shipdate >= '1993-02-01' and l_shipdate < '1993-03-01' in ldbs14,
l_shipdate >= '1993-03-01' and l_shipdate < '1993-04-01' in ldbs15,
l_shipdate >= '1993-04-01' and l_shipdate < '1993-05-01' in ldbs16,
l_shipdate >= '1993-05-01' and l_shipdate < '1993-06-01' in ldbs17,
l_shipdate >= '1993-06-01' and l_shipdate < '1993-07-01' in ldbs18,
l_shipdate >= '1993-07-01' and l_shipdate < '1993-08-01' in ldbs19,
l_shipdate >= '1993-08-01' and l_shipdate < '1993-09-01' in ldbs20,
l_shipdate >= '1993-09-01' and l_shipdate < '1993-10-01' in ldbs21,
l_shipdate >= '1993-10-01' and l_shipdate < '1993-11-01' in ldbs22,
l_shipdate >= '1993-11-01' and l_shipdate < '1993-12-01' in ldbs23,
l_shipdate >= '1993-12-01' and l_shipdate < '1994-01-01' in ldbs24,
l_shipdate >= '1994-01-01' in ldbs25;
```

注意，Informix 语法提供了上下不封顶的范围以捕获预期范围外的日期。通过添加使用 MINVALUE 和 MAXVALUE 的范围，可以将 DB2 语法修改为与 Informix 语法匹配。

示例 3: 以下示例将示例 1 修改为与 Informix 语法匹配:

```
CREATE TABLE orders
(
 l_orderkey  DECIMAL(10,0) NOT NULL,
 l_partkey  INTEGER,
 l_suppkey  INTEGER,
 l_linenumber INTEGER,
 l_quantity  DECIMAL(12,2),
 l_extendedprice DECIMAL(12,2),
 l_discount  DECIMAL(12,2),
 l_tax   DECIMAL(12,2),
 l_returnflag CHAR(1),
 l_linestatus CHAR(1),
 l_shipdate  DATE,
 l_commitdate DATE,
 l_receiptdate DATE,
 l_shipinstruct CHAR(25),
 l_shipmode  CHAR(10),
 l_comment  VARCHAR(44)
) PARTITION BY RANGE(l_shipdate)
 (STARTING MINVALUE,
  STARTING '1/1/1992' ENDING '12/31/1993' EVERY 1 MONTH,
  ENDING MAXVALUE);
```

这种技术允许将任何日期插入到表中。

使用生成列按表达式进行分区

虽然 DB2 数据库并不直接支持按表达式进行分区，但支持按生成列进行分区，因此有可能获得相同的结果。

在决定是否使用此方法前，请考虑下列用法准则:

- 生成列是真实的列，它占用物理磁盘空间。使用生成列的表会略微变大。
- 对于在分区表进行分区时所基于的列，不能改变其生成列表达式。尝试执行此操作将产生消息 SQL0190。如果按下一节描述的方式将新数据分区添加到使用生成列的表中，通常要求改变定义生成列的表达式。目前，不支持改变定义生成列的表达式。
- 当表使用生成列时，对于何时可以应用数据分区消除是有限制的。

示例

示例 1: 以下示例使用 Informix 语法，在此情况下适合使用生成列。在本示例中，进行分区时所基于的列存放了加拿大的省和地域。由于省列表不大可能更改，因此生成列表达式也不大可能更改。

```
CREATE TABLE customer (
 cust_id  INT,
 cust_prov CHAR(2))
 FRAGMENT BY EXPRESSION
  cust_prov = "AB" IN dbspace_ab
  cust_prov = "BC" IN dbspace_bc
  cust_prov = "MB" IN dbspace_mb
  ...
  cust_prov = "YT" IN dbspace_yt
  REMAINDER IN dbspace_remainder;
```

示例 2: 在本示例中，使用生成列对 DB2 表进行分区:

```
CREATE TABLE customer (
 cust_id     INT,
 cust_prov   CHAR(2),
 cust_prov_gen GENERATED ALWAYS AS (CASE
  WHEN cust_prov = 'AB' THEN 1
  WHEN cust_prov = 'BC' THEN 2
  WHEN cust_prov = 'MB' THEN 3
  ...
  WHEN cust_prov = 'YT' THEN 13
  ELSE 14 END))
 IN tbspace_ab, tbspace_bc, tbspace_mb, .... tbspace_remainder
 PARTITION BY RANGE (cust_prov_gen)
  (STARTING 1 ENDING 14 EVERY 1);
```

这里，CASE 语句中的表达式与 FRAGMENT BY EXPRESSION 子句中的相应表达式匹配。CASE 语句将每个原始表达式映射到一个数字，该数字存储在生成列（在本示例中是 cust_prov_gen）中。此列是存储在磁盘上的真实列，因此，表占用的空间量会比 DB2 通过表达式直接支持的分区所必需的空间量略多。本示例使用短语法格式。因此，必须在 CREATE TABLE 语句的 IN 子句中列示用来放置数据分区的表空间。如果使用长语法格式，那么每个数据分区都需要不同的 IN 子句。

注: 这种技术可以应用于任何 FRAGMENT BY EXPRESSION 子句。

表分区键

表分区键是一个或多个表列的有序集合。表分区键列中的值用来确定每个表行所属的数据分区。

要对表定义表分区键，请使用指定了 PARTITION BY 子句的 CREATE TABLE 语句。

选择有效的表分区键列对于充分利用表分区功能的优点来说十分关键。下列准则可以帮助您为分区表选择最有效的表分区键列。

- 将范围详细程度定义为与数据转出相匹配。最常见的情况是使用星期、月份或季度。
- 将范围定义成与数据转入大小相匹配。最常见的情况是根据日期或时间列对数据进行分区。
- 根据有益于消除分区的列进行分区。

支持的数据类型

表 5 显示了支持用作表分区键列的数据类型（包括同义词）：

表 5. 支持的数据类型

数据类型列 1	数据类型列 2
SMALLINT	INTEGER
INT	BIGINT
FLOAT	REAL
DOUBLE	DECIMAL
DEC	DECFLOAT
NUMERIC	NUM
CHARACTER	CHAR
VARCHAR	DATE

表 5. 支持的数据类型 (续)

数据类型列 1	数据类型列 2
TIME	GRAPHIC
VARGRAPHIC	CHARACTER VARYING
TIMESTAMP	CHAR VARYING
CHARACTER FOR BIT DATA	CHAR FOR BIT DATA
VARCHAR FOR BIT DATA	CHARACTER VARYING FOR BIT DATA
CHAR VARYING FOR BIT DATA	用户定义的类型（单值）

不支持的数据类型

分区表可以包含下列数据类型，但不支持将它们用作表分区键列:

- 用户定义的类型（结构化）
- LONG VARCHAR
- LONG VARCHAR FOR BIT DATA
- BLOB
- BINARY LARGE OBJECT
- CLOB
- CHARACTER LARGE OBJECT
- DBCLOB
- LONG VARGRAPHIC
- REF
- C 变长字符串
- Pascal 变长字符串
- XML

如果您选择使用 CREATE TABLE 语句的 EVERY 子句来自动生成数据分区，那么只能将一列用作表分区键。如果您选择通过在 CREATE TABLE 语句的 PARTITION BY 子句中指定每个范围来手动生成数据分区，那么可以将多个列用作表分区键，如以下示例所示:

```
CREATE TABLE sales (year INT, month INT)
   PARTITION BY RANGE(year, month)
   (STARTING FROM (2001, 1) ENDING (2001,3) IN tbsp1,
   ENDING (2001,6) IN tbsp2, ENDING (2001,9)
   IN tbsp3, ENDING (2001,12) IN tbsp4,
   ENDING (2002,3) IN tbsp5, ENDING (2002,6)
   IN tbsp6, ENDING (2002,9) IN tbsp7,
   ENDING (2002,12) IN tbsp8)
```

这将生成 8 个数据分区，即 2001 年和 2002 年的每个季度有一个数据分区。

注:

1. 当将多个列用作表分区键时，将把这些列视为组合键（类似于索引中的组合键），其中，后面的列依赖于前面的列。指定的每个起始值或结束值（所有列一起）不能超出 512 个字符。此限制与 SYSCAT.DATAPARTITIONS 目录视图中的 LOWVALUE 和 HIGHVALUE 列大小对应。如果指定超出 512 个字符的起始值或结束值，就会导致错误 SQL0636N，原因码为 9。

2. 表分区是多列的，而不是多维的。在表分区中，使用的所有列都包含在单个维中。

生成列

可以将生成列用作表分区键。此示例创建包含 12 个数据分区的表，即每个月一个数据分区。对于任何年份，一月份的所有行都将被放到第一个数据分区中，二月份的行将被放到第二个数据分区中，依此类推。

示例 1

```
CREATE TABLE monthly_sales (sales_date date,
 sales_month int GENERATED ALWAYS AS (month(sales_date)))
         PARTITION BY RANGE (sales_month)
         (STARTING FROM 1 ENDING AT 12 EVERY 1);
```

注:

1. 对于表分区键中使用的生成列，不能改变或删除其表达式。不允许对表分区键中使用的列添加生成列表达式。对于表分区键中使用的列，如果尝试添加、删除或改变该列的生成列表达式，就会导致错误（SQL0270N，原因码为 52）。
2. 如果生成列不是单调的，或者优化器无法检测出该列是否是单调的，就不会对范围谓词使用数据分区消除功能。如果存在非单调表达式，那么只能对等价或 IN 谓词执行数据分区消除功能。有关单调性的详细讨论和示例，请参阅第 41 页的『创建 MDC 或 ITC 表时的注意事项』。

分区表的装入注意事项

对目标表进行分区时，将支持所有现有装入功能，但存在以下常规限制:

- 当分区代理程序数大于 1 时，不支持一致点。
- 不支持将数据装入到数据分区子集中的同时保持其余数据分区完全联机。
- 装入操作使用的异常表不能分区。
- 如果目标表包含 XML 列，那么不能指定异常表。
- 当装入实用程序以插入方式或重新启动运行并且装入目标表具有任何已拆离的从属时，那么不能重建唯一索引。
- 与装入 MDC 表相同，装入分区表时将不会保留输入数据记录的精确排序。只有在单元或数据分区中才保留排序。
- 在每个数据库分区上利用多个格式化程序的装入操作仅保留输入记录的大致排序。在每个数据库分区上运行单个格式化程序，将输入记录按单元或表分区键进行分组。要在每个数据库分区上运行单个格式化程序，应显式请求 CPU_PARALLELISM 为 1。

一般装入行为

装入实用程序将数据记录插入到正确的数据分区中。在装入之前，不需要使用外部实用程序（如分割程序）来对输入数据进行分区。

装入实用程序不访问任何拆离的或相连的数据分区。数据仅插入到可视数据分区中。可视数据分区不会拆离，也不会相连。此外，装入替换操作不会截断拆离或相连的数据分区。因为装入实用程序获取针对目录系统表的锁定，所以它将等待任何未落实的 ALTER TABLE 事务。这些事务将获取针对目录表中的相关行的互斥锁定，并且互斥锁定必须先终止，装入操作才能继续。这意味着装入操作运行期间，可能没有未落实的 ALTER TABLE ...ATTACH、DETACH 或 ADD PARTITION 事务。将拒绝目标为拆离或相连的数据分区的所有输入源记

录，但可从异常表中检索它们（如果指定了异常表）。会有一条参考消息写入消息文件，以指示某些目标表数据分区处于相连或拆离状态。针对对应于目标表的相关目录表行的锁定使得用户无法通过在装入实用程序运行时发出 ALTER TABLE ...ATTACH、DETACH 或 ADD PARTITION 操作来更改目标表的分区。

处理无效行

当装入实用程序遇到的记录不属于任何可视数据分区时，将拒绝该记录并且装入实用程序继续进行处理。因为范围限制违例而拒绝的记录的个数不会显式显示出来，但会包括在拒绝的记录的总行数中。因为范围违例而拒绝记录不会增加行警告的数目。会有一条消息 (SQL0327N) 写入装入实用程序消息文件，指示发现范围违例，但不会对每一个记录来记录消息。除了目标表中的所有列之外，异常表还包括用于描述特定行发生的类型违例的列。包含无效数据的行（包括不能分区的数据）将写至转储文件。

因为异常表插入成本很高，所以可以控制插入到异常表中的约束违例。例如，装入实用程序的缺省行为是将本来有效但因为范围限制或唯一约束违例而拒绝的行插入到异常表中。通过对 FOR EXCEPTION 子句分别指定 NORANGEEXC 或 NOUNIQUEEXC 可以关闭此行为。如果指定不应将这些约束违例插入到异常表中，或者未指定异常表，那么有关违反范围限制或唯一约束的行的信息将会丢失。

历史记录文件

如果目标表已分区，那么相应的历史记录文件条目不会包括目标表跨越的表空间列表。另一操作详细程度标识（“R”而不是“T”）指示对分区表运行了装入操作。

终止装入操作

终止装入替换操作将完全截断所有可视数据分区，而终止装入插入操作会将所有可视数据分区截断至装入前的长度。如果 ALLOW READ ACCESS LOAD 操作在装入复制阶段失败，那么在终止该操作期间，索引会变得无效。在终止涉及索引的 ALLOW NO ACCESS LOAD 操作时，索引也会变得无效，这是因为重建索引方式或者增量维护期间插入了键而使得索引处于不一致状态。将数据装入到多个目标中不会影响装入恢复操作，但将无法从装入阶段期间获取的一致点重新启动装入操作。在此情况下，如果对目标表进行分区，那么将忽略 SAVECOUNT 装入选项。此行为与将数据装入到 MDC 目标表中的行为一致。

生成列

如果生成列在任何分区、维或分布键中，那么会忽略 generatedoverride 文件类型修饰符并且装入实用程序会生成值，就像指定了 generatedignore 文件类型修饰符一样。在此情况下，装入错误的生成列值可能导致将记录放置在错误的物理位置上，例如，错误的数据分区、MDC 块或数据库分区。例如，一旦记录放在错误的数据分区上，设置完整性就必须将其移至另一物理位置，这不能在联机设置完整性操作期间完成。

数据可用性

当前 ALLOW READ ACCESS 装入算法扩展至分区表。ALLOW READ ACCESS LOAD 操作允许并发阅读器访问整个表，包括装入和非装入数据分区。

要点: 从 V10.1 FP1 开始，已不推荐使用 ALLOW READ ACCESS 参数，在以后的发行版中可能会将其除去。有关更多详细信息，请参阅『不推荐使用 LOAD 命令中的 ALLOW READ ACCESS 参数』（网址为）。

Ingest 实用程序还支持分区表，并且比附带 ALLOW READ ACCESS 参数的 LOAD 命令更适合允许采用数据并行性和可用性。它可以从文件和管道中移动大量数据，而不用锁定目标表。此外，一旦根据耗用时间或行数落实了数据，数据就会变得可访问。

数据分区状态

在某些情况下，成功装入后可视数据分区可能切换至“设置完整性暂挂”状态和/或“只读访问”表状态。如果该表存在装入操作不能保留的约束，那么数据分区可能会置于这些状态。这种约束可能包括检查约束和拆离的具体化查询表。失败的装入操作会导致所有可视数据分区处于“装入暂挂”表状态。

错误隔离

不支持在数据分区级别进行错误隔离。隔离错误意味着在运行时未出现错误的数据分区上继续装入，而在运行时出现错误的数据分区上停止装入。可在不同数据库分区间隔离错误，但装入实用程序不能在一个可视数据分区子集上落实事务，而回滚其余可视数据分区。

其他注意事项

- 如果有任何索引标记为无效，那么不支持增量。如果索引需要重建或拆离的从属项需要使用 SET INTEGRITY 语句进行验证，那么认为索引无效。
- 同时支持装入到分区表中，这些表使用按范围分区、按散列分布或按维算法组织的任何组合进行分区。
- 对于包括受装入影响的对象和表空间标识列表的日志记录，这些日志记录的大小（LOAD START 和 COMMIT (PENDING LIST)）可能增长得很快，并且因此而降低可供其他应用程序使用的活动日志空间量。
- 当表同时进行了分区和分布时，分区数据库装入可能不会影响所有数据库分区。只有输出数据库分区上的对象才会更改。
- 在装入操作期间，分区表的内存消耗会随表数的增加而增加。注意，总增加量不是线性的，因为仅总内存要求的一小部分与数据分区数成正比。

复制型具体化查询表

*具体化查询表*由查询定义，该查询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查询的一部分可使用具体化查询表来解析，那么可重写该查询以使用具体化查询表。

在分区数据库环境中，可以复制型具体化查询表并使用它们来改善查询性能。*复制型具体化查询表*基于这样一个表：可能已经在单一分区数据库分区组中创建该表，但是您想在另一个数据库分区组中的所有数据库分区之间复制该表。要创建复制型具体化查询表，请使用带 REPLICATED 选项的 CREATE TABLE 语句。

通过使用复制型具体化查询表，可将未典型并置的表并置。对于大事实表和小维表的连接，复制型具体化查询表特别有用。要将所需的额外存储器以及必须更新每个副本所带来的影响降至最小，要复制的表应较小，且更新不频繁。

注: 还应考虑复制那些不常更新的更大的表: 复制的单次成本可通过并置获得的性能效益来抵消。

通过在用于定义复制表的 subselect 子句中指定适当的谓词，可以复制选择的列和/或选择的行。

不支持对复制的具体化查询表执行包含非确定性操作的 DELETE 或 UPDATE 语句。

数据库分区组中的表空间

通过将表空间放在多分区数据库分区组中，就将该表空间内的所有表划分或分区到该数据库分区组的每个数据库分区中。

由此该表空间被创建到了一个数据库分区组中。一旦位于某个数据库分区组中，该表空间就必须保留在该处；而不能更改至另一数据库分区组。CREATE TABLESPACE 语句用于将表空间与数据库分区组关联。

表分区和多维集群表

在同时是多维集群表和数据分区表的表中，可以同时在表分区的范围分区规范和多维集群 (MDC) 键中使用列。与只单独使用多维集群或分区功能相比，同时是多维集群表和分区表的表可以获取较详细的数据分区和块消除。

在许多应用中将 MDC 键列指定为不同于对表进行分区的列很有用。应该注意的是，表分区是多列的，而 MDC 是多维的。

主流 DB2 数据仓库的特征

下列建议主要针对 DB2 V9.1 中新增的典型主流仓库。假定下列特征:

- 数据库在多台机器或多个 AIX® 逻辑分区上运行。
- 使用分区数据库环境（使用 DISTRIBUTE BY HASH 子句来创建表）。
- 有 4 到 50 个数据分区。
- 考虑其 MDC 和表分区的表是主要事实表。
- 表的行数为 100,000,000 至 100,000,000,000。
- 在各种时间范围装入新数据: 每夜、每周和每月。
- 每日接受量为 1 万到 1 亿条记录。
- 数据量变化: 最多的一个月是最少的月的 5 倍。同样，最大维数（生产线，区域）具有 5 倍大小范围。
- 获取 1 到 5 年的详细数据。
- 每月或每个季度转出到期数据。
- 表使用大范围的查询类型。但是，相对于 OLTP 工作负载来说，该工作负载通常是具有下列特征的分析查询:
  - 较大的结果集，最多有 2 百万行
  - 大多数或全部查询都命中视图，而不是基本表
- SQL 子句按范围（BETWEEN 子句）、列表中的项等选择数据。

主流 DB2 V9.1 数据仓库事实表的特征

一个典型仓库事实表可能采用以下设计:

- 在 Month 列中创建数据分区。
- 为转出的每个时间段（例如，1 个月和 3 个月）定义数据分区。
- 在 Day 和 1 到 4 个其他维的基础上创建 MDC 维。典型的维有: 生产线和区域。
- 所有数据分区和 MDC 集群都分布在所有数据库分区中。

MDC 和表分区具有一些相同的好处。下表列示组织中的潜在需求并根据先前确定的特征确定建议的组织方案。

表 6. 将表分区与 MDC 表配合使用

问题	建议的方案	建议
转出期间的数据可用性	表分区	使用 DETACH PARTITION 子句来转出大量数据，并且只出现最少中断。
查询性能	表分区和 MDC	MDC 最适合用来查询多个维。表分区通过数据分区消除提高性能。
最少重组	MDC	MDC 维护集群，从而减少进行重组的必要性。
在传统脱机窗口中转出一个月或更长时间的数据	表分区	数据分区可以完全解决此需求。MDC 不起任何作用，并且仅 MDC 并不适合。
在短时间脱机窗口（小于 1 分钟）期间转出一个月或更长时间的数据	表分区	数据分区可以完全解决此需求。MDC 不起任何作用，并且仅 MDC 并不适合。
转出一个月或更长时间的数据，并同时在不损失任何服务的情况下使表对于提交查询的企业用户完全可用。	MDC	MDC 只能解决一部分此需求。由于表处于脱机状态的时间段太短，表分区并不适合。
每天装入数据（**LOAD** 或 **INGEST** 命令）	表分区和 MDC	此时 MDC 具有很多好处。表分区具有增量的好处。
“连续”装入数据（具有 ALLOW READ ACCESS 或 **INGEST** 命令的 **LOAD** 命令）	表分区和 MDC	此时 MDC 具有很多好处。表分区具有增量的好处。
“传统 BI”查询的查询执行性能	表分区和 MDC	MDC 特别适合用来查询立方体/多个维。表分区通过分区消除提高性能。
通过消除进行重组的必要性或降低执行任务所产生的不良影响，使重组所带来的不良影响降到最低。	MDC	MDC 维护集群，从而减少进行重组的必要性。如果使用 MDC，那么数据分区不提供增量好处。但是，如果不使用 MDC，那么表分区通过在分区级别维护一些粗粒度集群会有助于减少重组的必要性。

示例 1:

考虑一个具有键列 YearAndMonth 和 Province 的表。一种合理的规划此表方法是按日期进行分区，每 2 个月添加一个数据分区。此外，还可以按 Province 进行组织，以便任何两个月日期范围内的特定省份的所有行集群在一起，如图 6 中所示。

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

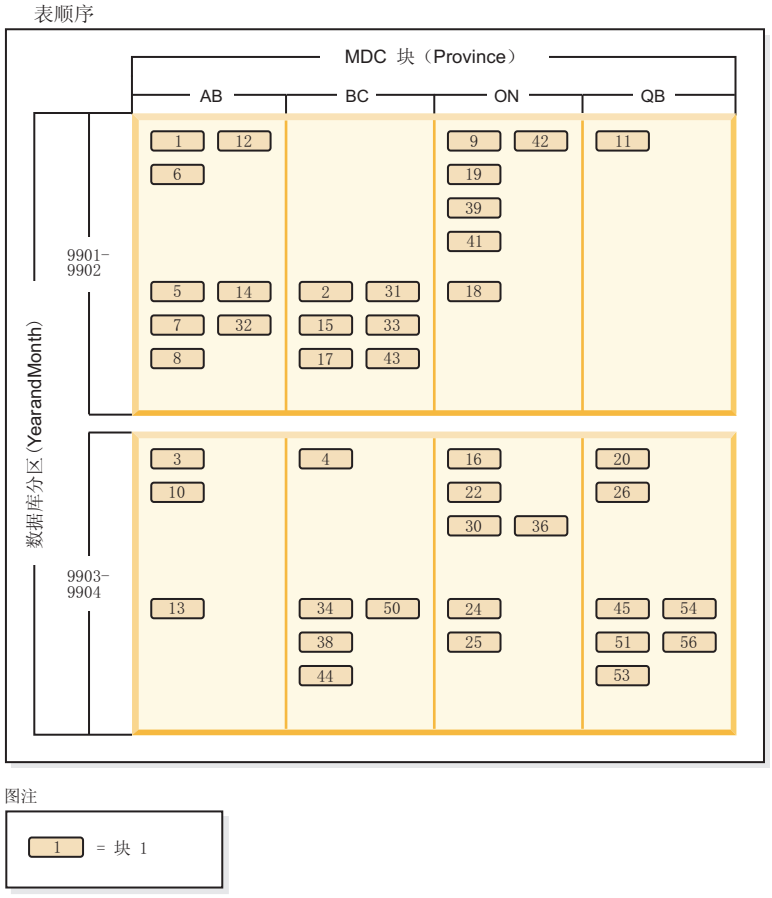


图 6. 按 YearAndMonth 分区并按 Province 组织的表

示例 2:

通过将 YearAndMonth 添加至 ORGANIZE BY DIMENSIONS 子句，可以获得较高的详细程度，如第 29 页的图 7 中所示。

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

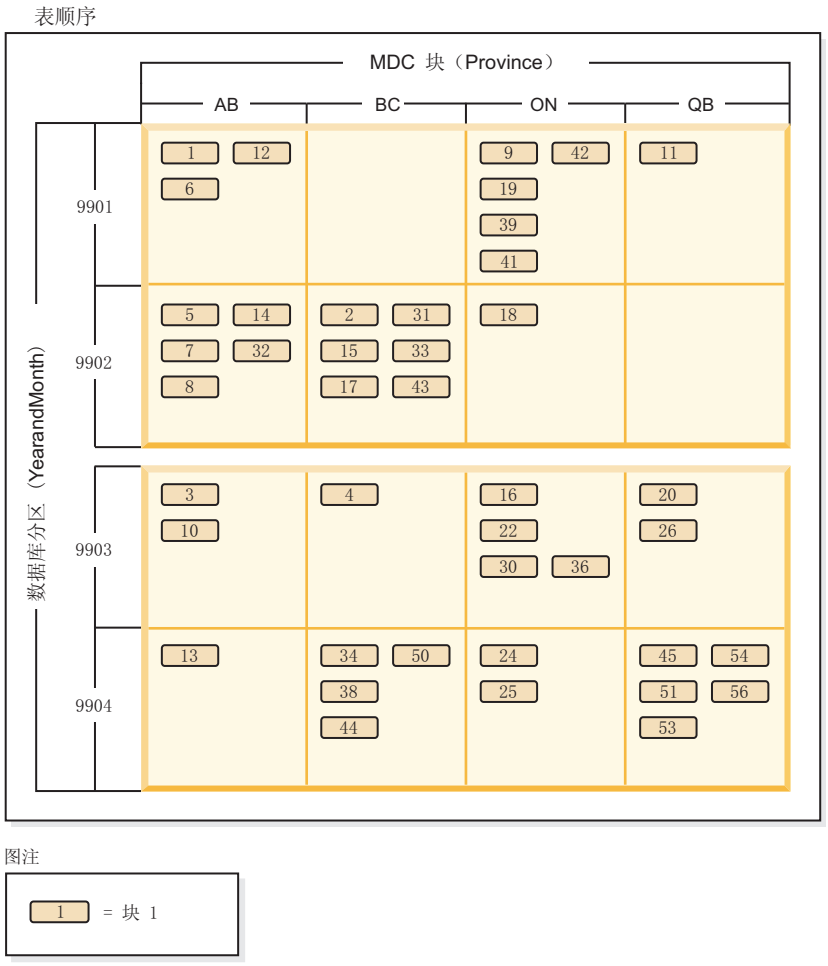


图 7. 按 YearAndMonth 分区并按 Province 和 YearAndMonth 组织的表

如果分区导致每个范围内只有单个值，那么通过在 MDC 键中包括表分区列不能获得任何好处。

注意事项

- 与基本表相比，MDC 表和分区表都需要一些存储器。这些存储器需求是附加的，但相对于所带来的好处来说，它们是合理的。
- 如果选择不在分区数据库环境中组合表分区和 MDC 功能，那么在您可以非常自信地预计数据分布情况时（通常也就是此处讨论的系统类型情况），使用表分区是最好的选择。否则，应考虑 MDC。
- 对于使用 DB2 V9.7 FP1 或更高发行版创建的数据分区 MDC 表，表的 MDC 块索引是分区索引。对于使用 DB2 V9.7 或更早发行版创建的数据分区 MDC 表，表的 MDC 块索引是非分区索引。

DB2 pureScale 环境中的表分区

可在 DB2 pureScale® 中使用表分区在多个分区间划分大表对象以获取更好性能。

可在 DB2 pureScale 表（包括使用 PARTITION BY RANGE 子句的表）中使用表分区。此外，可在 DB2 pureScale 环境中使用与表分区相关联的命令。

这意味着（例如）所有以下操作受支持:

- 通过 ALTER TABLE 语句提供的滚入和滚出分区操作
- CREATE INDEX 语句的 PARTITIONED 和 NOT PARTITIONED 子句
- 对于分区索引，REORG TABLE 和 REORG INDEXES ALL 语句的 ON DATA PARTITION 子句

此外，MON_GET_PAGE_ACCESS_INFO 表函数已更新以使用分区表。作用于数据分区的所有现有监视函数将使用 DB2 pureScale 表。

如果已在使用 DB2 pureScale Feature，那么可使用表分区来帮助解决页争用问题。通过在更大的范围内展开争用，可减少数据页争用；同样，可通过使用分区索引来减少针对索引页面的争用。

注: 从 DB2 pureScale 性能角度来看，使用的内存量取决于表分区数和索引数。用于成员的分区的内存资源取决于 **dbheap** 配置参数。在 CF 上，内存资源由 **cf_sca_sz** 配置参数定义。

第 2 章 范围集群表

范围集群表 (RCT) 具有一种表布局方案，在此方案中，表中的每一条记录都预先确定了记录标识 (RID)。RID 是用来在表中查找记录的内部标识。

将使用算法来使记录键值与特定表行的位置相关联。此方法使得对特定表行的访问非常快速。该算法不会使用散列，这是因为散列不会保持键值顺序。如果保持了此顺序，那么不需要随着时间的过去对表数据进行重组。

表中的每个记录键值必须是:

- 唯一
- 非空
- 整数（SMALLINT、INTEGER 或 BIGINT）
- 单调增大
- 在基于键中每一列的预先确定的一组范围内。（必要时，请在 CREATE TABLE 语句中使用 ALLOW OVERFLOW 选项，以允许行具有在所定义的值范围之外的键值。）

除了允许直接访问特定表行之外，使用范围集群表还有其他优点。

- 需要的维护更少。不存在辅助结构（例如，B+ 树索引，在每次执行插入、更新或删除操作之后都需要更新该索引）。
- 与大小类似的具有 B+ 树索引的常规表进行比较时，RCT 需要进行的日志记录更少。
- 需要的缓冲池内存更少。不需要额外的内存来存储辅助结构（例如，B+ 树索引）。

将为 RCT 预分配空间，并且将保留此空间供该表使用，即使该表中尚没有记录时也是如此。因此，范围集群表不需要可用空间控制记录 (FSCR)。创建表时，表中没有记录；然而，将预分配整个页范围。预分配是根据记录大小和要存储的最大记录数来进行的。如果定义了变长字段（例如，VARCHAR），那么将使用该字段的最大长度，并且整个记录大小的长度是固定的。这会导致无法充分利用空间。如果键值稀疏，那么未使用的空间对范围扫描性能具有负面影响。范围扫描必须访问范围内所有可能的行，即使是尚未包含数据的行。

如果需要修改范围集群表的模式，那么必须使用新模式名称重新创建该表，然后用旧表中的数据进行填充。例如，如果需要改变表的范围，那么必须创建具有新范围的表并用旧表中的数据进行填充。

如果 RCT 允许使用溢出记录，并且新记录具有在所定义的值范围之外的键值，那么该记录将放置在动态分配的溢出区域中。将更多记录添加到此溢出区域时，对涉及此溢出区域的表的操作将需要更多的处理时间。溢出区域越大，访问溢出区域所需的时间就越多。如果这成为了问题，请考虑通过将数据导出到具有更大范围的新 RCT 来减小溢出区域的大小。

对范围集群表的限制

在某些上下文中不能使用范围集群表，并且某些实用程序无法对范围集群表进行操作。

范围集群表存在以下限制:

- 不能在 DB2 pureScale 环境 中指定范围集群表 (SQLSTATE 42997)。
- 分区表不能是范围集群表。
- 已声明的临时表和已创建的临时表不能是范围集群表。
- 自动摘要表 (AST) 不能是范围集群表。
- 装入实用程序不受支持。可通过导入实用程序或通过并行插入应用程序将数据插入到范围集群表。
- **REORG** 实用程序不受支持。不需要重组通过 DISALLOW OVERFLOW 选项定义的范围集群表。通过 ALLOW OVERFLOW 选项定义的范围集群表不能使此溢出区域中的数据进行重组。
- 如果表是范围集群具体化查询表，那么无法指定 CREATE TABLE 语句的 DISALLOW OVERFLOW 子句。
- 设计顾问程序将不会建议范围集群表。
- 多维集群和集群索引与范围集群表不兼容。
- 值和缺省压缩不受支持。
- 不支持对范围集群表进行逆向扫描。
- **IMPORT** 命令上的 **REPLACE** 参数不受支持。
- 不支持 ALTER TABLE...ACTIVATE NOT LOGGED INITIALLY 语句使用 WITH EMPTY TABLE 选项。

第 3 章 多维集群 (MDC) 表

多维集群表

多维集群 (MDC) 提供了一种极佳的方法来将多个表中的数据集群在多个维中，它具有灵活、连续并且自动完成的特点。MDC 可显著提高查询性能。

此外，MDC 还可以极大地减少在插入、更新和删除操作期间执行数据维护（例如，重组）和索引维护操作的开销。MDC 主要用于数据仓储和大型数据库环境，但也可用于联机事务处理 (OLTP) 环境中。

常规表与 MDC 表的比较

常规表的索引是基于记录来建立的。索引的任何集群都仅限于单个维。在 V8 之前，数据库管理器仅支持通过集群索引对数据进行单维集群。通过使用集群索引，在表中插入和更新记录时，数据库管理器会尝试按索引的键顺序来维护各页上数据的物理顺序。

集群索引极大地提高了具有包含集群索引的一个或多个键的谓词的范围查询的性能。使用好的集群索引可以提高性能，这是因为只需要访问表的一部分，并且可以执行更有效地预取。

但是，使用集群索引的数据集群有一些缺点。首先，因为随时间的推移将逐渐填满数据页上的空间，因此，将不能保证集群。插入操作将尝试把记录添加至具有相同或相似集群键值的那些页附近的页，但是，如果在理想位置没有空间，那么将把它插入到表中的其他地方。因此，可能需要定期进行表重组，以便对表进行重新集群并设置具有附加可用空间的页以容纳将来的集群插入请求。

其次，只能将一个索引指定为“集群”索引，其他所有索引都将是非集群索引，这是因为只能在一维中以物理方式对数据进行集群。这种局限性与集群索引是基于记录的这样一个事实有关，因为所有索引都是在 V8.1 之前建立的。

最后，因为对于表中的每个记录，基于记录的索引都包含一个指针，所以它们可能非常大。

集群索引

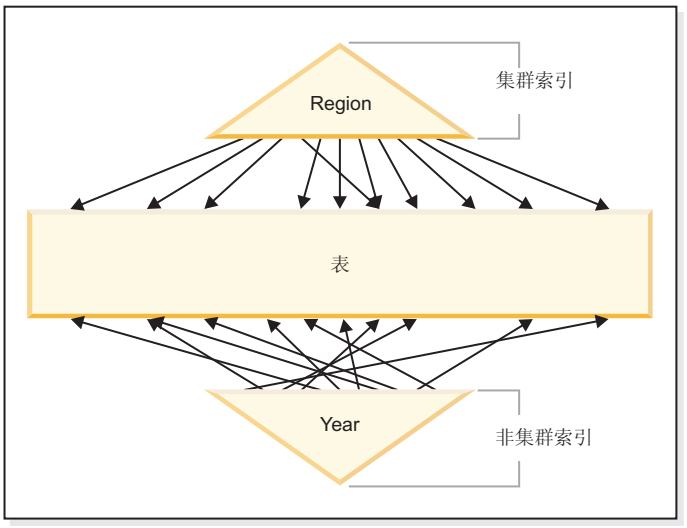


图 8. 具有集群索引的常规表

图 8 中的表上定义了两个基于记录的索引:

- “Region”上的集群索引
- “Year”上的另一个索引

“Region”索引是一个集群索引，它意味着当在索引中扫描键时，通常将在表中的同一页或邻近页上找到相应的记录。相反，“Year”索引是一个非集群索引，它意味着在该索引中扫描键时，很可能将在表的随机页上找到相应的记录。对集群索引进行扫描将获得更好的 I/O 性能，并且在该索引的数据集群程度越高时，顺序预取就会带来更多好处。

MDC 引入了基于块的索引。“块索引”指向记录块或记录组，而不是指向单个记录。通过从物理上根据集群值将 MDC 表中的数据组织成块，然后使用块索引来访问这些块，MDC 不仅能够解决集群索引的所有缺点，还能显著地改善性能。

首先，MDC 使表能够同时在多个键或维上进行物理集群。通过 MDC，多个维或集群键也具备了单维集群的优点。在对表的一个或多个指定维具有集群的情况下，查询性能会提高。这些查询不仅是只访问包含具有正确维值的记录的那些页，还会将这些满足要求的页组成块或扩展数据块。

其次，尽管具有集群索引的表经过一段时间之后可能会变成不是集群的，但是大多数情况下 MDC 表还是能够对所有维自动并持续地维护和保证它的集群。因此，无需频繁重组 MDC 表就可以复原数据的物理顺序。虽然始终将维护块中的记录顺序，但是在插入时（某些情况下甚至在初始装入时）不会维护块的物理排序（即，在块索引扫描时从一个块扫描到另一个块）。

最后，在 MDC 中，集群索引是基于块的。这些索引比常规的基于记录的索引要小很多，因此，占用的磁盘空间更少，并且扫描时速度会更快。

选择 MDC 表维

决定使用多维集群表后，您选择的维将不仅取决于将使用表并且受益于块级别集群的查询类型，更重要的是取决于实际数据的数量和分布情况。

将受益于 MDC 的查询

选择表的集群维时首先应注意确定哪些查询将受益于块级别的集群。通常，当根据组成将对数据完成的工作的查询来选择维时有一些候选维。这些候选维的等级是很重要的。在等同查询或范围谓词查询中涉及到的列（特别是具有较小基数的列）将最受益于集群维。请考虑为涉及与维表进行星型连接的 MDC 事实表中的外键创建维。请记住，对多个维进行自动持续的集群以及扩展数据块或块级别的集群会提高性能。

有许多种查询都可以利用多维集群。以下是这样一些查询的示例。在这些示例的某些示例中，假定存在一个 MDC 表 t1，具有 c1、c2 和 c3 维。在其他示例中，假定存在一个 MDC 表 mdctable，具有 color 和 nation 维。

示例 1:

```
SELECT .... FROM t1 WHERE c3 < 5000
```

此查询涉及到单个维的范围谓词，因此可以在内部重写以使用 c3 的维块索引来访问该表。将扫描索引以查找键值小于 5000 的块标识 (BID)，并且对块的结果集应用最小关系扫描以检索实际记录。

示例 2:

```
SELECT .... FROM t1 WHERE c2 IN (1,2037)
```

此查询涉及到单个维的 IN 谓词，并且可以触发基于块索引的扫描。可以内部重写此查询以使用 c2 的维块索引来访问该表。将扫描索引以查找具有键值 1 和 2037 的 BID，并且对块的结果集应用最小关系扫描以检索实际记录。

示例 3:

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' AND NATION='USA'
```

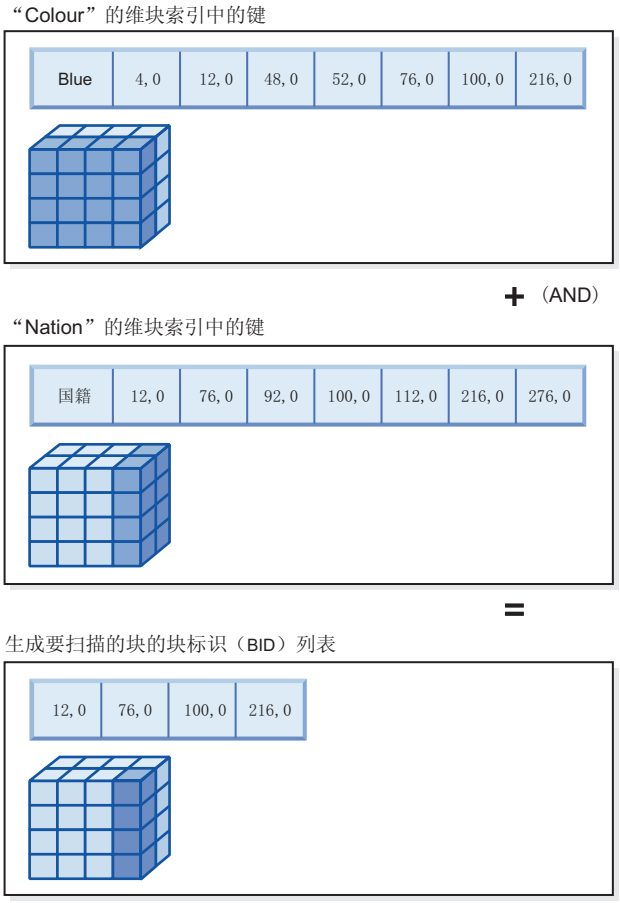


图 9. 对两个块索引使用逻辑 AND 操作的查询请求。

要执行此查询请求，完成下列步骤（显示在图 9 中）：

- 完成维块索引查找：对 Blue 片执行一次，对 USA 片执行一次。
- 执行块逻辑 AND 操作以确定两个片的交集。即，逻辑 AND 操作只确定在这两个片中都可以找到的那些块。
- 对表中获得的块执行最小关系扫描。

示例 4:

```
SELECT ... FROM t1
   WHERE c2 > 100 AND c1 = '16/03/1999' AND c3 > 1000 AND c3 < 5000
```

此查询涉及到 c2 和 c3 的范围谓词和 c1 的等价谓词，并且还要执行逻辑 AND 操作。可以内部重写此查询以访问每个维块索引上的表:

- 扫描 c2 块索引以查找具有大于 100 的键值的 BID
- 扫描 c3 块索引以查找键值在 1000 到 5000 之间的 BID
- 扫描 c1 块索引以查找键值为“16/03/1999”的 BID。

然后，对从每个块扫描获得的 BID 执行逻辑 AND 操作以查找它们的交集，并对块的结果集应用最小关系扫描以查找实际记录。

示例 5:

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' OR NATION='USA'
```

要执行此查询请求，完成下列步骤:

- 完成维块索引查找: 对每个片都执行一次。
- 执行逻辑 OR 操作以查找两个片的并集。
- 对表中获得的块执行最小关系扫描。

示例 6:

```
SELECT .... FROM t1 WHERE c1 < 5000 OR c2 IN (1,2,3)
```

此查询涉及到 c1 维的范围谓词、c2 维的 IN 谓词以及逻辑 OR 操作。可以内部重写此查询以访问维块索引 c1 和 c2 上的表。扫描 c1 维块索引以查找小于 5000 的值，并对 c2 维块索引执行另一个扫描以查找值 1、2 和 3。对从每个块索引扫描获得的 BID 执行逻辑 OR 操作，然后对块的结果集应用最小关系扫描以查找实际记录。

示例 7:

```
SELECT .... FROM t1 WHERE c1 = 15 AND c4 < 12
```

此查询涉及到 c1 维的等价谓词和不是维的一列的另一个范围谓词以及逻辑 AND 操作。可以内部重写该查询以访问 c1 的维块索引，以获取 c1 值为 15 的表的片的块列表。如果 c4 具有 RID 索引，那么可以进行索引扫描以检索 c4 小于 12 的记录的 RID，然后可以对获得的块列表和此记录列表执行逻辑 AND 操作。此交集将除去 c1 为 15 的块中不存在的 RID，而仅从表中检索合格块中存在的列示的 RID。

如果 c4 没有 RID 索引，那么可以扫描块索引以查找合格块的列表，并且在每个块的最小关系扫描期间，可以对发现的每个记录应用谓词 c4 < 12。

示例 8:

假定这样一个方案，存在 color、year 和 nation 维以及部件号的行标识（RID）索引，那么可以进行以下查询。

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' AND PARTNO < 1000
```

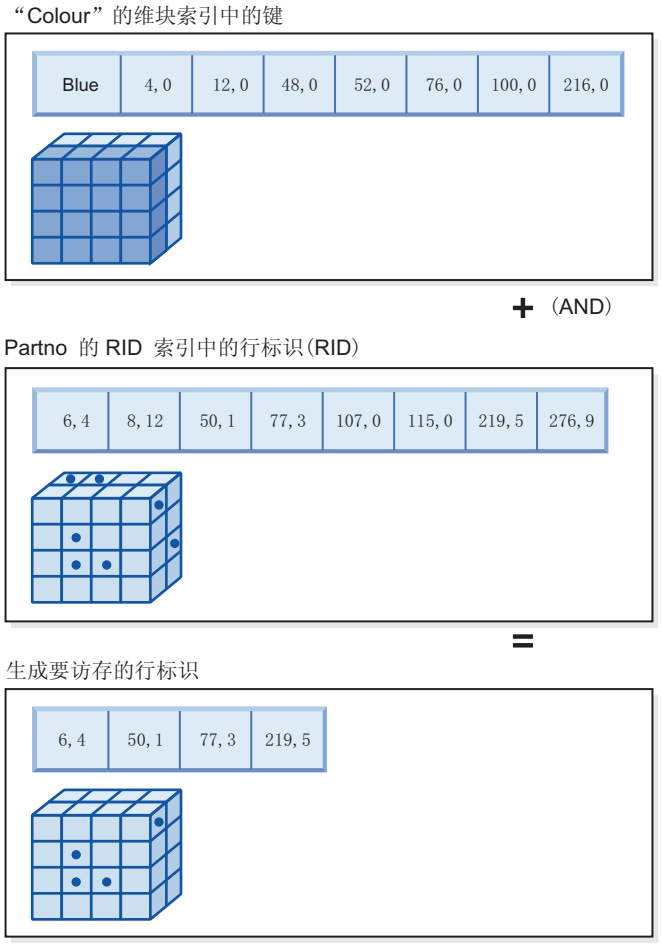


图 10. 对块索引和行标识（RID）索引使用逻辑 AND 操作的查询请求

要执行此查询请求，完成下列步骤（显示在图 10 中）：

- 完成维块索引查找和 RID 索引查找。
- 对块和 RID 使用逻辑 AND 操作来确定片与满足谓词条件的那些行的交集。
- 获得的结果仅为还属于满足条件的块的那些 RID。

示例 9:

```
SELECT * FROM MDCTABLE WHERE COLOR='BLUE' OR PARTNO < 1000
```

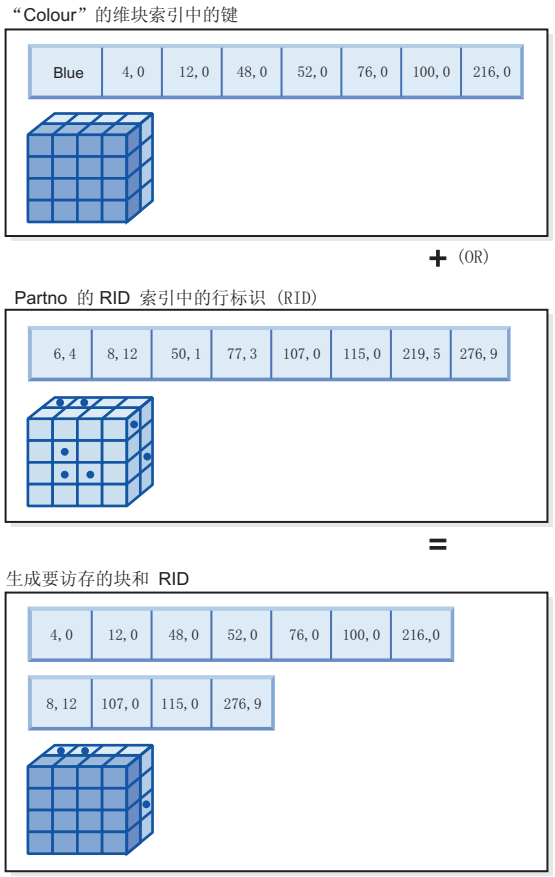


图 11. 使用逻辑 OR 操作的块索引和行标识的工作方式

要执行此查询请求，完成下列步骤（显示在图 11 中）：

- 完成维块索引查找和 RID 索引查找。
- 对块和 RID 使用逻辑 OR 操作来确定片与满足谓词条件的那些行的并集。
- 获得的结果是满足条件的块中的所有行以及满足谓词条件但在满足条件的块外部的其他 RID。对每个块执行最小关系扫描以检索它们的记录并逐个检索这些块之外的其他记录。

示例 10:

```
SELECT ... FROM t1 WHERE c1 < 5 OR c4 = 100
```

此查询涉及到 c1 维的范围谓词、非维列 c4 的等价谓词以及逻辑 OR 操作。如果 c4 列上具有 RID 索引，那么可以内部重写此查询以使用 c1 上的维块索引和 c4 上的 RID 索引来执行逻辑 OR 操作。如果 c4 上没有索引，那么可以转为选择表扫描，因为必须检查所有记录。逻辑 OR 操作将对 c1 使用块索引扫描以找到小于 4 的值，并对 c4 使用 RID 索引扫描以找到值 100。对每个满足条件的块执行最小关系扫描，因为这些块中的所有记录都满足条件，还要检索这些块外部的记录的任何其他 RID。

示例 11:

```
SELECT .... FROM t1,d1,d2,d3
      WHERE t1.c1 = d1.c1 and d1.region = 'NY'
            AND t2.c2 = d2.c3 and d2.year='1994'
            AND t3.c3 = d3.c3 and d3.product='basketball'
```

此查询涉及星型连接。在本示例中，t1 是事实表并且它具有外键 c1、c2 和 c3（与主键 d1、d2 和 d3 相对应）以及维表。维表不一定是 MDC 表。Region、year 和 product 是可以使用常规索引或块索引建立索引的维表的列（如果维表是 MDC 表）。当根据 c1、c2 和 c3 值访问事实表时，可以对这些列的维块索引进行块索引扫描，然后使用获得的 BID 来执行逻辑 AND 操作。当存在块列表时，可以对每个块进行最小关系扫描以获取记录。

单元的密度

选择适当的维和扩展数据块大小对于 MDC 设计来说**非常**重要。这些因素将确定表的期望单元密度。它们之所以重要是因为对每个现有单元都分配扩展数据块，无论该单元中的记录数是多少。正确的选择将利用基于块的索引和多维集群，从而获得更好的性能。目标是获得填充密度很高的块，以便从多维集群中获得最多益处并且最佳地利用空间。

因此，设计多维表时的非常重要的注意事项就是表中的期望单元密度（基于现有的和预期的数据）。可以根据查询性能选择一组维，它导致表中的潜在单元数很大（基于每个维的可能值的数目）。表中的可能单元的数目等于每个维的基数的笛卡尔乘积。例如，如果根据 Day、Region 和 Product 维来集群表且涵盖五年的数据，那么表中可能具有 `1821 days * 12 regions * 5 products = 109260` 个不同的单元。任何仅包含几个记录的单元仍需要整块的页，以存储该单元的记录。如果块大小较大，那么此表结果可能会比它需要的大得多。

要获得最佳单元密度，应考虑以下几种设计因素:

- 维数不同。
- 一个或多个维的详细程度不同。
- 表空间的块（扩展数据块）大小和页大小不同。

执行下列步骤来获得最佳设计:

1. 标识候选维。

   确定哪些查询将采用块级别集群。检查具有下列某些特征或所有特征的列可能具有的工作负载:

   - 任何 IN 列表谓词的范围和等价性
   - 数据的转入或转出
   - Group-by 和 order-by 子句
   - Join 子句（特别是在星型模式环境中）。

2. 估计单元数目。

   标识在根据一组候选维组织的表中可能存在多少个潜在的单元。确定数据中出现的维值的唯一组合的数目。如果表存在，那么可以通过选择每个列中将是表的维数的单值数目来确定当前数据的准确数目。或者，如果只有表的统计信息，那么可以通过乘以候选维的列基数来确定一个大概数目。

   注: 如果表处于分区数据库环境中，并且分布键与考虑的任何维都无关，那么请通过用所有数据除以数据库分区数来确定每个单元的平均数据量。

3. 估计空间占用率或密度。

按平均情况来说，认为每个单元都有一个部分填充的块，该块中只存储了很少的行。当每个单元的行数变得越来越少时，部分填充的块就会更多。另外还要注意，按平均情况来说（假定存在很少甚至没有数据偏差），可以通过用表中的记录数除以单元数来计算每个单元的记录数。但是，如果表处于分区数据库环境中，那么请考虑每个数据库分区上的每个单元中有多少条记录，因为是按照数据库分区来为数据分配块的。当估计分区数据库环境中的空间占用率和密度时，请考虑每个数据库分区（而不是整个表）上每个单元的平均记录数。

可以采用以下几种方法来提高密度:

- 减小块大小，以便使部分填充的块占用较少的空间。

  通过适当减小扩展数据块大小来减小每个块的大小。具有部分填充的块的每个单元或者只包含具有很少记录的一个块的每个单元都占用较少的空间。但是，采用一种折衷办法就是，对于具有很多记录的那些单元，需要更多的块来包含它们。这将增加块索引中的这些单元的块标识 (BID) 的数目，使这些索引更大并且潜在地导致更多的索引插入和删除操作，因为会更快地清空和填充块。与数目较少的集群数据的较大组合相比较，它还会导致表中的集群数据的更多更小的组合，以装入更多填充单元值。
- 通过减少维数或者通过增大具有生成列的单元的详细程度来减少单元数。

  可以将一个或多个维上滚为较低的详细程度，以给予它较低的基数。例如，可以仍然根据 Region 和 Product 对将前一个示例中的数据库进行集群，但将 Day 维替换为 YearAndMonth 维。这将为 YearAndMonth、Region 和 Product 维给予基数 60（12 个月乘以 5 年）、12 和 5，可能的单元数为 3600。于是，每个单元将具有更大范围的值，并且降低了只包含很少记录的可能性。

  请考虑对涉及到的列通常使用的谓词，例如，使用得较多的是 Month of Date、Quarter 或 Day。这将影响更改维的详细程度的期望度。例如，如果大多数谓词是针对特定日期的，并且您根据 Month 对表进行了集群，那么 DB2 for Linux, UNIX, and Windows可以使用 YearAndMonth 的块索引来快速缩小包含所需日期的月份的范围并且仅访问这些相关联的块。但是，在扫描块时，必须应用 Day 谓词来确定哪些日期合格。但是，如果根据 Day 进行集群（并且 Day 具有很高的基数），那么可以使用 Day 的块索引来确定扫描哪些块，并且只需要对合格的每个单元的第一条记录重新应用 Day 谓词。在此情况下，在使用用户定义的函数上滚 Region 列（它包含 12 个不同的值）至 Regions West、North、South 和 East 时，最好考虑上滚其他维之一以增大单元的密度。

创建 MDC 或 ITC 表时的注意事项

创建 MDC 或 ITC 表时要考虑许多因素。在决定如何创建、布置和使用 MDC 或 ITC 表时，会受到当前数据库环境（例如，是否具有分区数据库）和所选择的维的影响。

将数据从现有表移至 MDC 表

要在数据仓库或大型数据库环境中提高查询性能和降低对数据维护操作的需求，可以将数据从常规表移至多维集群 (MDC) 表。要将数据从现有表移至 MDC 表:

1. 导出数据,
2. 删除原始表（可选），

3. 创建多维集群 (MDC) 表（使用带 ORGANIZE BY DIMENSIONS 子句的 CREATE TABLE 语句），
4. 将数据装入 MDC 表。

可以使用称为 SYSPROC.ALTOBJ 的 ALTER TABLE 过程来将现有表中的数据转换为 MDC 表中的数据。可以从“DB2 设计顾问程序”中调用此过程。在这两个表之间转换数据所需要的时间可能很长，这取决于表的大小以及需要转换的数据量。

ALTOBJ 过程在改变表时将执行下列步骤:

1. 删除表的所有从属对象，
2. 重命名表，
3. 使用新定义来创建表，
4. 重新创建表的所有从属对象，
5. 将表中的现有数据变换为新表中所需要的数据。即，从旧表中选择数据，然后将该数据装入新表中，可以在新表中使用列函数来将旧的数据类型变换为新的数据类型。

将数据从现有表移至 ITC 表

要降低对数据维护操作的需求，可以将数据从常规表移至插入时间集群 (ITC) 表。要将数据从现有表移至 ITC 表，请使用联机表移动存储过程。

ExampleBank 方案显示了如何将现有表中的数据移至 ITC 表。此方案还说明使用 ITC 表时回收空间的便利程度。有关更多信息，请参阅“相关概念”链接。

DB2 设计顾问程序上的 MDC 顾问程序功能部件

DB2 设计顾问程序 (**db2advis**) 具有 MDC 功能部件。此功能部件建议用于 MDC 表中的集群维（包括基本列的粗糙度）以便提高工作负载性能。*粗糙度*这个术语表示用来减小集群维基数（单值的数目）的一个数学表达式。粗糙度的常见示例是粗糙度可为日期、日期所在的星期、日期所在的月份或一年中的季度。

使用“DB2 设计顾问程序”的 MDC 功能部件要求数据库中至少存在几个扩展数据块的数据。“DB2 设计顾问程序”使用数据来对数据密度和基数建立模型。

如果数据库的表中没有数据，那么“DB2 设计顾问程序”不会建议使用 MDC，即使该数据库包含空表，但有一组虚假的统计信息来表示它是一个已填充的数据库。

建议还标识了用来定义维的粗糙度的潜在生成列。建议中不包括可能的块大小。在为 MDC 表提供建议时，使用表空间的扩展数据块大小。假定将在现有表所在的表空间中创建建议的 MDC 表，因此该表具有相同的扩展数据块大小。对 MDC 维的建议将根据表空间的扩展数据块大小的变化而变化，这是因为扩展数据块大小会影响可以填充到块或单元中的记录数。此扩展数据块大小将直接影响单元的密度。

尽管可以为表建议单个维或多个维，但请只考虑单列维而不要考虑组合列维。MDC 功能部件将建议大多数受支持的数据类型使用的粗糙度，目标是减小所采用的 MDC 解决方案中的单元的基数。异常的数据类型包括：CHAR、VARCHAR、GRAPHIC 和 VARGRAPHIC 数据类型。所有受支持的数据类型都将被强制类型转换为 INTEGER 并通过生成的表达式来设置粗糙度。

“DB2 设计顾问程序”的 MDC 功能部件的目标是选择可提高性能的 MDC 解决方案。另一个目标是将数据库的存储扩展限制在适当的级别。使用统计方法来确定每个表的最大存储扩展。

顾问程序中的分析操作既会利用块索引访问的优点，也会受到 MDC 对表的维执行插入、更新和删除操作的影响。对表执行这些操作时可能会导致在各个单元之间移动记录。分析操作还会模拟对特定 MDC 维上的数据进行组织时产生的任何表扩充而对性能造成的潜在影响。

通过对 db2advis 实用程序使用 `-m <advise type>` 标志来运行 MDC 功能部件。使用“C”建议类型来指示多维集群表。建议类型为: “I”表示索引、“M”表示具体化查询表、“C”表示 MDC，而“P”表示分区数据库环境。建议类型可以相互组合使用。

注: “DB2 设计顾问程序”不会处理其大小不到 12 个扩展数据块的表。

当提出建议时，顾问程序将同时分析 MQT 和常规基本表。

MDC 功能部件的输出包括:

- 每个表的生成列表达式（用于对 MDC 解决方案中出现的维设置粗糙度）。
- 对每个表建议的 ORGANIZE BY DIMENSIONS 子句。

对标准输出和作为说明工具的一部分的 ADVISE 表报告了建议。

MDC 表和分区数据库环境

多维集群可用于分区数据库环境中。实际上，MDC 可以作为分区数据库环境的补充。分区数据库环境用来将一个表中的数据分配到多个物理数据库分区或逻辑数据库分区上，以便达到下列目的:

- 利用多台机器来并行增加并行处理请求，
- 将表的物理大小增大到超过单个数据库分区的限制，
- 提高数据库的可伸缩性。

分发表的原因与该表是 MDC 表还是常规表无关。例如，选择用来组成分布键的列的规则是相同的。MDC 表的分布键可以包括任何列，无论这些列是否组成表的维的一部分。

如果分布键与表的某个维完全相同，那么每个数据库分区都包含该表的不同部分。例如，如果作为示例的 MDC 表按颜色分布在两个数据库分区上，那么将使用 Color 列来划分数据。因此，可能在一个数据库分区上找到 Red 和 Blue 片，而在另一个数据库分区上找到 Yellow 片。如果分布键与表中的维不完全相同，那么每个数据库分区都具有每个片的数据的子集。当选择维和估计单元占用率时，请注意，每个单元的平均数据总量是通过用所有数据除以数据库分区数来确定的。

具有多个维的 MDC 表

如果知道查询中大量使用了某些谓词，那么可以根据涉及到的列对表进行集群。可以使用 ORGANIZE BY DIMENSIONS 子句来执行此操作。

示例 1:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
      ORGANIZE BY DIMENSIONS (c1, c3, c4)
```

示例 1 中的表根据形成逻辑立方体（即，具有三个维）的三个列中的值集群。现在可以在查询处理期间根据一个或多个维对表进行逻辑分片，以便涉及的关系运算符仅处理相应的片或单元中的块。块的大小（页数）是表的扩展数据块大小。

具有基于多列的维的 MDC 表

每个维可以由一列或多列组成。作为一个示例，可以创建一个根据包含两列的一个维来集群的表。

示例 2:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
      ORGANIZE BY DIMENSIONS (c1, (c3, c4))
```

在示例 2 中，表根据两个维 c1 和 (c3, c4) 来进行集群。这样，在查询处理期间，表可以根据 c1 维或组合 (c3, c4) 维逻辑分片。该表与示例 1 中的表具有相同数目的块，但是少一个维块索引。在示例 1 中，有三个维块索引，列 c1、c3 和 c4 各一个。在示例 2 中，有两个维块索引，一个针对列 c1，而另一个针对 c3 和 c4。这两个方法的主要差别在于，在示例 1 中，涉及 c4 的查询可以使用 c4 的维块索引来快速直接地访问相关数据块。在示例 2 中，c4 是维块索引中的辅助键部分，因此涉及 c4 的查询涉及更多的处理。但是，在示例 2 中，将少维护和存储一个块索引。

“DB2 设计顾问程序”将不对包含多列的维提供建议。

将列表达式作为维的 MDC 表

列表达式也可用于集群维。根据列表达式集群的功能对于将维上滚至更低的详细程度非常有用，例如，将地址上滚为地理位置或区域，或者将日期上滚为星期、月份或年。要以此方式实现维的上滚，可以使用生成列。此类型的列定义允许使用可以表示维的表达式来创建列。在示例 3 中，该语句创建根据一个基本列和两个列表达式进行集群的表。

示例 3:

```
CREATE TABLE T1(c1 DATE, c2 INT, c3 INT, c4 DOUBLE,
      c5 DOUBLE GENERATED ALWAYS AS (c3 + c4),
   c6 INT GENERATED ALWAYS AS (MONTH(C1)))
           ORGANIZE BY DIMENSIONS (c2, c5, c6)
```

在示例 3 中，c5 列是基于 c3 和 c4 列的表达式，而 c6 列会及时将 c1 列上滚至更低的详细程度。此语句根据 c2、c5 和 c6 列中的值来集群该表。

对生成列维的范围查询

对生成列维的范围查询需要单调列函数。表达式必须是单调的才能为生成列的维派生范围谓词。如果对生成列创建维，那么对基本列的查询能够利用生成列的块索引来提高性能（有一种情况例外）。要使基本列（例如，日期）的范围查询对维块索引使用范围扫描，用来在 CREATE TABLE 语句中生成列的表达式必须是单调的。尽管列表达式可以包括任何有效表达式（包括用户定义的函数 (UDF)），但是如果表达式不是单调的，那么当等价谓词或 IN 谓词都在基本列上时，它们才能够使用块索引来满足查询。

作为一个示例，假定使用生成列 month 的维来创建 MDC 表，其中 month = INTEGER (date)/100。对于该维 (month) 的查询，可以执行块索引扫描。对于基本列 (date) 的查询，也可以执行块索引扫描来缩小要扫描的块的范围，然后只将日期的谓词应用于这些块中的行。

编译器将生成要在块索引扫描中使用的其他谓词。例如，对于以下查询:

```
SELECT * FROM MDCTABLE WHERE DATE > "1999-03-03" AND DATE < "2000-01-15"
```

编译器将生成以下谓词: 『month >= 199903』 和『month <= 200001』，它们可以用作维块索引扫描的谓词。当对获得的块进行扫描时，会将原始谓词应用于这些块中的行。

非单调表达式允许对该维应用等价谓词。非单调函数的一个较好的示例是 MONTH( )，如示例 3 中的 c6 列的定义所示。如果 c1 列是日期、时间戳记或日期或时间戳记的有效字符串表示法，那么函数将返回范围是 1 到 12 的整数值。尽管函数的输出是确定的，但是实际上它生成的输出与阶跃函数（即，循环模式）相似:

```
MONTH(date('01/05/1999')) = 1
MONTH(date('02/08/1999')) = 2
MONTH(date('03/24/1999')) = 3
MONTH(date('04/30/1999')) = 4
...
MONTH(date('12/09/1999')) = 12
MONTH(date('01/18/2000')) = 1
MONTH(date('02/24/2000')) = 2
...
```

尽管此示例中的日期是连续增加的，但是 MONTH(date) 不会增加。更具体而言，每当 date1 大于 date2，并不能保证 MONTH(date1) 大于或等于 MONTH(date2)。这是单调性所要求的。此非单调性是允许的，但是它限制了维，基本列的范围谓词不能生成维的范围谓词。但是，表达式的范围谓词是可以的，例如，where month(c1) between 4 and 6。这可以采用常规方式使用维的索引，起始键为 4 而停止键为 6。

要使此函数单调，请将年包括为月份的高位部分。存在对 INTEGER 内置函数的扩展以帮助根据日期定义单调表达式。INTEGER(date) 返回日期的整数表示法，可以分开查找年和月份的整数表示法。例如，INTEGER(date('2000/05/24')) 返回 20000524，因此 INTEGER(date('2000/05/24'))/100 = 200005。函数 INTEGER(date)/100 是单调的。

相似的，内置函数 DECIMAL 和 BIGINT 也具有扩展，所以可以派生单调函数。DECIMAL(timestamp) 返回时间戳记的十进制表示法，可以在单调表达式中使用它来派生月份、天、小时或分钟等等的增加的值。BIGINT(date) 返回日期的大整数表示法，类似于 INTEGER(date)。

只要可能，数据库管理器将在为表创建生成列或者根据维子句中的表达式创建维时确定表达式的单调性。特定函数被识别为保留单调性，例如，DATENUM( )、DAYS( ) 和 YEAR( )。并且，列和常量的各种算术表达式，例如，除法、乘法或加法是保留单调性的。当 DB2 确定表达式不保留单调性时，或者如果它不能确定这一点，那么该维仅支持对其基本列使用等价谓词。

MDC 和 ITC 表的装入注意事项

如果定期将数据装入到数据仓库中，那么使用多维集群 (MDC) 表会很有帮助。在 MDC 表中，装入会先复用表中先前清空的块，然后才扩展该表并添加新块以便装入余下的数据。

在删除了一组数据之后（例如，一个月的数据），可以使用 LOAD 实用程序来装入下一个月的数据，并且它可以复用在（已落实）删除之后已经清空的块。还可以选择将 MDC 转出功能与延迟清除配合使用。在落实转出（它也是一项删除操作）之后，各个块并未被释放，尚不能复用这些块。将调用后台进程以维护基于记录标识（RID）的索引。完成维护之后，就会释放各个块，从而可以复用这些块。对于插入时间集群 (ITC) 表，在扩展表之前，将尽可能复用未在使用的块。这包括已回收的块。ITC 表不支持转出。

当将数据装入 MDC 表中时，可以对输入数据排序，也可以不对它进行排序。如果未进行排序并且表具有多个维，请考虑执行下列操作:

- 提高 **util_heap_sz** 配置参数。

  为了提高 LOAD 实用程序在装入 MDC 表时的性能，应该增大 **util_heap_sz** 数据库配置参数值。当有更多内存可供该实用程序使用时，MDC 装入算法的性能会有所提高。这将减少在装入阶段执行数据集群时的磁盘 I/O 次数。如果使用 **LOAD** 命令并行装入若干个 MDC 表，那么必须相应地增大 **util_heap_sz**。
- 增大使用 **LOAD** 命令的 **DATA BUFFER** 子句给定的值。

  增大此值将影响单个装入请求。实用程序堆大小必须足够大，以满足多个并发装入请求的可能性。自 V9.5 起，当系统中存在更多可用内存时，**LOAD** 命令的 **DATA BUFFER** 参数的值可以临时性超出 **util_heap_sz** 设置值。
- 确保用于缓冲池的页大小与临时表空间的最大页大小相同。

  在装入阶段，将执行附加的记录以维护块映射。对于分配的每个扩展数据块，大约有两个附加的日志记录。为了确保性能良好，在设置 **logbufsz** 数据库配置参数的值时必须考虑此情况。

将数据装入到 MDC 或 ITC 表中时，存在以下限制:

- 不支持 **LOAD** 命令中的 **SAVECOUNT** 参数。
- 由于这些表管理它们自己的可用空间，所以不支持 `totalfreespace` 文件类型修饰符。
- MDC 表或 ITC 表需要 `anyorder` 文件类型修饰符。如果对 MDC 表或 ITC 表执行装入，但未指定 `anyorder` 修饰符，那么实用程序将显式启用该修饰符。

对 MDC 或 ITC 表使用 **LOAD** 命令时，将按以下方式处理唯一约束违例:

- 如果在执行装入操作前该表包含唯一键，并且将重复记录装入该表，那么将保留原始记录，并且将在删除阶段删除新记录。
- 如果执行装入操作前该表未包含唯一键，并且将唯一键和重复记录都装入该表，那么将只装入其中一个带有唯一键的记录，并且在删除阶段删除其他记录。

  注: 没有确切的技术可用来确定将要装入的记录以及将要删除的记录。

装入将从块边界开始，所以最好将它用于属于新单元的数据、用于表的初始填充以及用于将其他数据装入 ITC 表。

由于所有 MDC 和 ITC 表都有块索引，所以 MDC 和 ITC 装入操作始终包括构建阶段。

MDC 和 ITC 表的日志记录注意事项

与使用 RID 索引时的情况相比，使用维并因此而使用块索引时将减少索引维护和日志记录工作量。

仅当删除了整个块中的最后一个记录时，数据库管理器才会从块索引中除去 BID。此时还会记录此索引操作。同样，仅当将记录插入到新块中时，数据库管理器才会将 BID 插入到块索引中。该记录必须是逻辑单元的第一个记录或者是插入到当前已满的块的逻辑单元中。此时还会记录此索引操作。

因为块可以具有 2 到 256 页记录，所以此块索引维护和日志记录工作量相对较小。仍将记录对于表和 RID 索引的插入与删除操作。对于转出删除，不会记录被删除的记录。而是，将通过重新格式化页的某些部分使包含这些记录的页表现为空页。将会记录对重新格式化的部分所作的更改，但不会记录这些记录本身。

MDC 和 ITC 表的块索引注意事项

当定义 MDC 表的维时，将创建维块索引。此外，如果定义了多个维，那么还可创建组合块索引。如果只为 MDC 表定义了一个维，或者表是插入时间集群 (ITC)，那么仅创建一个块索引，它将同时充当维块索引和组合块索引。对于数据分区 MDC 或 ITC 表，系统会对表的 MDC 或 ITC 块索引进行分区。

同样，如果创建的 MDC 表具有针对列 A 和针对列 A 和列 B 的维，那么对列 A 创建维块索引，并对列 A 和列 B 创建维块索引。因为组合块索引是表中的所有维的块索引，所以列 A 和列 B 的维块索引也将充当组合块索引。

对于 MDC 表，在查询处理中组合块索引也用于访问具有特定维值的表中的数据。组合块索引中的键部分的顺序可能会影响它对于查询处理的使用或适用性。它的键部分的顺序由创建 MDC 表时使用的整个 ORGANIZE BY DIMENSIONS 子句中的列的顺序确定。例如，如果表是使用下列语句创建的:

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
      ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)
```

那么会对列 (c4, c3, c1, c2) 创建组合块索引。尽管 c1 在维子句中指定了两次，但是它仅在作为组合块索引的键部分时使用了一次，并且是以第一次发现它的顺序使用的。组合块索引中的键部分的顺序对于插入处理没有影响，但对于查询处理可能是有影响的。因此，如果更希望组合块索引具有列顺序 (c1,c2,c3,c4)，那么应使用以下语句来创建表:

```
CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int)
      ORGANIZE BY DIMENSIONS (c1, c2, (c3,c1), c4)
```

MDC 表的块索引

本主题说明如何使用块索引组织 MDC 表中的记录。

第 48 页的图 12 中显示的 MDC 表是以物理方式组织的，这样就可以将具有相同“Region”和“Year”值的记录组成为单独的块或扩展数据块。扩展数据块是磁盘上的一组连续页，所以将这些记录的组集群在物理上是连续的数据页上。每个表页都只属于一个块，所有块的大小都相同（即，它们的页数相同）。一个块的大小等于表空间的扩展数据块大小，以便使块边界与扩展数据块边界对齐。在此情况下，将创建两个块索引，一个针对“Region”维，另一个针对“Year”维。这些块索引包含仅指向表中的块的指

针。当对“Region”块索引进行扫描以找到“Region”等于“East”的所有记录时将找到两个合格的块。将在这两个块中找到“Region”等于“East”的所有记录（并且也只能找到这些记录），而且将这些记录集群在这两组连续页或扩展数据块上。同时，完全独立地扫描“Year”索引（1999 与 2000 之间的记录）将找到三个合格的块。对这三个块中的每个块进行数据扫描时都将返回在 1999 与 2000 之间的所有记录，并且也只能返回这些记录，并且您将发现这些记录集群在每个块的连续页上。

多维集群索引

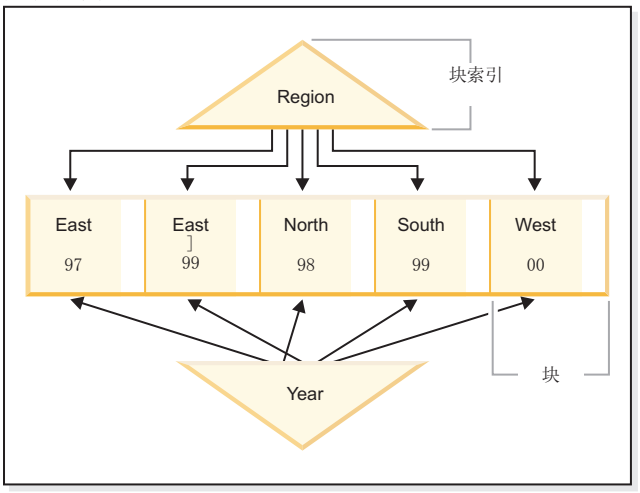


图 12. 多维集群表

除了对集群所作的这些改进之外，MDC 表还具有下列优点:

- 由于与基于记录的索引相比，由于块索引的大小是很小的，所以探测和扫描块索引要快
- 块索引和相应的数据组织允许更精细的“数据库分区忽略”或者有选择性地进行表访问
- 利用块索引的查询因为减小了索引大小、优化了对块的预取并且可以保证相应数据的集群而受益
- 对于某些查询，可以减少锁定和谓词求值
- 块索引用于日志记录和维护方面的开销很少，因为仅当将第一条记录添加至块或从块中除去最后一条记录时才需要更新它们
- 转入的数据可以复用先前转出的数据留下的连续空间。

注: 即使只是使用单个维定义的 MDC 表也可以因这些 MDC 属性而受益，并且可以是具有集群索引的常规表的可行备用。应该根据许多因素来作出此决定，这些因素包括: 组成工作负载的查询以及表中数据的性质和分布。请参阅第 35 页的『选择 MDC 表维』和第 41 页的『创建 MDC 或 ITC 表时的注意事项』。

创建表时，可以将一个或多个键指定为集群数据时所采用的维。每个 MDC 维都可以包括类似于常规索引键的一列或多列。将为每个指定的维自动创建*维块索引*，并且优化器将使用它根据每个维快速并有效地访问数据。还将自动创建*组合块索引*，它包含所有维中的所有列，并且将用来在插入和更新活动期间维护数据的集群。仅当单个维尚未包含所有的维键列时，才创建组合块索引。优化器还可以选择组合块索引来有效地访问满足一部分列维或所有列维中的值的数据。

注: 此索引在处理查询期间的用途取决于它的键部分的顺序。键部分顺序是由解析器在对 CREATE TABLE 语句的 ORGANIZE BY DIMENSIONS 子句中指定的维进行解析时遇到的列顺序来确定的。有关更多信息，请参阅 第 47 页的『MDC 和 ITC 表的块索引注意事项』。

块索引与常规索引在结构上是相同，只不过它们指向的是块而不是记录。块索引比常规索引小的倍数为每页平均记录数乘以块大小所得到的乘积。一个块中的页数等于表空间的扩展数据块大小，它的范围是 2 到 256 页。页大小可以为 4 KB、8 KB、16 KB 或 32 KB。

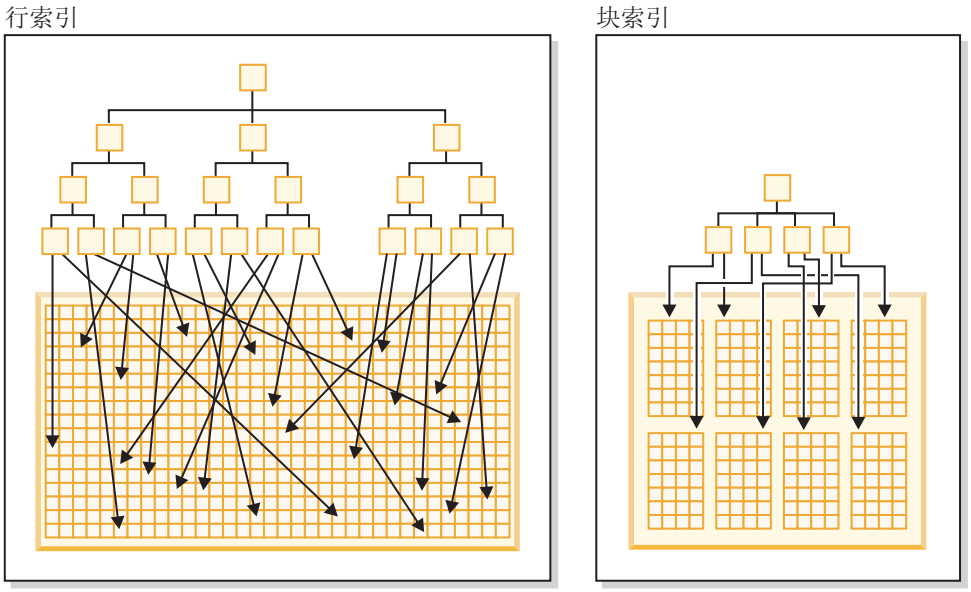


图 13. 行索引与块索引的区别

如图 13 中所示，与每一行具有单个条目相比，在块索引中，每个块都有单个索引条目。因此，块索引显著减少了磁盘的使用，并且极大地提高了数据访问速度。

在 MDC 表中，维值的每个唯一组合组成逻辑单元，它在物理上是由一个或多个页的块组成的。逻辑单元将仅使足够多的块与它相关联，以便存储具有该逻辑单元的维值的记录。如果表中没有具有特定逻辑单元的维值的记录，那么将不会为该逻辑单元分配块。包含具有特定维键值的数据的块的集合称为*片*。

MDC 表可以是分区表。针对分区 MDC 表的块索引可以是非分区索引或分区索引:

- 对于使用 DB2 V9.7 FP1 或更高发行版创建的分区 MDC 表，表的块索引是分区索引。
- 对于使用 DB2 V9.7 或更早发行版创建的分区 MDC 表，表的块索引是非分区索引。

将数据库升级到 DB2 V9.7 FP1 或更高发行版之后，即会支持分区块索引。

方案: 多维集群 (MDC) 表

通过一个方案说明如何使用 MDC 表: 我们假设一个名为“Sales”的 MDC 表，它记录有关国内零售商的销售数据。该表根据“YearAndMonth”和“Region”维进行集群。表中的记录是以块为单位存储的，块包含磁盘上足够的连续页以填充扩展数据块。

在图 14 中，块由矩形表示，并且根据表中分配的扩展数据块的逻辑顺序进行编号。图表中的网格表示这些块的逻辑数据库分区，并且每个正方形表示一个逻辑单元。网格中的列和行表示特定维的片。例如，“Region”列中包含“South-central”值的所有记录都可以在由网格中的“South-central”列定义的片中包含的块中找到。实际上，此片中的每个块也只包含“Region”字段中具有“South-central”的记录。这样，当且仅当一个块包含在“Region”字段中具有“South-central”的记录时，该块才包含在此片或网格的列中。

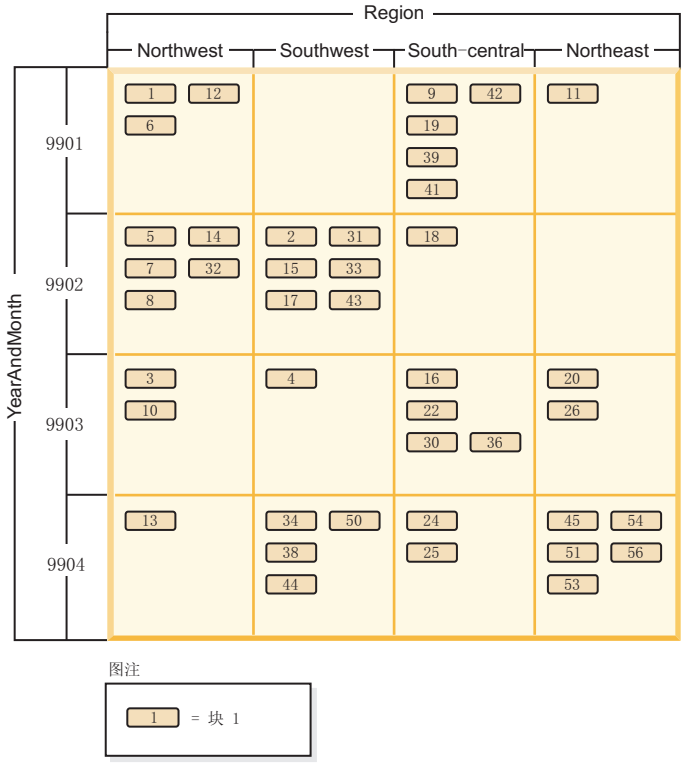


图 14. 具有“Region”和“YearAndMonth”维的称为 Sales 的多维表

为了便于确定哪些块组成片，或者哪些块包含具有特定维键值的所有记录，在创建表时将自动为每个维创建维块索引。

在第 51 页的图 15 中，一个维块索引是根据“YearAndMonth”维创建的，另一个维块索引是根据“Region”维创建的。每个维块索引的结构方式与传统 RID 索引的结构方式相同，但在叶级，这些键指向块标识 (BID) 而不是记录标识 (RID)。RID 通过物理页号和槽号（找到记录的页面上的槽）来标识表中记录的位置。BID 通过该扩展数据块的第一页的物理页号和虚槽 (0) 来表示块。因为块中的所有页在物理上是从该页开始连续的，并且我们知道该块的大小，所以可以使用此 BID 来找到该块中的所有记录。

片，或包含带有在维中具有特定键值的记录的页的一组块，将由该键值的 BID 列表在关联维块索引中表示。

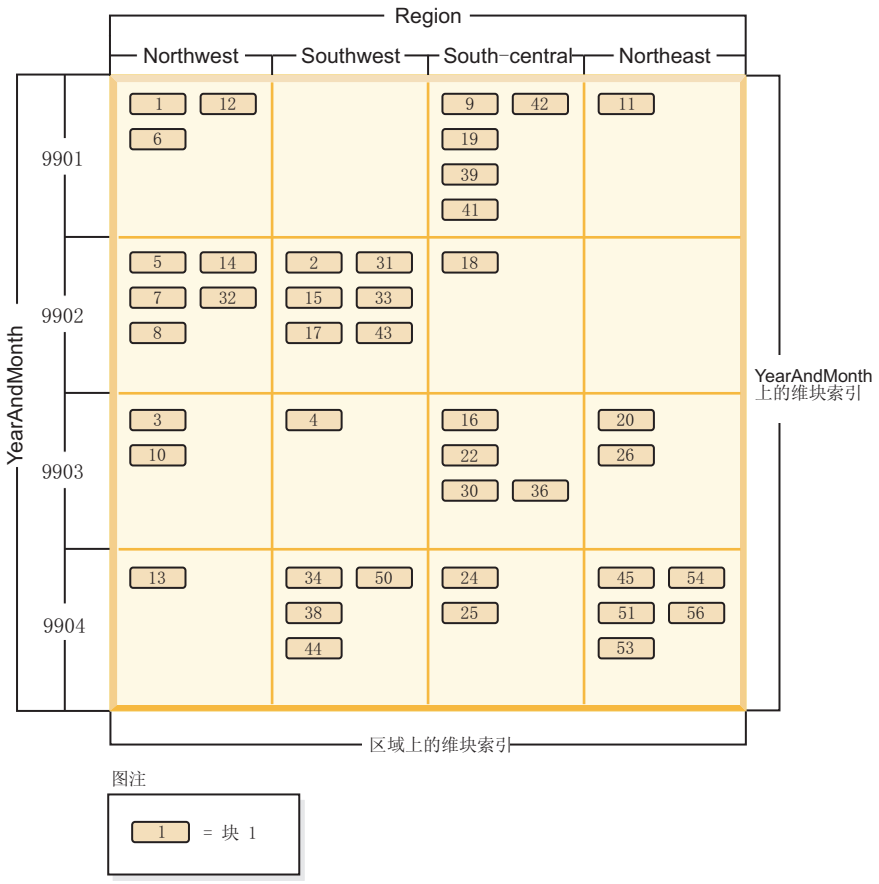


图 15. 显示维块索引且具有“Region”和“YearAndMonth”维的 Sales 表

图 16 显示“Region”的维块索引中的键如何出现。键由键值（即“South-central”）和 BID 列表组成。每个 BID 包含一个块位置。在图 16 中，列示的块号与在 Sales 表的网格中发现的“South-central”片相同（请参阅第 50 页的图 14）。

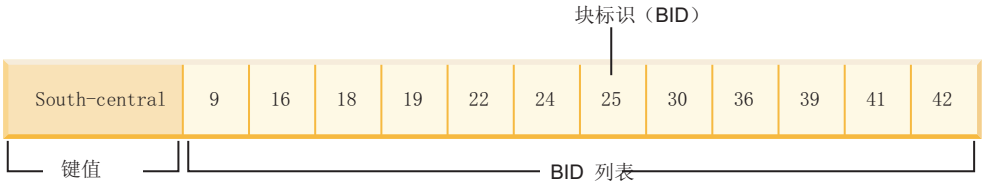


图 16. “Region”的维块索引中的键

类似的，要找到包含“YearAndMonth”维为“9902”的所有记录的块的列表，应在“YearAndMonth”维块索引中查找此值，如第 52 页的图 17 中所示。

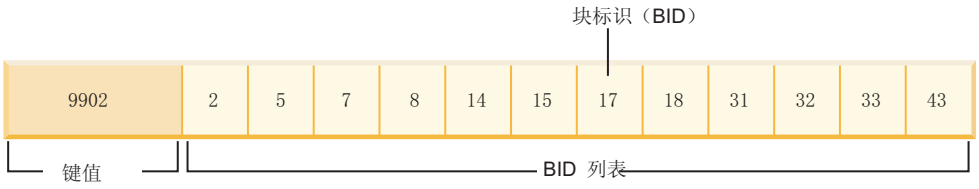


图 17. “YearAndMonth”的维块索引中的键

MDC 表的块索引和查询性能

扫描 MDC 表的任何块索引将提供集群数据访问，这是因为每个块标识 (BID) 都与保证包含具有指定维值的数据的表中的一组连续页相对应。此外，还可以通过维或片的块索引来单独访问维或片，而不会损害任何其他维或片的集群因子。这就提供了多维集群的多维性。

利用块索引访问的查询在很多方面提高了性能。

- 因为块索引比常规索引小很多，所以块索引扫描的效率很高。
- 当使用块索引时，预取数据页不依赖于顺序检测。DB2 数据库管理器在索引中向前查找，使用大块 I/O 将数据块预取到内存中并确保在表中访问数据页时扫描不会产生 I/O 成本。
- 表中的数据集群在连续页上，这样就优化了 I/O 并将结果集定位到表的选择的部分。
- 如果在基于块的缓冲池的块大小等于扩展数据块大小的情况下使用该缓冲池，那么将把 MDC 块从磁盘上的连续页预取到内存的连续页中，这样就进一步增强了集群对性能所起的积极作用。
- 每个块中的记录都是通过对它的数据页使用最小关系扫描来检索的，这种检索方法通常比通过基于 RID 的检索扫描数据更快。

查询可以使用块索引来缩小表中具有特定维值或某一范围内的值的部分。这就提供了“数据库分区忽略”（即，块忽略）的精确格式。这样可使表具有更好的并行性，因为其他查询、装入、插入、更新和删除可访问表中的其他块，而不需要与此查询的数据集交互。

如果 Sales 表根据三个维进行集群，那么还可以使用个别维块索引来查找包含满足对表的所有维的子集的查询的记录的一组块。如果表具有“YearAndMonth”、“Region”和“Product”维，那么可以认为它是逻辑立方体，如第 53 页的图 18 中所示。

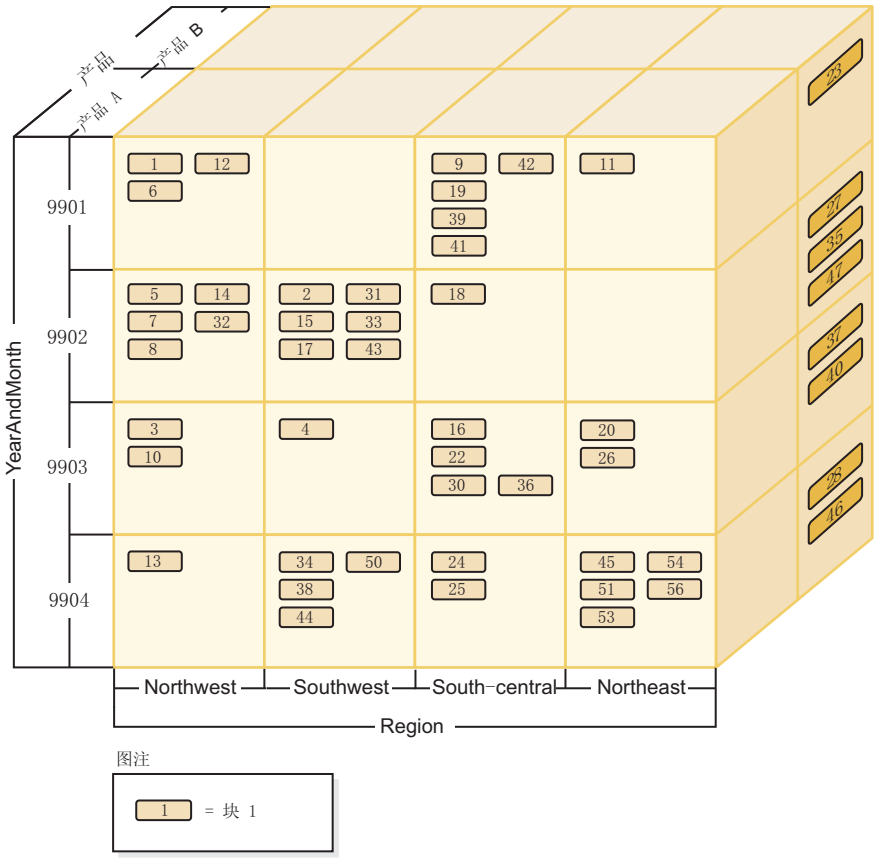


图 18. 具有“Region”、“YearAndMonth”和“Product”维的多维表

将对图 18 中显示的 MDC 表创建四个块索引，为“YearAndMonth”、“Region”和“Product”维每个都创建一个块索引；而另一个块索引将所有这些维列作为它的键。要检索“Product”等于“ProductA”并且“Region”等于“Northeast”的所有记录，数据库管理器将首先从“Product”维块索引中搜索 ProductA 键。（请参阅图 19。）然后，数据库管理器通过在“Region”维块索引中查找“Northeast”键来确定包含“Region”等于“Northeast”的所有记录的块。（请参阅图 20。）

产品 A	1	2	3	· · ·	11	· · ·	20	22	24	25	26	30	· · ·	56

图 19. “Product”的维块索引中的键

Northeast	11	20	23	26	27	28	35	37	40	45	46	47	51	53	54	56

图 20. “Region”的维块索引中的键

可以通过使用逻辑 AND 和逻辑 OR 运算符来组合块索引扫描，并且获得的要扫描的块列表还提供集群数据访问。

以先前的示例为例，要查找包含具有这两个维值的所有记录的一组块，就必须找到这两个片的交集。这是通过对两个块索引键中的 BID 列表使用逻辑 AND 操作来完成的。常用的 BID 值有 11、20、26、45、54、51、53 和 56。

以下示例说明如何对块索引使用逻辑 OR 操作来满足具有涉及到两个维的谓词的查询。图 21 假定一个 MDC 表具有两个维：“Colour”和“Nation”。目标是检索 MDC 表中满足以下条件的所有记录：“Colour”为“blue”或者“Nation”名称为“USA”。

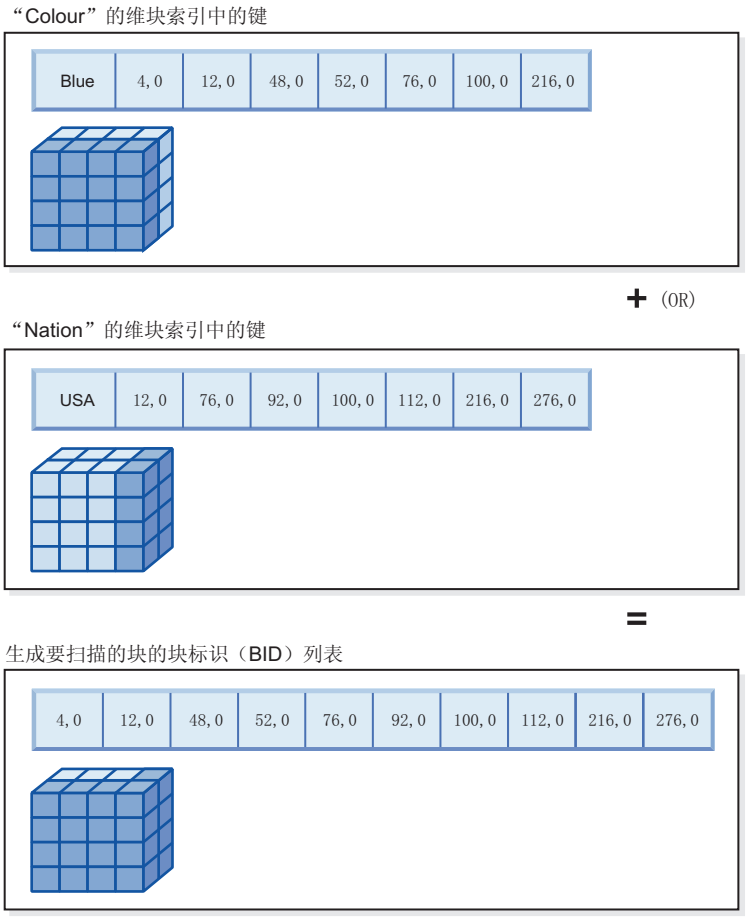


图 21. 可以如何对块索引使用逻辑 OR 操作

本图说明了如何组合两个单独的块索引扫描的结果以确定满足谓词限制的值的范围。（数目指示记录标识（RID）和槽字段。）

根据 SELECT 语句中的谓词，完成了两个单独的维块索引扫描；一个是针对 blue 片，另一个是针对 USA 片。在内存中完成了逻辑 OR 操作，以便找到两个片的并集并确定在这两个片中找到的组合块集合（包括除去重复的块）。

一旦数据库管理器具有要扫描的块列表，数据库管理器就可以对每个块执行最小关系扫描。可以完成块的预取，并且对于每个块，此操作将只涉及到一个 I/O，因为每个块在磁盘上作为扩展数据块存储并且可以作为一个单元读取到缓冲池中。如果需要对数据应用谓词，那么只需要对块中的一个记录应用维谓词，因为将保证块中的所有记录都具有相同的维键值。如果存在其他谓词，那么数据库管理器只需要对块中的其余记录检查这些谓词。

MDC 表还支持基于 RID 的常规索引。可以通过使用逻辑 AND 操作或逻辑 OR 操作来将 RID 和块索引与其他索引组合在一起。块索引为优化器提供了可供选择的其他存取方案，但是您仍然可以使用传统存取方案（RID 扫描、连接、表扫描和其他方案）。在用于特定查询的所有其他可能的存取方案中，优化器将采用块索引方案，并且将选择开销最低的方案。

“DB2 设计顾问程序”可以帮助对 MDC 表建议基于 RID 的索引，或者对表建议 MDC 维。

在 INSERT 操作期间自动维护集群

通过使用组合块索引确保了自动维护 MDC 表中的数据集群。它用来在 INSERT 操作期间根据表维动态管理和维护数据的物理集群。

对于包含记录的表的每个逻辑单元，此组合块索引中只有一个键。因此，在 INSERT 期间使用此块索引来快速高效地确定表中是否存在逻辑单元，并且仅当表中存在逻辑单元时，才能准确地确定哪些块包含具有该单元的一组特定维值的记录。

当进行插入时:

- 将检查组合块索引，以找到与要插入的记录的维值相对应的逻辑单元。
- 如果在索引中找到了该逻辑单元的键，那么它的块标识 (BID) 列表将提供表中具有该逻辑单元的维值的各个块的完整列表。（请参阅图 22。）这就限制了表的用来搜索要插入记录的空间的扩展数据块数目。
- 如果在索引中找不到该逻辑单元的键，或者如果包含这些值的扩展数据块已满，那么为该逻辑单元指定新块。如果可能，在使用页的另一个新扩展数据块（新块）来扩展表之前，先复用该表中的空块。

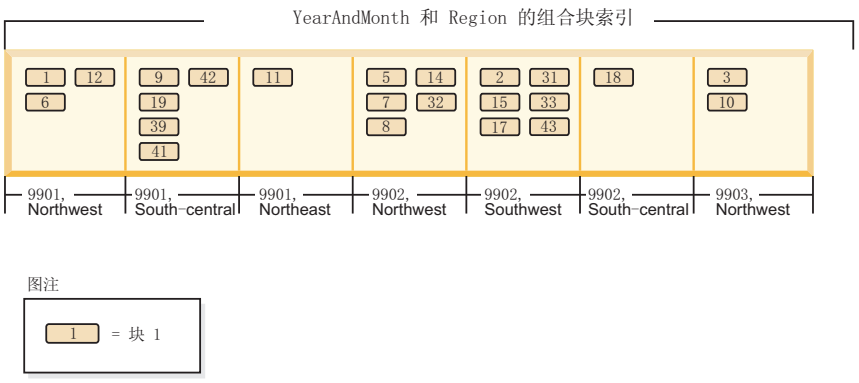


图 22. “YearAndMonth”和“Region”的组合块索引

已经保证了在包含并且仅包含具有特定维值的所有记录的一组块中将找到具有这些值的数据记录。这些块是由磁盘上的连续页组成的。因此，将按顺序访问这些记录，并且将提供集群。通过确保将记录从具有该记录的维值的单元中插入到块中来随时间推移自动维护此集群。当逻辑单元中的现有块已满时，将复用空块，或者将分配新块并将它添加至该逻辑单元的一组块。当一个块中没有数据记录时，将从块索引中除去块标识 (BID)。这将使该块不再与任何逻辑单元值相关联，以便将来另一个逻辑单元可以复用该块。因此，将根据需要从表中动态添加和除去单元以及与它们相关联的块索引

条目，以便只容纳表中存在的数据。组合块索引用来管理这种情况，因为它将逻辑单元值映射至包含具有这些值的记录的那些块。

因为集群是按这种方式自动维护的，所以从不需要重组 MDC 表来重新集群数据。但是，仍然可以使用重组来回收空间。例如，如果单元具有许多稀疏块（数据只能满足很少的块），或者如果表具有许多指针溢出对，那么重组表时就会将属于每个逻辑单元的记录压缩到需要的最小块数中，并且将除去指针溢出对。

以下示例说明了可以如何将组合块索引用于查询处理。如果想要查找第 55 页的图 22 的表中“Region”为“Northwest”且“YearAndMonth”为“9903”的所有记录，那么数据库管理器将在组合块索引中查找键值 9903, Northwest，如图 23 中所示。键由键值（即“9903, Northwest”）和 BID 列表组成。可以看到列示的 BID 仅有 3 和 10，并且实际上 Sales 表中只有两个块包含具有这两个特定值的记录。

图 23. “YearAndMonth”和“Region”的组合块索引中的键

为了说明在插入期间如何使用组合块索引，我们以插入具有维值 9903 和 Northwest 的另一条记录为例。数据库管理器将在组合块索引中查找此键值，并查找第 3 个和第 10 个块的 BID。这些块包含具有这些维键值的仅有全部记录。如果具有可用空间，那么数据库管理器将把新记录插入到其中一个块中。如果在这些块中的所有页上都没有空间，那么数据库管理器将为表分配新块或使用表中先前清空的块。注意，在本示例中，表当前没有使用第 48 个块。数据库管理器将记录插入到该块中，并通过将该块的 BID 添加至组合块索引和每个维块索引来使此块与当前逻辑单元相关联。请参阅图 24 以获取有关添加第 48 个块之后的维块索引的键的说明。

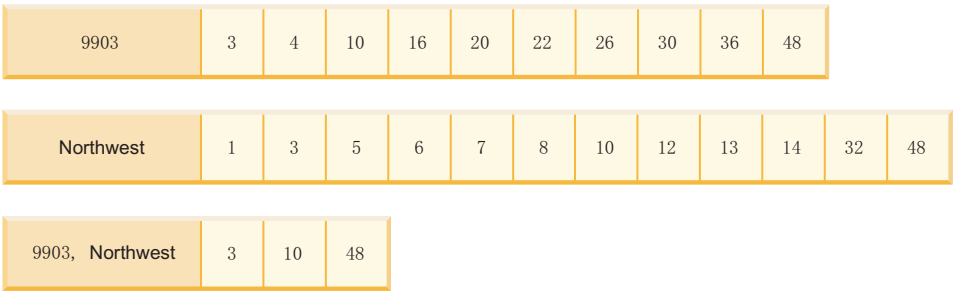


图 24. 添加第 48 个块之后维块索引中的键

MDC 和 ITC 表的块映射

对于 MDC 表，当清空某个块时，将通过从块索引中除去该块的 BID 来使解除它与当前逻辑单元值的关联。然后，该块可被另一个逻辑单元复用。对于 ITC 表，所有块与单个单元相关联。释放单元中的某个块就意味着后续插入操作可以复用该块。这种复用将减少使用新块来扩展表的需要。

当需要新块时，需要快速找到先前被清空的块，而不必搜索表来找到这些块。

块映射是用来便于找到 MDC 或 ITC 表中的空块的结构。块映射是作为单独的对象存储的:

- 在 SMS 中，作为单独的 .BKM 文件
- 在 DMS 中，作为对象表中的新对象描述符。

块映射是包含表的每个块的条目的一个数组。每个条目组成块的一组状态位。

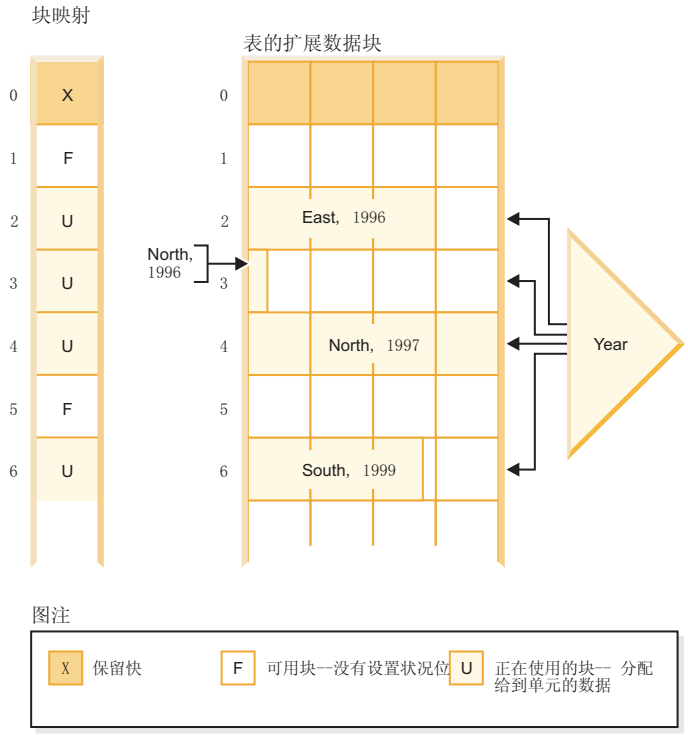


图 25. 块映射的工作方式

在图 25 的左边显示具有表中每个块的不同条目的块映射数组。右边显示了正在如何使用表的每个扩展数据块: 某些扩展数据块是可用的，而大多数扩展数据块正处于使用状态，并且仅在块映射中标记为正在使用的块中找到了记录。为了简单起见，在图中只显示两个维块索引的其中一个。

注:

1. 块索引中存在仅指向块映射中标记为 IN USE 的那些块的指针。
2. 第一个块是保留块。此块包含表的系统记录。

通过扫描块映射中的 FREE 块（即，未设置任何位的那些块），很容易找到可用块以便在单元中使用。

表扫描还使用块映射来仅访问当前包含数据的扩展数据块。根本不需要在表扫描中包括任何未在使用的扩展数据块。为了便于说明，此示例（图 25）中的表扫描将跳过第一个保留的扩展数据块和随后的空扩展数据块从表中的第三个扩展数据块（扩展数据块 2）开始，扫描表中的块 2、3 和 4，跳过下一个扩展数据块（不会涉及该扩展数据块的数据页），然后从那里继续扫描。

从 MDC 和 ITC 表中删除

在 MDC 或 ITC 表中删除记录时，如果该记录不是块中的最后一条记录，那么数据库管理器只删除该记录并从在该表上定义的任何基于记录的索引中除去其 RID。

当删除操作除去块中的最后一条记录时，数据库管理器将释放该块。将通过更改 IN_USE 状态位并从所有块索引中除去该块的 BID 来释放该块。如果还存在基于记录的索引，那么也会从这些索引中除去 RID。

注: 因此，仅当块被清空时，才会对整个块除去一次块索引条目，而不是在基于记录的索引中每删除一行时都会除去一次。

对 MDC 和 ITC 表的更新

在 MDC 表中，更新非维值是在适当位置完成的，正如对常规表更新非维值一样。如果更新 MDC 或 ITC 表中的一条记录时导致该记录长度增加并且不再适合该页面，那么将找到具有足够空间的另一页。

在同一个块中开始搜索此新页。如果该块中没有空间，那么使用插入新记录的算法来查找逻辑单元中具有足够空间的页。除非在单元中找不到空间并且需要将新块添加至该单元，否则不需要更新块索引。

对于 ITC 表，如果该块中没有足够的空间来放置所更新的行，那么会将该行移至新块。此移动会导致该行不再与相近时间插入的行集群在一起。

仅 MDC 表的注意事项

更新维值被视作删除当前记录之后插入已更改的记录，因为该记录将更改它所属于的逻辑单元。如果删除当前记录导致块被清空，那么需要更新块索引。类似地，如果插入新记录要求将它插入到新块中，那么需要更新块索引。

MDC 表被视作任何现有表；即，可以对这些表定义触发器、引用完整性、视图和具体化查询表。

MDC 和 ITC 表的注意事项

仅当将第一条记录插入到块中或者从块中删除最后一条记录时才需要更新块索引。因此，与块索引相关联的用于维护和日志记录的索引资源和需求远远小于常规索引。对于在其他情况下是常规索引的每个块索引，将显著减少维护和日志记录资源和需求。

重新使用最近清空的块时，必须对该块使用条件 Z 锁定以确保 UR 扫描程序当前未对其进行扫描。

多维集群和插入时间集群扩展数据块管理

通过重组多维集群 (MDC) 或插入时间集群 (ITC) 表，可以从该表中释放数据扩展数据块。

在 MDC 和 ITC 表中，块映射会跟踪属于表的所有数据扩展数据块，并且指示哪些块和扩展数据块上包含数据以及哪些块和扩展数据块上没有包含数据。具有数据的块标记为“正在使用”。每当对 MDC 或 ITC 表执行删除或对 MDC 表执行转出时，具有块映射的块条目不再标记为“正在使用”，而是被释放以供该表复用。

但是，表空间中的其他对象无法使用这些块和扩展数据块。可以通过重组表来从表中释放这些可用数据扩展数据块。可使用带 **RECLAIM EXTENTS** 参数的 **REORG TABLE** 命令，以使回收空间时该表对用户可用且处于联机状态。从 MDC 或 ITC 表释放扩展数据块仅受 DMS 表空间中的表支持。

REORG TABLE 命令使用 **RECLAIM EXTENTS** 参数来释放 MDC 或 ITC 表专用的扩展数据块，并且使该空间可供表空间内其他数据库对象使用。

该选项还允许您控制在释放扩展数据块期间对 MDC 或 ITC 表的并行访问。写访问权为缺省访问权，还可选择读访问权和无访问权来控制并行访问。

如果 MDC 或 ITC 表还是范围分区表或数据库分区表，那么缺省情况下，会在所有数据或数据库分区上释放扩展数据块。通过指定分区名称（对于数据分区）或分区号（对于数据库分区），可以运行该命令以仅在特定分区上释放扩展数据块。

REORG TABLE 命令和 db2Reorg API 均可用于释放扩展数据块。

自动支持可用于数据库的自动维护活动的释放扩展数据块部分。要启用重组以释放 MDC 或 ITC 表中的扩展数据块，**auto_maint**、**auto_tbl_maint** 和 **auto_reorg** 数据库配置参数必须全部具有值 ON。可以通过使用命令行来配置这些数据库配置参数。在启用了数据库分区功能的 DB2 实例上，必须在目录分区上发出对这些参数的配置。

维护策略控制何时执行 MDC 或 ITC 表的自动重组以释放未使用的扩展数据块。DB2 系统存储过程 AUTOMAINT_SET_POLICY 和 AUTOMAINT_SET_POLICYFILE 用来设置此维护策略。XML 用来存储自动维护策略。

表分区和多维集群表

在同时是多维集群表和数据分区表的表中，可以同时在表分区的范围分区规范和多维集群 (MDC) 键中使用列。与只单独使用多维集群或分区功能相比，同时是多维集群表和分区表的表可以获取较详细的数据分区和块消除。

在许多应用中将 MDC 键列指定为不同于对表进行分区的列很有用。应该注意的是，表分区是多列的，而 MDC 是多维的。

主流 DB2 数据仓库的特征

下列建议主要针对 DB2 V9.1 中新增的典型主流仓库。假定下列特征:

- 数据库在多台机器或多个 AIX 逻辑分区上运行。
- 使用分区数据库环境（使用 DISTRIBUTE BY HASH 子句来创建表）。
- 有 4 到 50 个数据分区。
- 考虑其 MDC 和表分区的表是主要事实表。
- 表的行数为 100,000,000 至 100,000,000,000。
- 在各种时间范围装入新数据: 每夜、每周和每月。
- 每日接受量为 1 万到 1 亿条记录。
- 数据量变化: 最多的一个月是最少的月的 5 倍。同样，最大维数（生产线，区域）具有 5 倍大小范围。
- 获取 1 到 5 年的详细数据。
- 每月或每个季度转出到期数据。

- 表使用大范围的查询类型。但是，相对于 OLTP 工作负载来说，该工作负载通常是具有下列特征的分析查询:
  - 较大的结果集，最多有 2 百万行
  - 大多数或全部查询都命中视图，而不是基本表
- SQL 子句按范围（BETWEEN 子句）、列表中的项等选择数据。

主流 DB2 V9.1 数据仓库事实表的特征

一个典型仓库事实表可能采用以下设计:

- 在 Month 列中创建数据分区。
- 为转出的每个时间段（例如，1 个月和 3 个月）定义数据分区。
- 在 Day 和 1 到 4 个其他维的基础上创建 MDC 维。典型的维有: 生产线和区域。
- 所有数据分区和 MDC 集群都分布在所有数据库分区中。

MDC 和表分区具有一些相同的好处。下表列示组织中的潜在需求并根据先前确定的特征确定建议的组织方案。

表 7. 将表分区与 MDC 表配合使用

问题	建议的方案	建议
转出期间的数据可用性	表分区	使用 DETACH PARTITION 子句来转出大量数据，并且只出现最少中断。
查询性能	表分区和 MDC	MDC 最适合用来查询多个维。表分区通过数据分区消除提高性能。
最少重组	MDC	MDC 维护集群，从而减少进行重组的必要性。
在传统脱机窗口中转出一个月或更长时间的数据	表分区	数据分区可以完全解决此需求。MDC 不起任何作用，并且仅 MDC 并不适合。
在短时间脱机窗口（小于 1 分钟）期间转出一个月或更长时间的数据	表分区	数据分区可以完全解决此需求。MDC 不起任何作用，并且仅 MDC 并不适合。
转出一个月或更长时间的数据，并同时在不损失任何服务的情况下使表对于提交查询的企业用户完全可用。	MDC	MDC 只能解决一部分此需求。由于表处于脱机状态的时间段太短，表分区并不适合。
每天装入数据（**LOAD** 或 **INGEST** 命令）	表分区和 MDC	此时 MDC 具有很多好处。表分区具有增量的好处。
“连续”装入数据（具有 ALLOW READ ACCESS 或 **INGEST** 命令的 **LOAD** 命令）	表分区和 MDC	此时 MDC 具有很多好处。表分区具有增量的好处。
“传统 BI”查询的查询执行性能	表分区和 MDC	MDC 特别适合用来查询立方体/多个维。表分区通过分区消除提高性能。

表 7. 将表分区与 MDC 表配合使用 (续)

问题	建议的方案	建议
通过消除进行重组的必要性或降低执行任务所产生的不良影响，使重组所带来的不良影响降到最低。	MDC	MDC 维护集群，从而减少进行重组的必要性。如果使用 MDC，那么数据分区不提供增量好处。但是，如果不使用 MDC，那么表分区通过在分区级别维护一些粗粒度集群会有助于减少重组的必要性。

示例 1:

考虑一个具有键列 YearAndMonth 和 Province 的表。一种合理的规划此表方法是按日期进行分区，每 2 个月添加一个数据分区。此外，还可以按 Province 进行组织，以便任何两个月日期范围内的特定省份的所有行集群在一起，如第 28 页的图 6 中所示。

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

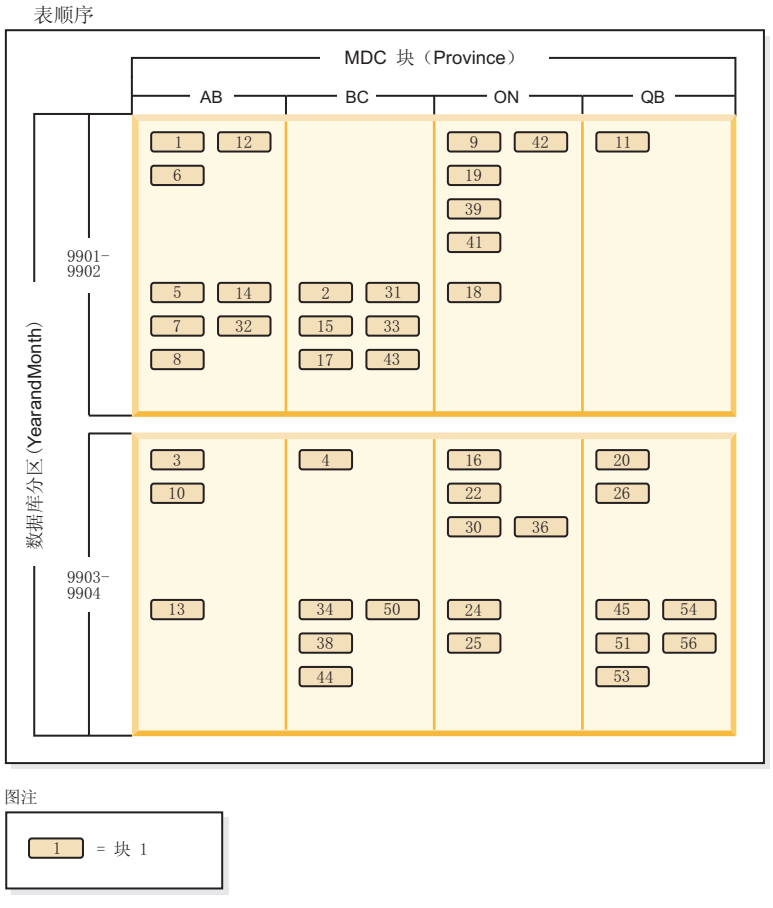


图 26. 按 YearAndMonth 分区并按 Province 组织的表

示例 2:

通过将 YearAndMonth 添加至 ORGANIZE BY DIMENSIONS 子句，可以获得较高的详细程度，如第 29 页的图 7 中所示。

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

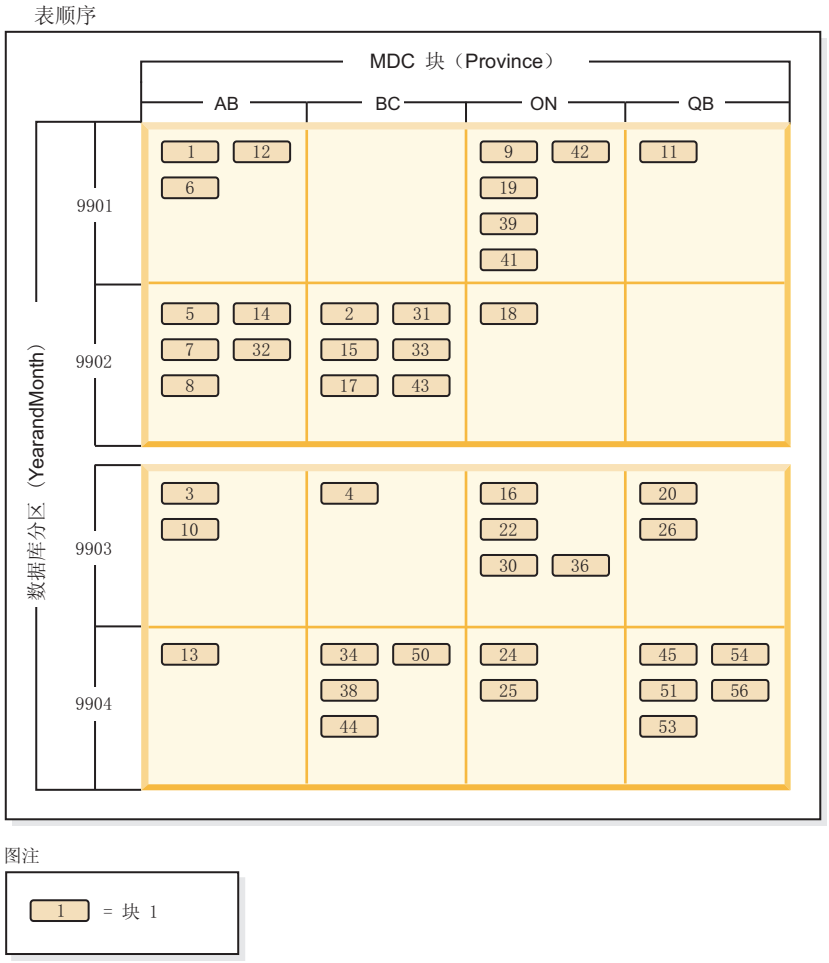


图 27. 按 YearAndMonth 分区并按 Province 和 YearAndMonth 组织的表

如果分区导致每个范围内只有单个值，那么通过在 MDC 键中包括表分区列不能获得任何好处。

注意事项

- 与基本表相比，MDC 表和分区表都需要一些存储器。这些存储器需求是附加的，但相对于所带来的好处来说，它们是合理的。
- 如果选择不在分区数据库环境中组合表分区和 MDC 功能，那么在您可以非常自信地预计数据分布情况时（通常也就是此处讨论的系统类型情况），使用表分区是最好的选择。否则，应考虑 MDC。
- 对于使用 DB2 V9.7 FP1 或更高发行版创建的数据分区 MDC 表，表的 MDC 块索引是分区索引。对于使用 DB2 V9.7 或更早发行版创建的数据分区 MDC 表，表的 MDC 块索引是非分区索引。

第 4 章 并行数据库系统

并行性

任务（例如，数据库查询）的各个组件可并行运行以大幅提高性能。任务的性质、数据库配置和硬件环境都确定 DB2 数据库产品将如何以并行方式执行任务。

这些因素是互相关联的。在决定数据库的物理和逻辑设计方案时，请一起考虑所有这些因素。DB2 数据库系统支持下列类型的并行性:

- I/O
- 查询
- 实用程序

输入/输出并行性

当一个表空间有多个容器时，数据库管理器可以使用*并行 I/O*。并行 I/O 指的是同时处理对两个或多个 I/O 设备的写入或读取；这能够显著地改进吞吐量。

查询并行性

有两种类型的查询并行性: 查询间并行性和查询内并行性。

*查询间并行性*是指数据库同时接受多个应用程序查询的能力。每个查询以独立于其他查询的方式运行，但数据库管理器同时运行所有查询。DB2 数据库产品始终支持这种类型的并行性。

*查询内并行性*是指使用分区内并行性和/或分区间并行性来同时处理单个查询的各部分。

分区内并行性

*分区内并行性*是指将一个查询分为多个部分的能力。某些 DB2 实用程序也执行此类型的并行性。

分区内并行性将通常认为是单个数据库的操作（例如，创建索引、装入数据库或 SQL 查询）细分成多个部分，其中的大部分或全部操作可以在单个数据库分区内以并行方式运行。

第 64 页的图 28 显示了一个查询，它被分为可并行运行的三个部分，返回结果的速度比按串行方式运行该查询的速度快。这几部分互为副本。要使用分区内并行性，必须适当地配置数据库。您可以选择并行度，或由系统为您选择。并行度表示一个查询中并行运行的部分数。

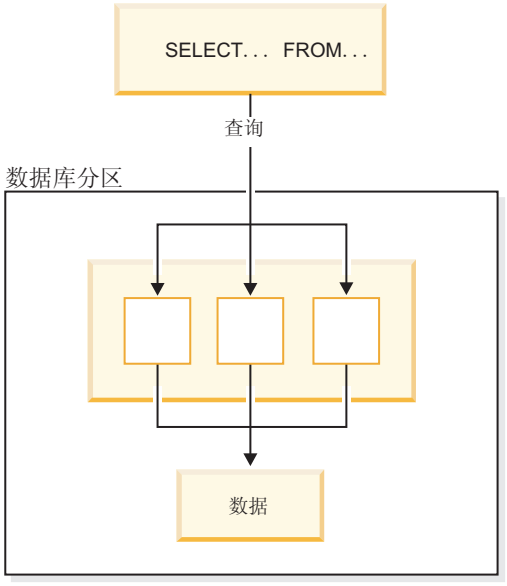


图 28. 分区内并行性

分区间并行性

*分区间并行性*是指将一个查询分为多个部分并将这几部分置于一个分区数据库的多个分区（位于一台或多台机器上）上的能力。该查询以并行方式运行。某些 DB2 实用程序也执行此类型的并行性。

分区间并行性将通常认为是单个数据库操作（例如，创建索引、装入数据库或 SQL 查询）的操作细分成多个部分，其中的大部分或全部操作可以在一台或多台机器上的一个分区数据库的多个分区中以并行方式运行。

第 65 页的图 29 显示了一个查询，它被分为可并行运行的三个部分，返回结果的速度比在单个数据库分区上按串行方式运行该查询的速度快。

并行度在很大程度上取决于您创建的数据库分区数和您定义数据库分区组的方式。

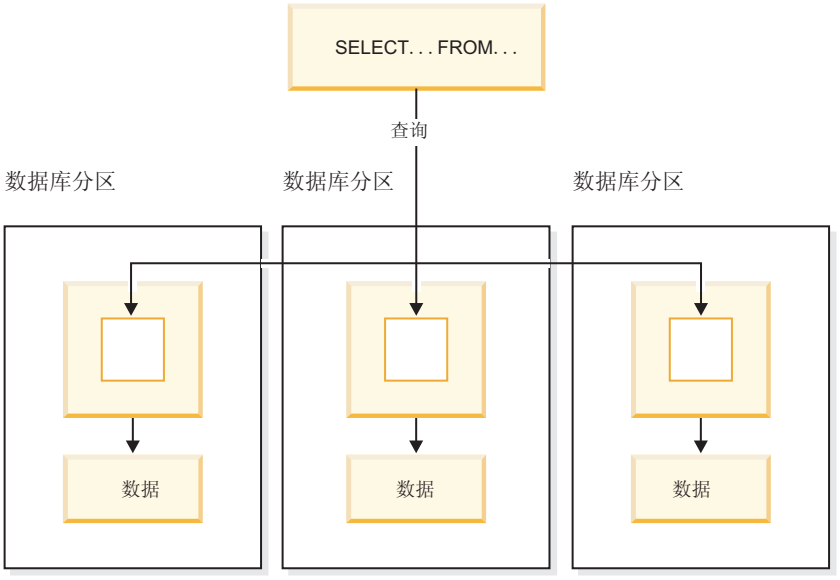


图 29. 分区间并行性

同时使用分区内和分区间并行性

可以同时使用分区内并行性和分区间并行性。此组合提供了两种并行性，这也使处理查询的速度显著加快。

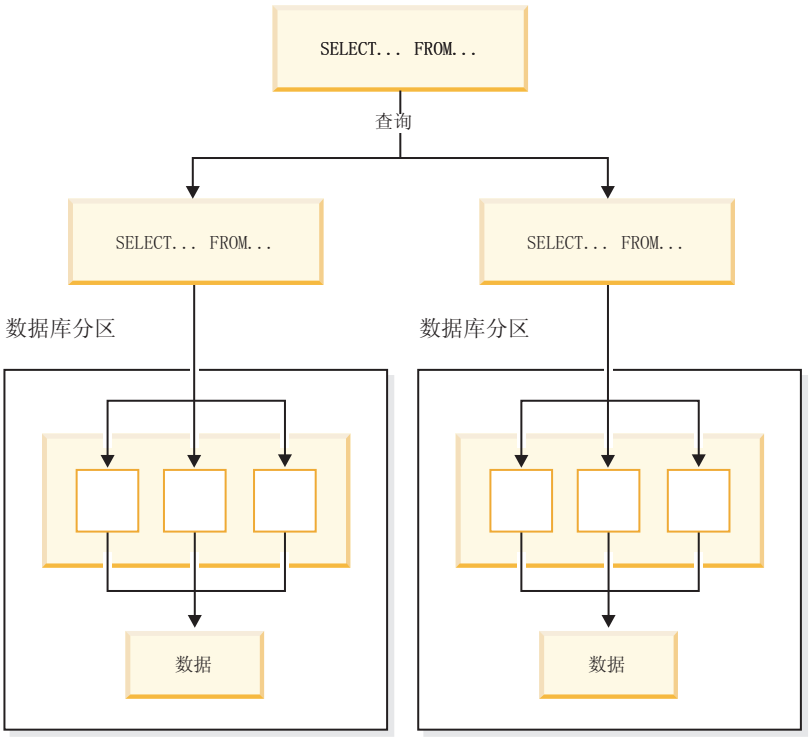


图 30. 同时使用分区间并行性和分区内并行性

实用程序并行性

DB2 实用程序可以利用分区内并行性。它们还可以利用分区间并行性；前提是存在多个数据库分区，而实用程序在每个数据库分区内以并行方式运行。

装入实用程序可以利用分区内并行性和 I/O 并行性。装入数据是一个大量使用 CPU 的任务。装入实用程序利用多个处理器来执行如解析和格式化数据这类任务。它也可使用并行 I/O 服务器来以并行方式将数据写入容器中。

在分区数据库环境中，**LOAD** 命令通过在表所在的每个数据库分区上进行并行调用来利用分区内、分区间和 I/O 并行性。

在创建索引期间，可并行执行数据的扫描和后续排序。在创建索引时，DB2 系统既利用 I/O 并行性又利用分区内并行性。这有助于在发出 CREATE INDEX 语句时、重新启动期间（若一个索引标记为无效）及数据重组期间加快索引创建的速度。

备份和复原数据任务是繁重地涉及 I/O 的任务。在执行备份和复原操作时，DB2 系统既利用 I/O 并行性又利用分区内并行性。备份操作通过以并行方式读取多个表空间容器并以并行方式异步写入多备份介质，来利用 I/O 并行性。

分区数据库环境

分区数据库环境是支持将数据分布到各数据库分区上的数据库安装。

- *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。分区数据库环境是支持将数据分布到各数据库分区上的数据库安装。
- *单一分区数据库*是只有一个数据库分区的数据库。该数据库中的所有数据都存储在这个单一数据库分区中。在此情况下，数据库分区组即使存在也不会提供任何其他功能。
- *多分区数据库*是有两个或更多个数据库分区的数据库。表可以存储在一个或多个数据库分区中。当表存储在包含多个数据库分区的数据库分区组中时，它的某些行存储在一个数据库分区中，而其他行存储在其他数据库分区中。

通常，在每台物理机器上都存在单个数据库分区，而在每个数据库分区中，数据库管理器使用每个系统上的处理器来管理该数据库的全部数据中属于该分区的那一部分。

由于数据分布在各数据库分区中，所以可以使用多台物理机器上多个处理器的能力来满足对信息的请求。数据检索和更新请求被自动分解成子请求，并在适用的数据库分区中并行执行。数据库分布在各数据库分区中的这个事实对于发出 SQL 语句的用户是透明的。

用户交互通过一个数据库分区发生，该数据库分区称为该用户的*协调程序分区*。协调程序分区与应用程序在同一个数据库分区中运行，对于远程应用程序来说，协调程序分区在该应用程序所连接至的数据库分区中运行。任何数据库分区都可用作协调程序分区。

数据库管理器允许将数据存储在数据库的多个数据库分区中。这意味着该数据以物理方式存储在多个数据库分区中，但是对其进行访问时，仍可以将数据视为位于同一位置。访问多分区数据库中的数据的应用程序和用户不知道该数据的物理位置。

虽然该数据在物理上是分离的，但却将它作为一个逻辑整体来使用和管理。用户可以通过声明分布键来选择如何分发数据。通过选择表空间和要存储数据的关联数据库分区组，用户还可以确定数据可以分布在哪些数据库分区上以及分布在多少个数据库分区上。可以使用 DB2 设计顾问程序来完成有关分布和复制的建议。另外，将可更新的分发映射与散列算法配合使用，来指定分布键值到数据库分区的映射，以确定每行数据的位置和检索。因此，可以将大型表的工作负载分布在多分区数据库上，并将较小的表存储在一个或多个数据库分区中。每个数据库分区都有它所存储数据的本地索引，这可以提高本地数据的访问性能。

注: 并不是必须将所有表都分布到数据库的所有数据库分区中。数据库管理器支持*部分分区*，这意味着可以将表及其表空间分布到系统中的数据库分区的子集上。

当您想要将表放在每个数据库分区中时，可考虑的替代方法是使用具体化查询表，然后复制那些表。您可以创建包含所需信息的具体化查询表，然后将它复制到每个数据库分区。

非 root 用户安装的 DB2 数据库产品不支持数据库分区。请勿手动更新 `db2nodes.cfg` 文件。手动更新会返回错误 (SQL6031N)。

数据库分区和处理器环境

本节概述了单一数据库分区配置和多数据库分区配置。前者包括单一处理器（单处理器）配置和多处理器 (SMP) 配置，后者包括具有一个处理器 (MPP) 或多个处理器（SMP 集群）的数据库分区以及逻辑数据库分区。

*容量*是指能访问数据库的用户数和应用程序数。这很大程度上取决于内存、代理程序数、锁定数、I/O 和存储器管理。*可伸缩性*是指一个数据库随着其增长而继续显示出相同的操作特征和响应时间的能力。下面讨论每种环境的容量和可伸缩性。

单处理器上的单个数据库分区

此环境由内存和磁盘组成，但仅包含一个 CPU（请参阅图 31）。此环境中的数据库可满足一个部门或小办公室的需要，其中的数据和系统资源（包括一个单处理器或 CPU）由单数据库管理器来管理。

单处理器环境

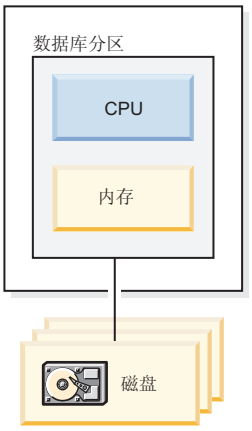


图 31. 单处理器上的单个数据库分区

容量和可伸缩性

在此环境中可以添加更多的磁盘。如果让每个磁盘拥有一个或多个 I/O 服务器，那么多个 I/O 操作可同时执行。

单处理器系统受处理器可处理的磁盘空间容量的限制。无论可以添加的其他组件（如内存或磁盘）如何，随着工作负载的增加，单个 CPU 可能无法更快地处理用户请求。如果已达到最大容量或最大可伸缩性，那么可考虑移到一个有多处理器的单个数据库分区系统中。

具有多处理器的单个数据库分区

此环境通常由同一台机器中几个能力相等的处理器组成（请参阅图 32），称为*对称多处理器 (SMP)* 系统。诸如磁盘空间和内存之类的资源是共享资源。

利用可用的多个处理器，可以更快地完成不同的数据库操作。DB2 数据库系统还可将单个查询的工作分布在可用的处理器中，以提高处理速度。其他数据库操作，例如，装入数据、备份和复原表空间以及对现有数据创建索引，都可以利用多个处理器。

对称多处理器（SMP）环境

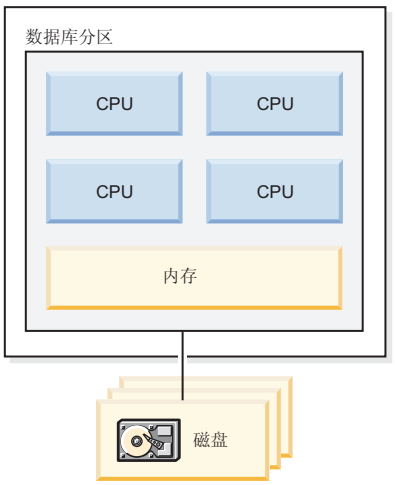


图 32. 单一分区数据库对称多处理器环境

容量和可伸缩性

可以通过增加磁盘数来增加与处理器关联的数据库分区的 I/O 容量。可以建立 I/O 服务器以专门处理 I/O 请求。如果让每个磁盘拥有一个或多个 I/O 服务器，那么多个 I/O 操作可同时执行。

如果已达到最大容量或最大可伸缩性，那么可考虑移到有多个数据库分区的系统中。

多个数据库分区配置

可以将一个数据库分布在多个数据库分区中，每个数据库分区在它自己的机器上。可将有多个数据库分区的多台机器编组在一起。本节描述下列数据库分区配置:

- 具有单处理器的系统上的数据库分区
- 具有多处理器的系统上的数据库分区
- 逻辑数据库分区

具有单处理器的数据库分区

在此环境中，有许多数据库分区。每个数据库分区都位于它自己的机器上，而且它有自己的处理器、内存和磁盘 (图 33)。所有机器通过通信工具连接在一起。可使用许多不同的名称来称呼此环境，包括：集群、单处理器集群、大规模并行处理 (MPP) 环境和“不共享”配置。最后一个名称准确地反映了此环境中的资源安排。与 SMP 环境不同，MPP 环境没有共享的内存或磁盘。MPP 环境避免了由共享内存和磁盘带来的限制。

一个分区数据库环境允许一个数据库保持在逻辑上的整体性，但它实际上分布于多个数据库分区中。数据是分布的这一事实对大多数用户是透明的。可以在数据库管理器之间划分工作；每个数据库分区中的每个数据库管理器都只为该数据库中它自己的那部分工作。

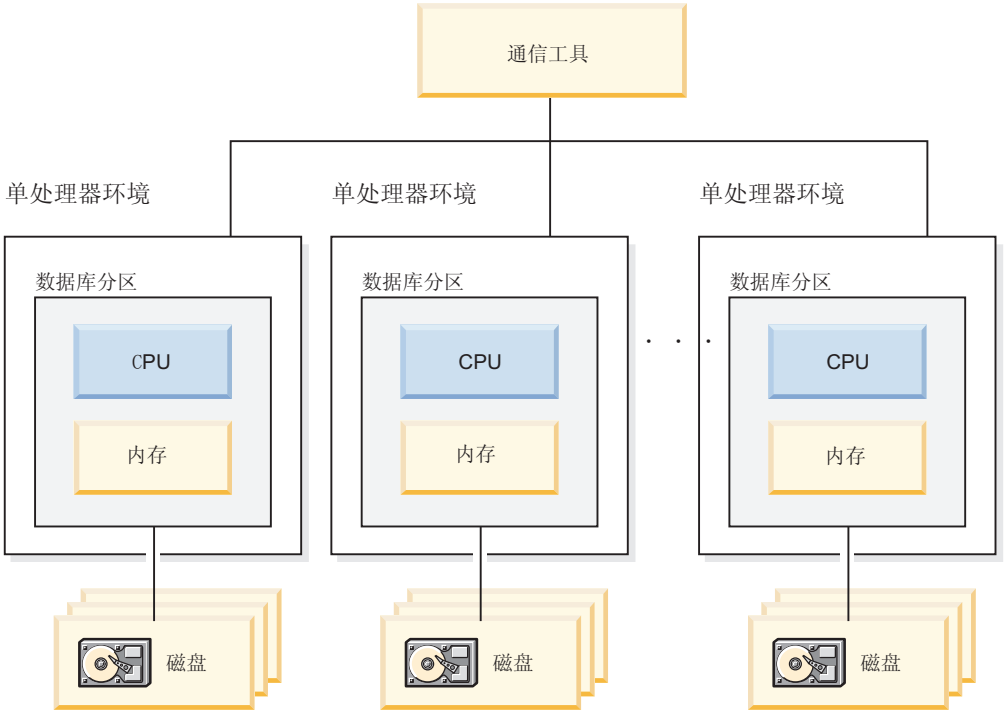


图 33. 大规模并行处理 (MPP) 环境

容量和可伸缩性

在此环境中可以向您的配置添加更多的数据库分区。在某些平台上，最大数目是 512 个数据库分区。但是，在管理很多机器和实例时，可能存在实际的限制。

如果已达到最大容量或最大可伸缩性，那么可考虑移到每个数据库分区都有多个处理器的系统中。

具有多处理器的数据库分区

每个数据库分区具有单处理器的替代配置是，每个数据库分区具有多处理器的配置。这称为 *SMP 集群* (第 70 页的图 34)。

此配置结合了 SMP 和 MPP 并行性的优点。这表示一个查询可以在跨多处理器的单个数据库分区中执行。亦即一个查询可以用并行方式在多个数据库分区中执行。

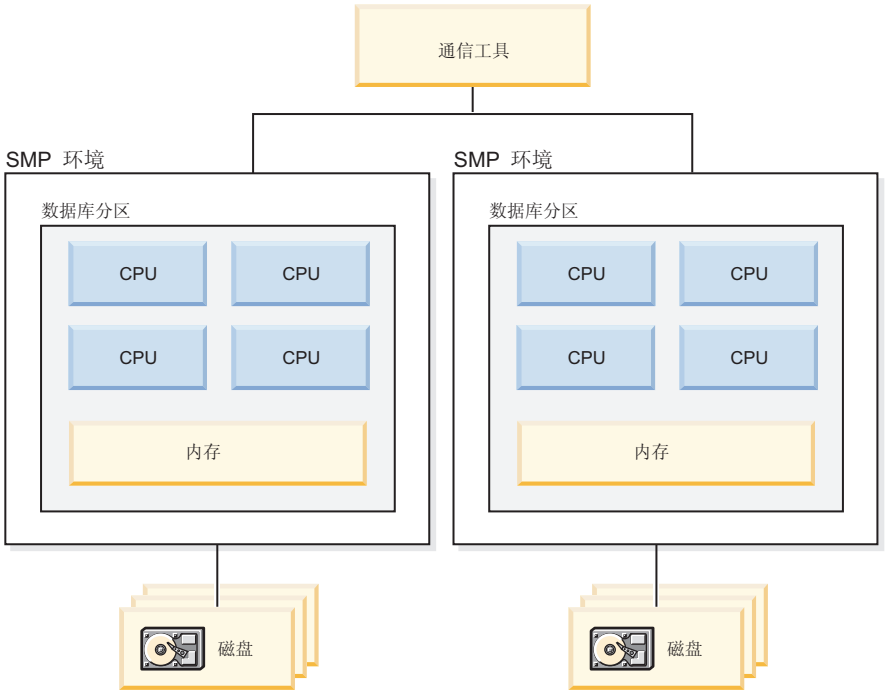


图 34. 集群中的几个对称多处理器 (SMP) 环境

容量和可伸缩性

在此环境中，可以添加更多的数据库分区，并可以向现有数据库分区添加更多的处理器。

逻辑数据库分区

逻辑数据库分区与物理分区的不同之处在于逻辑分区未被授予对整台机器的控制权。虽然机器已共享资源，但是数据库分区不共享资源。处理器是共享的，但磁盘和内存却不共享。

逻辑数据库分区提供了可伸缩性。在多逻辑分区上运行的多个数据库管理器可以比单个数据库管理器更能充分利用可用的资源。第 71 页的图 35说明了通过添加更多的数据库分区可以在一台 SMP 机器上获得更大的可伸缩性；对于那些具有许多处理器的机器而言更是如此。通过分配数据库，可以分别对每个数据库分区进行管理和恢复。

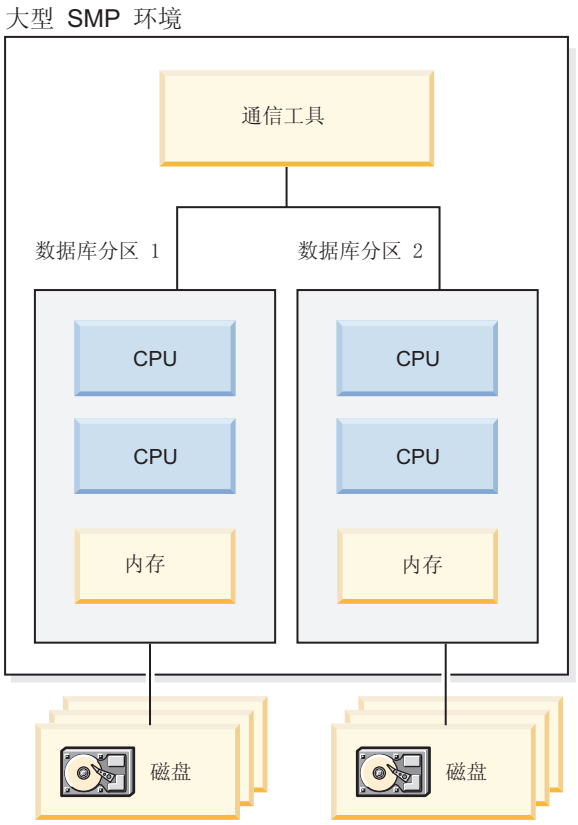


图 35. 带有对称多处理器环境的分区数据库

第 72 页的图 36 举例说明可以扩大图 35 中显示的配置以增强处理能力。

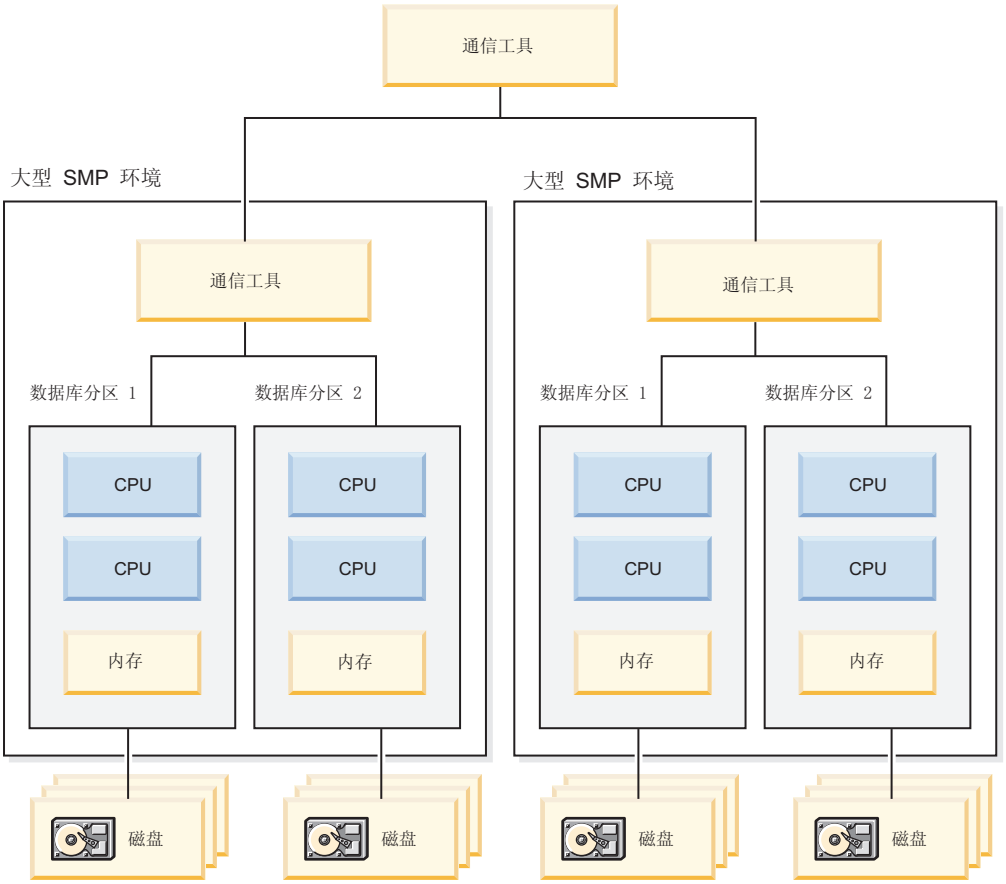


图 36. 带有集中在一起的对称多处理器环境的分区数据库

注：两个以上的数据库分区同时存在于同一台机器上的能力（不考虑处理器的数量），使得可以更灵活地设计高可用性配置和故障转移策略。机器发生故障之后，可以将一个数据库分区自动移至已包含同一数据库的另一个数据库分区的另一台机器上，然后重新启动该分区。

最适合每个硬件环境的并行性摘要

下表概述了最适合于利用各种硬件环境的并行性的类型。

表 8. 每种硬件环境中的可能并行性类型

硬件环境	I/O 并行性	查询内并行性	
		分区内并行性	分区间并行性
单个数据库分区，单处理器	是	否 [1]	否
单个数据库分区，多处理器 (SMP)	是	是	否
多个数据库分区，单处理器 (MPP)	是	否 [1]	是
多个数据库分区，多处理器（SMP 集群）	是	是	是
逻辑数据库分区	是	是	是
[1] 甚至是在单处理器系统中，使用一个配置参数将并行度设置为大于 1 的值也可能获得好处，特别是在查询并未充分使用 CPU 时（例如，如果它们受 I/O 约束）。			

第 2 部分 安装注意事项

第 5 章 安装先决条件

使用“DB2 安装”向导来安装 DB2 数据库服务器 (Windows)

此任务描述如何在 Windows 上启动“DB2 安装”向导。使用“DB2 安装”向导来定义安装以及在系统上安装 DB2 数据库产品。

开始之前

在启动“DB2 安装”向导之前:

- 如果您打算设置分区数据库环境，请参阅『设置分区数据库环境』。
- 确保系统满足安装、内存和磁盘要求。
- 如果您计划使用 LDAP 在 Windows 操作系统 Active Directory 中注册 DB2 服务器，请在安装之前扩展目录模式，否则必须手动注册节点并编目数据库。有关更多信息，请参阅“为 LDAP 目录服务扩展 Active Directory 模式 (Windows)”主题。
- 必须有具备建议的用户权限的本地管理员用户帐户才能执行安装。在 LocalSystem 可以用作 DAS 和 DB2 实例用户且您未在使用分区数据库环境的 DB2 数据库服务器中，具有提升特权的非管理员用户可以执行该安装。

  注: 如果打算使用非管理员用户帐户进行产品安装，那么在尝试安装 DB2 数据库产品之前，必须安装 VS2010 运行时库。操作系统上需要有 VS2010 运行时库，然后才能安装 DB2 数据库产品。可从 Microsoft 运行时库下载 Web 站点获得 VS2010 运行时库。有两个选项: 对于 32 位系统，选择 `vcredist_x86.exe`; 对于 64 位系统，选择 `vcredist_x64.exe`。
- 建议您关闭所有的程序（虽然不是强制），这样安装程序可以更新计算机上的任意文件而不需要重新引导。
- 从虚拟驱动器或未映射的网络驱动器（例如 Windows 资源管理器中 *hostname**sharename*）安装 DB2 产品不受支持。在尝试安装 DB2 产品之前，必须将网络驱动器映射至 Windows 盘符（例如 Z:）。

限制

- 通过任何用户帐户都无法运行多个“DB2 安装”向导实例。
- DB2 副本名称和实例名不能以数字值开始。 DB2 副本名称最长可为 64 个英语字符，这些字符可以是 A-Z、a-z 和 0-9。
- DB2 副本名称和实例名在所有的 DB2 副本中必须唯一。
- 只能对仅具有一个数据库分区的数据库使用 XML 功能。
- 如果已安装下列其中一项，那么其他 DB2 数据库产品均不能安装在同一路径中:
  - IBM® 数据服务器运行时客户机
  - IBM Data Server Driver Package
  - *DB2 信息中心*
- “DB2 安装”向导的字段不接受非英文字符。
- 如果在 Windows 或更高版本上启用扩展安全性，那么用户必须属于 DB2ADMNS 或 DB2USERS 组才能运行本地 DB2 命令和应用程序，这是因为有一个额外的安全性功

能（用户访问控制）在缺省情况下会限制本地管理员具有的特权。如果用户不属于其中任何一个组，那么他们将没有对本地 DB2 配置或应用程序数据的读访问权。

过程

在启动“DB2 安装”向导:

1. 使用已为 DB2 安装定义的本地本地管理员帐户来登录系统。
2. 如果您具有 DB2 数据库产品 DVD，那么将其插入驱动器中。如果启用了自动运行功能，那么它将自动启动“DB2 安装启动板”。如果自动运行功能不能工作，请使用 Windows 资源管理器来浏览 DB2 数据库产品 DVD，然后双击**安装**图标以启动“DB2 安装启动板”。
3. 如果您从 Passport Advantage® 下载了 DB2 数据库产品，那么运行该可执行文件来解压缩 DB2 数据库产品安装文件。使用 Windows 资源管理器来浏览 DB2 安装文件，然后双击**安装**图标来启动“DB2 安装启动板”。
4. 从“DB2 安装”启动板中，可以查看安装先决条件和发行说明，也可以直接进行安装。您可能要查看安装先决条件和发行说明以获取最新信息。
5. 单击**安装产品**，“安装产品”窗口将显示可供安装的产品。

   如果计算机上尚未安装任何 DB2 数据库产品，那么通过单击**安装新产品**来启动安装。遵循“DB2 安装”向导的提示逐步完成安装。

   如果计算机上至少已经安装了一个 DB2 数据库产品，那么可以:

   - 单击**安装新产品**以创建新的 DB2 副本。
   - 单击**使用现有产品**以更新现有 DB2 副本、将功能添加到现有 DB2 副本、升级现有 DB2 V9.7、V9.8 或 V10.1 副本或安装附加产品。
6. “DB2 安装向导”将确定系统语言，并启动该语言的安装程序。联机帮助可指导您完成其余步骤。要调用联机帮助，请单击**帮助**或按 **F1** 键。可随时单击**取消**来结束安装。
7. 使用“DB2 安装”向导时的样本面板将指导您完成安装过程。请参阅相关链接。

结果

缺省情况下，DB2 数据库产品将安装在 *Program_Files*\IBM\sqllib 目录中，其中 *Program_Files* 表示 Program Files 目录的位置。

如果您要在此目录已经在使用的系统上进行安装，那么 DB2 数据库产品安装路径会添加 *_xx*，其中 *xx* 是数字，从 01 开始并以您已安装的 DB2 副本数量递增。

您还可以指定自己的 DB2 数据库产品安装路径。

下一步做什么

- 验证安装。
- 执行必需的安装后任务。

有关安装期间遇到的错误的信息，请查看位于 My Documents\DB2LOG\ 目录中的安装日志文件。该日志文件使用以下格式：DB2-*ProductAbrrev-DateTime*.log，例如，DB2-ESE-Tue Apr 04 17_04_45 2012.log。

如果这是在 64 位 Windows 上新安装的 DB2 产品，而您使用 32 位 OLE DB 提供程序，那么必须手动注册 IBMDADB2 DLL。要注册此 DLL，请运行以下命令:

```
c:\windows\SysWOW64\regsvr32 /s c:\Program_Files\IBM\SQLLIB\bin\ibmdadb2.dll
```

其中 *Program_Files* 表示 Program Files 目录的位置。

如果想要 DB2 数据库产品能够访问本地计算机或网络中的另一计算机上的 DB2 文档，那么必须安装 *DB2 信息中心*。*DB2 信息中心*包含 DB2 数据库系统和 DB2 相关产品的文档。如果没有在本地安装 *DB2 信息中心*，那么缺省情况下将从 Web 访问 DB2 信息。

可以通过运行“DB2 安装”向导来安装 IBM Data Studio。

DB2 Express® Edition和 DB2 工作组服务器版的内存限制

如果要安装 DB2 Express Edition，那么允许用于实例的最大内存为 4 GB。

如果要安装 DB2 工作组服务器版，那么允许用于实例的最大内存为 64 GB。

分配给实例的内存量由 **INSTANCE_MEMORY** 数据库管理器配置参数确定。

从 V9.7、V9.8 或 V10.1 进行升级时的重要说明:

- 自调整内存管理器不会将实例内存总量限制增大到超出许可证限制。

为分区 DB2 服务器准备环境 (Windows)

本主题描述了在准备 Windows 环境以便对 DB2 数据库产品进行分区安装时需要执行的步骤。

开始之前

如果要将新机器作为一个分区添加到分区数据库环境中，那么新机器必须具备以下条件:

- 具有与拥有实例的机器相同的操作系统版本。
- 具有与拥有实例的机器相同的 CPU 体系结构（x32 位或 x64 位）

如果新机器不能满足这些要求，那么添加分区可能失败。

过程

要准备 Windows 环境以进行安装:

1. 确保主计算机和参与的计算机属于同一 Windows 域。 使用**系统属性**对话框来检查计算机所属的域，可以通过“控制面板”来访问此对话框。
2. 确保主计算机和参与的计算机上的时间与日期设置是一致的。 要使时间和日期设置被认为是一致的，所有计算机之间的 GMT 时差一定不能超过一个小时。

   可以使用**日期/时间属性**对话框修改系统的日期和时间，可以通过“控制面板”访问此对话框。可以使用 **max_time_diff** 配置参数来更改此限制。缺省值是 max_time_diff = 60，这允许时差小于 60 分钟。
3. 确保对每个参与分区数据库环境的计算机对象都标记了“信任计算机作为委派”特权。 您可以验证“Active Directory 用户和计算机”控制台中每台计算机的帐户**属性**对话框的**常规**选项卡上的“信任计算机作为委派”复选框是否已选中。

4. 确保所有参与的计算机都可以使用 TCP/IP 相互通信:
   a. 在一台参与的计算机上，输入 **hostname** 命令，它将返回该计算机的主机名。
   b. 在另一台参与的计算机上，输入以下命令:

      ```
      ping hostname
      ```

      其中 *hostname* 表示主计算机的主机名。如果测试成功，您将接收到类似如下的输出:

      ```
      Pinging ServerA.ibm.com [9.21.27.230] with 32 bytes of data:

      Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
      Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
      Reply from 9.21.27.230: bytes=32 time<10ms TTL=128
      ```

      重复执行这些步骤，直到确定所有参与的计算机都能使用 TCP/IP 相互进行通信为止。每台计算机必须具有静态 IP 地址。

      如果正打算使用多个网络适配器，可以指定要使用哪个适配器在数据库分区服务器之间进行通信。安装完成后，使用 **db2nchg** 命令在 db2nodes.cfg 文件中指定“网络名”字段。
5. 安装期间，系统将要求您提供 DB2 管理服务器用户帐户。 这是将由 DB2 管理服务器 (DAS) 使用的本地或域用户帐户。DAS 是用于支持 GUI 工具和协助完成管理任务的管理服务。您现在可以定义一个用户，也可以让“DB2 安装”向导创建一个用户。如果要使用“DB2 安装”向导创建新的域用户，那么用来执行安装的帐户必须具有创建域用户的权限。
6. 在将安装实例拥有的分区的主计算机上，必须具有属于本地 *Administrators* 组的域用户帐户。 安装 DB2 数据库产品时，您将作为此用户登录。必须将同一用户帐户添加至参与的每台计算机上的本地 *Administrators* 组。此用户必须具有*以操作系统方式操作*用户权限。
7. 确保实例中的所有计算机都具有同一个本地驱动器盘符上的数据库目录。 您可以通过运行 **GET DATABASE CONFIGURATION** 命令并验证 **dftdbpath** DBM 配置参数的值来检查是否满足此条件。
8. 安装期间，会要求您提供与 DB2 实例相关联的域用户帐户。 每个 DB2 实例都指定了一个用户。当启动实例时，DB2 数据库系统将使用此用户名进行登录。您现在可以定义一个用户，也可以让“DB2 安装”向导创建一个新的域用户。

   当添加新节点到分区环境中时，DB2 副本名称必须在所有的计算机上相同。

   如果要使用“DB2 安装”向导创建新的域用户，那么用来执行安装的帐户必须具有创建域用户的权限。实例用户域帐户在所有参与的计算机上必须属于本地 *Administrators* 组，并且将被授予下列用户权限:
   - 以操作系统方式操作
   - 创建标记对象
   - 锁定内存中的页
   - 作为服务登录
   - 增加限额
   - 替换进程级别标记

如果选择了扩展安全性，那么该帐户还必须是 DB2ADMNS 组的成员。DB2ADMNS 组已经具有这些特权，因此，已经对该帐户显式地添加了这些特权。

快速通信管理器（Windows）

在多成员环境中，每个成员具有一对 FCM 守护程序，它们用于支持与代理程序请求有关的成员之间的通信。一个守护程序用于发送通信，另一个用于接收通信。启动实例时会激活这些守护程序和支持基础结构。FCM 通信还用于在同一成员内部工作的代理程序；此类型通信也称为成员内部通信。

可以使用 **fcm_num_buffers** 数据库管理器配置参数来指定 FCM 消息缓冲区数。可以使用 **fcm_num_channels** 数据库管理器配置参数来指定 FCM 通道数。缺省情况下，**fcm_num_buffers** 和 **fcm_num_channels** 数据库管理器配置参数已设置为 AUTOMATIC。如果设置为 AUTOMATIC（这是建议的设置），那么 FCM 会监视资源使用情况并调整资源以满足工作负载需求。

DB2 数据库服务器安装概述（Linux 和 UNIX）

此主题概述了在 AIX、HP-UX、Linux 和 Solaris 上安装 DB2 服务器产品的步骤。

过程

要安装 DB2 服务器产品:

1. 查看 DB2 产品先决条件。
2. 如果提供了 DB2 升级信息，请查看此信息。
3. 在 HP-UX、Linux 和 Solaris 上修改内核参数。在除 Linux on x86_32 之外的所有平台上，您必须安装 64 位内核，然后才可以继续进行安装，否则安装将失败。
4. 准备安装介质:

   产品 DVD

   如果 DB2 产品 DVD 未自动安装，那么安装 DB2 产品 DVD。

   安装映像

   如果已下载安装映像，那么将该文件解压缩。
5. 使用其中一种可用方法来安装 DB2 产品:
   - “DB2 安装”向导
   - 使用响应文件进行静默安装
   - 有效内容文件部署

   对于 DB2 服务器，您可以使用“DB2 安装”向导来执行安装和配置任务，例如:
   - 选择 DB2 安装类型（典型、精简或定制）。
   - 选择 DB2 产品安装位置。
   - 安装您稍后可以指定的语言作为产品界面和消息的缺省语言。
   - 安装或升级 IBM Tivoli® System Automation for Multiplatforms（Linux 和 AIX）。
   - 设置 DB2 实例。
   - 设置 DB2 管理服务器（包括 DAS 用户设置）。
   - 设置 DB2 Text Search 服务器。
   - 设置管理联系人和运行状况监视器通知。

- 设置和配置实例（包括实例用户设置）。
- 设置 Informix 数据源支持。
- 准备 DB2 工具目录。
- 指定 DB2 信息中心端口。
- 创建响应文件。

6. 如果使用除“DB2 安装”向导之外的其他方法安装 DB2 服务器，那么需要执行安装后配置步骤。

DB2 安装方法

安装 DB2 数据库产品的方法有多种。每种安装方法都适合于特定的情况。

下表显示操作系统可用的安装方法。

表 9. 操作系统可用的安装方法

安装方法	Windows	Linux 或 UNIX
“DB2 安装”向导	是	是
响应文件安装	是	是
db2_install 命令	否	是
有效内容文件部署	否	是

要点：不推荐使用 **db2_install** 命令，将来的发行版中可能会除去此命令。请改为使用 **db2setup** 命令或响应文件安装方法。

以下列表描述了 DB2 安装方法。

“DB2 安装”向导

“DB2 安装”向导是可在 Linux、UNIX 和 Windows 操作系统上使用的一个 GUI 安装程序。“DB2 安装”向导提供了易于使用的界面，用于安装 DB2 数据库产品以及执行初始设置和配置任务。

“DB2 安装”向导还可以创建 DB2 实例和响应文件，它们可用于在其他机器上重复此安装过程。

注：对于 Linux 和 UNIX 操作系统上的非 root 用户安装，只能存在一个 DB2 实例。“DB2 安装”向导将自动创建非 root 用户实例。

在 Linux 和 UNIX 操作系统上，需要 X 服务器才能显示“DB2 安装”向导。

响应文件安装

响应文件是一个包含设置和配置值的文本文件。“DB2 安装”程序将读取该文件，并根据已指定的值来执行安装。

响应文件安装也称为静默安装。

响应文件的另一个优点是：它们提供了对那些不能使用“DB2 安装”向导设置的参数的访问。

在 Linux 和 UNIX 操作系统上，如果将 DB2 安装映像嵌入您自己的应用程序中，那么您的应用程序有可能从安装程序中以计算机可读的格式接收安装进度信息和提示。此行为由 **INTERACTIVE** 响应文件关键字控制。

可以采用下列方法来创建响应文件:

使用响应文件生成器

可以使用响应文件生成器来创建一个用于复制现有安装的响应文件。例如，可以安装 IBM 数据服务器客户机、完整配置该客户机，然后生成响应文件以将该客户机的安装和配置复制到其他计算机。

使用“DB2 安装”向导

“DB2 安装”向导可以根据您在完成“DB2 安装”向导过程中所作的选择来创建响应文件。您的选择会记录在一个响应文件中，可以将该响应文件保存至系统上的某个位置。如果选择分区数据库安装，那么会生成两个响应文件，一个用于拥有实例的计算机，另一个用于参与的计算机。

此安装方法的一个好处是：无需执行安装，即可创建响应文件。此功能在捕获安装 DB2 数据库产品所需的选项时非常有用。稍后可以使用响应文件以完全按照您指定的选项来安装 DB2 数据库产品。

可以使用 **db2cfexp** 命令来导出客户机或服务器概要文件，以便保存客户机或服务器配置。使用 **db2cfimp** 命令来导入概要文件。还可以通过使用 **CLIENT_IMPORT_PROFILE** 关键字在响应文件安装期间导入使用 **db2cfexp** 命令导出的客户机或服务器概要文件。

在执行安装和编目数据源之后，您应该导出客户机或服务器概要文件。

定制为每个 DB2 数据库产品提供的样本响应文件

除了使用响应文件生成器或“DB2 安装”向导来创建响应文件以外，还可以手动修改样本响应文件。DB2 数据库产品 DVD 上提供了样本响应文件。样本响应文件提供了关于每个产品的所有有效关键字的详细信息。

db2_install 命令（仅适用于 Linux 和 UNIX 操作系统）

db2_install 命令将通过英语界面支持来安装您指定的 DB2 数据库产品的*所有*组件。通过使用 **-L** 参数就可以选择要支持的其他语言。您不能选择或清除组件。

尽管 **db2_install** 命令会安装您指定的 DB2 数据库产品的所有组件，但它不会创建用户和组，不会创建实例，也不会执行配置。在安装之后执行配置时，此安装方法可能是首选。要在安装 DB2 数据库产品时配置此产品，请考虑使用“DB2 安装”向导。

在 Linux 和 UNIX 操作系统上，如果将 DB2 安装映像嵌入您自己的应用程序中，那么您的应用程序有可能从安装程序中以计算机可读的格式接收安装进度信息和提示。

此安装方法要求在部署产品文件之后进行手动配置。

切记： 不推荐使用 **db2_install** 命令，将来的发行版中可能会除去此命令。

有效内容文件部署（仅适用于 Linux 和 UNIX）

此方法是一种高级安装方法，对于大多数用户，不推荐使用此方法。它要求用户以物理方式安装有效内容文件。有效内容文件是一个压缩的 tarball，它包含可安装的组件的所有文件和元数据。

DB2 pureScale 安装不支持此方法。
此安装方法要求在部署产品文件之后进行手动配置。

注: DB2 数据库产品安装不再是 Linux 和 UNIX 上的操作系统程序包。因此，可以不再使用操作系统命令来进行安装。必须更改您用来与 DB2 数据库产品安装进行交互以及查询 DB2 数据库产品安装的任何现有脚本。

使用“DB2 安装”向导来安装 DB2 服务器（Linux 和 UNIX）

此任务描述如何在 Linux 和 UNIX 操作系统上启动“DB2 安装”向导。“DB2 安装”向导用来定义安装首选项以及在系统上安装 DB2 数据库产品。

开始之前

在启动“DB2 安装”向导之前:

- 如果您打算设置分区数据库环境，请参阅《安装 DB2 服务器》中的『设置分区数据库环境』
- 确保系统满足安装、内存和磁盘要求。
- 确保安装了受支持的浏览器。
- 可以使用 root 用户权限或者非 root 用户权限来安装 DB2 数据库服务器。有关非 root 用户安装的更多信息，请参阅*安装 DB2 服务器*中的『非 root 用户安装概述（Linux 和 UNIX）』。
- DB2 数据库产品映像必须可用。可通过购买实体 DB2 数据库产品 DVD 或者从 Passport Advantage 下载安装映像来获取 DB2 安装映像。
- 如果要安装非英文版本的 DB2 数据库产品，那么必须具有适当的本地语言程序包。
- “DB2 安装”向导是一个图形安装程序。必须具有能够提供图形用户界面的 X Windows 软件，才能使“DB2 安装”向导在机器上运行。确保 X windows 服务器正在运行。确保正确导出了显示内容。例如，`export DISPLAY=9.26.163.144:0.`
- 如果要在您所在环境中使用安全性软件，那么在启动“DB2 安装”向导之前必须手动创建必需的 DB2 用户。

限制

- 通过任何用户帐户都无法运行多个“DB2 安装”向导实例。
- 只能对使用代码集 UTF-8 定义的并且只有一个数据库分区的数据库使用 XML 功能。
- “DB2 安装”向导的字段不接受非英文字符。
- 对基于 Itanium 的 HP Integrity Series Systems 上的 HP-UX 11i V2，使用“DB2 安装”向导中指定的密码无法访问通过安装向导为 DB2 实例所有者、受防护的用户或 DAS 创建的用户。完成此安装向导之后，需要重新设置这些用户的密码。这不会影响使用此安装向导创建的实例或 DAS，因此，不需要重新创建该实例或 DAS。

过程

在启动“DB2 安装”向导:

1. 如果具有实际的 DB2 数据库产品 DVD，那么通过输入下列命令来切换至安装了此 DB2 数据库产品 DVD 的目录:

   ```
   cd /dvdrom
   ```

   其中 */dvdrom* 表示 DB2 数据库产品 DVD 的安装点。
2. 如果下载了 DB2 数据库产品映像，那么必须对产品文件进行解压缩。
   a. 解压缩产品文件:

```
gzip -d product.tar.gz
```

其中 *product* 是下载的产品的名称。

b. 解压产品文件:

在 Linux 操作系统上

```
tar -xvf product.tar
```

在 AIX、HP-UX 和 Solaris 操作系统上

```
gnutar -xvf product.tar
```

其中 *product* 是下载的产品的名称。

c. 更改目录:

```
cd ./product
```

其中 *product* 是下载的产品的名称。

注: 如果下载了本地语言程序包，那么将其解压缩至同一个目录中。这将会在同一目录中创建子目录（例如，`./nlpack`），并且允许安装程序自动查找安装映像而无需提示。

3. 通过从数据库产品映像所在目录中输入 **./db2setup** 命令来启动“DB2 安装”向导。
4. 将打开“IBM DB2 安装启动板”。在此窗口中，可以查看安装先决条件和发行说明，也可以直接进行安装。您还可以查看安装先决条件和发行说明以获取最新信息。
5. 单击**安装产品**，**安装产品**窗口将显示可供安装的产品。

   通过单击“**安装新产品**”启动安装。遵循“DB2 安装”向导的提示逐步完成安装。
6. 使用“DB2 安装”向导时的样本面板将指导您完成安装过程。请参阅相关链接。

   在启动安装后，请完成“DB2 安装”向导的安装面板并作出选择。安装帮助可用来指导您完成其余步骤。要调用安装帮助，请单击**帮助**或按 F1 键。可随时单击**取消**来结束安装。

结果

对于非 root 用户安装，DB2 数据库产品始终安装在 *$HOME*/sqllib 目录中，其中 *$HOME* 表示非 root 用户的主目录。

对于 root 用户安装，DB2 数据库产品在缺省情况下安装在下列其中一个目录中:

AIX、HP-UX 和 Solaris
/opt/IBM/db2/V10.5

Linux /opt/ibm/db2/V10.5

如果您要安装在一个已经在使用该目录的系统上，那么 DB2 数据库产品安装路径会添加 *_xx*，其中 *_xx* 是数字，从 01 开始并按照您已经安装的 DB2 副本数量递增。

您还可以指定自己的 DB2 数据库产品安装路径。

DB2 安装路径具有下列规则:

- 可以包含小写字母 (a-z)、大写字母 (A-Z) 和下划线字符 (_)
- 不能超过 128 个字符

- 不能包含空格
- 不能包含非英文字符

安装日志文件是:

- DB2 安装日志文件。此文件将捕获包括错误在内的所有 DB2 安装信息。
  - 对于 root 用户安装，DB2 安装日志文件名是 db2setup.log。
  - 对于非 root 用户安装，DB2 安装日志文件名是 db2setup_*username*.log，其中 *username* 是用于执行安装的非 root 用户标识。
- DB2 错误日志文件。此文件会捕获由 Java™ 返回的任何错误输出（例如，异常和陷阱信息）。
  - 对于 root 用户安装，DB2 错误日志文件名是 db2setup.err。
  - 对于非 root 用户安装，DB2 错误日志文件名是 db2setup_*username*.err，其中 *username* 是用于执行安装的非 root 用户标识。

缺省情况下，这些日志文件在 /tmp 目录中。可以指定日志文件的位置。

不再存在 db2setup.his 文件。但是，DB2 安装程序会将 DB2 安装日志文件的副本保存在 DB2_DIR/install/logs/ 目录中，并将该文件重命名为 db2install.history。如果该名称已存在，那么 DB2 安装程序会将其重命名为 db2install.history.*xxxx*，其中 *xxxx* 为 0000-9999，这取决于该机器上的安装版本数目。

每个安装副本都有一个单独的历史记录文件列表。如果除去了一个安装副本，那么此安装路径下的历史记录文件也将被除去。此复制操作是在安装快要结束时执行的，如果程序在完成之前就已停止或者异常中止，那么不会创建历史记录文件。

下一步做什么

- 验证安装。
- 执行必需的安装后任务。

可以通过运行“DB2 安装”向导来安装 IBM Data Studio。

在安装了 DB2 数据库产品之后，通过从本地语言包所在的目录运行 **./db2setup** 命令，也可以安装本地语言包。

在 Linux x86 上，如果想要 DB2 数据库产品能够访问本地计算机或网络上的另一计算机上的 DB2 文档，那么必须安装 *DB2 信息中心*。*DB2 信息中心*包含 DB2 数据库系统和 DB2 相关产品的文档。

DB2 Express Edition和 DB2 工作组服务器版的内存限制

如果要安装 DB2 Express Edition，那么允许用于实例的最大内存为 4 GB。

如果要安装 DB2 工作组服务器版，那么允许用于实例的最大内存为 64 GB。

分配给实例的内存量由 **INSTANCE_MEMORY** 数据库管理器配置参数确定。

从 V9.7、V9.8 或 V10.1 进行升级时的重要说明:

- 如果从 V9.7、V9.8 或 V10.1 DB2 数据库产品进行升级时的重要说明的内存配置超过允许的限制，那么在升级到当前版本之后，DB2 数据库产品可能无法启动。
- 自调整内存管理器不会将总的实例内存上限增大到超出许可证限制。

快速通信管理器（Linux 和 UNIX）

快速通信管理器 (FCM) 为分区数据库环境提供通信支持。

在多成员环境中，每个成员具有一对 FCM 守护程序，它们用于支持与代理程序请求有关的成员之间的通信。一个守护程序用于发送通信，另一个用于接收通信。启动实例时会激活这些守护程序和支持基础结构。FCM 通信还用于在同一成员内部工作的代理程序；此类型通信也称为成员内部通信。

FCM 守护程序将收集有关通信活动的信息。可以通过使用数据库系统监视器来获得有关 FCM 通信的信息。如果成员之间的通信失败，或者它们重新建立通信，那么 FCM 守护程序会使用此信息来更新监视器元素。FCM 守护程序还会为此事件触发适当的操作。例如，回滚受到影响的事务。可以使用数据库系统监视器来帮助您设置 FCM 配置参数。

可以使用 **fcm_num_buffers** 数据库管理器配置参数来指定 FCM 消息缓冲区数。可以使用 **fcm_num_channels** 数据库管理器配置参数来指定 FCM 通道数。缺省情况下，**fcm_num_buffers** 和 **fcm_num_channels** 数据库管理器配置参数已设置为 AUTOMATIC。如果设置为 AUTOMATIC（这是建议的设置），那么 FCM 会监视资源使用情况并调整资源以满足工作负载需求。

第 6 章 安装之前

其他分区数据库环境预安装任务（Linux 和 UNIX）

更新用于分区 DB2 安装的环境设置 (AIX)

此任务描述了将参与分区数据库系统的每台计算机上需要更新的环境设置。

过程

要更新 AIX 环境设置:

1. 作为具有 root 用户权限的用户登录计算机。
2. 通过输入以下命令，将 AIX maxuproc（每个用户的最大进程数）设备属性设置为 4096:

   ```
   chdev -l sys0 -a maxuproc='4096'
   ```

   注: 如果另一映像正在运行，那么可能需要将 bosboot/reboot 切换为 64 位内核。
3. 在参与分区数据库系统的所有工作站上，将 TCP/IP 网络参数设置为下列值。这些值都是这些参数的最小值。如果任何网络相关参数都已设置为较高的值，那么不要进行更改。

   ```
   thewall       = 65536
      sb_max        = 1310720
      rfc1323       = 1
      tcp_sendspace = 221184
      tcp_recvspace = 221184
      udp_sendspace = 65536
      udp_recvspace = 65536
      ipqmaxlen     = 250
      somaxconn     = 1024
   ```

   要列示所有网络相关参数的当前设置，输入以下命令:

   ```
   no -a | more
   ```

   要设置参数，输入以下命令:

   ```
   no -o parameter_name=value
   ```

   其中:

   - *parameter_name* 表示想要设置的参数。
   - *value* 表示想要对此参数设置的值。

   例如，要将 tcp_sendspace 参数设置为 221184，输入以下命令:

   ```
   no -o tcp_sendspace=221184
   ```
4. 如果正在使用高速互连，那么必须将 css0 的 spoolsize 和 rpoolsize 设置为下列值:

   ```
   spoolsize     16777216
   rpoolsize     16777216
   ```

   要列示这些参数的当前设置，输入以下命令:

```
lsattr -l css0 -E
```

要设置这些参数，请输入下列命令:

```
/usr/lpp/ssp/css/chgcss -l css0 -a spoolsize=16777216
  /usr/lpp/ssp/css/chgcss -l css0 -a rpoolsize=16777216
```

如果不使用 /tftpboot/tuning.cst 文件来调整系统，那么可以在安装之后使用 *DB2DIR*/misc/rc.local.sample 样本脚本文件（其中 *DB2DIR* 是 DB2 数据库产品的安装路径）来更新与网络相关的参数。要在安装之后使用样本脚本文件来更新网络相关参数，执行下列步骤:

a. 通过输入下列命令，将此脚本文件复制到 /etc 目录，并使它可由 root 用户执行:

```
cp /usr/opt/db2_09_01/misc/rc.local.sample /etc/rc.local
chown root:sys /etc/rc.local
chmod 744 /etc/rc.local
```

b. 查看 /etc/rc.local 文件并对其进行更新（如果有必要）。

c. 向 /etc/inittab 文件添加一个条目，以便每当机器重新引导时执行 /etc/rc.local 脚本。 可使用 **mkitab** 命令来向 /etc/inittab 文件添加一个条目。要添加此条目，输入以下命令:

```
mkitab "rclocal:2:wait:/etc/rc.local > /dev/console 2>&1"
```

d. 通过输入以下命令，确保 /etc/inittab 文件包括 /etc/rc.nfs 条目:

```
lsitab rcnfs
```

e. 通过输入以下命令，在不重新引导系统的情况下更新网络参数:

```
/etc/rc.local
```

5. 确保具有足够的调页空间来运行 DB2 Enterprise Server Edition 的分区安装。 如果没有足够的调页空间，那么操作系统将停止正在使用大部分虚拟内存（这可能是其中一个 DB2 进程）的进程。 要检查可用的调页空间，输入以下命令:

```
lsps -a
```

此命令将返回类似于以下的输出:

```
Page Space  Physical Volume  Volume Group  Size  %Used  Active  Auto  Type
paging00    hdisk1           rootvg        60MB     19     yes   yes    lv
hd6         hdisk0           rootvg        60MB     21     yes   yes    lv
hd6         hdisk2           rootvg        64MB     21     yes   yes    lv
```

可用调页空间应是在计算机上安装的物理内存量的两倍。

6. 如果正在创建小型到中型的分区数据库系统，那么拥有实例的计算机上的网络文件系统守护程序 (NFSD) 的数目应该接近于:

```
每台计算机上的 biod 数 × 实例中的计算机数
```

理想情况下，应该在每台计算机上运行 10 个 biod 进程。根据以上公式，在由四台计算机组成的系统上（每台计算机上有 10 个 biod 进程），应该使用 40 个 NFSD。

如果正在安装大型系统，那么计算机上最多可以安装 120 个 NFSD。

有关 NFS 的其他信息，请参阅 NFS 文档。

建立工作集合以将命令分发到多个 AIX 节点

在 AIX 上的分区数据库环境中，可以设置工作集合以将命令分发至参与分区数据库系统的一组 System p® SP 工作站。使用 **dsh** 命令可以将命令分发至工作站。

开始之前

在 AIX 上安装或管理分区数据库系统时这可能会很有用，它使您能够对环境中的所有计算机快速地执行相同命令并且减少了出错的可能性。

必须知道想要包括在工作集合中的每台计算机的主机名。

必须作为具有 root 用户权限的用户登录控制工作站。

您必须具有一个文件，该文件列示将参与分区数据库系统的所有工作站的主机名。

过程

要设置工作集合以将命令分发至一系列工作站:

1. 创建称为 nodelist.txt 的文件，该文件将列示将参与工作集合的所有工作站的主机名。

   例如，假定您想借助名为 workstation1 和 workstation2 的两个工作站来创建工作集合。 的内容nodelist.txt 可能是:

   ```
   workstation1
   workstation2
   ```

2. 更新工作集合环境变量。 要更新此列表，请输入以下命令:

   ```
   export DSH_NODE_LIST=path/nodelist.txt
   ```

   其中 *path* 是创建 nodelist.txt 的位置，而 nodelist.txt 是创建的文件的名称，该文件列示工作集合中的工作站。

3. 通过输入以下命令，验证工作集合中的名称是否确实是想要的工作站:

   ```
   dsh -q
   ```

   您将接收到与下列内容类似的输出:

   ```
   Working collective file /nodelist.txt:
      workstation1
      workstation2
      Fanout: 64
   ```

验证 NFS 是否正在运行（Linux 和 UNIX）

在设置数据库分区环境之前，应验证“网络文件系统”(NFS) 在将参与分区数据库系统的每台计算机上是否正在运行。

过程

要验证 NFS 是否在每台计算机上都正在运行:

- AIX 操作系统:

  在每台计算机上输入以下命令:

  ```
  lssrc -g nfs
  ```

NFS 进程的状态字段应该指示活动。确认了 NFS 正在每个系统上运行之后，应检查 DB2 数据库产品必需的特定 NFS 进程。必需的进程有:

```
rpc.lockd
rpc.statd
```

- HP-UX 和 Solaris 操作系统:

  在每台计算机上输入以下命令:

  ```
  showmount -e hostname
  ```

  输入不带 *hostname* 参数的 **showmount** 命令来检查本地系统。如果 NFS 不活动，那么您将接收到类似如下的一条消息:

  ```
  showmount: ServerA: RPC: Program not registered
  ```

  确认了 NFS 正在每个系统上运行之后，应检查 DB2 数据库产品必需的特定 NFS 进程:

  ```
  rpc.lockd
  rpc.statd
  ```

  可以使用下列命令来检查这些进程:

  ```
  ps -ef | grep rpc.lockd
  ps -ef | grep rpc.statd
  ```

- Linux 操作系统:

  在每台计算机上输入以下命令:

  ```
  showmount -e hostname
  ```

  输入不带 *hostname* 参数的 **showmount** 命令来检查本地系统。

  如果 NFS 不活动，那么您将接收到类似如下的一条消息:

  ```
  showmount: ServerA: RPC: Program not registered
  ```

  确认了 NFS 正在每个系统上运行之后，应检查 DB2 数据库产品必需的特定 NFS 进程。必需的进程是 rpc.statd。

  可以使用 **ps -ef | grep rpc.statd** 命令来检查此进程。

如果这些进程没有运行，那么参阅您的操作系统文档。

验证参与的计算机上的可用端口范围（Linux 和 UNIX）

此任务描述了验证参与的计算机上的可用端口范围时需要执行的步骤。端口范围由“快速通信管理器”(FCM) 使用。FCM 是 DB2 用来处理数据库分区服务器之间的通信的功能部件。

开始之前

应该在安装拥有实例的数据库分区服务器之后，并在安装任何参与的数据库分区服务器之前验证参与的计算机上的可用端口范围。

当在主计算机上安装拥有实例的数据库分区服务器时，DB2 根据指定的参与分区数据库环境的逻辑数据库分区服务器数目保留端口范围。缺省范围为四个端口。对于每个参与分区数据库环境的服务器，您必须为 FCM 端口手动配置 /etc/services 文件。FCM

端口范围取决于要在参与的计算机上使用的逻辑分区数目。至少需要两个条目: DB2_*instance* 和 DB2_*instance*_END。在参与的计算机上指定的 FCM 端口的其他要求:

- 起始端口号必须与主计算机的起始端口号相匹配
- 后续端口必须按顺序进行编号
- 指定的端口号必须是空闲的

要更改 services 文件，您需要具有 root 用户权限。

过程

要验证参与的计算机上的可用端口范围:

1. 打开位于 /etc/services 目录中的 services 文件。
2. 查找为 DB2 的“快速通信管理器”(FCM) 保留的端口。这些条目应该类似于以下示例:

```
DB2_db2inst1          60000/tcp
DB2_db2inst1_1        60001/tcp
DB2_db2inst1_2        60002/tcp
DB2_db2inst1_END      60003/tcp
```

   DB2 将保留 60000 之后的前四个可用端口。
3. 在参与的每台计算机上，打开 services 文件，并验证在主计算机的 services 文件中为 DB2 FCM 保留的端口是否未使用。
4. 如果参与的某台计算机正在使用需要的端口，那么应确定所有计算机的可用端口范围，并更新每个 services 文件（包括主计算机上的 services 文件）。

下一步做什么

在主计算机上安装拥有实例的数据库分区服务器之后，必须在参与的数据库分区服务器上安装 DB2 数据库产品。可以使用为分区服务器生成的响应文件（缺省名称为 db2ese_addpart.rsp），需要为 FCM 端口手动配置 /etc/services 文件。FCM 端口范围取决于要在当前机器上使用的逻辑分区数目。最小条目为 DB2_ 和 DB2__END 这两个具有连续可用端口号的条目。在参与的每台机器上使用的 FCM 端口号均必须具有相同的起始端口号，并且后续端口必须按顺序进行编号。

为分区数据库系统创建文件系统 (Linux)

作为在 Linux 操作系统上设置分区数据库系统的一部分，您需要创建 DB2 主文件系统。然后，必须通过 NFS 导出此主文件系统，并在参与分区数据库系统的每台计算机中安装此主文件系统。

关于此任务

您的文件系统必须可用于将参与分区数据库系统的所有机器。此文件系统将用作实例主目录。

对于对单个数据库实例使用多台机器的配置，使用 NFS（网络文件系统）来共享此文件系统。通常，集群中的一台机器用于使用 NFS 导出文件系统，而集群中的其余机器将从此机器安装 NFS 文件系统。对于导出文件系统的机器，将以本地方式安装该文件系统。

有关更多命令信息，请参阅 Linux 分发文档。

过程

要创建以 NFS 方式导出并以 NFS 方式安装的 DB2 主文件系统，请执行下列步骤:

1. 在一台机器上，选择一个磁盘分区或使用 **fdisk** 来创建一个磁盘分区。
2. 通过使用诸如 **mkfs** 之类的实用程序，在此分区上创建文件系统。 该文件系统应该足够大以包含必需的 DB2 程序文件以及数据库需要的足够空间。
3. 以本地方式安装您刚刚创建的文件系统，并向 /etc/fstab 文件添加一个条目，以便系统每次重新引导时都安装此文件系统。 例如:

   ```
   /dev/hda1    /db2home    ext3   defaults   1 2
   ```

4. 要在 Linux 上在引导时自动导出 NFS 文件系统，应向 /etc/exports 文件添加一个条目。 务必包括参与集群的所有主机名以及机器可能具有的所有名称。并且，还应该通过使用“root”选项来确保集群中每台机器对导出的文件系统都具有 root 用户权限。

   /etc/exports 文件是包含以下类型的信息的 ASCII 文件:

   ```
   /db2home machine1_name(rw) machine2_name(rw)
   ```

   要导出 NFS 目录，运行

   ```
   /usr/sbin/exportfs -r
   ```

5. 在集群中的其余每台机器上，向 /etc/fstab 文件添加一个条目，以便在引导时以 NFS 方式自动安装该文件系统。 如下例所示，当指定安装点选项时，确保该文件系统是可读写的并是在引导时硬安装的，它包括了 bg（后台）选项，且 **setuid** 程序可以正常运行。

   ```
   fusion-en:/db2home /db2home nfs rw,timeo=7,
         hard,intr,bg,suid,lock
   ```

   其中，*fusion-en* 表示机器名。

6. 以 NFS 方式在集群中的其余每台机器上安装所导出的文件系统。 输入以下命令:

   ```
   mount /db2home
   ```

   如果 **mount** 命令失败，那么使用 **showmount** 命令来检查 NFS 服务器的状态。例如:

   ```
   showmount -e fusion-en
   ```

   其中，*fusion-en* 表示机器名。

   此 **showmount** 命令应列示从机器 fusion-en 导出的文件系统。如果此命令失败，那么 NFS 服务器可能尚未启动。要手动启动 NFS 服务器，在该服务器上作为 root 用户运行以下命令:

   ```
   /etc/rc.d/init.d/nfs restart
   ```

   假定目前的运行级别为 3，那么可以通过将目录 /etc/rc.d/rc3.d 下的 K20nfs 重命名为 S20nfs，以便在引导时自动运行此命令。

结果

通过执行这些步骤，您就完成了下列任务:

1. 在分区数据库环境中的单一计算机上，已经创建要用作实例和主目录的文件系统。
2. 如果配置对单个数据库实例使用多台机器，那么通过使用 NFS 导出此文件系统。

3. 已在参与的每台计算机上安装了导出的文件系统。

为分区数据库系统创建 DB2 主文件系统 (AIX)

作为设置分区数据库系统的一部分，您需要创建 DB2 主文件系统。然后，必须通过 NFS 导出此主文件系统，并在参与分区数据库系统的每台计算机中安装此主文件系统。

开始之前

建议您创建一个能够容纳 DB2 数据库产品 DVD 上的内容的主文件系统。可以使用以下命令来检查大小 (KB):

```
du -sk DVD_mounting_point
```

DB2 实例至少需要 200 MB 空间。如果没有足够的可用空间，那么可以从参与的每台计算机安装 DB2 数据库产品 DVD，这是将内容复制到磁盘的另一种方法。

您必须具有:

- 创建文件系统的 root 用户权限
- 创建了创建了要用于实际放置文件系统的卷组。

过程

要创建以 NFS 方式导出并以 NFS 方式安装的 DB2 主文件系统，请执行下列步骤:

1. 创建 DB2 主文件系统。

   作为具有 root 用户权限的用户登录分区数据库系统的主计算机 (ServerA)，并为分区数据库系统创建名为 /db2home 的主文件系统。

   a. 输入 **smit jfs** 命令。
   b. 单击**添加日志文件系统**图标。
   c. 单击**添加标准日志文件系统**图标。
   d. 从**卷组名**列表中选择要将此文件系统实际放置于的现有卷组。
   e. 设置文件系统的大小（**文件系统大小（以 512 字节块计）**（**数字**）字段）。 此大小按 512 字节块计数缩放，如果只需要为实例主目录创建文件系统，那么可以使用 180 000，大约 90 MB。如果需要完整地复制产品 DVD 映像以运行安装，那么可以创建它为值 2 000 000，大约 1 GB。
   f. 在**安装点**字段中输入此文件系统的安装点。在此示例中，安装点为 /db2home。
   g. 将**在系统重新启动时自动安装**字段设置为是。

      其余字段可保留缺省设置。
   h. 单击**确定**。

2. 导出 DB2 主文件系统。

   以 NFS 方式导出 /db2home 文件系统，以使它可用于将参与分区数据库系统的所有计算机。

   a. 输入 **smit nfs** 命令。
   b. 单击**网络文件系统 (NFS)** 图标。
   c. 单击**将目录添加至导出列表**图标。

d. 在**要导出的目录的路径名**字段中，输入要导出的路径名和目录（例如，/db2home）。

e. 在**允许具有 root 用户访问权的主机**字段中输入将要参与分区数据库系统的每个工作站的名称。 使用逗号（,）作为每个名称之间的定界符。例如，ServerA，ServerB，ServerC。如果是在使用高速互连，我们建议您在此字段中还要指定每个工作站的高速互连名称。其余字段可保留缺省设置。

f. 单击**确定**。

3. 注销。

4. 在参与的每台计算机中都安装 DB2 主文件系统。

   登录参与的*每台*计算机（ServerB、ServerC 和 ServerD），通过执行下列步骤来以 NFS 方式安装已导出的文件系统:

   a. 输入 **smit nfs** 命令。

   b. 单击**网络文件系统 (NFS)** 图标。

   c. 单击**添加文件系统以进行安装**图标。

   d. 在**安装点的路径名（路径）**字段中输入安装点的路径名。

      安装点的路径名就是应创建 DB2 主目录的位置。对于此示例，使用 /db2home。

   e. 在**远程目录的路径名**字段中输入远程目录的路径名。

      对于此示例，应输入您在**安装点的路径名（路径）**字段中输入的值。

   f. 在**远程目录所在的主机**字段中输入导出了文件系统的机器的*主机名*。

      此值是创建了要安装的文件系统的机器的主机名。

      为了提高性能，您可能想通过高速互连以 NFS 方式安装您创建的文件系统。如果想要使用高速互连来安装此文件系统，那么必须在**远程目录所在的主机**字段中输入其名称。

      请注意: 如果高速互连由于某种原因而变得不可用，那么参与分区数据库系统的每个工作站都将无法访问此 DB2 主目录。

   g. 将**立即安装，将条目添加至 /etc/filesystems 还是全部**字段设置为全部。

   h. 将 **/etc/filesystems 条目将在系统重新启动时安装目录**字段设置为是。

   i. 将**此 NFS 文件系统的方式**字段设置为读写。

   j. 将**对文件系统进行软安装还是硬安装**字段设置为硬安装。

      软安装表示计算机*将不会*无限期地尝试以远程方式安装该目录。硬安装表示机器将不停地尝试安装该目录。这会在系统崩溃时产生问题。建议将此字段设置为硬安装。

      其余字段可保留缺省设置。

   k. 确保此文件系统是在**是否允许在此文件系统中执行 SUID 和 sgid 程序**字段设置为是的情况下安装的。 这是缺省设置。

   l. 单击**确定**。

   m. 注销。

安装 DB2 pureScale Feature 时所需的用户 (Linux)

在 Linux 操作系统上运行 DB2 数据库环境需要两个用户和组。

开始之前

- 必须具有 root 用户权限才能创建用户和组。
- 如果使用安全性软件来管理用户和组，那么在定义 DB2 用户和组时可能还需要执行其他步骤。

关于此任务

需要两个用户才能创建 DB2 pureScale 实例:

- 一个用户作为实例所有者
- 一个用户作为受防护的用户

应该将两个不同的用户与两个不同的组配合使用。两个用户各自应该在所有主机上具有相同的 UID、GID、组名和主目录。如果要使用的任何用户在任何主机中存在，请确保他们具有匹配的属性。不必在开始安装之前创建这些所需的用户。可以在执行“DB2 安装”向导的面板期间创建这些用户，也可以在响应文件中指定这些用户。如果使用现有用户，那么他们必须在所有主机上都存在并且满足列示的要求。

下列指示信息中使用的用户名和组名是缺省值，并在下表中进行了说明。可以指定您自己的用户名和组名，但是它们必须遵循系统命名规则和 DB2 命名规则。

表 10. 缺省用户和组

必需的用户	用户名	组名
实例所有者	db2sdin1	db2iadm1
受防护的用户	db2sdfe1	db2fadm1

下表说明在下列指令中使用的用户名和组名。可以指定您自己的用户名和组名，但是它们必须遵循系统命名规则和 DB2 命名规则。

如果您打算使用“DB2 安装”向导来安装 DB2 数据库产品，那么“DB2 安装”向导将创建这些用户。

限制

您创建的用户名必须同时符合操作系统命名规则和 DB2 数据库系统命名规则。

您将在不同主机上创建的相同用户名必须具有相同的主目录。但是，该用户名不能已在任何主机上存在。如果使用现有用户名，那么这些用户名必须在所有主机上都存在并具有相同的用户标识 (uid)、组标识 (gid)、组名和 HOME 目录。

过程

要创建这些用户，请执行下列步骤:

1. 登录主机。
2. 通过输入下列命令，为实例所有者创建一个组（例如，db2iadm1），并创建一个将运行 UDF 或存储过程的组（例如，db2fadm1）:

```
groupadd -g 999 db2iadm1
  groupadd -g 998 db2fadm1
```

确保正在使用的特定号码当前不存在于任何机器上。

3. 通过使用下列命令，为前一步骤中创建的每个组创建一个用户。每个用户的主目录将是您先前创建且共享的 DB2 主目录（db2home）。

```
useradd -u 1004 -g db2iadm1 -m -d /db2home/db2inst1 db2inst1
useradd -u 1003 -g db2fadm1 -m -d /db2home/db2fenc1 db2fenc1
```

4. 通过输入下列命令，为创建的每个用户设置初始密码:

```
passwd db2inst1    passwd db2fenc1
```

5. 注销。
6. 作为已创建的每个用户（db2inst1 和 db2fenc1）登录主计算机。因为这是这些用户第一次登录系统，所以可能会提示您更改每个用户的密码。
7. 注销。
8. 在将参与数据库环境的每台计算机上创建完全相同的用户和组帐户。

在分区数据库环境中为安装 DB2 服务器创建必需用户（AIX）

在 AIX 操作系统上的分区数据库环境中运行 DB2 数据库需要三个用户和组。

开始之前

- 必须具有 root 用户权限才能创建用户和组。
- 如果使用安全性软件来管理用户和组，那么在定义 DB2 用户和组时可能还需要执行其他步骤。

关于此任务

下表说明在下列指令中使用的用户名和组名。可以指定您自己的用户名和组名，但是它们必须遵循系统命名规则和 DB2 命名规则。

如果您打算使用“DB2 安装”向导来安装 DB2 数据库产品，那么“DB2 安装”向导将创建这些用户。

表 11. 必需的用户和组

必需的用户	用户名	组名
实例所有者	db2inst1	db2iadm1
受防护的用户	db2fenc1	db2fadm1
DB2 管理服务器用户	dasusr1	dasadm1

如果 DB2 管理服务器用户是现有用户，那么此用户必须存在于所有参与的计算机上，然后才能安装。如果使用“DB2 安装”向导在拥有实例的计算机上创建新的 DB2 管理服务器用户，那么在安装响应文件期间也会在参与的计算机上创建该新用户（如有必要）。如果该用户在参与的计算机上已存在，那么它必须具有相同的主组。

限制

您创建的用户名必须同时符合操作系统命名规则和 DB2 数据库系统命名规则。

过程

要创建全部这三个用户，执行下列步骤:

1. 登录主计算机。
2. 通过输入下列命令，为实例所有者创建一个组（例如，db2iadm1），为将运行 UDF 或存储过程的组创建一个组（例如，db2fadm1），并为将拥有 DB2 管理服务器的组创建一个组（例如，dasadm1）:

   ```
   mkgroup id=999 db2iadm1
      mkgroup id=998 db2fadm1
      mkgroup id=997 dasadm1
   ```

3. 通过使用下列命令，为前一步骤中创建的每个组创建一个用户。每个用户的主目录将是您先前创建且共享的 DB2 主目录（db2home）。

   ```
   mkuser id=1004 pgrp=db2iadm1 groups=db2iadm1 home=/db2home/db2inst1
         core=-1 data=491519 stack=32767 rss=-1 fsize=-1 db2inst1
   mkuser id=1003 pgrp=db2fadm1 groups=db2fadm1 home=/db2home/db2fenc1
         db2fenc1
   mkuser id=1002 pgrp=dasadm1 groups=dasadm1 home=/home/dasusr1
         dasusr1
   ```

4. 通过输入下列命令，为创建的每个用户设置初始密码:

   ```
   passwd db2inst1
   passwd db2fenc1
   passwd dasusr1
   ```

5. 注销。
6. 作为已创建的每个用户（db2inst1、db2fenc1 和 dasusr1）登录主计算机。因为这是这些用户第一次登录系统，所以可能会提示您更改每个用户的密码。
7. 注销。
8. 在将参与分区数据库环境的每台计算机上创建完全相同的用户和组帐户。

第 7 章 安装 DB2 服务器产品

设置分区数据库环境

本主题描述如何设置分区数据库环境。将使用“DB2 安装”向导来安装实例拥有的数据库服务器和创建响应文件，然后再使用这些响应文件来创建参与的数据库服务器。

开始之前

注: 分区数据库环境在非 root 用户安装中不受支持。

- 确保您具有将需要复制到所有参与计算机中的 InfoSphere® 仓库激活 CD 许可证密钥。
- 将要参与分区数据库环境的每台计算机上相同数量的连续端口必须空闲。例如，如果分区数据库环境将由四台计算机组成，那么这些计算机上相同的四个连续端口均必须空闲。在创建实例期间，将在下列文件中保留端口，而且其数目与当前服务器上的逻辑分区数目相等: 在 Linux 和 UNIX 上，将保留在 `/etc/services` 中；而在 Windows 上，将保留在 `%SystemRoot%\system32\drivers\etc\services` 中。这些端口将由“快速通信管理器”使用。保留端口将使用以下格式:

  ```
  DB2_InstanceName
  DB2_InstanceName_1
  DB2_InstanceName_2
  DB2_InstanceName_END
  ```

  唯一必填的条目是起始端口 (DB2_InstanceName) 和结束端口 (DB2_InstanceName_END)。其他条目都保留在 services 文件中，以使其他应用程序不使用这些端口。
- 为了能支持多个参与 DB2 数据库服务器，要安装 DB2 的计算机必须属于一个可访问域。但是，可以将本地分区添加至该计算机，即使该计算机不属于某个域。
- 在 Linux 和 UNIX 系统上，分区数据库系统需要远程 shell 实用程序。DB2 数据库系统支持下列远程 shell 实用程序:
  - rsh
  - ssh

  缺省情况下，当对远程 DB2 节点执行命令时（例如，启动远程 DB2 数据库分区时），DB2 数据库系统就会使用 rsh。要使用 DB2 缺省值，必须安装 rsh-server 程序包。有关更多信息，请参阅数据库安全性指南中的『安装和使用 DB2 数据库管理器时的安全性注意事项』。

  如果选择使用远程 shell 实用程序 rsh，那么还必须安装并运行 inetd（或 xinetd）。如果选择使用远程 shell 实用程序 ssh，那么需要在完成 DB2 安装后立即设置 **DB2RSHCMD** 注册表变量。如果不设置此注册表变量，那么使用 rsh 实用程序。
- 在 Linux 和 UNIX 操作系统上，确保 `etc` 目录中的 `hosts` 文件中不包含“127.0.0.2”的条目（如果该 IP 地址映射至机器的标准主机名）。

关于此任务

数据库分区是数据库的一部分，它由自己的数据、索引、配置文件和事务日志组成。分区数据库是有两个或更多个分区的数据库。

过程

要设置分区数据库环境:

1. 使用“DB2 安装”向导来安装实例拥有的数据库服务器。有关详细的指示信息，请参阅适合您平台的相应“安装 DB2 服务器”主题。
   - 在**选择安装和/或响应文件创建**窗口中，确保选择**将安装设置保存在响应文件中**选项。完成安装之后，下面两个文件将被复制到在“DB2 安装”向导中指定的以下目录中: PROD_ESE.rsp 和 PROD_ESE_addpart.rsp。PROD_ESE.rsp 文件是实例拥有的数据库服务器的响应文件。PROD_ESE_addpart.rsp 文件是参与的数据库服务器的响应文件。
   - 在**为 DB2 实例设置分区选项**窗口上，务必选择**多个分区实例**并输入最大逻辑分区数。
2. 使 DB2 安装映像可用于分区数据库环境中的所有分区计算机。
3. 分发参与数据库服务器响应文件（PROD_ESE_addpart.rsp）。
4. 在参与的每台计算机上，使用 **db2setup** 命令（在 Linux 和 UNIX 上）或者 **setup**（在 Windows 上）来安装 DB2 数据库服务器:

   Linux 和 UNIX

   转至提供了 DB2 数据库产品代码的目录，然后运行:

   ```
   ./db2setup -r /responsefile_directory/response_file_name
   ```

   Windows

   ```
   setup -u x:\responsefile_directory\response_file_name
   ```

   例如，下面就是一个使用 PROD_ESE_addpart.rsp 作为响应文件的命令:

   Linux 和 UNIX

   转至提供了 DB2 数据库产品代码的目录，然后运行:

   ```
   ./db2setup -r /db2home/PROD_ESE_addpart.rsp
   ```

   其中 /db2home 是已经复制了响应文件的目录。

   Windows

   ```
   setup -u c:\resp_files\PROD_ESE_addpart.rsp
   ```

   其中 c:\resp_files\ 是已经复制了响应文件的目录。
5. （仅适用于 Linux 和 UNIX）配置 db2nodes.cfg 文件。DB2 安装仅保留您要对当前计算机使用的最大逻辑分区数量，但不配置 db2nodes.cfg 文件。如果不配置 db2nodes.cfg 文件，那么实例仍是单一分区实例。
6. 更新参与服务器上的 services 文件，以便为 DB2 实例定义相应的 FCM 端口。 该 services 文件位于下列位置:
   - /etc/services（在 Linux 和 UNIX 上）
   - %SystemRoot%\system32\drivers\etc\services（在 Windows 上）

7. 对于 Windows 2000 或更高版本上的分区数据库环境，请启动“DB2 远程命令服务”安全性功能部件以保护数据和资源。

   为了确保十分安全，请启动要作为委派的计算机（如果服务正在 LocalSystem 帐户的上下文下运行）或用户（如果服务正在用户的登录上下文下运行）。

   要启动“DB2 远程命令服务”安全性功能部件:

   a. 在域控制器上打开“Active Directory 用户和计算机”窗口，然后单击**开始**并选择**程序 > 管理工具 > Active Directory 用户和计算机**
   b. 在右窗口面板中，右键单击要启动的计算机或用户，然后选择**属性**
   c. 单击**常规**选项卡并选中**信任计算机作为委派**复选框。对于用户设置，请单击**帐户**选项卡并在**帐户选项**组中选中**信任帐户作为委派**复选框。确保未选中**帐户很敏感，不能进行委派**框。
   d. 单击**确定**以启动要作为委派的计算机或用户。

   对需要启动的每个计算机或用户重复上述步骤。必须重新启动计算机才能使安全性更改生效。

使用响应文件在参与的计算机上安装数据库分区服务器（Windows）

在此任务中，将使用您通过“DB2 安装”向导创建的响应文件，在参与的计算机上安装数据库分区服务器。

开始之前

- 已经在主计算机上使用“DB2 安装”向导安装了 DB2 副本。
- 您已经创建响应文件以安装在参与的计算机和将其复制到参与的计算机。
- 您在参与的计算机上必须具有管理权限。

过程

要使用响应文件安装其他数据库分区服务器:

1. 使用已为 DB2 安装定义的本地管理员帐户来登录将参与分区数据库环境的计算机。
2. 切换至包含 DB2 数据库产品 DVD 的目录。 例如:

   ```
   cd c:\db2dvd
   ```

   其中 db2dvd 表示包含 DB2 数据库产品 DVD 的目录的名称。
3. 从命令提示符处输入 **setup** 命令，如下所示:

   ```
   setup -u responsefile_directory\response_file_name
   ```

   在下列示例中，响应文件 Addpart.file 可在 c:\responsefile 目录中找到。此示例的命令将是:

   ```
   setup -u c:\reponsefile\Addpart.file
   ```
4. 完成安装之后，检查日志文件中的消息。您可以在 My Documents\DB2LOG\ 目录中找到该日志文件。您应该在日志文件的末尾看到与下列信息类似的输出:

   ```
   === Logging stopped: 5/9/2007  10:41:32 ===
   MSI (c) (C0:A8) [10:41:32:984]: Product: DB2
   Enterprise Server Edition - DB2COPY1 -- Installation
   operation completed successfully.
   ```

5. 当在主计算机上安装拥有实例的数据库分区服务器时，DB2 数据库产品根据指定的参与分区数据库环境的逻辑数据库分区服务器数目保留端口范围。缺省范围为四个端口。对于每个参与分区数据库环境的服务器，您必须为 FCM 端口手动配置 /etc/services 文件。FCM 端口范围取决于要在参与的计算机上使用的逻辑分区数目。至少需要两个条目：DB2_*instance* 和 DB2_*instance*_END。在参与的计算机上指定的 FCM 端口的其他要求：
   - 起始端口号必须与主计算机的起始端口号相匹配。
   - 后续端口必须按顺序进行编号。
   - 指定的端口号必须是空闲的。

结果

必须登录参与的每台计算机并重复这些步骤。

下一步做什么

如果想要 DB2 数据库产品能够访问本地计算机或网络中的另一计算机上的 DB2 文档，那么必须安装 *DB2 信息中心*。*DB2 信息中心*包含 DB2 数据库系统和 DB2 相关产品的文档。

使用响应文件在参与的计算机上安装数据库分区服务器（Linux 和 UNIX）

在此任务中，将使用您通过“DB2 安装”向导创建的响应文件，在参与的计算机上安装数据库分区服务器。

开始之前

- 已经在主计算机上使用“DB2 安装”向导安装了 DB2 数据库产品，并在参与的计算机上创建了用于安装的响应文件。
- 您在参与的计算机上必须具有 root 用户权限。

过程

要使用响应文件安装其他数据库分区服务器：

1. 作为 root 用户登录将参与分区数据库环境的计算机。
2. 切换至在其中复制 DB2 数据库产品 DVD 内容的目录。 例如：

   ```
   cd /db2home/db2dvd
   ```

3. 输入 **db2setup** 命令，如下所示：

   ```
   ./db2setup -r /responsefile_directory/response_file_name
   ```

   在本示例中，已将响应文件 AddPartitionResponse.file 保存到 /db2home 目录中。此情况的命令将是：

   ```
   ./db2setup -r /db2home/AddPartitionResponse.file
   ```

4. 完成安装之后，检查日志文件中的消息。

结果

必须登录参与的每台计算机并执行响应文件安装。

下一步做什么

如果想要 DB2 数据库产品能够访问本地计算机或网络中的另一计算机上的 DB2 数据库文档，那么必须安装 *DB2 信息中心*。*DB2 信息中心*包含 DB2 数据库系统和 DB2 数据库相关产品的文档。

第 8 章 安装之后

验证安装

验证分区数据库环境安装 (Windows)

为了验证是否成功安装了 DB2 数据库服务器，您将创建样本数据库并运行 SQL 命令来检索样本数据以及验证数据是否已分发至所有参与的数据库分区服务器。

开始之前

已完成所有安装步骤。

过程

要创建 SAMPLE 数据库:

1. 作为具有 SYSADM 权限的用户登录主计算机（ServerA）。
2. 输入 **db2sampl** 命令来创建 SAMPLE 数据库。

   处理此命令可能要花几分钟。当返回命令提示符时，该过程完成。

   创建 SAMPLE 数据库时，该数据库自动以数据库别名 SAMPLE 进行编目。
3. 输入 **db2start** 命令来启动数据库管理器。
4. 在 DB2 命令窗口中输入下列 DB2 命令来连接至 SAMPLE 数据库，并检索在部门 20 工作的所有职员的列表:

   ```
   db2 connect to sample
   db2 "select * from staff where dept = 20"
   ```
5. 要验证是否已将数据分发至数据库分区服务器，在 DB2 命令窗口中输入下列命令:

   ```
   db2 "select distinct dbpartitionnum(empno) from employee"
   ```

   输出将列示 employee 表使用的数据库分区。特定输出将取决于数据库中的数据库分区数以及创建 employee 表的表空间所使用的数据库分区组中的数据库分区数。

下一步做什么

在验证安装后，可除去 SAMPLE 数据库以释放磁盘空间。但是，如果打算使用样本应用程序，那么保留样本数据库很有用。

输入 **db2 drop database sample** 命令以删除 SAMPLE 数据库。

验证分区数据库服务器安装（Linux 和 UNIX）

使用 **db2val** 工具来验证安装文件、实例、数据库创建情况、与该数据库的连接以及分区数据库环境的状态，以验证 DB2 副本的核心功能。

有关详细信息，请参阅“验证 DB2 副本”。仅当至少有两个节点时，才会验证分区数据库环境的状态。另外，为了验证是否已成功安装 DB2 数据库服务器，您将创建样本数据库并运行 SQL 命令来检索样本数据以及验证数据是否已分发至所有参与的数据库分区服务器。

开始之前

在执行这些步骤之前，确保已完成所有安装步骤。

过程

要创建 SAMPLE 数据库:

1. 作为拥有实例的用户登录主计算机（ServerA）。 对于此示例，db2inst1 是拥有实例的用户。
2. 输入 **db2sampl** 命令来创建 SAMPLE 数据库。 缺省情况下，将在实例所有者的主目录中创建样本数据库。在本示例中，/db2home/db2inst1/ 是实例所有者的主目录。实例所有者的主目录是缺省数据库路径。

   处理此命令可能要花几分钟。没有完成消息；当返回命令提示符时，该过程完成。

   创建 SAMPLE 数据库时，该数据库自动以数据库别名 SAMPLE 进行编目。
3. 输入 **db2start** 命令来启动数据库管理器。
4. 在 DB2 命令窗口中输入下列 DB2 命令来连接至 SAMPLE 数据库，并检索在部门 20 工作的所有职员的列表:

   ```
   db2 connect to sample
   db2 "select * from staff where dept = 20"
   ```
5. 要验证是否已将数据分发至数据库分区服务器，在 DB2 命令窗口中输入下列命令:

   ```
   db2 "select distinct dbpartitionnum(empno) from employee"
   ```

   输出将列示 employee 表使用的数据库分区。特定输出将取决于:

   - 数据库中数据库分区的数目
   - 创建 employee 表的表空间所使用的数据库分区组中的数据库分区数

下一步做什么

在验证安装后，可除去 SAMPLE 数据库以释放磁盘空间。输入 **db2 drop database sample** 命令以删除 SAMPLE 数据库。

第 3 部分 实施和维护

第 9 章 创建数据库之前

设置分区数据库环境

必须在创建数据库之前决定创建多分区数据库。在作出数据库设计决定时，必须确定是否应利用数据库分区可以提供的性能提高。

关于此任务

在分区数据库环境中，仍然使用 **CREATE DATABASE** 命令或 sqlecrea() 函数来创建数据库。无论使用哪种方法，都可以通过 db2nodes.cfg 文件中列示的任何分区来发出请求。db2nodes.cfg 文件是数据库分区服务器配置文件。

除了在 Windows 操作系统环境上之外，可以使用任何编辑器来查看和更新数据库分区服务器配置文件 (db2nodes.cfg) 的内容。在 Windows 操作系统环境上，请使用 **db2ncrt** 和 **db2nchg** 命令来创建和更改数据库分区服务器配置文件

在创建多分区数据库之前，必须选择将作为数据库的目录分区的数据库分区。然后，可以直接从该数据库分区创建数据库，也可以从连接至该数据库分区的远程客户机创建数据库。您要连接并对其执行 **CREATE DATABASE** 命令的数据库分区成为该特定数据库的*目录分区*。

目录分区是用于存储所有系统目录表的数据库分区。对系统表的所有访问都必须通过此数据库分区进行。所有联合数据库对象（例如，包装器、服务器和昵称）都存储在此数据库分区上的系统目录表中。

若可能，应该在独立的实例中创建每个数据库。若不可能做到此点（即，必须在每个实例中创建多个数据库），应该将目录分区分布至可用的数据库分区中。这样做可以减少在单个数据库分区中对目录信息的争用。

注：应该定期备份目录分区，同时，因为其他数据会增加备份所需的时间，所以要避免将用户数据置于该节点上（任何可能的时候）。

当创建数据库时，它会在 db2nodes.cfg 文件中定义的所有数据库分区之间自动创建。

创建系统中的第一个数据库时，就会形成一个系统数据库目录。并追加有关您创建的任何其他数据库的信息。在 UNIX 上工作时，系统数据库目录是 sqldbdir，位于主目录下的 sqllib 目录中或安装 DB2 数据库的目录下面。在 UNIX 上工作时，由于组成分区数据库环境的所有数据库分区只有一个系统数据库目录，所以此目录必须位于共享文件系统（例如，UNIX 平台上的 NFS）上。在 Windows 上工作时，系统数据库目录位于实例目录中。

位于 sqldbdir 目录中的还有系统意向文件。它称为 sqldbins，用于确保数据库分区保持同步。该文件也必须位于共享文件系统中，因为所有数据库分区中只有一个目录。该文件由组成数据库的所有数据库分区共享。

必须修改配置参数，才能利用数据库分区。使用 **GET DATABASE CONFIGURATION** 和 **GET DATABASE MANAGER CONFIGURATION** 命令以了解特定数据库或数据库管理器配置文件中的

个别条目的值。要修改特定数据库或数据库管理器配置文件中的个别条目，可分别使用 **UPDATE DATABASE CONFIGURATION** 和 **UPDATE DATABASE MANAGER CONFIGURATION** 命令。

影响分区数据库环境的数据库管理器配置参数包括 **conn_elapse**、**fcm_num_buffers**、**fcm_num_channels**、**max_connretries**、**max_coordagents**、**max_time_diff**、**num_poolagents** 和 **start_stop_time**。

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 12. 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。

DB2 节点配置文件的格式

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。

Linux 和 UNIX 操作系统上的 db2nodes.cfg 文件的格式如下:

```
dbpartitionnum hostname logicalport netname resourcesetname
```

在下一节中定义了 *dbpartitionnum*、*hostname*、*logicalport*、*netname* 和 *resourcesetname*。

Windows 操作系统上的 db2nodes.cfg 文件的格式如下:

```
dbpartitionnum hostname computername logicalport netname resourcesetname
```

在 Windows 操作系统上，**db2ncrt** 或 START DBM ADD DBPARTITIONNUM 命令会将这些条目添加至 db2nodes.cfg。还可以通过 **db2nchg** 命令来修改这些条目。您不应该直接添加这些行或编辑此文件。

dbpartitionnum

这是一个唯一号码，其范围是 0 到 999，用来标识分区数据库系统中的数据库分区服务器。

要扩充分区数据库系统，对每个数据库分区服务器，向 db2nodes.cfg 文件添加一个条目。为其他数据库分区服务器选择的 *dbpartitionnum* 值必须按升序排序，但是此序列中的值之间可以存在间隔。如果您打算添加逻辑分区服务器，并且希望使节点在此文件中保持按逻辑分组，那么可以选择使 *dbpartitionnum* 值之间保持一定间隔。

此条目是必需的。

hostname

供 FCM 使用的数据库分区服务器的 TCP/IP 主机名。此条目是必需的。*强烈*建议使用规范主机名。

如果 db2nodes.cfg 文件提供的是主机名而不是 IP 地址，那么数据库管理器将以动态方式尝试解析主机名。解析可能是本地解析或通过在已注册域名服务器（DNS）上查询来进行解析，这由机器上的操作系统设置确定。

从 DB2 版本 9.1 开始支持 TCP/IPv4 和 TCP/IPv6 协议。用于解析主机名的方法已更改。

如果在 db2nodes.cfg 文件中定义了短名称，那么版本 9.1 之前的发行版将按 db2nodes.cfg 文件中的定义解析字符串，而版本 9.1 或更高版本将尝试解析标准域名（FQDN）。如果指定对标准主机名称配置的短名称，那么可能导致解析主机名的进程中出现不必要的延迟。

为避免需要解析主机名的 DB2 命令中出现任何延迟，请使用下列任一变通方法:

1. 如果在 db2nodes.cfg 文件和操作系统主机名文件中指定了短名称，那么应对操作系统主机文件中的主机名指定短名称和标准域名。
2. 要仅在知道 DB2 服务器侦听 IPv4 端口时才使用 IPv4 地址，请发出以下命令:

   ```
   db2 catalog tcpip4 node db2tcp2 remote 192.0.32.67  server db2inst1
   with "Look up IPv4 address from 192.0.32.67"
   ```

3. 要仅在知道 DB2 服务器侦听 IPv6 端口时才使用 IPv6 地址，请发出以下命令:

   ```
   db2 catalog tcpip6 node db2tcp3 1080:0:0:0:8:800:200C:417A
   server 50000     with "Look up IPv6 address from
   1080:0:0:0:8:800:200C:417A"
   ```

logicalport

指定数据库分区服务器的逻辑端口号。此字段用来在正在运行逻辑数据库分区服务器的工作站上指定特定数据库分区服务器。

在安装时，DB2 会在 /etc/services 文件中保留某个端口范围（例如，60000 - 60003）以用于分区间通信。db2nodes.cfg 中的 *logicalport* 字段指定您要将该范围中的哪个端口分配给特定的逻辑分区服务器。

如果此字段中无任何条目，那么缺省值为 0。但是，如果对 *netname* 字段添加一个条目，那么必须对 *logicalport* 字段输入一个数字。

如果正在使用逻辑数据库分区，那么指定的 *logicalport* 值*必须*从 0 开始，并按升序依次递增（例如，0、1、2）。

此外，如果为一个数据库分区服务器指定 *logicalport* 条目，那么必须为 db2nodes.cfg 文件中列示的每个数据库分区服务器指定 *logicalport*。

每个物理服务器都必须具有逻辑节点 0。

仅当未使用逻辑数据库分区或高速互连时，此字段才是可选的。

netname

指定用于 FCM 通信的高速互连的主机名或 IP 地址。

如果为此字段指定了一个条目，那么数据库分区服务器之间的所有通信（除了由于 **db2start**、**db2stop** 和 **db2_all** 命令而进行的通信之外）都是通过高速互连来处理的。

仅当您要使用高速互连来进行数据库分区通信时，才需要此参数。

resourcesetname

resourcesetname 定义应在其中启动节点的操作系统资源。*resourcesetname* 用于进程相似性支持、用于多逻辑节点（MLN）。此支持与先前被称为 quadname 的字符串类型字段一起提供。

此参数仅在 AIX、HP-UX 和 Solaris 操作系统上受支持。

在 AIX 上，此概念被称作“资源集”，而在 Solaris 操作系统上，它被称为“项目”。有关资源管理的更多信息，请参阅操作系统文档。

在 HP-UX 上，*resourcesetname* 参数是 PRM 组的名称。请参阅 HP 的“HP-UX Process Resource Manager User Guide（B8733-90007）”文档以获取更多信息。

在 Windows 操作系统上，可以通过 **DB2PROCESSORS** 注册表变量来定义逻辑节点的进程相似性。

在 Linux 操作系统上，*resourcesetname* 列定义与系统上“非一致性内存访问”（NUMA）节点相对应的一个数字。除了具有 NUMA 策略支持的 2.6 内核之外，系统实用程序 **numactl** 必须可用。

如果使用 *resourcesetname* 参数，那么必须指定 *netname* 参数。

示例配置

使用下面的示例配置来确定环境的相应配置。

一台计算机，四个数据库分区服务器

如果未使用集群环境，且想要在一台名为 ServerA 的物理工作站上具有四个数据库分区服务器，那么应对 db2nodes.cfg 文件作如下更新:

```
0          ServerA        0
1          ServerA        1
2          ServerA        2
3          ServerA        3
```

两台计算机，每台计算机一个数据库分区服务器

如果想要让分区数据库系统包含两个物理工作站：ServerA 和 ServerB，那么应对 db2nodes.cfg 文件作如下更新：

```
0          ServerA        0
1          ServerB        0
```

两台计算机，一台计算机上有三个数据库分区服务器

如果想要分区数据库系统包含两个物理工作站：ServerA 和 ServerB，并且 ServerA 要运行 3 个数据库分区服务器，那么应对 db2nodes.cfg 文件作如下更新：

```
4          ServerA        0
6          ServerA        1
8          ServerA        2
9          ServerB        0
```

两台计算机，带有高速交换机的三个数据库分区服务器

如果想要分区数据库系统包含两台计算机：ServerA 和 ServerB（且ServerB 运行两个数据库分区服务器），并且使用名为 switch1 和 switch2 的高速互连，那么应对 db2nodes.cfg 文件作如下更新：

```
0          ServerA        0              switch1
1          ServerB        0              switch2
2          ServerB        1              switch2
```

使用 resourcesetname 的示例

这些限制适用于以下示例：

- 本示例说明在配置中没有高速互连时 *resourcesetname* 的使用。
- *netname* 是第四列，在该列没有交换机名而您却想要使用 *resourcesetname* 的情况下，您还可以在该列上指定 *hostname*。第五个参数是 *resourcesetname*（如果已定义）。此资源组规范只可显示为 db2nodes.cfg 文件中的第五列。这意味着，要指定资源组，还必须输入第四列。第四列打算用于高速交换机。
- 如果没有高速交换机或者不想使用它，那么必须输入 *hostname*（与第二列相同）。换句话说，DB2 数据库管理系统不支持 db2nodes.cfg 文件中存在列间隔（或列交换）。此限制以前适用于前三列，现在它适用于所有五列。

AIX 示例

此处是如何为 AIX 操作系统设置资源集的示例。

在此示例中，有一个具有 32 个处理器和 8 个逻辑数据库分区（MLN）的物理节点。此示例说明如何为每个 MLN 提供进程相似性。

1. 在 /etc/rset 中定义资源集：

```
DB2/MLN1:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00000,sys/cpu.00001,sys/cpu.00002,sys/cpu.00003

DB2/MLN2:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00004,sys/cpu.00005,sys/cpu.00006,sys/cpu.00007
```

```
DB2/MLN3:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00008,sys/cpu.00009,sys/cpu.00010,sys/cpu.00011

DB2/MLN4:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00012,sys/cpu.00013,sys/cpu.00014,sys/cpu.00015

DB2/MLN5:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00016,sys/cpu.00017,sys/cpu.00018,sys/cpu.00019

DB2/MLN6:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00020,sys/cpu.00021,sys/cpu.00022,sys/cpu.00023

DB2/MLN7:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00024,sys/cpu.00025,sys/cpu.00026,sys/cpu.00027

DB2/MLN8:
    owner     = db2inst1
    group     = system
    perm      = rwr-r-
    resources = sys/cpu.00028,sys/cpu.00029,sys/cpu.00030,sys/cpu.00031
```

2. 通过输入以下命令来启用内存相似性:

```
vmo -p -o memory_affinity=1
```

3. 允许实例使用资源集:

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

4. 将资源集名称作为第五列添加到 db2nodes.cfg:

```
1 regatta 0 regatta DB2/MLN1
2 regatta 1 regatta DB2/MLN2
3 regatta 2 regatta DB2/MLN3
4 regatta 3 regatta DB2/MLN4
5 regatta 4 regatta DB2/MLN5
6 regatta 5 regatta DB2/MLN6
7 regatta 6 regatta DB2/MLN7
8 regatta 7 regatta DB2/MLN8
```

HP-UX 示例

此示例说明如何使用 PRM 组在使用 4 个 CPU 和 4 个 MLN 的机器上分享 CPU 资源，为每个 MLN 设置 24% 的 CPU 份额，为其他应用程序留下 4%。DB2 实例名为 db2inst1。

1. 编辑 /etc/prmconf 的 GROUP 段:

```
 OTHERS:1:4::
db2prm1:50:24::
 db2prm2:51:24::
  db2prm3:52:24::
 db2prm4:53:24::
```

2. 向 /etc/prmconf 添加实例所有者条目:

```
db2inst1::::OTHERS,db2prm1,db2prm2,db2prm3,db2prm4
```

3. 通过输入以下命令来初始化组并启用 CPU 管理器:

```
prmconfig -i
   prmconfig -e CPU
```

4. 将 PRM 组名作为第五列添加到 db2nodes.cfg:

```
1 voyager 0 voyager db2prm1
   2 voyager 1 voyager db2prm2
   3 voyager 2 voyager db2prm3
   4 voyager 3 voyager db2prm4
```

可以使用交互式 GUI 工具 **xprm** 执行 PRM 配置（步骤 1-3）。

Linux 示例

在 Linux 操作系统上，*resourcesetname* 列定义与系统上“非一致性内存访问”（NUMA）节点相对应的一个数字。作为对支持 NUMA 策略的 2.6 内核的补充，还必须具有 **numactl** 系统实用程序。有关 Linux 操作系统上的 NUMA 支持的更多信息，请参阅 **numactl** 的联机帮助页。

本示例说明如何设置一个具有四个 NUMA 节点的计算机，并使每个逻辑节点都与一个 NUMA 节点相关联。

1. 确保系统上具备 NUMA 功能。
2. 发出下列命令:

```
$ numactl --hardware
```

将显示与以下内容相似的输出:

```
available: 4 nodes (0-3)
node 0 size: 1901 MB
node 0 free: 1457 MB
node 1 size: 1910 MB
node 1 free: 1841 MB
node 2 size: 1910 MB
node 2 free: 1851 MB
node 3 size: 1905 MB
node 3 free: 1796 MB
```

3. 在此示例中，系统上有四个 NUMA 节点。按如下所示编辑 db2nodes.cfg 文件，以使每个 MLN 都与系统上的一个 NUMA 节点相关联:

```
0 hostname 0 hostname 0
1 hostname 1 hostname 1
2 hostname 2 hostname 2
3 hostname 3 hostname 3
```

Solaris 示例

此处是如何为 Solaris V9 设置项目的示例。

在此示例中，有一个带有 8 个处理器的物理节点：其中 1 个 CPU 将用于缺省项目，3 个 CPU 由应用程序服务器使用，4 个 CPU 用于 DB2。实例名为 db2inst1。

1. 使用编辑器创建资源池配置文件。对于此示例，该文件将被称为 pool.db2。其内容如下:

```
create system hostname
create pset pset_default (uint pset.min = 1)
create pset db0_pset (uint pset.min = 1; uint pset.max = 1)
create pset db1_pset (uint pset.min = 1; uint pset.max = 1)
create pset db2_pset (uint pset.min = 1; uint pset.max = 1)
create pset db3_pset (uint pset.min = 1; uint pset.max = 1)
create pset appsrv_pset (uint pset.min = 3; uint pset.max = 3)
create pool pool_default (string pool.scheduler="TS";
     boolean pool.default = true)
create pool db0_pool (string pool.scheduler="TS")
create pool db1_pool (string pool.scheduler="TS")
create pool db2_pool (string pool.scheduler="TS")
create pool db3_pool (string pool.scheduler="TS")
create pool appsrv_pool (string pool.scheduler="TS")
associate pool pool_default (pset pset_default)
associate pool db0_pool (pset db0_pset)
associate pool db1_pool (pset db1_pset)
associate pool db2_pool (pset db2_pset)
associate pool db3_pool (pset db3_pset)
associate pool appsrv_pool (pset appsrv_pset)
```

2. 编辑 /etc/project 文件以添加 DB2 项目和 appsrv 项目，如下所示:

```
system:0::::
user.root:1::::
noproject:2::::
default:3::::
group.staff:10::::
appsrv:4000:App Serv project:root::project.pool=appsrv_pool
db2proj0:5000:DB2 Node 0 project:db2inst1,root::project.pool=db0_pool
db2proj1:5001:DB2 Node 1 project:db2inst1,root::project.pool=db1_pool
db2proj2:5002:DB2 Node 2 project:db2inst1,root::project.pool=db2_pool
db2proj3:5003:DB2 Node 3 project:db2inst1,root::project.pool=db3_pool
```

3. 创建资源池: # poolcfg -f pool.db2。
4. 激活资源池: # pooladm -c
5. 将项目名作为第五列添加到 db2nodes.cfg 文件:

```
0 hostname 0 hostname db2proj0
1 hostname 1 hostname db2proj1
2 hostname 2 hostname db2proj2
3 hostname 3 hostname db2proj3
```

指定分区数据库环境中的机器列表

缺省情况下，从数据库分区配置文件 db2nodes.cfg 中获取计算机列表。

关于此任务

注: 在 Windows 上，要避免将不一致引入该数据库分区配置文件，不要以手动方式对其进行编辑。要获取实例中的计算机列表，可使用 **db2nlist** 命令。

过程

要覆盖 db2nodes.cfg 中的计算机列表，请执行下列操作:

- 通过导出（在 Linux 和 UNIX 操作系统上）或设置（在 Windows 上）环境变量 **RAHOSTFILE**，指定包含计算机列表的文件的路径名。
- 通过导出（在 Linux 和 UNIX 操作系统上）或设置（在 Windows 上）环境变量 **RAHOSTLIST**，明确指定该列表为由空格分隔的一连串名称。

注: 如果这两个环境变量都已指定，那么 **RAHOSTLIST** 具有优先顺序。

除去分区数据库环境内机器列表中的重复条目

如果正在一台计算机上运行多个逻辑数据库分区服务器，那么 db2nodes.cfg 文件包含该计算机的多个条目。

关于此任务

在这种情况下，**rah** 命令需要知道您是希望该命令在每台计算机上只执行一次，还是对 db2nodes.cfg 文件中列示的每个逻辑数据库分区均执行一次。使用 **rah** 命令来指定计算机。使用 **db2_all** 命令来指定逻辑数据库分区。

注: 在 Linux 和 UNIX 操作系统上，如果指定计算机，那么 **rah** 将会正常地从计算机列表中删除重复的项，例外情况是: 如果指定逻辑数据库分区，那么 **db2_all** 会将下列赋值附加到您的命令之前:

```
export DB2NODE=nnn（对于 Korn shell 程序语法）
```

其中 *nnn* 是 db2nodes.cfg 文件的相应行中的数据库分区号，以便将命令路由至所希望的数据库分区服务器。

当指定逻辑数据库分区时，可以使用 <<-*nnn*< 和 <<+*nnn*< 前缀序列来限制列表包括除某个逻辑数据库分区外的所有逻辑数据库分区，或只指定一个逻辑数据库分区。如果想要首先运行对数据库分区进行编目的命令，并在该命令完成时，在所有其他数据库分区服务器上运行同一命令（可能以并行方式），那么可能要执行此操作。当运行 **RESTART DATABASE** 命令时，它常常是必需的。需要知道目录分区的数据库分区号才能执行此操作。

如果使用 **rah** 命令来执行 **RESTART DATABASE**，重复条目就会从计算机列表中被删除。但是，如果指定 " 前缀，那么不删除重复项，因为认为使用 " 前缀会隐含发送至每个数据库分区服务器，而不是发送至每台计算机。

更新节点配置文件（Linux 和 UNIX）

在 DB2 分区数据库环境中，此任务提供了用于更新 db2nodes.cfg 文件的步骤，以包括参与的计算机的条目。

开始之前

- 必须在所有参与的计算机上安装 DB2 数据库产品。
- DB2 实例必须存在于主计算机上。
- 您必须是具有 SYSADM 权限的用户。
- 如果是以下任何一种情况，请查看 *DB2 节点配置文件的格式*主题中提供的配置示例和文件格式信息:
  - 您计划使用高速交换机在数据库分区服务器之间进行通信
  - 您的分区配置将具有多个逻辑分区

关于此任务

节点配置文件 (db2nodes.cfg) 位于实例所有者的主目录中，它包含一些配置信息，告诉 DB2 数据库系统有哪些服务器参与分区数据库环境的实例。分区数据库环境中的每个实例都有一个 db2nodes.cfg 文件。

对于每个参与实例的服务器，db2nodes.cfg 文件必须包含一个条目。当创建实例时，会自动创建 db2nodes.cfg 文件并对拥有实例的服务器添加条目。

例如，在拥有实例的服务器 ServerA 上使用“DB2 安装”向导创建了 DB2 实例时，db2nodes.cfg 文件将更新为以下内容:

```
0     ServerA     0
```

限制

在“过程”部分的步骤中使用的主机名必须是标准主机名。

过程

要更新 db2nodes.cfg 文件:

1. 作为实例所有者登录。 例如，db2inst1 是这些步骤中的实例所有者。
2. 通过输入以下命令确保已停止 DB2 实例:

   INSTHOME/sqllib/adm/db2stop

   其中 *INSTHOME* 是实例所有者的主目录（db2nodes.cfg 文件在实例运行时被锁定，并且仅当实例停止时才可以编辑该文件）。

   例如，如果实例主目录为 /db2home/db2inst1，那么输入以下命令:

   ```
   /db2home/db2inst1/sqllib/adm/db2stop
   ```

3. 对于每个 DB2 实例，向 .rhosts 文件添加一个条目。 通过添加以下项来更新文件:

   ```
   hostname db2instance
   ```

   其中 *hostname* 是数据库服务器的 TCP/IP 主机名，*db2instance* 是用来访问数据库服务器的实例的名称。

4. 向每个参与的服务器的 db2nodes.cfg 文件添加一个条目。 当第一次查看 db2nodes.cfg 文件时，它应该包含类似于以下内容的条目:

   ```
   0     ServerA     0
   ```

   此条目包括数据库分区服务器号（节点号）、数据库分区服务器驻留的服务器的 TCP/IP 主机名以及数据库分区服务器的逻辑端口号。

   例如，如果正在对分区配置（有四台计算机，每台计算机上安装一个数据库分区服务器）进行安装，那么已更新的 db2nodes.cfg 应该类似于以下内容:

   ```
   0     ServerA     0
   1     ServerB     0
   2     ServerC     0
   3     ServerD     0
   ```

5. 完成更新 db2nodes.cfg 文件时，请输入 *INSTHOME*/sqllib/adm/db2start 命令，其中 *INSTHOME* 是实例所有者的主目录。 例如，如果实例主目录为 /db2home/db2inst1，那么输入以下命令:

   ```
   /db2home/db2inst1/sqllib/adm/db2start
   ```

6. 注销。

设置多逻辑分区

在几种情况下，在同一台计算机上运行数个数据库分区服务器很有好处。

这意味着配置包含的数据库分区数可以多于计算机数。在这些情况下，如果这些逻辑分区参与同一实例，我们便称该计算机正在运行*多逻辑分区*。如果它们参与不同的实例，那么此计算机不主管多逻辑分区。

借助多逻辑分区支持，您可以从三种类型的配置中进行选择:

- 标准配置，即每台计算机只有一个数据库分区服务器
- 多逻辑分区配置，即计算机有多个数据库分区服务器
- 在数台计算机的每一台上运行数个逻辑分区的配置

当系统在具有对称多处理器（SMP）体系结构的计算机上运行查询时，使用多逻辑分区的配置非常有用。如果计算机出现故障，那么在计算机上配置多逻辑分区的能力也很有用。如果一台计算机出现故障（导致其上的数据库分区服务器失败），那么可以使用 **START DBM DBPARTITIONNUM** 命令来在另一机器上重新启动数据库分区服务器。这确保用户数据保持可用。

另一个好处是多逻辑分区可使用 SMP 硬件配置。另外，因为数据库分区较小，所以当执行诸如备份和复原数据库和表空间以及创建索引之类的任务时，可以获得较佳的性能。

配置多逻辑分区

可使用两种方法来配置多逻辑分区。

关于此任务

- 在 db2nodes.cfg 文件中配置逻辑分区（数据库分区）。然后，可使用 **db2start** 命令或它的关联 API 启动所有逻辑分区和远程分区。

  注: 对于 Windows，如果系统中没有数据库，那么必须使用 **db2ncrt** 来添加数据库分区；或者，如果有一个或多个数据库，那么应使用 **db2start addnode** 命令。在 Windows 中，绝对不应手动编辑 db2nodes.cfg 文件。
- 在另一个处理器上重新启动一个逻辑分区，其他逻辑分区已在该处理器上运行。这允许您覆盖在 db2nodes.cfg 中为逻辑分区指定的主机名和端口号。

要在 db2nodes.cfg 中配置一个逻辑数据库分区，您必须在该文件中建立一个条目以便为该数据库分区分配一个逻辑端口号。以下是应使用的语法:

```
nodenumber hostname logical-port netname
```

对于 IBM DB2 pureScale Feature，请确保有一个节点号为“0”的成员。

注: 对于 Windows，如果系统中没有数据库，那么必须使用 **db2ncrt** 来添加数据库分区；或者，如果有一个或多个数据库，那么应使用 **db2start addnode** 命令。在 Windows 中，绝对不应手动编辑 db2nodes.cfg 文件。

Windows 上 db2nodes.cfg 文件的格式与 UNIX 上同一文件的格式不同。在 Windows 上，列格式为:

```
nodenumber hostname computername logical_port netname
```

使用主机名的标准名称。/etc/hosts 文件也应该使用标准名称。如果未在 db2nodes.cfg 文件和 /etc/hosts 文件中使用标准名称，那么可能接收到错误消息 SQL30082N RC=3。

必须确保在 etc 目录的 services 文件中为 FCM 通信定义了足够的端口。

启用分区间的查询并行性

根据数据库分区的数目以及数据在这些数据库分区上的分布，分区间并行性自动生效。

关于此任务

必须修改配置参数，以利用数据库分区内或非分区数据库内的并行性。例如，可以使用分区内并行性来利用对称多处理器 (SMP) 机器上的多个处理器。

过程

- 要在装入数据时启用并行性:

  LOAD 实用程序自动利用并行性，也可在 **LOAD** 命令上使用下列参数:

  – **CPU_PARALLELISM**
  – **DISK_PARALLELISM**

  在分区数据库环境中，当对多个数据库分区定义目标表时，用于数据装入的分区间并行性就会自动发生。用于数据装入的分区间并行性可通过指定 **OUTPUT_DBPARTNUMS** 来覆盖。LOAD 实用程序还会根据目标数据库分区大小来智能地启用数据库分区并行性。可使用 **MAX_NUM_PART_AGENTS** 来控制 load 实用程序选择的最大并行度。可在指定 **ANYORDER** 的同时指定 **PARTITIONING_DBPARTNUMS** 以覆盖数据库分区并行性。
- 要在创建索引时启用并行性:
  – 表必须足够大，以便能从并行性受益
  – 在 SMP 计算机上必须启用多个处理器。
- 要在备份数据库或表空间时启用 I/O 并行性:
  – 使用多个目标媒体。
  – 通过定义多个容器配置用于并行 I/O的表空间，或将单个容器与多个磁盘配合使用，并适当地使用 **DB2_PARALLEL_IO** 注册表变量。如果想要利用并行 I/O，那么必须考虑在定义任何容器之前必须完成的操作。不能在您认为需要时才执行此操作；必须在需要备份数据库或表空间之前进行计划。
  – 在 **BACKUP** 命令上使用 **PARALLELISM** 参数以指定并行度。
  – 在 **BACKUP** 命令上使用 **WITH** num-buffers **BUFFERS** 参数以保证提供足够的缓冲区来满足该并行度。缓冲区数应比已有的目标媒体数与选择的并行度之和略大。

同时使用满足以下条件的备份缓冲区大小:

- 尽可能大。4 MB 或 8 MB（1024 或 2048 页）比较合适。
- 至少等于要备份的表空间（扩展数据块大小 * 容器数之积）的最大产品。

- 要在复原数据库或表空间时启用 I/O 并行性:
  - 使用多个源媒体。
  - 配置用于并行 I/O 的表空间。在定义容器之前，必须决定是否使用此选项。不能在您认为需要时才执行此操作；必须在需要复原数据库或表空间之前进行计划。
  - 在 **RESTORE** 命令上使用 **PARALLELISM** 参数以指定并行度。
  - 在 **RESTORE** 命令上使用 **WITH** num-buffers **BUFFERS** 参数以保证提供足够的缓冲区来满足该并行度。缓冲区数应比已有的目标媒体数与选择的并行度之和略大。

    同时使用满足以下条件的复原缓冲区大小:

    - 尽可能大。4 MB 或 8 MB（1024 或 2048 页）比较合适。
    - 至少等于要复原的表空间中（扩展数据块大小 * 容器数之积）的最大产品。
    - 等于备份缓冲区大小或是其偶数倍。

对查询启用分区内并行性

要启用分区内查询并行性，请修改一个或多个数据库配置参数或数据库管理器配置参数、预编译或绑定选项或者专用寄存器。或者，对 CREATE 或 ALTER WORKLOAD 语句或者 ADMIN_SET_INTRA_PARALLEL 过程使用 MAXIMUM DEGREE 选项在事务级别启用或禁用分区内并行性。

开始之前

使用以下控制来指定优化器要使用的分区内并行度:

- CURRENT DEGREE 专用寄存器（用于动态 SQL）
- DEGREE 绑定选项（用于静态 SQL）
- **dft_degree** 数据库配置参数（提供前两个参数的缺省值）

使用以下控制来限制运行时的分区内并行度。运行时设置将覆盖优化器设置。

- **max_querydegree** 数据库管理器配置参数
- SET RUNTIME DEGREE 命令
- MAXIMUM DEGREE 工作负载选项

使用以下控制来启用或禁用分区内并行度:

- **intra_parallel** 数据库管理器配置参数
- ADMIN_SET_INTRA_PARALLEL 存储过程
- MAXIMUM DEGREE 工作负载选项（设置为 1）

关于此任务

使用 **GET DATABASE CONFIGURATION** 或 **GET DATABASE MANAGER CONFIGURATION** 命令来查找特定数据库或实例配置文件中的个别条目的值。要修改这些条目中的一个或多个，应使用 **UPDATE DATABASE CONFIGURATION** 或 **UPDATE DATABASE MANAGER CONFIGURATION** 命令。

intra_parallel

这是一个用来指定数据库管理器是否可以使用分区内并行性的数据库管理器配置参数。缺省值为 NO，表示此实例中运行的应用程序未启用分区内并行性。例如:

```
update dbm cfg using intra_parallel yes;
get dbm cfg;
```

max_querydegree

指定用于在此实例上运行的任何 SQL 语句的最大程度分区内并行性的数据库管理器配置参数。在数据库分区中运行并行操作时，SQL 语句将不会使用大于此值的值。缺省值为 -1，表示系统使用由优化器确定的分区内并行度，而不使用用户指定的值。例如:

```
update dbm cfg using max_querydegree any;
get dbm cfg;
```

还必须将数据库管理器配置参数 **intra_parallel** 设置为 YES，才能使用 **max_querydegree** 的值。

dft_degree

这是用于指定预编译或绑定选项 **DEGREE** 和 CURRENT DEGREE 专用寄存器的缺省值的数据库配置参数。缺省值为 1。值 -1（或者 ANY）表示系统使用由优化器确定的分区内并行度。例如:

```
connect to sample;
update db cfg using dft_degree -1;
get db cfg;
connect reset;
```

DEGREE 这是用于指定在对称多处理 (SMP) 系统上执行静态 SQL 语句时采用的分区内并行度的预编译或绑定选项。例如:

```
connect to prod;
prep demoapp.sqc bindfile;
bind demoapp.bnd degree 2;
...
```

CURRENT DEGREE

这是用于指定在执行动态 SQL 语句时采用的分区内并行度的专用寄存器。使用 SET CURRENT DEGREE 语句对 CURRENT DEGREE 专用寄存器指定值。例如:

```
connect to sample;
set current degree = '1';
connect reset;
```

还必须将数据库管理器配置参数 **intra_parallel** 设置为 YES，才能使用分区内并行性。如果此参数的值设置为 NO，那么将忽略此专用寄存器的值，并且该语句将不会使用分区内并行性。可以通过设置 MAXIMUM DEGREE 工作负载属性来覆盖工作负载中的数据库管理器配置参数 **intra_parallel** 和 CURRENT DEGREE 专用寄存器的值。

MAXIMUM DEGREE

用于指定工作负载的最大运行时并行度的 CREATE WORKLOAD 语句（或者 ALTER WORKLOAD 语句）选项。

例如，假定 bank_trans 是一个已打包的应用程序，主要执行 OLTP 短事务；bank_report 是另一个已打包的应用程序，用于运行复杂查询以生成商业智能 (BI) 报告。这两个应用程序都不可修改，并且这两个应用程序都在并行度为 4

的情况下绑定至数据库。在 bank_trans 正在运行时，会将它分配给工作负载 trans，这将禁用分区内并行性。此 OLTP 应用程序将运行，并不存在与分区内并行性开销相关联的任何性能下降。在 bank_report 正在运行时，会将它分配给工作负载 bi，这将启用分区内并行性并指定最大运行时并行度 8。因为此程序包的编译度为 4，所以此应用程序中的静态 SQL 语句将仅以并行度 4 运行。如果此 BI 应用程序包含动态 SQL 语句，并且未设置 CURRENT DEGREE 专用寄存器，那么这些语句将以并行度 8 运行。

```
connect to sample;

create workload trans
  applname('bank_trans')
  maximum degree 1
  enable;

create workload bi
  applname('bank_report')
  maximum degree 8
  enable;

connect reset;
```

ADMIN_SET_INTRA_PARALLEL

这是用于对数据库应用程序启用或禁用分区内并行性的过程。尽管是在当前事务中调用此过程，但是它将在下一个事务开始时才生效。例如，假定 demoapp 应用程序中包含以下代码，该应用程序将 ADMIN_SET_INTRA_PARALLEL 过程与静态 SQL 语句和动态 SQL 语句配合使用:

```
EXEC SQL CONNECT TO prod;

// Disable intrapartition parallelism:
EXEC SQL CALL SYSPROC.ADMIN_SET_INTRA_PARALLEL('NO');
// Commit so that the effect of this call
// starts in the next statement:
EXEC SQL COMMIT;

// All statements in the next two transactions run
// without intrapartition parallelism:
strcpy(stmt, "SELECT deptname FROM org");
EXEC SQL PREPARE rstmt FROM :stmt;
EXEC SQL DECLARE c1 CURSOR FOR rstmt;
EXEC SQL OPEN c1;
EXEC SQL FETCH c1 INTO :deptname;
EXEC SQL CLOSE c1;
...
// New section for this static statement:
EXEC SQL SELECT COUNT(*) INTO :numRecords FROM org;
...
EXEC SQL COMMIT;

// Enable intrapartition parallelism:
EXEC SQL CALL SYSPROC.ADMIN_SET_INTRA_PARALLEL('YES');
// Commit so that the effect of this call
// starts in the next statement:
EXEC SQL COMMIT;

strcpy(stmt, "SET CURRENT DEGREE='4'");
// Set the degree of parallelism to 4:
EXEC SQL EXECUTE IMMEDIATE :stmt;

// All dynamic statements in the next two transactions
// run with intrapartition parallelism and degree 4:
strcpy(stmt, "SELECT deptname FROM org");
EXEC SQL PREPARE rstmt FROM :stmt;
```

```
EXEC SQL DECLARE c2 CURSOR FOR rstmt;
EXEC SQL OPEN c2;
EXEC SQL FETCH c2 INTO :deptname;
EXEC SQL CLOSE c2;
...
// All static statements in the next two transactions
// run with intrapartition parallelism and degree 2:
EXEC SQL SELECT COUNT(*) INTO :numRecords FROM org;
...
EXEC SQL COMMIT;
```

动态 SQL 语句的分区内并行度是通过 CURRENT DEGREE 专用寄存器指定的；对于静态 SQL 语句，分区内并行度是通过 DEGREE 绑定选项指定的。下列命令用来预编译和绑定 demoapp 应用程序:

```
connect to prod;
prep demoapp.sqc bindfile;
bind demoapp.bnd degree 2;
...
```

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。

快速通信管理器

快速通信管理器（Windows）

在多成员环境中，每个成员具有一对 FCM 守护程序，它们用于支持与代理程序请求有关的成员之间的通信。一个守护程序用于发送通信，另一个用于接收通信。启动实例时会激活这些守护程序和支持基础结构。FCM 通信还用于在同一成员内部工作的代理程序；此类型通信也称为成员内部通信。

可以使用 **fcm_num_buffers** 数据库管理器配置参数来指定 FCM 消息缓冲区数。可以使用 **fcm_num_channels** 数据库管理器配置参数来指定 FCM 通道数。缺省情况下，**fcm_num_buffers** 和 **fcm_num_channels** 数据库管理器配置参数已设置为 AUTOMATIC。如果设置为 AUTOMATIC（这是建议的设置），那么 FCM 会监视资源使用情况并调整资源以满足工作负载需求。

快速通信管理器（Linux 和 UNIX）

快速通信管理器 (FCM) 为分区数据库环境提供通信支持。

在多成员环境中，每个成员具有一对 FCM 守护程序，它们用于支持与代理程序请求有关的成员之间的通信。一个守护程序用于发送通信，另一个用于接收通信。启动实例时会激活这些守护程序和支持基础结构。FCM 通信还用于在同一成员内部工作的代理程序；此类型通信也称为成员内部通信。

FCM 守护程序将收集有关通信活动的信息。可以通过使用数据库系统监视器来获得有关 FCM 通信的信息。如果成员之间的通信失败，或者它们重新建立通信，那么 FCM 守护程序会使用此信息来更新监视器元素。FCM 守护程序还会为此事件触发适当的操作。例如，回滚受到影响的事务。可以使用数据库系统监视器来帮助您设置 FCM 配置参数。

可以使用 **fcm_num_buffers** 数据库管理器配置参数来指定 FCM 消息缓冲区数。可以使用 **fcm_num_channels** 数据库管理器配置参数来指定 FCM 通道数。缺省情况下，**fcm_num_buffers** 和 **fcm_num_channels** 数据库管理器配置参数已设置为 AUTOMATIC。如果设置为 AUTOMATIC（这是建议的设置），那么 FCM 会监视资源使用情况并调整资源以满足工作负载需求。

使用 FCM 通信来启用数据库分区之间的通信

在分区数据库环境中，数据库分区之间的大多数通信都是由快速通信管理器 (FCM) 来处理。

要在一个数据库分区上启用 FCM 并允许与其他数据库分区通信，必须在该数据库分区的 etc 目录的 services 文件中创建一个服务条目，如本部分后面所示。FCM 将使用指定的端口来通信。如果已在同一主机上定义了多个数据库分区，那么必须定义一个端口范围，如本部分后面所示。

在尝试为快速通信管理器（FCM）手动配置内存之前，建议从 FCM 缓冲区数（**fcm_num_buffers**） 和 FCM 信道数 （**fcm_num_channels**）的自动设置开始（缺省设置）。使用 FCM 活动的系统监视器数据来判定此设置是否合适。

Windows 注意事项

TCP/IP 端口范围将由以下程序自动添加到服务文件:

- 安装程序，在创建实例或添加新的数据库分区时
- **db2icrt** 实用程序，在创建新实例时
- **db2ncrt** 实用程序，在计算机上添加第一个数据库分区时

服务条目的语法如下所示:

```
DB2_instance port/tcp #comment
```

DB2_*instance*

instance 的值是数据库管理器实例的名称。该名称的所有字符必须为小写。假定实例名为 DB2PUSER，那么应指定 DB2_db2puser。

port/**tcp**

要为该数据库分区保留的 TCP/IP 端口。

#*comment*

想要与该条目关联的任何注释。注释之前必须加 # 符号。

如果 etc 目录的 services 文件是共享的，那么必须确保在该文件中分配的端口的数目大于或等于该实例中多个数据库分区的最大数目。当分配端口时，还要确保考虑了可以用作备份的任何处理器。

如果 etc 目录的 services 文件不是共享的，那么注意事项基本相同，但它有一个附加注意事项: 必须确保为 DB2 数据库实例定义的条目在 etc 目录的所有 services 文件中都是相同的（而不适用于分区数据库环境的条目不必相同）。

如果在一个实例中的相同主机上有多个数据库分区，那么必须定义多个端口以供 FCM 使用。为此，在 etc 目录的 services 文件中包括两行，以指示正在分配的端口范围。第一行指定第一个端口，而第二行指示端口块的末尾。在下列示例中，为实例 SALES 分配了五个端口。这意味着该实例中不会有处理器具有多于五个的数据库分区。例如:

```
DB2_sales         9000/tcp
DB2_sales_END     9004/tcp
```

注: 只能用大写字母来指定 END。还必须确保包括了两个下划线 (_) 字符。

启用数据库分区服务器之间的通信（Linux 和 UNIX）

此任务描述了如何启用参与分区数据库系统的数据库分区服务器之间的通信。

数据库分区服务器之间的通信由“快速通信管理器”(FCM) 处理。要启用 FCM，必须在分区数据库系统中的每台计算机上的 /etc/services 文件中保留一个端口或端口范围。

开始之前

您的用户标识必须具有 root 用户权限。

您必须在参与实例的所有计算机上执行此任务。

关于此任务

为 FCM 保留的端口数目等于实例中由任何计算机主管或可能主管的数据库分区的最大数目。

在以下示例中，db2nodes.cfg 文件包含这些条目:

```
0 server1 0
1 server1 1
2 server2 0
3 server2 1
4 server2 2
5 server3 0
6 server3 1
7 server3 2
8 server3 3
```

假设 FCM 端口从 60000 开始编号。在此情况下:

- server1 对它的两个数据库分区使用两个端口（60000 和 60001）
- server2 对它的三个数据库分区使用三个端口（60000、60001 和 60002）
- server3 对它的四个数据库分区使用四个端口（60000、60001、60002 和 60003）

所有计算机均必须保留 60000、60001、60002 和 60003，因为这是实例中的任何计算机所需的最大端口范围。

如果使用诸如 Tivoli System Automation 或 IBM PowerHA® SystemMirror for AIX 之类的高可用性解决方案来将数据库分区从一台计算机故障转移到另一台计算机，那么必须考虑潜在的端口要求。例如，如果计算机通常主管四个数据库分区，但另一计算机的两个数据库分区有可能故障转移到该计算机，那么您必须为该计算机规划六个端口。

当创建实例时，将在主计算机上保留某个端口范围。主计算机也称为拥有实例的计算机。但是，如果最初添加到 /etc/services 文件的端口范围不够满足需要，那么需要手动添加其他条目来扩展保留端口的范围。

过程

要使用 /etc/services 在分区数据库环境中的服务器之间启用通信:

1. 作为具有 root 用户权限的用户登录主计算机（拥有实例的计算机）。
2. 创建实例。
3. 查看 /etc/services 文件中已保留的缺省端口范围。 除了基本配置外，FCM 端口应类似如下所示:

   ```
   db2c_db2inst1         50000/tcp
   #Add FCM port information
   DB2_db2inst1         60000/tcp
   DB2_db2inst1_1        60001/tcp
   DB2_db2inst1_2        60002/tcp
   DB2_db2inst1_END      60003/tcp
   ```

   缺省情况下，第一个端口（50000）保留给连接请求使用，而 60000 以上的前四个可用端口保留给 FCM 通信使用。一个端口用于拥有实例的数据库分区服务器，三个端口用于逻辑数据库分区服务器，可以在完成安装后选择将其添加至计算机。

端口范围必须包括起始条目和结束条目。中间条目为可选项。显式地包括中间值对于防止其他应用程序使用这些端口很有用，但这些条目未经数据库管理器验证。

DB2 端口条目的格式如下:

```
DB2_instance_name_suffix  port_number/tcp  # comment
```

其中:

- *instance_name* 是分区实例的名称。
- *suffix* 不用于第一个 FCM 端口。中间条目是介于最低和最高端口之间的端口。如果端口范围包括介于第一个 FCM 端口和最后一个 FCM 端口之间的中间条目，那么 *suffix* 包含一个整数，其他每个端口依据此整数递增。例如，第二个端口编号为 1，那么第三个端口编号为 2，以此类推以确保唯一性。END 一词必须用作最后一个条目的 *suffix*。
- *port_number* 是为数据库分区服务器通信保留的端口号。
- *comment* 是用于描述条目的可选注释。

4. 确保您保留了足够的端口供 FCM 通信使用。 如果保留端口范围不够用，那么将新的条目添加至该文件。
5. 作为 root 用户登录参与实例的每台计算机，并将相同的条目添加至 /etc/services 文件。

第 10 章 创建和管理分区数据库环境

管理数据库分区

可启动或停止分区、删除分区或跟踪分区。

开始之前

要使用数据库分区，需要连接至实例的权限。具有 SECADM 或 ACCESSCTRL 权限的任何人都可以授予您访问特定实例的权限。

过程

- 要启动或停止特定数据库分区，请使用带有 **DBPARTITIONNUM** 参数的 **START DATABASE MANAGER** 命令或 **STOP DATABASE MANAGER** 命令。
- 要从 `db2nodes.cfg` 配置文件中删除特定数据库分区，请使用带有 **DROP DBPARTITIONNUM** 参数的 **STOP DATABASE MANAGER** 命令。 在使用 **DROP DBPARTITIONNUM** 参数之前，请运行 **DROP DBPARTITIONNUM VERIFY** 命令以确保此数据库分区上没有用户数据。
- 要跟踪数据库分区上的活动，请使用 IBM 支持机构指定的选项。

  注意: 仅当 IBM 支持机构或技术支持代表要求时，才应使用跟踪实用程序。
  跟踪实用程序记录有关 DB2 for Linux, UNIX, and Windows 操作的信息并对其进行格式化。有关更多详细信息，请参阅“db2trc - 跟踪命令”主题。

在分区数据库环境中添加数据库分区

可在分区数据库系统运行或停止时，向其添加数据库分区。因为添加新服务器可能要花费较长时间，因此您可能希望在数据库管理器已经运行时来添加。

使用 **ADD DBPARTITIONNUM** 命令将数据库分区添加到系统。可以按下列方式调用此命令:

- 作为 **START DBM** 命令中的一个选项
- 与 **ADD DBPARTITIONNUM** 命令配合使用
- 与 sqleaddn API 配合使用
- 与 sqlepstart API 配合使用

如果系统已停止，那么请使用 **START DBM** 命令。如果系统在运行，可以使用任何其他选项。

当使用 **ADD DBPARTITIONNUM** 命令将一个新数据库分区添加至系统时，会将实例中的所有现有的数据库扩充到新数据库分区。还可指定要将临时表空间的哪些容器用于数据库。这些容器可能有下列特征:

- 与为每个数据库的目录分区定义的那些容器相同。（这是缺省情况。）
- 与为另一个数据库分区定义的那些容器相同。
- 根本没有创建。必须使用 ALTER TABLESPACE 语句来将临时表空间容器添加至每个数据库之后，才能使用数据库。

注: 当添加新数据库分区时，不会识别任何未编目的数据库。在新数据库分区上将不存在未编目的数据库。尝试与新数据库分区上的数据库进行连接会返回错误消息 SQL1013N。

在改变一个或多个数据库分区组以包括新数据库分区之前，不能使用新数据库分区上的数据库来包含数据。

不能通过对系统添加数据库分区来将单分区数据库更改为多分区数据库。这是因为跨数据库分区重新分发数据需要每个受影响的表上的分布键。分布键是在多分区数据库中创建表时自动生成的。在单分区数据库中，可使用 CREATE TABLE 或 ALTER TABLE SQL 语句显式地创建分布键。

注: 如果系统中没有定义任何数据库且您正在 UNIX 操作系统上运行企业服务器版，那么编辑 db2nodes.cfg 文件以添加新的数据库分区定义；不要使用所述的任何一个过程，因为它们只有在数据库存在时才适用。

Windows 注意事项: 如果在 Windows 操作系统上使用企业服务器版，并且实例中没有任何数据库，那么请使用 **db2ncrt** 命令来调整数据库系统。但是，如果已有数据库，那么请使用 **START DBM ADD DBPARTITIONNUM** 命令，以确保在调整系统时为每个现有数据库都创建一个数据库分区。在 Windows 操作系统上，请勿手动编辑数据库分区配置文件 (db2nodes.cfg)，因为这可以使文件产生不一致。

添加联机数据库分区

可在分区数据库环境运行以及应用程序连接至数据库时将联机的新数据库分区添加至此分区数据库环境。

过程

要使用命令行将联机数据库分区添加至正在运行的数据库管理器，请执行以下操作:

1. 在任何现有数据库分区上，运行 **START DBM** 命令。

   在所有平台上，为 **DBPARTITIONNUM**、**ADD DBPARTITIONNUM**、**HOSTNAME**、**PORT** 和 **NETNAME** 参数指定新数据库分区值。在 Windows 平台上，还应该指定 **COMPUTER**、**USER** 和 **PASSWORD** 参数。

   您还可为必须在该数据库中创建的任何“临时表空间容器定义”指定源。如果未提供表空间信息，那么从每个数据库的目录分区检索临时表空间容器的定义。

   例如，要将三个新数据库分区添加至现有数据库，请发出以下命令:

   ```
   START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME HOSTNAME3
    PORT PORT3;

   START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME HOSTNAME4
    PORT PORT4;

   START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME HOSTNAME5
    PORT PORT5;
   ```

2. 可选: 改变数据库分区组以合并新数据库分区。 将数据重新分发至新数据库分区时也可选择此操作。

3. 可选: 将数据重新分发至新数据库分区。 如果要利用新数据库分区，那么需要执行此操作。还可将改变数据库分区组选项作为重新分发操作的一部分来包括。否则，改变数据库分区组以合并新数据库分区必须先做为单独操作完成，才能将数据重新分发至新数据库分区。
4. 可选: 在新数据库分区上备份所有数据库。 虽然可选，但对于新数据库分区以及其他数据库分区，特别是在新旧数据库分区之间都已重新分发数据之后，使用此操作将很有用。

当以联机方式工作来添加数据库分区时的限制

在新数据库分区添加至实例之后，该数据库分区的状态取决于原始数据库分区的状态。如果应用程序使用 WITH HOLD 游标，那么在新数据库分区添加至实例之后，应用程序可能会或不会意识到该数据库分区的存在。

将新数据库分区添加至单一分区数据库实例时:

- 如果添加该数据库分区时原始数据库分区已启动，那么添加数据库分区操作完成时，新数据库分区会停止。
- 如果添加该数据库分区时原始数据库分区已停止，那么添加数据库分区操作完成时，新数据库分区会启动。

对于使用 WITH HOLD 游标的应用程序，如果在添加数据库分区操作运行之前它们已启动，那么添加数据库分区操作完成时，它们不会意识到新数据库分区的存在。如果在添加数据库分区操作运行之前 WITH HOLD 游标已关闭，那么添加数据库分区操作完成时，应用程序会意识到新数据库分区的存在

在脱机状态下添加数据库分区 (Windows)

当分区数据库系统停止时，可以将新数据库分区添加至其中。当再次启动数据库管理器时，新添加的数据库分区可用于所有数据库。

开始之前

- 必须先安装新服务器，然后才可以在该服务器上创建数据库分区。
- 将 **DB2_FORCE_OFFLINE_ADD_PARTITION** 注册表变量的缺省值设置为 TRUE，以强制任何已添加的数据库分区脱机。

过程

要使用命令行将数据库分区添加到已停止的分区数据库服务器:

1. 发出 **STOP DBM** 以停止所有数据库分区。
2. 在新服务器上运行 **ADD DBPARTITIONNUM** 命令。

   存在于系统中的每个数据库在本地创建一个数据库分区。将新数据库分区的数据库参数设置为缺省值，并且在将数据移动至其中之前，每个数据库分区保持为空。更新数据库配置参数值，以便与其他数据库分区上的值相匹配。
3. 运行 **START DBM** 命令以启动数据库系统。 注意，在安装新服务器期间，数据库分区配置文件已由数据库管理器更新为包括新服务器。
4. 按如下所示更新新数据库分区上的配置文件:
   a. 在任何现有数据库分区上，运行 **START DBM** 命令。

为 **DBPARTITIONNUM**、**ADD DBPARTITIONNUM**、**HOSTNAME**、**PORT** 和 **NETNAME** 参数以及 **COMPUTER**、**USER** 和 **PASSWORD** 参数指定新数据库分区值。

您还可为需要在该数据库中创建的任何“临时表空间容器定义”指定源。如果未提供表空间信息，那么从每个数据库的目录分区检索临时表空间容器的定义。

例如，要将三个新数据库分区添加至现有数据库，请发出以下命令:

```
START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME HOSTNAME3
 PORT PORT3;
START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME HOSTNAME4
 PORT PORT4;
START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME HOSTNAME5
 PORT PORT5;
```

完成 **START DBM** 命令后，将停止新服务器。

b. 通过运行 **STOP DBM** 命令来停止数据库管理器。

停止系统中的所有数据库分区 时，会更新 节点配置文件 以包括新数据库分区。直到执行了 **STOP DBM** 之后，才会使用新服务器信息更新节点配置文件。这确保 **ADD DBPARTITIONNUM** 命令（对 **START DBM** 命令指定 **ADD DBPARTITIONNUM** 参数时，将调用此命令）在正确的数据库分区上运行。实用程序结束时，新服务器分区会停止。

5. 通过运行 **START DBM** 命令来启动数据库管理器。

现在，新添加的数据库分区与系统的其余部分一起启动。

当系统中的所有数据库分区正运行时，可执行系统范围内的活动，如创建或删除数据库。

注: 可能必须对所有数据库分区服务器发出 **START DBM** 命令两次，才能访问新的 db2nodes.cfg 文件。

6. 可选: 改变数据库分区组以合并新数据库分区。 将数据重新分发至新数据库分区时也可选择此操作。
7. 可选: 将数据重新分发至新数据库分区。 如果要利用新数据库分区，那么不能真正地选择此操作。还可将改变数据库分区组选项作为重新分发操作的一部分来包括。否则，在将数据重新分发至新数据库分区之前，必须将改变数据库分区组以合并新数据库分区作为单独的操作来完成。
8. 可选: 在新数据库分区上备份所有数据库。 虽然可选，但对于新数据库分区以及其他数据库分区，特别是在新旧数据库分区之间都已重新分发数据之后，使用此操作将很有用。

添加脱机数据库分区（Linux 和 UNIX）

可将脱机的新数据库分区添加至分区数据库系统。当再次启动数据库管理器时，新添加的数据库分区可用于所有数据库。

开始之前

- 如果新服务器还不存在，那么应先安装服务器，才能在此服务器上创建数据库分区。
- 使用共享文件系统安装或本地副本来使可执行文件可供访问。

- 使操作系统文件与现有处理器上的操作系统文件同步。
- 确保 sqllib 目录可作为共享文件系统访问。
- 确保相关操作系统参数（例如，最大进程数）设置为适当的值。
- 向名称服务器注册主机名或在所有数据库分区上的 /etc 目录的 hosts 文件中注册主机名。必须在 .rhosts 中注册计算机的主机名，才能使用 **rsh** 或 **rah** 来运行远程命令。
- 将 **DB2_FORCE_OFFLINE_ADD_PARTITION** 注册表变量的缺省值设置为 TRUE，以强制已添加的数据库分区脱机。

过程

- 要使用命令行将数据库分区添加到已停止的分区数据库服务器:
  1. 发出 **STOP DBM** 以停止所有数据库分区。
  2. 在新服务器上运行 **ADD DBPARTITIONNUM** 命令。

     存在于系统中的每个数据库在本地创建一个数据库分区。将新数据库分区的数据库参数设置为缺省值，并且在将数据移动至其中之前，每个数据库分区保持为空。更新数据库配置参数值，以便与其他数据库分区上的值相匹配。
  3. 运行 **START DBM** 命令以启动数据库系统。 注意，新服务器安装期间，数据库分区配置文件 (db2nodes.cfg) 已由数据库管理器更新为包括新服务器。
  4. 按如下所示更新新数据库分区上的配置文件:
     a. 在任何现有数据库分区上，运行 **START DBM** 命令。

        为 **DBPARTITIONNUM**、**ADD DBPARTITIONNUM**、**HOSTNAME**、**PORT** 和 **NETNAME** 参数以及 **COMPUTER**、**USER** 和 **PASSWORD** 参数指定新数据库分区值。

        您还可为必须在该数据库中创建的任何“临时表空间容器定义”指定源。如果未提供表空间信息，那么从每个数据库的目录分区检索临时表空间容器的定义。

        例如，要将三个新数据库分区添加至现有数据库，请发出以下命令:

        ```
        START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME HOSTNAME3
         PORT PORT3;

        START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME HOSTNAME4
         PORT PORT4;

        START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME HOSTNAME5
         PORT PORT5;
        ```

        完成 **START DBM** 命令后，将停止新服务器。
     b. 通过运行 **STOP DBM** 命令来停止整个数据库管理器。

        停止系统中的所有数据库分区 时，会更新 节点配置文件 以包括新数据库分区。直到执行了 **STOP DBM** 之后，才会使用新服务器信息更新节点配置文件。这确保 **ADD DBPARTITIONNUM** 命令（对 **START DBM** 命令指定 **ADD DBPARTITIONNUM** 参数时，将调用此命令）在正确的数据库分区上运行。实用程序结束时，新服务器分区会停止。
  5. 通过运行 **START DBM** 命令来启动数据库管理器。

     现在，新添加的数据库分区与系统的其余部分一起启动。

当系统中的所有数据库分区正运行时，可执行系统范围内的活动，如创建或删除数据库。

注: 可能必须对所有数据库分区服务器发出 **START DBM** 命令两次，才能访问新的 db2nodes.cfg 文件。

6. 可选: 改变数据库分区组以合并新数据库分区。 将数据重新分发至新数据库分区时也可选择此操作。
7. 可选: 将数据重新分发至新数据库分区。 如果要利用新数据库分区，那么不能真正地选择此操作。还可将改变数据库分区组选项作为重新分发操作的一部分来包括。否则，在将数据重新分发至新数据库分区之前，必须将改变数据库分区组以合并新数据库分区作为单独的操作来完成。
8. 可选: 在新数据库分区上备份所有数据库。 虽然可选，但对于新数据库分区以及其他数据库分区，特别是在新旧数据库分区之间都已重新分发数据之后，使用此操作将很有用。

- 也可以手动更新配置文件，如下所示:
  1. 编辑 db2nodes.cfg 文件，并将新数据库分区添加至该文件。
  2. 发出以下命令来启动新数据库分区: START DBM DBPARTITIONNUM *partitionnum*

     将您分配给新数据库分区的编号指定为 *partitionnum* 的值。
  3. 如果要使新服务器成为逻辑分区（即，它不是数据库分区 0），请使用 **db2set** 命令来更新 **DBPARTITIONNUM** 注册表变量。 指定要添加的数据库分区的号码。
  4. 在新数据库分区上运行 **ADD DBPARTITIONNUM** 命令。

     此命令还为已存在于系统中的每个数据库在本地创建一个数据库分区。将新数据库分区的数据库参数设置为缺省值，并且在将数据移动至其中之前，每个数据库分区保持为空。更新数据库配置参数值，以便与其他数据库分区上的值相匹配。
  5. 当 **ADD DBPARTITIONNUM** 命令完成时，请发出 **START DBM** 命令来启动系统中的其他数据库分区。

     在成功启动所有数据库分区之前，不要执行任何系统范围内的活动，如创建或删除数据库。

添加数据库分区时的错误恢复

因为数据库管理器会创建系统缓冲池以对所有缓冲池页大小提供缺省自动支持，所以添加数据库分区不会因不存在的缓冲池而失败。

但是，因为这些系统缓冲池非常小，所以如果使用其中一个系统缓冲池，那么可能会严重影响性能。如果使用系统缓冲池，管理通知记录中会写入一条消息。在下列情况的数据库分区添加方案中使用系统缓冲池:

- 将数据库分区添加到分区数据库环境中，该数据库具与缺省值 4KB 不同的页大小的一个或多个系统临时表空间。当创建数据库分区时，仅 IBMDEFAULTDP 缓冲池存在，此缓冲池页大小为 4 KB。

  请考虑以下示例:

  1. 使用 **START DBM** 命令来将数据库分区添加到当前的多分区数据库:

     ```
     START DBM DBPARTITIONNUM 2 ADD DBPARTITIONNUM HOSTNAME newhost PORT 2
     ```

2. 在使用新的数据库分区描述手动更新 db2nodes.cfg 文件之后，使用 **ADD DBPARTITIONNUM** 命令。

防止这些问题发生的一种方法是在 **ADD DBPARTITIONNUM** 或 **START DBM** 命令上指定 WITHOUT TABLESPACES 子句。此后，请使用 CREATE BUFFERPOOL 语句来创建使用适当 SIZE 和 PAGESIZE 值的缓冲池，并使用 ALTER TABLESPACE 语句将系统临时表空间与缓冲池相关联。

- 将数据库分区添加到现有的数据库分区组，该数据库分区组具与缺省页大小 4KB 不同的页大小的一个或多个表空间。发生此情况的原因是尚未对表空间激活在新数据库分区上创建的非缺省页大小缓冲池。

  注: 在先前版本中，此命令使用了 NODEGROUP 关键字，而不是 DATABASE PARTITION GROUP 关键字。

  请考虑以下示例:

  – 可以使用 ALTER DATABASE PARTITION GROUP 语句来向数据库分区组添加数据库分区，如下所示:

  ```
  START DBM
     CONNECT TO mpp1
  ALTER DATABASE PARTITION GROUP ng1 ADD DBPARTITIONNUM (2)
  ```

  防止此问题发生的一种方法是在发出 ALTER DATABASE PARTITION GROUP 语句之前，为每个页大小创建缓冲池，然后重新连接至数据库:

  ```
  START DBM
     CONNECT TO mpp1
     CREATE BUFFERPOOL bp1 SIZE 1000 PAGESIZE 8192
  CONNECT RESET
     CONNECT TO mpp1
  ALTER DATABASE PARTITION GROUP ng1 ADD DBPARTITIONNUM (2)
  ```

注: 如果数据库分区组具有页大小为缺省值的表空间，那么返回消息 SQL1759W。

删除数据库分区

您可以删除任何数据库都未在使用的数据库分区，并为其他的用户空出计算机。

开始之前

通过发出 **DROP DBPARTITIONNUM VERIFY** 命令或 sqledrpn API，验证是否未使用此数据库分区。

- 如果接收到消息 SQL6034W（未在任何数据库中使用数据库分区），那么可以删除该数据库分区。
- 如果接收到消息 SQL6035W（数据库正在使用数据库分区），那么使用 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令，以将要删除的数据库分区中的数据重新分发到数据库别名不同的其他数据库分区。

还应确保所有事务（此数据库分区是其协调程序）已成功落实或回滚。这可能需要在其他服务器上执行崩溃恢复。例如，如果您删除协调程序分区，而参与事务的另一个数据库分区在删除协调程序分区之前崩溃，那么该崩溃的数据库分区将无法查询协调程序分区来获取任何不确定事务的输出。

过程

要使用命令行来删除数据库分区:

发出带 **DROP DBPARTITIONNUM** 参数的 **STOP DBM** 命令来删除该数据库分区。
在成功完成命令后，停止系统。然后，使用 **START DBM** 命令来启动数据库管理器。

在实例中列示数据库分区服务器 (Windows)

在 Windows 上，使用 **db2nlist** 命令来获取参与实例的数据库分区服务器的列表。

关于此任务

按如下所示使用该命令:

```
db2nlist
```

当按以上所示使用此命令时，缺省实例是当前实例（由 **DB2INSTANCE** 环境变量设置）。
要指定特定的实例，可使用以下命令指定该实例:

```
db2nlist /i:instName
```

其中 *instName* 是想要的特定实例名。

（可选）也可使用以下命令请求每个数据库分区服务器的状态:

```
db2nlist /s
```

每个数据库分区服务器的状态可能是: 正在启动、正在运行、正在停止或已停止。

将数据库分区服务器添加至实例（Windows）

在 Windows 上，使用 **db2ncrt** 命令来将数据库分区服务器添加至实例。

关于此任务

注: 如果此实例已包含数据库，那么不要使用 **db2ncrt** 命令。请改为使用 **START DBM ADD DBPARTITIONNUM** 命令。这确保可正确地将该数据库添加至新的数据库分区服务器。**不要编辑** db2nodes.cfg 文件，因为更改文件可能导致分区数据库环境中的不一致性。

该命令具有下列必需参数:

```
db2ncrt /n:partition_number
        /u:username,password
        /p:logical_port
```

/n:partition_number
用于标识数据库分区服务器的唯一数据库分区号。该号码可以是按递升顺序排列的 1 到 999 中的任何一个值。

/u:username,password
DB2 服务的登录帐户名和密码。

/p:logical_port
用于数据库分区服务器的逻辑端口号（若逻辑端口不是零（0））。如果不指定，那么将逻辑端口号指定为 0。

仅当在计算机上创建第一个数据库分区时，逻辑端口参数才是可选的。如果创建逻辑数据库分区，那么必须指定此参数并选择一个没有在使用的逻辑端口号。有几项限制:

- 在每台计算机上，都必须要有一个逻辑端口为 0 的数据库分区服务器。
- 端口号不能超过 %SystemRoot%\system32\drivers\etc 目录中的 services 文件中为 FCM 通信保留的端口范围。例如，如果为当前实例保留四个端口，那么最大端口号将是 3（端口 1、2 和 3；端口 0 用于缺省逻辑数据库分区）。端口范围是在将 **db2icrt** 命令与 /r:base_port, end_port 参数配合使用时定义的。

还有几个可选参数:

/g:network_name
指定数据库分区服务器的网络名。如果不指定此参数，那么 DB2 使用它在系统上检测到的第一个 IP 地址。

如果计算机上有多个 IP 地址，且您想要对数据库分区服务器指定特定 IP 地址，那么使用此参数。可输入使用网络名或 IP 地址的 network_name 参数。

/h:host_name
FCM 用于内部通信的 TCP/IP 主机名（如果该主机名不是本地主机名）。如果在远程计算机上添加数据库分区服务器，那么此参数是必需的。

/i:instance_name
实例名；缺省值为当前实例。

/m:computer_name
该数据库分区所在的 Windows 工作站的计算机名称；缺省名称是本地计算机的计算机名称。

/o:instance_owning_computer
作为拥有实例的计算机的计算机名称；缺省值是本地计算机的计算机名称。当在任何非拥有实例的计算机上调用 **db2ncrt** 命令时，此参数是必需的。

例如，如果要向拥有实例的计算机 MYMACHIN 上的实例 TESTMPP 添加新的数据库分区服务器（以便运行多逻辑数据库分区），且您想要让这个新数据库分区成为使用逻辑端口 1 的数据库分区 2，那么输入:

```
db2ncrt /n:2 /p:1 /u:my_id,my_pword /i:TESTMPP
   /M:TEST /o:MYMACHIN
```

更改数据库分区（Windows）

在 Windows 上，可使用 **db2nchg** 命令来更改数据库分区。

关于此任务

- 将数据库分区从一台计算机移至另一台计算机。
- 更改计算机的 TCP/IP 主机名。

  如果打算使用多个网络适配器，那么必须使用此命令为 db2nodes.cfg 文件中的“netname”字段指定 TCP/IP 地址。
- 使用另一逻辑端口号。
- 对数据库分区服务器使用另一名称。

该命令具有以下必需参数:

```
db2nchg /n:node_number
```

参数 **/n:** 是要更改的数据库分区服务器的编号。此参数是必需的。

可选参数包括:

/i:*instance_name*
指定此数据库分区服务器所参与的实例。如果未指定此参数，那么缺省值是当前实例。

/u:*username*,*password*
更改 DB2 数据库服务的登录帐户名和密码。如果未指定此参数，那么登录帐户和密码保持不变。

/p:*logical_port*
更改数据库分区服务器的逻辑端口。如果将数据库分区服务器移至另一台计算机，那么必须指定此参数。如果未指定此参数，那么逻辑端口号保持不变。

/h:*host_name*
更改由 FCM 用于内部通信的 TCP/IP 主机名。如果未指定此参数，那么主机名不更改。

/m:*computer_name*
将数据库分区服务器移至另一台计算机。仅当实例中没有现有数据库时，才可移动数据库分区服务器。

/g:*network_name*
更改数据库分区服务器的网络名。

如果计算机上有多个 IP 地址，且您想要对数据库分区服务器使用特定 IP 地址，那么使用此参数。可使用网络名或 IP 地址来输入 *network_name*。

例如，要将指定给数据库分区 2（它参与实例 TESTMPP）的逻辑端口更改为使用逻辑端口 3，那么输入以下命令:

```
db2nchg /n:2 /i:TESTMPP /p:3
```

DB2 数据库管理器提供访问远程计算机上实例级别的 DB2 数据库系统注册表变量的能力。当前，以三种不同的级别存储 DB2 数据库系统注册表变量：计算机或全局级、实例级和数据库分区级。对于以实例级（包括数据库分区级）存储的注册表变量，可使用 **DB2REMOTEPREG** 将其重定向到其他计算机。如果设置了 **DB2REMOTEPREG**，那么 DB2 数据库管理器会从 **DB2REMOTEPREG** 所指向的计算机访问 DB2 数据库系统注册表变量。**db2set** 命令会显示为:

```
db2set DB2REMOTEPREG=remote_workstation
```

其中 *remote_workstation* 是远程工作站名称。

注:

- 必须小心设置此选项，因为所有 DB2 数据库实例概要文件和实例列表都将位于指定的远程计算机名称上。
- 如果您的环境包括域中的用户，那么确保与 DB2 实例服务相关联的登录帐户是域帐户。这样可以确保该 DB2 实例具有适当的特权来枚举域级别的组。

此功能可与设置 **DBINSTPROF** 结合起来使用，以指向包含该注册表的同一台计算机上的远程 LAN 驱动器。

向数据库分区上的 SMS 表空间添加容器

只能向数据库分区上当前没有任何容器的 SMS 表空间添加容器。

过程

要使用命令行来向 SMS 表空间添加容器，请输入以下内容:

```
ALTER TABLESPACE name
        ADD ('path')
   ON DBPARTITIONNUM (database_partition_number)
```

按编号指定的数据库分区以及数据库分区范围内的每个分区都必须存在于定义表空间的数据库分区组中。*database_partition_number* 可能只能显式地出现或者只出现在语句的某个 *db-partitions* 子句的范围内。

示例

以下示例显示在 UNIX 操作系统上，如何将对表空间“plans”所使用的数据库分区组的 3 号数据库分区添加新容器:

```
ALTER TABLESPACE plans
      ADD ('/dev/rhdisk0')
      ON DBPARTITIONNUM (3)
```

从实例中删除数据库分区（Windows）

在 Windows 上，使用 **db2ndrop** 命令从没有数据库的实例中删除数据库分区服务器。如果删除一个数据库分区服务器，那么它的数据库分区号可以再次用于新的数据库分区服务器。

关于此任务

当从实例中删除数据库分区服务器时务必小心。如果从该实例中删除拥有实例的数据库分区服务器 0，那么该实例将变得不可用。如果要删除该实例，请使用 **db2idrop** 命令。

注: 如果此实例包含数据库，那么不要使用 **db2ndrop** 命令。请改为使用 **STOP DBM DROP DBPARTITIONNUM** 命令。这确保可正确地从数据库分区中除去该数据库。**不要编辑** db2nodes.cfg 文件，因为更改文件可能导致分区数据库环境中的不一致性。

如果要从运行多个逻辑数据库分区的计算机中删除分配了逻辑端口 0 的数据库分区，那么在删除分配了逻辑端口 0 的数据库分区之前，必须删除分配给其他逻辑端口的所有其他数据库分区。每个数据库分区服务器都必须带有一个分配给逻辑端口 0 的数据库分区。

该命令具有下列参数:

```
db2ndrop /n:dbpartitionnum /i:instance_name
```

/n:*dbpartitionnum*

用于标识数据库分区服务器的唯一数据库分区号（*dbpartitionnum*）。这是一个必需参数。该号码可以是按递升顺序排列的零 (0) 到 999 中的任何一个值。请记住数据库分区零 (0) 表示拥有实例的计算机。

/i:*instance_name*

实例名称（*instance_name*）。这是一个可选参数。若不给出，则缺省值是当前实例（由 **DB2INSTANCE** 注册表变量设置）。

方案: 在新数据库分区中重新分发数据

此方案说明如何将新的数据库分区添加至数据库并在数据库分区之间重新分发数据。**REDISTRIBUTE DATABASE PARTITION GROUP** 命令是在显示如何在数据库分区组中的不同表集上重新分发数据时演示的。

关于此任务

方案: 数据库 DBPG1 有两个指定为 (0, 1) 的数据库分区和一个数据库分区组定义 (0, 1)。

在数据库分区组 DBPG_1 上定义了下列表空间:

- 表空间 TS1 - 此表空间有两个表 T1 和 T2
- 表空间 TS2 - 此表空间定义了三个表 T3、T4 和 T5

从 V9.7 开始，您可以在数据库运行过程中以及将应用程序连接至数据库时添加数据库分区。然而，通过将 **DB2_FORCE_OFFLINE_ADD_PARTITION** 注册表变量的缺省值更改为 TRUE，可以在此情况下脱机执行此操作。

过程

要在 DBPG1 中的数据库分区之间重新分发数据:

1. 确定在重新分发之前必须禁用或除去的对象。
   a. 复制 MQT: 重新分发操作不支持此类型的 MQT。必须在运行重新分发之前将它们删除，并在之后重新创建。

      ```
      SELECT tabschema, tabname
        FROM syscat.tables
        WHERE partition_mode = 'R'
      ```

   b. “写入表”事件监视器: 如果任何自动激活的“写入表”事件监视器具有包含在要重新分发的数据库分区组中的表，请禁用这些事件监视器。

      ```
      SELECT distinct evmonname
        FROM syscat.eventtables E
        JOIN syscat.tables T on T.tabname = E.tabname
          AND T.tabschema = E.tabschema
        JOIN syscat.tablespaces S on S.tbspace = T.tbspace
          AND S.ngname = 'DBPG_1'
      ```

   c. 说明表: 建议在单一分区数据库分区组中创建说明表。然而，如果在需要重新分发的数据库分区组中定义了说明表，并且目前为止所生成的数据不需要进行维护，请考虑删除这些说明表。一旦完成了重新分发，就可重新定义这些说明表。
   d. 表访问方式和状态: 确保要重新分发的数据库分区组中所有表都处于完全访问方式并且处于正常的表状态。

      ```
      SELECT DISTINCT TRIM(T.OWNER) || \'.\' || TRIM(T.TABNAME)
        AS NAME, T.ACCESS_MODE, A.LOAD_STATUS
       FROM SYSCAT.TABLES T, SYSCAT.DBPARTITIONGROUPS
        N, SYSIBMADM.ADMINTABINFO A
       WHERE T.PMAP_ID = N.PMAP_ID
        AND A.TABSCHEMA = T.OWNER
      ```

```
  AND A.TABNAME = T.TABNAME
  AND N.DBPGNAME = 'DBPG_1'
  AND (T.ACCESS_MODE <> 'F' OR A.LOAD_STATUS IS NOT NULL)
```

e. 统计信息概要文件：如果为表定义了统计信息概要文件，那么作为重新分发过程的一部分，可以对表统计信息进行更新。使用 REDISTRIBUTE 实用程序来更新表的统计信息时，将减少 I/O，这是因为在执行重新分发时将扫描所有数据，而在执行 **RUNSTATS** 时，将不需要再对数据进行扫描。

```
RUNSTATS on table schema.table
  USE PROFILE runstats_profile
  SET PROFILE ONLY
```

2. 查看数据库配置。 **util_heap_sz** 对于数据库分区之间的数据移动处理很关键 - 在重新分发的持续时间内，为 **util_heap_sz** 尽可能多地分配内存。如果在重新分发期间重建了索引，那么需要足够的 **sortheap**。按需要增大 **util_heap_sz** 和 **sortheap**，以提高重新分发性能。

3. 检索要用于新数据库分区的数据库配置设置。 当添加数据库分区时，会使用缺省数据库配置。因此，切记在发出 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令之前对新数据库分区的数据库配置进行更新。事件的此顺序将确保配置处于平衡状态。

```
SELECT name,
  CASE WHEN deferred_value_flags = 'AUTOMATIC'
    THEN deferred_value_flags
    ELSE substr(deferred_value,1,20)
      END  AS deferred_value
  FROM sysibmadm.dbcfg
  WHERE dbpartitionnum = existing-node
    AND deferred_value != ''
    AND name NOT IN ('hadr_local_host','hadr_local_svc','hadr_peer_window',
                'hadr_remote_host','hadr_remote_inst','hadr_remote_svc',
                'hadr_syncmode','hadr_timeout','backup_pending','codepage',
                'codeset','collate_info','country','database_consistent',
                'database_level','hadr_db_role','log_retain_status',
                'loghead','logpath','multipage_alloc','numsegs','pagesize',
                'release','restore_pending','restrict_access',
                'rollfwd_pending','territory','user_exit_status',
                'number_compat','varchar2_compat','database_memory')
```

4. 在启动重新分发过程之前对数据库（或相关数据库分区组中的表空间）进行备份。 此操作将确保恢复点是最新的。

5. 将三个新数据库分区添加到数据库。 发出下列命令:

```
START DBM DBPARTITIONNUM 3 ADD DBPARTITIONNUM HOSTNAME HOSTNAME3
 PORT PORT3 WITHOUT TABLESPACES;

START DBM DBPARTITIONNUM 4 ADD DBPARTITIONNUM HOSTNAME HOSTNAME4
 PORT PORT4 WITHOUT TABLESPACES;

START DBM DBPARTITIONNUM 5 ADD DBPARTITIONNUM HOSTNAME HOSTNAME5
 PORT PORT5 WITHOUT TABLESPACES;
```

如果 **DB2_FORCE_OFFLINE_ADD_PARTITION** 已设置为 TRUE，那么在关闭并重新启动实例之后，新数据库分区才对该实例可视。例如:

```
STOP DBM;START DBM;
```

6. 在新定义的数据库分区上定义系统临时表空间容器。

```
ALTER TABLESPACE tablespace_name
  ADD container_information
  ON dbpartitionnums (3 to 5)
```

7. 将新数据库分区添加到数据库分区组。 以下命令将 DBPG_1 定义从 (0, 1) 更改为 (0, 1, 3, 4, 5):

```
ALTER DATABASE PARTITION GROUP DBPG_1
  ADD dbpartitionnums (3 to 5)
  WITHOUT TABLESPACES
```

8. 在新定义的数据库分区上定义永久数据表空间容器。

```
ALTER TABLESPACE tablespace_name
  ADD container_information
  ON dbpartitionnums (3 to 5)
```

9. 将数据库配置设置应用于新数据库分区（或对所有数据库分区发出单个 **UPDATE DB CFG** 命令）。
10. 捕获要重新分发的数据库分区组中存在的任何复制型 MQT 的定义，然后将这些 MQT 删除。

```
db2look -d DBPG1 -e -z
  schema -t replicated_MQT_table_names
  -o repMQTs.clp
```

11. 禁用要重新分发的数据库分区组中存在的任何“写入表”事件监视器。

```
SET EVENT MONITOR monitor_name STATE 0
```

12. 运行 REDISTRIBUTE 实用程序以在所有数据库分区之间均匀地重新分发。

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 UNIFORM STOP AT 2006-03-10-07.00.00.000000;
```

假定该命令对表 T1、T2 和 T3 成功运行，然后由于指定了 **STOP AT** 选项而停止。

要异中止数据库分区组的数据重新分发并还原对表 T1、T2 和 T3 所作的更改，请发出以下命令:

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1
NOT ROLLFORWARD RECOVERABLE ABORT;
```

在发生错误或出现中断，并且您不希望继续执行重新分发操作时，您可以中止数据重新分发。对于此方案，假定此命令运行成功并且表 T1 和 T2 已还原为它们的原始状态。

要在只有 5000 个 4K 页面作为 **DATA BUFFER** 的情况下重新分发 T5 和 T4:

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 UNIFORM TABLE (T5, T4) ONLY DATA BUFFER 5000;
```

如果该命令运行成功，那么已成功地重新分发表 T4 和 T5 中的数据。

要按指定顺序完成表 T1、T2 和 T3 中数据的重新分发，请发出:

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1 NOT ROLLFORWARD RECOVERABLE
 CONTINUE TABLE (T1) FIRST;
```

指定 **TABLE (T1) FIRST** 时，将强制数据库管理器首先处理表 T1，以便该表可以在其他表之前返回到联机（只读）状态。所有其他表按数据库管理器确定的顺序处理。

注:

- 作为在步骤 7 和 8 中执行 ALTER DATABASE PARTITION GROUP 和 ALTER TABLESPACE 语句的替代方法，可以在 **REDISTRIBUTE DATABASE PARTITION**

GROUP 命令中指定 **ADD DBPARTITIONNUM** 参数。使用此命令参数添加数据库分区时，表空间的容器将基于数据库分区组中编号最小的现有分区上相应表空间的容器。

- 此示例中的 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令将不可前滚恢复。
- 完成 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令之后，它所访问的所有表空间都将保持 BACKUP PENDING 状态。必须先备份这类表空间，它们包含的表才可供写操作访问。

有关更多信息，请参阅“REDISTRIBUTE DATABASE PARTITION GROUP 命令”。

还应该考虑将表列表指定为 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令的输入，以强制实施处理这些表的顺序。REDISTRIBUTE 实用程序将移动压缩数据。（可选）将重建索引和更新统计信息（如果定义了统计信息概要文件）。因此，不运行前面的命令，可改为运行以下脚本:

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1
  NOT ROLLFORWARD RECOVERABLE uniform
  TABLE (t1, t2,...) FIRST;
```

在分区数据库环境中发出命令

在分区数据库环境中，您可能想要发出将在实例中的计算机或在多个数据库分区服务器上运行的命令。为此，您可以使用 **rah** 命令或 **db2_all** 命令。**rah** 命令允许您发出将在实例中的计算机上运行的命令。

如果想要命令在实例中的多个数据库分区服务器上运行，那么运行 **db2_all** 命令。本章节概述了这些命令。以下信息仅适用于分区数据库环境。

在 Windows 上，要运行 **rah** 命令或 **db2_all** 命令，您必须使用 Administrators 组成员的用户帐户来登录。

在 Linux 和 UNIX 操作系统上，您的登录 shell 程序可以是 Korn shell 程序或任何其他的 shell；但是，不同 shell 处理包含特殊字符的命令所用的方式不同。

另外，在 Linux 和 UNIX 操作系统上，**rah** 使用 **DB2RSHCMD** 注册表变量指定的远程 shell 程序。可以在两个远程 shell 程序之间选择: ssh（用于更高的安全性要求）或 rsh（对于 HP-UX，则为 remsh）。如果没有设置 **DB2RSHCMD**，那么会使用 rsh（对于 HP-UX，则为 remsh）。ssh 远程 shell 程序用来防止在 UNIX 操作系统环境中以明文形式传输密码。

如果命令在一个数据库分区服务器上运行，而您想让该命令在它们所有上面运行，请使用 **db2_all**。**db2trc** 命令例外，该命令在计算机的所有逻辑数据库分区服务器上运行。如果要在所有计算机的所有逻辑数据库分区服务器上运行 **db2trc**，请使用 **rah**。

注: **db2_all** 命令不支持需要交互式用户输入的命令。

rah 和 db2_all 命令概述

可以依次在各个数据库分区服务器上按顺序运行命令，也可以用并行的方式运行命令。

在 Linux 和 UNIX 操作系统上，如果以并行方式运行这些命令，那么可以选择将输出发送至缓冲区并收集该输出以便显示（缺省行为），或者可在发出该命令的计算机上显示该输出。在 Windows 上，如果以并行方式运行这些命令，那么在发出该命令的计算机上显示输出。

要使用 **rah** 命令，请输入:

```
rah command
```

要使用 **db2_all** 命令，请输入:

```
db2_all command
```

要获取关于 **rah** 语法的帮助，请输入:

```
rah "?"
```

该命令几乎可以是在交互式提示符下输入的任何内容，例如，要按顺序运行的多个命令。在 Linux 和 UNIX 操作系统上，使用分号 (;) 将多个命令分开。在 Windows 上，使用 & 符号将多个命令分隔开。不要在最后一个命令之后使用分隔符。

以下示例显示如何使用 **db2_all** 命令来更改在数据库分区配置文件中指定的所有数据库分区上的数据库配置。因为 ; 字符位于双引号内，所以请求将同时运行。

```
db2_all ";DB2 UPDATE DB CFG FOR sample USING LOGFILSIZ 100"
```

注: **db2_all** 命令不支持需要交互式用户输入的命令。

指定 rah 和 db2_all 命令

可以在命令行中将 **rah** 命令指定为参数，或者在未指定任何参数时，为响应提示而指定该命令。

如果命令包含下列特殊字符，那么请使用提示方法:

```
| & ; < > ( ) { } [ ] unsubstituted $
```

若在命令行上指定命令作为参数，而且若它包含刚才列示的任何特殊字符，必须以双引号将命令括起来。

注: 在 Linux 和 UNIX 操作系统上，将该命令添加至命令历史记录中，就像您在提示符处输入它一样。

可以正常输入命令中的所有特殊字符（除 \ 外，不必以双引号括起来）。如果需要在命令中使用一个 (\)，那么必须输入两个反斜杠 (\\)。

注: 在 Linux 和 UNIX 操作系统上，如果未使用 Korn shell 程序，那么可以正常输入命令中的所有特殊字符（除 ″、 \ 、不可替换的 $ 和单引号 (') 外，其他字符都不需要用引号引起来）。如果需要在命令中使用这些字符之一，那么必须在字符前加三个反斜杠 (\\\)。例如，如果需要在命令中使用一个 (\)，那么必须输入四个反斜杠 (\\\\)。

如果需要在命令中使用双引号 (″)，那么必须在双引号前加三个反斜杠，例如，(\\\″)。

注:

1. 在 Linux 和 UNIX 操作系统上，除非命令 shell 提供了在被括上单引号的字符串中输入单引号的某种方法，否则您不能在命令中包括单引号 (')。

2. 在 Windows 上，除非命令窗口提供了在被括上单引号的字符串中输入单引号的某种方法，否则您不能在命令中包括单引号 (')。

当运行任何 korn-shell 程序的 shell 脚本，而该脚本包含从后台中的标准输入进行读取的逻辑时，请明确地将标准输入重定向到一个源，在此源上进程可以读取而不必在终端上停止（SIGTTIN 消息）。要重定向标准输入，可以用下列格式运行脚本:

```
shell_script </dev/null &
```

若没有提供输入。

同样，在后台运行 **db2_all** 时，请始终指定 </dev/null。例如:

```
db2_all ";run_this_command" </dev/null &
```

通过执行此操作，可以重定向标准输入，并避免在终端上停止。

当您不关心来自远程命令的输出时，此方法的一种替代方法是在 **db2_all** 前缀中使用“daemonize”选项:

```
db2_all ";daemonize_this_command" &
```

以并行方式运行命令（Linux 和 UNIX）

缺省情况下，命令在每台计算机上顺序运行，但通过在命令前加上某些前缀序列，可以指定使用后台 rshell 以并行方式运行命令。如果 rshell 在后台运行，那么每个命令将输出放置在其远程计算机的缓冲区文件中。

注: 本节中的信息只适用于 Linux 和 UNIX 操作系统。

此进程分两个部分检索输出:

1. 在远程命令完成后。
2. 在 rshell 终止后，如果某些进程仍在运行，那么 rshell 可能会过一段时间才终止。

缺省情况下，缓冲区文件的名称为 /tmp/$USER/rahout，但可以通过环境变量 $**RAHBUFDIR** 或 $**RAHBUFNAME** 指定该名称。

当指定想要命令同时运行时，缺省情况下，此脚本将附加命令作为前缀加到发送至所有主机的命令上，以检查 $**RAHBUFDIR** 和 $**RAHBUFNAME** 是否可用于缓冲区文件。这会创建 $**RAHBUFDIR**。为避免此类情况，导出环境变量 RAHCHECKBUF=no。如果知道目录存在且可用，那么可以执行此操作以节省时间。

在使用 **rah** 同时在多台计算机上运行命令之前:

- 确保对于您的用户标识，目录 /tmp/$USER 在每台计算机上存在。要在目录尚不存在的情况下创建目录，请运行:

  ```
  rah ")mkdir /tmp/$USER"
  ```

- 将下一行添加至 .kshrc （对于 Korn shell 程序语法）或 .profile，并将它输入到当前会话中:

  ```
  export RAHCHECKBUF=no
  ```

- 确保您运行远程命令的每台计算机的标识在其 .rhosts 文件中有一个条目对应于运行 **rah** 的标识；并且运行 **rah** 的标识在其 .rhosts 文件中有一个条目对应于运行远程命令的每台计算机的标识。

扩展 rah 命令以使用树逻辑（AIX 和 Solaris）

为了增强性能，在大型系统上扩展了 rah 以使用 tree_logic。也就是说，rah 将检查该列表包含多少个数据库分区，若该数目超过阈值，它会构造列表的一个子集，并将它自己的递归调用发送到那些数据库分区。

在那些数据库分区上，递归调用的 rah 遵循相同的逻辑，直到该列表小得能够符合将该命令发送到列表中所有数据库分区的标准逻辑（现在称为“树叶”逻辑）为止。该阈值可由环境变量 **RAHTREETHRESH** 指定，或缺省为 15。

对于每个物理数据库分区存在多个逻辑数据库分区的系统，**db2_all** 更愿意将递归调用发送到各个不同的物理数据库分区，然后 rsh 到同一个物理数据库分区上的其他逻辑数据库分区，这样可减少物理数据库分区间的通信量。（这种方法只适合 **db2_all**，不适合 rah，因为 rah 始终只发送到不同的物理数据库分区。）

rah 和 db2_all 命令

本主题包括对 **rah** 和 **db2_all** 命令的描述。

命令 **描述**

rah 在所有计算机上运行该命令。

db2_all

在指定的所有数据库分区服务器上运行非交互式命令。**db2_all** 命令不支持需要交互式用户输入的命令。

db2_kill

突然停止正在多个数据库服务器上运行的所有进程，并清除所有数据库分区服务器上的所有资源。此命令使数据库变得不一致。除非在 IBM 软件支持机构的指导下，否则，请不要发出此命令；或者按指示从持续陷阱进行恢复。

db2_call_stack

在 Linux 和 UNIX 操作系统上，使在所有数据库分区服务器上运行的所有进程将调用回溯写入 syslog。

在 Linux 和 UNIX 操作系统上，这些命令执行带特定隐式设置的 **rah**，例如:

- 以并行方式在所有计算机上运行
- 将命令输出分别缓存到 `/tmp/$USER/db2_kill` 和 `/tmp/$USER/db2_call_stack` 中。

命令 **db2_call_stack** 在 Windows 上不可用。请改为使用 **db2pd -stack** 命令。

rah 和 db2_all 命令前缀顺序

前缀序列是一个或多个特殊字符。

在命令字符前输入一个或多个前缀序列而不插入任何空格。若想指定多个序列，可以任何顺序输入它们，但任何多字符序列中的输入字符必须按顺序输入。如果您输入任何前缀序列，那么您必须将整个命令（包括该前缀序列）置于双引号内，如下例所示:

- 在 Linux 和 UNIX 操作系统上:

```
  rah "};ps -F pid,ppid,etime,args -u $USER"  db2_all "};
ps -F pid,ppid,etime,args -u $USER"
```

- 在 Windows 操作系统上:

```
rah "||db2 get db cfg for sample"  db2_all "||db2 get db cfg for sample"
```

前缀序列有:

序列	用途
|	在后台按顺序运行命令。
|&	在后台按顺序运行这些命令，并在所有远程命令完成之后，终止该命令，即使有一些进程仍在运行。例如，如果子进程（在 Linux 和 UNIX 操作系统上）或后台进程（在 Windows 操作系统上）仍在运行，那么可能会造成延迟。在此情况下，该命令启动独立的后台进程来检索命令终止之后生成的任何远程输出，并将该输出写回至源计算机。 **注**: 在 Linux 和 UNIX 操作系统上，指定 & 会降低性能，因为需要运行更多的 **rsh** 命令。
||	在后台以并行方式运行命令。
||&	在后台以并行方式运行命令并在所有远程命令完成之后终止该命令，如先前对 |& 情况的描述一样。 **注**: 在 Linux 和 UNIX 操作系统上，指定 & 会降低性能，因为需要运行更多的 **rsh** 命令。
;	与 ||& 相同。这是一个较短的替代格式。 **注**: 在 Linux 和 UNIX 操作系统上，指定 ; 会降低性能（相对于 || 而言），因为需要运行更多的 **rsh** 命令。
]	在执行命令之前预先暂挂用户概要文件的点执行。 **注**: 仅在 Linux 和 UNIX 操作系统上可用。
}	在执行命令之前预先暂挂在 $**RAHENV** 中指定的文件的点执行（可能是 .kshrc）。 **注**: 仅在 Linux 和 UNIX 操作系统上可用。
]}	在执行命令之前，预先暂挂用户概要文件的点执行，然后执行在 $**RAHENV** 中指定的文件（可能是 .kshrc）。 **注**: 仅在 Linux 和 UNIX 操作系统上可用。
)	停止执行用户概要文件和 $**RAHENV** 中指定的文件。 **注**: 仅在 Linux 和 UNIX 操作系统上可用。
'	将命令调用回传至计算机。
<	发送至除此计算机外的所有计算机。
<<-*nnn***<**	发送至除数据库分区服务器 *nnn* 外的所有数据库分区服务器（db2nodes.cfg 中除数据库分区号为 *nnn* 之外的所有数据库分区服务器，请参阅本表中最后一个前缀序列后的第一段）。 *nnn* 是 1 位、2 位或 3 位的数据库分区号，该分区号与 db2nodes.cfg 文件中的 *nodenum* 值对应。

<<-nnn< 仅适用于 **db2_all**。

<<+*nnn***<**

仅发送至数据库分区服务器 *nnn*（db2nodes.cfg 中数据库分区号为 *nnn* 的数据库分区服务器，请参阅本表中最后一个前缀序列后的第一段）。

nnn 是 1 位、2 位或 3 位的数据库分区号，该分区号与 db2nodes.cfg 文件中的 *nodenum* 值对应。

<<+nnn< 仅适用于 **db2_all**。

（空白字符）

在后台运行远程命令，stdin、stdout 和 stderr 全部关闭。此选项仅当在后台运行命令时才有效，即仅在还包括 \ 或 ; 的前缀序列中有效。它允许命令尽快完成（远程命令一启动就完成）。如果在 **rah** 命令行上指定此前缀序列，那么将该命令用单引号括起来，或用双引号括起该命令并在前缀字符之前加 \。例如，

```
rah '; mydaemon'
```

或者

```
rah "; \ mydaemon"
```

当作为后台进程运行时，**rah** 命令从不会等待任何要返回的输出。

> 用计算机名称替换找到的 >。

″ 用计算机索引替换找到的 ()，用数据库分区号替换找到的 ##。

- 计算机索引是与数据库系统中的计算机关联的号码。如果没有在运行多逻辑分区，那么计算机的计算机索引对应于数据库分区配置文件中该计算机的数据库分区号。要在多逻辑分区数据库环境中获取计算机的计算机索引，不要计算那些运行多逻辑分区的计算机的重复条目。例如，如果 MACH1 正在运行两个逻辑分区，MACH2 也正在运行两个逻辑分区，那么数据库分区配置文件中 MACH3 的数据库分区号为 5。但是，MACH3 的计算机索引应是 3。
  - 在 Windows 操作系统上，不要编辑数据库分区配置文件。要获取计算机索引，可使用 **db2nlist** 命令。
- 当指定了 " 时，不会从计算机列表中删除重复项。

使用说明

- 前缀序列被认为是命令的一部分。若指定前缀序列作为命令的一部分，必须将整个命令，包括前缀序列，括在双引号内。

控制 rah 命令

本主题列示用于控制 **rah** 命令的环境变量。

*表 13. 控制 **rah** 命令的环境变量*

名称	含义	缺省值
$**RAHBUFDIR** **注**: 仅在 Linux 和 UNIX 操作系统上可用。	缓冲区目录	/tmp/$USER

*表 13. 控制 **rah** 命令的环境变量 (续)*

名称	含义	缺省值
$**RAHBUFNAME** **注**: 仅在 Linux 和 UNIX 操作系统上可用。	缓冲区文件名	rahout
$**RAHOSTFILE**（在 Linux 和 UNIX 操作系统上）; **RAHOSTFILE**（在 Windows 操作系统上）	包含主机列表的文件	db2nodes.cfg
$**RAHOSTLIST**（在 Linux 和 UNIX 操作系统上）; **RAHOSTLIST**（在 Windows	字符串形式的主机列表	抽取自 $**RAHOSTFILE**
$**RAHCHECKBUF** **注**: 仅在 Linux 和 UNIX 操作系统上可用。	若设置为“no”，则绕过检查	未设置
$**RAHSLEEPTIME**（在 Linux 和 UNIX 操作系统上）; **RAHSLEEPTIME**（在 Windows 操作系统上）	以秒计的时间，此脚本将在此时间内等待来自以并行形式运行的命令的初始输出。	对于 **db2_kill** 为 86400 秒，对于所有其他命令则为 200 秒
$**RAHWAITTIME**（在 Linux 和 UNIX 操作系统上）; **RAHWAITTIME**（在 Windows 操作系统上）	在 Windows 操作系统上，连续检查远程作业是否仍在运行的时间间隔（以秒计）。 在 Linux 和 UNIX 操作系统上，连续检查远程作业是否仍在运行以及 rah: waiting for pid> ... 消息的时间间隔（以秒计）。 对于所有操作系统，指定任何正整数。具有前导零的前缀值可抑制消息，例如，export RAHWAITTIME=045。 不必指定较小的值，因为 **rah** 不依靠这些检查来检测作业是否完成。	45 秒
$**RAHENV** **注**: 仅在 Linux 和 UNIX 操作系统上可用。	如果 $**RAHDOTFILES**=E、K、PE 或 B，那么指定要执行的文件名	$**ENV**
$**RAHUSER**（在 Linux 和 UNIX 操作系统上）; **RAHUSER**（在 Windows 操作系统上）	在 Linux 和 UNIX 操作系统上，运行远程命令将使用的用户标识。 在 Windows 操作系统上，与 DB2“远程命令服务”关联的登录帐户	$**USER**

注: 在 Linux 和 UNIX 操作系统上，将使用运行 **rah** 的 $**RAHENV** 的值，而不是远程 shell 设置的值（如果有）。

指定与 rah 一起运行的 . 文件（Linux 和 UNIX）

本主题列示在未指定前缀序列时运行的 . 文件。

注: 本节中的信息只适用于 Linux 和 UNIX 操作系统。

P .profile

E 在 $**RAHENV** 中指定的文件（可能是 .kshrc）

K 与 E 相同

PE .profile 后跟在 $**RAHENV** 中指定的文件（可能是 .kshrc）

B 与 PE 相同

N 无

注: 如果登录 shell 程序不是 Korn shell 程序，那么将在 Korn shell 程序进程中执行您指定要执行的任何点文件，因此，必须遵守 Korn shell 程序语法。例如，如果登录 shell 程序是 C shell，要对 **rah** 执行的命令设置 .cshrc 环境，应该创建等效于 .cshrc 的 Korn shell 程序 *INSTHOME*/.profile，并在 *INSTHOME*/.cshrc 中指定:

```
setenv RAHDOTFILES P
```

或者应该创建等价于 .cshrc 的 Korn shell 程序 *INSTHOME*/.kshrc，并在 *INSTHOME*/.cshrc 中指定:

```
setenv RAHDOTFILES E
setenv RAHENV INSTHOME/.kshrc
```

此外，如果没有 tty（由 **rsh** 调用时），那么 .cshrc 不得写入标准输出。通过将写至标准输出的任何行用引号括起来，可确保这一点，例如,

```
if { tty -s } then echo "executed .cshrc";
  endif
```

确定 rah 的问题（Linux 和 UNIX）

本主题提供了一些建议，告诉您如何处理在运行 **rah** 时可能遇到的某些问题。

注: 本节中的信息只适用于 Linux 和 UNIX 操作系统。

1. **rah** 挂起（或运行很长的时间）

   此问题可能是由下列原因引起的:

   - **rah** 已确定它需要缓冲输出，而您未导出 RAHCHECKBUF=no。因此，在运行命令之前，**rah** 向所有计算机发出一条命令以检查缓冲区目录是否存在，如果该目录不存在，那么会创建该目录。
   - 发送命令的一台或多台计算机不响应。**rsh** 命令最终将超时，但超时时间间隔相当长，通常为 60 秒左右。

2. 已接受到下列各类消息:
   - 登录不正确
   - 拒绝许可权

其中一台计算机没有在其 /etc/hosts 文件中正确定义运行 **rah** 的标识，或运行 **rah** 的标识没有在其 .rhosts 文件中正确定义其中一台计算机。如果已将 **DB2RSHCMD** 注册表变量配置为使用 ssh，则可能是未正确配置每台计算机上的 ssh 客户机和服务器。

注: 对于以明文方式在数据库分区之间传输密码的情况，可能需要使用更高的安全性。这将取决于正在使用的远程 shell 程序。**rah** 使用 **DB2RSHCMD** 注册表变量指定的远程 shell 程序。可以在两个远程 shell 程序之间选择: ssh（用于更高的安全性要求）或 rsh（对于 HP-UX，则为 remsh）。如果不设置此注册表变量，那么使用 rsh（对于 HP-UX，则为 remsh）。

3. 当使用后台远程 shell 以并行方式运行命令时，虽然这些命令在期望的时间内在计算机上运行并完成，但是 **rah** 要耗费较长的时间来检测它并设置 shell 提示符。

   运行 **rah** 的标识没有在其 .rhosts 文件中正确定义其中一台计算机，或者如果已将 **DB2RSHCMD** 注册表变量配置为使用 ssh，那么可能是未正确配置每台计算机上的 ssh 客户机和服务器。

4. 虽然 **rah** 从 shell 命令行运行时运行情况良好，但是如果使用 rsh 远程运行 **rah**，例如，

   ```
   rsh somewher -l $USER db2_kill
   ```

   ，那么 **rah** 将永不会完成。

   这是正常的。**rah** 会启动后台监视进程，这些进程在 rah 退出后继续运行。那些进程通常将在与您运行的命令关联的所有进程将它们终止之后才结束。在 **db2_kill** 的情况下，这意味着终止所有数据库管理器。可通过查找其命令是 **rahwaitfor** 和 kill *process_id*> 的进程来终止监视进程。不要指定信号编号。而是要使用缺省值（15）。

5. 当在同一 $**RAHUSER** 下发出多个 **rah** 命令时，**rah** 的输出未正确显示，或 **rah** 错误地报告 $**RAHBUFNAME** 不存在。

   这是因为，并行执行多个 **rah** 正在尝试使用同一缓冲区文件（例如，$**RAHBUFDIR** 或 $**RAHBUFNAME**）对输出进行缓冲。要避免此问题，将不同的 $**RAHBUFNAME** 用于每个并行 **rah** 命令，例如，在下列 ksh 中:

   ```
   export RAHBUFNAME=rahout
     rah ";$command_1" &
     export RAHBUFNAME=rah2out
     rah ";$command_2" &
   ```

   或使用使 shell 自动选择唯一名称的方法，如:

   ```
   RAHBUFNAME=rahout.$$ db2_all "....."
   ```

   如果磁盘空间有限，那么无论使用哪种方法，都必须确保在某个时间清除缓冲区文件。**rah** 不会在执行结束时擦除缓冲区文件，不过，下次您指定与现有文件相同的缓冲区文件时，它将擦除并复用现有文件。

6. 输入

   ```
   rah '"print from ()'
   ```

   并接收到消息:

   ```
   ksh: syntax error at line 1 : (' unexpected
   ```

   替代 () 和 ## 的先决条件是:

- 使用 **db2_all**，而不是使用 **rah**。
- 确保通过导出 **RAHOSTFILE** 或缺省为您的 /sqllib/db2nodes.cfg 文件来使用 **RAHOSTFILE**。没有这些先决条件，**rah** 将照原样保持 () 和 ##。您接收到错误，因为命令 **print from ()** 无效。

要获取以并行方式运行命令时的性能提示，除非确实需要由 & 提供的功能，否则，使用 | 代替 |&，并使用 || 代替 ||& 或 ;。指定 & 需要更多远程 shell 命令，因此会降低性能。

监视 rah 进程（Linux 和 UNIX）

当任何远程命令仍在运行或仍在累加缓冲输出时，由 **rah** 启动的进程将监视活动以将指示哪些活动尚未运行的消息写入终端并检索缓冲输出。

关于此任务

注: 本节中的信息只适用于 Linux 和 UNIX 操作系统。

按环境变量 **RAHWAITTIME** 所控制的时间间隔写出参考消息。请参阅帮助信息以获取有关如何指定此设置的详细信息。通过导出 RAHWAITTIME=0，可以不显示所有参考消息。

主监视过程是一个命令，其命令名（由 **ps** 命令显示）为 **rahwaitfor**。第一条参考消息告诉您此进程的 pid（进程标识）。所有其他监视进程将表现为运行 **rah** 脚本（或符号链接的名称）的 **ksh** 命令。若愿意，可通过以下命令停止所有监视进程:

```
kill pid
```

其中 *pid* 是主监视进程的进程标识。不要指定信号编号。让它为缺省值 15。这根本不会影响远程命令，但会阻止自动显示缓冲输出。注意，在执行单个 **rah** 时，可能有两组或更多组不同监视进程在不同时间执行。但是，如果任何时候停止当前组监视进程，那么不会再启动其他监视进程。

如果正规登录 shell 程序不是 Korn shell 程序（例如，/bin/ksh），那么可以使用 **rah**，但如何输入包含下列特殊字符的命令的规则稍微有些差异:

```
"  unsubstituted $ '
```

有关更多信息，请输入 rah "?"。而且，在 Linux 或 UNIX 操作系统中，如果执行远程命令的标识的登录 shell 程序不是 Korn shell 程序，那么以该标识执行 **rah** 的登录 shell 程序也不得是 Korn shell 程序。（**rah** 根据本地标识来决定远程标识的 shell 是否为 Korn shell 程序）。该 shell 不得对单引号内的字符串执行任何替换或特殊处理。必须照原样保持。

在 Windows 上为 rah 设置缺省环境概要文件

要为 **rah** 命令设置缺省环境概要文件，可使用文件 db2rah.env，该文件应在实例目录中创建。

关于此任务

注: 本节中的信息只适用于 Windows。

该文件应该有以下格式:

```
; This is a comment line
   DB2INSTANCE=instancename
   DB2DBDFT=database
   ; End of file
```

可以指定为 **rah** 进行环境初始化所需要的所有环境变量。

第 11 章 创建表和其他相关表对象

分区数据库环境中的表

在分区数据库环境中跨几个数据库分区创建表在性能上有几个优点。与检索数据相关联的工作可分成几部分在各个数据库分区中进行。

开始之前

在创建将以物理方式划分或分布的一个表之前，需要考虑下列事宜:

- 表空间可以横跨多个数据库分区。它们所跨的数据库分区数取决于数据库分区组中的数据库分区数。
- 可以通过如下方法来并置表: 将表置于同一个表空间中，或置于另一个表空间中，该表空间与第一个表空间一起，与同一个数据库分区组相关联。

关于此任务

在创建表时，指定创建的表将成为若干数据库分区的一部分。当在分区数据库环境中创建表时，有一个附加选项: *分布键*。分布键是作为一个表定义的一部分的键。它确定用于存储每行数据的数据库分区。

若不显式指定分布键，会使用下列缺省值。*确保缺省分布键适合*。

- 如果在 CREATE TABLE 语句中指定了主键，那么该主键的第一列会用作分布键。
- 对于多分区数据库分区组，如果不存在主键，那么使用非长整型字段的第一列。
- 如果没有列满足缺省分布键的要求，那么会不带关键字创建该表（这只在单一分区数据库分区组中允许）。

必须小心地选择适当的分布键，因为*以后再也不能更改它*。再者，必须将任何唯一索引（因此也是唯一键或主键）定义为分布键的一个超集。即，如果定义了分布键，那么唯一键和主键必须包括所有与分布键相同的列（它们可能有多列）。

数据库分区的表大小是与使用的表空间和页大小的类型关联的特定限制和可用的磁盘空间大小这两者中的较小者。例如，假定有一个 4 KB 页大小的大型 DMS 表空间，表大小是 8 TB 与数据库分区数目的乘积和可用的磁盘空间大小这两者中的较小者。请参阅相关链接，以获取数据库管理器页大小限制的完整列表。

要使用命令行来在分区数据库环境中创建一个表，请输入:

```
CREATE TABLE name>
        (<column_name>  <data_type>  <null_attribute>)
       IN <tagle_space_name>
         INDEX IN <index_space_name>
         LONG  IN <long_space_name>      DISTRIBUTE BY HASH (<column_name>)
```

以下是一个示例:

```
CREATE TABLE MIXREC (MIX_CNTL INTEGER NOT NULL,
                     MIX_DESC CHAR(20) NOT NULL,
                     MIX_CHR  CHAR(9) NOT NULL,
                     MIX_INT INTEGER NOT NULL,
                     MIX_INTS SMALLINT NOT NULL,
```

```
MIX_DEC DECIMAL NOT NULL,
MIX_FLT FLOAT NOT NULL,
MIX_DATE DATE NOT NULL,
MIX_TIME TIME NOT NULL,
MIX_TMSTMP TIMESTAMP NOT NULL)
                          IN MIXTS12
DISTRIBUTE BY HASH (MIX_INT)
```

在上一个示例中，表空间是 MIXTS12，而分布键是 MIX_INT。如果未显式指定分布键，那么它是 MIX_CNTL。（如果未指定主键且未定义分布键，那么分布键是该列表中的第一个非长型列。）

表的一行和有关该行的所有信息始终位于同一个数据库分区上。

分区表中的大对象行为

分区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。根据 CREATE TABLE 语句的 PARTITION BY 子句中指定的内容，给定表的数据被划分到多个存储对象中。这些存储对象可以在不同的表空间中，也可以在相同表空间中。

缺省情况下，分区表的大对象与其相应的数据对象存储在相同表空间中。这适用于只使用一个表空间或使用多个表空间的分区表。当分区表的数据存储在多个表空间中时，大对象数据也存储在多个表空间中。

使用 CREATE TABLE 语句的 LONG IN 子句来覆盖此缺省行为。可以为表指定将存储长型数据的表空间列表。如果选择覆盖缺省行为，那么在 LONG IN 子句中指定的表空间必须是大型表空间。如果指定将一个或多个数据分区的长型数据存储在单独的表空间中，那么必须对表的所有数据分区都如此。也就是说，不能将某些数据分区的数据存储在远地，而将其他数据分区的数据存储在本地。无论是使用缺省行为还是使用 LONG IN 子句来覆盖缺省行为，都会创建一个与每个数据分区相对应的长型对象。用于存储与每个数据分区对应的长型数据对象的所有表空间必须具有相同的页大小、扩展数据块大小、存储机制（DMS 或 AMS）和类型（常规或大型）。远程大型表空间的类型必须是 LARGE，不能是 SMS。

例如，以下 CREATE TABLE 语句在 CLOB 数据所在的表空间中为每个数据分区的 CLOB 数据创建对象:

```
CREATE TABLE document(id INT, contents CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1,
 STARTING FROM 101 ENDING AT 200 IN tbsp2,
 STARTING FROM 201 ENDING AT 300 IN tbsp3,
 STARTING FROM 301 ENDING AT 400 IN tbsp4);
```

可以使用 LONG IN 将 CLOB 数据放在一个或多个与该数据所在的大型表空间不同的大型表空间中。

```
CREATE TABLE document(id INT, contents CLOB)
PARTITION BY RANGE(id)
(STARTING FROM 1   ENDING AT 100 IN tbsp1 LONG IN large1,
STARTING FROM 101 ENDING AT 200 IN tbsp2 LONG IN large1,
STARTING FROM 201 ENDING AT 300 IN tbsp3 LONG IN large2,
STARTING FROM 301 ENDING AT 400 IN tbsp4 LONG IN large2);
```

注: 对于每个数据分区，在表级别只允许使用一个 LONG IN 子句。

创建分区表

分区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。根据 CREATE TABLE 语句的 PARTITION BY 子句中指定的内容，给定表的数据被划分到多个存储对象中。这些存储对象可以在不同的表空间中，也可以在相同表空间中。

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或特权:

- 对数据库的 CREATETAB 权限、对该表使用的所有表空间的 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

可以使用 CREATE TABLE 语句创建分区表。

过程

要使用命令行来创建分区表，请发出 CREATE TABLE 语句。

```
CREATE TABLE NAME (column_name data_type null_attribute) IN
 table_space_list PARTITION BY RANGE (column_expression)
 STARTING FROM constant ENDING constant EVERY constant
```

例如，以下语句将创建一个表，在该表中，满足 a ≥ 1 且 a ≤ 20 的行位于 PART0（第一个数据分区）中，满足 21 ≤ a ≤ 40 的行位于 PART1（第二个数据分区）中，而满足 81 ≤ a ≤ 100 的行位于 PART4（最后一个数据分区）中。

```
CREATE TABLE foo(a INT)
 PARTITION BY RANGE (a) (STARTING FROM (1)
 ENDING AT (100) EVERY (20))
```

定义分区表的范围

在创建分区表时，可以为每个数据分区指定范围。分区表使用了数据组织方案，即，表数据根据该表中表分区键列的值分布到多个数据分区中。

关于此任务

根据 CREATE TABLE 语句的 PARTITION BY 子句中指定的内容，给定表的数据被划分到多个存储对象中。范围由 PARTITION BY 子句的 STARTING FROM 和 ENDING AT 值指定。

要全面地定义每个数据分区的范围，必须指定足够的边界。以下是定义分区表的范围时要考虑的一系列准则:

- STARTING 子句指定数据分区范围的下界。对于最低数据分区范围来说，此子句是必需的（尽管可以将边界定义为 MINVALUE）。最低数据分区范围是具有最低指定边界的数据分区。

- ENDING（或 VALUES）子句指定数据分区范围的上界。对于最高数据分区范围来说，此子句是必需的（尽管可以将边界定义为 MAXVALUE）。最高数据分区范围是具有最高指定边界的数据分区。
- 如果未对某个数据分区指定 ENDING 子句，那么下一个更大数据分区就必须指定 STARTING 子句。否则，如果未指定 STARTING 子句，那么上一个数据分区就必须指定 ENDING 子句。
- MINVALUE 指定一个值，该值小于所用列类型的任何可能值。不能将 MINVALUE 与 INCLUSIVE 或 EXCLUSIVE 一起指定。
- MAXVALUE 指定一个值，该值大于所用列类型的任何可能值。不能将 MAXVALUE 与 INCLUSIVE 或 EXCLUSIVE 一起指定。
- INCLUSIVE 表示将所有等于指定值的值都包括在包含此边界的数据分区中。
- EXCLUSIVE 表示所有等于指定值的值都不包括在包含此边界的数据分区中。
- CREATE TABLE 语句的 NULL 子句指定考虑数据分区布置时是将空值安排在高位置还是低位置。缺省情况下，将空值安排在高位置。在此情况下，将把表分区键列中的空值视为正无穷并放到以 MAXVALUE 结尾的范围中。如果未定义这样的数据分区，就会将空值视为超出范围的值。如果要排除表分区键列中的空值，请使用 NOT NULL 约束。LAST 指定让空值在排序的值列表中最后出现。FIRST 指定让空值在排序的值列表中最先出现。
- 当使用长语法格式时，必须对每个数据分区至少指定一个边界。

提示: 在开始对表定义数据分区之前，您应该了解表分区是否能使表受益以及影响分区列选择的那些因素，这一点十分重要。

可以自动生成对每个数据分区指定的范围，也可以手动生成这些范围。

自动生成

自动生成方法十分简单，它使您能够快速方便地创建许多数据分区。此方法适合于创建基于日期或数值并且大小相等的范围。

示例 1 和 2 演示如何使用 CREATE TABLE 语句来自动定义和生成对每个数据分区指定的范围。

示例 1:

发出定义了下列范围的 CREATE TABLE 语句:

```
CREATE TABLE lineitem (
    l_orderkey    DECIMAL(10,0) NOT NULL,
    l_quantity    DECIMAL(12,2),
    l_shipdate    DATE,
    l_year_month  INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 +
                                           MONTH(l_shipdate)))
         PARTITION BY RANGE(l_shipdate)
        (STARTING ('1/1/1992') ENDING ('12/31/1992') EVERY 1 MONTH);
```

此语句生成 12 个数据分区，每个数据分区包含 1 个键值: (l_shipdate) >= ('1/1/1992'), (l_shipdate) < ('3/1/1992'), (l_shipdate) < ('4/1/1992'), (l_shipdate) < ('5/1/1992'), ..., (l_shipdate) < ('12/1/1992'), (l_shipdate) < ('12/31/1992')。

由于整体起始界限（'1/1/1992'）包括端值（缺省情况），所以第一个数据分区的起始值包括端值。同样，由于整体结束界限（'12/31/1992'）包括端值（缺省情况），所以

最后一个数据分区的结束界限包括端值。其余 STARTING 值都包括端值，并且其余 ENDING 值也都包括端值。每个数据分区都存放 n 个键值，其中 n 由 EVERY 子句指定。使用公式 (start + every) 来确定每个数据分区的范围末端。如果 START 到 END 的范围无法整除 EVERY 值，最后一个数据分区包含的键值就会较少。

示例 2:

发出定义了下列范围的 CREATE TABLE 语句:

```
CREATE TABLE t(a INT, b INT)
 PARTITION BY RANGE(b) (STARTING FROM (1)
  EXCLUSIVE ENDING AT (1000) EVERY (100))
```

此语句生成 10 个数据分区，每个数据分区包含 100 个键值: (1 < b <= 101, 101 < b <= 201, ..., 901 < b <= 1000)。

由于整体起始界限（1）不包括端值，所以第一个数据分区（b > 1 and b <= 101）的起始值不包括端值。同样，由于整体结束界限（1000）包括端值，所以最后一个数据分区（b > 901 b <= 1000）的结束界限包括端值。其余 STARTING 值都不包括端值，并且其余 ENDING 值全都包括端值。每个数据分区都存放 n 个键值，其中 n 由 EVERY 子句指定。最后，如果整个子句的起始和结束界限都不包括端值，那么由于整体起始界限（1）不包括端值，所以第一个数据分区的起始值不包括端值。同样，由于整体结束界限（1000）不包括端值，所以最后一个数据分区的结束界限不包括端值。其余 STARTING 值都不包括端值，并且 ENDING 值全都包括端值。每个数据分区（最后一个数据分区除外）都存放 n 个键值，其中 n 由 EVERY 子句指定。

手动生成

手动生成方法为 PARTITION BY 子句中列示的每个范围创建一个新数据分区。这种语法格式提高了定义范围时的灵活性，从而增加了数据和 LOB 布置选项。示例 3 和 4 演示如何使用 CREATE TABLE 语句来以手动方式定义和生成对数据分区指定的范围。

示例 3:

此语句对两个日期列进行分区，这两个日期列都是生成列。请注意自动生成的 CREATE TABLE 语法格式的使用，并注意每个范围都只指定了一端。另一端由相邻数据分区隐式确定并且要使用 INCLUSIVE 选项:

```
CREATE TABLE sales(invoice_date date, inv_month int NOT NULL
GENERATED ALWAYS AS (month(invoice_date)), inv_year INT NOT
NULL GENERATED ALWAYS AS ( year(invoice_date)), item_id int NOT NULL,
cust_id int NOT NULL) PARTITION BY RANGE (inv_year, inv_month)
(PART Q1_02 STARTING (2002,1) ENDING (2002, 3) INCLUSIVE,
PART Q2_02 ENDING (2002, 6) INCLUSIVE,
PART Q3_02 ENDING (2002, 9) INCLUSIVE,
PART Q4_02 ENDING (2002,12) INCLUSIVE,
PART CURRENT ENDING (MAXVALUE, MAXVALUE));
```

在范围之间允许存在间隔。对于未紧贴上一数据分区 ENDING 值的范围，CREATE TABLE 语法允许您对该范围指定 STARTING 值，从而支持间隔。

示例 4:

创建一个表，并且在值 101 与 200 之间存在间隔。

```
CREATE TABLE foo(a INT)
 PARTITION BY RANGE(a)
      (STARTING FROM (1) ENDING AT (100),
      STARTING FROM (201) ENDING AT (300))
```

使用允许添加或除去数据分区的 ALTER TABLE 语句还会导致范围中出现间隔。

把行插入分区表时，根据该行的键值以及它所处的范围自动将其放入正确的数据分区。如果该行处于对该表定义的所有范围之外，插入就会失败，并且将把以下错误返回给应用程序:

```
SQL0327N  由于该行处于已定义数据分区范围的界限外部，所以无法将其插入到表
<tablename> 中。    SQLSTATE=22525
```

限制

- 表级别限制:
  - 使用自动生成的语法格式（包含 EVERY 子句）创建的表在表分区键中只能使用数字或日期时间类型。
- 语句级限制:
  - 在自动生成的语法格式中，不支持 MINVALUE 和 MAXVALUE。
  - 范围按升序排列。
  - 在自动生成的语法格式中，只能指定一列。
  - EVERY 子句中的增量必须大于零。
  - ENDING 值必须大于或等于 STARTING 值。

数据分区数据，索引和长整型数据的放置

从本质上说，创建分区表允许您替换表的各个部分以及特定表空间中的关联表对象。

创建表时，可指定将替换哪个表空间中的完整表数据和关联表对象。或者，可替换特定表空间中的表索引、长数据或大数据或表分区。所有表空间都必须在同一个数据库分区组中。

CREATE TABLE 语句具有下列子句，演示了替换特定表空间中的表数据和关联表对象的功能。

```
CREATE TABLE table_name IN table_space_name1
   INDEX IN table_space_name2
   LONG IN table_space_name3
   PARTITIONED BY ...
     PARTITION partition_name | boundary specification | IN table_space_name4
       INDEX IN table_space_name5
       LONG IN table_space_name6
```

可在不同表空间中替换该分区表的每个分区。

还可使用 CREATE INDEX ... IN *table_space_name1* 语句来对分区表上用户创建的非分区索引指定表空间，该表空间可能与 CREATE TABLE ... INDEX IN *table_space_name2* 语句中指定的索引表空间不同。CREATE INDEX 语句的 IN 子句仅用于分区表。如果未在 CREATE TABLE 或 CREATE INDEX 语句上指定 INDEX IN 子句，那么索引将放在该表的第一个可视或连接分区所在的表空间上。

系统生成的非分区索引（例如 XML 列路径索引）放在 CREATE TABLE 语句的 INDEX IN 子句中指定的表空间上。

在带有 XML 数据的分区表上，XML 区域索引与表数据始终以相同方式分区。分区索引的表空间是在分区级别上定义的

XML 数据位于表的长整型数据使用的表空间中。分区表上 XML 数据的放置遵从长整型数据放置规则。

可以显式指定或由数据库管理器隐式确定用于长整型数据的表空间。对于分区表，可以将表级别 LONG IN 子句与分区级别 LONG IN 子句配合使用。如果同时指定了两者，那么分区级别 LONG IN 子句优先于任何表级别 LONG IN 子句。

将现有表和视图迁移到分区表

可将非分区表或 UNION ALL 视图迁移至空分区表。

开始之前

如果特定列的 SYSCAT.COLUMNS.IMPLICITVALUE 对于源列和目标列为非空值，并且这些值不匹配，那么不允许连接数据分区。在不允许连接数据分区的情况下，必须将源表删除，然后重新创建该表。

如果符合下列任一条件，那么列的 SYSCAT.COLUMNS IMPLICITVALUE 字段中可具有非空值:

- IMPLICITVALUE 字段是在连接操作期间从源表传播而来的。
- IMPLICITVALUE 字段是在拆离操作期间从源表继承而来的。
- IMPLICITVALUE 字段是从 V8 迁移到 V9 期间设置的，它被确定为已添加列或可能为已添加列。已添加列是因为执行 ALTER TABLE...ADD COLUMN 语句而创建的列。

始终使用已定义的相同列创建连接操作涉及的源表和目标表。特别是，决不应使用 ALTER TABLE 语句向连接操作的目标表添加列。

有关使用分区表时避免不匹配的建议，请参阅第 187 页的『将数据分区连接至分区表的准则』。

关于此任务

迁移常规表时，使用 **EXPORT** 命令或高性能卸载来卸载源表。创建新的空分区表，并使用 **LOAD** 命令来填充该分区表。要将数据从旧表直接移至分区表而不需要任何中间步骤，请使用 **LOAD FROM CURSOR** 命令（请参阅步骤 1）。

可将 UNION ALL 视图中的非分区数据转换为分区表（请参阅步骤 2）。UNION ALL 视图用于管理大型的表，它简化了表数据的转入和转出，并且提供了分支消除的性能优势。通过使用 ALTER TABLE...ATTACH PARTITION 语句，可以在不移动基本表中的数据的情况下完成转换。转换后，必须重新创建非分区索引和从属视图或具体化查询表 (MQT)。用于将 UNION ALL 视图转换为分区表的建议的策略是创建带有单一哑数据分区的分区表，然后连接 UNION ALL 视图的所有表。务必在处理过程中尽早删除虚拟数据分区以避免范围重叠问题。

过程

1. 将常规表迁移至分区表。 使用 **LOAD FROM CURSOR** 命令来避免任何中间步骤。以下示例说明如何将表 T1 迁移至 SALES_DP 表。

a. 创建并填充常规表 T1。

```
CREATE TABLE t1 (c1 int, c2 int);INSERT INTO t1 VALUES (0,1), (4, 2), (6, 3);
```

b. 创建空分区表。

```
CREATE TABLE sales_dp (c1 int, c2 int)
  PARTITION BY RANGE (c1)
  (STARTING FROM 0 ENDING AT 10 EVERY 2);
```

c. 使用 **LOAD FROM CURSOR** 命令将 SQL 查询中的数据直接拉入至新分区表。

```
 SELECT * FROM t1;
 DECLARE c1 CURSOR FOR SELECT * FROM t1;
 LOAD FROM c1 of CURSOR INSERT INTO sales_dp;SELECT * FROM sales_dp;
```

2. 将 UNION ALL 视图中的非分区数据转换为分区表。 以下示例说明如何将名为 ALL_SALES 的 UNION ALL 视图转换为 SALES_DP 表。

a. 创建 UNION ALL 视图。

```
CREATE VIEW all_sales AS
 (
 SELECT * FROM sales_0198
 WHERE sales_date BETWEEN '01-01-1998' AND '01-31-1998'
 UNION ALL
 SELECT * FROM sales_0298
 WHERE sales_date BETWEEN '02-01-1998' AND '02-28-1998'
 UNION ALL
 ...
 UNION ALL
 SELECT * FROM sales_1200
 WHERE sales_date BETWEEN '12-01-2000' AND '12-31-2000'
 );
```

b. 创建带有单一虚拟分区的分区表。选择范围以使它不会与要连接的第一个数据分区重叠。

```
CREATE TABLE sales_dp (
 sales_date DATE NOT NULL,
 prod_id INTEGER,
 city_id INTEGER,
 channel_id INTEGER,
 revenue DECIMAL(20,2))
 PARTITION BY RANGE (sales_date)
 (PART dummy STARTING FROM '01-01-1900' ENDING AT '01-01-1900');
```

c. 连接第一个表。

```
ALTER TABLE sales_dp ATTACH PARTITION
  STARTING FROM '01-01-1998' ENDING AT '01-31-1998'
  FROM sales_0198;
```

d. 删除哑分区。

```
ALTER TABLE sales_dp DETACH PARTITION dummy
  INTO dummy;
DROP TABLE dummy;
```

e. 连接余下分区。

```
ALTER TABLE sales_dp ATTACH PARTITION
  STARTING FROM '02-01-1998' ENDING AT '02-28-1998'
  FROM sales_0298;
...
ALTER TABLE sales_dp ATTACH PARTITION
  STARTING FROM '12-01-2000' ENDING AT '12-31-2000'
  FROM sales_1200;
```

f. 发出 SET INTEGRITY 语句以使查询可以访问新连接的分区中的数据。

```
SET INTEGRITY FOR sales_dp IMMEDIATE CHECKED
FOR EXCEPTION IN sales_dp USE sales_ex;
```

提示: 如果在执行连接操作之前可以通过独立于数据服务器的应用程序逻辑执行数据完整性检查（包括范围验证和其他约束检查），那么可以更快地使新连接的数据可供使用。通过使用 SET INTEGRITY...ALL IMMEDIATE UNCHECKED 语句来跳过范围检查和约束违例检查，可以优化数据转入过程。在此情况下，表将脱离 SET INTEGRITY 暂挂状态，并且只要目标表中没有非分区用户索引，新数据就可以立即供应用程序使用。

g. 根据情况，创建索引。

将现有索引转换为分区索引

可能需要将系统创建的索引和用户创建的索引从非分区索引迁移到分区索引。对于大部分此类迁移，在对表和索引维护可用性期间可以转换用户创建的索引。在进行转换期间，系统创建的索引（用于强制执行主键约束或唯一键约束）将不能维护这些约束。

开始之前

在产品的较低发行版中创建的索引可能为非分区索引。这可包括您创建的索引，也可包括由数据库管理器创建的系统创建索引。系统创建索引的示例有用于强制执行唯一约束和主约束的索引以及 MDC 表的块索引。

关于此任务

您创建的索引可以从非分区索引转换为分区索引，同时对于使用该索引的数据保持连续可用性。可以创建与对应的非分区索引具有相同键的分区索引。在创建分区索引期间，仍然可以使用当前索引和正在其中创建索引的表。一旦创建了分区索引，就可以删除对应的非分区索引并在需要时重命名新的分区索引。

结果

下面的示例演示如何将现有非分区索引转换为分区索引。

示例

以下是将您创建的非分区索引转换为分区索引的示例:

```
UPDATE COMMAND OPTIONS USING C OFF;
CREATE INDEX data_part ON sales(sale_date) PARTITIONED;
DROP INDEX dateidx;
RENAME INDEX data_part TO dateidx;
COMMIT;
```

以下是将数据库管理器创建的非分区索引转换为分区索引的示例。此情况下，在删除原始约束与创建新约束之间将存在时间段。

```
ALTER TABLE employees DROP CONSTRAINT emp_uniq;
ALTER TABLE employees ADD CONSTRAINT emp_uniq UNIQUE (employee_id);
```

使用 DB2 V9.7 及更低发行版创建的 MDC 表具有非分区块索引。要利用分区表数据的可用性功能，例如，数据的转入和转出以及表数据和索引的分区级别重组，必须将使用 DB2 V9.7 及更低发行版创建的多维集群 (MDC) 表中的数据移到具有使用 DB2 V9.7 FP1 或更高发行版创建的分区块索引的分区 MDC 表。

联机移动分区 MDC 表以使用分区块索引

可通过联机表移动将数据从具有非分区块索引的 MDC 表移到具有分区块索引的 MDC 表。

在以下示例中，company1.parts 表中 **region** 和 **color** 为 MDC 键列；相应的块索引为非分区块索引。

```
CALL SYSPROC.ADMIN_MOVE_TABLE(
 'COMPANY1',  --Table schema
 'PARTS',     --Table name
 ' ',         --null; No change to columns definition
 ' ',         --null; No additional options
 'MOVE');     --Move the table in one step
```

脱机移动分区 MDC 表以使用分区块索引

为了最小化数据移动，可以在表脱机时将数据从具有非分区块索引的 MDC 表移到具有分区块索引的 MDC 表。此过程使用下列步骤:

1. 创建一个与要转换的表具有相同定义的新单分区 MDC 表。为分区指定范围时，请使用在要转换的 MDC 表的范围之外的范围。

   新单分区 MDC 表的块索引已分区。在稍后步骤中将拆离指定范围时所创建的分区。
2. 拆离 MDC 表的每个分区。每个分区将成为独立的 MDC 表。

   当拆离分区时，分区数据将连接至新目标表，不需要移动分区中的数据。

   注: 不能拆离 MDC 表的最后一个分区。它是具有非分区块索引的单分区 MDC 表。
3. 对于由拆离 MDC 表分区创建的每个独立表，以及具有非分区块索引的单分区 MDC 表，将表连接至在步骤 1 中创建的新分区 MDC 表。

   在连接表时，表数据将连接至新分区 MDC 表，无需移动数据，而块索引将创建为分区块索引。
4. 在连接第一个独立 MDC 表之后，可以连接创建新 MDC 表时所创建的空分区。
5. 对新分区 MDC 表发出 SET INTEGRITY 语句。

下一步做什么

已分区的具体化查询表 (MQT) 行为

分区表支持所有类型的具体化查询表 (MQT)。使用已分区的 MQT 时，可使用一些准则来帮助您更有效地管理已连接的数据分区和已拆离的数据分区。

当使用已分区的 MQT 或带有已拆离的从属表的分区表时，下列准则和限制适用:

- 当您发出 ALTER TABLE ... DETACH PARTITION 语句时，DETACH 操作将为已拆离分区数据创建目标表。如果有任何需要根据已拆离数据分区进行增量维护的从属表（这些从属表称为已拆离的从属表），那么需要对已拆离的从属表运行 SET INTEGRITY 语句，以便对这些表进行增量维护。利用 DB2 V9.7 FP1 或更高发行版，在对所有已拆离的从属表运行 SET INTEGRITY 语句后，异步分区拆离任务将使数据分区进入独立的目标表。直到异步分区拆离操作完成，目标表将不可用。对于目标表，在 SYSCAT.TABLES 目录视图的 TYPE 列中将标记“L”。此表称为已拆离表。这样，在运行 SET INTEGRITY 语句以增量方式维护已拆离的从属表之前，无法读取、修改或删除目标表。在对所有已拆离的从属表运行 SET INTEGRITY 语

句后，数据分区在逻辑上将从源表拆离，异步分区拆离操作将该数据分区从源表拆离到目标表中。直到异步分区拆离操作完成，目标表将不可用。

- 要检测已拆离表是否仍不可访问，请查询 SYSCAT.TABDETACHEDDEP 目录视图。如果检测到任何不可访问的已拆离表，那么对所有已拆离的从属表运行带有 IMMEDIATE CHECKED 选项的 SET INTEGRITY 语句，以将已拆离表转换为可访问的常规表。如果在维护已拆离表的所有已拆离从属表之前试图访问该已拆离表，那么将返回错误代码 SQL20285N。
- 不能在具体化查询表 (MQT) 定义中使用 DATAPARTITIONNUM 函数。尝试使用此函数来创建 MQT 将返回错误（SQLCODE SQL20058N 和 SQLSTATE 428EC）。
- 当为带有已拆离数据分区且在 SYSCAT.DATAPARTITIONS 中的 STATUS 为“D”的表创建非分区索引时，索引将不包括已拆离分区中的数据，除非已拆离分区具有需要根据该数据分区进行增量刷新的从属具体化查询表 (MQT)。在后面这种情况下，索引将包括此已拆离数据分区的数据。
- 不允许将带有已连接数据分区的表改变为 MQT。
- 不支持已分区的登台表。
- 不直接支持与 MQT 连接。有关详细信息，请参阅示例 1。

示例 1: 将已分区的 MQT 转换为一个普通表

虽然不直接支持对已分区的 MQT 执行 ATTACH 操作，但可以通过下列操作获得相同的效果: 将已分区的 MQT 转换为一个普通表，对表数据执行期望的转入和转出，然后将该表转换回 MQT。下列 CREATE TABLE 和 ALTER TABLE 语句演示了该效果:

```
CREATE TABLE lineitem (
  l_orderkey    DECIMAL(10,0) NOT NULL,
  l_quantity    DECIMAL(12,2),
  l_shipdate    DATE,
  l_year_month  INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 +
                                         MONTH(l_shipdate)))
    PARTITION BY RANGE(l_shipdate)
    (STARTING ('1/1/1992') ENDING ('12/31/1993') EVERY 1 MONTH);
CREATE TABLE lineitem_ex (
  l_orderkey    DECIMAL(10,0) NOT NULL,
  l_quantity    DECIMAL(12,2),
  l_shipdate    DATE,
  l_year_month  INT,
  ts            TIMESTAMP,
  msg           CLOB(32K));

CREATE TABLE quan_by_month (
  q_year_month, q_count) AS
      (SELECT l_year_month AS q_year_month, COUNT(*) AS q_count
          FROM lineitem
            GROUP BY l_year_month)
     DATA INITIALLY DEFERRED REFRESH IMMEDIATE
     PARTITION BY RANGE(q_year_month)
     (STARTING (199201) ENDING (199212) EVERY (1),
      STARTING (199301) ENDING (199312) EVERY (1));
CREATE TABLE quan_by_month_ex(
  q_year_month  INT,
  q_count       INT NOT NULL,
  ts            TIMESTAMP,
  msg           CLOB(32K));

SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED;
CREATE INDEX qbmx ON quan_by_month(q_year_month);

ALTER TABLE quan_by_month DROP MATERIALIZED QUERY;
```

```
ALTER TABLE lineitem DETACH PARTITION part0 INTO li_reuse;
ALTER TABLE quan_by_month DETACH PARTITION part0 INTO qm_reuse;

SET INTEGRITY FOR li_reuse OFF;
ALTER TABLE li_reuse ALTER l_year_month SET GENERATED ALWAYS AS
(YEAR(l_shipdate)*100 + MONTH(l_shipdate));

LOAD FROM part_mqt_rotate.del OF DEL MODIFIED BY GENERATEDIGNORE
MESSAGES load.msg REPLACE INTO li_reuse;

DECLARE load_cursor CURSOR FOR
    SELECT l_year_month, COUNT(*)
           FROM li_reuse
        GROUP BY l_year_month;
LOAD FROM load_cursor OF CURSOR MESSAGES load.msg
    REPLACE INTO qm_reuse;

ALTER TABLE lineitem ATTACH PARTITION STARTING '1/1/1994'
    ENDING '1/31/1994' FROM li_reuse;

SET INTEGRITY FOR lineitem ALLOW WRITE ACCESS IMMEDIATE CHECKED
FOR EXCEPTION IN lineitem USE lineitem_ex;

ALTER TABLE quan_by_month ATTACH PARTITION STARTING 199401
    ENDING 199401 FROM qm_reuse;

SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED
    FOR EXCEPTION IN quan_by_month USE quan_by_month_ex;

ALTER TABLE quan_by_month ADD MATERIALIZED QUERY
    (SELECT l_year_month AS q_year_month, COUNT(*) AS q_count
          FROM lineitem
              GROUP BY l_year_month)
                  DATA INITIALLY DEFERRED REFRESH IMMEDIATE;

SET INTEGRITY FOR QUAN_BY_MONTH ALL IMMEDIATE UNCHECKED;
```

使用带有 IMMEDIATE CHECKED 选项的 SET INTEGRITY 语句，以检查已连接的数据分区是否存在完整性违例。在将表更改回 MQT 之前，需要执行此步骤。使用带有 IMMEDIATE UNCHECKED 选项的 SET INTEGRITY 语句来绕过对 MQT 进行必需的完全刷新。要获得最佳性能，MQT 的索引是必需的。建议您在适当的时候将异常表与 SET INTEGRITY 语句配合使用。

通常，在同样进行分区了的大型事实表上创建已分区的 MQT。如果在大型事实表上转入或转出表数据，那么必须手动调整已分区的 MQT，如示例 2 中所示。

示例 2: 手动调整已分区的 MQT

改变 MQT (quan_by_month) 以将它转换为一个普通的分区表:

```
ALTER TABLE quan_by_month DROP MATERIALIZED QUERY;
```

从事实表 (lineitem) 和 MQT 中拆离要转出的数据，并使用要转入的新数据重新装入登台表 li_reuse:

```
ALTER TABLE lineitem DETACH PARTITION part0 INTO li_reuse;

LOAD FROM part_mqt_rotate.del OF DEL MESSAGES load.msg REPLACE INTO li_reuse;

ALTER TABLE quan_by_month DETACH PARTITION part0 INTO qm_reuse;
```

进行插入之前修剪 qm_reuse。这将在插入子查询数据之前删除已拆离的数据。这是通过替换装入 MQT 来实现的，其中装入的数据文件是子查询的内容。

```
db2 load from datafile.del of del replace into qm_reuse
```

可以使用 INSERT INTO ... (SELECT ...) 来手动刷新表。这仅对新数据来说是必需的，因此应该在连接之前发出该语句:

```
INSERT INTO qm_reuse
    (SELECT COUNT(*) AS q_count, l_year_month AS q_year_month
              FROM li_reuse
                  GROUP BY l_year_month);
```

现在，可以转入事实表的新数据:

```
ALTER TABLE lineitem ATTACH PARTITION STARTING '1/1/1994'
ENDING '1/31/1994' FROM TABLE li_reuse;
SET INTEGRITY FOR lineitem ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR
EXCEPTION IN li_reuse USE li_reuse_ex;
```

接着，转入 MQT 的数据:

```
ALTER TABLE quan_by_month ATTACH PARTITION STARTING 199401
ENDING 199401 FROM TABLE qm_reuse;
SET INTEGRITY FOR quan_by_month IMMEDIATE CHECKED;
```

在连接数据分区之后，必须验证新数据以确保它在范围内。

```
ALTER TABLE quan_by_month ADD MATERIALIZED QUERY
    (SELECT COUNT(*) AS q_count, l_year_month AS q_year_month
          FROM lineitem
              GROUP BY l_year_month)
                  DATA INITIALLY DEFERRED REFRESH IMMEDIATE;
SET INTEGRITY FOR QUAN_BY_MONTH ALL IMMEDIATE UNCHECKED;
```

直到 SET INTEGRITY 语句验证了这些数据之后，才能对它们进行访问。虽然支持 REFRESH TABLE 操作，但此情况演示了通过 ATTACH PARTITION 和 DETACH PARTITION 操作对已分区的 MQT 进行手动维护。用户通过 SET INTEGRITY 语句的 IMMEDIATE UNCHECKED 子句将数据标记为已验证。

创建范围集群表

关于使用范围集群表的准则

本主题列示了处理范围集群表 (RCT) 时要遵循的一些准则。

- 因为创建范围集群表的过程中将预分配必需的磁盘空间，所以该空间必须可用。
- 定义键值范围时，最小值是可选的；如果未指定最小值，那么缺省值是 1。必须显式指定负数最小值 例如:

  ```
  ORGANIZE BY KEY SEQUENCE (f1 STARTING FROM -100 ENDING AT -10)
  ```

- 不能对用来定义范围集群表的相同键值创建常规索引。
- 不允许使用会影响表的物理结构的 ALTER TABLE 语句选项。

方案: 范围集群表

范围集群表可以具有单列或多列键，并且可以允许或不允许行具有在所定义的值范围之外的键值。本节包含一些方案，这些方案说明了如何才能创建这样的表。

方案 1: 创建范围集群表（允许溢出）

以下示例显示了一个范围集群表，可以使用该表来检索特定学生的信息。每个学生记录都包含下列信息:

- 学校标识
- 程序标识
- 学生编号
- 学生标识
- 学生的名
- 学生的姓
- 学生的学年平均成绩（GPA）

```
CREATE TABLE students (
 school_id      INT NOT NULL,
 program_id     INT NOT NULL,
 student_num    INT NOT NULL,
 student_id     INT NOT NULL,
 first_name     CHAR(30),
 last_name      CHAR(30),
 gpa            FLOAT
)
 ORGANIZE BY KEY SEQUENCE
  (student_id STARTING FROM 1 ENDING AT 1000000)
  ALLOW OVERFLOW
;
```

在此示例中，使用了 STUDENT_ID 列（它充当表键）来添加、更新或删除学生记录。

每条记录的大小基于列长度之和。在此示例中，每条记录的长度为 97 个字节（10 个字节的标题 + 4 + 4 + 4 + 4 + 30 + 30 + 8 + 3 个用于可空列的字节）。使用 4 KB（或 4096 个字节）页大小时，在除去开销之后，每页有 4038 字节（足够容纳 41 条记录）可用。要容纳一百万条学生记录，总共需要 24391 个这样的页。假定四页用于表开销和三页用于扩展数据块映射，那么创建此表时将预分配 24384 个 4 KB 页。（扩展数据块映射假定使用单个三页容器来存储此表。）

方案 2: 创建范围集群表（不允许溢出）

在以下示例中，教育局管理 200 所学校，每所学校有 20 间教室，每间教室可容纳 35 个学生。此教育局最多可以招收 140,000 名学生。

```
CREATE TABLE students (
 school_id      INT NOT NULL,
 class_id       INT NOT NULL,
 student_num    INT NOT NULL,
 student_id     INT NOT NULL,
 first_name     CHAR(30),
 last_name      CHAR(30),
 gpa            FLOAT
)
 ORGANIZE BY KEY SEQUENCE
  (school_id STARTING FROM 1 ENDING AT 200,
   class_id STARTING FROM 1 ENDING AT 20,
   student_num STARTING FROM 1 ENDING AT 35)
  DISALLOW OVERFLOW
;
```

在此示例中，SCHOOL_ID、CLASS_ID 和 STUDENT_NUM 列共同充当表键，将使用该表键来添加、更新或删除学生记录。

不允许溢出，这是因为教育局策略会限制每间教室中的学生数，并且此教育局管理的学校和教室的数目是固定的。对于某些较小的学校（与最大的学校相比教室数更少的学校），此表中将存在可能永远不会使用的预分配空间。

创建 MDC 或 ITC 表时的注意事项

创建 MDC 或 ITC 表时要考虑许多因素。在决定如何创建、布置和使用 MDC 或 ITC 表时，会受到当前数据库环境（例如，是否具有分区数据库）和所选择的维的影响。

将数据从现有表移至 MDC 表

要在数据仓库或大型数据库环境中提高查询性能和降低对数据维护操作的需求，可以将数据从常规表移至多维集群 (MDC) 表。要将数据从现有表移至 MDC 表:

1. 导出数据,
2. 删除原始表（可选），
3. 创建多维集群 (MDC) 表（使用带 ORGANIZE BY DIMENSIONS 子句的 CREATE TABLE 语句），
4. 将数据装入 MDC 表。

可以使用称为 SYSPROC.ALTOBJ 的 ALTER TABLE 过程来将现有表中的数据转换为 MDC 表中的数据。可以从“DB2 设计顾问程序”中调用此过程。在这两个表之间转换数据所需要的时间可能很长，这取决于表的大小以及需要转换的数据量。

ALTOBJ 过程在改变表时将执行下列步骤:

1. 删除表的所有从属对象,
2. 重命名表,
3. 使用新定义来创建表,
4. 重新创建表的所有从属对象,
5. 将表中的现有数据变换为新表中所需要的数据。即，从旧表中选择数据，然后将该数据装入新表中，可以在新表中使用列函数来将旧的数据类型变换为新的数据类型。

将数据从现有表移至 ITC 表

要降低对数据维护操作的需求，可以将数据从常规表移至插入时间集群 (ITC) 表。要将数据从现有表移至 ITC 表，请使用联机表移动存储过程。

ExampleBank 方案显示了如何将现有表中的数据移至 ITC 表。此方案还说明使用 ITC 表时回收空间的便利程度。有关更多信息，请参阅“相关概念”链接。

DB2 设计顾问程序上的 MDC 顾问程序功能部件

DB2 设计顾问程序 (**db2advis**) 具有 MDC 功能部件。此功能部件建议用于 MDC 表中的集群维（包括基本列的粗糙度）以便提高工作负载性能。*粗糙度*这个术语表示用来减小集群维基数（单值的数目）的一个数学表达式。粗糙度的常见示例是粗糙度可为日期、日期所在的星期、日期所在的月份或一年中的季度。

使用“DB2 设计顾问程序”的 MDC 功能部件要求数据库中至少存在几个扩展数据块的数据。“DB2 设计顾问程序”使用数据来对数据密度和基数建立模型。

如果数据库的表中没有数据，那么“DB2 设计顾问程序”不会建议使用 MDC，即使该数据库包含空表，但有一组虚假的统计信息来表示它是一个已填充的数据库。

建议还标识了用来定义维的粗糙度的潜在生成列。建议中不包括可能的块大小。在为 MDC 表提供建议时，使用表空间的扩展数据块大小。假定将在现有表所在的表空间中创建建议的 MDC 表，因此该表具有相同的扩展数据块大小。对 MDC 维的建议将根据表空间的扩展数据块大小的变化而变化，这是因为扩展数据块大小会影响可以填充到块或单元中的记录数。此扩展数据块大小将直接影响单元的密度。

尽管可以为表建议单个维或多个维，但请只考虑单列维而不要考虑组合列维。MDC 功能部件将建议大多数受支持的数据类型使用的粗糙度，目标是减小所采用的 MDC 解决方案中的单元的基数。异常的数据类型包括：CHAR、VARCHAR、GRAPHIC 和 VARGRAPHIC 数据类型。所有受支持的数据类型都将被强制类型转换为 INTEGER 并通过生成的表达式来设置粗糙度。

“DB2 设计顾问程序”的 MDC 功能部件的目标是选择可提高性能的 MDC 解决方案。另一个目标是将数据库的存储扩展限制在适当的级别。使用统计方法来确定每个表的最大存储扩展。

顾问程序中的分析操作既会利用块索引访问的优点，也会受到 MDC 对表的维执行插入、更新和删除操作的影响。对表执行这些操作时可能会导致在各个单元之间移动记录。分析操作还会模拟对特定 MDC 维上的数据进行组织时产生的任何表扩充而对性能造成的潜在影响。

通过对 db2advis 实用程序使用 `-m <advise type>` 标志来运行 MDC 功能部件。使用“C”建议类型来指示多维集群表。建议类型为：“I”表示索引、“M”表示具体化查询表、“C”表示 MDC，而“P”表示分区数据库环境。建议类型可以相互组合使用。

注： “DB2 设计顾问程序”不会处理其大小不到 12 个扩展数据块的表。

当提出建议时，顾问程序将同时分析 MQT 和常规基本表。

MDC 功能部件的输出包括:

- 每个表的生成列表达式（用于对 MDC 解决方案中出现的维设置粗糙度）。
- 对每个表建议的 ORGANIZE BY DIMENSIONS 子句。

对标准输出和作为说明工具的一部分的 ADVISE 表报告了建议。

MDC 表和分区数据库环境

多维集群可用于分区数据库环境中。实际上，MDC 可以作为分区数据库环境的补充。分区数据库环境用来将一个表中的数据分配到多个物理数据库分区或逻辑数据库分区上，以便达到下列目的:

- 利用多台机器来并行增加并行处理请求,
- 将表的物理大小增大到超过单个数据库分区的限制,
- 提高数据库的可伸缩性。

分发表的原因与该表是 MDC 表还是常规表无关。例如，选择用来组成分布键的列的规则是相同的。MDC 表的分布键可以包括任何列，无论这些列是否组成表的维的一部分。

如果分布键与表的某个维完全相同，那么每个数据库分区都包含该表的不同部分。例如，如果作为示例的 MDC 表按颜色分布在两个数据库分区上，那么将使用 Color 列来划分数据。因此，可能在一个数据库分区上找到 Red 和 Blue 片，而在另一个数据库分区上找到 Yellow 片。如果分布键与表中的维不完全相同，那么每个数据库分区都具有每个片的数据的子集。当选择维和估计单元占用率时，请注意，每个单元的平均数据总量是通过用所有数据除以数据库分区数来确定的。

具有多个维的 MDC 表

如果知道查询中大量使用了某些谓词，那么可以根据涉及到的列对表进行集群。可以使用 ORGANIZE BY DIMENSIONS 子句来执行此操作。

示例 1:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
      ORGANIZE BY DIMENSIONS (c1, c3, c4)
```

示例 1 中的表根据形成逻辑立方体（即，具有三个维）的三个列中的值集群。现在可以在查询处理期间根据一个或多个维对表进行逻辑分片，以便涉及的关系运算符仅处理相应的片或单元中的块。块的大小（页数）是表的扩展数据块大小。

具有基于多列的维的 MDC 表

每个维可以由一列或多列组成。作为一个示例，可以创建一个根据包含两列的一个维来集群的表。

示例 2:

```
CREATE TABLE T1 (c1 DATE, c2 INT, c3 INT, c4 DOUBLE)
      ORGANIZE BY DIMENSIONS (c1, (c3, c4))
```

在示例 2 中，表根据两个维 c1 和 (c3, c4) 来进行集群。这样，在查询处理期间，表可以根据 c1 维或组合 (c3, c4) 维逻辑分片。该表与示例 1 中的表具有相同数目的块，但是少一个维块索引。在示例 1 中，有三个维块索引，列 c1、c3 和 c4 各一个。在示例 2 中，有两个维块索引，一个针对列 c1，而另一个针对 c3 和 c4。这两个方法的主要差别在于，在示例 1 中，涉及 c4 的查询可以使用 c4 的维块索引来快速直接地访问相关数据块。在示例 2 中，c4 是维块索引中的辅助键部分，因此涉及 c4 的查询涉及更多的处理。但是，在示例 2 中，将少维护和存储一个块索引。

“DB2 设计顾问程序”将不对包含多列的维提供建议。

将列表达式作为维的 MDC 表

列表达式也可用于集群维。根据列表达式集群的功能对于将维上滚至更低的详细程度非常有用，例如，将地址上滚为地理位置或区域，或者将日期上滚为星期、月份或年。要以此方式实现维的上滚，可以使用生成列。此类型的列定义允许使用可以表示维的表达式来创建列。在示例 3 中，该语句创建根据一个基本列和两个列表达式进行集群的表。

示例 3:

```
CREATE TABLE T1(c1 DATE, c2 INT, c3 INT, c4 DOUBLE,
      c5 DOUBLE GENERATED ALWAYS AS (c3 + c4),
   c6 INT GENERATED ALWAYS AS (MONTH(C1)))
           ORGANIZE BY DIMENSIONS (c2, c5, c6)
```

在示例 3 中，c5 列是基于 c3 和 c4 列的表达式，而 c6 列会及时将 c1 列上滚至更低的详细程度。此语句根据 c2、c5 和 c6 列中的值来集群该表。

对生成列维的范围查询

对生成列维的范围查询需要单调列函数。表达式必须是单调的才能为生成列的维派生范围谓词。如果对生成列创建维，那么对基本列的查询能够利用生成列的块索引来提高性能（有一种情况例外）。要使基本列（例如，日期）的范围查询对维块索引使用范围扫描，用来在 CREATE TABLE 语句中生成列的表达式必须是单调的。尽管列表达式可以包括任何有效表达式（包括用户定义的函数 (UDF)），但是如果表达式不是单调的，那么当等价谓词或 IN 谓词都在基本列上时，它们才能够使用块索引来满足查询。

作为一个示例，假定使用生成列 month 的维来创建 MDC 表，其中 `month = INTEGER (date)/100`。对于该维 (month) 的查询，可以执行块索引扫描。对于基本列 (date) 的查询，也可以执行块索引扫描来缩小要扫描的块的范围，然后只将日期的谓词应用于这些块中的行。

编译器将生成要在块索引扫描中使用的其他谓词。例如，对于以下查询:

```
SELECT * FROM MDCTABLE WHERE DATE > "1999-03-03" AND DATE < "2000-01-15"
```

编译器将生成以下谓词: 『month >= 199903』 和『month <= 200001』，它们可以用作维块索引扫描的谓词。当对获得的块进行扫描时，会将原始谓词应用于这些块中的行。

非单调表达式允许对该维应用等价谓词。非单调函数的一个较好的示例是 MONTH( )，如示例 3 中的 c6 列的定义所示。如果 c1 列是日期、时间戳记或日期或时间戳记的有效字符串表示法，那么函数将返回范围是 1 到 12 的整数值。尽管函数的输出是确定的，但是实际上它生成的输出与阶跃函数（即，循环模式）相似:

```
MONTH(date('01/05/1999')) = 1
MONTH(date('02/08/1999')) = 2
MONTH(date('03/24/1999')) = 3
MONTH(date('04/30/1999')) = 4
...
MONTH(date('12/09/1999')) = 12
MONTH(date('01/18/2000')) = 1
MONTH(date('02/24/2000')) = 2
...
```

尽管此示例中的日期是连续增加的，但是 MONTH(date) 不会增加。更具体而言，每当 date1 大于 date2，并不能保证 MONTH(date1) 大于或等于 MONTH(date2)。这是单调性所要求的。此非单调性是允许的，但是它限制了维，基本列的范围谓词不能生成维的范围谓词。但是，表达式的范围谓词是可以的，例如，`where month(c1) between 4 and 6`。这可以采用常规方式使用维的索引，起始键为 4 而停止键为 6。

要使此函数单调，请将年包括为月份的高位部分。存在对 INTEGER 内置函数的扩展以帮助根据日期定义单调表达式。INTEGER(date) 返回日期的整数表示法，可以分开查找年和月份的整数表示法。例如，`INTEGER(date('2000/05/24'))` 返回 `20000524`，因此 `INTEGER(date('2000/05/24'))/100 = 200005`。函数 INTEGER(date)/100 是单调的。

相似的，内置函数 DECIMAL 和 BIGINT 也具有扩展，所以可以派生单调函数。DECIMAL(timestamp) 返回时间戳记的十进制表示法，可以在单调表达式中使用它来派生月份、天、小时或分钟等等的增加的值。BIGINT(date) 返回日期的大整数表示法，类似于 INTEGER(date)。

只要可能，数据库管理器将在为表创建生成列或者根据维子句中的表达式创建维时确定表达式的单调性。特定函数被识别为保留单调性，例如，DATENUM( )、DAYS( ) 和 YEAR( )。并且，列和常量的各种算术表达式，例如，除法、乘法或加法是保留单调性的。当 DB2 确定表达式不保留单调性时，或者如果它不能确定这一点，那么该维仅支持对其基本列使用等价谓词。

第 12 章 改变数据库

改变实例

更改多个数据库分区中的数据库配置

当数据库分布在多个数据库分区上时，数据库配置文件在所有数据库分区上应该相同。

关于此任务

一致性是必需的，因为 SQL 编译器根据数据库分区配置文件中的信息来编译分布式 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。使用 **db2_all** 来跨所有数据库分区维护配置文件。

改变数据库

第 13 章 改变表和其他相关表对象

改变分区表

分区表支持 ALTER TABLE 语句的所有相关子句。此外，ALTER TABLE 语句允许添加新数据分区、转入（*连接*）新数据分区和转出（*拆离*）现有数据分区。

开始之前

要改变分区表以拆离数据分区，用户必须拥有下列权限或特权:

- 执行 DETACH PARTITION 操作的用户必须有权对源表执行 ALTER、SELECT 和 DELETE 操作。
- 该用户还必须有权创建目标表。因此，要改变一个表以拆离数据分区，语句授权标识拥有的特权必须至少包括对目标表的下列其中一项权限或特权:
  - DBADM 权限
  - 对数据库的 CREATETAB 权限、对该表所使用表空间的 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）

要改变分区表以连接数据分区，语句授权标识拥有的特权必须至少包括对源表的下列其中一项权限或特权:

- 对源表的 DATAACCESS 权限或 SELECT 特权以及对源表模式的 DBADM 权限或 DROPIN 特权
- 对源表的 CONTROL 特权

要改变分区表以添加数据分区，语句授权标识必须有权使用要添加新分区的表空间，并且其拥有的特权必须至少包括对源表的下列其中一项权限或特权:

- ALTER 特权
- CONTROL 特权
- DBADM
- 对表模式的 ALTERIN 特权

关于此任务

- 在指定了 PARTITION 子句情况下发出的每个 ALTER TABLE 语句都必须在单独的 SQL 语句中。
- 在包含 ALTER TABLE...PARTITION 操作的 SQL 语句中，不允许执行任何其他 ALTER 操作。例如，在单个 SQL 语句中，不能同时连接数据分区和对表添加列。
- 可以执行多个 ALTER 语句，接着执行一个 SET INTEGRITY 语句。

过程

要使用命令行来改变分区表，请发出 ALTER TABLE 语句。

有关改变分区表的准则和限制

本主题标识了在存在已连接和已拆离的数据分区的情况下最常用的改变表操作和注意事项。

SYSCAT.DATAPARTITIONS 目录视图的 STATUS 列包含表的分区的状态信息。

- 当 STATUS 为空字符串时，表示分区可视且处于正常状态。
- 当 STATUS 为“A”时，表示分区为新连接的分区，必须发出 SET INTEGRITY 语句以将连接的分区转为正常状态。
- 当 STATUS 为“D”、“L”或“I”时，表示正在连接分区，但是连接操作尚未完成。
  - 对于处于“D”状态的分区，必须对所有已拆离的从属表发出 SET INTEGRITY 语句，以将分区转换为在逻辑上已拆离状态。
  - 对于处于“L”状态的分区，该分区为在逻辑上已拆离的分区且异步分区拆离任务正在为 DB2 V9.7 FP1 及更高发行版完成该分区的拆离。
  - 对于处于“I”状态的分区，异步分区拆离任务已完成，异步索引清除正在更新该分区上定义的非分区索引。

添加或改变约束

已连接和已拆离的数据分区支持添加检查约束或外键约束。当分区表带有处于状态“D”或“L”的已拆离分区时，如果系统必须生成新的分区索引才能强制执行约束，那么添加主约束或唯一约束将返回错误。对于处于“L”状态的分区，操作将返回 SQL20285N（SQLSTATE 55057）。对于处于“D”状态的分区，操作将返回 SQL20054（SQLSTATE 55019）。

添加列 在对带有已连接数据分区的表添加列时，还将对已连接数据分区添加该列。由于已拆离数据分区在物理上不再与该表相关联，所以在对带有处于“L”状态的已拆离数据分区的表添加列时，不会对已拆离数据分区添加该列。

对于处于“L”或“D”状态的已拆离分区，操作将失败并返回错误。 对于处于“L”状态的分区，操作将返回 SQL20285N（SQLSTATE 55057）。对于处于“D”状态的分区，操作将返回 SQL20296N（SQLSTATE 55057）。

改变列 当改变带有已连接数据分区的表中的列时，还将对已连接数据分区改变该列。当改变带有已拆离数据分区的表中的列时，由于已拆离数据分区在物理上不再与该表相关联，不会改变已拆离数据分区上的该列。

当删除或重命名带有处于“L”或“D”状态的分区的列时，操作将失败并将返回错误。对于处于“L”状态的分区，操作将返回 SQL20285N（SQLSTATE 55057）。对于处于“D”状态的分区，操作将返回 SQL0270N（SQLSTATE 42997）。

添加生成列

当将生成列添加到带有已连接或已拆离数据分区的分区表时，必须遵循任何其他列类型的添加规则。

添加或修改非分区索引

对带有已连接数据分区的表创建、重新创建或重组索引时，该索引不包括已连接数据分区中的数据，这是因为 SET INTEGRITY 语句维护所有已连接数据分区的所有索引。对带有已拆离数据分区的表创建、重新创建或重组索引时，除非已拆离数据分区带有需要根据该数据分区（该分区处于“D”状态）进行递增刷新的已拆离从属表或登台表，否则索引不包括已拆离数据分区中的数据。在后面这种情况下，索引将包括此已拆离数据分区的数据。

添加或修改分区索引

当在有已连接数据分区的情况下创建分区索引时，将创建每个已连接数据分区的索引分区。直到运行了 SET INTEGRITY 语句以使连接的数据分区联机，所连接数据分区上索引分区的索引条目才可视。注意，因为创建索引涉及已连接数据分区，所以创建唯一分区索引时可找到已连接数据分区中作为重复键值的行，从而使索引创建失败。为了避免此问题，建议用户在有已连接分区的情况下不要尝试创建分区索引。

如果表具有任何已拆离从属表，那么不支持在带有已拆离从属表的分区表上创建分区索引。在此情况下，进行创建分区索引的任何尝试都将导致 SQLSTATE 55019。如果为具有处于"L"状态分区的表创建分区索引，那么操作将返回 SQL20285N（SQLSTATE 55057）。

WITH EMPTY TABLE

不能清空带有已连接数据分区的表。

ADD MATERIALIZED QUERY AS

不允许将带有已连接数据分区的表改变为 MQT。

改变数据分区中存储的其他表属性

在数据分区中还存储了下列表属性。对这些属性所作的更改将反映到已连接数据分区中，但不会反映到已拆离数据分区中。

- DATA CAPTURE
- VALUE COMPRESSION
- APPEND
- COMPACT/LOGGED FOR LOB COLUMNS

在同一事务中创建和访问数据分区

如果某个表具有非分区索引，那么您将无法访问该表中与创建了该分区的添加或连接操作位于同一事务中的新数据分区，条件是该事务未以互斥方式锁定此表（SQL0668N，原因码为 11）。

改变表以添加 (ADD)，连接 (ATTACH) 或拆离 (DETACH) 分区时针对 XML 索引的特殊注意事项

与非分区关系索引类似，基于 XML 列的非分区索引是在分区表的所有数据分区之间共享的独立对象。通过添加、连接或拆离分区改变表时，XML 区域索引和列路径索引会受到影响。基于 XML 列路径的索引始终是非分区索引，而基于 XML 数据的索引在缺省情况下生成为分区索引。

XML 区域索引

ADD PARTITION 将为要添加的新空数据分区创建新的区域索引分区。区域索引分区的新条目将被添加至 SYSINDEXPARTITIONS 表。新分区上分区索引对象的表空间将由 ADD PARTITION 子句中的 INDEX IN <table space> 确定。如果没有为 ADD PARTITION 子句指定任何 INDEX IN <table space>，那么分区索引对象的表空间将与缺省情况下对应数据分区使用的表空间相同。

分区表上由系统生成的 XML 区域索引始终为分区索引。分区索引使用索引组织方案，即，索引数据根据表的表分区方案划分到多个存储对象（称为索引分区）中。每个索引分区都只引用相应数据分区中的表行。

对于 ATTACH 操作，因为带有 XML 列的分区表上区域索引始终为分区索引，所以在完成 ATTACH 操作之后，可以将源表上的区域索引保留为新表分区的新区域索引。数据和索引对象不会移动，因此，需要更新目录表条目。在 ATTACH 操作时将除去源表上区域索引的目录表条目，并且将在 SYSINDEXPARTITIONS 表中添加一个区域索引分区。池标识和对象标识将保留不变，与它们在源表上时的情况相同。索引标识 (IID) 将被修改为与目标上区域索引的标识匹配。

在完成 DETACH 操作之后，会将区域索引保留在已拆离的表上。将从 SYSINDEXPARTITIONS 表中除去与要拆离的分区相关联的索引分区条目。将在已拆离表的 SYSINDEXES 目录表中添加一个新的区域索引条目，它将与 DETACH 操作之前的区域索引分区具有相同的池标识和对象标识。

基于 XML 数据的索引

从 DB2 V9.7 修订包 1 开始，可为分区表将基于 XML 数据的索引创建为分区索引或非分区索引。缺省值是分区索引。

在 ATTACH 和 DETACH 操作期间，基于 XML 数据的分区索引和非分区索引会被视为任何其他关系索引。

在 ATTACH 操作期间，将删除源表上的索引。这适用于逻辑和物理 XML 索引。在 ATTACH 操作期间，将除去它们在系统目录中的条目。

在 ATTACH 操作之后，必须运行 Set integrity，才能维护目标表上基于 XML 数据的非分区索引。

对于 DETACH 操作，源表上基于 XML 列的非分区索引不由目标表继承。

XML 列路径索引

基于 XML 列路径的索引始终是非分区索引。在转入和转出操作期间，将维护源表和目标表上的 XML 列路径索引。

对于 ATTACH 操作，DB2 数据库管理器将维护目标表上的非分区 XML 列路径索引（这与其他非分区索引不同，其他非分区索引是在完成 ATTACH 操作之后在 SET INTEGRITY 期间进行维护）。以后，将删除源表上的 XML 列路径索引并且将除去它们的目录条目，因为目标表上的列路径索引为非分区索引。

对于转出，请记住 XML 列路径索引为非分区索引，而非分区索引不会一起转移到独立目标表中。但是，在带有 XML 列的表可供外部用户访问之前，该表上必须存在 XML 列路径索引（对于每列都有一个索引），因此，必须在目标表上创建 XML 列路径索引才能使用该目标表。将创建列路径索引的时间取决于在 DETACH 操作期间是否存在任何已拆离的从属表。如果不存在任何已拆离的从属表，那么在 DETACH 操作期间将创建路径索引，否则，SET INTEGRITY 或 MQT 刷新将创建路径索引以维护拆离从属对象。

在 DETACH 操作之后，目标表上创建的 XML 列路径索引将随该表上的所有其他索引保留在同一索引对象中。

连接数据分区

表分区功能提高了表数据的转入和转出效率。通过带 ATTACH PARTITION 子句的 ALTER TABLE 语句，可以更容易地进行数据转入。

开始之前

如果在执行连接操作之前可以通过独立于数据服务器的应用程序逻辑执行数据完整性检查（包括范围验证和其他约束检查），那么可以更快地使新连接的数据可供使用。通过使用 SET INTEGRITY...ALL IMMEDIATE UNCHECKED 语句来跳过范围检查和约束违例检查，可以优化数据转入过程。在此情况下，表将脱离 SET INTEGRITY 暂挂状态，并且只要目标表的所有用户索引是分区索引，新数据就可以立即供应用程序使用。

如果在执行连接操作之后，表中存在要维护的非分区索引（XML 列路径索引除外），那么 SET INTEGRITY...ALL IMMEDIATE UNCHECKED 语句的行为就好像它是 SET INTEGRITY...IMMEDIATE CHECKED 语句一样。将执行所有完整性处理、非分区索引维护和表状态过渡，就好像发出了 SET INTEGRITY...IMMEDIATE CHECKED 语句一样。此行为确保使用 SET INTEGRITY...ALL IMMEDIATE UNCHECKED 的转入脚本在交付使用之后，不会因为有时为目标表创建了非分区索引而停止工作。

要改变表以连接数据分区，语句授权标识拥有的特权必须至少包括对源表的下列其中一项权限或特权:

- 对表的 SELECT 特权以及对表的模式的 DROPIN 特权
- 对表的 CONTROL 特权
- DATAACCESS 权限

关于此任务

连接数据分区时，将使用现有表（源表）并将其作为新数据分区连接至目标表。使用带有 ATTACH PARTITION 子句的 ALTER TABLE 语句将数据分区连接到分区表时，目标分区表仍然处于联机状态，并且在 RS、CS 或 UR 隔离级别下运行的针对该表的动态查询将继续运行。

限制和用法准则

要连接数据分区，必须符合下列条件:

- 要连接新数据分区的目标表必须是现有的分区表。
- 源表必须是现有的非分区表，或者是带有单个数据分区且没有连接数据分区或拆离数据分区的分区表。要连接多个数据分区，必须发出多个 ATTACH 语句。
- 源表不能为类型表。
- 源表不能为范围集群表。
- 源表和目标表定义必须匹配。
- 源列和目标列在数目、类型和顺序方面必须匹配。
- 源表列和目标表列在是否包含缺省值方面必须匹配。
- 源表列和目标表列在是否允许空值方面必须匹配。
- 源表和目标表的压缩指定内容（包括 VALUE COMPRESSION 和 COMPRESS SYSTEM DEFAULT 子句）必须匹配。

- 源表和目标表的 DATA CAPTURE、ACTIVATE NOT LOGGED INITIALLY 和 APPEND 选项的指定内容必须匹配。
- 即使在目标列是生成列而相应源列不是生成列时，也允许连接数据分区。以下语句为连接行的生成列生成值:

  ```
  SET INTEGRITY FOR table-name
    ALLOW WRITE ACCESS  IMMEDIATE CHECKED FORCE GENERATED
  ```

  与生成列匹配的源表列在类型和可空性方面必须匹配；但是，不要求缺省值也匹配。建议的方法是保证连接操作的源表在生成列中包含正确的生成值。如果遵循建议的方法，那么不要求您使用 FORCE GENERATED 选项，并可使用以下语句。

  ```
  SET INTEGRITY FOR table-name
    GENERATED COLUMN  IMMEDIATE UNCHECKED
  ```

  此语句指示即将绕过对生成的列的检查。

  ```
  SET INTEGRITY FOR table-name
    ALLOW WRITE ACCESS  IMMEDIATE CHECKED
    FOR EXCEPTION IN table-name USE table-name
  ```

  此语句对连接的数据分区执行完整性检查，但是不检查生成列。
- 即使在目标列是标识列而源列不是标识列时，也允许连接数据分区。语句 SET INTEGRITY IMMEDIATE CHECKED 不为连接的行生成标识值。语句 SET INTEGRITY FOR T GENERATE IDENTITY ALLOW WRITE ACCESS IMMEDIATE CHECKED 填写所连接的行的标识值。与标识列匹配的列在类型和可空性方面必须匹配。对此列的缺省值没有要求。建议的方法是在登台表中填写正确的标识值。于是，由于已在源表中保证了标识值，所以不需要在执行 ATTACH 后使用 GENERATE IDENTITY 选项。
- 对于数据分布在多个数据库分区中的表来说，源表必须也使用同一分布键和同一个分发映射分布在同一个数据库分区组中。
- 源表必须是可删除的（即，不能对其设置 RESTRICT DROP）。
- 如果指定数据分区名称，那么它不得存在于目标表中。
- 如果目标表是多维集群 (MDC) 表，那么源表也必须是 MDC 表。
- 使用非分区表时，源表的数据表空间在类型（即 DMS 或 SMS）、页大小、扩展数据块大小和数据库分区组方面必须与目标表的数据表空间匹配。如果预取大小不匹配，那么会返回警告。源表的索引表空间在类型、数据库分区组、页大小和扩展数据块大小方面必须与目标表的分区索引使用的索引表空间匹配。源表的大型表空间在类型、数据库分区组和页大小方面必须与目标表的大型表空间匹配。使用分区表时，源表的数据表空间在类型、页大小、扩展数据块大小和数据库分区组方面必须与目标表的数据表空间匹配。
- 当对带有任何结构化列、XML 列或 LOB 列的分区表发出 ALTER TABLE ATTACH 语句时，源表上任何结构化列、XML 列或 LOB 列的 INLINE LENGTH 必须与目标表上对应结构化列、XML 列或 LOB 列的 INLINE LENGTH 匹配。
- 当将 REQUIRE MATCHING INDEXES 子句与 ATTACH PARTITION 子句配合使用时，如果目标表上出现任何在源表上没有匹配项的分区索引，那么将返回 SQL20307N。
- 如果源表对于目标表上每个分区唯一索引没有匹配的索引，那么与该源表进行连接将导致连接操作失败并且错误为 SQL20307N，原因码为 17。

- 对于分区索引，使用延迟索引清除机制的 MDC 转出不受支持。当表具有正在进行的延迟索引清除操作（作为 MDC 转出的结果）时，如果连接操作会将任何受延迟索引清除操作影响的 RID 索引保留在源表上，那么不允许执行连接操作。将在连接操作期间重新构建（而非保留）的索引不受此限制影响。
- 如果源表与目标表使用不同的 XML 数据格式，那么将不支持连接该源表。
- 如果表包含了使用 V9.5 或更低版本的 XML 记录格式的 XML 列，那么不支持将该表连接至某个分区表，该分区表包含了使用 V 9.7 或更高版本的记录格式的 XML 列。

  在连接该表之前，必须将它的 XML 记录格式更新为与目标分区表的记录格式匹配。以下两种方法中的任何一种都会更新表的 XML 记录格式:
  - 使用 ADMIN_MOVE_TABLE 过程来对该表执行联机表移动。
  - 执行下列步骤:
    1. 使用 EXPORT 命令来创建表数据的副本。
    2. 使用 TRUNCATE 语句来从该表中删除所有行并释放已分配给该表的存储器。
    3. 使用 LOAD 命令来将数据添加到该表中。

  在已更新该表的 XML 记录格式之后，将该表连接至目标分区表。
- 如果某个表具有非分区索引，那么您将无法访问该表中与创建了该分区的添加或连接操作位于同一事务中的新数据分区，条件是该事务未以互斥方式锁定此表（SQL0668N，原因码为 11）。

在运行连接操作之前，在源表上创建与目标表中各个分区索引匹配的索引。与分区索引匹配会使转入操作更为高效并且需要较少活动日志空间。如果源表上的索引未正确准备，那么数据库管理器将需要为您维护这些索引。为了确保转入操作不会导致维护这些分区索引的任何附加成本，可以在连接分区操作上指定 REQUIRE MATCHING INDEXES。指定 REQUIRE MATCHING INDEXES 将确保在源表没有与目标上分区索引匹配的索引的情况下连接操作失败。然后，可使用更正操作并重新发出连接操作。

此外，在运行连接操作之前删除源表上任何多余的索引。多余的索引是源表上满足以下条件的索引: 在目标表上没有匹配项或与目标表上非分区索引匹配。在运行连接操作之前删除多余的索引会使该操作的运行速度更快。

例如，假定有一个称为 ORDERS 的分区表，它具有 12 个数据分区（对于年度的每个月份都有一个数据分区）。在每个月份结束时，有一个称为 NEWORDERS 的单独表连接至分区表 ORDERS。

1. 在 ORDERS 表上创建分区索引。

   ```
   CREATE INDEX idx_delivery_date ON orders(delivery) PARTITIONED
   CREATE INDEX idx_order_price ON orders(price) PARTITIONED
   ```

2. 通过在表 NEWORDERS 上创建对应索引来准备连接操作。

   ```
   CREATE INDEX idx_delivery_date_for_attach ON neworders(delivery)
   CREATE INDEX idx_order_price_for_attach ON neworders(price)
   ```

3. 对于连接操作，存在两个步骤:
   a. ATTACH。在表 NEWORDERS 上，保留与表 ORDERS 上分区索引匹配的索引。

      ```
      ALTER TABLE orders ATTACH PARTITION part_jan2009 STARTING FROM ('01/01/2009')
         ENDING AT ('01/31/2009') FROM TABLE neworders
      ```

表 ORDERS 会自动置于 Set Integrity Pending 状态。在连接操作完成之后，idx_delivery_date_for_attach 索引和 idx_order_price_for _attach 索引成为表 ORDERS 的一部分。在此操作期间，不会发生数据移动。

b. SET INTEGRITY。在新连接的分区上完成范围检查。会强制执行存在的任何约束。完成后，新连接的数据便在数据库中可视。

```
SET INTEGRITY FOR orders IMMEDIATE CHECKED
```

当目标表上存在非分区索引时，INTEGRITY 语句必须保留索引及其他任务，例如，针对来自新连接分区的数据执行范围验证和约束检查。非分区索引维护需要大量活动日志空间，该空间与新连接分区中的数据量、每个非分区索引的键大小以及非分区索引的数目成比例。

将为新数据分区上的每个分区索引提供使用源表中的表空间标识和对象标识的 SYSINDEXPARTITIONS 目录表。标识信息来源于 SYSINDEXES 表（对于非分区表）或 SYSINDEXPARTITIONS 表（对于分区表）。索引标识来源于匹配的目标表的分区索引。

当源表为分区表时，源表上与目标表上分区索引匹配的那些分区索引会保留为连接操作的一部分。SYSINDEXPARTITIONS 表中的索引分区条目会更新，以表明它们是新目标表上带有新索引标识的索引分区。

当连接数据分区时，索引的一些统计信息以及数据会被从源表转移到新分区的目标表中。具体地说，目标上新分区的 SYSDATAPARTITIONS 和 SYSINDEXPARTITIONS 表中所有字段都由源中的信息填充。当源表为非分区表时，这些统计信息来自 SYSTABLES 和 SYSINDEXES 表。当源表为单一分区分区表时，这些统计信息来自该单一源分区的 SYSDATAPARTITIONS 和 SYSINDEXPARTITIONS 表。

注：在连接操作完成后执行运行统计操作，因为转移的统计信息将不会影响 SYSINDEXES 和 SYSTABLES 表中聚集的统计信息。

SET INTEGRITY...ALL IMMEDIATE UNCHECKED 期间的非分区索引维护。对分区表发出 SET INTEGRITY...ALL IMMEDIATE UNCHECKED 以跳过对新连接的分区进行范围检查时，如果该表中存在任何非分区索引（XML 列路径索引除外），那么 SET INTEGRITY...ALL IMMEDIATE UNCHECKED 将具有下列行为:

- 如果 SET INTEGRITY...ALL IMMEDIATE UNCHECKED 语句引用了一个目标表，那么其行为就好像改为发出了 SET INTEGRITY...ALLOW WRITE ACCESS...IMMEDIATE CHECKED 语句一样。SET INTEGRITY...ALL IMMEDIATE UNCHECKED 语句维护所有非分区索引（XML 列路径索引除外）、执行所有其他完整性处理、更新 SYSCAT.TABLES 目录视图的 CONST_CHECKED 列中的约束检查标志值并且在检测到约束违例时返回错误并立即停止。
- 如果 SET INTEGRITY...ALL IMMEDIATE UNCHECKED 语句引用了多个目标表，那么将返回错误（SQL20209N，原因码为 13）。

在 SET INTEGRITY 期间重建无效分区索引。SET INTEGRITY 语句可以检测新连接的分区的分区索引对象是否无效，并在必要时执行分区索引重建。

将数据分区连接至分区表的准则

本主题提供了一些准则，可用于指导通过发出 ALTER TABLE ...ATTACH PARTITION 语句来更正尝试将数据分区连接至数据库时出现的各种类型的不匹配情况。通过将源表修改为与目标表特征匹配，或者通过将目标表修改为与源表特征匹配，可以使这两个表匹配。

源表是想要连接至目标表的现有表。目标表是想要将新数据分区连接至的表。

要成功执行连接，一种建议的方法是对源表使用与目标表完全相同的 CREATE TABLE 语句，但不带 PARTITION BY 子句。在难以修改源表或目标表特征以实现兼容的情况下，可以创建与目标表兼容的新源表。有关创建新源表的详细信息，请参阅“创建与现有表类似的表”。

为了帮助您避免出现不匹配情况，请参阅“连接数据分区”中的“限制和用法准则”一节。该节概述了成功连接数据分区之前必须满足的条件。如果无法满足所列示的条件，就会返回错误 SQL20408N 或 SQL20307N。

下列各节描述了可能出现的各种类型的不匹配情况，并提供了一些建议步骤来使两个表匹配:

（值）压缩子句（SYSCAT.TABLES 的 COMPRESSION 列）不匹配。（SQL20307N 原因码 2）

要使值压缩一致，请使用下列其中一个语句:

```
ALTER TABLE... ACTIVATE VALUE COMPRESSION
或者
ALTER TABLE... DEACTIVATE VALUE COMPRESSION
```

要使行压缩值匹配，请使用下列其中一个语句:

```
ALTER TABLE... COMPRESS YES
或者
ALTER TABLE... COMPRESS NO
```

表的 APPEND 方式不匹配。（SQL20307N 原因码 3）

要使追加方式匹配，请使用下列其中一个语句:

```
ALTER TABLE ... APPEND ON
或者
ALTER TABLE ... APPEND OFF
```

源表与目标表的代码页不匹配。（SQL20307N 原因码 4）

创建一个新的源表

源表是包含多个数据分区或包含已连接或已拆离数据分区的分区表。（SQL20307N 原因码 5）

使用以下语句来从源表中拆离数据分区，直到只剩下一个可视数据分区为止:

```
ALTER TABLE ... DETACH PARTITION
```

已拆离的分区保持拆离状态，直到完成以下所有步骤:

1. 执行任何必要的 SET INTEGRITY 语句以增量刷新已拆离的从属项。

2. 在 V9.7.1 及更高版本中，请等待拆离以异步方式完成。要提高此过程的速度，请确保拆离操作之前对该表进行的所有访问都已完成或终止。
3. 如果源表具有非分区的索引，请等待异步索引清除完成。要提高此过程的速度，可以选择删除源表上的非分区索引。

如果想要立即执行连接操作，可以选择创建新源表。

源表是系统表、视图、类型表、按键序列组织的表、已创建的临时表或已声明的临时表。（SQL20307N 原因码 6）

创建一个新的源表。

目标表与源表是同一个表。（SQL20307N 原因码 7）

不能将表连接到它自身。确定正确的表以用作源表或目标表。

对源表或目标表指定了 NOT LOGGED INITIALLY 子句，但未对两个表都指定此子句。（SQL20307N 原因码 8）

通过发出 COMMIT 语句，对最初未进行日志记录的表进行日志记录，或者通过输入以下语句来将进行了日志记录的表更改为最初不进行日志记录:

```
ALTER TABLE ... ACTIVATE NOT LOGGED INITIALLY
```

对源表或目标表指定了 DATA CAPTURE CHANGES 子句，但未对两个表都指定此子句。（SQL20307N 原因码 9）

要对未打开数据捕获更改功能的表启用数据捕获更改功能，请运行以下语句:

```
ALTER TABLE ... DATA CAPTURE CHANGES
```

要对已打开数据捕获更改功能的表禁用数据捕获更改功能，请运行以下语句:

```
ALTER TABLE ... DATA CAPTURE NONE
```

表的分发子句不匹配。源表与目标表的分布键必须相同。（SQL20307N 原因码 10）

建议您创建一个新的源表。对于跨多个数据库分区的表，不能更改其分布键。要对单一分区数据库中的表更改分布键，请运行下列语句:

```
ALTER TABLE ... DROP DISTRIBUTION;
       ALTER TABLE ... ADD DISTRIBUTION(key-specification)
```

连接操作期间缺少索引时会返回错误（SQL20307N 原因码 18）

连接操作会隐式地对源表构建与目标表上分区索引对应的所缺少索引。隐式创建这些所缺少索引确实要用一些时间来完成。为您提供了一个选项，以在连接操作遇到任何缺少索引的问题时创建错误条件。该选项称为 ERROR ON MISSING INDEXES，是连接操作选项之一。当发生此情况时返回的错误为 SQL20307N（SQLSTATE 428GE，原因码 18）。有关不匹配索引的信息放在管理日志中。

连接操作会删除源表上与目标表上的分区索引不匹配的索引。识别和删除这些不匹配索引要用一些时间来完成。在尝试连接操作之前，应该删除这些索引。

在连接操作期间，当目标表上不匹配索引是唯一索引，或者 XML 索引是使用 REJECT INVALID VALUES 子句定义的时，会返回错误（SQL20307N 原因码 17）

当不具有源表上任何匹配索引的目标表上存在分区索引并且未使用 ERROR ON MISSING INDEXES 时，可预计下列结果:

1. 如果目标表上不匹配索引是唯一索引，或者 XML 索引是使用 REJECT INVALID VALUES 子句定义的，那么连接操作将失败，并返回错误消息 SQL20307N（SQLSTATE 428GE，原因码 17）。
2. 如果目标表上不匹配索引不满足前一点的条件，那么在连接操作期间，源表上索引对象被标记为无效。连接操作成功完成，但是，新数据分区上索引对象被标记为无效。SET INTEGRITY 操作用来重建新连接的分区上的索引对象。通常，这是在连接数据分区之后将执行的下一操作。重新创建这些索引要用一些时间。

管理日志将包含有关源表与目标表上索引之间任何不匹配项的详细信息。

仅对其中一个表指定了 ORGANIZE BY DIMENSIONS 子句，或者组织维不同。（SQL20307N 原因码 11）

创建一个新的源表。

列的数据类型 (TYPENAME) 不匹配。（SQL20408N 原因码 1）

要更正数据类型的不匹配，可发出以下语句:

```
ALTER TABLE ... ALTER COLUMN ... SET DATA TYPE...
```

列的可空性 (NULLS) 不匹配。（SQL20408N 原因码 2）

要改变其中一个表中不匹配的列的可空性，发出下列语句:

```
ALTER TABLE... ALTER COLUMN...
                          DROP NOT NULL
或者
ALTER TABLE... ALTER COLUMN...
                          SET NOT NULL
```

列的隐式缺省值 (SYSCAT.COLUMNS IMPLICITVALUE) 不兼容。（SQL20408N 原因码 3）

创建一个新的源表。如果目标表列和源表列都具有隐式缺省值（且 IMPLICITVALUE 不为 NULL），那么这两个隐式缺省值必须完全匹配。

如果对于目标表中的一列，IMPLICITVALUE 不是 NULL，并且对于源表的相应列，IMPLICITVALUE 不是 NULL，那么会在对表执行原始的 CREATE TABLE 语句之后添加每一列。在此情况下，存储在 IMPLICITVALUE 中的值必须对于此列是相匹配的。

有一种情况是，通过从版本低于 V9.1 的表进行迁移或者从版本低于 V9.1 的表连接数据分区之后，IMPLICITVALUE 不是 NULL，因为系统不知道在执行原始 CREATE TABLE 语句之后是否添加了列。如果数据库无法确定该列是否是添加的列，那么将其视为如此。添加的列就是作为 ALTER TABLE ...ADD COLUMN 语句的结果创建的列。在此情况下，不允许执行该语句，因为如果允许继续进行连接，该列的值可能会被破坏。必须将源表中的数据复制到新表中（此列的 IMPLICITVALUE 为 NULL），并将新表用作连接操作的源表。

列的代码页 (COMPOSITE_CODEPAGE) 不匹配。（SQL20408N 原因码 4）

创建一个新的源表。

系统压缩缺省子句 (COMPRESS) 不匹配。（SQL20408N 原因码 5）

要改变列的系统压缩，发出下面的其中一个语句来更正不匹配的情况:

```
ALTER TABLE ... ALTER COLUMN ...
                COMPRESS SYSTEM DEFAULT
或者
ALTER TABLE ... ALTER COLUMN ...
                COMPRESS OFF
```

在 ATTACH PARTITION 期间源表索引与目标表分区索引匹配的条件

如果要将源表上的索引复用为目标表上的索引分区，那么源表上的索引的所有索引键列或表达式都必须与目标表上的分区索引的索引键列或表达式匹配。

如果源表具有任何基于表达式的索引，那么在执行 ATTACH 处理期间，将隐式删除与索引相关联的系统生成的统计信息视图和软件包。如果 ATTACH 处理的目标表具有基于表达式的分区索引，那么在确定源表是否具有匹配索引时，将使用该表达式。如果索引的所有其他属性都相同，那么源表上的索引将视为与目标表上的分区索引匹配。即，源表上的索引可以用作目标表上的索引。

下表仅在目标索引为分区索引时有用并且适用。在认为源索引是匹配项的所有情况下，目标索引属性都将采用源索引属性。

表 14. 确定目标索引属性不同于源索引属性时源索引是否匹配。

规则编号	目标索引属性	源索引属性	源索引匹配吗?
1.	非唯一	唯一	是，如果索引并非 XML 索引。
2.	唯一	非唯一	否
3.	列 X 为降序	列 X 为升序	否
4	列 X 为升序	列 X 为降序	否
5.	列 X 为随机顺序	列 X 为升序	否
6.	列 X 为升序	列 X 为随机顺序	否
7.	列 X 为随机顺序	列 X 为降序	否
8.	列 X 为降序	列 X 为随机顺序	否
9.	分区	非分区	否。注: 这假定源表为分区表。
10.	pctfree n1	pctfree n2	是
11.	level2pctfree n1	level2pctfree n2	是
12.	minpctused n1	minpctused n2	是
13.	不允许逆向扫描	允许逆向扫描	是，无论是否允许逆向扫描，物理索引结构都相同。
14.	允许逆向扫描	不允许逆向扫描	是，原因与（9）相同。
15.	pagesplit [L|H|S]	pagesplit [L|H|S]	是
16.	抽样统计信息	详细统计信息	是
17.	详细统计信息	抽样统计信息	是
18.	未集群	CLUSTER	是

表 14. 确定目标索引属性不同于源索引属性时源索引是否匹配。 (续)

规则编号	目标索引属性	源索引属性	源索引匹配吗?
19.	CLUSTER	未集群	是。该索引将成为集群索引，但是直到重组了数据，才将根据此索引对数据进行集群。在进行连接以根据此索引分区对数据进行集群之后，可以使用分区级别重组。
20.	忽略无效	拒绝无效	是
21.	拒绝无效	忽略无效	否。需要考虑拒绝无效值的目标索引属性，并且源表可能包含违反此索引约束的行。
22.	已启用索引压缩	未启用索引压缩	是。注：直到重建了索引，才将压缩底层的索引数据。
23.	未启用索引压缩	已启用索引压缩	是。注：直到重建了索引，才将解压缩索引数据。

注: 如果您尝试将使用 DB2 V9.7 或更低发行版创建的多维集群 (MDC) 表（具有非分区块索引）连接至使用 DB2 V9.7 FP1 或更高发行版创建的新 MDC 分区表（具有分区块索引），并且使用了 ERROR ON MISSING INDEXES 子句，那么根据规则编号 9，ALTER TABLE ... ATTACH PARTITION 语句将失败并返回错误消息 SQL20307N (SQLSTATE 428GE)。除去 ERROR ON MISSING INDEXES 子句会允许连接完成，因为在连接操作期间数据库管理器会维护索引。如果接收到错误消息 SQL20307N（SQLSTATE 428GE），那么应该考虑除去 ERROR ON MISSING INDEXES 子句。

或者也可以使用联机表移动过程，以将具有非分区块索引的 MDC 分区表转换为具有分区块索引的表。

拆离数据分区

表分区功能提高了表数据的转入和转出效率。这是通过使用 ALTER TABLE 语句的 ATTACH PARTITION 和 DETACH PARTITION 子句实现的。

开始之前

要从分区表拆离数据分区，您必须具有下列权限或特权:

- 执行 DETACH PARTITION 操作的用户必须有权对源表执行 ALTER、SELECT 和 DELETE 操作。
- 该用户还必须有权创建目标表。因此，要改变一个表以拆离数据分区，语句授权标识拥有的特权必须至少包括对目标表的下列其中一项权限或特权:
  - DBADM 权限
  - 对数据库的 CREATETAB 权限、对该表所使用表空间的 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）

注: 拆离数据分区时，语句授权标识将会有效地执行 CREATE TABLE 语句，因此必须具有执行该操作所必需的特权。ALTER TABLE 语句的授权标识将成为新表的定义者（具有 CONTROL 权限），就像是该用户发出了 CREATE TABLE 语句一样。不会将所改变的表的任何特权传递至新表。只有 ALTER TABLE 语句的授权标识以及具有 DBADM 或 DATAACCESS 权限的用户才有权在 ALTER TABLE...DETACH PARTITION 语句执行之后立即访问数据。

关于此任务

通过转出分区表数据，可以方便地从分区表中分离出某些范围的数据。一旦将数据分区拆离成单独的表，就可以通过多种方法处理该表。可以删除单独的表（这样就破坏了数据分区中的数据）；可以对它进行归档或者以别的方式将它作为单独的表使用；将它连接到另一个分区表（例如历史表）；也可以对它进行操纵、清理和变换以及将它重新连接到原始分区表或另一分区表。

借助 DB2 V9.7 FP1 和更高发行版，在使用带 DETACH PARTITION 子句的 ALTER TABLE 语句从分区表拆离数据分区时，源分区表将保持联机。针对表运行的查询继续运行。在以下两阶段进程中，要拆离的数据分区会被转换为独立表:

1. ALTER TABLE...DETACH PARTITION 操作在逻辑上将数据分区从分区表拆离。
2. 异步分区拆离任务将在逻辑上已拆离的分区转换为独立的表。

如果有任何需要根据已拆离数据分区进行增量维护的从属表（这些从属表称为已拆离的从属表），那么仅在对所有已拆离的从属表运行 SET INTEGRITY 语句之后，异步分区拆离任务才会开始。

如果没有已拆离的从属表，那么在发出 ALTER TABLE...DETACH PARTITION 语句的事务落实之后，异步分区拆离任务即会开始。

限制

如果源表是由 DB2 版本 9.7 或更低发行版创建的 MDC 表，那么块索引为非分区索引。不允许在 ALTER TABLE...DETACH PARTITION 操作所在的工作单元中访问新拆离的表。MDC 表不支持分区块索引。在此情况下，在落实 ALTER TABLE...DETACH PARTITION 操作之后，会在对表进行第一次访问时创建块索引。如果在拆离时间之前源表具有任何其他分区索引，那么目标表的索引对象会被标记为无效以允许创建块索引。因此，在创建块索引以及重新创建任何分区索引期间，访问时间将增加。

当源表是由 DB2 V9.7 FP1 或更高发行版创建的 MDC 时，块索引为分区索引，且分区索引成为拆离目标表的索引，不需要重新创建。

要执行 DETACH PARTITION 操作，必须符合下列条件:

- 要从中拆离数据分区的表（源表）必须存在，并且必须是分区表。
- 要拆离的数据分区必须存在于源表中。
- 源表必须有多个数据分区。分区表必须至少有一个数据分区。只有已连接的可视数据分区才能用于此上下文。已连接数据分区就是已连接但尚未使用 SET INTEGRITY 语句进行验证的数据分区。
- DETACH PARTITION 操作将要创建的表（目标表）的名称不能已存在。

- 不允许对强制实施的引用完整性 (RI) 关系的父表执行 DETACH PARTITION。如果您具有包含强制执行的 RI 关系的表并且希望从父表拆离数据分区，那么可使用变通方法。在以下示例中，所有语句都在同一工作单元 (UOW) 内运行，以锁定并发更新:

```
// Change the RI constraint to informational:
ALTER TABLE child ALTER FOREIGN KEY fk NOT ENFORCED;

ALTER TABLE parent DETACH PARTITION p0 INTO TABLE pdet;

SET INTEGRITY FOR child OFF;

// Change the RI constraint back to enforced:
ALTER TABLE child ALTER FOREIGN KEY fk ENFORCED;

SET INTEGRITY FOR child ALL IMMEDIATE UNCHECKED;
// Assuming that the CHILD table does not have any dependencies on partition P0,
//   and that no updates on the CHILD table are permitted until this UOW is
complete,
//   no RI violation is possible during this UOW.

COMMIT WORK;
```

- 如果有任何需要根据已拆离数据分区进行递增维护的从属表（这些从属表称为已拆离的从属表），那么需要对已拆离的从属表运行 SET INTEGRITY 语句，以便对这些表进行递增维护。利用 DB2 版本 9.7 FP 1 或更高发行版，在对所有已拆离的从属表运行 SET INTEGRITY 语句后，异步分区拆离任务将使数据分区进入独立的目标表。直到异步分区拆离操作完成，目标表将不可用。

过程

1. 要改变分区表以及从该表中拆离数据分区，请发出带 DETACH PARTITION 子句的 ALTER TABLE 语句。
2. 可选: 如果希望新拆离的独立表上具有相同约束，那么在完成拆离操作之后，请在目标表上运行 ALTER TABLE... ADD CONSTRAINT。

   如果已在源表上对索引划分分区，那么目标表已存在满足该约束所需的任何索引。

结果

已拆离分区将使用系统生成的名称重命名（使用格式 SQL*yymmddhhmmssxxx*），以便后续连接可直接重复使用已拆离分区名。

对于要拆离的数据分区，源表上定义的每个索引分区都成为目标表上的索引。执行拆离分区操作期间，不会实际移动索引对象。但是，会从目录表 SYSINDEXPARTITIONS 中除去要拆离的表分区的索引分区元数据。系统会由于拆离分区操作而在 SYSINDEXES 中为新表添加新索引条目。原始索引标识 (IID) 会保留并且保持唯一，就如它在源表上一样。

目标表的余留索引的索引名是系统生成的（使用格式 SQL*yymmddhhmmssxxx*）。除了采用 SYSIBM 模式的任何路径索引、区域索引和 MDC 或 ITC 块索引之外，这些索引的模式与目标表的模式相同。其他由系统生成的索引（如那些用于强制执行唯一键和主键约束的索引）将具有目标表的模式，因为索引转移到所拆离表上，但约束则不然。可以使用 RENAME 语句来重命名未采用 SYSIBM 模式的索引。

在创建源表时指定的表级别 INDEX IN 选项不会由目标表继承。而是，分区级别 INDEX IN（如果指定）或拆离分区的缺省索引表空间会继续成为目标表的索引表空间。

当拆离数据分区时，会将一些统计信息从要拆离的分区转移到目标表中。具体来说，来自分区索引的 SYSINDEXPARTITIONS 的统计信息将转移到新拆离表的条目 SYSINDEXES 中。来自 SYSDATAPARTITIONS 的统计信息将被复制到新拆离表的 SYSTABLES 中。

下一步做什么

对新拆离的表和源表完成 DETACH PARTITION 操作之后运行 **RUNSTATS**，因为在完成拆离分区操作后将不会转移其中的大量统计信息。

已拆离数据分区的属性

使用 ALTER TABLE 语句的 DETACH PARTITION 子句来从分区表拆离数据分区后，该数据分区成为独立非分区目标表。

目标表的许多属性继承自源表。对于任何未从源表继承的属性来说，它们的设置方式就像是启动了 DETACH 操作的用户在创建目标表一样。如果源表上存在分区索引，那么该索引将传递到目标表。如果源表包含了具有基于表达式的键部件的分区索引，那么将创建系统生成的统计信息视图和软件包，并使其与目标表中的索引相关联。

在 DETACH 操作之后，目标表将继承源表上定义的所有分区索引。这些索引包括系统生成的索引和用户定义的索引。在拆离操作期间，不会物理地移动索引对象。将从 SYSINDEXPARTITIONS 目录中除去要拆离的数据分区的索引分区元数据。会在新表的 SYSINDEXES 中添加新条目。源表中任何给定分区索引的索引标识（IID）将是目标表上索引的 IID（对于表，该 IID 将保持唯一，并且在拆离期间保持不变）。

新表上余留索引的索引名由系统生成，使用的格式如下：SQLyymmddhhmmssxxx。路径索引、区域索引和 MDC 或 ITC 索引包含在 SYSIBM 模式中。所有其他索引则包含在新表的模式中。用于强制执行唯一键和主键约束的此类由系统生成的索引包含在新表的模式中，因为这些索引已被转移到新表上。在 DETACH 操作之后，源表上的约束将不由目标表继承。

可在其他时间使用 RENAME 语句来重命名未采用 SYSIBM 模式的索引。

在完成拆离操作之后，可在新表上使用 ALTER TABLE ... ADD CONSTRAINT 语句来对新表强制执行源表上的约束。

源表上表级别 INDEX IN 子句指定的表空间位置不由新的目标表继承。而是，分区级别 INDEX IN 子句指定的表空间位置或新表的缺省索引表空间继续充当新表的索引表空间位置。

目标表继承的属性

目标表继承的属性包括:

- 下列列定义:
  - 列名
  - 数据类型（对于具有长度和精度的数据类型（例如 CHAR 和 DECIMAL）来说，还包括长度和精度）
  - 可空性
  - 列缺省值

  - INLINE LENGTH
  - 代码页（SYSCAT.COLUMNS 目录视图的 CODEPAGE 列）
  - LOB 日志记录（SYSCAT.COLUMNS 目录视图的 LOGGED 列）
  - LOB 压缩（SYSCAT.COLUMNS 目录视图的 COMPACT 列）
  - 压缩（SYSCAT.COLUMNS 目录视图的 COMPRESS 列）
  - 隐藏列类型（SYSCAT.COLUMNS 目录视图的 HIDDEN 列）
  - 列顺序
- 如果源表是多维集群 (MDC) 表或插入时间集群 (ITC) 表，那么目标表也是 MDC 表或 ITC 表，并且是使用相同的维列定义的。
- 块索引定义。在落实 DETACH 操作后第一次访问新拆离的独立表时，将重建索引。
- 表空间标识和表对象标识继承自数据分区，而不是继承自源表。这是因为在 DETACH 操作期间未移动任何表数据。在目录方面，源数据分区中 SYSCAT.DATAPARTITIONS 目录视图的 TBSPACEID 列将成为 SYSCAT.TABLES 目录视图的 TBSPACEID 列。当转换成表空间名称时，目标表中有 SYSCAT.TABLES 目录视图的 TBSPACE 列。源数据分区中 SYSCAT.DATAPARTITIONS 目录视图的 PARTITIONOBJECTID 列将成为目标表中 SYSCAT.TABLES 目录视图的 TABLEID 列。
- 源数据分区中 SYSCAT.DATAPARTITIONS 目录视图的 LONG_TBSPACEID 列将转换为表空间名并成为目标表中 SYSCAT.TABLES 的 LONG_TBSPACE 列。
- 源数据分区（分区级别索引表空间）的 SYSDATAPARTITIONS 中 INDEX_TBSPACEID 列值将转换为表空间名称并成为目标表的 SYSTABLES 中的 INDEX_TBSPACE 值。CREATE TABLE 语句中表级别 INDEX IN <table space> 指定的索引表空间将不由目标表继承。
- 表空间位置
- 多分区数据库的分发映射标识（SYSCAT.TABLES 目录视图的 PMAP_ID 列）
- 可用百分比（SYSCAT.TABLES 目录视图的 PCTFREE 列）
- 追加方式（SYSCAT.TABLES 目录视图的 APPEND_MODE 列）
- 首选锁定详细程度（SYSCAT.TABLES 目录视图的 LOCKSIZE 列）
- 数据捕获（SYSCAT.TABLES 目录视图的 DATA_CAPTURE 列）
- VOLATILE（SYSCAT.TABLES 目录视图的 VOLATILE 列）
- DROPRULE（SYSCAT.TABLES 目录视图的 DROPRULE 列）
- 压缩（SYSCAT.TABLES 目录视图的 COMPRESSION 列）
- 搜索最大可用空间（SYSCAT.TABLES 目录视图的 MAXFREESPACESEARCH 列）

注: 不支持分区的分层表或临时表、范围集群表和分区视图。

不从源表继承的属性

不从源表继承的属性包括:

- 不继承目标表类型。目标表始终是常规表。
- 特权和权限
- 模式
- 生成列、标识列、检查约束和引用约束。在源列是生成列或标识列的情况下，相应的目标列没有显式的缺省值，这表示它的缺省值为 NULL。

- 表级别索引表空间（SYSCAT.TABLES 目录视图的 INDEX_TBSPACE 列）。DETACH 操作产生的表的索引与该表在同一个表空间中。
- 触发器
- 主键约束和唯一键约束
- 将不继承非分区索引的统计信息。
- 未包括在显式从源表继承的属性列表中的所有其他属性。

数据分区拆离阶段

利用 DB2 V9.7 FP1 及更高发行版，从数据分区表拆离数据分区包含两个阶段。第一个阶段在逻辑上将分区从表拆离，第二个阶段将数据分区转换为独立的表。

发出 ALTER TABLE...DETACH PARTITION 语句时，即会启动拆离过程。

1. ALTER TABLE...DETACH PARTITION 操作在逻辑上将数据分区从分区表拆离。
2. 异步分区拆离任务将在逻辑上已拆离的分区转换为独立的表。

如果有任何需要根据已拆离数据分区进行增量维护的从属表（这些从属表称为已拆离的从属表），那么仅在对所有已拆离的从属表运行 SET INTEGRITY 语句之后，异步分区拆离任务才会开始。

如果没有已拆离的从属表，那么在发出 ALTER TABLE...DETACH PARTITION 语句的事务落实之后，异步分区拆离任务即会开始。

DETACH 操作

ALTER TABLE...DETACH PARTITION 操作以下列方式执行:

- DETACH 操作不会等待动态未落实的读（UR）隔离级别查询即会继续，它也不会中断任何当前正在运行的动态 UR 查询。即使 UR 查询正在访问要拆离的分区，此行为也会发生。
- 如果动态非 UR 查询（读取或写入查询）未锁定要拆离的分区，那么在针对表运行动态非 UR 查询期间，DETACH 操作可以完成。
- 如果动态非 UR 查询已锁定要拆离的分区，那么 DETACH 操作会等待释放锁定。
- 必须对依赖于该表的所有静态程序包执行硬失效，DETACH 操作才能继续。
- 适用于数据定义语言（DDL）语句的下列限制也适用于 DETACH 操作，因为 DETACH 需要更新目录:
  - 不能针对表编译新查询。
  - 不能对针对表运行的查询执行绑定或重新绑定。

  要尽量降低这些限制的影响，请在 DETACH 操作后立即发出 COMMIT。

在 DETACH 操作期间，数据分区名称会更改为 SQL*yymmddhhmmssxxx* 格式的系统生成名称，并且在 SYSCAT.DATAPARTITIONS 中，如果没有已拆离的从属表，那么会将分区的状态设置为“L”；如果有已拆离的从属表，那么会将分区的状态设置为“D”。

在 DETACH 操作期间，将在 SYSCAT.TABLES 中为目标表创建条目。如果有已拆离的从属表，那么会将表 TYPE 设置为“L”。针对所有已拆离的从属表运行 SET INTEGRITY 之后，会将 TYPE 设置为“T”，但是，目标表仍然不可用。异步分区拆离任务将完成拆离操作并使目标表可用。

在 DETACH 操作期间，动态 SQL 的软失效可让在 ALTER TABLE...DETACH PARTITION 语句之前启动的动态 SQL 查询与 DETACH 操作并行继续运行。ALTER TABLE...DETACH PARTITION 语句获取对分区表的 IX 锁定和对要拆离的数据分区的 X 锁定。

异步分区拆离任务

在 DETACH 操作落实且刷新任何已拆离的从属表之后，异步分区拆离任务将在逻辑上已拆离的分区转换为独立的表。

异步分区拆离任务会等待在拆离操作的阶段 1 之前开始的、分区表的所有访问操作完成。如果分区表有非分区索引，那么异步分区拆离任务会创建异步索引清除任务以执行延迟的索引清除。访问完成后，通过将在逻辑上已拆离的分区转换为独立的表，异步分区拆离任务将完成拆离操作的阶段 2。

LIST UTILITIES 命令可用于监视异步分区拆离任务的过程。**LIST UTILITIES** 命令指出异步分区拆离任务是否处于下列其中一种状态:

- 等待对分区表的旧访问完成
- 最终化拆离操作并使目标表可用

数据分区表的异步分区拆离

对于 DB2 V9.7 FP1 及更高发行版，异步分区拆离任务完成将数据分区从分区表拆离的操作（以前由 ALTER TABLE...DETACH 操作启动）。该任务是一个异步后台进程 (ABP)，在分区成为在逻辑上已拆离的分区后启动。

异步分区拆离任务可以加快将数据分区从分区表拆离的过程。如果分区表具有从属具体化查询表 (MQT)，那么直到针对 MQT 执行 SET INTEGRITY 语句之后，才会启动该任务。

通过以异步方式完成拆离数据分区的操作，在发出 ALTER TABLE...DETACH PARTITION 语句之前启动的访问分区表的查询在一旦拆离分区后立即继续。

如果有任何需要根据已拆离数据分区进行增量维护的从属表（这些从属表称为已拆离的从属表），那么仅在对所有已拆离的从属表运行 SET INTEGRITY 语句之后，异步分区拆离任务才会开始。

如果没有已拆离的从属表，那么在发出 ALTER TABLE...DETACH PARTITION 语句的事务落实之后，异步分区拆离任务即会开始。

异步分区拆离任务执行下列操作:

- 对 ALTER TABLE...DETACH 操作先前执行软失效的高速缓存语句执行硬失效。
- 更新源分区表和目标独立表的目录条目，并使目标表可用。
- 对于具有非分区块索引但没有其他分区索引的多维集群 (MDC) 表，请为目标表创建索引对象。块索引在异步分区拆离任务落实后首次访问目标表时创建。
- 为包含 XML 列的表的目标表创建系统路径索引。
- 更新包含已拆离分区的表空间的最短恢复时间 (MRT)。
- 为非分区索引创建异步索引清除 AIC 任务。AIC 任务在异步分区拆离完成后执行索引清除。

- 如果非分区索引不存在于表上，那么释放数据分区标识。

异步分区拆离任务对性能的影响

异步分区拆离任务产生很小的性能影响。通过对 ALTER TABLE...DETACH 操作先前执行软失效的高速缓存语句执行硬失效，任务将等待对已拆离分区的所有访问完成。然后，任务获取对表和分区的所需锁定，再继续处理以使已拆离的分区成为独立的表。

监视异步分区拆离任务

分发守护程序和异步分区拆离任务代理程序都是内部系统应用程序，它们分别使用应用程序名称 **db2taskd** 和 **db2apd** 出现在 **LIST APPLICATIONS** 命令输出中。为了防止意外中断，不能强制系统应用程序。只要数据库处于活动状态，分发守护程序就会保持联机状态。任务将保持活动，直到拆离完成为止。如果拆离在进行中时取消激活数据库，那么异步分区拆离任务将在重新激活数据库后继续。

LIST UTILITIES 命令指出异步分区拆离任务是否处于下列其中一种状态:

- 等待对分区表的旧访问完成
- 最终化拆离操作并使目标表可用

LIST UTILITIES SHOW DETAIL 命令的以下样本输出显示 WSDB 数据库中的异步分区拆离任务:

```
ID                   = 1
Type                 = ASYNCHRONOUS PARTITION DETACH
Database Name        = WSDB
Partition Number     = 0
Description          = Finalize the detach for partition '4' of table 'USER1.ORDERS'.
Start Time           = 07/15/2009 14:52:14.476131
State                = Executing
Invocation Type      = Automatic
Progress Monitoring:
    Description      = Waiting for old access to the partitioned table to complete.
    Start Time       = 07/15/2009 14:52:51.268119
```

在 LIST UTILITIES 命令的输出中，异步分区拆离任务的主要描述标识要拆离的数据分区和拆离操作创建的目标表。进度监视描述提供关于异步分区拆离任务的当前状态的信息。

注: 异步分区拆离任务是异步进程。要知道拆离操作的目标表何时可用，可创建一个存储过程，该存储过程查询 SYSCAT.DATAPARTITIONS 目录视图的 STATUS 列并在拆离操作完成时返回。

分区数据库环境中的异步分区拆离处理

在分区数据库环境中，不管有多个少数据库分区，将为每个 DETACH 操作创建一个异步分区拆离任务。此任务是在目录数据库分区上创建的，并根据需要将工作分发至余下数据库分区。

异步分区拆离任务的错误处理

异步分区拆离任务基于事务。如果任务失败，那么将在内部回滚它所进行的所有更改。在异步分区拆离处理期间发生的任何错误都将记录在 **db2diag** 日志文件中。系统稍后会重试失败的任务。

对分区表添加数据分区

创建分区表后，可以使用 ALTER TABLE 语句来修改该表。确切地说，可以使用 ADD PARTITION 子句来对现有分区表添加新数据分区。

关于此任务

在下列情况下，对分区表添加数据分区要比连接数据分区更为合适：数据是随着时间的推移而添加到数据分区中的；数据是缓慢移动的而不是从外部源转入的；您直接将数据插入或装入到分区表中。特定的示例包括每天将数据装入到一月份数据的数据分区中或者持续不断地插入各行。

为了将新数据分区添加至特定表空间位置，会将 IN 子句作为 ALTER TABLE ADD PARTITION 语句中的选项来添加。

为了将新数据分区的分区索引添加至特定表空间位置（不同于该数据分区的表空间位置），会将分区级别 INDEX IN 子句作为 ALTER TABLE ADD PARTITION 语句中的选项来添加。如果没有指定 INDEX IN 选项，那么缺省情况下，新数据分区上的任何分区索引都将与该数据分区位于同一表空间中。如果分区表上存在任何分区索引，那么 ADD PARTITION 子句会为新分区创建对应的空索引分区。对于每个分区索引，系统会在 SYSCAT.INDEXPARTITIONS 目录视图中插入一个新条目。

要将新数据分区的 LONG、LOB 或 XML 数据添加至特定表空间位置（不同于该数据分区的表空间位置），应在 ALTER TABLE ADD PARTITION 语句上添加分区级别 LONG IN 子句作为选项。

使用带有 ADD PARTITION 子句的 ALTER TABLE 语句将数据分区添加到分区表时，目标分区表仍然处于联机状态，并且在 RS、CS 或 UR 隔离级别下运行的针对该表的动态查询将继续运行。

限制和用法准则

- 不能向非分区表添加数据分区。要了解有关将现有表迁移到分区表的详细信息，请参阅第 163 页的『将现有表和视图迁移到分区表』。
- 每个新数据分区的值的范围由 STARTING 和 ENDING 子句确定。
- 必须至少提供 STARTING 和 ENDING 子句两者中的一个。
- 新范围不能与现有数据分区的范围重叠。
- 在第一个现有数据分区前面添加新数据分区时，必须指定 STARTING 子句。使用 MINVALUE 来使此范围的末端是开放的。
- 同样，如果要在最后一个现有数据分区后面添加新数据分区，那么必须指定 ENDING 子句。使用 MAXVALUE 来使此范围的末端是开放的。
- 如果省略了 STARTING 子句，那么数据库将在上一数据分区结束边界的紧后面建立开始边界。同样，如果省略 ENDING 子句，那么数据库会正好在下一个数据分区的起始边界前创建结尾边界。
- 起始子句和结束子句的语法与 CREATE TABLE 语句中的指定语法相同。
- 如果未对 ADD PARTITION 指定任何 IN、INDEX IN 或 LONG IN 子句，那么用来存放数据分区的表空间是使用 CREATE TABLE 语句所使用的方法选择的。
- 包在 ALTER TABLE ...ADD PARTITION 操作期间会失效。
- 一旦落实 ALTER TABLE 语句，新添加的数据分区就会变为可用。

- 如果某个表具有非分区索引，那么您将无法访问该表中与创建了该分区的添加或连接操作位于同一事务中的新数据分区，条件是该事务未以互斥方式锁定此表（SQL0668N，原因码为 11）。

通过在 ADD 操作中省略 STARTING 或 ENDING 边界，可以填充范围值间隔。以下示例使用 ADD 操作来填充间隔，在该操作中只指定了起始边界:

```
CREATE TABLE hole (c1 int) PARTITION BY RANGE (c1)
(STARTING FROM 1 ENDING AT 10, STARTING FROM 20 ENDING AT 30);
DB20000I 已成功完成 SQL 命令。

ALTER TABLE hole ADD PARTITION STARTING 15;
DB20000I 已成功完成 SQL 命令。

SELECT SUBSTR(tabname, 1,12) tabname,
SUBSTR(datapartitionname, 1, 12) datapartitionname,
seqno, SUBSTR(lowvalue, 1, 4) lowvalue, SUBSTR(highvalue, 1, 4) highvalue
FROM SYSCAT.DATAPARTITIONS WHERE TABNAME='HOLE' ORDER BY seqno;

TABNAME DATAPARTITIONNAME SEQNO LOWVALUE HIGHVALUE
------------ ---------------- ----------- -------- ---------
HOLE PART0 0 1 10
HOLE PART2 1 15 20
HOLE PART1 2 20 30

3 record(s) selected.
```

示例 1: 对现有分区表添加数据分区，该分区存放 901 到 1000 之间的值（包含 901 和 1000）。假定 SALES 表包含 9 个范围: 0 到 100、101 到 200 等等直到值 900。本示例在表的末尾添加一个范围，此范围由未包括的 STARTING 子句指示:

```
 ALTER TABLE sales ADD PARTITION dp10
ENDING AT 1000 INCLUSIVE
```

为了将新数据分区的分区索引添加至特定表空间位置（不同于该数据分区的表空间位置），会将分区级别 INDEX IN 子句作为 ALTER TABLE ADD PARTITION 语句中的选项来添加。如果没有指定任何 INDEX IN 选项，那么缺省情况下，新数据分区上的任何分区索引都将与该数据分区位于同一表空间中。如果分区表上存在任何分区索引，那么 ADD PARTITION 会为新分区创建对应的空索引分区。对于每个分区索引，系统会在 SYSCAT.INDEXPARTITIONS 目录视图中插入一个新条目。

示例 2: 将数据分区添加至现有分区表中，从而将长整型数据和索引与该数据分区的其余内容分离。

```
ALTER TABLE newbusiness ADD PARTITION IN tsnewdata
INDEX IN tsnewindex LONG IN tsnewlong
```

删除数据分区

要删除数据分区，请拆离该分区，然后删除由拆离操作创建的表。使用带有 DETACH PARTITION 子句的 ALTER TABLE 语句拆离分区并创建独立表，然后使用 DROP TABLE 语句删除该表。

开始之前

要从分区表拆离数据分区，用户必须拥有下列权限或特权:

- 执行 DETACH 操作的用户必须有权对源表执行 ALTER、SELECT 和 DELETE 操作。

- 该用户还必须有权创建目标表。因此，要改变一个表以拆离数据分区，语句授权标识拥有的特权必须至少包括对目标表的下列其中一项权限或特权:
  - DBADM 权限
  - 对数据库的 CREATETAB 权限、对该表所使用表空间的 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）

要删除表，用户必须拥有下列权限或特权:

- 您必须是 SYSCAT.TABLES 的 DEFINER 列中记录的定义者，或者至少拥有下列其中一项特权:
  - DBADM 权限
  - 对表模式的 DROPIN 特权
  - 对表的 CONTROL 特权

注: 拆离数据分区情况的含义是，语句的授权标识实际上将要发出 CREATE TABLE 语句，所以必须拥有执行该操作所必需的特权。表空间就是正在拆离的数据分区所在的表空间。ALTER TABLE 语句的授权标识将成为新表的定义者（具有 CONTROL 权限），就像是该用户发出了 CREATE TABLE 语句一样。不会将所改变的表的任何特权传递至新表。只有 ALTER TABLE 语句的授权标识以及 DBADM 或 SYSADM 才有权在 ALTER TABLE ...DETACH PARTITION 操作执行后立即访问数据。

过程

要拆离分区表的数据分区，请发出带 DETACH PARTITION 子句的 ALTER TABLE 语句。

示例

在以下示例中，将数据分区 dec01 从表 STOCK 中拆离并放到表 JUNK 中。确定异步分区拆离任务已经使目标表 JUNK 可用之后，可以删除表 JUNK，从而有效地删除关联的数据分区。

```
ALTER TABLE stock DETACH PART dec01 INTO junk;
 -- After the target table becomes available, issue the DROP TABLE statement
 DROP TABLE junk;
```

下一步做什么

为了使 ALTER TABLE...DETACH 在 DB2 V9.7 FP1 及更高发行版中尽可能快地执行，异步分区拆离任务以异步方式完成拆离操作。如果有已拆离的从属表，那么异步分区拆离任务将不会开始且已拆离数据分区不会变成独立表。在此情况下，必须对所有已拆离的从属表发出 SET INTEGRITY 语句。在 SET INTEGRITY 完成后，异步分区拆离任务将会开始并使目标表可访问。目标表可访问时，就可以将其删除。

方案: 旋转分区表中的数据

旋转 DB2 数据库中的数据指的是一种复用数据分区中的空间的方法，它是通过除去表中的旧数据（拆离分区操作）然后添加新数据（连接分区操作）来实现的。

开始之前

或者，可以在执行连接操作之前归档已拆离的分区并将新数据装入其他源表中。在以下方案中，将在其他步骤之前执行拆离操作；拆离操作可能与上一个步骤一样容易，这取决于您的特定需求。

要改变表以拆离数据分区，该语句的授权标识必须具有以下特权和权限:

- 至少具有对已拆离分区的目标表的下列其中一个权限:
  - 对数据库的 CREATETAB 权限、对该表使用的表空间的 USE 特权以及下列其中一个权限或特权:
    - 对数据库的 IMPLICIT_SCHEMA 权限（如果新表的隐式或显式模式名不存在）
    - 对模式的 CREATEIN 特权（如果新表的模式名指的是现有模式）
  - DBADM 权限
- 至少具有对源表的下列其中一个特权和权限:
  - 对表的 SELECT、ALTER 和 DELETE 特权
  - 对表的 CONTROL 特权
  - DATAACCESS 权限

要改变表以连接数据分区，该语句的授权标识必须包括以下特权和权限:

- 至少具有对源表的下列其中一个权限或特权:
  - 对表的 SELECT 特权以及对表的模式的 DROPIN 特权
  - 对表的 CONTROL 特权
  - DATAACCESS 权限
- 至少具有对目标表的下列其中一个特权和权限:
  - 对表的 ALTER 和 INSERT 特权
  - 对表的 CONTROL 特权
  - DATAACCESS 权限

过程

要旋转分区表中的数据，请发出 ALTER TABLE 语句。 以下示例通过除去 2008 年 12 月的数据并将其替换为 2010 年 12 月的最新数据来显示如何更新 STOCK 表。

1. 从 STOCK 表中除去旧数据。

   ```
   ALTER TABLE stock DETACH PARTITION dec08 INTO newtable;
   ```

2. 装入新数据。使用带 REPLACE 选项的 **LOAD** 命令时，将覆盖现有数据。

   ```
   LOAD FROM data_file OF DEL REPLACE INTO newtable
   ```

   注: 如果有已拆离的从属表，请在装入已拆离表之前对已拆离的从属表发出 SET INTEGRITY 语句。如果返回了 SQL20285N，请等至异步分区拆离任务完成，然后再次发出 SET INTEGRITY 语句。

3. 必要时，请执行数据清理活动，这些活动可包括以下操作:
   - 填充缺少的值
   - 删除不一致的和不完整的数据
   - 除去来自多个来源的冗余数据

- 变换数据
  - *规范化*。对于来自不同来源并且以不同方式表示同一个值的数据来说，在将这些数据转入仓库时必须对其进行协调。
  - *聚集*。在转入太详细而无法存储在仓库中的原始数据之前，必须对这些数据进行聚集。

4. 将数据作为新范围进行连接。

```
ALTER TABLE stock
  ATTACH PARTITION dec10
    STARTING '12/01/2008' ENDING '12/31/2010'
  FROM newtable;
```

5. 使用 SET INTEGRITY 语句来更新索引和其他从属对象。在执行 SET INTEGRITY 语句期间，允许进行读写访问。

```
SET INTEGRITY FOR stock ALLOW WRITE ACCESS
  IMMEDIATE CHECKED  FOR EXCEPTION IN stock USE stock_ex;
```

方案: 转入和转出分区表数据

数据仓库中的一种常见管理操作是定期转入新数据并转出过时数据。以下方案说明了这些任务。

方案 1: 通过拆离数据分区来转出过时数据

以下示例显示如何从名为 STOCK 的分区表中拆离不需要的数据分区 (DEC01)。拆离的数据分区用于创建一个名为 STOCK_DROP 的表，而无需移动任何数据。

```
ALTER TABLE stock DETACH PART dec01 INTO stock_drop;
COMMIT WORK;
```

为了加速拆离操作，源表上的索引清除是通过异步索引清除进程在后台自动完成的。如果源表上没有定义已拆离的从属表，那么不需要发出 SET INTEGRITY 语句就可以完成拆离操作。

可以删除新表，也可以将其连接到另一个表，或者也可首先将其截断并装入新数据，然后再将其重新连接到源表。即使异步索引清除尚未完成也可以立即执行这些操作，除非源表有已拆离的从属表。

要确定已拆离表是否可访问，请查询 SYSCAT.TABDETACHEDDEP 目录视图。如果发现已拆离表不可访问，那么对所有已拆离的从属表发出带有 IMMEDIATE CHECKED 选项的 SET INTEGRITY 语句。如果在维护已拆离表的所有已拆离从属表之前试图访问该已拆离表，那么将返回错误 (SQL20285N)。

方案 2: 创建一个新的空范围

以下示例显示如何向名为 STOCK 的分区表添加空数据分区 (DEC02)。STARTING FROM 和 ENDING AT 子句指定新数据分区的值范围。

```
ALTER TABLE stock ADD PARTITION dec02
 STARTING FROM '12/01/2002' ENDING AT '12/31/2002';
```

此 ALTER TABLE...ADD PARTITION 语句将等待正在对 STOCK 表运行的现有静态查询或可重复读查询完成，然后使相关程序包无效。这样的现有查询通常在添加操作开始处理 STOCK 表之前完成。针对 STOCK 表的现有动态不可重复读查询将继续运

行，并且可以与添加操作同时运行。在 ALTER TABLE...ADD PARTITION 语句开始处理 STOCK 表之后，任何访问该表的新查询必须等待添加操作完成。

将数据装入到表中:

```
LOAD FROM data_file OF DEL
 INSERT INTO stock
 ALLOW READ ACCESS;
```

发出 SET INTEGRITY 语句来验证约束并刷新从属具体化查询表 (MQT)。违反已定义的约束的任何行均会移动到异常表 STOCK_EX。

```
SET INTEGRITY FOR stock
 ALLOW READ ACCESS IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;

COMMIT WORK;
```

方案 3: 通过连接装入的数据分区来转入新数据

以下示例显示连接操作如何用于帮助将新范围的数据装入到现有分区表（名为 STOCK 的目标表）。数据会装入到一个新的空表 (DEC03)，会在其中对数据进行检查和清理（如果需要），而不会影响目标表。数据清理活动包括:

- 填充缺少的值
- 删除不一致的和不完整的数据
- 除去来自多个源的冗余数据
- 通过规范化或聚集来变换数据:
  - *规范化*。对于来自不同来源并且以不同方式表示相同值的数据，在转入过程中必须对其进行协调。
  - *聚集*。在转入期间，必须对由于太详细而无法存储在仓库中的原始数据进行聚集。

以这种方式准备数据后，便可将新装入的数据分区连接到目标表。

```
CREATE TABLE dec03(...);
LOAD FROM data_file OF DEL REPLACE INTO dec03;
(data cleansing, if necessary)
ALTER TABLE stock ATTACH PARTITION dec03
 STARTING FROM '12/01/2003' ENDING AT '12/31/2003'
 FROM dec03;
```

在连接操作期间，必须同时指定 STARTING FROM 和/或 ENDING AT 子句，并且下边界（STARTING FROM 子句）必须小于或等于上边界（ENDING AT 子句）。新连接的数据分区不能与目标表中的现有数据分区范围重叠。如果已将现有最高范围的上限定义为 MAXVALUE，那么任何尝试连接新上限范围的操作都将失败，因为该新范围会与现有上限范围重叠。结束于 MINVALUE 的下限也存在类似的限制。此外，除非新数据分区的新范围落在现有范围内，否则不能中途添加或连接新数据分区。如果用户未指定边界，那么将在创建表时确定边界。

此 ALTER TABLE...ATTACH PARTITION 语句将等待正在对 STOCK 表运行的现有静态查询或可重复读查询完成，然后使相关程序包无效。这样的现有查询通常在连接操作开始处理 STOCK 表之前完成。针对 STOCK 表的现有动态不可重复读查询将继续运行，并且可以与连接操作同时运行。在 ALTER TABLE...ATTACH PARTITION 语句开始处理 STOCK 表之后，任何访问该表的新查询必须等待连接操作完成。

已连接的数据分区中的数据仍不可视，因为尚未经 SET INTEGRITY 语句验证。SET INTEGRITY 语句是验证新连接的数据是否在定义的范围内所必需的。它还对索引和其他从属对象（例如，MQT）执行任何必需的维护活动。在 SET INTEGRITY 语句落实前，新数据将不可见；但是，如果 SET INTEGRITY 语句正在联机运行，那么 STOCK 表中的现有数据完全可供访问以进行读写操作。

提示: 如果在执行连接操作之前可以通过独立于数据服务器的应用程序逻辑执行数据完整性检查（包括范围验证和其他约束检查），那么可以更快地使新连接的数据可供使用。通过使用 SET INTEGRITY...ALL IMMEDIATE UNCHECKED 语句来跳过范围检查和约束违例检查，可以优化数据转入过程。在此情况下，表将脱离 SET INTEGRITY 暂挂状态，并且只要目标表中没有非分区用户索引，新数据就可以立即供应用程序使用。

注: 当 SET INTEGRITY 语句正在运行的情况下，您不能针对表执行数据定义语言 (DDL) 语句或实用程序操作。这些操作包括（但不限于）以下语句和命令:

- LOAD 命令
- REDISTRIBUTE DATABASE PARTITION GROUP 命令
- REORG INDEXES/TABLE 命令
- ALTER TABLE 语句
  - ADD COLUMN
  - ADD PARTITION
  - ATTACH PARTITION
  - DETACH PARTITION
- CREATE INDEX 语句

SET INTEGRITY 语句验证新连接的数据分区中的数据:

```
SET INTEGRITY FOR stock
 ALLOW WRITE ACCESS IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex;
```

落实事务将使表可供使用:

```
COMMIT WORK;
```

任何超出范围或违反其他约束的行都被移至异常表 STOCK_EX。您可查询此表，修正各行并将其插入到 STOCK 表中。

第 14 章 装入

并行性和装入

LOAD 实用程序利用使用多个处理器或多个存储设备的硬件配置，如对称多处理器 (SMP) 环境。

通过使用 LOAD 实用程序，有多种方法可用来并行处理大量数据。一种方法是通过使用多个存储设备，这允许在装入操作期间利用 I/O 并行性（请参阅图 37）。另一种方法涉及在 SMP 环境中使用多个处理器，这允许利用分区内并行性（请参阅图 38）。两种方法可一起使用以提高数据装入速度。

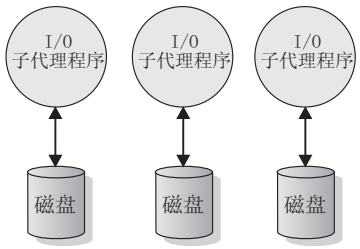


图 37. 在装入数据时利用 I/O 并行性

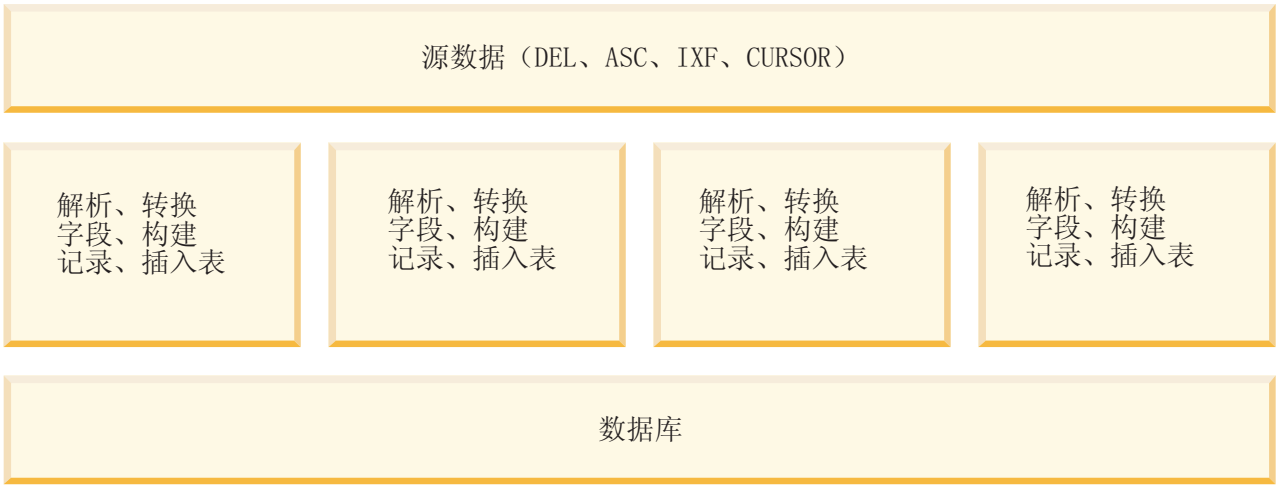


图 38. 在装入数据时利用分区内并行性

MDC 和 ITC 注意事项

将数据装入到多维集群 (MDC) 和插入时间集群 (ITC) 表中时，存在以下限制:

- 不支持 **LOAD** 命令的 SAVECOUNT 选项。
- 由于这些表管理它们自己的可用空间，所以不支持 totalfreespace 文件类型修饰符。

- MDC 表或 ITC 表需要 anyorder 文件类型修饰符。如果对 MDC 表或 ITC 表执行装入，但未指定 anyorder 修饰符，那么实用程序将显式启用该修饰符。

对 MDC 或 ITC 表使用 **LOAD** 命令时，将按以下方式处理唯一约束违例:

- 如果在执行装入操作前该表包含唯一键，并且将重复记录装入该表，那么将保留原始记录，并且将在删除阶段删除新记录。
- 如果在执行装入操作前该表未包含唯一键，并且将唯一键和重复记录都装入该表，那么将只装入其中一个带有唯一键的记录，并且将在删除阶段删除其他记录。

注: 没有确切的技术可用来确定将要装入的记录以及将要删除的记录。

性能注意事项

要提高 LOAD 实用程序在装入具有多个维的 MDC 表时的性能，应该增大 *util_heap_sz* 数据库配置参数值。当有更多内存可供该实用程序使用时，MDC 装入算法的性能会显著提高。这将减少在装入阶段执行数据集群时的磁盘 I/O 次数。自 V9.5 起，当系统中存在更多可用内存时，**LOAD** 命令的 DATA BUFFER 选项的值可以临时性超出 *util_heap_sz* 设置值。.

由于所有 MDC 和 ITC 表都有块索引，所以 MDC 或 ITC 装入操作始终包括构建阶段。

在装入阶段，将执行附加的记录以维护块映射。对于分配的每个扩展数据块，大约有两个附加的日志记录。为了确保性能良好，在设置 *logbufsz* 数据库配置参数的值时应该考虑此情况。

带索引的系统临时表用于将数据装入到 MDC 和 ITC 表中。该表的大小与装入的单个单元数成正比。该表中每一行的大小与 MDC 维键的大小成正比。ITC 表仅具有一个单元并使用双字节维键。为了最大程度地减少装入操作期间处理此表时执行的磁盘 I/O 次数，请确保临时表空间的缓冲池足够大。

分区表的装入注意事项

对目标表进行分区时，将支持所有现有装入功能，但存在以下常规限制:

- 当分区代理程序数大于 1 时，不支持一致点。
- 不支持将数据装入到数据分区子集中的同时保持其余数据分区完全联机。
- 装入操作使用的异常表不能分区。
- 如果目标表包含 XML 列，那么不能指定异常表。
- 当装入实用程序以插入方式或重新启动运行并且装入目标表具有任何已拆离的从属时，那么不能重建唯一索引。
- 与装入 MDC 表相同，装入分区表时将不会保留输入数据记录的精确排序。只有在单元或数据分区中才保留排序。
- 在每个数据库分区上利用多个格式化程序的装入操作仅保留输入记录的大致排序。在每个数据库分区上运行单个格式化程序，将输入记录按单元或表分区键进行分组。要在每个数据库分区上运行单个格式化程序，应显式请求 CPU_PARALLELISM 为 1。

一般装入行为

装入实用程序将数据记录插入到正确的数据分区中。在装入之前，不需要使用外部实用程序（如分割程序）来对输入数据进行分区。

装入实用程序不访问任何拆离的或相连的数据分区。数据仅插入到可视数据分区中。可视数据分区不会拆离，也不会相连。此外，装入替换操作不会截断拆离或相连的数据分区。因为装入实用程序获取针对目录系统表的锁定，所以它将等待任何未落实的 ALTER TABLE 事务。这些事务将获取针对目录表中的相关行的互斥锁定，并且互斥锁定必须先终止，装入操作才能继续。这意味着装入操作运行期间，可能没有未落实的 ALTER TABLE ...ATTACH、DETACH 或 ADD PARTITION 事务。将拒绝目标为拆离或相连的数据分区的所有输入源记录，但可从异常表中检索它们（如果指定了异常表）。会有一条参考消息写入消息文件，以指示某些目标表数据分区处于相连或拆离状态。针对对应于目标表的相关目录表行的锁定使得用户无法通过在装入实用程序运行时发出 ALTER TABLE ...ATTACH、DETACH 或 ADD PARTITION 操作来更改目标表的分区。

处理无效行

当装入实用程序遇到的记录不属于任何可视数据分区时，将拒绝该记录并且装入实用程序继续进行处理。因为范围限制违例而拒绝的记录的个数不会显式显示出来，但会包括在拒绝的记录的总行数中。因为范围违例而拒绝记录不会增加行警告的数目。会有一条消息 (SQL0327N) 写入装入实用程序消息文件，指示发现范围违例，但不会对每一个记录来记录消息。除了目标表中的所有列之外，异常表还包括用于描述特定行发生的类型违例的列。包含无效数据的行（包括不能分区的数据）将写至转储文件。

因为异常表插入成本很高，所以可以控制插入到异常表中的约束违例。例如，装入实用程序的缺省行为是将本来有效但因为范围限制或唯一约束违例而拒绝的行插入到异常表中。通过对 FOR EXCEPTION 子句分别指定 NORANGEEXC 或 NOUNIQUEEXC 可以关闭此行为。如果指定不应将这些约束违例插入到异常表中，或者未指定异常表，那么有关违反范围限制或唯一约束的行的信息将会丢失。

历史记录文件

如果目标表已分区，那么相应的历史记录文件条目不会包括目标表跨越的表空间列表。另一操作详细程度标识（“R”而不是“T”）指示对分区表运行了装入操作。

终止装入操作

终止装入替换操作将完全截断所有可视数据分区，而终止装入插入操作会将所有可视数据分区截断至装入前的长度。如果 ALLOW READ ACCESS LOAD 操作在装入复制阶段失败，那么在终止该操作期间，索引会变得无效。在终止涉及索引的 ALLOW NO ACCESS LOAD 操作时，索引也会变得无效，这是因为重建索引方式或者增量维护期间插入了键而使得索引处于不一致状态。将数据装入到多个目标中不会影响装入恢复操作，但将无法从装入阶段期间获取的一致点重新启动装入操作。在此情况下，如果对目标表进行分区，那么将忽略 SAVECOUNT 装入选项。此行为与将数据装入到 MDC 目标表中的行为一致。

生成列

如果生成列在任何分区、维或分布键中，那么会忽略 generatedoverride 文件类型修饰符并且装入实用程序会生成值，就像指定了 generatedignore 文件类

型修饰符一样。在此情况下，装入错误的生成列值可能导致将记录放置在错误的物理位置上，例如，错误的数据分区、MDC 块或数据库分区。例如，一旦记录放在错误的数据分区上，设置完整性就必须将其移至另一物理位置，这不能在联机设置完整性操作期间完成。

数据可用性

当前 ALLOW READ ACCESS 装入算法扩展至分区表。ALLOW READ ACCESS LOAD 操作允许并发阅读器访问整个表，包括装入和非装入数据分区。

要点: 从 V10.1 FP1 开始，已不推荐使用 ALLOW READ ACCESS 参数，在以后的发行版中可能会将其除去。有关更多详细信息，请参阅『不推荐使用 LOAD 命令中的 ALLOW READ ACCESS 参数』（网址为）。

Ingest 实用程序还支持分区表，并且比附带 ALLOW READ ACCESS 参数的 LOAD 命令更适合允许采用数据并行性和可用性。它可以从文件和管道中移动大量数据，而不用锁定目标表。此外，一旦根据耗用时间或行数落实了数据，数据就会变得可访问。

数据分区状态

在某些情况下，成功装入后可视数据分区可能切换至“设置完整性暂挂”状态和/或“只读访问”表状态。如果该表存在装入操作不能保留的约束，那么数据分区可能会置于这些状态。这种约束可能包括检查约束和拆离的具体化查询表。失败的装入操作会导致所有可视数据分区处于“装入暂挂”表状态。

错误隔离

不支持在数据分区级别进行错误隔离。隔离错误意味着在运行时未出现错误的数据分区上继续装入，而在运行时出现错误的数据分区上停止装入。可在不同数据库分区间隔离错误，但装入实用程序不能在一个可视数据分区子集上落实事务，而回滚其余可视数据分区。

其他注意事项

- 如果有任何索引标记为无效，那么不支持增量。如果索引需要重建或拆离的从属项需要使用 SET INTEGRITY 语句进行验证，那么认为索引无效。
- 同时支持装入到分区表中，这些表使用按范围分区、按散列分布或按维算法组织的任何组合进行分区。
- 对于包括受装入影响的对象和表空间标识列表的日志记录，这些日志记录的大小（LOAD START 和 COMMIT (PENDING LIST)）可能增长得很快，并且因此而降低可供其他应用程序使用的活动日志空间量。
- 当表同时进行了分区和分布时，分区数据库装入可能不会影响所有数据库分区。只有输出数据库分区上的对象才会更改。
- 在装入操作期间，分区表的内存消耗会随表数的增加而增加。注意，总增加量不是线性的，因为仅总内存要求的一小部分与数据分区数成正比。

第 15 章 在分区数据库环境中装入数据

装入概述 - 分区数据库环境

在多分区数据库环境中，大量的数据放在多个数据库分区中。分布键用来确定每部分数据所在的数据库分区。必须先*分布*数据，然后才能将该数据装入到正确的数据库分区中。

在多分区数据库中装入表时，装入实用程序可以:

- 并行地分布输入数据
- 同时在各个相应数据库分区中装入数据
- 将数据从一个系统传输到另一个系统

将数据装入到多分区数据库中分两阶段完成: 第一阶段为*设置阶段*，在此阶段获取数据库分区资源（如表锁定）；第二阶段为*装入阶段*，在此阶段将数据装入到数据库分区中。可以使用 **LOAD** 命令的 ISOLATE_PART_ERRS 选项来选择这些阶段的错误处理方式，并可以选择一个或多个数据库分区上的错误对未发生错误的数据库分区上的装入操作的影响。

在将数据装入多分区数据库时，可以使用下列其中一种方式:

PARTITION_AND_LOAD
对数据进行分布（有可能以并行方式进行分布），并且同时在各个相应数据库分区上装入数据。

PARTITION_ONLY
对数据进行分布（有可能以并行方式进行分布），并将输出写入每个装入数据库分区上指定位置中的文件。每个文件都包含分区头，该分区头指定数据在数据库分区上的分布方式，并指定可以使用 LOAD_ONLY 方式将该文件装入到数据库中。

LOAD_ONLY
假定数据已分布在数据库分区上；将跳过分布过程，并且在相应的数据库分区上同时装入数据。

LOAD_ONLY_VERIFY_PART
假定数据已分布在数据库分区上，但数据文件未包含分区头。将跳过分布过程，并且在相应的数据库分区上同时装入数据。在装入操作期间，将检查每一行以验证它是否在正确的数据库分区中。如果指定了 dumpfile 文件类型修饰符，那么会将发生数据库分区违例的行放到转储文件中。否则会删除那些行。如果特定装入数据库分区上存在数据库分区违例，那么会将一条有关该数据库分区的警告写至装入消息文件。

ANALYZE
生成最佳分发映射（在所有数据库分区之间均匀地分布数据）。

概念和术语

在讨论装入实用程序在带有多个数据库分区的分区数据库环境中的行为和操作时，将使用以下术语:

- *协调程序分区*是一个数据库分区，用户连接到该分区以执行装入操作。在 PARTITION_AND_LOAD、PARTITION_ONLY 和 ANALYZE 方式下，除非指定了 **LOAD** 命令的 CLIENT 选项，否则假定数据文件在此数据库分区上。如果指定 CLIENT，那么表示要装入的数据在连接的远程客户机上。
- 在 PARTITION_AND_LOAD、PARTITION_ONLY 和 ANALYZE 方式下，*预分区代理程序*读取用户数据并以循环方式将其分发给*分区代理程序*，后者将分布该数据。此过程始终是在协调程序分区上执行的。对于任何装入操作，每个数据库分区最多允许一个分区代理程序。
- 在 PARTITION_AND_LOAD、LOAD_ONLY 和 LOAD_ONLY_VERIFY_PART 方式下，在每个输出数据库分区上都运行*装入代理程序*，它协调该数据库分区上的数据装入操作。
- 在 PARTITION_ONLY 装入操作期间，在每个输出数据库分区上运行*“装入到文件”代理程序*。它们从分区代理程序接收数据并将该数据写入所在数据库分区上的文件中。
- SOURCEUSEREXIT 选项提供了一种工具，装入实用程序可通过该工具执行定制脚本或可执行文件（此处称为*用户出口*）。

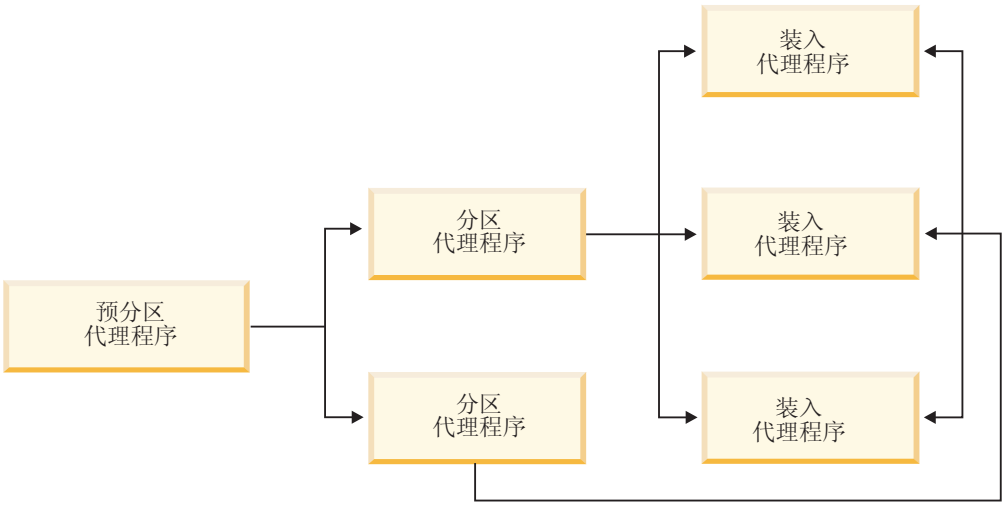


图 39. 分区数据库 load 概述. 预分区代理程序读取源数据，然后向两个分区代理程序中的每个分区代理程序各发送接近一半的数据，这两个分区代理程序分发数据并将该数据发送给三个数据库分区中的一个数据库分区。每个数据库分区上的装入代理程序装入数据。

在分区数据库环境中装入数据 - 提示与技巧

以下是在多分区数据库中装入表前要考虑的一些信息:

- 对少量数据使用装入实用程序，熟悉装入配置选项。

- 如果输入数据已排序或者具有某种选择的顺序，并且要在装入过程中维护该顺序，那么只应该将一个数据库分区用于分布。并行分布无法保证按照数据的接收顺序来装入该数据。缺省情况下，如果在 **LOAD** 命令中未指定 anyorder 修饰符，那么装入实用程序就会选择单分区代理程序。
- 如果正在从不同的文件装入大对象 (LOB)（即，如果使用装入实用程序时指定了 lobsinfile 修饰符），那么所有执行装入的数据库分区都必须能够对所有包含 LOB 文件的目录进行读访问。处理 LOB 时，LOAD *lob-path* 参数必须是标准路径。
- 通过将 ISOLATE_PART_ERRS 选项设置为 SETUP_ERRS_ONLY 或 SETUP_AND_LOAD_ERRS，可以强制在多分区数据库中运行的作业继续运行，即使装入操作在启动时检测到某些装入数据库分区或关联表空间或表处于脱机状态亦如此。
- 使用 STATUS_INTERVAL 装入配置选项来监视在多分区数据库中运行的作业的进度。装入操作按指定的时间间隔生成消息，以指示预分区代理程序已读取的数据的兆字节数。这些消息将被转储到预分区代理程序消息文件中。要在装入操作期间查看此文件的内容，请连接到协调程序分区并对目标表发出 **LOAD QUERY** 命令。
- 如果参与分布过程的数据库分区（由 PARTITIONING_DBPARTNUMS 选项定义）与装入数据库分区（由 OUTPUT_DBPARTNUMS 选项定义）不同，就会由于 CPU 周期争用情况减少而提高性能。将数据装入到多分区数据库中时，对未参与分布或装入操作的数据库分区调用装入实用程序。
- 如果在 **LOAD** 命令中指定了 MESSAGES 参数，就会将预分区代理程序、分区代理程序和装入代理程序生成的消息文件保存下来，以供装入操作完成后参考。要在执行装入操作期间查看这些文件的内容，请连接到适当的数据库分区并对目标表发出 **LOAD QUERY** 命令。
- 装入实用程序仅选择一个输出数据库分区以便在该分区中收集统计信息。可以使用 RUN_STAT_DBPARTNUM 数据库配置选项来指定该数据库分区。
- 在多分区数据库中装入数据之前，请运行设计顾问程序以确定每个表的最佳分区。有关更多信息，请参阅《*故障诊断和调整数据库性能*》中的“设计顾问程序”。

故障诊断

如果装入实用程序挂起，您可以:

- 使用 STATUS_INTERVAL 参数来监视多分区数据库装入操作的进度。会将状态时间间隔信息转储到协调程序分区上的预分区代理程序消息文件中。
- 检查分区代理程序消息文件，了解每个数据库分区上的分区代理进程状态。如果执行装入操作时未出错，并且设置了 TRACE 选项，那么这些消息文件应该会包含许多记录的跟踪消息。
- 检查装入消息文件以了解是否有任何装入错误消息。

  注: 必须指定 **LOAD** 命令的 MESSAGES 选项，这样这些文件才能存在。
- 如果有错误指示某个装入进程出错，请中断当前装入操作。

在分区数据库环境中装入数据

使用 LOAD 实用程序将数据装入到分区数据库环境中。

开始之前

在多分区数据库中装入表之前:

- 确保正确设置了 **svcename** 数据库管理器配置参数和 **DB2COMM** 概要文件注册表变量。此步骤很重要，因为 LOAD 实用程序使用 TCP/IP 将数据从预分区代理程序传输至分区代理程序以及从分区代理程序传输至装入数据库分区。
- 在调用 LOAD 实用程序前，必须连接至（或能够隐式连接至）要将数据装入到其中的数据库。
- 因为 LOAD 实用程序发出 COMMIT 语句，所以在开始执行装入操作前，应该通过发出 COMMIT 或 ROLLBACK 语句完成所有事务并释放任何锁定。如果使用的是 PARTITION_AND_LOAD、PARTITION_ONLY 或 ANALYZE 方式，那么装入的数据文件必须在此数据库分区上，但在下列情况下除外:
  1. 已指定 **CLIENT** 参数，在此情况下数据必须驻留在客户机上;
  2. 输入源类型为 CURSOR，在此情况下没有输入文件。
- 运行“设计顾问程序”以确定每个表的最佳数据库分区。有关更多信息，请参阅《*故障诊断和调整数据库性能*》中的“设计顾问程序”。

限制

在使用 LOAD 实用程序以在多分区数据库环境中装入数据时，下列限制适用:

- 装入操作的输入文件位置不能是磁带设备。
- **ROWCOUNT** 参数不受支持，除非正使用 ANALYZE 方式。
- 如果目标表包含分发所需的标识列，并且未指定 **identityoverride** 文件类型修饰符，或者您正使用多个数据库分区来进行分发然后装入数据，那么不支持在 **LOAD** 命令上对 **SAVECOUNT** 使用大于 0 的值。
- 如果分布键包含标识列，那么只支持 PARTITION_AND_LOAD 方式。
- LOAD_ONLY 和 LOAD_ONLY_VERIFY_PART 方式不能与 **LOAD** 命令的 **CLIENT** 参数配合使用。
- LOAD_ONLY_VERIFY_PART 方式不能与 CURSOR 输入源类型配合使用。
- 分发错误隔离方式 LOAD_ERRS_ONLY 和 SETUP_AND_LOAD_ERRS 不能与 **LOAD** 命令的 **ALLOW READ ACCESS** 和 **COPY YES** 参数配合使用。
- 如果 **OUTPUT_DBPARTNUMS** 和 **PARTITIONING_DBPARTNUMS** 选项指定的数据库分区不重叠，那么多个装入操作可同时将数据装入到同一个表中。例如，如果表是在数据库分区 0 至 3 上定义的，那么一个装入操作可以将数据装入到数据库分区 0 和 1 中，而另一个装入操作可以将数据装入到数据库分区 2 和 3 中。如果 **PARTITIONING_DBPARTNUMS** 选项指定的数据库分区重叠，那么装入操作自动选择 **PARTITIONING_DBPARTNUMS** 参数（在此情况下，没有任何装入分区子代理程序已在针对此表执行）或失败（如果没有任何参数）。

  从 V9.7 FP6 开始，如果 **PARTITIONING_DBPARTNUMS** 选项指定的数据库分区重叠，那么 LOAD 实用程序将自动从由 **OUTPUT_DBPARTNUMS** 指示的数据库分区尝试选择 **PARTITIONING_DBPARTNUMS** 参数（在此情况下，没有任何装入分区子代理程序已在针对此表执行）或失败（如果没有任何参数）。

  如果您要使用 **PARTITIONING_DBPARTNUMS** 选项显式指定分区，那么强烈建议您应该使用具有所有并发 **LOAD** 命令（每个命令指定不同的分区）的选项。如果您仅在某些并

发装入命令上指定 **PARTITIONING_DBPARTNUMS**，或如果指定重叠分区，那么 **LOAD** 命令至少将需要为某些并发装入选择备用分区节点，且该命令可能会极少失败 (SQL2038N)。

- 对于跨多个数据库分区的表来说，只能将非定界 ASCII (ASC) 和定界 ASCII (DEL) 文件分布到这些表中。不能分布 PC/IXF 文件，但可使用 LOAD_ONLY_VERIFY_PART 方式的装入操作将 PC/IXF 文件装入到分布在多个数据库分区的表中。

示例

下列示例说明如何使用 **LOAD** 命令来启动各种类型的装入操作。下列示例中使用的数据库有 5 个数据库分区：0、1、2、3 和 4。每个数据库分区都有一个本地目录 /db2/data/。在数据库分区 0、1、3 和 4 上定义了两个表 TABLE1 和 TABLE2。从客户机装入数据时，用户能够访问除数据库分区以外的远程客户机。

分发和装入示例

在此方案中，您连接到一个数据库分区，该数据库分区可能是也可能不是用来定义 TABLE1 的数据库分区。数据文件 load.del 在此数据库分区的当前工作目录中。要将 load.del 中的数据装入到所有定义了 TABLE1 的数据库分区中，请发出以下命令：

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
```

注：在此示例中，系统对分区数据库环境的所有配置参数使用缺省值：**MODE** 参数缺省为 PARTITION_AND_LOAD。**OUTPUT_DBPARTNUMS** 参数缺省为在其上定义了 TABLE1 的所有数据库分区。**PARTITIONING_DBPARTNUMS** 缺省为根据 **LOAD** 命令规则（用于在未指定任何数据库分区的情况下选择数据库分区）选择的数据库分区集合。

要在数据分布在数据库分区 3 和数据库分区 4 上的位置执行装入操作，请发出以下命令：

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
 PARTITIONED DB CONFIG PARTITIONING_DBPARTNUMS (3,4)
```

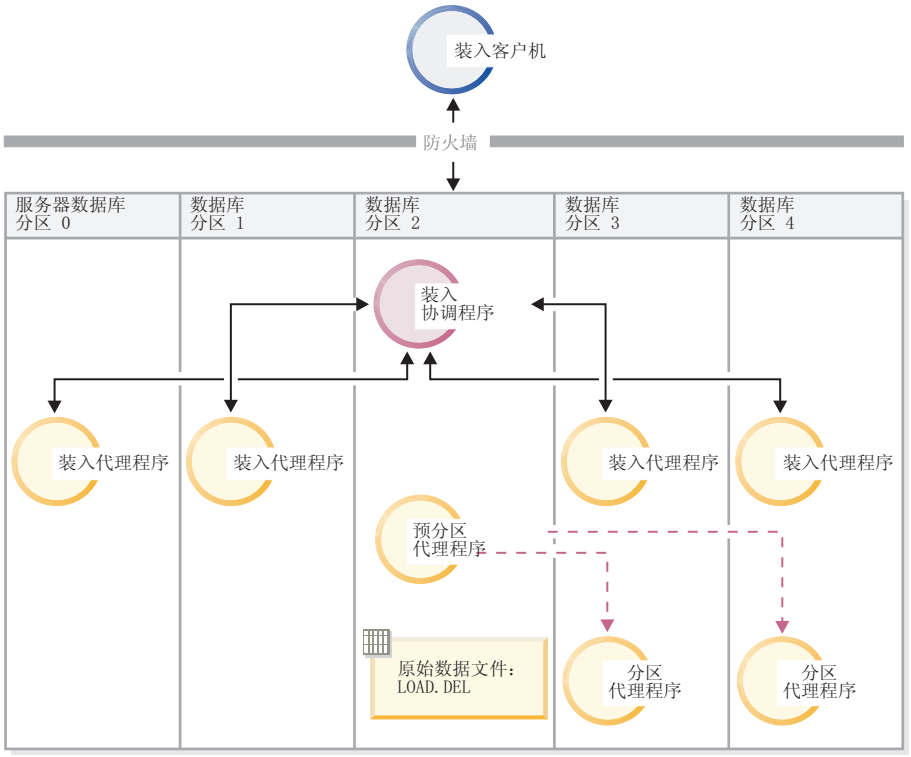


图 40. 将数据装入到数据库分区 3 和 4 中。. 此图说明发出以上命令后产生的行为。数据将装入到数据库分区 3 和 4 中。

仅分发示例

在此方案中，您连接到一个数据库分区，该数据库分区可能是也可能不是用来定义 TABLE1 的数据库分区。数据文件 load.del 在此数据库分区的当前工作目录中。在使用数据库分区 3 和数据库分区 4 的情况下，要将 load.del 分布（而不装入）到所有定义 TABLE1 的数据库分区中，请发出以下命令:

```
LOAD FROM LOAD.DEL of DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE PARTITION_ONLY
 PART_FILE_LOCATION /db2/data
 PARTITIONING_DBPARTNUMS (3,4)
```

这导致将文件 load.del.*xxx* 存储在每个数据库分区上的 /db2/data 目录中，其中 *xxx* 是由 3 位数字表示的数据库分区号。

在仅使用数据库分区 0（缺省 **PARTITIONING_DBPARTNUMS** 值）上运行的 1 个分区代理程序的情况下，要将 load.del 文件分布到数据库分区 1 和 3，请发出以下命令:

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE PARTITION_ONLY
 PART_FILE_LOCATION /db2/data
 OUTPUT_DBPARTNUMS (1,3)
```

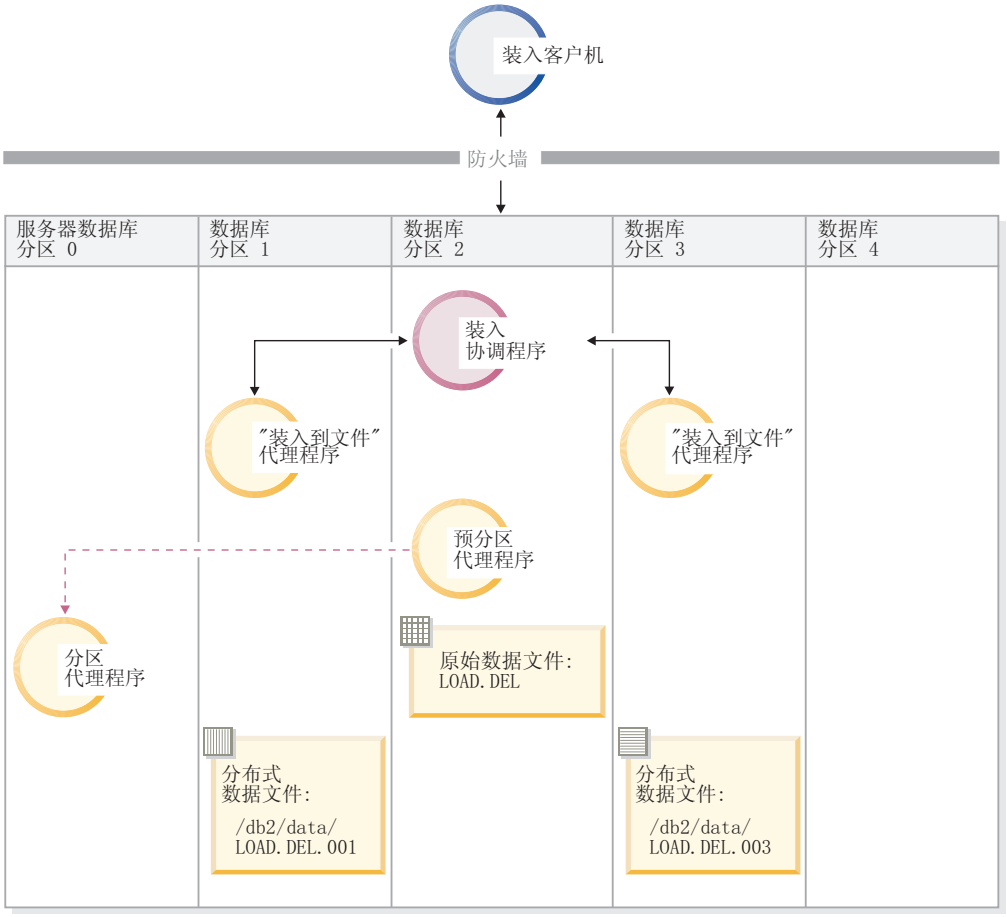


图 41. 使用一个分区代理程序将数据装入到数据库分区 1 和 3 中。. 此图说明发出以上命令后产生的行为。将使用数据库分区 0 上运行的 1 个分区代理程序将数据装入到数据库分区 1 和 3 中。

仅装入示例

如果已经以 PARTITION_ONLY 方式执行了装入操作，并且要将每个装入数据库分区的 /db2/data 目录中的分区文件装入到所有定义了 TABLE1 的数据库分区中，请发出以下命令:

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY
 PART_FILE_LOCATION /db2/data
```

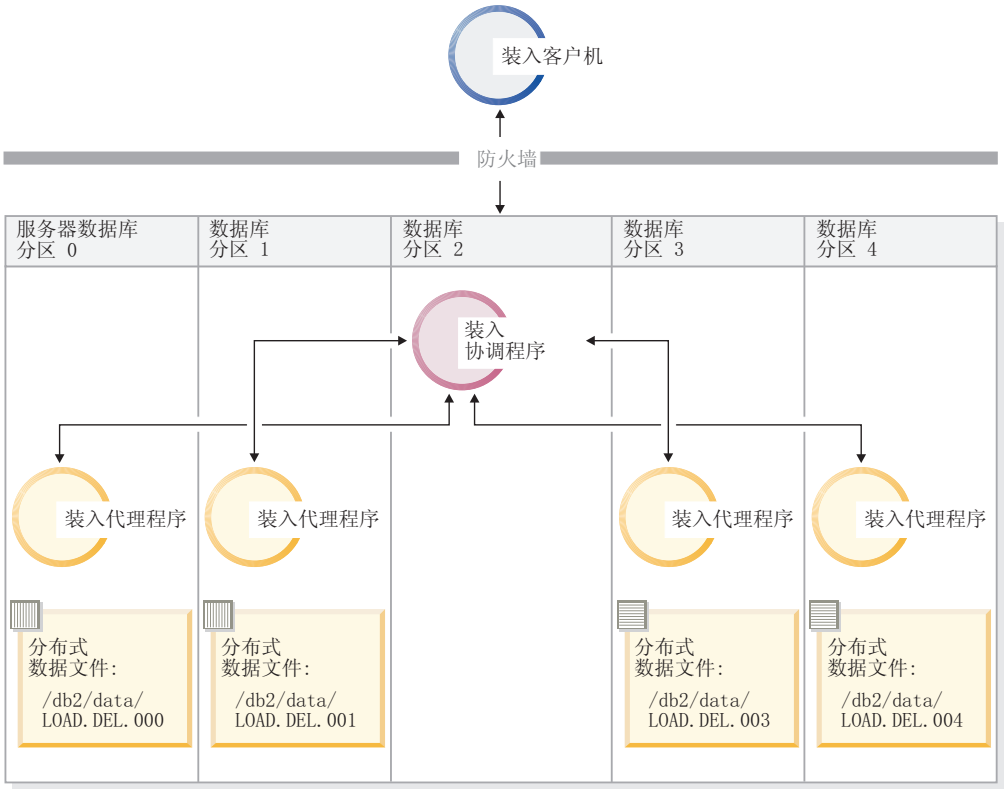


图 42. 将数据装入到其中定义了特定表的所有数据库分区中。. 此图说明发出以上命令后产生的行为。将分布式数据装入到所有用来定义 TABLE1 的数据库分区中。

要仅装入到数据库分区 4 中，请发出以下命令:

```
LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY
 PART_FILE_LOCATION /db2/data
 OUTPUT_DBPARTNUMS (4)
```

装入不带分发映射头的预分发文件

可以使用 **LOAD** 命令来将不带分布头的装入数据文件直接装入到数个数据库分区中。如果数据文件在每个用来定义 TABLE1 的数据库分区上的 /db2/data 目录中，并且名为 load.del.*xxx*（其中 *xxx* 是数据库分区号），那么可以通过发出以下命令来装入那些文件:

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
 PART_FILE_LOCATION /db2/data
```

要仅将数据装入到数据库分区 1 中，请发出以下命令:

```
LOAD FROM LOAD.DEL OF DEL modified by dumpfile=rejected.rows
REPLACE INTO TABLE1
PARTITIONED DB CONFIG MODE LOAD_ONLY_VERIFY_PART
 PART_FILE_LOCATION /db2/data
 OUTPUT_DBPARTNUMS (1)
```

注: 如果指定了转储文件，那么将拒绝装入不属于源数据库分区的行并将它们放入转储文件。

从远程客户机装入到多分区数据库

要将远程客户机上的文件中的数据装入到多分区数据库中，必须指定 **LOAD** 命令的 **CLIENT** 参数。此参数指示数据文件不在服务器分区上。例如:

```
LOAD CLIENT FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

注: 不能将 LOAD_ONLY 或 LOAD_ONLY_VERIFY_PART 方式与 **CLIENT** 参数配合使用。

从游标装入

与在单一分区数据库中一样，可以从游标装入到多分区数据库中。在此示例中，对于 PARTITION_ONLY 和 LOAD_ONLY 方式，**PART_FILE_LOCATION** 参数必须指定标准文件名。此名称是在每个输出数据库分区上创建或装入的分布文件的标准基本文件名。如果目标表包含 LOB 列，那么可以使用指定的基本名称来创建多个文件。

要将语句 SELECT * FROM TABLE1 的应答集中的所有行分布至名为 /db2/data/select.out.*xxx*（其中 *xxx* 是数据库分区号）的每个数据库分区上的文件，以便将来装入到 TABLE2 中，请发出以下命令:

```
DECLARE C1 CURSOR FOR SELECT * FROM TABLE1

LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
PARTITIONED DB CONFIG MODE PARTITION_ONLY
 PART_FILE_LOCATION /db2/data/select.out
```

然后，可以通过发出以下 **LOAD** 命令来装入先前操作所生成的数据文件:

```
LOAD FROM C1 OF CURSOR REPLACE INTO TABLE2
PARTITIONED CB CONFIG MODE LOAD_ONLY
 PART_FILE_LOCATION /db2/data/select.out
```

使用 LOAD QUERY 命令来在分区数据库环境中监视装入操作

在分区数据库环境中执行装入操作期间，某些装入进程会在它们执行时所在的数据库分区上创建消息文件。

这些消息文件存储装入操作执行期间生成的所有参考消息、警告消息和错误消息。用户可以查看以下三个能生成的消息文件的装入进程: 装入代理程序、预分区代理程序和分区代理程序。只有在装入操作完成后，消息文件的内容才可用。

在装入操作期间，可连接至各个数据库分区并对目标表发出 **LOAD QUERY** 命令。从 CLP 中发出此命令时，此命令将显示由 **LOAD QUERY** 命令指定的表的数据库分区上当前存在的装入消息文件的内容。

例如，在数据库 WSDB 中，表 TABLE1 是在数据库分区 0 至 3 上定义的。您将连接至数据库分区 0 并发出以下 **LOAD** 命令:

```
load from load.del of del replace into table1 partitioned db config
    partitioning_dbpartnums (1)
```

此命令将启动装入操作，该操作包括: 在数据库分区 0、1、2 和 3 上运行的装入代理程序; 在数据库分区 1 上运行的分区代理程序; 在数据库分区 0 上运行的预分区代理程序。

数据库分区 0 将包含预分区代理程序的消息文件以及该数据库分区上装入代理程序的消息文件。要同时查看这些文件的内容，请启动新会话并从 CLP 中发出下列命令:

```
set client connect_node 0
connect to wsdb
load query table table1
```

数据库分区 1 将包含装入代理程序的消息文件和分区代理程序的消息文件。要查看这些文件的内容，请启动新会话并从 CLP 中发出下列命令:

```
set client connect_node 1
connect to wsdb
load query table table1
```

注: STATUS_INTERVAL 装入配置选项生成的消息将出现在预分区代理程序消息文件中。要在装入操作期间查看这些消息，必须连接至协调程序分区并发出 **LOAD QUERY** 命令。

保存消息文件的内容

如果通过 **db2Load** API 启动装入操作，那么必须指定消息选项（piLocalMsgFileName），将消息文件从服务器传输到客户机并存储下来以供查看。

对于从 CLP 启动的多分区数据库装入操作来说，不会在控制台上显示或保留消息文件。要在多分区数据库装入完成后保存或查看这些文件的内容，必须指定 **LOAD** 命令的 MESSAGES 选项。如果使用了此选项，那么装入操作一旦完成，就会将每个数据库分区上的消息文件传输到客户机并使用 MESSAGES 选项中指示的基本名称存储在文件中。对于多分区数据库装入操作，下表列示了相应装入进程所生成的文件的名称:

进程类型	文件名
装入代理程序	<*message-file-name*>.LOAD.<*dbpartition-number*>
分区代理程序	<*message-file-name*>.PART.<*dbpartition-number*>
预分区代理程序	<*message-file-name*>.PREP.<*dbpartition-number*>

例如，如果 MESSAGES 选项指定 /wsdb/messages/load，那么数据库分区 2 的装入代理程序消息文件将是 /wsdb/messages/load.LOAD.002。

注: 强烈建议对从 CLP 中启动的多分区数据库装入操作使用 MESSAGES 选项。

在分区数据库环境中继续、重新启动或终止装入操作

如果装入操作在分区数据库环境中失败，那么接下来需要执行的操作取决于出现故障的时间。

多分区数据库中的装入过程由两个阶段组成:

1. *设置阶段*，在该阶段中获取数据库分区级别的资源，例如，输出数据库分区上的表锁定

   通常，如果设置阶段发生故障，不必执行重新启动和终止操作。您需要执行的操作取决于对失败的装入操作指定的错误隔离方式。

   如果装入操作指定了不隔离设置阶段的错误，那么将取消整个装入操作并且每个数据库分区上表的状态都将回滚到该表在执行装入操作之前所处的状态。

如果装入操作指定了要隔离设置阶段的错误，那么将在设置阶段已成功完成的数据库分区上继续执行装入操作，但每个失败数据库分区上的表都将回滚到该表在执行装入操作之前所处的状态。这意味着如果某些分区在设置阶段失败，而其他分区在装入阶段失败，那么单个装入操作可能在不同阶段失败。

2. *装入阶段*，在该阶段中格式化数据并将它们装入到数据库分区上的表中。

   在多分区数据库装入操作的装入阶段，如果装入操作在至少一个数据库分区上失败，那么必须发出 **LOAD RESTART** 或 **LOAD TERMINATE** 命令。因为多分区数据库中的数据装入操作将通过单个事务完成，所以有必要执行此操作。

   如果您可以解决导致装入操作失败的问题，那么选择 **LOAD RESTART**。这样可以节省时间，因为在启动了装入重新启动操作时，装入操作会在所有数据库分区上的中断位置继续执行。

   如果您希望表返回到该表在初始装入操作前所处的状态，那么选择 **LOAD TERMINATE**。

确定装入失败的时间

如果装入操作在分区环境中失败，那么您需要做的第一件事是确定操作在哪个分区上失败以及每个操作在哪个阶段失败。这可通过查看分区摘要来完成。如果从 CLP 发出了 **LOAD** 命令，那么在结束装入时将显示分区摘要（请参阅以下示例）。如果从 db2Load API 发出 **LOAD** 命令，那么分区摘要包含在 db2PartLoadOut 结构的 **poAgentInfoList** 字段中。

如果对于给定分区，“代理程序类型”有一个“LOAD”条目，那么该分区已到达装入阶段，否则在设置阶段出现故障。SQL 代码为负数表示失败。在以下示例中，装入操作在装入阶段在分区 1 上失败。

```
代理程序类型      节点      SQL 代码      结果
_____________________________________________________________
LOAD            000       +00000000     Success.
_____________________________________________________________
LOAD            001       -00000289     Error. May require RESTART.
_____________________________________________________________
LOAD            002       +00000000     Success.
_____________________________________________________________
LOAD            003       +00000000     Success.
.
.
.
```

恢复、重新启动或终止失败的装入操作

在设置阶段，只有 ISOLATE_PART_ERRS 选项指定为 SETUP_ERRS_ONLY 或 SETUP_AND_LOAD_ERRS 的装入才会失败。对于在此阶段在至少一个输出数据库分区上失败的装入来说，可以发出 **LOAD REPLACE** 或 **LOAD INSERT** 命令。使用 OUTPUT_DBPARTNUMS 选项来仅指定装入操作在其上失败的那些数据库分区。

对于在装入阶段在至少一个输出数据库分区上失败的装入来说，发出 **LOAD RESTART** 或 **LOAD TERMINATE** 命令。

对于在设置阶段在至少一个输出数据库分区上失败并且在装入阶段在至少一个输出数据库分区上失败的装入来说，需要执行两个装入操作以继续失败的装入 - 一个装入操作用于设置阶段故障，另一个用于装入阶段故障，如上所述。要有效地撤销此类型失败

的装入操作，发出 **LOAD TERMINATE** 命令。但是，在发出该命令后，您必须说明所有分区，因为没有对在设置阶段失败的分区上的表进行任何更改，并且已撤销在装入阶段失败的分区的所有更改。

例如，在数据库 WSDB 中，TABLE1 是在数据库分区 0 至 3 上定义的。发出以下命令:

```
load from load.del of del insert into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

在设置阶段，输出数据库分区 1 上出现故障。由于隔离了设置阶段错误，所以装入操作继续，但在装入阶段在分区 3 上出现故障。要继续装入操作，应发出下列命令:

```
load from load.del of del replace into table1 partitioned db config
output_dbpartnums (1)

load from load.del of del restart into table1 partitioned db config
isolate_part_errs setup_and_load_errs
```

注: 对于装入重新启动操作，将使用在 **LOAD RESTART** 命令中指定的选项，因此，这些选项应该与原始 **LOAD** 命令中指定的选项完全相同，这一点非常重要。

为分区数据库环境装入配置选项

有许多可用来修改分区数据库环境中的装入操作的配置选项。

MODE *X*

指定装入多分区数据库时装入操作采用的方式。PARTITION_AND_LOAD 是缺省值。有效值为:

- PARTITION_AND_LOAD。对数据进行分布（有可能以并行方式进行分布），并且同时在各个相应数据库分区上装入数据。
- PARTITION_ONLY。对数据进行分布（有可能以并行方式进行分布），并将输出写入每个装入数据库分区上指定位置中的文件。对于 CURSOR 以外的文件类型，每个数据库分区上的输出文件名格式为 *filename.xxx*，其中 *filename* 是 **LOAD** 命令中指定的输入文件名，*xxx* 是 3 位的数据库分区号。对于 CURSOR 文件类型，每个数据库分区上的输出文件名由 PART_FILE_LOCATION 选项确定。请参阅 PART_FILE_LOCATION 选项以了解有关如何指定每个数据库分区的分布文件位置的详细信息。

  注:

  1. 此方式不能用于 CLI 装入操作。
  2. 如果表包含进行分布时所需的标识列，那么除非指定了 **identityoverride** 文件类型修饰符，否则不支持此方式。
  3. 为文件类型 CURSOR 生成的分布文件在 DB2 发行版之间不兼容。这意味着不能使用 LOAD_ONLY 方式装入在先前发行版中生成的文件类型为 CURSOR 的分布文件。同样，不能使用 LOAD_ONLY 方式在将来发行版中装入在当前发行版中生成的文件类型为 CURSOR 的分布文件。

- LOAD_ONLY。假定已对数据进行分布；将跳过分布过程，并且在相应的数据库分区上同时装入数据。对于 CURSOR 以外的文件类型，每个数据库分区的输入文件名格式应该是 *filename.xxx*，其中 *filename* 是 **LOAD** 命令中指定的文件名，*xxx* 是 3 位的数据库分区号。对于 CURSOR 文件类型，每个数据库分区上的输入文件

名由 PART_FILE_LOCATION 选项确定。请参阅 PART_FILE_LOCATION 选项以了解有关如何指定每个数据库分区的分布文件位置的详细信息。

注:

1. 此方式不能用于 CLI 装入操作；或者当指定了 **LOAD** 命令的 **CLIENT** 参数时，也不能使用此方式。
2. 如果表包含进行分布时所需的标识列，那么除非指定了 **identityoverride** 文件类型修饰符，否则不支持此方式。

- LOAD_ONLY_VERIFY_PART。假定已对数据进行分布，但数据文件未包含分区头。将跳过分布过程，并且在相应的数据库分区上同时装入数据。在装入操作期间，将检查每一行以验证它是否在正确的数据库分区中。如果指定了 **dumpfile** 文件类型修饰符，那么会将发生数据库分区违例的行放到转储文件中。否则会删除那些行。如果特定装入数据库分区上存在数据库分区违例，那么会将一条有关该数据库分区的警告写至装入消息文件。每个数据库分区的输入文件名格式应该是 *filename.xxx*，其中 *filename* 是 **LOAD** 命令中指定的文件名，*xxx* 是 3 位的数据库分区号。请参阅 PART_FILE_LOCATION 选项以了解有关如何指定每个数据库分区的分布文件位置的详细信息。

  注:

  1. 此方式不能用于 CLI 装入操作；或者当指定了 **LOAD** 命令的 **CLIENT** 参数时，也不能使用此方式。
  2. 如果表包含进行分布时所需的标识列，那么除非指定了 **identityoverride** 文件类型修饰符，否则不支持此方式。
- ANALYZE。生成最佳分发映射（在所有数据库分区之间均匀地分布数据）。

PART_FILE_LOCATION *X*

在 PARTITION_ONLY、LOAD_ONLY 和 LOAD_ONLY_VERIFY_PART 方式下，此参数可用来指定分布文件的位置。在 OUTPUT_DBPARTNUMS 选项指定的每个数据库分区上，此位置必须存在。如果指定的位置是相对路径名，那么会将该路径追加至当前目录以创建分布式文件位置。

对于 CURSOR 文件类型，必须指定此选项，并且位置必须引用标准文件名。在 PARTITION_ONLY 方式下，此名称是在每个输出数据库分区上创建的分布式文件的标准基本文件名，或者在 LOAD_ONLY 方式，此名称是对于每个数据库分区要读取的文件的位置。使用 PARTITION_ONLY 方式时，如果目标表包含 LOB 列，那么可以使用指定的基本名称来创建多个文件。

对于 CURSOR 以外的文件类型，如果未指定此选项，则将使用当前目录来存储分布式文件。

OUTPUT_DBPARTNUMS *X*

X 表示数据库分区号列表。数据库分区号表示要执行装入操作的数据库分区。数据库分区号必须是定义了该表的数据库分区的子集。缺省情况下，选择了所有数据库分区。必须将此列表括在圆括号中，并且列表项必须由逗号分隔。允许指定范围（例如，(0, 2 to 10, 15)）。

PARTITIONING_DBPARTNUMS *X*

X 表示分布过程中使用的数据库分区号列表。必须将此列表括在圆括号中，并且列表项必须由逗号分隔。允许指定范围（例如，(0, 2 to 10, 15)）。对分布过程指定的数据库分区可能与要装入的数据库分区不同。如果未指定

PARTITIONING_DBPARTNUMS，那么 LOAD 实用程序会确定需要的数据库分区数以及为获得最优性能而需要使用的数据库分区。

如果在 **LOAD** 命令中未指定 **anyorder** 文件类型修饰符，那么在装入会话中将只使用一个分区代理程序。此外，如果仅对 OUTPUT_DBPARTNUMS 选项指定了一个数据库分区，或者装入操作的协调程序分区不是 OUTPUT_DBPARTNUMS 的元素，那么会在分布过程中使用装入操作的协调程序分区。否则，在分布过程中使用 OUTPUT_DBPARTNUMS 中的第一个数据库分区（不是协调程序分区）。

如果指定了 **anyorder** 文件类型修饰符，那么按以下方式确定分布过程中使用的数据库分区数：(OUTPUT_DBPARTNUMS 中的分区数/4 + 1)。

MAX_NUM_PART_AGENTS *X*
指定装入会话中要使用的最大分区代理程序数。缺省值为 25。

ISOLATE_PART_ERRS *X*
指示装入操作如何对各个数据库分区上发生的错误作出反应。除非同时指定了 **LOAD** 命令的 ALLOW READ ACCESS 和 **COPY YES** 参数（在此情况下，缺省值为 **NO_ISOLATION**），否则缺省值为 LOAD_ERRS_ONLY。有效值为：

- SETUP_ERRS_ONLY。设置期间在数据库分区上发生的错误（例如，访问数据库分区时发生的问题，或者访问数据库分区上的表空间或表时发生的问题）将导致装入操作在发生故障的数据库分区上停止运行，但在其余数据库分区上继续运行。装入数据时在数据库分区上发生的错误将导致整个操作失败。
- LOAD_ERRS_ONLY。设置期间在数据库分区上发生的错误将导致整个装入操作失败。如果在装入数据时发生错误，那么装入操作将在出错的数据库分区上停止运行。装入操作将在其余数据库分区上继续运行，直到发生故障或者装入了所有数据为止。在执行装入重新启动操作并成功完成之前，新装入的数据将不可视。

  注：在同时指定了 **LOAD** 命令的 **ALLOW READ ACCESS** 和 **COPY YES** 参数时，不能使用此方式。
- SETUP_AND_LOAD_ERRS。在此方式下，设置期间或装入数据期间发生的数据库分区级别错误将导致仅在受影响的数据库分区上停止处理装入操作。对于 LOAD_ERRS_ONLY 方式，如果在装入数据时发生分区错误，那么在执行装入重新启动操作并成功完成之前，新装入的数据将不可视。

  注：在同时指定了 **LOAD** 命令的 **ALLOW READ ACCESS** 和 **COPY YES** 选项时，不能使用此方式。
- NO_ISOLATION。装入操作期间发生的任何错误都会导致装入操作失败。

STATUS_INTERVAL *X*
X 表示读取多少数据量时发出通知。计量单位是兆字节 (MB)。缺省值为 100 MB。有效值是 1 到 4000 之间的整数。

PORT_RANGE *X*
X 表示用来创建内部通信套接字的 TCP 端口的范围。缺省范围是 49152 到 65535。如果在调用时定义了 **DB2ATLD_PORTS** 注册表变量的值，那么该值将替换 PORT_RANGE 装入配置选项的值。对于 **DB2ATLD_PORTS** 注册表变量，应该使用以下格式来提供范围：

<lower-port-number:higher-port-number>

在 CLP 中，格式为：

(*lower-port-number*, *higher-port-number*)

CHECK_TRUNCATION

指定程序应该在输入/输出时检查数据记录截断情况。缺省行为是: 输入/输出时不检查数据截断情况。

MAP_FILE_INPUT *X*

X 指定分发映射的输入文件名。由于此参数指向包含定制分发映射的文件，所以，如果使用定制分发映射，就必须指定此参数。通过使用 **db2gpmap** 程序从数据库系统目录表中抽取映射，或者使用 **LOAD** 命令的 ANALYZE 方式来生成最佳映射，可以创建定制分发映射。必须先将使用 ANALYZE 方式生成的映射移至数据库中的每个数据库分区，这样装入操作才能继续运行。

MAP_FILE_OUTPUT *X*

X 表示分发映射的输出文件名。将在发出 **LOAD** 命令的数据库分区上创建输出文件（假定执行分区操作的数据库分区组包含该数据库分区）。如果在未参与分区的数据库分区（由 **PARTITIONING_DBPARTNUMS** 定义）上调用 **LOAD** 命令，那么会在使用 **PARTITIONING_DBPARTNUMS** 参数定义的第一个数据库分区上创建输出文件。考虑以下分区数据库环境设置:

```
1 serv1 0
2 serv1 1
3 serv2 0
4 serv2 1
5 serv3 0
```

在 serv3 上运行以下 **LOAD** 命令将在 serv1 上创建分布图。

```
LOAD FROM file OF ASC METHOD L ( ...) INSERT INTO table CONFIG
MODE ANALYZE PARTITIONING_DBPARTNUMS(1,2,3,4)
MAP_FILE_OUTPUT '/home/db2user/distribution.map'
```

指定了 ANALYZE 方式时，应该使用此参数。生成最佳分发映射（在所有数据库分区之间均匀地分布数据）。如果未指定此参数但指定了 ANALYZE 方式，那么程序将出错并退出。

TRACE *X*

当您要求复查数据转换过程转储和散列值输出时，指定要跟踪的记录的个数。缺省值为 0。

NEWLINE

当输入数据文件是 ASC 文件（各个记录由换行符定界），并且在 **LOAD** 命令中指定了 **reclen** 文件类型修饰符时，使用此选项。当指定了此选项时，将对每个记录检查换行符。还将检查 **reclen** 文件类型修饰符中指定的记录长度。

DISTFILE *X*

如果指定了此选项，那么 LOAD 实用程序将生成具有给定名称的数据库分区分布文件。数据库分区分布文件包含 32 768 个整数: 目标表分发映射中的每个条目都有一个对应的整数。此文件中的每个整数都表示所装入输入文件中被分散到相应分发映射条目的行数。此信息可以帮助您标识数据偏差，并且还可以帮助您确定是否应该使用实用程序的 ANALYZE 方式来生成表的新分发映射。如果未指定此选项，那么 LOAD 实用程序的缺省行为是不生成分布文件。

注: 当指定了此选项时，对装入操作最多使用一个分区代理程序。即使您显式请求多个分区代理程序，也只使用一个。

OMIT_HEADER
指定在分布文件中不应包括分发映射头。如果未指定，那么生成头。

RUN_STAT_DBPARTNUM *X*
如果在 **LOAD** 命令中指定了 **STATISTICS USE PROFILE** 参数，那么将只在一个数据库分区上收集统计信息。此参数指定要收集统计信息的数据库分区。如果值为 -1，或者根本未指定值，那么将在输出数据库分区列表中的第一个数据库分区上收集统计信息。

分区数据库环境中的装入会话 - CLP 示例

下列示例说明如何在多分区数据库中装入数据。

数据库有四个数据库分区，其编号从 0 到 3。数据库 WSDB 是在所有数据库分区上定义的，表 TABLE1 在缺省数据库分区组中，该数据库分区组也是在所有数据库分区上定义的。

示例 1

要将用户数据文件 load.del 中的数据装入到 TABLE1 中（该文件在数据库分区 0 上），请连接到数据库分区 0，然后发出以下命令:

```
load from load.del of del replace into table1
```

如果装入操作成功，那么输出将如下所示:

```
   代理程序类型      节点      SQL 代码      结果
------------------------------------------------------------
   LOAD            000       +00000000     Success.
------------------------------------------------------------
   LOAD            001       +00000000     Success.
------------------------------------------------------------
   LOAD            002       +00000000     Success.
------------------------------------------------------------
   LOAD            003       +00000000     Success.
------------------------------------------------------------
PARTITION       001       +00000000     Success.
------------------------------------------------------------
PRE_PARTITION   000       +00000000     Success.
------------------------------------------------------------
   结果:          成功完成了 4 个装入。
------------------------------------------------------------

   分区代理程序的摘要:
   读取的行数                    = 100000
   拒绝的行数                    = 0
   分区的行数                    = 100000

   装入代理程序的摘要:
   读取的行数                    = 100000
   跳过的行数                    = 0
   装入的行数                    = 100000
   拒绝的行数                    = 0
   删除的行数                    = 0
   落实的行数                    = 100000
```

输出指示在每个数据库分区上有一个装入代理程序，并且每个装入代理程序都运行成功。输出还显示在协调程序分区上运行了一个预分区代理程序，在数据库分区 1 上运行了一个分区代理程序。这些进程都成功完成并返回正常 SQL 返回码 0。统计摘要显示

预分区代理程序读取了 100,000 行，分区代理程序分布了 100,000 行，装入代理程序装入的总行数为 100,000。

示例 2

在以下示例中，以 PARTITION_ONLY 方式将数据装入到 TABLE1 中。分布式输出文件存储在每个输出数据库分区上的 /db/data 目录中:

```
load from load.del of del replace into table1 partitioned db config mode
partition_only part_file_location /db/data
```

以上 LOAD 命令的输出如下所示:

```
代理程序类型      节点      SQL 代码      结果
------------------------------------------------------------
LOAD_TO_FILE    000       +00000000     Success.
------------------------------------------------------------
LOAD_TO_FILE    001       +00000000     Success.
------------------------------------------------------------
LOAD_TO_FILE    002       +00000000     Success.
------------------------------------------------------------
LOAD_TO_FILE    003       +00000000     Success.
------------------------------------------------------------
PARTITION       001       +00000000     Success.
------------------------------------------------------------
PRE_PARTITION   000       +00000000     Success.
------------------------------------------------------------

分区代理程序的摘要:
读取的行数                  = 100000
拒绝的行数                  = 0
分区的行数                  = 100000
```

输出指示在每个输出数据库分区上都运行了“装入到文件”代理程序，这些代理程序运行成功。在协调程序分区上运行了一个预分区代理程序，在数据库分区 1 上运行了一个分区代理程序。统计摘要显示预分区代理程序成功读取了 100,000 行，分区代理程序成功分布了 100,000 行。由于未将任何行装入到表中，因此未显示已装入行数摘要。

示例 3

要装入在先前显示的 PARTITION_ONLY 装入操作期间生成的文件，请发出以下命令:

```
load from load.del of del replace into table1 partitioned db config mode
    load_only part_file_location /db/data
```

load 命令的输出如下所示:

```
代理程序类型      节点      SQL 代码      结果
------------------------------------------------------------
LOAD            000       +00000000     Success.
------------------------------------------------------------
LOAD            001       +00000000     Success.
------------------------------------------------------------
LOAD            002       +00000000     Success.
------------------------------------------------------------
LOAD            003       +00000000     Success.
------------------------------------------------------------
结果:          成功完成了 4 个装入。
------------------------------------------------------------

装入代理程序的摘要:
读取的行数                  = 100000
跳过的行数                  = 0
装入的行数                  = 100000
```

```
拒绝的行数                  = 0
删除的行数                  = 0
落实的行数                  = 100000
```

此输出显示每个输出数据库分区上的装入代理程序都运行成功，并且所有装入代理程序装入的总行数为 100,000。由于未执行分布操作，因此未显示分布行数摘要。

示例 4

如果发出以下 LOAD 命令:

```
load from load.del of del replace into table1
```

并且其中一个装入数据库分区在装入操作期间耗尽表空间，那么将返回以下输出:

```
SQL0289N  不能在表空间"DMS4KT"中分配新页。
SQLSTATE=57011

代理程序类型      节点      SQL 代码      结果
_____________________________________________________________________
LOAD            000       +00000000     Success.
_____________________________________________________________________
LOAD            001       -00000289     Error. May require RESTART.
_____________________________________________________________________
LOAD            002       +00000000     Success.
_____________________________________________________________________
LOAD            003       +00000000     Success.
_____________________________________________________________________
PARTITION       001       +00000000     Success.
_____________________________________________________________________
PRE_PARTITION   000       +00000000     Success.
_____________________________________________________________________
结果:          成功完成了 4 个装入中的 3 个。
_____________________________________________________________________

分区代理程序的摘要:
读取的行数                  = 0
拒绝的行数                  = 0
分区的行数                  = 0

装入代理程序的摘要:
读取的行数                  = 0
跳过的行数                  = 0
装入的行数                  = 0
拒绝的行数                  = 0
删除的行数                  = 0
已落实的行数                = 0
```

输出指示装入操作返回了错误 SQL0289。数据库分区摘要指示数据库分区 1 耗尽空间。如果对数据库分区 1 上的表空间容器添加了更多空间，那么可以按如下方式重新启动装入操作:

```
load from load.del of del restart into table1
```

迁移和版本兼容性

在多分区数据库中，**DB2_PARTITIONEDLOAD_DEFAULT** 注册表变量可用来还原为 DB2 Universal Database™ V8 之前的装入行为。

注: 不推荐使用 **DB2_PARTITIONEDLOAD_DEFAULT** 注册表变量并且在以后的发行版中可能会将其除去。

通过在多分区数据库中还原为 **LOAD** 命令的 DB2 UDB V8 之前的行为，可以将带有有效分布头的文件装入到单一数据库分区中，而不必指定任何其他分区数据库配置选项。将 **DB2_PARTITIONEDLOAD_DEFAULT** 的值设置为 NO 可以达到此目的。如果要避免修改对单一数据库分区发出 **LOAD** 命令的现有脚本，那么可以选择使用此选项。例如，要将分布文件装入到一个表的数据库分区 3 中（而该表所在的数据库分区组包含 4 个数据库分区），请发出以下命令:

```
db2set DB2_PARTITIONEDLOAD_DEFAULT=NO
```

然后，从 DB2 命令行处理器中发出下列命令:

```
CONNECT RESET

   SET CLIENT CONNECT_NODE 3

   CONNECT TO DB MYDB

   LOAD FROM LOAD.DEL OF DEL REPLACE INTO TABLE1
```

在多分区数据库中，当未指定多分区数据库装入配置选项时，将在所有用来定义该表的数据库分区上执行装入操作。输入文件不需要分布头，并且 MODE 选项缺省为 PARTITION_AND_LOAD。要装入单一数据库分区，必须指定 OUTPUT_DBPARTNUMS 选项。

第 16 章 分区数据库的迁移环境

迁移分区数据库

迁移分区数据库环境要求您在所有数据库分区服务器上安装最新发行版的数据库产品，迁移这些实例，然后迁移这些数据库。

可以从目录数据库分区服务器或任何其他数据库分区服务器迁移数据库分区服务器。如果迁移过程失败，那么可以从目录数据库分区服务器或任何其他数据库分区服务器重新尝试迁移。

因为这种类型的迁移是一项重大任务（对迁移过程的描述），所以其先决条件和限制超出了本书的范围。《*迁移指南*》中的『迁移分区数据库环境』主题中不仅提供了详细描述，还提供许多其他在执行迁移之前应查看的参考主题。

第 17 章 使用快照和事件监视器

使用快照监视器数据来监视分区表的重组

下列信息描述了一些最有用的全局表重组状态监视方法。

关于此任务

没有单独的用于指示分区表整体表重组状态的数据组。分区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。但是，可以根据所重组的各个数据分区数据组中的元素值来推断全局表重组状态。下列信息描述了一些最有用的全局表重组状态监视方法。

确定要重组的数据分区数

通过计算表名和模式名相同的表数据监视器数据块数，可以确定表中重组的数据分区总数。此值指示启动了重组的数据分区数。示例 1 和 2 指示正在对 3 个数据分区进行重组。

标识要重组的数据分区

可以根据阶段开始时间（reorg_phase_start）来推断当前正在重组的数据分区。在 SORT/BUILD/REPLACE 阶段，与正在重组的数据分区相对应的监视器数据显示了最新阶段开始时间。在 INDEX_RECREATE 阶段，所有数据分区的阶段开始时间都是相同的。在示例 1 和 2 中，指示了 INDEX_RECREATE 阶段，因此所有数据分区的开始时间都是相同的。

标识索引重建要求

通过获取与任何一个正在重组的数据分区相对应的最大重组阶段数（reorg_max_phase）元素值，可以确定是否需要重建索引。如果 reorg_max_phase 值为 3 或 4，那么表示需要重建索引。示例 1 和 2 报告的 reorg_max_phase 值为 3，即表示需要重建索引。

示例

以下样本输出来自一台 3 节点服务器，该服务器包含一个带有 3 个数据分区的表:

```
CREATE TABLE  sales (c1 INT, c2 INT, c3 INT)
    PARTITION BY RANGE (c1)
                (PART P1 STARTING FROM (1)  ENDING AT (10) IN parttbs,
         PART P2 STARTING FROM (11) ENDING AT (20) IN parttbs,
         PART P3 STARTING FROM (21) ENDING AT (30) IN parttbs)
    DISTRIBUTE BY (c2)
```

执行的语句:

```
REORG TABLE sales ALLOW NO ACCESS ON ALL DBPARTITIONNUMS
```

示例 1:

```
GET SNAPSHOT FOR TABLES ON DPARTDB GLOBAL
```

已将输出修改为仅包括相关表的表信息。

```
                表快照

第一个数据库连接时间戳记                = 06/28/2005 13:46:43.061690
```

```
上次重置时间戳记                        = 06/28/2005 13:46:47.440046
快照时间戳记                            = 06/28/2005 13:46:50.964033
数据库名称                              = DPARTDB
数据库路径                              = /work/sales/NODE0000/SQL00001/
输入数据库别名                          = DPARTDB
已访问的表的数目                        = 5

表列表
 表模式                 = NEWTON
 表名                   = SALES
 表类型                 = 用户
 数据分区标识           = 0
 数据对象页             = 3
  读取的行数             = 12
 写入的行数             = 1
 溢出数                 = 0
 重组的页数             = 0
 表重组信息:
   节点号                                = 0
   重组类型             =
        回收
        表重组
        不允许访问
        通过表扫描重新集群
        仅重组数据
   重组索引数           = 0
   重组表空间数         = 3
 长临时空间标识         = 3
   开始时间             = 06/28/2005 13:46:49.816883
   重组阶段             = 3 - 索引重建
   最大阶段             = 3
   阶段开始时间         = 06/28/2005 13:46:50.362918
   状态                 = 已完成
   当前计数器           = 0
   最大计数器           = 0
   完成                 = 0
   结束时间             = 06/28/2005 13:46:50.821244

 表重组信息:
   节点数               = 1
   重组类型             =
        回收
        表重组
        不允许访问
        通过表扫描重新集群
        仅重组数据
   重组索引数           = 0
   重组表空间数         = 3
 长临时空间标识         = 3
   开始时间             = 06/28/2005 13:46:49.822701
   重组阶段             = 3 - 索引重建
   最大阶段             = 3
   阶段开始时间         = 06/28/2005 13:46:50.420741
   状态                 = 已完成
   当前计数器           = 0
   最大计数器           = 0
   完成                 = 0
   结束时间             = 06/28/2005 13:46:50.899543


 表重组信息:
   节点数               = 2
   重组类型             =
        回收
        表重组
        不允许访问
        通过表扫描重新集群
```

```
        仅重组数据
  重组索引数          = 0
  重组表空间数        = 3
长临时空间标识        = 3
  开始时间            = 06/28/2005 13:46:49.814813
  重组阶段            = 3 - 索引重建
  最大阶段            = 3
  阶段开始时间        = 06/28/2005 13:46:50.344277
  状态                = 已完成
  当前计数器          = 0
  最大计数器          = 0
  完成                = 0
  结束时间            = 06/28/2005 13:46:50.803619


表模式                = NEWTON
表名                  = SALES
表类型                = 用户
数据分区标识          = 1
数据对象页            = 3
读取的行数            = 8
写入的行数            = 1
溢出数                = 0
重组的页数            = 0
表重组信息:
  节点号                              = 0
  重组类型            =
        回收
        表重组
        不允许访问
        通过表扫描重新集群
        仅重组数据
  重组索引数          = 0
  重组表空间数        = 3
长临时空间标识        = 3
  开始时间            = 06/28/2005 13:46:50.014617
  重组阶段            = 3 - 索引重建
  最大阶段            = 3
  阶段开始时间        = 06/28/2005 13:46:50.362918
  状态                = 已完成
  当前计数器          = 0
  最大计数器          = 0
  完成                = 0
  结束时间            = 06/28/2005 13:46:50.821244

表重组信息:
  节点数              = 1
  重组类型            =
        回收
        表重组
        不允许访问
        通过表扫描重新集群
        仅重组数据
  重组索引数          = 0
  重组表空间数        = 3
长临时空间标识        = 3
  开始时间            = 06/28/2005 13:46:50.026278
  重组阶段            = 3 - 索引重建
  最大阶段            = 3
  阶段开始时间        = 06/28/2005 13:46:50.420741
  状态                = 已完成
  当前计数器          = 0
  最大计数器          = 0
  完成                = 0
  结束时间            = 06/28/2005 13:46:50.899543

表重组信息:
```

```
  节点数             = 2
  重组类型           =
       回收
       表重组
       不允许访问
       通过表扫描重新集群
       仅重组数据
  重组索引数         = 0
  重组表空间数       = 3
长临时空间标识       = 3
  开始时间           = 06/28/2005 13:46:50.006392
  重组阶段           = 3 - 索引重建
  最大阶段           = 3
  阶段开始时间       = 06/28/2005 13:46:50.344277
  状态               = 已完成
  当前计数器         = 0
  最大计数器         = 0
  完成               = 0
  结束时间           = 06/28/2005 13:46:50.803619


表模式               = NEWTON
表名                 = SALES
表类型               = 用户
数据分区标识         = 2
数据对象页           = 3
读取的行数         = 4
写入的行数           = 1
溢出数               = 0
重组的页数           = 0
表重组信息:
  节点号                           = 0
  重组类型           =
       回收
       表重组
       不允许访问
       通过表扫描重新集群
       仅重组数据
  重组索引数         = 0
  重组表空间数       = 3
长临时空间标识       = 3
  开始时间           = 06/28/2005 13:46:50.199971
  重组阶段           = 3 - 索引重建
  最大阶段           = 3
  阶段开始时间       = 06/28/2005 13:46:50.362918
  状态               = 已完成
  当前计数器         = 0
  最大计数器         = 0
  完成               = 0
  结束时间           = 06/28/2005 13:46:50.821244

表重组信息:
  节点数             = 1
  重组类型           =
       回收
       表重组
       不允许访问
       通过表扫描重新集群
       仅重组数据
  重组索引数         = 0
  重组表空间数       = 3
长临时空间标识       = 3
  开始时间           = 06/28/2005 13:46:50.223742
  重组阶段           = 3 - 索引重建
  最大阶段           = 3
  阶段开始时间       = 06/28/2005 13:46:50.420741
  状态               = 已完成
```

```
  当前计数器         = 0
  最大计数器         = 0
  完成               = 0
  结束时间           = 06/28/2005 13:46:50.899543

表重组信息:
  节点数             = 2
  重组类型           =
       回收
       表重组
       不允许访问
       通过表扫描重新集群
       仅重组数据
  重组索引数         = 0
  重组表空间数       = 3
长临时空间标识       = 3
  开始时间           = 06/28/2005 13:46:50.179922
  重组阶段           = 3 - 索引重建
  最大阶段           = 3
  阶段开始时间       = 06/28/2005 13:46:50.344277
  状态               = 已完成
  当前计数器         = 0
  最大计数器         = 0
  完成               = 0
  结束时间           = 06/28/2005 13:46:50.803619
```

示例 2:

```
GET SNAPSHOT FOR TABLES ON DPARTDB AT DBPARTITIONNUM 2
```

已将输出修改为仅包括相关表的表信息。

```
                表快照

第一个数据库连接时间戳记              = 06/28/2005 13:46:43.617833
上次重置时间戳记                       =
快照时间戳记                          = 06/28/2005 13:46:51.016787
数据库名称                            = DPARTDB
数据库路径                            = /work/sales/NODE0000/SQL00001/
输入数据库别名                        = DPARTDB
已访问的表的数目                      = 3

表列表
 表模式               = NEWTON
 表名                 = SALES
 表类型               = 用户
 数据分区标识         = 0
 数据对象页           = 1
 读取的行数                  = 0
 写入的行数           = 0
 溢出数               = 0
 重组的页数           = 0
 表重组信息:
   节点数             = 2
   重组类型           =
        回收
        表重组
        不允许访问
        通过表扫描重新集群
        仅重组数据
   重组索引数         = 0
   重组表空间数       = 3
 长临时空间标识       = 3
   开始时间           = 06/28/2005 13:46:49.814813
   重组阶段           = 3 - 索引重建
   最大阶段           = 3
   阶段开始时间       = 06/28/2005 13:46:50.344277
```

```
  状态               = 已完成
  当前计数器         = 0
  最大计数器         = 0
  完成               = 0
  结束时间           = 06/28/2005 13:46:50.803619


表模式               = NEWTON
表名                 = SALES
表类型               = 用户
数据分区标识         = 1
数据对象页           = 1
读取的行数                 = 0
写入的行数           = 0
溢出数               = 0
重组的页数           = 0
表重组信息:
  节点数             = 2
  重组类型           =
       回收
       表重组
       不允许访问
       通过表扫描重新集群
       仅重组数据
  重组索引数         = 0
  重组表空间数       = 3
长临时空间标识       = 3
  开始时间           = 06/28/2005 13:46:50.006392
  重组阶段           = 3 - 索引重建
  最大阶段           = 3
  阶段开始时间       = 06/28/2005 13:46:50.344277
  状态               = 已完成
  当前计数器         = 0
  最大计数器         = 0
  完成               = 0
  结束时间           = 06/28/2005 13:46:50.803619


表模式               = NEWTON
表名                 = SALES
表类型               = 用户
数据分区标识         = 2
数据对象页           = 1
读取的行数         = 4
写入的行数           = 1
溢出数               = 0
重组的页数           = 0
表重组信息:
  节点数             = 2
  重组类型           =
       回收
       表重组
       不允许访问
       通过表扫描重新集群
       仅重组数据
  重组索引数         = 0
  重组表空间数       = 3
长临时空间标识       = 3
  开始时间           = 06/28/2005 13:46:50.179922
  重组阶段           = 3 - 索引重建
  最大阶段           = 3
  阶段开始时间       = 06/28/2005 13:46:50.344277
  状态               = 已完成
  当前计数器         = 0
  最大计数器         = 0
  完成               = 0
  结束时间           = 06/28/2005 13:46:50.803619
```

示例 **3**:

```
SELECT * FROM SYSIBMADM.SNAPLOCK WHERE tabname = 'SALES';
```

已将输出修改为仅包括相关表的表信息的子集。

```
...   TBSP_NAME TABNAME LOCK_OBJECT_TYPE   LOCK_MODE  LOCK_STATUS ...
     --------- ------- ------------------ ---------- -----------
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...
...   PARTTBS   SALES      ROW_LOCK        X          GRNT        ...
...   -         SALES      TABLE_LOCK      IX         GRNT        ...
...   PARTTBS   SALES      TABLE_PART_LOCK IX         GRNT        ...

 9 record(s) selected.
```

此查询（已继续）的输出。

```
... LOCK_ESCALATION LOCK_ATTRIBUTES DATA_PARTITION_ID DBPARTITIONNUM
    --------------- --------------- ----------------- --------------
...               0 INSERT                          2              2
...               0 NONE                            -              2
...               0 NONE                            2              2
...               0 INSERT                          0              0
...               0 NONE                            -              0
...               0 NONE                            0              0
...               0 INSERT                          1              1
...               0 NONE                            -              1
...               0 NONE                            1              1
```

示例 **4**:

```
SELECT * FROM SYSIBMADM.SNAPTAB WHERE tabname = 'SALES';
```

已将输出修改为仅包括相关表的表信息的子集。

```
... TABSCHEMA TABNAME TAB_FILE_ID TAB_TYPE   DATA_OBJECT_PAGES
ROWS_WRITTEN ...
... --------- ------- ----------- ---------- ----------------- ------------ ...
... NEWTON    SALES             2 USER_TABLE                 1            1
...
... NEWTON    SALES             4 USER_TABLE                 1            1
...
... NEWTON    SALES             3 USER_TABLE                 1            1
...

 3 record(s) selected.
```

此查询（已继续）的输出。

```
... OVERFLOW_ACCESSES PAGE_REORGS DBPARTITIONNUM TBSP_ID
DATA_PARTITION_ID
... ----------------- ----------- -------------- ------- -----------------
...                 0           0              0       3                 0
...                 0           0              2       3                 2
...                 0           0              1       3                 1
```

示例 **5**:

```
SELECT * FROM SYSIBMADM.SNAPTAB_REORG WHERE tabname = 'SALES';;
```

已将输出修改为仅包括相关表的表信息的子集。

```
    REORG_PHASE   REORG_MAX_PHASE REORG_TYPE
...
     ------------- --------------- ------------------------------------------
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+
DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+
DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+
DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+
DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+
DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+
DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+
DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+
DATAONLY  ...
 INDEX_RECREATE             3  RECLAIM+OFFLINE+ALLOW_NONE+TABLESCAN+
DATAONLY  ...

 9 record(s) selected.
```

此查询（已继续）的输出。

```
... REORG_STATUS REORG_TBSPC_ID DBPARTITIONNUM DATA_PARTITION_ID
    ------------------------- -------------- -----------------
... COMPLETED               3             2                 0
... COMPLETED               3             2                 1
... COMPLETED               3             2                 2
... COMPLETED               3             1                 0
... COMPLETED               3             1                 1
... COMPLETED               3             1                 2
... COMPLETED               3             0                 0
... COMPLETED               3             0                 1
... COMPLETED               3             0                 2
```

示例 6:

表重组信息包括有关重组操作执行期间回收扩展数据块的信息。以下示例显示了相关的输出。

```
db2 -v "get snapshot for tables on wsdb"

 表重组信息:
   重组类型          =
        回收扩展数据块
        允许写访问
   重组索引数        = 0
   重组表空间数      = 0
   开始时间          = 10/22/2008 15:49:35.477532
   重组阶段          = 12 - 释放
   最大阶段          = 3
```

注: 来自 SQLM_DBMON_VERSION9_7 以前的监视器版本的任何快照请求都不会将任何回收重组状态返回给发出请求的客户机。

分区数据库系统上的全局快照

在分区数据库系统上，可以使用快照监视器来获取当前分区、指定分区或所有分区的快照。对分区数据库的所有分区获取全局快照时，会先聚集数据，然后返回结果。

对不同元素类型聚集数据的方式如下所示:

- **计数器、时间和标尺**

  包含从实例中的每个分区收集的所有可能值的总和。例如，GET SNAPSHOT FOR DATABASE XYZ ON TEST GLOBAL 对分区数据库实例中的所有分区返回从数据库读取的行数（rows_read）。

- **水位标记**

  返回分区数据库系统中的任何分区的最高（高水位）或最低（低水位）值。如果返回的值值得关注，那么可以获取各个分区的快照以确定特定分区是否使用过度或者问题是否为实例范围内的问题。

- **时间戳记**

  设为连接快照监视器实例代理程序的分区的时间戳记值。注意，所有时间戳记值都在 timestamp 监视开关控制之下。

- **信息**

  返回可能妨碍工作的分区的最重要信息。例如，对于元素 appl_status，如果一个分区上的状态为“正在执行 UOW”，而另一个分区上的状态为“等待锁定”，那么返回“等待锁定”，原因是这是挂起应用程序的执行的状态。

您也可以重置计数器，设置监视开关，以及检索分区数据库中的个别分区或所有分区的监视开关设置。

注: 获取全局快照时，如果一个或多个分区遇到错误，那么将从成功获取快照的分区收集数据，同时返回一个警告（sqlcode 1629）。如果以全局方式获取或更新监视开关，或者计数器在一个或多个分区上重置失败，那么不会设置这些分区的监视开关或进行数据重置。

为分区数据库或为 DB2 pureScale 环境中的数据库创建事件监视器

一般来说，分区数据库系统上或 DB2 pureScale 环境中的事件监视器的工作方式与在单成员数据库上运行的事件监视器的工作方式相似。但是，当为这些环境创建事件监视器时，有些差别需要牢记。

过程

- 指定要监视的分区。

```
CREATE EVENT MONITOR tabmon FOR TABLES
                    WRITE TO FILE '/tmp/tabevents'
                    ON PARTITION 3
```

  tabmon 代表事件监视器的名称。

  /tmp/tab events 是目录路径（在 UNIX 上）的名称，事件监视器会将事件文件写至该目录。

3 表示要监视的分区号。

- 指定是在局部作用域还是全局作用域收集事件监视器数据。 例如，要从所有分区收集事件监视器报告，请发出以下语句:

```
CREATE EVENT MONITOR dlmon FOR DEADLOCKS
                    WRITE TO FILE '/tmp/dlevents'
                    ON PARTITION 3 GLOBAL
```

注: 只有死锁和与带有详细信息事件监视器的死锁才能定义为 GLOBAL。所有分区会将与死锁有关的事件记录报告至分区 3。

- 要仅从局部分区收集事件监视器报告，那么发出以下语句:

```
CREATE EVENT MONITOR dlmon FOR TABLES
                    WRITE TO FILE '/tmp/dlevents'
                    ON PARTITION 3 LOCAL
```

这是分区数据库中的文件和管道事件监视器的缺省行为。对于写至表事件监视器，将忽略 LOCAL 和 GLOBAL 子句。

- 可查看现有事件监视器的监视器分区和作用域值。为此，请使用以下语句查询 SYSCAT.EVENTMONITORS 表:

```
SELECT EVMONNAME, NODENUM, MONSCOPE FROM SYSCAT.EVENTMONITORS
```

结果

创建并激活事件监视器后，该事件监视器就会在指定的事件发生时记录监视数据。

第 18 章 开发好的备份和恢复策略

崩溃恢复

对数据库执行的事务（也称工作单元）可能被意外中断。如果在作为工作单元一部分的所有更改完成、落实并写入磁盘之前发生故障，那么该数据库就会处于不一致和不可用的状态。

*崩溃恢复*是将数据库移动回一致并可用状态的进程。为此，回滚未完成的事务，并完成当发生崩溃时仍在内存中的已落实事务 (图 43)。当数据库处于一致和可用状态时，它已达到一种称为一致点的状态。

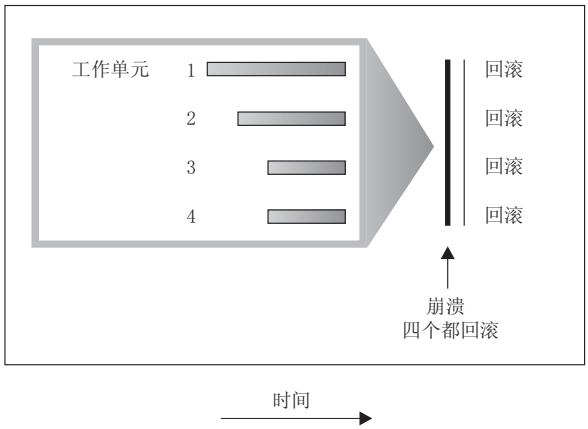


图 43. 回滚工作单元（崩溃恢复）

如果是使用 IBM DB2 pureScale Feature，那么需要了解两种特定类型的崩溃恢复：*成员崩溃恢复*和*组崩溃恢复*。成员崩溃恢复是在成员失败之后，使用单个成员的日志流来恢复部分数据库的过程。成员崩溃恢复通常在成员重新启动时自动启动，它是一种联机操作，这表示其他成员仍可以访问数据库。多个成员可以同时执行成员崩溃恢复。组崩溃恢复是在因失败而导致集群中未保留任何可用的集群高速缓存设施之后，使用多个成员的日志流来恢复数据库的过程。组崩溃恢复通常也会（在组重新启动时）自动启动，而正在进行组崩溃恢复时，无法访问数据库，因为 DB2 崩溃恢复在 DB2 pureScale 环境外部进行操作。

如果数据库或数据库管理器失败，那么数据库可能处于不一致状态。数据库内容可能包括发生失败时未完成的事务所进行的更改。数据库可能会丢失由失败之前已完成但尚未清空至磁盘的事务所进行的更改。必须执行崩溃恢复操作才能回滚部分完成的事务，并将已完成的事务先前只在内存中进行的更改写入磁盘。

必须执行崩溃恢复的情况包括:

- 机器上的断电故障，它会导致使用该机器的数据库管理器和数据库分区崩溃
- 硬件故障，例如内存、磁盘、CPU 或网络故障。
- 导致 DB2 实例异常结束的严重操作系统错误

如果您希望由数据库管理器自动执行崩溃恢复，请将 **autorestart** 数据库配置参数设置为 ON，以启用该自动重新启动参数。（这是缺省值。）如果不想要重新启动行为，那么将 **autorestart** 数据库配置参数设置为 OFF。因此，必须在数据库故障发生时发出 **RESTART DATABASE** 命令。如果数据库 I/O 在发生崩溃之前已处于暂挂状态，那么必须指定 **RESTART DATABASE** 命令的 **WRITE RESUME** 选项才能使崩溃恢复继续进行。管理通知日志记录数据库重新启动操作开始的时间。

如果崩溃恢复发生在用于前滚恢复的数据库上（即，未将 **logarchmeth1** 配置参数设置为 OFF），且在崩溃恢复期间因个别表空间而发生错误，那么会让该表空间脱机，直到修复后才能对其进行访问。崩溃恢复将继续在其他表空间上执行。在崩溃恢复完成时，该数据库中的其他表空间将是可访问的，并且可与该数据库建立连接。但是，如果脱机的表空间包含系统目录，那么必须先修复表空间才允许进行所有连接。此行为不适用于 DB2 pureScale 环境。如果成员崩溃恢复或组崩溃恢复期间出错，那么崩溃恢复操作将失败。

从分区数据库环境中的事务故障进行恢复

如果事务处理失败发生在分区数据库环境中，通常需要对发生了故障的数据库分区服务器和参与了该事务的任何其他数据库分区服务器都进行数据库恢复。

存在两种类型的数据库恢复:

- 对发生了故障的数据库分区服务器的崩溃恢复发生在更正了故障情况后。
- 对其他（仍活动的）数据库分区服务器的*数据库分区故障恢复*紧接在检测到故障后发生。

在分区数据库环境中，提交事务的数据库分区服务器是协调程序分区，而处理该事务的第一个代理程序是协调代理程序。协调代理程序负责将工作分布至其他数据库分区服务器上，并跟踪那些参与了该事务的服务器。当应用程序对一个事务发出 COMMIT 语句时，该协调代理程序使用两阶段落实协议来落实该事务。在第一阶段期间，协调程序分区将 PREPARE 请求分布至所有其他参与该事务的数据库分区服务器。然后，这些服务器用以下其中一项应答:

READ-ONLY
在此服务器中未发生任何数据更改

YES 在此服务器中发生了数据更改

NO 由于错误，服务器未准备落实

如果其中一个服务器应答 NO，那么回滚该事务。否则，协调程序分区开始第二阶段。

在第二阶段，协调程序分区写入一条 COMMIT 日志记录，然后将 COMMIT 请求分布至所有应答了 YES 的服务器。在所有其他数据库分区服务器都已落实后，它们会将 COMMIT 的应答发送至协调程序分区。当协调代理程序从所有参与服务器接收到所有 COMMIT 应答时，该事务完成。在此时间点，协调代理程序会写入一条 FORGET 日志记录。

活动数据库分区服务器上的事务故障恢复

如果任何数据库分区服务器检测到另一个服务器当机，那么与该发生故障的数据库分区服务器相关的所有工作都会停止:

- 如果仍处于活动状态的数据库分区服务器是某个应用程序的协调程序分区，且该应用程序在发生故障的数据库分区服务器上运行（尚未准备 COMMIT），那么会中断该协调代理程序，以便执行故障恢复。如果该协调代理程序处于 COMMIT 处理的第二个阶段，会将 SQL0279N 返回给应用程序，应用程序随之会丢失它的数据库连接。否则，协调代理程序将一个 ROLLBACK 请求分布至所有其他参与该事务的服务器，并将 SQL1229N 返回至该应用程序。
- 如果发生故障的数据库分区服务器是该应用程序的协调程序分区，那么仍在活动服务器上为该应用程序工作的代理程序会被中断，以便执行故障恢复。在事务未处于就绪状态的每个数据库分区上本地回滚事务。在事务处于就绪状态的那些数据库分区上，事务变得不确定。由于协调程序数据库分区不可用，所以协调程序数据库分区不知道事务在某些数据库分区上处于不确定状态。
- 如果该应用程序与发生故障的数据库分区服务器连接（在它发生故障之前），但是本地数据库分区服务器和发生故障的数据库分区服务器都不是协调程序分区，那么会中断为此应用程序工作的代理程序。协调程序分区将向其他数据库分区服务器发送 ROLLBACK 或 DISCONNECT 消息。如果协调程序分区返回 SQL0279，那么事务将仅在仍然活动的数据库分区服务器上处于不确定状态。

试图向该发生故障的服务器发送请求的任何进程（如，代理程序或死锁检测器）都会得到通知：它不能发送该请求。

发生故障的数据库分区服务器上的事务故障恢复

如果事务失败导致数据库管理器异常结束，那么可以发出具有 RESTART 选项的 **db2start** 命令，以便在重新启动数据库分区后立即重新启动数据库管理器。如果无法重新启动数据库分区，那么可以发出 **db2start**，以便在另一数据库分区上重新启动数据库管理器。

如果数据库管理器异常结束，那么服务器上的数据库分区可能会处于不一致状态。要使它们可用，可以在数据库分区服务器上触发崩溃恢复：

- 通过 **RESTART DATABASE** 命令显式地触发
- 在 *autorestart* 数据库配置参数已设置为 ON 后，通过 CONNECT 请求隐式触发

崩溃恢复将重新应用活动日志文件中的日志记录，以确保所有已完成的事务的结果都在数据库中。重新应用了这些更改后，除不确定事务外的所有未落实的事务都将本地回滚。分区数据库环境中有两种类型的不确定事务：

- 在不是协调程序分区的数据库分区服务器上，已就绪但未落实的事务就是不确定的。
- 在协调程序分区上，已落实但还未被记录为完成（即，还未写入 FORGET 记录）的事务是不确定的。当协调代理程序未从为该应用程序工作的所有服务器接收到全部 COMMIT 应答时，会发生这种情况。

崩溃恢复试图通过以下其中一项操作解决所有不确定事务。要执行的操作取决于数据库分区服务器是否为应用程序的协调程序分区：

- 如果重新启动的服务器不是该应用程序的协调程序分区，它会将一个查询消息发送至该协调代理程序，以发现该事务的结果。
- 如果重新启动的服务器是该应用程序的协调程序分区，它会将一个消息发送至协调代理程序仍在等待它们的 COMMIT 应答的所有其他代理程序（下级代理）。

崩溃恢复可能并不能解决所有不确定事务。例如，某些数据库分区服务器可能会不可用。如果协调程序分区在参与事务的其他数据库分区之前完成崩溃恢复，那么崩溃恢复将不能解决不确定事务。因为崩溃恢复由每个数据库分区独立执行，所以上述情况是意料之中的事情。在这种情况中，会返回 SQL 警告消息 SQL1061W。由于不确定事务占用了资源（例如锁定和活动日志空间），有可能会导致不能对数据库进行任何更改，因为不确定事务占用了活动日志空间。因此，应确定在崩溃恢复之后是否还有不确定事务，并尽快恢复解决这些不确定事务所需的所有数据库分区服务器。

注: 在分区数据库环境中，会对每个节点运行 RESTART 数据库命令。为确保对所有节点重新启动该数据库，请使用以下建议命令:

```
db2_all "db2 restart database <database_name>"
```

如果解决不确定事务所需的一个或多个服务器不能及时恢复，且需要访问其他服务器上的数据库分区，可以通过作出启发式决策来手动解决这些不确定事务。可以使用 **LIST INDOUBT TRANSACTIONS** 命令来查询、落实和回滚服务器上的不确定事务。

注: **LIST INDOUBT TRANSACTIONS** 命令还用于分布式事务环境中。为了区分这两种类型的不确定事务，**LIST INDOUBT TRANSACTIONS** 命令返回的输出中的 *originator* 字段显示以下其中一项:

- DB2 企业服务器版，指示该事务始发于分区数据库环境。
- XA，它指示该事务始发于分布式环境中。

标识发生故障的数据库分区服务器

当一个数据库分区服务器发生故障时，应用程序通常会接收到下列其中一个 SQLCODE。检测哪个数据库管理器发生故障的方法取决于接收到的 SQLCODE:

SQL0279N
当在 COMMIT 处理期间终止的事务中涉及了数据库分区服务器时，会接收到此 SQLCODE。

SQL1224N
当发生故障的数据库分区服务器是该事务的协调程序分区时，会接收到此 SQLCODE。

SQL1229N
当发生故障的数据库分区服务器不是该事务的协调程序分区时，会接收到此 SQLCODE。

确定哪个发生了故障的数据库分区服务器是一个两阶段进程。

1. 通过检查 SQLCA 来查找已检测到故障的分区服务器。与 SQLCODE SQL1229N 相关的 SQLCA 在 *sqlerrd* 字段的第六个数组位置包含检测到错误的服务器的节点号。（为服务器写入的节点号与 `db2nodes.cfg` 文件中的节点号对应。）
2. 针对发生了故障的服务器的节点号，检查在步骤一中找到的有关服务器的管理通知日志。

注: 如果正在一个处理器上使用多逻辑节点，那么一个逻辑节点发生故障会导致同一个处理器上的其他逻辑节点发生故障。

从数据库分区服务器的故障恢复

您可以通过找出故障并解决导致故障的问题，从而从失败的数据库分区服务器进行恢复。

过程

要从数据库分区服务器的故障中恢复，请执行下列步骤。

1. 校正导致该故障的问题。
2. 通过从任何数据库分区服务器发出 **db2start** 命令，重新启动数据库管理器。
3. 通过在发生故障的一个或多个数据库分区服务器上发出 **RESTART DATABASE** 命令，重新启动数据库。

重建分区数据库

要重建分区数据库，分别重建每个数据库分区。对于每个数据库分区，从目录分区开始，首先复原需要的所有表空间。未复原的所有表空间都处于复原暂挂状态。

复原了所有数据库分区之后，在目录分区上发出 **ROLLFORWARD DATABASE** 命令以前滚所有数据库分区。

关于此任务

注: 如果在将来的某一天您需要复原最初未包括在重建阶段中的任何表空间，那么需要确保在后来前滚表空间时 ROLLFORWARD 实用程序使数据库分区上的所有数据保持同步。如果在原始复原和前滚操作中丢失了某个表空间，那么直到尝试访问数据时才会检测到这种情况，将出现数据访问错误。需要复原和前滚丢失的表空间，以使它恢复与其余分区的同步。

要使用表空间级备份映像来重建分区数据库，请考虑以下示例。

在此示例中，有一个称为 SAMPLE 的可恢复数据库，它具有三个数据库分区:

- 数据库分区 1 包含表空间 SYSCATSPACE、USERSP1 和 USERSP2，它是目录分区
- 数据库分区 2 包含表空间 USERSP1 和 USERSP3
- 数据库分区 3 包含表空间 USERSP1、USERSP2 和 USERSP3

进行了下列备份，其中 BK*xy* 表示分区 *y* 上的备份编号 *x*:

- BK11 是 SYSCATSPACE、USERSP1 和 USERSP2 的备份
- BK12 是 USERSP2 和 USERSP3 的备份
- BK13 是 USERSP1、USERSP2 和 USERSP3 的备份
- BK21 是 USERSP1 的备份
- BK22 是 USERSP1 的备份
- BK23 是 USERSP1 的备份
- BK31 是 USERSP2 的备份
- BK33 是 USERSP2 的备份
- BK42 是 USERSP3 的备份
- BK43 是 USERSP3 的备份

下列过程演示了使用 CLP 发出的 **RESTORE DATABASE** 和 **ROLLFORWARD DATABASE** 命令将整个数据库重建至日志末尾。

过程

1. 在数据库分区 1 上，发出带有 **REBUILD** 选项的 **RESTORE DATABASE** 命令:

   ```
   db2 restore db sample rebuild with all tablespaces in database
         taken at BK31 without prompting
   ```

2. 在数据库分区 2 上，发出带有 **REBUILD** 选项的 **RESTORE DATABASE** 命令:

   ```
   db2 restore db sample rebuild with tablespaces in database
         taken at BK42 without prompting
   ```

3. 在数据库分区 3 上，发出带有 **REBUILD** 选项的 **RESTORE DATABASE** 命令:

   ```
   db2 restore db sample rebuild with all tablespaces in database
         taken at BK43 without prompting
   ```

4. 在目录分区上，发出带有 **TO END OF LOGS** 选项的 **ROLLFORWARD DATABASE** 命令:

   ```
   db2 rollforward db sample to end of logs
   ```

5. 发出带有 **STOP** 选项的 **ROLLFORWARD DATABASE** 命令:

   ```
   db2 rollforward db sample stop
   ```

下一步做什么

此时，该数据库在所有数据库分区上都是可连接的，并且所有表空间都处于 NORMAL 状态。

使用 db2adutl 来恢复数据

可以使用带有 **logarchopt1** 和 **vendoropt** 数据库配置参数的 **db2adutl** 命令来执行跨节点恢复。来自一些不同 Tivoli Storage Manager (TSM) 环境中的示例演示了如何执行此恢复。

在下列示例中，计算机 1 名为 bar，它正在运行 AIX 操作系统。此机器上的用户是 roecken。bar 上的数据库名为 zample。计算机 2 名为 dps。此计算机也在运行 AIX 操作系统，且用户是 regress9。

示例 1: TSM 服务器自动管理密码（PASSWORDACCESS 选项设置为 GENERATE）

此跨节点恢复示例显示如何设置两台计算机，以便当日志归档和备份都存储在 TSM 服务器上且在该服务器上使用 PASSWORDACCESS=GENERATE 选项管理密码时，您可以将数据从一台计算机恢复至另一台计算机。

注: 在更新数据库配置之后，可能需要对数据库进行脱机备份。

1. 要启用数据库以将 bar 计算机的日志归档到 TSM 服务器，请使用下列命令更新 zample 数据库的数据库配置参数 **logarchmeth1**:

   ```
   bar:/home/roecken> db2 update db cfg for zample using LOGARCHMETH1 tsm
   ```

   将返回以下信息:

   ```
   成功完成 DB20000I UPDATE DATABASE CONFIGURATION 命令。
   ```

2. 使用下列命令从数据库断开所有用户和应用程序的连接:

   ```
   db2 force applications all
   ```

3. 使用下列命令验证是否已没有应用程序连接到数据库:

```
db2 list applications
```

您应该会接收到一条消息，说明未返回任何数据。

注: 在分区数据库环境中，必须对所有数据库分区都执行此步骤。

4. 使用下列命令在 TSM 服务器上创建数据库备份:

```
db2 backup db zample use tsm
```

将返回类似以下的信息:

```
Backup successful. The timestamp for this backup imagge is : 20090216151025
```

注: 在分区数据库环境中，必须对所有数据库分区都执行此步骤。根据您正在执行联机备份还是脱机备份，在数据库分区上执行此步骤的顺序有所不同。有关更多信息，请参阅第 393 页的『备份数据』。

5. 使用下列命令连接至 zample 数据库:

```
db2 connect to zample
```

6. 通过使用下列命令创建一个表并将数据装入 TSM 服务器来生成数据库的新事务日志:

```
bar:/home/roecken> db2 load from mr of del modified by noheader replace
   into employee copy yes use tsm
```

在此示例中，表名为 employee，并且正在从名为 mr 的定界 ASCII 文件中装入数据。指定了 **COPY** YES 选项以生成所装入的数据的副本，并且 **USE TSM** 选项指定该数据的副本存储在 TSM 服务器上。

注: 仅当数据库启用了前滚恢复功能时才能指定 **COPY** YES 选项；即，必须将 **logarchmeth1** 数据库配置参数设置为 USEREXIT、LOGRETAIN、DISK 或 TSM。

为了指示它的进度，装入实用程序将返回一系列消息:

```
SQL3109N  实用程序正开始从"/home/roecken/mr"文件中装入数据。
SQL3500W  实用程序正在开始"LOAD"阶段，开始时间为"02/16/2009
15:12:13.392633"。

SQL3519W  开始装入一致点。输入记录计数 ="0"。

   SQL3520W  装入一致点成功。
SQL3110N  实用程序已完成处理。从输入文件
读取了"1"行。

SQL3519W  开始装入一致点。输入记录数 ="1"。

   SQL3520W  装入一致点成功。
SQL3515W 实用程序已完成"LOAD"阶段，完成时间为"02/16/2009
15:12:13.445718"。


      读取的行数        = 1
   跳过的行数        = 0
   装入的行数        = 1
   拒绝的行数        = 0
   删除的行数        = 0
   落实的行数        = 1
```

7. 将数据装入表中后，请通过在 zample 数据库中运行下列查询来确认在 TSM 服务器上存在一个备份映像、一个装入副本映像和一个日志文件:

```
bar:/home/roecken/sqllib/adsm> db2adutl query db zample
```

将返回以下信息:

```
正在检索 FULL DATABASE BACKUP 信息。
 1 时间: 20090216151025  最早的日志: S0000000.LOG  日志流: 0
 会话数: 1

正在检索 INCREMENTAL DATABASE BACKUP 信息。
 找不到 ZAMPLE 的 INCREMENTAL DATABASE BACKUP 映像

正在检索 DELTA DATABASE BACKUP 信息。
 找不到 ZAMPLE 的 DELTA DATABASE BACKUP 映像

正在检索 TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 TABLESPACE BACKUP 映像

正在检索 INCREMENTAL TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 INCREMENTAL TABLESPACE BACKUP 映像

正在检索 DELTA TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 DELTA TABLESPACE BACKUP 映像

正在检索 LOAD COPY 信息。
 1 时间:  20090216151213

正在检索 LOG ARCHIVE 信息。
日志文件: S0000000.LOG, 链号: 0, 日志流: 0,
生成时间: 2009-02-16-15.10.38
```

8. 要启用跨节点恢复，必须允许另一台计算机和帐户访问与 bar 计算机相关联的对象。在此示例中，使用下列命令允许计算机 dps 和用户 regress9 进行访问:

```
bar:/home/roecken/sqllib/adsm> db2adutl grant user regress9
        on nodename dps for db zample
```

将返回以下信息:

```
成功添加了 regress9 访问 dps 节点上的 ZAMPLE 的许可权。
```

注: 您可以通过发出下列命令检索当前节点的当前访问列表来确认 **db2adutl** 授权操作的结果:

```
bar:/home/roecken/sqllib/adsm> db2adutl queryaccess
```

将返回以下信息:

```
节点                 用户名             数据库名称    类型
-----------------------------------------------------------
  DPS                regress9           ZAMPLE          A
-----------------------------------------------------------
   访问类型:    B - 备份映像    L - 日志    A - 同时使用这两种访问类型
```

9. 在此示例中，尚未设置计算机 2 dps 以用于 zample 数据库的跨节点恢复。使用下列命令验证在 TSM 服务器上是否已没有数据与此用户和计算机相关联:

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
```

将返回以下信息:

```
  --- 数据库目录是空的 ---
警告: 在 ADSM 服务器上 DB2 没有创建文件空间
警告: 在 ADSM 中找不到任何别名的 DB2 备份映像。
```

10. 使用下列命令查询 TSM 服务器以获得与用户 roecken 和计算机 bar 相关联的 zample 数据库的对象列表:

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename
           bar owner roecken
```

将返回以下信息:

```
--- 数据库目录是空的 ---

对 ZAMPLE 数据库的查询

正在检索 FULL DATABASE BACKUP 信息。
 1 时间: 20090216151025  最早的日志: S0000000.LOG  日志流: 0
 会话数: 1

正在检索 INCREMENTAL DATABASE BACKUP 信息。
 找不到 ZAMPLE 的 INCREMENTAL DATABASE BACKUP 映像

正在检索 DELTA DATABASE BACKUP 信息。
 找不到 ZAMPLE 的 DELTA DATABASE BACKUP 映像

正在检索 TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 TABLESPACE BACKUP 映像

正在检索 INCREMENTAL TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 INCREMENTAL TABLESPACE BACKUP 映像

正在检索 DELTA TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 DELTA TABLESPACE BACKUP 映像

正在检索 LOAD COPY 信息。
 1 时间:  20090216151213

正在检索 LOG ARCHIVE 信息。
日志文件: S0000000.LOG, 链号: 0, 日志流: 0,
生成时间: 2009-02-16-15.10.38
```

此信息与先前生成的 TSM 信息匹配，并确认可以将此映像复原到 dps 计算机上。

11. 使用下列命令将 zample 数据库从 TSM 服务器复原至 dps 计算机:

```
dps:/home/regress9> db2 restore db zample use tsm options
"'-fromnode=bar -fromowner=roecken'" without prompting
```

将返回以下信息:

```
DB20000I  已成功完成 RESTORE DATABASE 命令。
```

注: 如果 dps 上已经存在 zample 数据库，那么将忽略 **OPTIONS** 参数，而使用数据库配置参数 **vendoropt**。此配置参数将覆盖备份或复原操作的 **OPTIONS** 参数。

12. 执行前滚操作以应用在创建新表并装入新数据时记录在 zample 数据库日志文件中的事务。在此示例中，因为未指定用户和计算机信息而导致 ROLLFORWARD 实用程序找不到日志文件，所以前滚操作的下列尝试将失败:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

该命令会返回下列错误:

```
SQL4970N  数据库"ZAMPLE"上的前滚恢复不能达到指定的停止点（日志结束
或时间点），原因是在节点"0"上丢失了日志文件。
```

使用适当的 **logarchopt** 值强制 ROLLFORWARD 实用程序查找与另一台计算机相关联的日志文件。在此示例中，使用下列命令设置 **logarchopt1** 数据库配置参数并搜索与用户 roecken 和计算机 bar 相关联的日志文件:

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
"'-fromnode=bar -fromowner=roecken'"
```

13. 通过使用下列命令来设置 **vendoropt** 数据库配置参数，使 ROLLFORWARD 实用程序能够使用备份和装入副本映像:

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
"'-fromnode=bar -fromowner=roecken'"
```

14. 使用下列命令通过应用记录在 zample 数据库日志文件中的事务来完成跨节点数据恢复:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

将返回以下信息:

```
                                  前滚状态

 输入数据库别名                   = zample
 返回了状态的成员数               = 1

 成员数          前滚                          要读取的        已处理的日志文件              上次落实的事务
                  状态                            下一个日志
 -------------  -----------  -----------  -------------------------  ------------------------------
       0        未暂挂                 S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC

DB20000I  已成功完成 ROLLFORWARD 命令。
```

已将计算机 dps 上用户 regress9 下的数据库 zample 恢复到计算机 bar 上用户 roecken 下数据库的同一位置。

示例 2：密码由用户管理（PASSWORDACCESS 选项设置为 PROMPT）

此跨节点恢复示例显示如何设置两台计算机，以便当日志归档和备份都存储在 TSM 服务器上且在该服务器上由用户管理密码时，您可以将数据从一台计算机恢复至另一台计算机。在这些环境中，需要额外的信息，特别是用于创建对象的计算机的 TSM 节点名和密码。

1. 通过添加下列行来更新客户机 dsm.sys 文件，因为计算机 bar 是源计算机的名称

   ```
   NODENAME bar
   ```

   注: 在 Windows 操作系统上，此文件名为 dsm.opt 文件。更新此文件时，必需重新引导系统才能使更改生效。

2. 使用下列命令查询 TSM 服务器以获得与用户 roecken 和计算机 bar 相关联的对象列表:

   ```
   dps:/home/regress9/sqllib/adsm> db2adutl query db zample nodename bar
      owner roecken password *******
   ```

   将返回以下信息:

   ```
   对 ZAMPLE 数据库的查询

      正在检索 FULL DATABASE BACKUP 信息。
       1 时间: 20090216151025  最早的日志: S0000000.LOG  日志流: 0
       会话数: 1

      正在检索 INCREMENTAL DATABASE BACKUP 信息。
       找不到 ZAMPLE 的 INCREMENTAL DATABASE BACKUP 映像

      正在检索 DELTA DATABASE BACKUP 信息。
       找不到 ZAMPLE 的 DELTA DATABASE BACKUP 映像

      正在检索 TABLESPACE BACKUP 信息。
       找不到 ZAMPLE 的 TABLESPACE BACKUP 映像

      正在检索 INCREMENTAL TABLESPACE BACKUP 信息。
       找不到 ZAMPLE 的 INCREMENTAL TABLESPACE BACKUP 映像

      正在检索 DELTA TABLESPACE BACKUP 信息。
       找不到 ZAMPLE 的 DELTA TABLESPACE BACKUP 映像

     正在检索 LOAD COPY 信息。
       1 时间:  20090216151213
   ```

```
正在检索 LOG ARCHIVE 信息。
日志文件: S0000000.LOG, 链号: 0, 日志流: 0,
生成时间: 2009-02-16-15.10.38
```

3. 如果在计算机 dps 上不存在 zample 数据库，那么请执行下列步骤:
   a. 使用下列命令创建空数据库 zample:
      ```
      dps:/home/regress9> db2 create db zample
      ```
   b. 使用下列命令来更新数据库配置参数 **tsm_nodename**:
      ```
      dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
      ```
   c. 使用下列命令来更新数据库配置参数 **tsm_password**:
      ```
      dps:/home/regress9> db2 update db cfg for zample using
      tsm_password ********
      ```
4. 尝试使用下列命令复原 zample 数据库:
   ```
   dps:/home/regress9> db2 restore db zample use tsm options
   "'-fromnode=bar -fromowner=roecken'" without prompting
   ```

   成功完成复原操作，但是发出了一条警告:
   ```
      SQL2540W  复原成功，但是在以"无中断"方式进行处理时，在"数据库复原"期间
   遇到了警告"2523"。
   ```
5. 使用下列命令来执行前滚操作:
   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   在此示例中，由于复原操作替换了数据库配置文件，因此 ROLLFORWARD 实用程序找不到正确的日志文件并且将返回下列错误消息:
   ```
     SQL1268N  由于检索节点"0"上的数据库"ZAMPLE"的日志文件"S0000000.LOG"时
   发生错误"-2112880618"，前滚恢复已停止。
   ```

   请将下列 TSM 数据库配置值重置为正确的值:
   a. 使用下列命令来设置 **tsm_nodename** 配置参数:
      ```
      dps:/home/regress9> db2 update db cfg for zample using tsm_nodename bar
      ```
   b. 使用下列命令来设置 **tsm_password** 数据库配置参数:
      ```
      dps:/home/regress9> db2 update db cfg for zample using tsm_password *******
      ```
   c. 使用下列命令来设置 **logarchopt1** 数据库配置参数，以便 ROLLFORWARD 实用程序可以找到正确的日志文件:
      ```
      dps:/home/regress9> db2 update db cfg for zample using logarchopt1
      "'-fromnode=bar -fromowner=roecken'"
      ```
   d. 使用下列命令来设置 **vendoropt** 数据库配置参数，以便在前滚操作期间也可以使用装入恢复文件:
      ```
      dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
      "'-fromnode=bar -fromowner=roecken'"
      ```
6. 通过使用下列命令来执行前滚操作，可以完成跨节点恢复:
   ```
   dps:/home/regress9> db2 rollforward db zample to end of logs and stop
   ```

   将返回以下信息:
   ```
                                   前滚状态

    输入数据库别名                  = zample
    返回了状态的成员数              = 1


    成员数       前滚                        要读取的          已处理的日志文件        上次落实的事务
                  状态                        下一个日志
    ------------- ----------- ----------- ------------------------- ------------------------------
   ```

```
                0      未暂挂               S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC

DB20000I  已成功完成 ROLLFORWARD 命令。
```

已将计算机 dps 上用户 regress9 下的数据库 zample 恢复到计算机 bar 上用户 roecken 下数据库的同一位置

示例 3: TSM 服务器已配置为使用客户机代理节点

此跨节点恢复示例显示如何将两台计算机设置为代理节点，以便当日志归档和备份都存储在 TSM 服务器上且在该服务器上使用 PASSWORDACCESS=GENERATE 选项管理密码时，您可以将数据从一台计算机恢复至另一台计算机。

注: 在更新数据库配置之后，可能需要对数据库进行脱机备份。

在此示例中，计算机 bar 和 dps 注册在 clusternode 代理名称下。这些计算机已经设置为代理节点。

1. 使用下列命令将计算机 bar 和 dps 在 TSM 服务器上注册为代理节点:

   ```
   REGISTER NODE clusternode mypassword
   GRANT PROXYNODE TARGET=clusternode AGENT=bar,dps
   ```

2. 要启用数据库以将日志归档到 TSM 服务器，请使用下列命令更新 zample 数据库的数据库配置参数 **logarchmeth1**:

   ```
   bar:/home/roecken> db2 update db cfg for zample using
      LOGARCHMETH1 tsm logarchopt1 "'-asnodename=clusternode'"
   ```

   将返回以下信息:

   ```
   成功完成 DB20000I UPDATE DATABASE CONFIGURATION 命令。
   ```

3. 使用下列命令从数据库断开所有用户和应用程序的连接:

   ```
   db2 force applications all
   ```

4. 使用下列命令验证是否已没有应用程序连接到数据库:

   ```
      db2 list applications
   ```

   您应该会接收到一条消息，说明未返回任何数据。

   注: 在分区数据库环境中，必须对所有数据库分区都执行此步骤。

5. 使用下列命令在 TSM 服务器上创建数据库备份:

   ```
   db2 backup db zample use tsm options "'-asnodename=clusternode'"
   ```

   将返回类似以下的信息:

   ```
   Backup successful. The timestamp for this backup image is : 20090216151025
   ```

   如果不在 **BACKUP DATABASE** 命令中指定 **-asnodename** 选项，那么可以改为更新 **vendoropt** 数据库配置参数。

   注: 在分区数据库环境中，必须对所有数据库分区都执行此步骤。根据您正在执行联机备份还是脱机备份，在数据库分区上执行此步骤的顺序有所不同。有关更多信息，请参阅第 393 页的『备份数据』。

6. 使用下列命令连接至 zample 数据库:

   ```
   db2 connect to zample
   ```

7. 通过使用下列命令创建一个表并将数据装入 TSM 服务器来生成数据库的新事务日志:

```
bar:/home/roecken> db2 load from mr of del modified by noheader
into employee copy yes use tsmwhere
```

在此示例中，表名为 employee，并且正在从名为 mr 的定界 ASCII 文件中装入数据。指定了 **COPY** YES 选项以生成所装入的数据的副本，并且 **USE TSM** 选项指定该数据的副本存储在 TSM 服务器上。

注: 仅当数据库启用了前滚恢复功能时才能指定 **COPY** YES 选项；即，必须将 **logarchmeth1** 数据库配置参数设置为 USEREXIT、LOGRETAIN、DISK 或 TSM。

为了指示它的进度，装入实用程序将返回一系列消息:

```
SQL3109N  实用程序正开始从"/home/roecken/mr"文件中装入数据。
SQL3500W  实用程序正在开始"LOAD"阶段，开始时间为"02/16/2009
15:12:13.392633"。

SQL3519W  开始装入一致点。输入记录计数 ="0"。

   SQL3520W  装入一致点成功。
SQL3110N  实用程序已完成处理。从输入文件
读取了"1"行。

SQL3519W  开始装入一致点。输入记录数 ="1"。

   SQL3520W  装入一致点成功。
SQL3515W 实用程序已完成"LOAD"阶段，完成时间为"02/16/2009
15:12:13.445718"。


   读取的行数        = 1
跳过的行数        = 0
装入的行数        = 1
拒绝的行数        = 0
删除的行数        = 0
落实的行数        = 1
```

8. 将数据装入表中后，请通过在 zample 数据库中运行下列查询来确认在 TSM 服务器上存在一个备份映像、一个装入副本映像和一个日志文件:

```
bar:/home/roecken/sqllib/adsm> db2adutl query db zample
   options "-asnodename=clusternode"
```

将返回以下信息:

```
正在检索 FULL DATABASE BACKUP 信息。
 1 时间: 20090216151025  最早的日志: S0000000.LOG  日志流: 0
 会话数: 1

正在检索 INCREMENTAL DATABASE BACKUP 信息。
 找不到 ZAMPLE 的 INCREMENTAL DATABASE BACKUP 映像

正在检索 DELTA DATABASE BACKUP 信息。
 找不到 ZAMPLE 的 DELTA DATABASE BACKUP 映像

正在检索 TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 TABLESPACE BACKUP 映像

正在检索 INCREMENTAL TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 INCREMENTAL TABLESPACE BACKUP 映像

正在检索 DELTA TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 DELTA TABLESPACE BACKUP 映像

正在检索 LOAD COPY 信息。
 1 时间:  20090216151213
```

```
正在检索 LOG ARCHIVE 信息。
日志文件: S0000000.LOG, 链号: 0, 日志流: 0,
生成时间: 2009-02-16-15.10.38
```

9. 在此示例中，尚未设置计算机 2 dps 以用于 zample 数据库的跨节点恢复。使用下列命令验证是否已没有数据与此用户和计算机相关联:

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
```

将返回以下信息:

```
  --- 数据库目录是空的 ---
警告: 在 ADSM 服务器上 DB2 没有创建文件空间
警告: 在 ADSM 中找不到任何别名的 DB2 备份映像。
```

10. 使用下列命令查询 TSM 服务器以获得与代理节点 clusternode 相关联的 zample 数据库的对象列表:

```
dps:/home/regress9/sqllib/adsm> db2adutl query db zample
   options="-asnodename=clusternode"
```

将返回以下信息:

```
--- 数据库目录是空的 ---

对 ZAMPLE 数据库的查询

正在检索 FULL DATABASE BACKUP 信息。
 1 时间: 20090216151025  最早的日志: S0000000.LOG  日志流: 0
 会话数: 1

正在检索 INCREMENTAL DATABASE BACKUP 信息。
 找不到 ZAMPLE 的 INCREMENTAL DATABASE BACKUP 映像

正在检索 DELTA DATABASE BACKUP 信息。
 找不到 ZAMPLE 的 DELTA DATABASE BACKUP 映像

正在检索 TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 TABLESPACE BACKUP 映像

正在检索 INCREMENTAL TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 INCREMENTAL TABLESPACE BACKUP 映像

正在检索 DELTA TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 DELTA TABLESPACE BACKUP 映像

正在检索 LOAD COPY 信息。
 1 时间:  20090216151213

正在检索 LOG ARCHIVE 信息。
日志文件: S0000000.LOG, 链号: 0, 日志流: 0,
生成时间: 2009-02-16-15.10.38
```

此信息与先前生成的 TSM 信息匹配，并确认可以将此映像复原到 dps 计算机上。

11. 使用下列命令将 zample 数据库从 TSM 服务器复原至 dps 计算机:

```
dps:/home/regress9> db2 restore db zample use tsm options
   "'-asnodename=clusternode'" without prompting
```

将返回以下信息:

```
DB20000I  已成功完成 RESTORE DATABASE 命令。
```

注: 如果 dps 上已经存在 zample 数据库，那么将忽略 **OPTIONS** 参数，而使用数据库配置参数 **vendoropt**。此配置参数将覆盖备份或复原操作的 **OPTIONS** 参数。

12. 执行前滚操作以应用在创建新表并装入新数据时记录在 zample 数据库日志文件中的事务。在此示例中，因为未指定用户和计算机信息而导致 ROLLFORWARD 实用程序找不到日志文件，所以前滚操作的下列尝试将失败:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

该命令会返回下列错误:

```
SQL4970N  数据库"ZAMPLE"上的前滚恢复不能达到指定的停止点（日志结束
或时间点），原因是在节点"0"上丢失了日志文件。
```

使用适当的 **logarchopt** 值强制 ROLLFORWARD 实用程序查找另一台计算机上的日志文件。在此示例中，使用下列命令设置 **logarchopt1** 数据库配置参数并搜索与用户 roecken 和计算机 bar 相关联的日志文件:

```
dps:/home/regress9> db2 update db cfg for zample using logarchopt1
   "'-asnodename=clusternode'"
```

13. 通过使用下列命令来设置 **vendoropt** 数据库配置参数，使 ROLLFORWARD 实用程序能够使用备份和装入副本映像:

```
dps:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-asnodename=clusternode'"
```

14. 使用下列命令通过应用记录在 zample 数据库日志文件中的事务来完成跨节点数据恢复:

```
dps:/home/regress9> db2 rollforward db zample to end of logs and stop
```

将返回以下信息:

```
                                 前滚状态

 输入数据库别名                   = zample
 返回了状态的成员数               = 1


 成员数        前滚                     要读取的           已处理的日志文件           上次落实的事务
                状态                         下一个日志
 -------------  -----------  -----------  -------------------------  ------------------------------
        0       未暂挂                S0000000.LOG-S0000000.LOG  2009-05-06-15.28.11.000000 UTC


DB20000I  已成功完成 ROLLFORWARD 命令。
```

已将计算机 dps 上用户 regress9 下的数据库 zample 恢复到计算机 bar 上用户 roecken 下数据库的同一位置。

示例 4: TSM 服务器已配置为使用 DB2 pureScale环境中的客户机代理节点

此示例显示如何将两个成员设置为代理节点，以便当日志归档和备份都存储在 TSM 服务器上且在该服务器上使用 PASSWORDACCESS=GENERATE 选项来管理密码时，您可以将数据从一个成员恢复至另一个成员。

注: 在更新数据库配置之后，可能需要对数据库进行脱机备份。

在此示例中，成员 member1 和 member2 注册在 clusternode 代理名称下。在 DB2 pureScale环境中，您可以从任何成员执行备份或数据恢复操作。在此示例中，会将数据从 member2 恢复

1. 使用下列命令将成员 member1 和 member2 在 TSM 服务器上注册为代理节点:

```
REGISTER NODE clusternode mypassword
GRANT PROXYNODE TARGET=clusternode AGENT=member1,member2
```

2. 要启用数据库以将日志归档到 TSM 服务器，请使用下列命令更新 zample 数据库的数据库配置参数 **logarchmeth1**:

```
member1:/home/roecken> db2 update db cfg for zample using
   LOGARCHMETH1 tsm logarchopt1 "'-asnodename=clusternode'"
```

注: 在 DB2 pureScale环境中，您可以从任何成员设置一次全局 **logarchmeth1** 数据库配置参数。

将返回以下信息:

```
成功完成 DB20000I UPDATE DATABASE CONFIGURATION 命令。
```

3. 使用下列命令从数据库断开所有用户和应用程序的连接:

```
db2 force applications all
```

4. 使用下列命令验证是否已没有应用程序连接到数据库:

```
db2 list applications global
```

您应该会接收到一条消息，说明未返回任何数据。

5. 使用下列命令在 TSM 服务器上创建数据库备份:

```
db2 backup db zample use tsm options '-asnodename=clusternode'
```

将返回类似以下的信息:

```
Backup successful. The timestamp for this backup image is : 20090216151025
```

如果不在 **BACKUP DATABASE** 命令中指定 **-asnodename** 选项，那么可以改为更新 **vendoropt** 数据库配置参数。

注: 在 DB2 pureScale环境中，您可以从任何成员运行此命令来备份数据库的所有数据。

6. 使用下列命令连接至 zample 数据库:

```
db2 connect to zample
```

7. 通过使用下列命令创建一个表并将数据装入 TSM 服务器来生成数据库的新事务日志:

```
member1:/home/roecken> db2 load from mr of del modified by noheader replace
      into employee copy yes use tsmwhere
```

在此示例中，表名为 employee，并且正在从名为 mr 的定界 ASCII 文件中装入数据。指定了 **COPY** YES 选项以生成所装入的数据的副本，并且 **USE TSM** 选项指定该数据的副本存储在 TSM 服务器上。

注: 仅当数据库启用了前滚恢复功能时才能指定 **COPY** YES 选项；即，必须将 **logarchmeth1** 数据库配置参数设置为 USEREXIT、LOGRETAIN、DISK 或 TSM。

为了指示它的进度，装入实用程序将返回一系列消息:

```
SQL3109N  实用程序正开始从"/home/roecken/mr"文件中装入数据。
SQL3500W  实用程序正在开始"LOAD"阶段，开始时间为"02/16/2009
15:12:13.392633"。

SQL3519W  开始装入一致点。输入记录计数 ="0"。

   SQL3520W  装入一致点成功。
SQL3110N  实用程序已完成处理。从输入文件
读取了"1"行。

SQL3519W  开始装入一致点。输入记录数 ="1"。

   SQL3520W  装入一致点成功。
SQL3515W 实用程序已完成"LOAD"阶段，完成时间为"02/16/2009
15:12:13.445718"。


      读取的行数         = 1
```

```
跳过的行数         = 0
装入的行数         = 1
拒绝的行数         = 0
删除的行数         = 0
落实的行数         = 1
```

8. 将数据装入表中后，请通过在 zample 数据库中运行下列查询来确认在 TSM 服务器上存在一个备份映像、一个装入副本映像和一个日志文件:

```
member1:/home/roecken/sqllib/adsm> db2adutl query db zample
   options "-asnodename=clusternode"
```

将返回以下信息:

```
正在检索 FULL DATABASE BACKUP 信息。
 1 时间: 20090216151025  最早的日志: S0000000.LOG  日志流: 0
 会话数: 1

正在检索 INCREMENTAL DATABASE BACKUP 信息。
 找不到 ZAMPLE 的 INCREMENTAL DATABASE BACKUP 映像

正在检索 DELTA DATABASE BACKUP 信息。
 找不到 ZAMPLE 的 DELTA DATABASE BACKUP 映像

正在检索 TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 TABLESPACE BACKUP 映像

正在检索 INCREMENTAL TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 INCREMENTAL TABLESPACE BACKUP 映像

正在检索 DELTA TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 DELTA TABLESPACE BACKUP 映像

正在检索 LOAD COPY 信息。
 1 时间:  20090216151213

正在检索 LOG ARCHIVE 信息。

日志文件: S0000000.LOG, 链号: 1, 日志流: 1, 生成时间: 2009-02-16-13.01.10

日志文件: S0000000.LOG, 链号: 1, 日志流: 0, 生成时间: 2009-02-16-13.01.11

日志文件: S0000000.LOG, 链号: 1, 日志流: 2, 生成时间: 2009-02-16-13.01.19

日志文件: S0000001.LOG, 链号: 1, 日志流: 0, 生成时间: 2009-02-16-13.02.49

日志文件: S0000001.LOG, 链号: 1, 日志流: 1, 生成时间: 2009-02-16-13.02.49

日志文件: S0000001.LOG, 链号: 1, 日志流: 2, 生成时间: 2009-02-16-13.02.49

日志文件: S0000002.LOG, 链号: 1, 日志流: 1, 生成时间: 2009-02-16-13.03.15

日志文件: S0000002.LOG, 链号: 1, 日志流: 2, 生成时间: 2009-02-16-13.03.15

日志文件: S0000002.LOG, 链号: 1, 日志流: 0, 生成时间: 2009-02-16-13.03.16
```

9. 使用下列命令查询 TSM 服务器以获得与代理节点 clusternode 相关联的 zample 数据库的对象列表:

```
member2:/home/regress9/sqllib/adsm> db2adutl query db zample
   options="-asnodename=clusternode"
```

将返回以下信息:

```
--- 数据库目录是空的 ---

对 ZAMPLE 数据库的查询

正在检索 FULL DATABASE BACKUP 信息。
 1 时间: 20090216151025  最早的日志: S0000000.LOG  日志流: 0
 会话数: 1

正在检索 INCREMENTAL DATABASE BACKUP 信息。
 找不到 ZAMPLE 的 INCREMENTAL DATABASE BACKUP 映像
```

```
正在检索 DELTA DATABASE BACKUP 信息。
 找不到 ZAMPLE 的 DELTA DATABASE BACKUP 映像

正在检索 TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 TABLESPACE BACKUP 映像

正在检索 INCREMENTAL TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 INCREMENTAL TABLESPACE BACKUP 映像

正在检索 DELTA TABLESPACE BACKUP 信息。
 找不到 ZAMPLE 的 DELTA TABLESPACE BACKUP 映像

正在检索 LOAD COPY 信息。
  1 时间:   20090216151213

正在检索 LOG ARCHIVE 信息。

日志文件: S0000000.LOG, 链号: 1, 日志流: 1, 生成时间: 2009-02-16-13.01.10

日志文件: S0000000.LOG, 链号: 1, 日志流: 0, 生成时间: 2009-02-16-13.01.11

日志文件: S0000000.LOG, 链号: 1, 日志流: 2, 生成时间: 2009-02-16-13.01.19

日志文件: S0000001.LOG, 链号: 1, 日志流: 0, 生成时间: 2009-02-16-13.02.49

日志文件: S0000001.LOG, 链号: 1, 日志流: 1, 生成时间: 2009-02-16-13.02.49

日志文件: S0000001.LOG, 链号: 1, 日志流: 2, 生成时间: 2009-02-16-13.02.49

日志文件: S0000002.LOG, 链号: 1, 日志流: 1, 生成时间: 2009-02-16-13.03.15

日志文件: S0000002.LOG, 链号: 1, 日志流: 2, 生成时间: 2009-02-16-13.03.15

日志文件: S0000002.LOG, 链号: 1, 日志流: 0, 生成时间: 2009-02-16-13.03.16
```

此信息与先前生成的 TSM 信息匹配，并确认可以将此映像复原到 member2 成员上。

10. 使用下列命令从 member2 成员复原 TSM 服务器上的 zample 数据库:

```
member2:/home/regress9> db2 restore db zample use tsm options
   '-asnodename=clusternode' without prompting
```

将返回以下信息:

```
DB20000I  已成功完成 RESTORE DATABASE 命令。
```

注: 如果 member2 上已经存在 zample 数据库，那么将忽略 **OPTIONS** 参数，而使用数据库配置参数 **vendoropt**。此配置参数将覆盖备份或复原操作的 **OPTIONS** 参数。

11. 通过使用下列命令来设置 **vendoropt** 数据库配置参数，使 ROLLFORWARD 实用程序能够使用备份和装入副本映像:

```
member2:/home/regress9> db2 update db cfg for zample using VENDOROPT
   "'-asnodename=clusternode'"
```

注: 在 DB2 pureScale环境中，您可以从任何成员设置一次全局 **vendoropt** 数据库配置参数。

12. 使用下列命令通过应用记录在 zample 数据库日志文件中的事务来完成跨成员数据恢复:

```
member2:/home/regress9> db2 rollforward db zample to end of logs and stop
```

将返回以下信息:

```
                                     前滚状态

 输入数据库别名                     = zample
 返回了状态的成员数                 = 3

 成员数         前滚                        要读取的             已处理的日志文件          上次落实的事务
                 状态                         下一个日志
 -------------  -----------  -----------  -------------------------  ------------------------------
             0  未暂挂                       S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC
             1  未暂挂                       S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC
             2  未暂挂                       S0000001.LOG-S0000012.LOG  2009-05-06-15.28.11.000000 UTC

DB20000I  已成功完成 ROLLFORWARD 命令。
```

已将成员 member2 上用户 regress9 下的数据库 zample 恢复到成员 member1 上用户 roecken 下数据库的同一位置。

使分区数据库环境中的时钟同步

应该使所有数据库分区服务器的系统时钟保持相对同步，以确保数据库操作顺利进行以及正向可恢复性不受限制。

数据库分区服务器之间的时差加上事务的任何潜在操作和通信延迟，应小于对 *max_time_diff*（节点之间的最大时差）数据库管理器配置参数指定的值。

为确保日志记录时间戳记反映分区数据库环境中的事务顺序，DB2 使用每台机器上的系统时钟以及文件 SQLOGCTL.LFH 中存储的虚拟时间戳记作为日志记录时间戳记的基准。但是，如果将系统时钟设置得提前，就会自动将日志时钟设置得提前。虽然可以将系统时钟往后调，但是日志时钟却不能这样设置，它会保持*相同的*超前时间，直至系统时钟与此时间匹配为止。于是，这两个时钟便同步了。这意味着一个数据库节点上的短期系统时钟错误可能会对数据库日志时间戳记产生长期的影响。

例如，假定数据库分区服务器 A 上的系统时钟被错误地设置为 2005 年 11 月 7 日，而当前年份是 2003 年，并假定在该数据库分区服务器上的数据库分区中落实了更新事务*之后*更正了此错误。如果继续使用该数据库，并且随着时间的推移定期对其进行更新，那么 2003 年 11 月 7 日至 2005 年 11 月 7 日之间的任何时间点实际上是无法通过前滚恢复到达的。当数据库分区服务器 A 上的 COMMIT 完成时，数据库日志中的时间戳记被设置为 2005，而日志时钟会停留在 2005 年 11 月 7 日，直到系统时钟与此时间相匹配为止。如果尝试前滚到这个时间范围内的某个时间点，那么操作将在指定停止点后的第一个时间戳记（即 2003 年 11 月 7 日）处停止。

虽然 DB2 无法控制对系统时钟的更新，但是 *max_time_diff* 数据库管理器配置参数减少了发生此类问题的机会:

- 此参数的可配置值的范围是 1 分钟至 24 小时。
- 当对非目录分区发出第一个连接请求时，数据库分区服务器会将它的时间发送至该数据库的目录分区。该目录分区就会检查请求连接的数据库分区上的时间与它自己的时间是否在 *max_time_diff* 参数指定的范围之内。如果超出此范围，那么拒绝该连接。
- 涉及到数据库中两个以上数据库分区服务器的更新事务，必须先验证参与数据库分区服务器上的时钟是否同步，然后才可落实该更新。如果两个或更多个数据库分区服务器的时差超出 *max_time_diff* 允许的限制，那么会回滚该事务，以防止将不正确的时间传播至其他数据库分区服务器。

第 19 章 故障诊断

诊断分区数据库环境

在分区数据库环境中发出命令

在分区数据库环境中，您可能想要发出将在实例中的计算机或在多个数据库分区服务器上运行的命令。为此，您可以使用 **rah** 命令或 **db2_all** 命令。**rah** 命令允许您发出将在实例中的计算机上运行的命令。

如果想要命令在实例中的多个数据库分区服务器上运行，那么运行 **db2_all** 命令。本章节概述了这些命令。以下信息仅适用于分区数据库环境。

在 Windows 上，要运行 **rah** 命令或 **db2_all** 命令，您必须使用 Administrators 组成员的用户帐户来登录。

在 Linux 和 UNIX 操作系统上，您的登录 shell 程序可以是 Korn shell 程序或任何其他的 shell；但是，不同 shell 处理包含特殊字符的命令所用的方式不同。

另外，在 Linux 和 UNIX 操作系统上，**rah** 使用 **DB2RSHCMD** 注册表变量指定的远程 shell 程序。可以在两个远程 shell 程序之间选择：ssh（用于更高的安全性要求）或 rsh（对于 HP-UX，则为 remsh）。如果没有设置 **DB2RSHCMD**，那么会使用 rsh（对于 HP-UX，则为 remsh）。ssh 远程 shell 程序用来防止在 UNIX 操作系统环境中以明文形式传输密码。

如果命令在一个数据库分区服务器上运行，而您想让该命令在它们所有上面运行，请使用 **db2_all**。**db2trc** 命令例外，该命令在计算机的所有逻辑数据库分区服务器上运行。如果要在所有计算机的所有逻辑数据库分区服务器上运行 **db2trc**，请使用 **rah**。

注： **db2_all** 命令不支持需要交互式用户输入的命令。

第 4 部分 性能问题

第 20 章 数据库设计中的性能问题

性能增强功能

表分区和多维集群表

在同时是多维集群表和数据分区表的表中，可以同时在表分区的范围分区规范和多维集群 (MDC) 键中使用列。与只单独使用多维集群或分区功能相比，同时是多维集群表和分区表的表可以获取较详细的数据分区和块消除。

在许多应用中将 MDC 键列指定为不同于对表进行分区的列很有用。应该注意的是，表分区是多列的，而 MDC 是多维的。

主流 DB2 数据仓库的特征

下列建议主要针对 DB2 V9.1 中新增的典型主流仓库。假定下列特征:

- 数据库在多台机器或多个 AIX 逻辑分区上运行。
- 使用分区数据库环境（使用 DISTRIBUTE BY HASH 子句来创建表）。
- 有 4 到 50 个数据分区。
- 考虑其 MDC 和表分区的表是主要事实表。
- 表的行数为 100,000,000 至 100,000,000,000。
- 在各种时间范围装入新数据: 每夜、每周和每月。
- 每日接受量为 1 万到 1 亿条记录。
- 数据量变化: 最多的一个月是最少的月的 5 倍。同样，最大维数（生产线，区域）具有 5 倍大小范围。
- 获取 1 到 5 年的详细数据。
- 每月或每个季度转出到期数据。
- 表使用大范围的查询类型。但是，相对于 OLTP 工作负载来说，该工作负载通常是具有下列特征的分析查询:
  - 较大的结果集，最多有 2 百万行
  - 大多数或全部查询都命中视图，而不是基本表
- SQL 子句按范围（BETWEEN 子句）、列表中的项等选择数据。

主流 DB2 V9.1 数据仓库事实表的特征

一个典型仓库事实表可能采用以下设计:

- 在 Month 列中创建数据分区。
- 为转出的每个时间段（例如，1 个月和 3 个月）定义数据分区。
- 在 Day 和 1 到 4 个其他维的基础上创建 MDC 维。典型的维有: 生产线和区域。
- 所有数据分区和 MDC 集群都分布在所有数据库分区中。

MDC 和表分区具有一些相同的好处。下表列示组织中的潜在需求并根据先前确定的特征确定建议的组织方案。

表 15. 将表分区与 MDC 表配合使用

问题	建议的方案	建议
转出期间的数据可用性	表分区	使用 DETACH PARTITION 子句来转出大量数据，并且只出现最少中断。
查询性能	表分区和 MDC	MDC 最适合用来查询多个维。表分区通过数据分区消除提高性能。
最少重组	MDC	MDC 维护集群，从而减少进行重组的必要性。
在传统脱机窗口中转出一个月或更长时间的数据	表分区	数据分区可以完全解决此需求。MDC 不起任何作用，并且仅 MDC 并不适合。
在短时间脱机窗口（小于 1 分钟）期间转出一个月或更长时间的数据	表分区	数据分区可以完全解决此需求。MDC 不起任何作用，并且仅 MDC 并不适合。
转出一个月或更长时间的数据，并同时在不损失任何服务的情况下使表对于提交查询的企业用户完全可用。	MDC	MDC 只能解决一部分此需求。由于表处于脱机状态的时间段太短，表分区并不适合。
每天装入数据（**LOAD** 或 **INGEST** 命令）	表分区和 MDC	此时 MDC 具有很多好处。表分区具有增量的好处。
“连续”装入数据（具有 ALLOW READ ACCESS 或 **INGEST** 命令的 **LOAD** 命令）	表分区和 MDC	此时 MDC 具有很多好处。表分区具有增量的好处。
“传统 BI”查询的查询执行性能	表分区和 MDC	MDC 特别适合用来查询立方体/多个维。表分区通过分区消除提高性能。
通过消除进行重组的必要性或降低执行任务所产生的不良影响，使重组所带来的不良影响降到最低。	MDC	MDC 维护集群，从而减少进行重组的必要性。如果使用 MDC，那么数据分区不提供增量好处。但是，如果不使用 MDC，那么表分区通过在分区级别维护一些粗粒度集群会有助于减少重组的必要性。

示例 1:

考虑一个具有键列 YearAndMonth 和 Province 的表。一种合理的规划此表方法是按日期进行分区，每 2 个月添加一个数据分区。此外，还可以按 Province 进行组织，以便任何两个月日期范围内的特定省份的所有行集群在一起，如第 28 页的图 6 中所示。

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (Province);
```

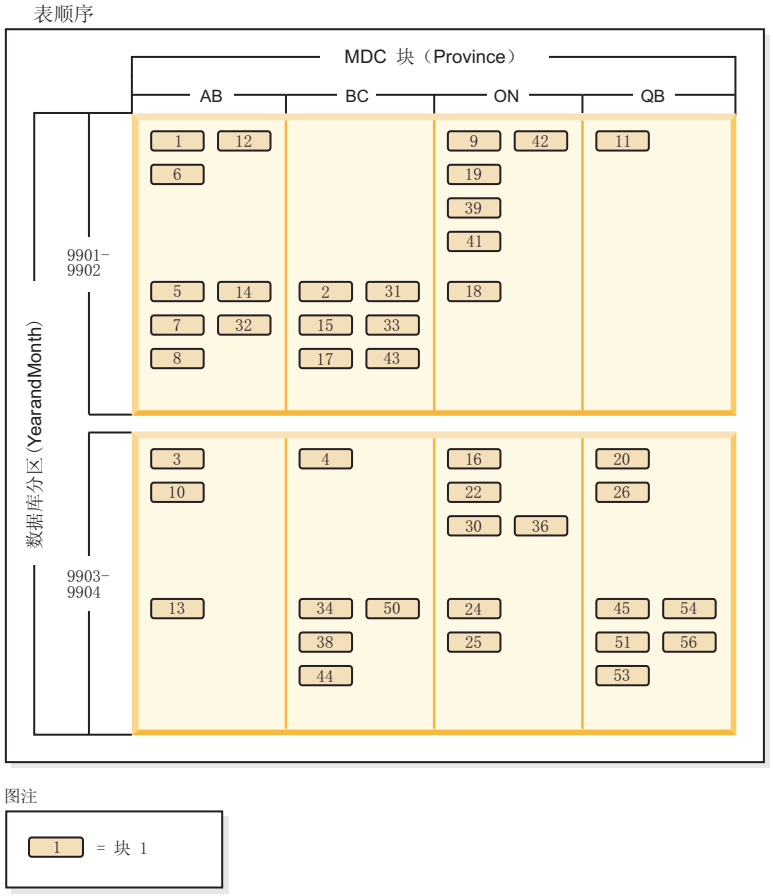


图 44. 按 YearAndMonth 分区并按 Province 组织的表

示例 2:

通过将 YearAndMonth 添加至 ORGANIZE BY DIMENSIONS 子句，可以获得较高的详细程度，如第 29 页的图 7 中所示。

```
CREATE TABLE orders (YearAndMonth INT, Province CHAR(2))
PARTITION BY RANGE (YearAndMonth)
(STARTING 9901 ENDING 9904 EVERY 2)
ORGANIZE BY (YearAndMonth, Province);
```

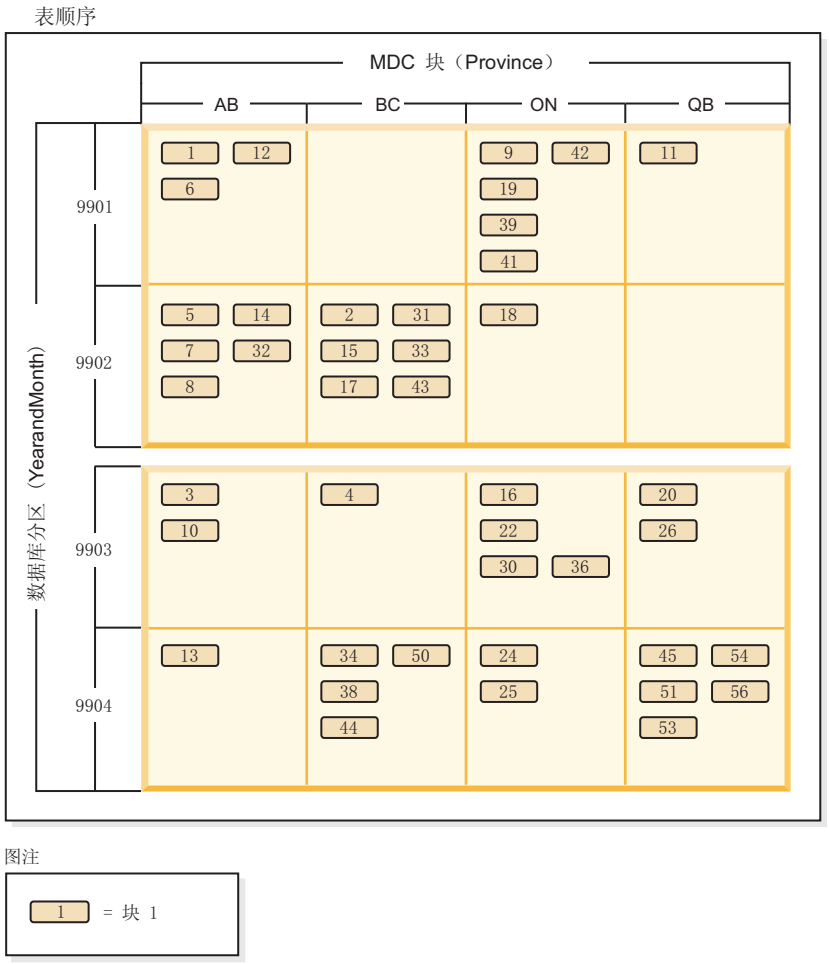


图 45. 按 YearAndMonth 分区并按 Province 和 YearAndMonth 组织的表

如果分区导致每个范围内只有单个值，那么通过在 MDC 键中包括表分区列不能获得任何好处。

注意事项

- 与基本表相比，MDC 表和分区表都需要一些存储器。这些存储器需求是附加的，但相对于所带来的好处来说，它们是合理的。
- 如果选择不在分区数据库环境中组合表分区和 MDC 功能，那么在您可以非常自信地预计数据分布情况时（通常也就是此处讨论的系统类型情况），使用表分区是最好的选择。否则，应考虑 MDC。
- 对于使用 DB2 V9.7 FP1 或更高发行版创建的数据分区 MDC 表，表的 MDC 块索引是分区索引。对于使用 DB2 V9.7 或更早发行版创建的数据分区 MDC 表，表的 MDC 块索引是非分区索引。

分区表的优化策略

“数据分区消除”功能是指数据库服务器能够根据查询谓词确定，只需要访问表的部分数据分区即可应答查询。对分区表运行决策支持查询时，“数据分区消除”功能特别有用。

分区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。根据 CREATE TABLE 语句的 PARTITION BY 子句中指定的内容，表的数据被划分到多个存储对象中。这些存储对象可以在不同的表空间中，也可以在同一个表空间中。

以下示例演示数据分区消除功能在性能方面的好处。

```
create table custlist(
  subsdate date, province char(2), accountid int)
  partition by range(subsdate) (
    starting from '1/1/1990' in ts1,
    starting from '1/1/1991' in ts1,
    starting from '1/1/1992' in ts1,
    starting from '1/1/1993' in ts2,
    starting from '1/1/1994' in ts2,
    starting from '1/1/1995' in ts2,
    starting from '1/1/1996' in ts3,
    starting from '1/1/1997' in ts3,
    starting from '1/1/1998' in ts3,
    starting from '1/1/1999' in ts4,
    starting from '1/1/2000' in ts4,
    starting from '1/1/2001'
    ending '12/31/2001' in ts4)
```

假定您只对 2000 年的客户信息感兴趣。

```
select * from custlist
  where subsdate between '1/1/2000' and '12/31/2000'
```

如图 46 所示，数据库服务器确定只需要访问表空间 TS4 中的一个数据分区即可解决此查询。

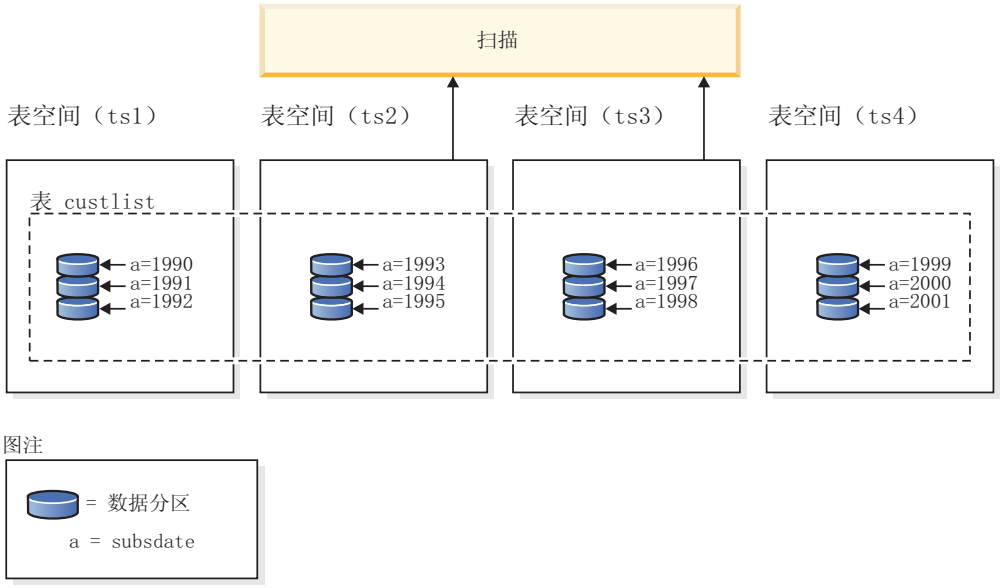


图 46. 数据分区消除功能在性能方面的好处

另一个数据分区消除示例基于以下方案:

```
create table multi (
  sale_date date, region char(2))
  partition by (sale_date) (
    starting '01/01/2005'
    ending '12/31/2005'
```

```
      every 1 month)

  create index sx on multi(sale_date)

  create index rx on multi(region)
```

假定您发出下列查询:

```
  select * from multi
    where sale_date between '6/1/2005'
      and '7/31/2005' and region = 'NW'
```

在不进行表分区时，一种可能的方案是索引“与”（AND）。索引“与”（AND）执行下列任务:

- 读取每个索引中的所有相关索引条目
- 保存两组行标识（RID）
- 对 RID 进行匹配，以确定哪些 RID 同时出现在这两个索引中
- 使用 RID 对行进行访存

如图 47 所示，在进行表分区的情况下，将读取索引以查找 REGION 和 SALE_DATE 的匹配项，从而快速检索匹配的行。

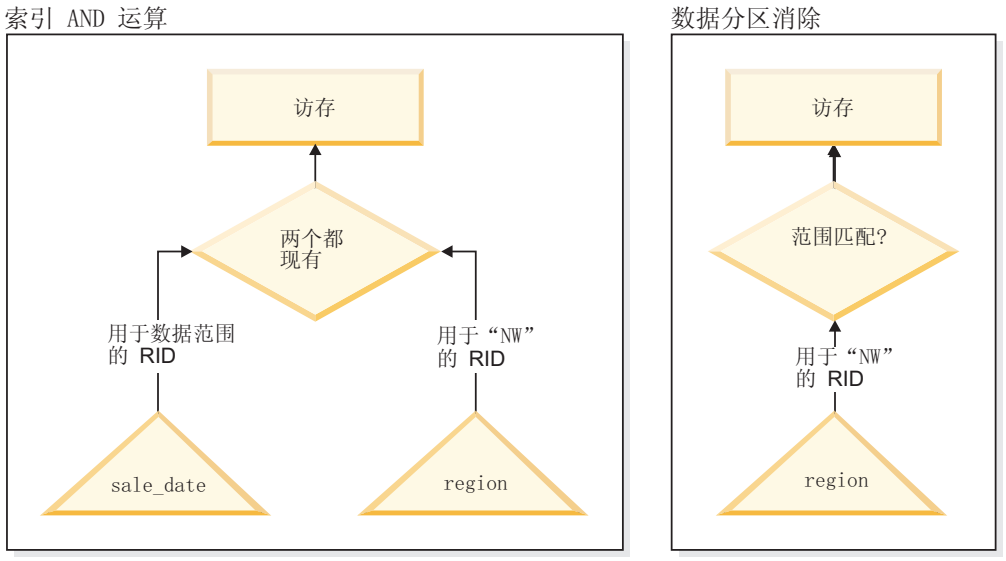


图 47. 用于进行表分区和索引“与”（AND）操作的优化器决策路径

DB2 Explain

您还可以使用说明工具来确定查询优化器所选择的数据分区消除方案。“DP Elim Predicates”信息显示扫描了哪些数据分区来解决下列查询:

```
  select * from custlist
    where subsdate between '12/31/1999' and '1/1/2001'

Arguments:
---------
DPESTFLG: (Number of data partitions accessed are Estimated)
     FALSE
DPLSTPRT: (List of data partitions accessed)
     9-11
DPNUMPRT: (Number of data partitions accessed)
```

```
   3

DP Elim Predicates:
------------------
Range 1)
   Stop  Predicate: (Q1.A <= '01/01/2001')
 Start Predicate: ('12/31/1999' <= Q1.A)



Objects Used in Access Plan:
 ---------------------------

   Schema: MRSRINI
   Name:    CUSTLIST
   Type:    Data Partitioned Table
   Time of creation:      2005-11-30-14.21.33.857039
   Last statistics update:    2005-11-30-14.21.34.339392
     Number of columns:      3
     Number of rows:      100000
   Width of rows:      19
   Number of buffer pool pages:    1200
     Number of data partitions:    12
     Distinct row values:      No
    Tablespace name:      <VARIOUS>
```

多列支持

在将多个列用作表分区键的情况下，数据分区消除功能将起作用。例如:

```
create table sales (
  year int, month int)
 partition by range(year, month) (
    starting from (2001,1)
    ending at (2001,3) in ts1,
   ending at (2001,6) in ts2,
   ending at (2001,9) in ts3,
   ending at (2001,12) in ts4,
   ending at (2002,3) in ts5,
   ending at (2002,6) in ts6,
   ending at (2002,9) in ts7,
   ending at (2002,12) in ts8)

select * from sales where year = 2001 and month < 8
```

查询优化器判断只需要访问 TS1、TS2 和 TS3 中的数据分区即可解决此查询。

注: 在表分区键由多个列构成的情况下，仅当存在作用于组合键的前导列的谓词时才能实现数据分区消除，这是因为，用于表分区键的非前导列不是独立的。

多范围支持

可以对包含多个范围的数据分区（即，通过“或”（OR）运算聚集到一起的数据分区）实现数据分区消除。在使用上一个示例中创建的 SALES 表的情况下，执行下列查询:

```
select * from sales
  where (year = 2001 and month <= 3)
    or (year = 2002 and month >= 10)
```

数据库服务器将只访问 2001 年第一季度和 2002 年最后一个季度的数据。

生成列

可以将生成列用作表分区键。例如:

```
create table sales (
  a int, b int generated always as (a / 5))
  in ts1,ts2,ts3,ts4,ts5,ts6,ts7,ts8,ts9,ts10
  partition by range(b) (
    starting from (0)
    ending at (1000) every (50))
```

在本示例中，将作用于生成列的谓词用于数据分区消除。此外，如果用于生成列的表达式是单调的，那么数据库服务器将把作用于源列的谓词转换为作用于生成列的谓词，从而对生成列启用数据分区消除。例如:

```
select * from sales where a > 35
```

数据库服务器根据作用于 a 的谓词（a > 35）来生成作用于 b 的额外谓词（b > 7），从而允许进行数据分区消除。

连接谓词

如果将连接谓词下推到表访问级别，那么连接谓词也可以用于数据分区消除。只有嵌套循环连接（NLJN）的内连接才会将连接谓词下推到表访问级别。

请考虑下列表:

```
create table t1 (a int, b int)
  partition by range(a,b) (
    starting from (1,1)
    ending (1,10) in ts1,
    ending (1,20) in ts2,
    ending (2,10) in ts3,
    ending (2,20) in ts4,
    ending (3,10) in ts5,
    ending (3,20) in ts6,
    ending (4,10) in ts7,
    ending (4,20) in ts8)

create table t2 (a int, b int)
```

将使用下面这两个谓词:

```
P1: T1.A = T2.A
P2: T1.B > 15
```

在本示例中，由于不知道连接的外值，因此无法确定将在编译时访问的确切数据分区。在这种情况下，以及在使用主变量或参数标记的情况下，将在运行时绑定必需的值时进行数据分区消除。

在运行时，当 T1 是 NLJN 的内表时，将根据 T2.A 的每个外值的谓词自动进行数据分区消除。在运行时，将对外值 T2.A = 3 应用谓词 T1.A = 3 和 T1.B > 15，这样就限定了在表空间 TS6 中要访问的数据分区。

假定表 T1 和 T2 中的列 A 包含下列值:

外表 T2: 列 A	内表 T1: 列 A	内表 T1: 列 B	内表 T1: 数据分区位置
2	3	20	TS6
3	2	10	TS3
3	2	18	TS4
	3	15	TS6
	1	40	TS3

要执行嵌套循环连接（假定对内表进行表扫描），数据库管理器执行下列步骤:

1. 读取 T2 中的第一行。A 的值是 2。
2. 在连接谓词 T1.A = T2.A 中将 T2.A 值（此值为 2）与列 T2.A 绑定。该谓词变成 T1.A = 2。
3. 使用谓词 T1.A = 2 和 T1.B > 15 来应用数据分区消除。这将限定表空间 TS4 中的数据分区。
4. 在应用 T1.A = 2 和 T1.B > 15 之后，扫描表 T1 的表空间 TS4 中的数据分区，直到找到一行为止。找到的第一个合格行是 T1 的行 3。
5. 连接匹配的行。
6. 扫描表 T1 的表空间 TS4 中的数据分区，直到找到下一个匹配项（T1.A = 2 并且 T1.B > 15）为止。再也找不到其他行。
7. 对 T2 的下一行（将 A 的值替换为 3）重复步骤 1 至 6，直到处理完 T2 的所有行为止。

基于 XML 数据的索引

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。

在表插入、更新和删除操作期间，数据库管理器将按照维护任何其他基于分区表的关系索引的方式来维护分区 XML 索引和非分区 XML 索引。为了提高查询处理速度，将按照使用基于非分区表中 XML 数据的索引的方式来使用基于分区表中 XML 数据的非分区索引。通过使用查询谓词，有可能确定只需要访问分区表中的部分数据分区即可应答查询。

基于 XML 列的数据分区消除功能和索引可以配合工作，以提高查询性能。请考虑以下分区表:

```
create table employee (a int, b xml, c xml)
  index in tbspx
  partition by (a) (
    starting 0 ending 10,
    ending 20,
    ending 30,
    ending 40)
```

现在，考虑下列查询:

```
select * from employee
  where a > 21
  and xmlexist('$doc/Person/Name/First[.="Eric"]'
    passing "EMPLOYEE"."B" as "doc")
```

优化器可以根据谓词 a > 21 立即消除前两个分区。如果优化器在查询方案中选择基于列表 B 中 XML 数据的非分区索引，那么使用基于 XML 数据的索引的索引扫描能够利用优化器的数据分区消除结果，并且将只返回属于关系数据分区消除谓词未消除的分区的结果。

MDC 表的优化策略

如果创建多维集群 (MDC) 表，那么可以提高许多查询的性能，这是因为优化器可以应用附加的优化策略。这些策略主要依赖于块索引效率有所提高，但根据多个维进行集群这一优点还能提高数据检索速度。

MDC 表优化策略还可以利用分区内并行性和分区间并行性的性能优点。请考虑 MDC 表的下列具体优点:

- 维块索引查找操作可以标识表的所需部分，并且能够快速地仅扫描所需的块。
- 因为块索引小于记录标识（RID）索引，所以查找速度更快。
- 可以在块级别执行索引 AND 和 OR 运算，并可以将这些运算与 RID 相结合。
- 保证在扩展数据块内集群数据，这有助于提高检索速度。
- 如果可以使用转出方法，那么删除行的速度将更快。

请考虑名为 SALES 的 MDC 表的以下简单示例，这个表对 REGION 和 MONTH 列定义了维:

```
select * from sales
  where month = 'March' and region = 'SE'
```

对于此查询，优化器可以执行维块索引查找操作，以寻找月份为三月且地区为 SE 的块。然后，它可以只扫描那些块，以便快速地访存结果集。

转出删除

当条件允许使用转出方法来进行删除时，将使用这种从 MDC 表中删除行的更高效方法。必需的条件包括:

- 该 DELETE 语句是搜索型 DELETE，而不是定位型 DELETE（该语句不使用 WHERE CURRENT OF 子句）。
- 没有 WHERE 子句（将删除所有行），或者 WHERE 子句只包含应用于维的条件。
- 定义表时，未指定 DATA CAPTURE CHANGES 子句。
- 该表不是引用完整性关系中的父表。
- 未对该表定义 ON DELETE 触发器。
- 未在任何立即刷新的 MQT 中使用该表。
- 如果级联删除操作的外键是该表的维列的子集，那么它可能适合于转出。
- 在由 CREATE TRIGGER 语句的 OLD TABLE AS 子句指定的触发 SQL 操作之前，该 DELETE 语句不能出现在对临时表执行并标识了受影响行集的 SELECT 语句中。

在转出删除期间，不会记录所删除的记录。而是，将通过重新格式化页的某些部分使包含这些记录的页表现为空页。将会记录对重新格式化的部分所作的更改，但不会记录这些记录本身。

*立即清除转出*这一缺省行为是指，在删除时清除 RID 索引。还可以通过将注册表变量 **DB2_MDC_ROLLOUT** 设为 IMMEDIATE，或者通过对 SET CURRENT MDC ROLLOUT MODE 语句指定 IMMEDIATE 来指定此方式。与标准删除操作相比，索引更新的日志记录没有变化，因此，性能提高取决于 RID 索引的数目。RID 索引越少，性能就越好，衡量标准是总时间和日志空间所占的百分比。

可以使用以下公式来估算可以节省的日志空间量:

```
S + 38*N - 50*P
```

其中，*N* 是已删除的记录数，*S* 是已删除的记录的总大小（包括空指示符和 VARCHAR 长度之类的开销），*P* 是包含已删除的记录的块中的页数。此数值是实际日志数据的缩减量。节省的所需活动日志空间量是此值的两倍，这是因为，还将节省为回滚操作保留的空间。

另外，在落实事务之后，可以使用*延迟清除转出*方法来更新 RID 索引。还可以通过将注册表变量 **DB2_MDC_ROLLOUT** 设为 DEFER，或者通过对 SET CURRENT MDC ROLLOUT MODE 语句指定 DEFERRED 来指定此方式。在延迟转出方式下，将在删除操作落实后在后台以异步方式清除 RID 索引。在删除任务非常大型或者已对表定义大量 RID 索引的情况下，使用这种转出方法可以非常快速地执行删除。整体清除操作的速度也有所提高，这是因为，执行延迟索引清除时将以并行方式清除索引，而执行立即索引清除时将逐行清除索引中的每一行。并且，DELETE 语句的事务日志空间需求显著降低，这是因为，索引按索引页而不是按索引键来更新异步索引清除日志。

注: 延迟清除转出操作需要更多内存资源，这些内存资源将从数据库堆中获取。如果数据库管理器无法分配它所需的内存结构，那么延迟清除转出操作将失败，并将一条消息写入管理通知日志。

何时使用延迟清除转出方法

如果删除性能对于您而言是最重要的因素，并且已对表定义 RID 索引，那么应使用延迟清除转出方法。注意，在进行索引清除之前，对已转出的块进行基于索引的扫描会稍微降低性能，这取决于已转出的数据量。在决定执行立即索引清除操作和延迟索引清除操作时，还应该考虑下列问题:

- 删除操作的规模

  对于非常大型的删除任务，请选择延迟清除转出方法。在对许多小型 MDC 表频繁发出维 DELETE 语句的情况下，异步清除索引对象所产生的开销要比删除操作期间节省的时间的价值更高。
- 索引的数目和类型

  如果表包含大量 RID 索引，并且需要对这些索引执行行级别处理，那么应使用延迟清除转出方法。
- 块可用性

  如果您希望由删除操作释放的块空间在 DELETE 语句落实后立即可用，那么请使用立即清除转出方法。
- 日志空间

  如果日志空间有限，那么应对大型删除任务使用延迟清除转出方法。
- 内存约束

  对于所有已暂挂延迟清除操作的表，延迟清除转出操作将耗用更多的数据库堆空间。

要在删除期间禁止转出行为，请将 **DB2_MDC_ROLLOUT** 注册表变量设为 OFF，或者对 SET CURRENT MDC ROLLOUT MODE 语句指定 NONE。

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。

第 21 章 索引

分区表中的索引

分区表的索引行为

分区表的索引的工作方式与非分区表的索引类似。但是，分区表的索引是使用不同存储模型存储的（根据索引是分区索引还是非分区索引）。

尽管常规非分区表的索引全都驻留在共享的索引对象中，但分区表的*非分区索引*将在单一表空间中它自己的索引对象中创建，即使数据分区跨多个表空间也是如此。数据库管理的空间 (DMS) 和系统管理的空间 (SMS) 表空间都支持使用不同于表数据所在位置的位置中的索引。可以将每个非分区索引放入它自己的表空间，其中包括大型表空间。每个索引表空间都必须使用与数据分区相同的存储机制，即 DMS 或 SMS。大型表空间中的索引可以包含多达 2^{29} 页。所有表空间都必须在同一个数据库分区组中。

*分区索引*使用了索引组织方案，即，索引数据根据表的分区方案划分到多个*索引分区*中。每个索引分区都只引用相应数据分区中的表行。特定数据分区的所有索引分区都驻留在同一个索引对象中。

从 DB2 V9.7 FP1 开始，用户根据分区表中 XML 列的 XML 数据创建的索引可以是分区索引或非分区索引。缺省情况下为分区索引。系统生成的 XML 区域索引始终为分区索引，而系统生成的列路径索引始终为非分区索引。在 DB2 V9.7 中，基于 XML 数据的索引为非分区索引。

非分区索引的优势包括:

- 能够为每个索引定义不同的表空间特征（例如，不同的页大小可能有助于确保更好地利用空间）
- 可以相互独立的方式重组索引
- 能够提高删除索引操作的性能
- 减少 I/O 争用，这有助于更高效地对索引数据进行并发访问
- 删除各个索引时，空间将立即可供系统使用，而无需进行索引重组

分区索引的优势包括:

- 能够提高数据滚入和滚出性能
- 由于索引进行分区，因此能够减少对索引页的争用
- 每个索引分区均采用索引 B 树结构，这有如下优点:
  - 提高插入、更新、删除和扫描性能，这是因为，索引分区的 B 树所包含的层数通常少于引用表中所有数据的索引
  - 改进分区消除生效时的扫描性能和并行性。尽管分区消除功能既可用于分区索引扫描也可用于非分区索引扫描，但用于分区索引扫描却更为有效，这是因为，每个索引分区都只包含相应数据分区的键。此配置可导致必须扫描的键数和索引页数少于对非分区索引执行的类似查询。

虽然非分区索引始终保留索引列的顺序，但分区索引在某些情况下可能会在各分区之间丢失一些顺序；例如，如果分区列与索引列不匹配，并且将访问多个分区。

在联机索引创建期间，允许对表进行并行读写访问。构建联机索引后，在创建索引期间对表进行的更改将应用于新索引。对该表所作的写访问将被阻塞，直到索引创建完成并且事务落实为止。对于分区索引而言，仅当应用创建索引分区期间对数据分区所作的更改时，才会停顿每个数据分区以便进行只读访问。

当您使用 ALTER TABLE...ATTACH PARTITION 语句滚入数据时，分区索引支持变得特别有用。如果存在非分区索引（不包括 XML 列路径索引，如果表包含 XML 数据的话），请在连接分区之后发出 SET INTEGRITY 语句。非分区索引维护、范围验证、约束检查和具体化查询表 (MQT) 维护需要此语句。非分区索引的维护工作可能相当耗时并需要大量日志空间。请使用分区索引来避免此维护成本。

如果表上有一些非分区索引（XML 列路径索引除外）在连接操作后要保留，那么 SET INTEGRITY...ALL IMMEDIATE UNCHECKED 语句的行为方式就像 SET INTEGRITY...IMMEDIATE CHECKED 语句一样。所有完整性处理、非分区索引维护和表状态过渡就像发出 SET INTEGRITY...IMMEDIATE CHECKED 语句一样执行。

图 48 图显示了分区表的两个非分区索引，这两个索引在不同的表空间中。

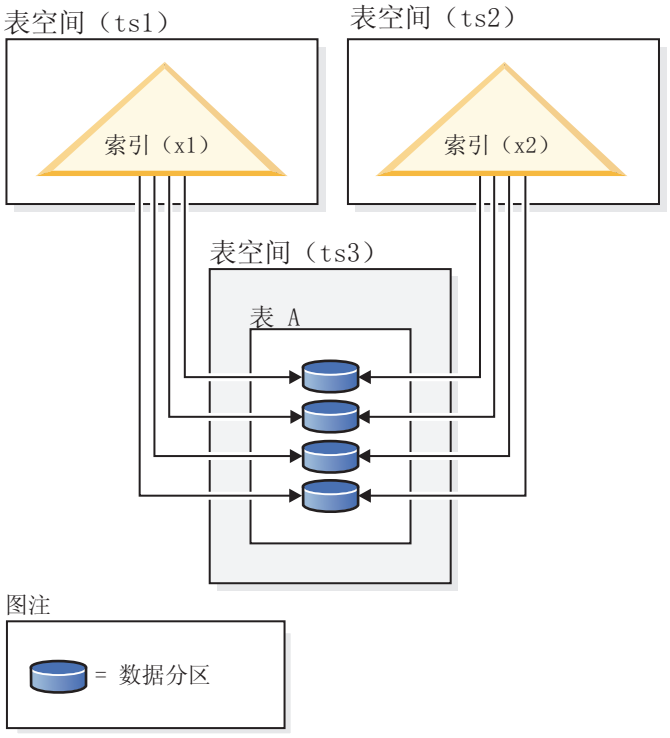


图 48. 分区表的非分区索引

第 281 页的图 49 图显示了分区表的分区索引，此索引跨两个数据库分区并驻留在单一表空间中。

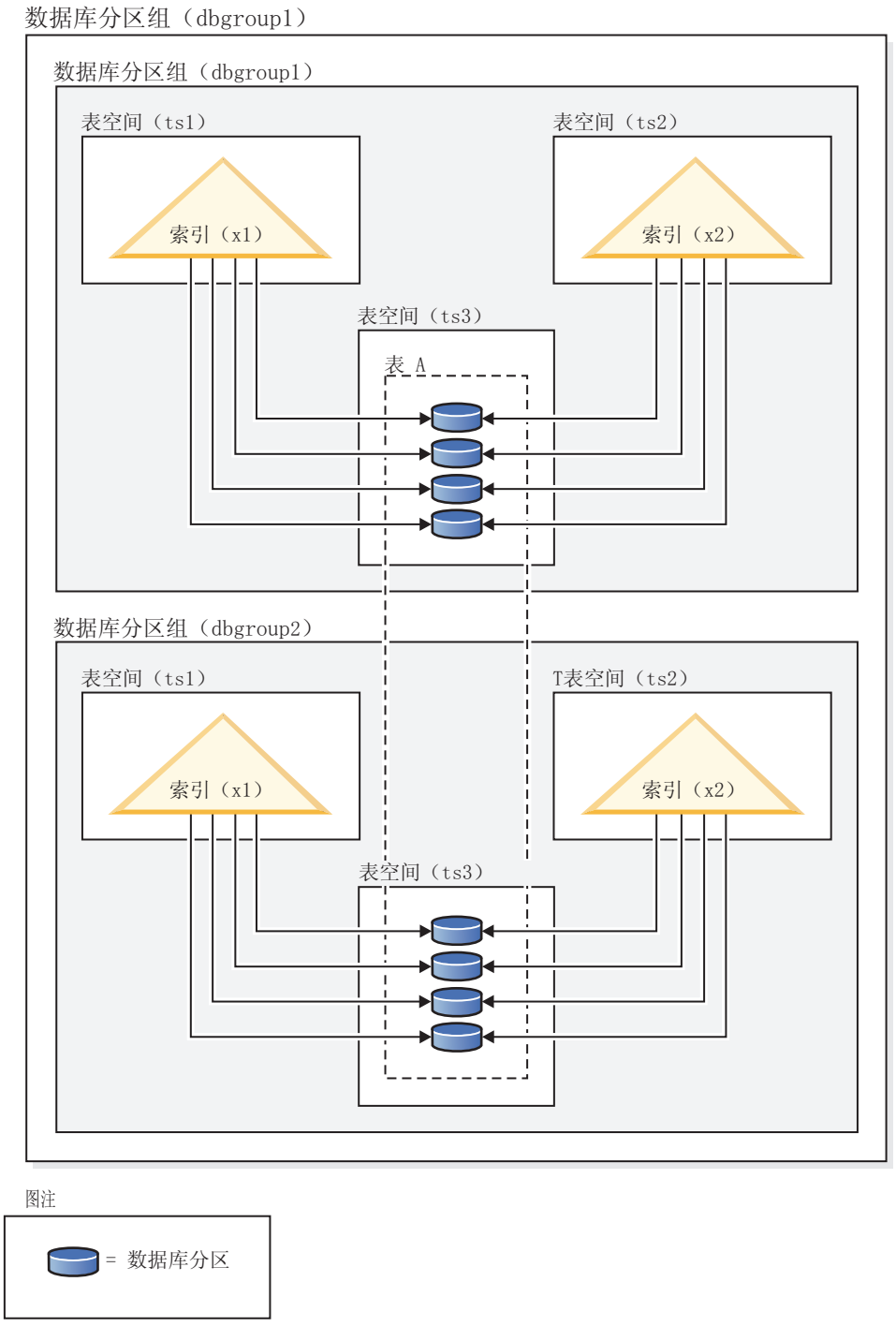


图 49. 分布式分区表的非分区索引

第 282 页的图 50 图显示了分区表的混合分区索引和非分区索引。

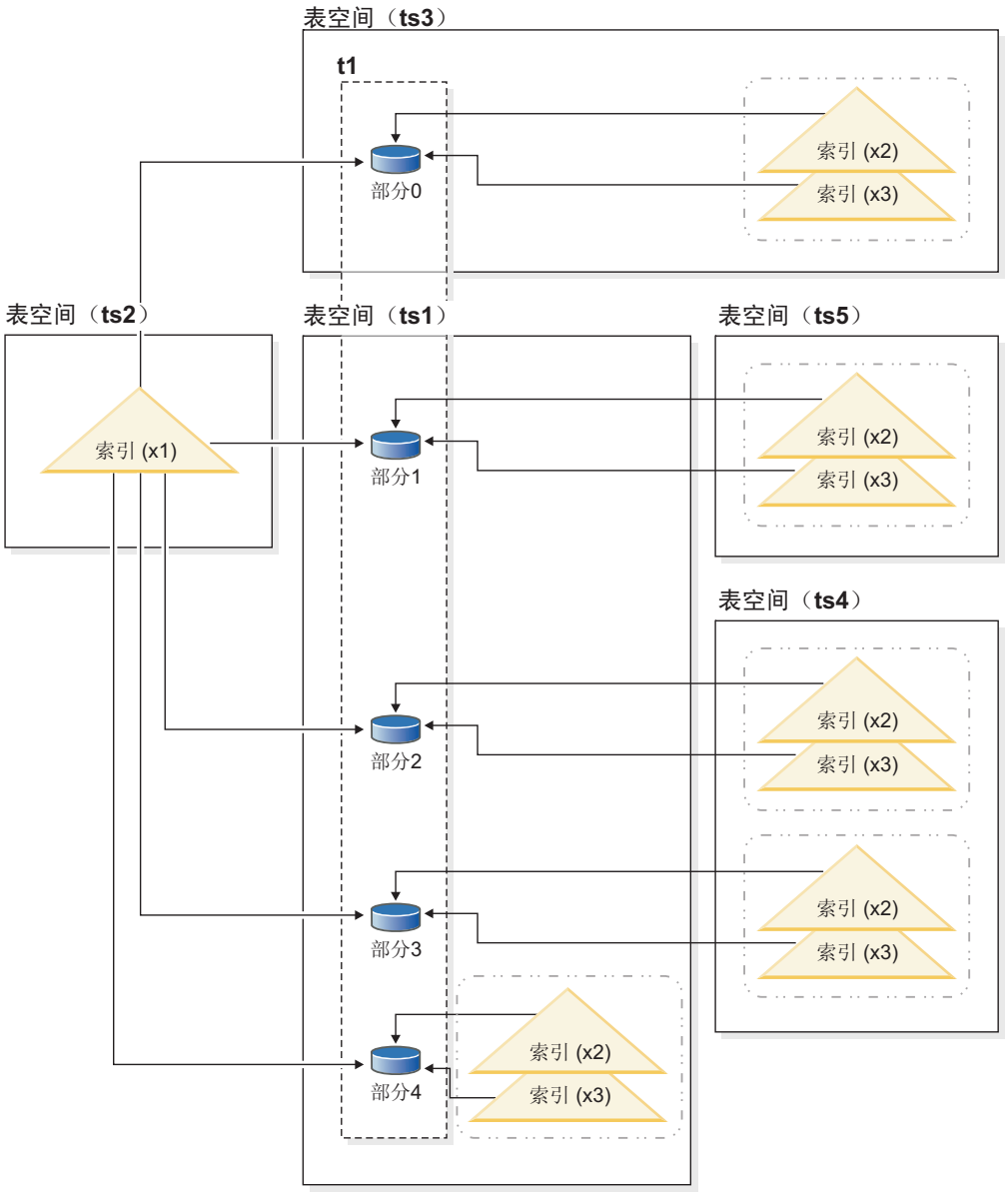


图 50. 分区表的分区索引和非分区索引

非分区索引 X1 引用所有数据分区中的行。相反，分区索引 X2 和 X3 只引用与其相关联的数据分区中的行。表空间 TS3 还显示了一些索引分区，这些索引分区共享与其相关联的数据分区的表空间。对于分区索引而言，此配置是缺省情况。

您可以覆盖非分区索引和分区索引的缺省位置，尽管为这两种索引执行此操作的方法有所不同。对于非分区索引，可以在创建该索引时指定表空间；对于分区索引，您需要在创建该表时确定用于存储索引分区的表空间。

非分区索引

要覆盖非分区索引的索引位置，请使用 CREATE INDEX 语句的 IN 子句为索引指定另一个表空间位置。根据需要，可以将不同的索引放入不同的表空间。如果创建分区表时未指定它的非分区索引的放置位置，并且使用未指定表空间的 CREATE INDEX 语句来创建索引，那么将在已连接的第一个数据分区或可

视数据分区的表空间中创建该索引。按顺序对下面三种可能情况进行评估（从情况 1 开始），以确定创建索引的位置。找到匹配的情况时，此项用于确定索引的表空间位置的评估即停止。

情况 1:
如果将索引表空间指定于 CREATE INDEX...IN *tbspace* 语句中，那么将指定的表空间用于此索引。

情况 2:
如果将索引表空间指定于 CREATE INDEX...IN INDEX IN *tbspace* 语句中，那么将制定的表空间用于索引。

情况 3:
未指定任何表空间时，选择已连接的第一个数据分区或可视数据分区所使用的表空间。

分区索引

缺省情况下，索引分区将被放入它们所引用的数据分区所在的表空间。要覆盖这种缺省行为，必须对您使用 CREATE TABLE 语句定义的每个数据分区使用 INDEX IN 子句。换而言之，如果您计划对分区表使用分区索引，那么必须在创建该表时预测索引分区的存储位置。如果您尝试在创建分区索引时使用 INDEX IN 子句，那么将接收到错误消息。

示例 1: 给定分区表 SALES (a int, b int, c int)，创建唯一索引 A_IDX。

```
create unique index a_idx on sales (a)
```

由于表 SALES 是分区表，因此索引 a_idx 也将被创建为分区索引。

示例 2: 创建索引 B_IDX。

```
create index b_idx on sales (b)
```

示例 3: 覆盖分区索引中索引分区的缺省位置，对您创建分区表时定义的每个分区使用 INDEX IN 子句。在以下示例中，将在表空间 TS3 中创建表 Z 的索引。

```
create table z (a int, b int)
      partition by range (a) (starting from (1)
      ending at (100)  index in ts3)

   create index c_idx on z (a) partitioned
```

分区表的非分区索引的集群

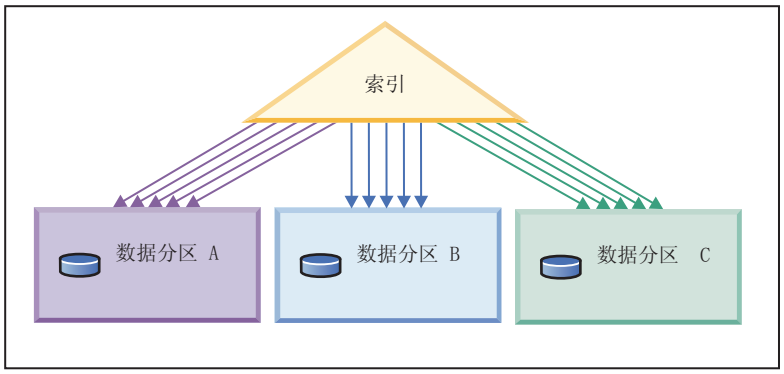
对分区表使用集群索引可获得与对常规表使用集群索引相同的好处。但是，在选择集群索引时，必须谨慎地对待表分区键定义。

您可以使用任何集群键对分区表创建集群索引。数据库服务器将尝试使用集群索引以本地方式对每个数据分区中的数据进行集群。在集群插入操作期间，将执行索引查找操作以查找适合的记录标识 (RID)。将从此 RID 开始在表中寻找空间以插入记录。为了实现性能优良的最佳集群，索引列与表分区键列之间应该存在关联。确保这种关联的一种方法是将表分区键列置于索引列之前，如以下示例所示:

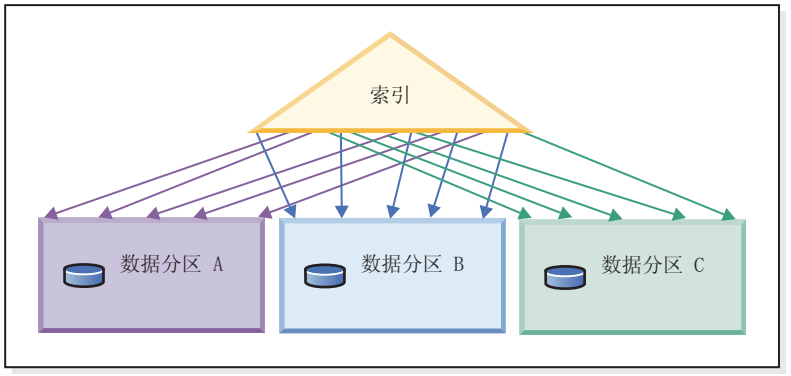
```
partition by range (month, region)
create index...(month, region, department) cluster
```

虽然数据库服务器不强制此关联，但还是希望索引中的所有键按分区标识进行聚集，以实现优良的集群。例如，假定一个表按 QUARTER 进行分区，并且对 DATE 定义了集群索引。在 QUARTER 与 DATE 之间存在关系，由于任何数据分区的所有键在该索引中都聚集在一起，因此能够实现性能良好的最佳数据集群。第 285 页的图 51 表明，仅当集群与表分区键相关时，才能实现最佳的扫描性能。

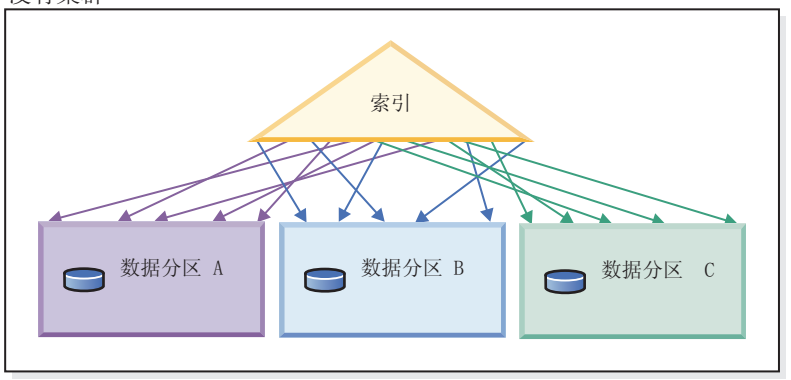
将分区键作为前缀的集群（相关）



集群与分区键不匹配（以本地方式集群）



没有集群



图注

图 51. 集群索引可能对分区表产生的影响。

集群的好处包括:

- 在每个数据分区中，各个行按键顺序排列。
- 集群索引能够提高按键顺序遍历表的扫描的性能，这是因为扫描者将先访存第一页的第一行，接着访存该页的每一行，然后再移至下一页。这意味着，在任何给定时

间，缓冲池都只需要包含表的一页。相反，如果未对表进行集群，那么有可能访存不同的页中的行。除非缓冲池能够容纳整个表，否则大多数页都可能被访存多次，从而导致扫描速度显著下降。

如果集群键与表分区键不相关，但数据以局部方式进行集群，那么当缓冲池有足够空间来容纳每个数据分区的一页时，仍然可以获得集群索引的全部好处。这是因为，从特定数据分区中访存的每一行都在先前从该分区中访存的行附近（参见第 285 页的图 51 中的第二个示例）。

第 22 章 设计顾问程序

使用设计顾问程序将单分区数据库转换为多分区数据库

您可以使用设计顾问程序帮助您将单分区数据库转换为多分区数据库。

关于此任务

除了提供有关新索引、具体化查询表 (MQT) 和多维集群 (MDC) 表的建议以外，设计顾问程序还可向您提供有关如何分布数据的建议。

过程

1. 使用 **db2licm** 命令来注册分区数据库环境许可证密钥。
2. 在多分区数据库分区组中至少创建一个表空间。

   注: 设计顾问程序只能提供有关如何将数据重新分布到现有表空间的建议。
3. 运行设计顾问程序，并在 **db2advis** 命令中指定分区选项。
4. 在运行设计顾问程序生成的 DDL 语句之前，请稍微修改 **db2advis** 输出文件。 由于必须先设置数据库分区方式，然后才能运行设计顾问程序所生成的 DDL 脚本，因此会在返回的脚本中将建议注释掉。您负责按照这些建议对表进行变换。

第 23 章 管理并行性

MDC 表和 ITC 表及 RID 索引扫描的锁定方式

多维集群 (MDC) 表或插入时间集群 (ITC) 表在表或 RID 索引扫描期间获取的锁定类型取决于生效中的隔离级别以及所使用的数据存取方案。

下列各表列示不同存取方案在各种隔离级别下获取的 MDC 表和 ITC 表锁定类型。每个条目都分为三部分：表锁定、块锁定和行锁定。连字符表明特定的锁定粒度不可用。

表 9-14 列示了将数据页读操作推迟时获取的 RID 索引扫描锁定类型。在 UR 隔离级别下，如果存在应用于索引中包括列的谓词，那么隔离级别将升级到 CS，并且锁定将升级到 IS 表锁定、IS 块锁定或 NS 行锁定。

- 表 1. 不使用谓词的表扫描的锁定方式
- 表 2. 仅使用维列上的谓词的表扫描的锁定方式
- 表 3. 使用其他谓词（sargs 和 resids）的表扫描的锁定方式
- 表 4. 不使用谓词的 RID 索引扫描的锁定方式
- 表 5. 使用单个合格行的 RID 索引扫描的锁定方式
- 表 6. 仅使用 Start 和 Stop 谓词的 RID 索引扫描的锁定方式
- 表 7. 仅使用索引谓词的 RID 索引扫描的锁定方式
- 表 8. 使用其他谓词（sargs 和 resids）的 RID 索引扫描的锁定方式
- 表 9. 用于延迟型数据页访问的索引扫描的锁定方式：不使用谓词的 RID 索引扫描
- 表 10. 用于延迟型数据页访问的索引扫描的锁定方式：在不使用谓词的 RID 索引扫描之后
- 表 11. 用于延迟型数据页访问的索引扫描的锁定方式：使用谓词（sargs 和 resids）的 RID 索引扫描
- 表 12. 用于延迟型数据页访问的索引扫描的锁定方式：在使用谓词（sargs 和 resids）的 RID 索引扫描之后
- 表 13. 用于延迟型数据页访问的索引扫描的锁定方式：仅使用 Start 和 Stop 谓词的 RID 索引扫描
- 表 14. 用于延迟型数据页访问的索引扫描的锁定方式：在仅使用 Start 和 Stop 谓词的 RID 索引扫描之后

注： 可以使用 SELECT 语句的 *lock-request-clause* 显式地更改锁定方式。

表 16. 不使用谓词的表扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	S/-/-	U/-/-	SIX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/U	IX/X/-	IX/I/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

表 16. 不使用谓词的表扫描的锁定方式 (续)

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

表 17. 仅使用维列上的谓词的表扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/X/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/-	X/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/U/-	X/X/-

表 18. 使用其他谓词（sargs 和 resids）的表扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	S/-/-	U/-/-	SIX/IX/X	U/-/-	SIX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 19. 不使用谓词的 RID 索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	S/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

表 20. 使用单个合格行的 RID 索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/IS/S	IX/IX/U	IX/IX/X	X/X/X	X/X/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/X	X/X/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/X	X/X/X

表 21. 仅使用 Start 和 Stop 谓词的 RID 索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/IS/S	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

表 21. 仅使用 Start 和 Stop 谓词的 RID 索引扫描的锁定方式 (续)

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

表 22. 仅使用索引谓词的 RID 索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 23. 使用其他谓词（sargs 和 resids）的 RID 索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S/S	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 24. 用于延迟型数据页访问的索引扫描的锁定方式: 不使用谓词的 RID 索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S/S	IX/IX/S		X/-/-	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

表 25. 用于延迟型数据页访问的索引扫描的锁定方式: 在不使用谓词的 RID 索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IN/IN/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/X	IX/IX/X

表 26. 用于延迟型数据页访问的索引扫描的锁定方式：使用谓词（sargs 和 resids）的 RID 索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S/-	IX/IX/S		IX/IX/S	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

表 27. 用于延迟型数据页访问的索引扫描的锁定方式：在使用谓词（sargs 和 resids）的 RID 索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 28. 用于延迟型数据页访问的索引扫描的锁定方式：仅使用 Start 和 Stop 谓词的 RID 索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/IS/S	IX/IX/S		IX/IX/X	
RS	IN/IN/-	IN/IN/-		IN/IN/-	
CS	IN/IN/-	IN/IN/-		IN/IN/-	
UR	IN/IN/-	IN/IN/-		IN/IN/-	

表 29. 用于延迟型数据页访问的索引扫描的锁定方式：在仅使用 Start 和 Stop 谓词的 RID 索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IN/IN/-	IX/IX/S	IX/IX/X	IX/IX/X	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IS/-/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

MDC 块索引扫描的锁定方式

多维集群 (MDC) 表在块索引扫描期间获取的锁定类型取决于生效中的隔离级别以及所使用的数据存取方案。

下列各表列示不同存取方案在各种隔离级别下获取的 MDC 表锁定类型。每个条目都分为三部分：表锁定、块锁定和行锁定。连字符表明特定的锁定粒度不可用。

表 5-12 列示了将数据页读操作推迟时获取的块索引扫描锁定类型。

- 表 1. 不使用谓词的索引扫描的锁定方式
- 表 2. 仅使用维列上的谓词的谓词扫描的锁定方式
- 表 3. 仅使用 Start 和 Stop 谓词的索引扫描的锁定方式
- 表 4. 使用谓词的索引扫描的锁定方式
- 表 5. 用于延迟型数据页访问的索引扫描的锁定方式: 不使用谓词的块索引扫描
- 表 6. 用于延迟型数据页访问的索引扫描的锁定方式: 在不使用谓词的块索引扫描之后
- 表 7. 用于延迟型数据页访问的索引扫描的锁定方式: 仅对维列使用谓词的块索引扫描
- 表 8. 用于延迟型数据页访问的索引扫描的锁定方式: 在仅对维列使用谓词的块索引扫描之后
- 表 9. 用于延迟型数据页访问的索引扫描的锁定方式: 仅使用 Start 和 Stop 谓词的块索引扫描
- 表 10. 用于延迟型数据页访问的索引扫描的锁定方式: 在仅使用 Start 和 Stop 谓词的块索引扫描之后
- 表 11. 用于延迟型数据页访问的索引扫描的锁定方式: 使用其他谓词（sargs 和 resids）的块索引扫描
- 表 12. 用于延迟型数据页访问的索引扫描的锁定方式: 在使用其他谓词（sargs 和 resids）的块索引扫描之后

注: 可以使用 SELECT 语句的 *lock-request-clause* 显式地更改锁定方式。

表 30. 不使用谓词的索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	S/--/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/-	IX/IX/U	IX/IX/X	X/X/--	X/X/--

表 31. 仅使用维列上的谓词的索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/-/-	IX/IX/S	IX/IX/X	X/-/-	X/-/-
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/X/-	IX/X/-

表 32. 仅使用 Start 和 Stop 谓词的索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S/-	IX/IX/S	IX/IX/S	IX/IX/S	IX/IX/S

表 32. 仅使用 Start 和 Stop 谓词的索引扫描的锁定方式 (续)

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
CS	IX/IX/S	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/-	IX/IX/-

表 33. 使用谓词的索引扫描的锁定方式

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S/-	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/-	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

表 34. 用于延迟型数据页访问的索引扫描的锁定方式: 不使用谓词的块索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S/--	IX/IX/S		X/--/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

表 35. 用于延迟型数据页访问的索引扫描的锁定方式: 在不使用谓词的块索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IN/IN/--	IX/IX/S	IX/IX/X	X/--/--	X/--/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	X/X/--	X/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	X/X/--	X/X/--

表 36. 用于延迟型数据页访问的索引扫描的锁定方式: 仅对维列使用谓词的块索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S/--	IX/IX/--		IX/S/--	
RS	IS/IS/NS	IX/--/--		IX/--/--	
CS	IS/IS/NS	IX/--/--		IX/--/--	
UR	IN/IN/--	IX/--/--		IX/--/--	

表 37. 用于延迟型数据页访问的索引扫描的锁定方式: 在仅对维列使用谓词的块索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/S/--	IX/X/--
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/U/--	IX/X/--

表 38. 用于延迟型数据页访问的索引扫描的锁定方式: 仅使用 Start 和 Stop 谓词的块索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S/--	IX/IX/--		IX/X/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

表 39. 用于延迟型数据页访问的索引扫描的锁定方式: 在仅使用 Start 和 Stop 谓词的块索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IN/IN/--	IX/IX/X		IX/X/--	
RS	IS/IS/NS	IN/IN/--		IN/IN/--	
CS	IS/IS/NS	IN/IN/--		IN/IN/--	
UR	IS/--/--	IN/IN/--		IN/IN/--	

表 40. 用于延迟型数据页访问的索引扫描的锁定方式: 使用其他谓词（sargs 和 resids）的块索引扫描

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IS/S/--	IX/IX/--		IX/IX/--	
RS	IN/IN/--	IN/IN/--		IN/IN/--	
CS	IN/IN/--	IN/IN/--		IN/IN/--	
UR	IN/IN/--	IN/IN/--		IN/IN/--	

表 41. 用于延迟型数据页访问的索引扫描的锁定方式: 在使用其他谓词（sargs 和 resids）的块索引扫描之后

隔离级别	只读和模糊扫描	游标操作		搜索型更新或删除	
		扫描	当前位置	扫描	更新或删除
RR	IN/IN/--	IX/IX/S	IX/IX/X	IX/IX/S	IX/IX/X
RS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
CS	IS/IS/NS	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X
UR	IN/IN/--	IX/IX/U	IX/IX/X	IX/IX/U	IX/IX/X

对分区表的锁定行为

除锁定整个表以外，还可以锁定分区表的每个数据分区。

与非分区表相比，这支持更高的粒度并提高了并行性。**db2pd** 命令、事件监视器、管理视图和表函数的输出都将标识数据分区锁定。

访问表时，将首先获取表锁定，然后根据需要获取数据分区锁定。访问方法和隔离级别可能要求锁定结果集未涉及到的数据分区。获取这些数据分区锁定之后，在表锁定保持期间可能会一直挂起这些锁定。例如，对索引执行的游标稳定性 (CS) 扫描可能保持对先前访问的数据分区的锁定，以便降低以后重新获取数据分区锁定的成本。

数据分区锁定还承担用于确保对表空间进行访问的成本。对于非分区表而言，表空间访问由表锁定处理。即使在表级别存在互斥锁定或共享锁定，也会进行数据分区锁定。

更高的粒度允许一个事务对特定数据分区具有互斥访问权并避免进行行锁定，而其他事务能够访问其他数据分区。这可能是为批量更新选择的方案或者将锁定升级到数据分区级别的结果。许多访问方法的表锁定通常是意向锁定，即使以共享或互斥方式锁定数据分区亦如此。这将提高并行性。但是，如果在数据分区级别需要非意向锁定并且方案表明可能会访问所有数据分区，那么可能会在表级别选择非意向锁定，以防止并发事务之间发生数据分区死锁。

LOCK TABLE 语句

对于分区表而言，LOCK TABLE 语句获取的唯一锁定是表级别锁定。这将防止后续数据操作语言 (DML) 语句执行行锁定，并避免在行、块或数据分区级别出现死锁情况。更新索引时，可以使用 IN EXCLUSIVE MODE 选项来保证互斥访问，这对于在大型更新期间限制索引增大很有用。

ALTER TABLE 语句的 LOCKSIZE TABLE 选项的影响

LOCKSIZE TABLE 选项确保以共享或互斥方式来锁定表，而不进行意向锁定。对于分区表而言，这种锁定策略将同时应用于表锁定和数据分区锁定。

行级别锁定和块级锁定升级

分区表中的行级别锁定和块级锁定可以升级到数据分区级别。发生这种情况后，其他事务可以更容易地访问该表，即使数据分区锁定升级到共享、互斥或超级互斥方式亦如此，这是因为其他数据分区保持不受影响。升级操作的通知日志条目指示了所影响的数据分区以及该表的名称。

锁定升级无法确保对非分区索引进行互斥访问。要进行互斥访问，必须符合下列其中一个条件:

- 语句必须使用互斥表级别锁定
- 必须发出显式的 LOCK TABLE IN EXCLUSIVE MODE 语句
- 该表必须具有 LOCKSIZE TABLE 属性

对于分区索引而言，对索引分区的互斥访问由数据分区到互斥或超级互斥访问方式的锁定升级确保。

解释锁定信息

SNAPLOCK 管理视图可以帮助您解释对某个分区表返回的锁定信息。以下是脱机索引重组期间捕获的 SNAPLOCK 管理视图。

```
SELECT SUBSTR(TABNAME, 1, 15) TABNAME, TAB_FILE_ID, SUBSTR(TBSP_NAME, 1, 15) TBSP_NAME,
       DATA_PARTITION_ID, LOCK_OBJECT_TYPE, LOCK_MODE, LOCK_ESCALATION L_ESCALATION
  FROM SYSIBMADM.SNAPLOCK
  WHERE TABNAME like 'TP1' and LOCK_OBJECT_TYPE like 'TABLE_%'
  ORDER BY TABNAME, DATA_PARTITION_ID, LOCK_OBJECT_TYPE, TAB_FILE_ID, LOCK_MODE


TABNAME TAB_FILE_ID TBSP_NAME  DATA_PARTITION_ID LOCK_OBJECT_TYPE LOCK_MODE L_ESCALATION
------- ----------- ---------- ----------------- ---------------- --------- ------------
TP1           32768 -                         -1 TABLE_LOCK       Z                   0
TP1               4 USERSPACE1                   0 TABLE_PART_LOCK  Z                   0
TP1               5 USERSPACE1                   1 TABLE_PART_LOCK  Z                   0
TP1               6 USERSPACE1                   2 TABLE_PART_LOCK  Z                   0
TP1               7 USERSPACE1                   3 TABLE_PART_LOCK  Z                   0
TP1               8 USERSPACE1                   4 TABLE_PART_LOCK  Z                   0
TP1               9 USERSPACE1                   5 TABLE_PART_LOCK  Z                   0
TP1              10 USERSPACE1                   6 TABLE_PART_LOCK  Z                   0
TP1              11 USERSPACE1                   7 TABLE_PART_LOCK  Z                   0
TP1              12 USERSPACE1                   8 TABLE_PART_LOCK  Z                   0
TP1              13 USERSPACE1                   9 TABLE_PART_LOCK  Z                   0
TP1              14 USERSPACE1                  10 TABLE_PART_LOCK  Z                   0
TP1              15 USERSPACE1                  11 TABLE_PART_LOCK  Z                   0
TP1               4 USERSPACE1                   - TABLE_LOCK       Z                   0
TP1               5 USERSPACE1                   - TABLE_LOCK       Z                   0
TP1               6 USERSPACE1                   - TABLE_LOCK       Z                   0
TP1               7 USERSPACE1                   - TABLE_LOCK       Z                   0
TP1               8 USERSPACE1                   - TABLE_LOCK       Z                   0
TP1               9 USERSPACE1                   - TABLE_LOCK       Z                   0
TP1              10 USERSPACE1                   - TABLE_LOCK       Z                   0
TP1              11 USERSPACE1                   - TABLE_LOCK       Z                   0
TP1              12 USERSPACE1                   - TABLE_LOCK       Z                   0
TP1              13 USERSPACE1                   - TABLE_LOCK       Z                   0
TP1              14 USERSPACE1                   - TABLE_LOCK       Z                   0
TP1              15 USERSPACE1                   - TABLE_LOCK       Z                   0
TP1              16 USERSPACE1                   - TABLE_LOCK       Z                   0

  已选择 26 个记录。
```

在此示例中，类型为 TABLE_LOCK 且 DATA_PARTITION_ID 为 -1 的锁定对象用于控制对分区表 TP1 的访问权以及并行性。类型为 TABLE_PART_LOCK 的锁定对象用于控制每个数据分区的大多数访问权和并行性。

此输出还捕获了其他类型为 TABLE_LOCK 的锁定对象（TAB_FILE_ID 4 到 16），这些锁定对象没有 DATA_PARTITION_ID 值。如果某种类型的锁定中对象的 TAB_FILE_ID 和 TBSP_NAME 与分区表中的数据分区或索引相对应，那么可以使用这种类型的锁定来控制脱机备份实用程序的并行性。

第 24 章 代理程序管理

分区数据库中的代理程序

在分区数据库环境或者启用了分区内并行性的环境中，每个数据库分区都有其自己的代理程序池，可以从中抽取子代理程序。

由于存在这个池，因此不必在每次需要子代理程序时将其创建或者在它完成工作时将其破坏。这些子代理程序仍然可以作为此池中的相关联代理程序，并可以由数据库管理器使用，以执行来自与它们相关联的应用程序或者新应用程序的新请求。

对系统性能和内存耗用量的影响与代理程序池的设置有很大关系。有关代理程序池大小的数据库管理器配置参数 (**num_poolagents**) 将影响可以与一个数据库分区中的应用程序保持关联的代理程序和子代理程序的总数。如果池太小并且该池已满，那么子代理程序将解除自己与它所处理应用程序的关联并终止。由于必须经常创建子代理程序并重新使它们与应用程序相关联，因此性能将下降。

缺省情况下，**num_poolagents** 设为 AUTOMATIC 并且值为 100，数据库管理器将自动管理池中的空闲代理程序数。

如果以手动方式将 **num_poolagents** 的值设置得过小，那么一个应用程序的相关联子代理程序就可能填满整个池。因此，当另一个应用程序需要新的子代理程序，并且在其代理程序池中没有子代理程序时，它将从其他应用程序的代理程序池中重新启动不活动的子代理程序。这种行为将确保充分利用资源。

如果以手动方式将 **num_poolagents** 的值设置得过大，那么相关联的子代理程序可能会长时间停留在池中未被使用并耗用数据库管理器资源，导致那些资源不可用于其他任务。

当连接集中器处于启用状态时，**num_poolagents** 的值不一定反映任何时候在池中可能处于空闲状态的代理程序的准确数目。可能会临时需要代理程序以处理工作负载更高的活动。

除数据库代理程序以外，其他异步数据库管理器活动也作为独立的进程或线程运行，其中包括:

- 数据库 I/O 服务器或 I/O 预取程序
- 数据库异步页清除程序
- 数据库记录器
- 数据库死锁检测器
- 通信和 IPC 侦听器
- 表空间容器重平衡程序

第 25 章 优化存取方案

索引访问和集群比率

优化器在选择存取方案时，它估算将所需的页从磁盘读入缓冲池所需执行的 I/O 次数。此估算包括预测缓冲池的使用情况，这是因为从已包含在缓冲池中的页读取行时，无需另外执行 I/O。

对于索引扫描而言，系统目录中的信息可以帮助优化器估算将数据页读入缓冲池的 I/O 成本。它将使用 SYSCAT.INDEXES 视图的以下各列中的信息:

- CLUSTERRATIO 信息指示表数据相对于此索引的集群度。此数值越大，各行按索引键顺序排列的情况越好。如果表行顺序接近于索引键顺序，那么当数据页包含在缓冲区中时，可以从该页读取行。如果此列的值为 -1，并且可以获得 PAGE_FETCH_PAIRS 和 CLUSTERFACTOR 信息，那么优化器将使用该信息。
- PAGE_FETCH_PAIRS 列包含多对数值以及 CLUSTERFACTOR 信息，这些数值模拟将数据页读入各种大小的缓冲池时所需执行的 I/O 次数。只有在指定 DETAILED 子句的情况下对索引调用 **RUNSTATS** 命令时，才会为这些列收集数据。

如果没有可用的索引集群统计信息，那么优化器将使用缺省值，即假定数据相对于索引而言集群情况不佳。数据的集群度将显著影响性能，您应该尝试使其中一个对表定义的索引的集群度接近 100%。通常，只有一个索引可达到 100% 集群度，但下列情况除外: 索引的键是集群索引键的超集，或者两个索引的键列之间存在实际的关联。

重组表时，您可以指定一个索引，此索引将用于对行进行集群并在插入处理期间保持这些行处于集群状态。由于更新和插入操作会降低表相对于索引而言的集群度，因此您可能需要定期重组表。为了减少频繁执行插入、更新或删除操作的表的重组次数，请在 ALTER TABLE 语句中指定 PCTFREE 子句。

MDC 和 ITC 表的表和索引管理

多维集群 (MDC) 和插入时间集群 (ITC) 表的表和索引组织与标准表组织基于相同逻辑结构。

与标准表类似，MDC 和 ITC 表按页组织，这些页包含分为许多列的数据行。每一页中的行由记录标识 (RID) 标识。但是，MDC 和 ITC 表的各个页还分组为具有扩展数据块大小的块。例如，第 302 页的图 52 显示了扩展数据块大小为 4 的表。前四页（编号为 0 到 3）是表中的第一个块。接着的四页（编号为 4 到 7）是表中的第二个块。

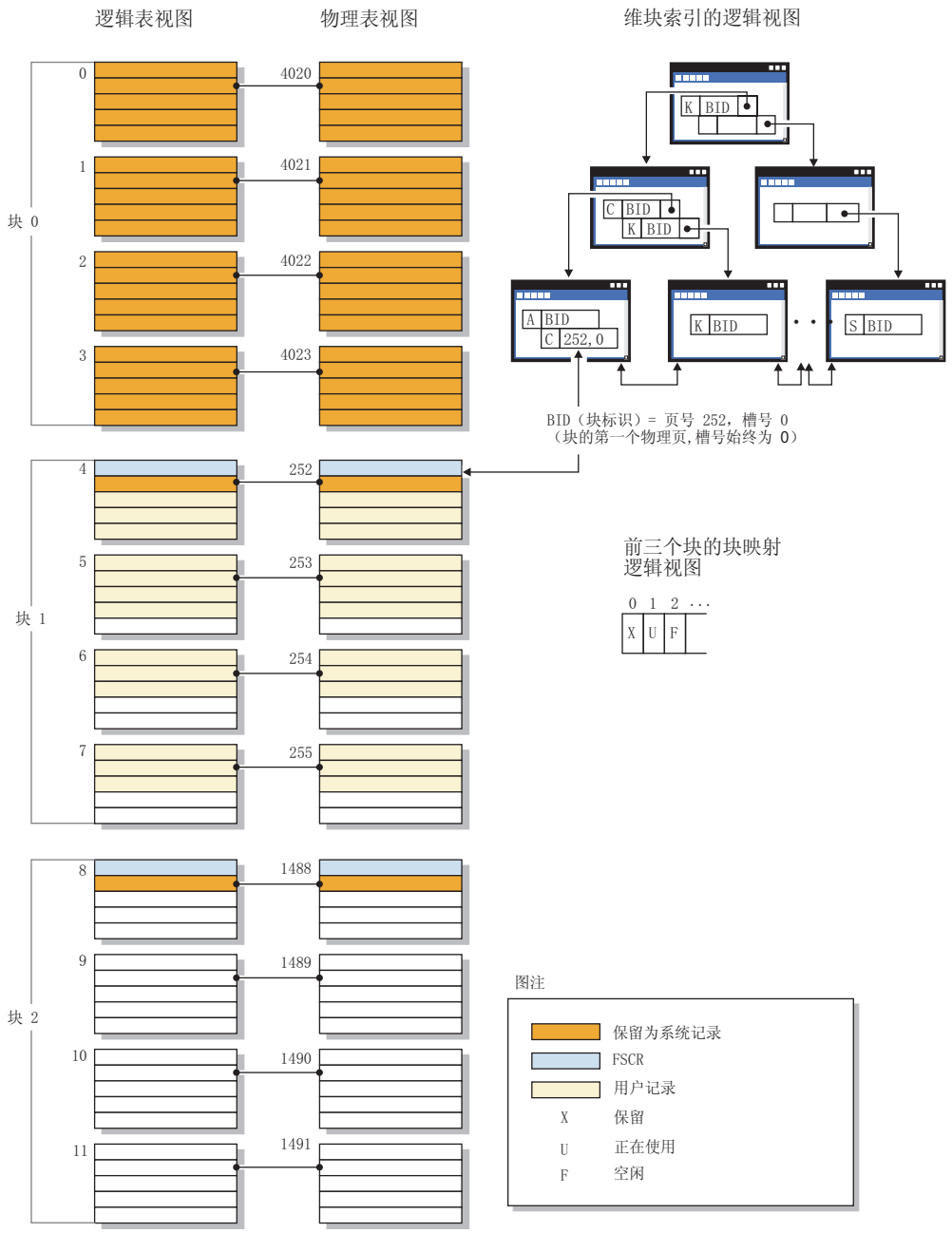


图 52. MDC 表和 ITC 表的逻辑表、记录和索引结构

第一个块包含 DB2 服务器用来管理表的特殊内部记录，其中包括可用空间控制记录（FSCR）。在后续块中，第一页包含 FSCR。FSCR 为新记录映射块中每一页中存在的可用空间。将记录插入到表中时，将使用这部分可用空间。

顾名思义，MDC 表对多个维的数据进行集群。每个维都由您在 CREATE TABLE 语句的 ORGANIZE BY DIMENSIONS 子句中指定一列或一组列确定。创建 MDC 表时，将自动创建下面这两个索引:

- 维块索引，它包含指向单个维的每个被占用块的指针
- 组合块索引，它包含所有维键列并用于在插入和更新活动期间维护集群

当优化器确定特定查询的最高效存取方案时，它将考虑使用维块索引的存取方案。当查询包含应用于维值的谓词时，优化器可以使用维块索引来标识包含这些值的扩展数据块以及访存那些扩展数据块的内容。由于扩展数据块在物理上是磁盘中相邻的页，因此这将最大程度地减少 I/O 操作，从而提高性能。

如果对数据存取方案进行的分析表明特定 RID 索引有助于提高查询性能，那么您可以创建这样的索引。

顾名思义，ITC 表根据行插入时间对数据进行集群。MDC 表与 ITC 表之间的差别在于:

- 不会对任何数据访问使用块索引
- 仅对表创建单个组合块索引，并且该索引由虚拟维组成，并且
- 优化器未对方案选择任何索引，因为任何 SQL 语句都不能引用它包含的列。

MDC 和 ITC 表可将其空块释放到表空间。

分区内并行性的优化策略

如果编译 SQL 语句时指定了并行度，那么优化器可以选择存取方案，以便在单一数据库分区中以并行方式执行查询。

在运行时，将创建多个称为“子代理程序”的数据库代理程序来执行该查询。子代理程序的数目将小于或等于编译该 SQL 语句时指定的并行度。

为了将存取方案并行化，优化器将它划分为两个部分，每个子代理程序运行一部分，协调代理程序运行另一部分。子代理程序通过表队列将数据传递至协调代理程序或其他子代理程序。在分区数据库环境中，子代理程序与其他数据库分区中的子代理程序之间能够通过表队列来发送或接收数据。

分区内并行扫描策略

可以采用并行方式对同一个表或索引执行关系扫描和索引扫描。要进行并行关系扫描，需将表划分为由页或行组成的范围，然后将范围分配给子代理程序。子代理程序将扫描分配给它的范围，处理当前范围完毕后，它将被分配另一个范围。

要进行并行索引扫描，需根据索引键值以及键值的索引条目数将索引划分为多个记录范围。并行索引扫描的执行方式类似于并行表扫描，即，将某个范围内的记录分配给子代理程序。子代理程序处理当前范围完毕后，将被分配新的范围。

可对范围分区表运行并行表扫描，并且同样，可对分区索引运行并行索引扫描。对于并行扫描，分区索引分为记录范围（根据索引键值和键值的键条目数）。并行扫描开始时，会对子代理程序指定一定范围的记录，子代理程序完成一个范围时，会对它指定一个新范围。索引分区可能在任意时间点使用正在扫描未保留索引分区的子代理程序进行顺序扫描，而不必等待其他子代理程序完成扫描。只会扫描与基于数据分区消除分析的查询相关的索引分区部分。

优化器确定扫描单位（页或行）和扫描粒度。

并行扫描在各个子代理程序之间均匀地分布工作。并行扫描的目标是，平衡所有子代理程序的负载并使它们保持相同的繁忙程度。如果繁忙子代理程序数等于可用处理器数，且磁盘未过度处理 I/O 请求，那么表明机器资源得到高效利用。

执行查询时，其他存取方案策略可能会导致数据不平衡。优化器选择并行策略，以便在子代理程序之间维持数据平衡。

分区内并行排序策略

优化器可以选择下列其中一种并行排序策略:

- 循环排序

  这也称为*再分布排序*。这种方法使用共享内存，高效地将数据尽可能均匀地重新分布到所有子代理程序。它使用轮询算法来确保分布均匀。首先，为每个子代理程序创建单个排序。在插入阶段，子代理程序以循环方式对每个排序执行插入，以使数据分布更加均匀。

- 分区排序

  这类似于循环排序，即，为每个子代理程序创建一个排序。子代理程序将一个散列函数应用于排序列，以便确定应该将行插入到哪个排序。例如，如果合并连接的内表和外表是分区排序，那么子代理程序可使用合并连接来连接相应的表部分并以并行方式执行。

- 复制排序

  如果每个子代理程序都需要所有排序输出，那么使用这种排序。将创建一个排序，并且在将行插入到排序时使各个子代理程序同步。排序完成后，每个子代理程序都读取整个排序。如果行数较小，那么可使用此排序对数据流进行重新平衡。

- 共享排序

  此排序与复制排序相同，只是子代理程序要对已排序的结果打开一个并行扫描，以便采用一种与循环排序相似的方式在子代理程序之间分布数据。

分区内并行临时表

子代理程序可以协同工作，通过将行插入到同一个表来生成临时表。这称为*共享临时表*。根据是要复制还是要分割数据流，子代理程序可以对共享临时表进行专用扫描或并行扫描。

分区内并行聚集策略

子代理程序可以采用并行方式来执行聚集操作。聚集操作要求根据分组列对数据进行排序。如果可以保证一个子代理程序接收一组分组列值的所有行，那么该程序可以执行完整的聚集。发生这种情况的条件是，先前已执行分区排序，致使已根据分组列对数据流进行分割。

否则，子代理程序可以执行部分聚集，并使用另一种策略来完成该聚集。某些策略如下所示:

- 通过合并表队列将部分聚集的数据发送至协调代理程序。协调代理程序完成聚集。
- 将部分聚集的数据插入到分区排序。该排序根据分组列进行分割，并保证一组分组列的所有行都包含在一个排序分区中。
- 如果需要复制数据流以便平衡处理，那么可将部分聚集的数据插入到复制排序。每个子代理程序都使用该复制排序来完成聚集，并接收完全相同的聚集结果副本。

分区内并行连接策略

子代理程序可以采用并行方式来执行连接操作。并行连接策略由数据流的特征确定。

通过对连接的内表和/或外表进行分区或者复制数据流，可以将连接并行化。例如，如果因为并行扫描已将嵌套循环连接的外流分区，而且内流由每个子代理程序独立重新求值，那么可以将该连接并行化。如果合并连接的内流和外流由于分区排序而按值分区，那么可将该连接并行化。

执行查询时，数据过滤和数据偏差可能导致子代理程序之间的工作负载不平衡。不平衡的工作负载的无效率会因为连接和其他计算成本很高的操作放大。优化器会查找查询存取方案中导致不平衡的来源并应用均衡策略，以确保在子代理程序间平均划分工作。对于无序的外部数据流，优化器会对外部数据流使用 REBAL 运算符来平衡连接。对于有序列数据流（其中有序数据由索引访问或排序生成），优化器会使用共享排序平衡数据。如果排序溢出到临时表中，那么将不会使用共享排序，因为排序溢出的成本很高。

连接

*连接*是指根据信息的某些公共领域对来自两个或更多个表的数据进行组合的过程。如果连接条件（*连接谓词*）确定对应的行中的信息匹配，那么一个表中的行就会与另一个表中的行配对。

例如，考虑下面这两个表:

TABLE1

PROJ	PROJ_ID
A	1
B	2
C	3
D	4

TABLE2

PROJ_ID	NAME
1	Sam
3	Joe
4	Mary
1	Sue
2	Mike

要将 TABLE1 与 TABLE2 连接，以使 PROJ_ID 列包含相同的值，请使用以下 SQL 语句:

```
select proj, x.proj_id, name
  from table1 x, table2 y
  where x.proj_id = y.proj_id
```

在这种情况下，适当的连接谓词是: `where x.proj_id = y.proj_id`。

此查询将生成以下结果集:

PROJ	PROJ_ID	NAME
A	1	Sam
A	1	Sue
B	2	Mike
C	3	Joe
D	4	Mary

根据任何连接谓词的性质以及通过表和索引统计信息确定的任何成本，优化器将选择下列其中一种连接方法:

- 嵌套循环连接
- 合并连接
- 散列连接

连接两个表时，将选择其中一个表作为外表，而将另一个表视为该连接的内表。首先访问外表，并且只对其执行一次扫描。是否对内表执行多次扫描取决于该连接的类型以及可用的索引。即使一个查询连接两个以上的表，优化器每次也只连接两个表。必要时，将创建临时表来保存中间结果。

您可以提供显式的连接运算符（例如 INNER 或 LEFT OUTER JOIN），以确定如何在连接中使用表。但是，以这种方式更改查询之前，应允许优化器确定如何连接表，然后分析查询性能以确定是否添加连接运算符。

数据库分区组对查询优化的影响

在分区数据库环境中，优化器在确定查询的最佳存取方案时，它能够识别并使用表的并置。

如果在连接查询中频繁涉及某些表，那么应该将那些表划分到各个数据库分区，以使每个所连接的表中的行位于同一个数据库分区中。在执行连接操作期间，所连接的两个表中数据的并置能够避免将数据从一个数据库分区移至另一个数据库分区。请将这两个表置于同一个数据库分区组中，以确保对这两个表中的数据进行并置。

根据表的大小，将数据分布到多个数据库分区有助于缩短执行查询所需的估计时间。表的数目、表的大小、那些表中数据的位置以及查询类型（例如，是否需要连接）都会影响查询的成本。

分区数据库的连接策略

分区数据库环境的连接策略可以与未分区数据库环境的策略不同。您可以将其他技术应用于标准连接方法以提高性能。

对于频繁连接的表，应该考虑进行表并置。在分区数据库环境中，*表并置*是指两个表将相同数目的兼容分区键存储到同一个数据库分区组时出现的状态。发生这种情况时，可以在存储数据的数据库分区中执行连接处理，并且只需要将结果集移至协调程序数据库分区。

表队列

分区数据库环境中连接技术的描述使用下列术语:

- *表队列*（有时称为 TQ）是一种在数据库分区之间或单一分区数据库中的处理器之间传送行的机制。
- *定向型表队列*（有时称为 DTQ）对行进行散列，以便将其发送到其中一个接收数据库分区。
- *广播表队列*（有时称为 BTQ）将行发送到所有接收数据库分区，而不进行散列。

表队列用于传递表数据:

- 使用分区间并行性时，将数据从一个数据库分区传递到另一个数据库分区
- 使用分区内并行性时，在数据库分区内传递数据
- 使用单一分区数据库时，在数据库分区中传递数据

每个表队列都按单一方向传递数据。编译器决定何处需要使用表队列并将其包括在方案中。执行该方案时，数据库分区之间的连接将启动表队列。表队列在进程结束时关闭。

表队列分为多种类型:

- 异步表队列

  这些表队列被称为异步队列的原因是，它们在应用程序发出任何访存请求之前读取行。发出 FETCH 语句时，将从表队列中检索该行。

  如果您在 SELECT 语句中指定了 FOR FETCH ONLY 子句，那么将使用异步表队列。如果您只访存行，那么异步表队列的速度较快。

- 同步表队列

  这些表队列被称为同步队列的原因是，对于应用程序发出的每个 FETCH 语句，这些队列读取一行。在每个数据库分区中，游标都定位在要从该数据库分区读取的下一行上。

  如果您未在 SELECT 语句中指定 FOR FETCH ONLY 子句，那么将使用同步表队列。在分区数据库环境中，如果您要更新行，那么数据库管理器将使用同步表队列。

- 合并表队列

  这些表队列维护顺序。

- 非合并表队列

  这些表队列也被称为*常规表队列*，它们不维护顺序。

- 侦听器表队列（有时称为 LTQ）

  这些表队列与相关的子查询配合使用。使用这种类型的表队列时，将关联值向下传递至子查询，然后将结果向上传递回给父查询块。

用于分区数据库的连接方法

有多种连接方法可用于分区数据库环境，其中包括：并置连接、广播外表连接、定向外表连接、定向内表和外表连接、广播内表连接以及定向内表连接。

在下面的图中，q1、q2、和 q3 表示表队列。所引用的表包含在两个数据库分区中，箭头指示表队列的发送方向。协调程序数据库分区是数据库分区 0。

如果编译器选择的连接方法是散列连接，那么在每个远程数据库分区上创建的过滤器可用于在元组发送至处理该散列连接的数据库分区之前消除这些元组，从而改进分区。

并置连接

并置连接以本地方式在数据所在的数据库分区中发生。该数据库分区在完成连接以后将数据发送到其他数据库分区。要使优化器考虑并置连接，必须对所连接的表进行并置，并且所有各对相应分布键都必须参与等式连接谓词。第 308 页的图 53 提供了一个示例。

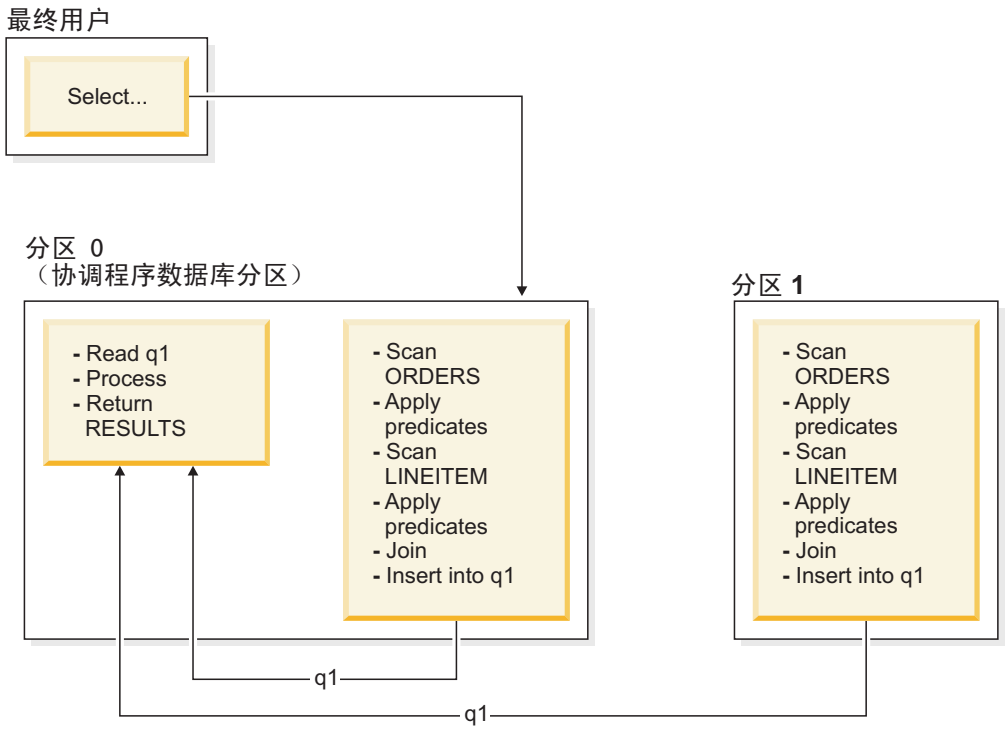


图 53. 并置连接示例

LINEITEM 和 ORDERS 表都根据 ORDERKEY 列进行分区。此连接以本地方式在每个数据库分区中执行。在此示例中，假定连接谓词为：`orders.orderkey = lineitem.orderkey`。

复制型具体化查询表 (MQT) 能够提高并置连接的可能性。

广播外表连接

广播外表连接代表一种并行连接策略，如果所连接的表之间没有等式连接谓词，那么可以使用此连接。此连接也可用于其他证实此连接方法最合乎成本效益的情况。例如，当有一个很大的表和一个很小的表，并且未根据连接谓词列对任何一个表进行分割时，可能会发生广播外表连接。低成本方法是将较小的表广播至较大的表，而不是分割这两个表。第 309 页的图 54 提供了一个示例。

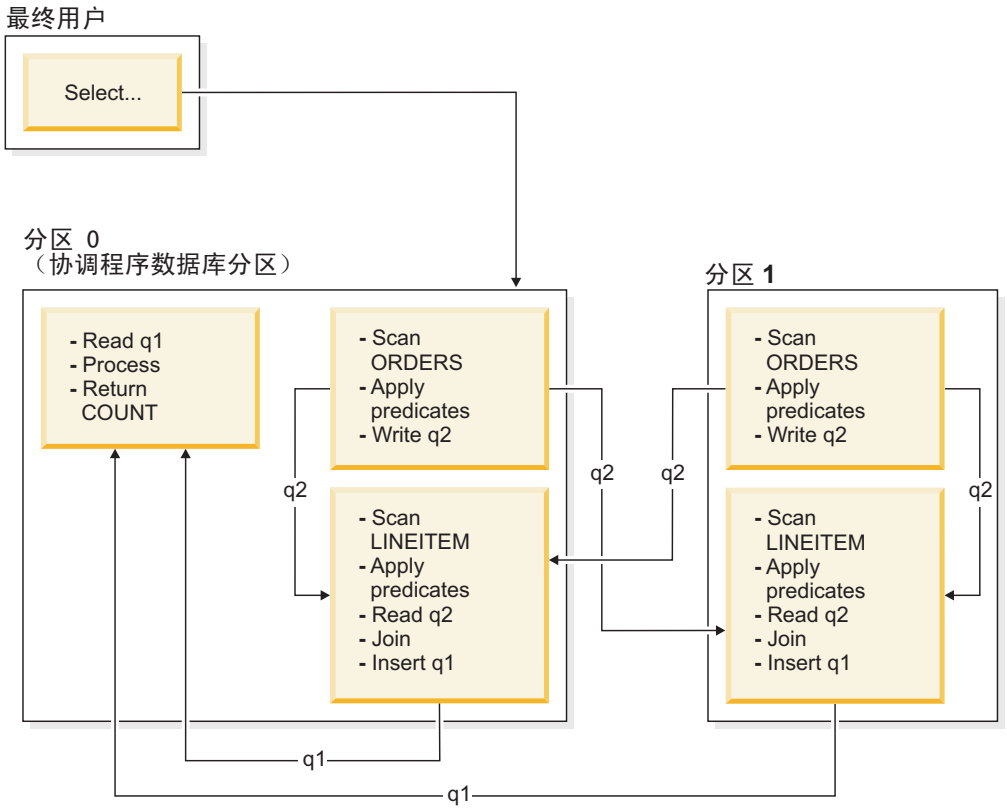
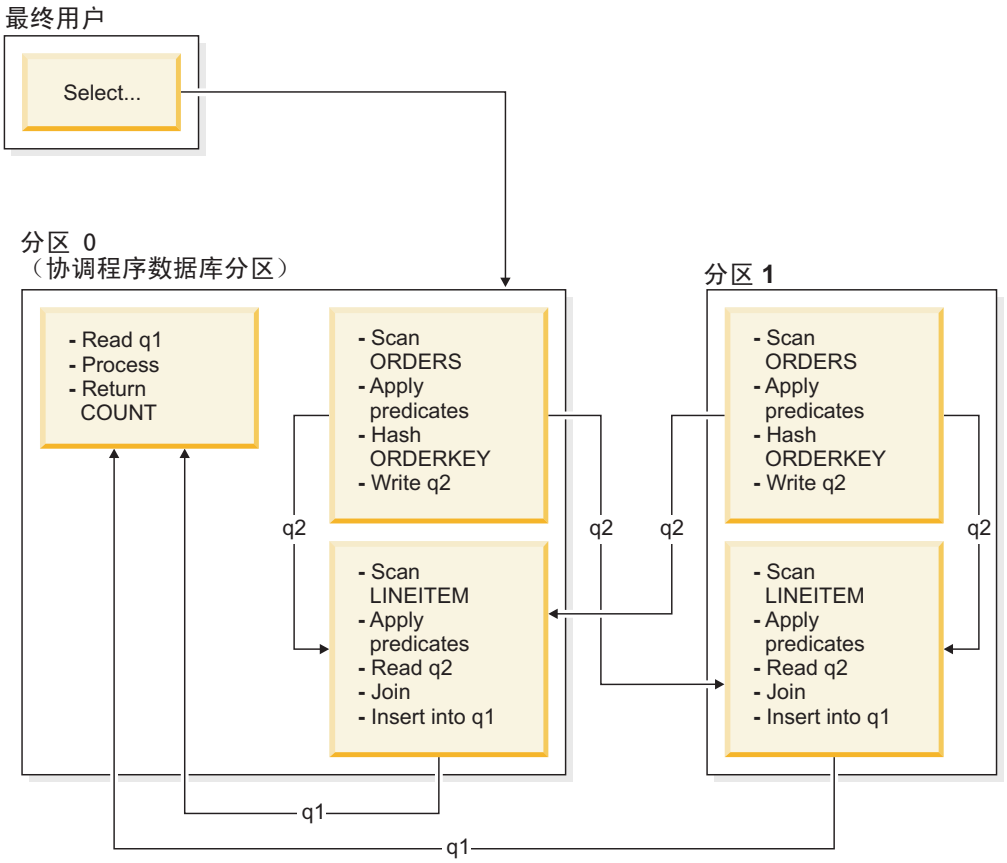


图 54. 广播外表连接示例

ORDERS 表被发送到所有包含 LINEITEM 表的数据库分区。表队列 q2 被广播至内表的所有数据库分区。

定向外表连接

在定向外表连接策略中，根据内表的分割属性将外表的每一行发送至内表的一个部分。此连接在此数据库分区中进行。第 310 页的图 55 提供了一个示例。



LINEITEM 表根据 ORDERKEY 列进行分区。ORDERS 表根据另一个列进行分区。ORDERS 表将进行散列并被发送到 LINEITEM 表的正确数据库分区。在此示例中，假定连接谓词为: `orders.orderkey = lineitem.orderkey`。

图 55. 定向外表连接示例

定向内表和外表连接

在定向内表和外表连接策略中，根据连接列的值，将外表和内表的行定向到一组数据库分区。此连接在这些数据库分区中进行。第 311 页的图 56 提供了一个示例。

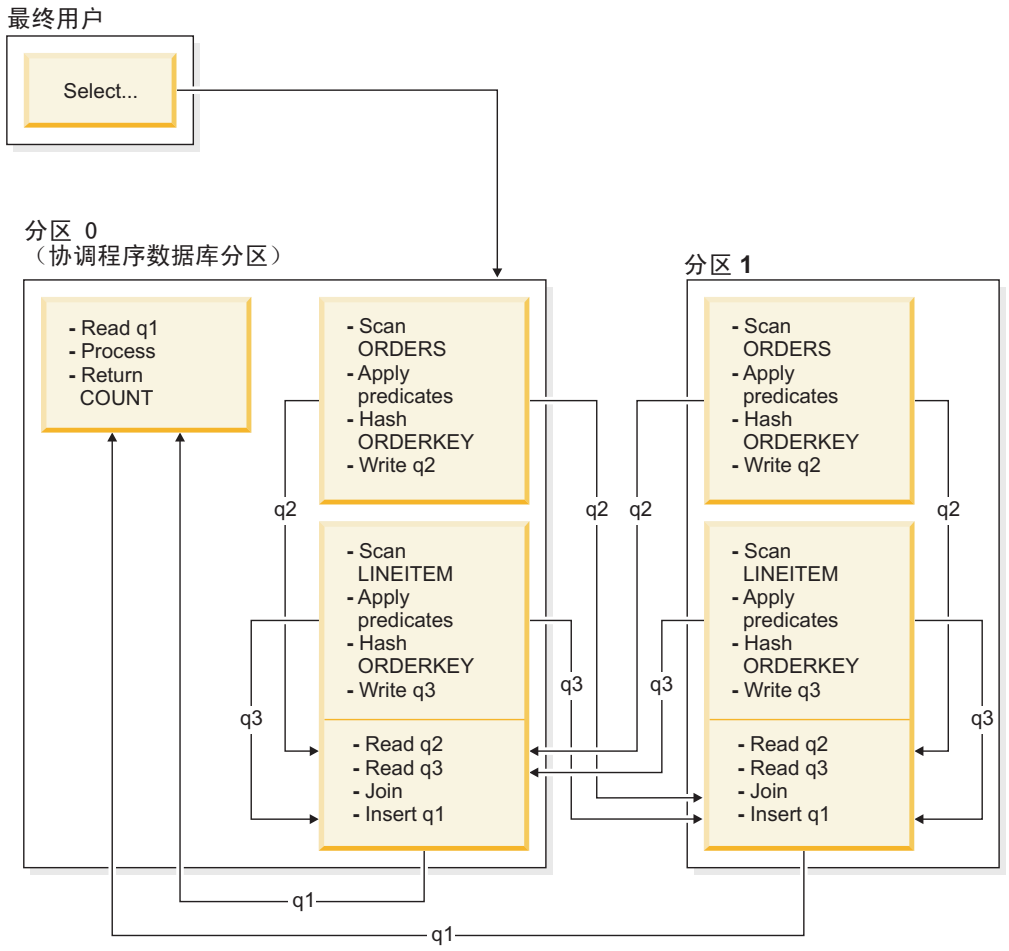
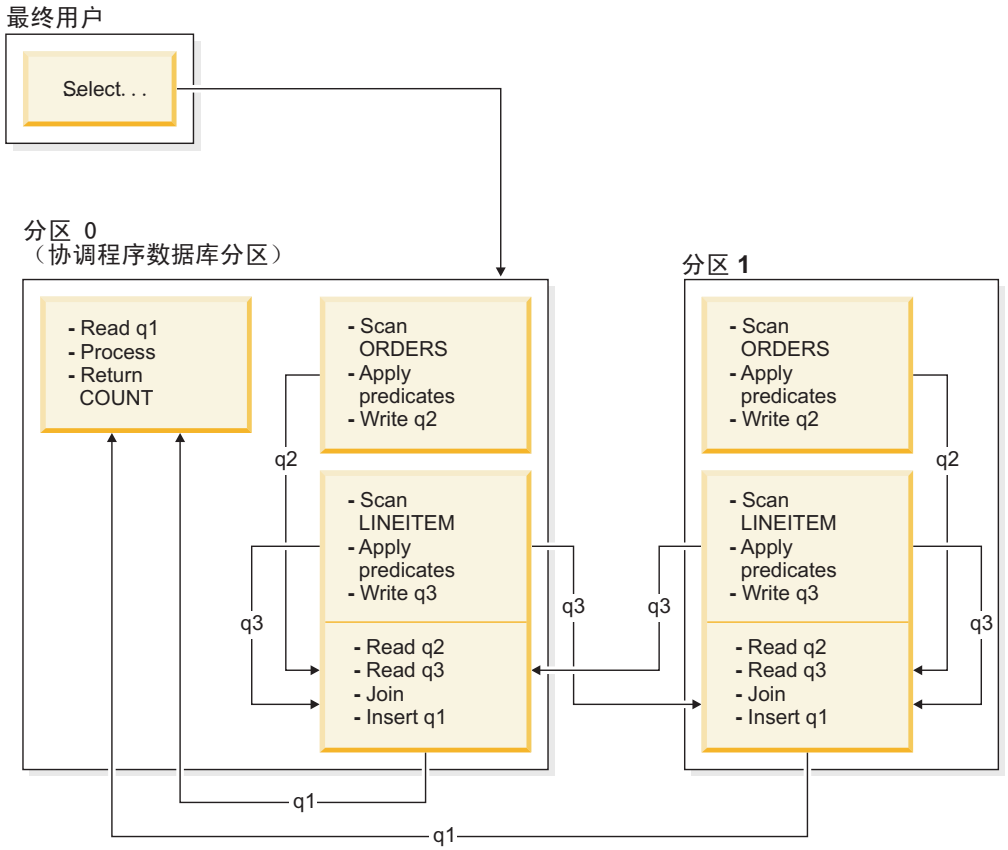


图 56. 定向内表和外表连接示例

两个表均未根据 ORDERKEY 列进行分区。这两个表都将进行散列并被发送到新的数据库分区，它们将在那些数据库分区中进行连接。表队列 q2 和 q3 都将被定向。在此示例中，假定连接谓词为：`orders.orderkey = lineitem.orderkey`。

广播内表连接

在广播内表连接策略中，将内表广播至外表的所有数据库分区。第 312 页的图 57 提供了一个示例。

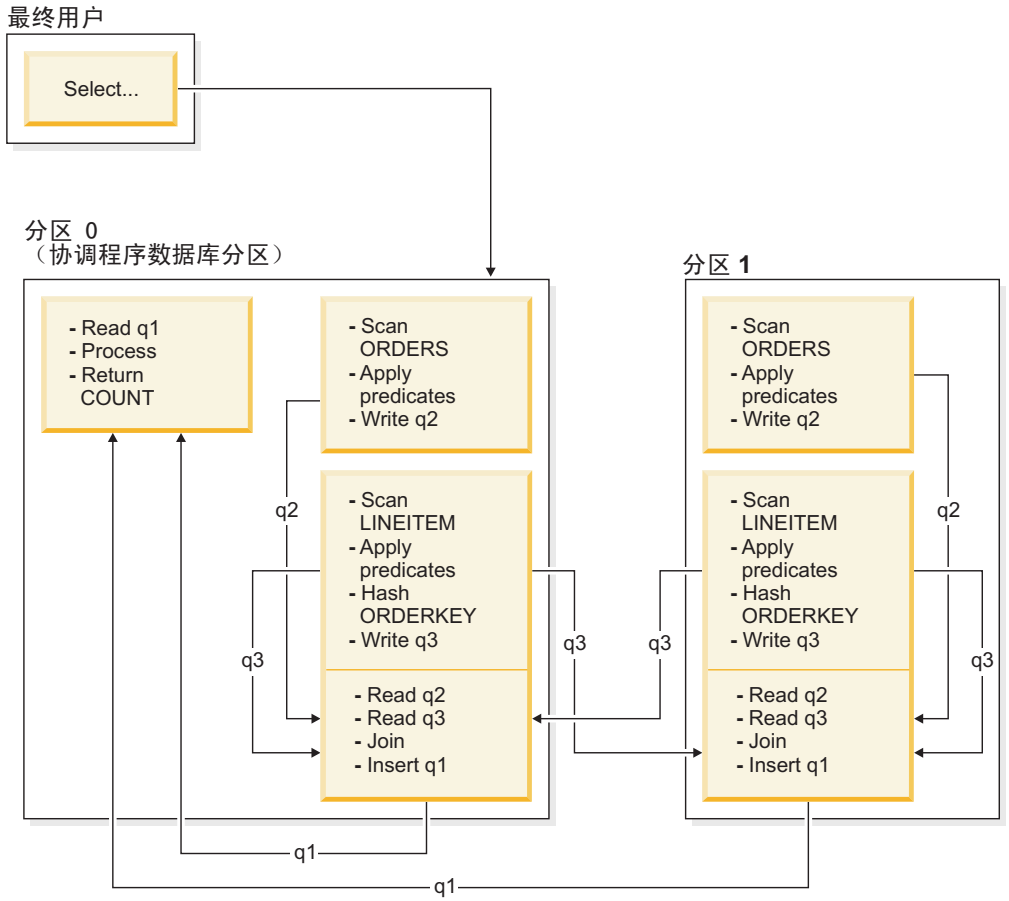


LINEITEM 表被发送到所有包含 ORDERS 表的数据库分区。表队列 q3 被广播至外表的所有数据库分区。

图 57. 广播内表连接示例

定向内表连接

在定向内表连接策略中，根据外表的分割属性，将内表的每一行发送至外表的一个数据库分区。此连接在此数据库分区中进行。第 313 页的图 58 提供了一个示例。



ORDERS 表根据 ORDERKEY 列进行分区。LINEITEM 表根据另一个列进行分区。LINEITEM 表将进行散列并被发送到 ORDERS 表的正确数据库分区。在此示例中，假定连接谓词为: `orders.orderkey = lineitem.orderkey`。

图 58. 定向内表连接示例

分区数据库环境中的复制型具体化查询表

复制型具体化查询表 (MQT) 允许数据库管理预先计算的表数据值，从而提高分区数据库环境中频繁执行的连接的性能。

注意，在此上下文中，复制型 MQT 与数据库内复制相关。数据库间复制与预订、控制表以及不同数据库和不同操作系统中的数据相关。

在以下示例中:

- SALES 表在多分区表空间 REGIONTABLESPACE 中，并且根据 REGION 列进行分割。
- EMPLOYEE 和 DEPARTMENT 表在单一分区数据库分区组中。

根据 EMPLOYEE 表中的信息创建复制型 MQT。

```
create table r_employee as (
  select empno, firstnme, midinit, lastname, workdept
    from employee
```

```
)
data initially deferred refresh immediate
in regiontablespace
复制型
```

更新复制型 MQT 的内容:

```
refresh table r_employee
```

使用 REFRESH 语句之后，应该对复制的表调用 RUNSTATS 实用程序，就像对任何其他表执行此调用一样。

下列查询计算职员的销售额、部门总销售额和总计:

```
select d.mgrno, e.empno, sum(s.sales)
  from department as d, employee as e, sales as s
  where
    s.sales_person = e.lastname and
    e.workdept = d.deptno
  group by rollup(d.mgrno, e.empno)
  order by d.mgrno, e.empno
```

数据库管理器并非使用仅驻留在一个数据库分区中的 EMPLOYEE 表，而是使用 R_EMPLOYEE，这是在每个存储 SALES 表的数据库分区中均进行复制的 MQT。这将提高性能，其原因在于，执行连接时不必通过网络将职员信息传送到每个数据库分区。

并置连接中的复制型具体化查询表

复制型 MQT 也有助于并置连接。例如，如果星型模式包含分布于 20 个数据库分区中的大型事实表，那么对事实表和维表进行并置后，事实表与维表之间的连接效率最高。如果所有表都在同一个数据库分区组中，那么对于一个并置连接，最多能够对一个维表进行正确分区。其他维表不能在并置连接中使用，这是因为事实表中的连接列与事实表的分布键不对应。

考虑到将名称为 FACT (C1, C2, C3, ...) 的表在 C1 处断开; 名称为 DIM1 (C1, dim1a, dim1b, ...) 的表在 C1 处断开; 名称为 DIM2 (C2, dim2a, dim2b, ...) 的表在 C2 处断开; 以此类推。在这种情况下，FACT 与 DIM1 之间的连接最好，这是因为将并置谓词 `dim1.c1 = fact.c1`。这两个表都根据列 C1 进行分割。

但是，由于 FACT 根据列 C1 进行分割，而不是根据列 C2 进行分割，因此不能对涉及 DIM2 和谓词 `dim2.c2 = fact.c2` 的连接进行并置。在这种情况下，可以在事实表的数据库分区组中复制 DIM2，以便在每个数据库分区中以局部方式进行连接。

在创建复制型 MQT 时，源表可以是数据库分区组中的单一分区表或多分区表。在大多数情况下，复制的表不大，并且可以放入单一分区数据库分区组。通过只指定表中的部分列或者使用谓词来限制合格行的数目，可以限制所要复制的数据。

也可以在多分区数据库分区组中创建复制型 MQT，以便在所有数据库分区中创建源表的副本。与采用广播方式将源表传送到所有数据库分区相比，在此环境中，大型事实表与维表之间的连接更有可能以局部方式进行。

不会自动创建基于所复制的表的索引。您可以创建与基于源表的索引不同的索引。但是，为了防止未出现在源表中的约束违例，不能对复制的表创建唯一索引或定义约束，即使源表中存在相同的约束亦如此。

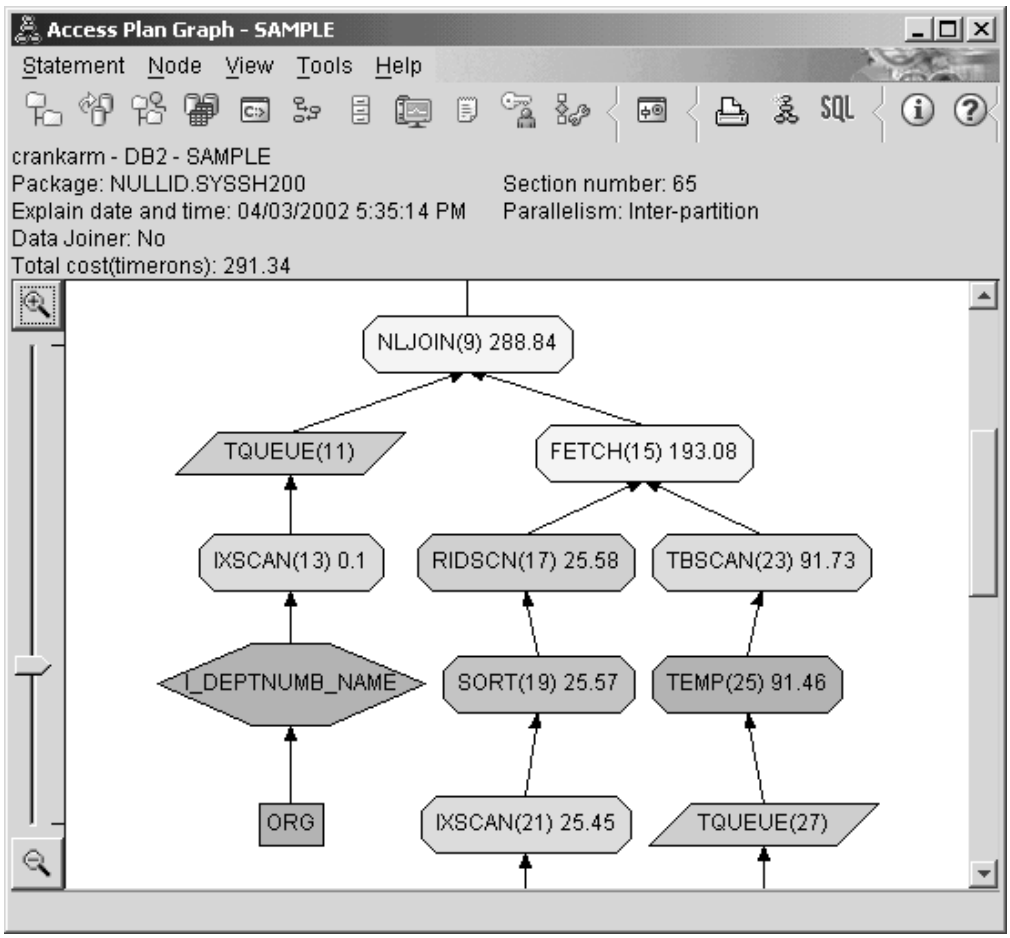
可以在查询中直接引用复制的表，但不可以通过将 DBPARTITIONNUM 标量函数用于复制的表以便查看特定数据库分区中的表数据。

请使用 DB2 Explain 工具来确定查询的存取方案是否已使用复制型 MQT。优化器选择的存取方案是否使用复制型 MQT 取决于要连接的数据。如果优化器确定以广播方式向数据库分区组中的其他数据库分区传送原始源表的成本更低，那么可能不会使用复制型 MQT。

在分区数据库环境中对表列创建其他索引

此示例是在查询 3 中描述的存取方案上构建的，方法是对 STAFF 表中的 JOB 列创建索引，并将 DEPTNAME 添加至 ORG 表中的现有索引。（添加另一索引可能导致其他访问。）

要查看此查询（查询 4）的存取方案图，请执行以下操作：在“说明语句历史记录”窗口中，双击标识为“查询号 4”的条目。将对此语句执行打开“存取方案图”窗口。

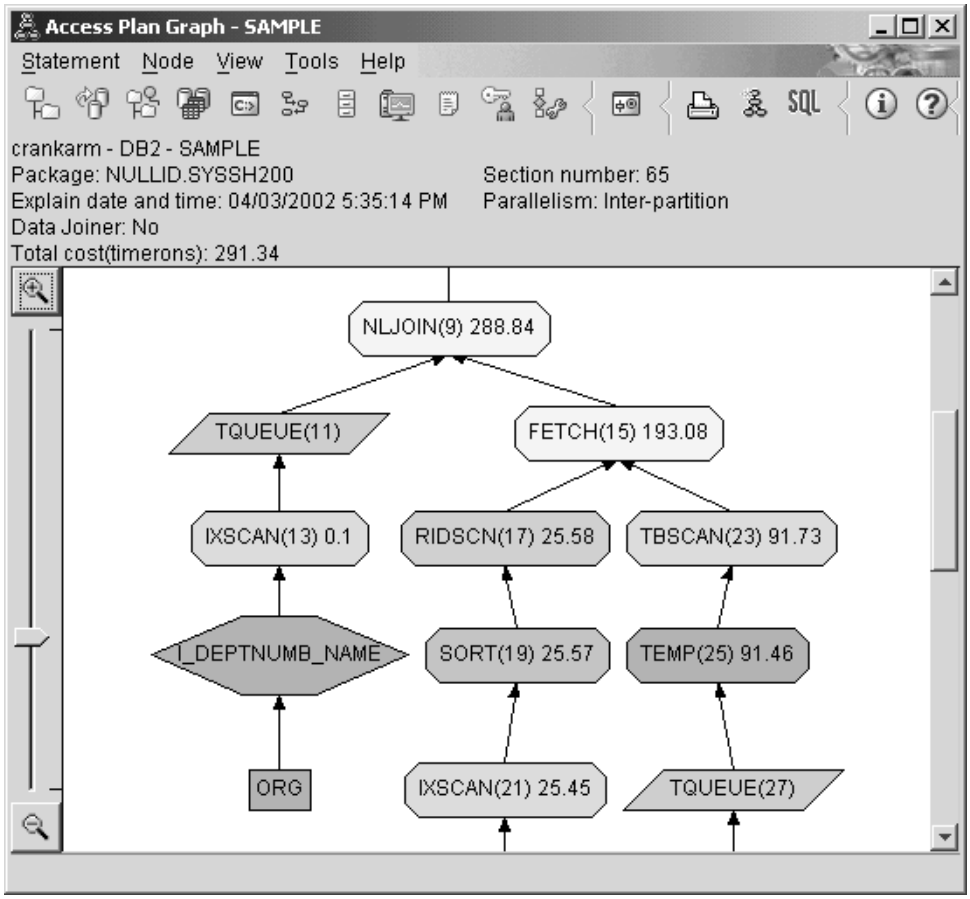


对下列问题的回答将有助于您了解如何改进查询。

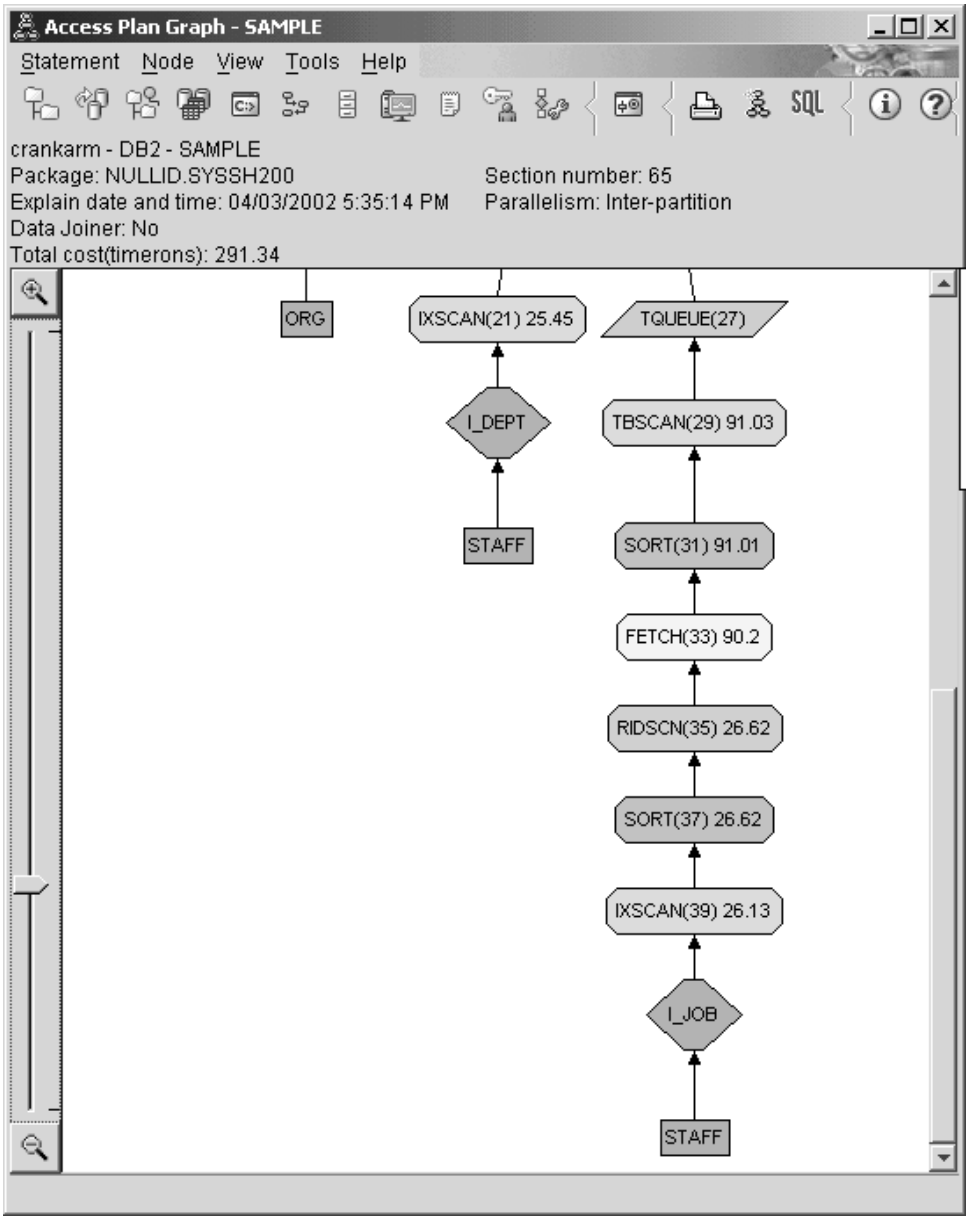
1. 此存取方案由于创建附加索引而更改了哪些内容？

   注意，在存取方案图的中间部分，对于 ORG 表，先前的表扫描已更改为索引扫描 IXSCAN (7)。将 DEPTNAME 列添加至 ORG 表的索引已使优化器可改进涉及表扫

描的访问。



注意，在存取方案图的底部，对于 STAFF 表，先前的索引扫描和访存已更改为仅索引扫描 IXSCAN (39)。对 STAFF 表创建 JOB 索引已使优化器不必为进行访存而做额外的访问工作。



2. 此存取方案的效果如何？

   此存取方案比先前示例中的存取方案的成本更低。累计成本已从“查询 3”中的大约 753 timeron 减少至“查询 4”中的大约 288 timeron。

下一步如何操作

提高您自己的 SQL 或 XQuery 语句的性能。

请参阅 *DB2 信息中心*以了解有关可用于提高性能的其他步骤的详细信息。然后可返回至 Visual Explain 以了解操作带来的影响。

第 26 章 数据重新分发

*数据重新分发*是一种数据库管理操作，可以在添加或除去分区后执行此操作，主要用于移动分区数据库环境中的数据。此操作的目的通常是平衡存储空间的使用、提高数据库系统性能或满足其他系统要求。

可以使用下列其中一个接口来执行数据重新分发:

- **REDISTRIBUTE DATABASE PARTITION GROUP** 命令
- ADMIN_CMD 内置过程
- STEPWISE_REDISTRIBUTE_DBPG 内置过程
- sqludrdt API

将基于下列其中一个理由来完成分区数据库中的数据重新分发:

- 每当向数据库环境添加新的数据库分区或除去现有数据库分区时，重新平衡数据。
- 要在分区之间引入特定于用户的数据分布。
- 通过在特定分区中隔离敏感数据来保护这些数据。

要执行数据重新分发，请连接至目录数据库分区中的数据库并使用某个受支持的接口开始对特定分区组执行数据重新分发操作。数据重新分发依赖于分区组中的表的分布键定义存在于否。该表中某行数据的分布键值用于确定该行数据将存储在哪个分区中。在多分区数据库分区组中创建表时，将自动生成分布键。还可以使用 CREATE TABLE 或 ALTER TABLE 语句显式定义分布键。缺省情况下，在数据重新分发期间，对于指定的数据库分区组中的每个表，将对表数据进行分割并将其在数据库分区之间均匀地重新分发。通过指定输入分发映射（用于定义数据的分发方式），可以实现其他分发，例如，非均匀分发。可在数据重新分发操作期间生成分发映射以供未来使用，也可以手动创建分发映射。

进行日志记录的可恢复重新分发与进行最少日志记录的不可前滚恢复重新分发的比较

使用 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令或内置过程 ADMIN_CMD 来执行数据重新分发时，可以在两种数据重新分发方法之间进行选择：进行日志记录的可恢复重新分发和进行最少日志记录的不可前滚恢复重新分发。通过使用 **NOT ROLLFORWARD RECOVERABLE** 命令参数来指定第二种方法。

如果在负载均衡或性能调整期间容量增加，那么这时进行数据重新分发可能需要昂贵的维护窗口时间、相当长的规划时间以及日志空间和额外的容器空间，开销将会很大。对重新分发方法的选择取决于您注重可恢复性还是速度:

- 使用进行日志记录的可恢复重新分发方法时，将对所有行移动执行全面的日志记录，以便在发生任何中断、错误或有其他业务需要时可以恢复数据库。
- 不可前滚恢复重新分发方法提供了更高的性能，这是因为将成批移动数据，并且插入和删除操作不再需要日志记录。

在过去，如果活动日志空间和存储器要求较大而强制您将单个数据重新分发操作拆分为多个较小的重新分发任务（这可能导致完成端到端数据重新分发操作需要甚至更多的时间），那么后一种方法特别有用。

不可前滚恢复重新分发方法是大多数情况下的最佳实践，这是因为该数据重新分发需要的时间更少、更不容易出错并且使用的系统资源更少。因此，执行数据重新分发的总成本将减少，这意味着可以节省时间和资源以用于其他业务操作。

进行最少日志记录的不可前滚恢复重新分发

发出 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令并指定 **NOT ROLLFORWARD RECOVERABLE** 参数时，将使用最少日志记录策略，该策略会使写入每个移动行的日志记录最少。此类型的日志记录对于重新分发操作的可用性很重要，这是因为对于大型系统来说，完全记录所有数据移动的方法所需要的活动日志空间量和永久日志空间量可能不切实践，并且性能通常较低。

还有一些仅当您选择了不可前滚恢复重新分发方法时才可用的功能部件和可选参数。例如，缺省情况下，此重新分发方法会停顿数据库并执行预检查，以确保满足先决条件。您还可以选择指定在重新分发操作过程中重建索引和收集表统计信息。通过将这些不同的手动任务组合在一起并使它们自动执行，将使这些任务不容易出错、速度更快并且效率更高，同时您还可以对这些操作进行更多的控制。

不可前滚恢复重新分发方法会自动重组表，这可以释放磁盘空间。这个表重组过程不会对重新分发操作的性能有其他影响。对于带有集群索引的表，重组过程不会尝试维护集群。如果需要完整的集群，那么在完成数据重新分发之后，必须对带有集群索引的表执行 **REORG TABLE** 命令。对于多维集群 (MDC) 表，重组过程会维护该表的集群并释放未使用的块以供复用；然而，在重新分发之后，该表的总大小不会改变。

注: 在完成重新分发操作之后，对每个受影响的表空间或整个数据库进行备份非常关键，这是因为对此类型的重新分发操作进行前滚会导致将所有已重新分发的表标记为无效。这样的表只能删除，这意味着没有任何方法来恢复这些表中的数据。对于可恢复的数据库，这就是在指定了 **NOT ROLLFORWARD RECOVERABLE** 选项的情况下发出 **REDISTRIBUTE DATABASE PARTITION GROUP** 实用程序时，该实用程序使它涉及的所有表空间处于 BACKUP PENDING 状态的原因。此状态强制您在结束成功的重新分发操作时备份所有已重新分发的表空间。借助在重新分发操作之后生成的备份，您应该不需要对重新分发操作本身进行前滚。

缺乏前滚可恢复性有一个很严重的后果: 如果在重新分发操作运行期间（包括结束重新分发时对重新分发所涉及的表空间进行备份的那段时间），您选择允许对数据库中的表进行更新（即使是正在重新分发的数据库分区组外部的表），那么在发生严重故障（例如，毁坏了数据库容器）时，可能会丢失这样的更新。可能丢失这样的更新的原因是该重新分发操作不可前滚恢复。如果必须从重新分发操作之前生成的备份中复原数据库，那么要重放重新分发操作期间进行的更新，那么就不可能只通过日志进行前滚而不同时对重新分发（如上所述，该操作会使重新分发的表处于 UNAVAILABLE 状态）进行前滚。因此，在此情况下唯一可以执行的操作是从重新分发之前生成的备份中复原数据库，但不执行前滚。然后可以再次执行重新分发操作。不幸的是，将丢失原始重新分发操作期间进行的所有更新。

无论怎样强调此缺点的严重性都不会过份。要确信重新分发操作期间不会丢失更新，必须满足下列其中一项:

- 必须避免在 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令的操作期间（包括完成该命令之后对受影响的表空间进行备份的那段时间）进行更新。
- 通过将 **QUIESCE DATABASE** 命令参数设置为 YES 来执行重新分发操作。您仍然必须确保允许访问已停顿的数据库的任何应用程序或用户不会进行更新。
- 重新分发操作期间应用的更新来自可重复的源，这意味着可以在任何时候再次应用这些更新。例如，如果更新源是文件中存储的数据并且在批处理期间应用了这些更新，那么很明显，即使发生了需要进行数据库复原的故障，更新也不会丢失，这是因为只需在任何时候再次应用这些更新即可。

关于是否允许在重新分发操作期间更新数据库，您必须根据在执行数据库复原（如果有必要）之后是否可以重复这些更新来确定这样的更新是否适当。

注: 并非 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令的操作期间发生的每个故障都会导致此问题。事实上，大多数故障都不会导致此问题。**REDISTRIBUTE DATABASE PARTITION GROUP** 命令是完全可重复的，这意味着当该实用程序在运行中失败时，可以使用 **CONTINUE** 或 **ABORT** 选项轻松地继续执行或者中止该命令。上面提到的故障需要用户通过在重新分发操作之前生成的备份进行复原。

进行日志记录的可恢复重新分发

REDISTRIBUTE DATABASE PARTITION GROUP 命令的原始和缺省版本，此方法使用标准 SQL 插入和删除操作来重新分发数据。将对所有行移动执行全面的日志记录，以便数据库是可恢复的，因此可以使用 **RESTORE DATABASE** 命令来复原数据库，然后使用 **ROLLFORWARD DATABASE** 命令来前滚所有更改。

在执行数据重新分发之后，源表将包含空的空间，这是因为删除了行并将其发送到新的数据库分区。如果要释放空的空间，必须重组这些表。要重组这些表，必须在完成重新分发之后使用单独的操作。要提高此方法的性能，请在完成重新分发之后删除索引并重新创建。

数据重新分发的先决条件

必须先满足特定先决条件，才能成功地对数据库分区组中的一组表执行数据重新分发。

以下是必要先决条件列表:

- 有权从所选的受支持数据重新分发接口执行数据重新分发。
- 在系统活动较少的时间段内需要大量时间来执行重新分发操作。
- 包含要在数据重新分发操作期间重新分发的数据的所有表必须都处于正常状态。例如，这些表不能处于 LOAD PENDING 状态或其他不可访问的装入表状态。要检查表状态，请在数据库分区组中的每个分区建立连接并发出 **LOAD QUERY** 命令。此命令的输出包含有关表状态的信息。**LOAD QUERY** 命令的文档说明了每个表状态的含义以及如何将表从一种状态移至另一种状态。
- 必须已使用分布键定义了数据库分区中要重新分发的所有表。如果已向单一分区系统添加了新数据库分区，那么直到这些分区中的所有表都有分布键时，才能执行数据重新分发。对于使用 CREATE TABLE 语句创建的表，如果这些表具有未包含分布键的定义，那么必须使用 ALTER TABLE 语句来改变这些表，以便在重新分发数据之前添加分布键。

- 重新分发数据之前，必须先删除包含在数据库分区组中的重复具体化查询表。存储具体化查询表定义的副本，以便可在数据重新分发完成后重新创建那些定义。
- 如果期望进行非统一重新分发，那么必须创建重新分发映射作为目标分发映射，以便用作重新分发接口的参数。
- 必须使用 **BACKUP DATABASE** 命令来创建数据库的备份。虽然此备份并非必须满足的先决条件，但强烈建议完成此备份。
- 必须建立从目录数据库分区至数据库的连接。
- 在数据重新分发期间或之后，重新构建所有索引必须有充分的可用空间。**INDEXING MODE** 命令参数会影响何时重新构建索引。
- 当指定了 **NOT ROLLFORWARD RECOVERABLE** 命令参数时，应提供足够的空间以便将状态信息写入由 IBM 服务中心用于确定问题的控制文件。将在下列路径中生成控制文件，当完成数据重新分发操作时应手动删除这些控制文件。
  - 在 Linux 和 UNIX 操作系统上：**diagpath**/redist/*db_name*/*db_partitiongroup_name*/*timestamp*/
  - 在 Windows 操作系统上：**diagpath**\redist*db_name**db_partitiongroup_name**timestamp*\

  可以使用以下公式来计算控制文件需要的空间量（以字节计）：

  （*数据库分区组中所有表的页数*）* 64 字节 + *数据库分区组中 LOB 值的数目*）* 600 字节

  要估计*数据库分区组中 LOB 值的数目*，请将所有表中的 LOB 列数相加，再乘以最大的表中的行数。
- 未指定 **NOT ROLLFORWARD RECOVERABLE** 命令参数时，必须提供足够的日志文件空间，以包含与数据重新分发期间执行的 INSERT 和 DELETE 操作相关联的日志条目，否则数据重新分发将中断或失败。

数据库配置参数 **util_heap_sz** 对数据库分区间的数据移动的处理很关键 - 系统会在重新分发操作持续时间内将尽可能多的内存分配给 **util_heap_sz**。如果在重新分发操作期间重建索引，那么还需要足够的 **sortheap**。应根据需要增大 **util_heap_sz** 和 **sortheap** 数据库配置参数的值以提高重新分发性能。

对数据重新分发的限制

在进行数据重新分发之前或者对与数据重新分发相关的问题进行故障诊断时，一定要注意对数据重新分发的一些限制。

数据重新分发存在下列限制:

- 没有分区键定义的表所在的分区上的数据重新分发受到限制。
- 进行数据重新分发时:
  - 在同一数据库分区组上启动另一个重新分发操作受到限制。
  - 删除数据库分区组受到限制。
  - 改变数据库分区组受到限制。
  - 对数据库分区组中的任何表执行 ALTER TABLE 语句受到限制。
  - 进行数据重新分发时在表中创建新索引受到限制。
  - 进行数据重新分发时删除对表定义的索引受到限制。
  - 进行数据重新分发时查询表中的数据受到限制。

  - 进行数据重新分发时更新表受到限制。
- 如果使用其中指定了 **NOT ROLLFORWARD RECOVERABLE** 命令参数的 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令启动了数据重新分发，那么对正在经历该数据重新分发的数据库中的表进行更新将受到限制。尽管可以进行更新，但如果数据重新分发中断，那么对数据所作的更改可能丢失，所以强烈建议不要这样做。
- 当发出 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令并指定了 **NOT ROLLFORWARD RECOVERABLE** 命令参数时:
  - 重新分发期间进行的数据分布更改将不可前滚恢复。
  - 如果数据库是可恢复的，那么在访问表空间内的第一个表之后，表空间将进入备份暂挂状态。要让该表脱离此状态，则必须在重新分发操作完成时创建表空间更改的备份。
  - 在数据重新分发期间，不能更新数据库分区组中要重新分发的表中的数据，该数据是只读的。要主动重新分发的表是不可访问的。
- 对于类型（层次结构）表，如果使用 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令并且对 **TABLE** 参数指定值 ONLY，那么表名将限于仅作为根表的名称。不能指定子表名称。
- 支持数据重新分发在数据库分区之间移动数据。然而，对于分区表，除非同时满足下列两个条件，否则在数据分区表的范围之间移动数据受到限制:
  - 分区表在 SYSTABLES.ACCESS_MODE 目录表中的访问方式为 FULL ACCESS。
  - 分区表没有当前相连或拆离的任何分区。
- 对于重复的具体化查询表，如果数据库分区组中的数据包含重复的具体化查询表，那么必须在重新分发数据之前删除这些表。重新分发数据之后，可以重新创建具体化查询表。
- 对于包含多维集群表 (MDC) 的数据库分区，如果数据库分区组中存在任何包含已转出块（正在暂挂清除）的多维集群表，那么使用 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令受到限制并且无法继续运行。必须先清除 MDC 表，才能恢复或重新启动数据重新分发。
- 删除 DB2 目录视图中当前标记为处于“正在重新分发”状态的表时受到限制。要删除处于此状态的表，首先对适当的表列表运行带 **ABORT** 或 **CONTINUE** 参数的 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令，以便完成或中止该表的重新分发操作。

确定是否需要重新分发数据

在确定是否需要重新分发数据时，确定数据库分区组或表的当前数据分布很有用。还可以使用有关当前数据分布的详细信息来创建定制分发映射，此映射指定如何分发数据。

关于此任务

如果向数据库分区组添加了新的数据库分区，或者从数据库分区组删除了现有数据库分区，请执行数据重新分发以平衡所有数据库分区之间的数据。

如果未在数据库分区组中添加或删除任何数据库分区，那么通常仅当数据库分区组的数据库分区间的数据存在不等数据分布时才需要进行数据重新分发。注意，在某些情况下，数据的不均匀分发可能是所需的。例如，如果某些数据库分区驻留在功能强大的机器上，那么让这些数据库分区包含比其他分区更多的数据量可能是有利的。

过程

要确定是否需要重新分发数据:

1. 获取有关数据库分区组中数据库分区之间的当前数据分布的信息。

   对数据库分区组中最大的表（或者有代表性的表）运行以下查询:

   ```
   SELECT DBPARTITIONNUM(column_name), COUNT(*) FROM table_name
         GROUP BY DBPARTITIONNUM(column_name)
         ORDER BY DBPARTITIONNUM(column_name) DESC
   ```

   此处 *column_name* 是表 *table_name* 的分布键的名称。

   此查询的输出显示每个数据库分区中驻留了 *table_name* 中的多少记录。如果数据库分区之间的数据分布不是所需的，请继续执行下一个步骤。

2. 获取有关散列分区之间的数据分布的信息。

   使用上一步骤中使用的 *column_name* 和 *table_name* 运行以下查询:

   ```
   SELECT HASHEDVALUE(column_name), COUNT(*) FROM table_name
         GROUP BY HASHEDVALUE(column_name)
         ORDER BY HASHEDVALUE(column_name) DESC
   ```

   可以轻松地使用此查询的输出来构造当指定了 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令的 **USING DISTFILE** 参数时所需的分发文件。要了解该分发文件的格式描述，请参阅 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令参考信息。

3. 可选: 如果数据需要重新分发，那么您可以计划在系统维护期间执行此操作。

   当指定了 **USING DISTFILE** 参数时，**REDISTRIBUTE DATABASE PARTITION GROUP** 命令将使用分发文件中的信息来为数据库分区组生成新的分区映射。此操作将在数据库分区之间均匀分发数据。

   如果均匀分发不是所需的，那么您可以为重新分发操作构造自己的目标分区映射。可以使用 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令中的 **USING TARGETMAP** 参数来指定目标分区映射。

结果

进行此调查后，您将知道数据的分发是否均匀或者是否需要重新分发数据。

通过使用 REDISTRIBUTE DATABASE PARTITION GROUP 命令在数据库分区之间重新分发数据

REDISTRIBUTE DATABASE PARTITION GROUP 命令是执行数据重新分发时的建议接口。

过程

要在数据库分区组中的数据库分区之间重新分发数据:

1. 可选: 对数据库执行备份。 请参阅 **BACKUP DATABASE** 命令。

   强烈建议您在执行不可前滚恢复数据重新分发之前，创建数据库的备份副本。

2. 连接至包含系统目录表的数据库分区。 请参阅 CONNECT 语句。
3. 发出 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令。

注: 在先前版本的 DB2 数据库产品中，此命令使用 **NODEGROUP** 关键字而不是 **DATABASE PARTITION GROUP** 关键字。

指定下列自变量:

数据库分区组名
必须指定要在其中重新分发数据的数据库分区组。

UNIFORM
可选: 指定将均匀分发数据。**UNIFORM** 是未指定分布类型时的缺省值，所以尚未指定其他分布类型时，可以省略此选项。

USING DISTFILE *distfile-name*
可选: 指定希望进行定制分发，并指定分发文件的路径名，该文件包含的数据将定义期望的数据偏差。此文件的内容用于生成目标分发映射。

USING TARGETMAP *targetmap-name*
可选: 指定要使用目标数据重新分发映射，并指定包含目标重新分发映射的文件的名称。

有关详细信息，请参阅 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令行实用程序信息。

4. 允许该命令以非中断方式运行。 该命令完成时，如果成功进行了数据重新分发，请执行以下操作:
   - 备份数据库分区组中所有处于 BACKUP PENDING 状态的表空间。也可以执行完整数据库备份。

     注: 仅当数据库可恢复并且在 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令中使用了 **NOT ROLLFORWARD RECOVERABLE** 命令参数时，表空间才会进入备份暂挂状态。
   - 重新创建在重新分发之前删除的任何重复的具体化查询表。
   - 如果满足下列条件，请执行 **RUNSTATS** 命令:
     – 在 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令中指定了 **STATISTICS NONE** 命令参数，或者省略了 **NOT ROLLFORWARD RECOVERABLE** 命令参数。这两个条件都意味着在数据重新分发时期未收集统计信息。
     – 数据库分区组中有一些表具有统计信息概要文件。

     RUNSTATS 命令将收集在为查询选择数据存取方案时要使用的 SQL 编译器和优化器的数据分布统计信息。
   - 如果指定了 **NOT ROLLFORWARD RECOVERABLE** 命令参数，请删除位于下列路径中的控制文件:
     – 在 Linux 和 UNIX 操作系统上: **diagpath**/redist/*db_name*/*db_partitiongroup_name*/*timestamp*/
     – 在 Windows 操作系统上: **diagpath**\redist*db_name**db_partitiongroup_name**timestamp*\

结果

已完成数据重新分发，并且重新分发日志文件中提供了有关数据重新分发过程的信息。有关已使用的分发映射的信息可在 DB2 说明表中找到。

在数据库分区组中重新分发数据

要为数据库分区组创建有效重新分发计划并重新分发数据，请发出 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令或调用 sqludrdt API。

开始之前

要使用数据库分区组，必须具有 SYSADM、SYSCTRL 或 DBADM 权限。

过程

要在数据库分区组中重新分发数据，请执行以下操作:

- 在命令行处理器 (CLP) 中发出 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令。
- 通过使用 ADMIN_CMD 过程来发出 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令。
- 调用 sqludrdt API

数据重新分发的日志空间要求

要成功执行数据重新分发操作，必须分配足够的日志文件空间以确保不会中断数据重新分发。当您指定 **NOT ROLLFORWARD RECOVERABLE** 命令参数时，日志空间要求较小，因为在该类型的数据重新分发期间，将进行最少的日志记录。

所需的日志文件空间量取决于多个因素，其中包括使用了 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令的哪些选项。

在数据重新分发可前滚恢复的情况下从任何受支持的接口执行重新分发时:

- 日志必须大到足以允许要重新分发的数据所在的每个数据库分区上的 INSERT 和 DELETE 操作。最大的日志记录要求在将失去最多数据或将获得最多数据的数据库上产生。
- 如果要将数据移至大量数据库分区，请使用当前数据库分区数目与新数据库分区数目之比来估计 INSERT 和 DELETE 操作的数目。例如，考虑重新分发在重新分发之前统一分发的数据。如果将数据从四个数据库分区移至五个数据库分区，四个原始数据库分区的大约 20% 的数据将移至新的数据库分区。这意味着，20% 的 DELETE 操作将在四个原始数据库分区上进行，而所有 INSERT 操作将在新数据库分区上进行。
- 考虑非统一数据分布，例如，分布键包含许多 NULL 值的情况。在此情况下，分布键包含 NULL 值的所有行将从旧分发方案中的一个数据库分区移至新分发方案中的另一个数据库分区。因此，这两个数据库分区上所需的日志空间量将增加，并且可能超出采用统一分发计算出的日志空间量。
- 每个表的重新分发都是单个事务。因此，估计日志空间时，应将最大表大小乘以更改的百分比（如 20%）。但是，考虑最大的表可能是以统一形式分发的，而第二大的表（例如）可能有一个或多个膨胀的数据库分区。在此情况下，应考虑使用非统一分发的表，而不是最大的表。

注: 估计了要在数据库分区上插入或删除的最大数据量后，将估计值翻倍以确定活动日志的峰值大小。如果此估计值大于活动日志限制 1024 GB，那么必须分步执行数据重新分发。例如，将 STEPWISE_REDISTRIBUTE_DBPG 过程与某个数目的步骤配合使

用，该数目与估计值超过活动日志限制的大小成正比。还可以将 **logsecond** 数据库配置参数设置为 -1，以避免大多数日志空间问题。

在数据重新分发不可前滚恢复情况下从任何受支持的接口执行重新分发时:

- 在数据重新分发期间移动行时不会创建日志记录。此行为将极大地降低日志文件空间要求；但是当此选项与数据库前滚恢复配合使用时，重新分发操作日志记录不能前滚，并且前滚操作期间处理的所有表都将保持 UNAVAILABLE 状态。
- 如果进行数据重新分发时数据库分区组包含的表中带有长字段 (LF) 或大对象 (LOB) 数据，那么数据重新分发期间生成的日志记录数较高，原因是将为每行数据创建一个日志记录。在此情况下，期望每个数据库分区的日志空间要求大约是在该分区上移动的数据量（即，发送和/或接收的数据）的三分之一。

重新分发事件日志文件

在数据重新分发期间将执行事件日志记录。将事件信息记录至事件日志文件，以便日后用于执行错误恢复。

执行数据重新分发时，会将有关所处理的每个表的信息全部记录在一对事件日志文件中。将事件日志文件命名为 *database-name.database-partition-group-name.timestamp*.log 和 *database-name.database-partition-group-name.timestamp*。

日志文件的位置如下:

- 在 Linux 和 UNIX 操作系统上为 *homeinst*/sqllib/redist 目录
- 在 Windows 操作系统上为 *db2instprof*\instance\redist 目录，其中 *db2instprof* 是 **DB2INSTPROF** 注册表变量的值

以下是事件日志文件名的示例:

```
SAMPLE.IBMDEFAULTGROUP.2012012620240204
SAMPLE.IBMDEFAULTGROUP.2012012620240204.log
```

这些文件可用于称为 SAMPLE 的数据库（具有称为 IBMDEFAULTGROUP 的数据库分区组）上的重新分发操作。这些文件创建于 2012 年 1 月 26 日下午 8:24（本地时间）。

事件日志文件的三个主要用途为如下所示:

- 提供有关重新分发操作的一般信息，例如，旧的分发映射和新的分发映射。
- 为用户提供一些信息，以帮助他们确定迄今为止已通过实用程序重新分发了哪些表。
- 提供有关已经重新分发的每个表的信息，这些信息包括：用于该表的建立索引方式、指示是否已成功重新分发该表以及对该表执行重新分发操作的开始时间和结束时间。

使用 STEPWISE_REDISTRIBUTE_DBPG 过程来重新分发数据库分区组

可以使用内置过程来执行数据重新分发。

过程

要使用 STEPWISE_REDISTRIBUTE_DBPG 过程来重新分发数据库分区组:

1. 使用 ANALYZE_LOG_SPACE 过程来分析与日志空间可用性和数据偏差有关的数据库分区组。

   ANALYZE_LOG_SPACE 过程将返回日志空间分析结果的结果集（开放式游标），其中包含给定数据库分区组的每个数据库分区的字段。
2. 使用 GENERATE_DISTFILE 过程为给定的表创建数据分布文件。

   GENERATE_DISTFILE 过程将为给定的表生成数据分布文件，并使用提供的文件名来保存该文件。
3. 使用 STEPWISE_REDISTRIBUTE_DBPG 过程为数据库分区组创建并报告按步骤重新分发计划的内容。
4. 使用 GET_SWRD_SETTINGS 和 SET_SWRD_SETTINGS 过程为给定的表创建数据分布文件。

   GET_SWRD_SETTINGS 过程将读取给定数据库分区组的现有重新分发注册表记录。

   SET_SWRD_SETTINGS 过程将创建或更改重新分发注册表。如果该注册表不存在，那么将创建它并向其中添加记录。如果该注册表已存在，那么它会使用 *overwriteSpec* 来标识需要覆盖的字段值。*overwriteSpec* 字段使此函数能够将 NULL 输入用于不需要更新的字段。
5. 使用 STEPWISE_REDISTRIBUTE_DBPG 过程按照计划重新分发数据库分区组。

   STEPWISE_REDISTRIBUTE_DBPG 过程将根据输入和设置文件来重新分发数据库分区组的一部分。

示例

以下是 AIX 上的 CLP 脚本的示例:

```
# ------------------------------------------------------------------------------------
# Set the database you wish to connect to
# ------------------------------------------------------------------------------------
dbName="SAMPLE"

# ------------------------------------------------------------------------------------
# Set the target database partition group name
# ------------------------------------------------------------------------------------
dbpgName="IBMDEFAULTGROUP"

# ------------------------------------------------------------------------------------
# Specify the table name and schema
# ------------------------------------------------------------------------------------
tbSchema="$USER"
tbName="STAFF"

# ------------------------------------------------------------------------------------
# Specify the name of the data distribution file
# ------------------------------------------------------------------------------------
distFile="$HOME/sqllib/function/$dbName.IBMDEFAULTGROUP_swrdData.dst"

export DB2INSTANCE=$USER
export DB2COMM=TCPIP

# ------------------------------------------------------------------------------------
```

```
# Invoke call statements in clp
# ------------------------------------------------------------------------------------
   db2start db2 -v "connect to $dbName"

# ------------------------------------------------------------------------------------
# Analysing the effect of adding a database partition without applying the changes - a 'what if'
# hypothetical analysis
#
# - In the following case, the hypothesis is adding database partition 40, 50 and 60 to the
#   database partition group, and for database partitions 10,20,30,40,50,60, using a respective
#   target ratio of 1:2:1:2:1:2.
#
# NOTE: in this example only partitions 10, 20 and 30 actually exist in the database
#       partition group
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'A', '40,50,60', '10,20,30,40,50,60', '1,2,1,2,1,2')"

# ------------------------------------------------------------------------------------
# Analysing the effect of dropping a database partition without applying the changes
#
# - In the following case, the hypothesis is dropping database partition 30 from the database
#   partition group, and redistributing the data in database partitions 10 and 20 using a
#   respective target ratio of 1 : 1
#
# NOTE: In this example all database partitions 10, 20 and 30 should exist in the database
#       partition group
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.analyze_log_space('$dbpgName', '$tbSchema', '$tbName', 2, ' ',
'D', '30', '10,20','1,1')"

# ------------------------------------------------------------------------------------
# Generate a data distribution file to be used by the redistribute process
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.generate_distfile('$tbSchema', '$tbName', '$distFile')"

# ------------------------------------------------------------------------------------
# Write a step wise redistribution plan into a registry
#
# Setting the 10th parameter to 1, may cause a currently running step wise redistribute
# stored procedure to complete the current step and stop, until this parameter is reset
# to 0, and the redistribute stored procedure is called again.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.set_swrd_settings('$dbpgName', 255, 0, ' ', '$distFile', 1000,
12, 2, 1, 0, '10,20,30', '50,50,50')"

# ------------------------------------------------------------------------------------
# Report the content of the step wise redistribution plan for the given database
# partition group.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.get_swrd_settings('$dbpgName', 255, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)"

# ------------------------------------------------------------------------------------
# Redistribute the database partition group "dbpgName" according to the redistribution
# plan stored in the registry by set_swrd_settings. It starting with step 3 and
# redistributes the data until 2 steps in the redistribution plan are completed.
# ------------------------------------------------------------------------------------
db2 -v "call sysproc.stepwise_redistribute_dbpg('$dbpgName', 3, 2)"
```

第 27 章 配置自调整内存

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。

第 28 章 DB2 配置参数和变量

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的更改的更多详细信息，请参阅 *升级到 DB2 V10.5* 中的『DB2 服务器行为更改』。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。

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。

分区数据库环境配置参数

通信

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。

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，缓冲区数只能增加为比起始值多 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** 参数设置为特定值和 AUTOMATIC 并且您不希望系统控制器线程将资源量调整为小于指定值，请将 **DB2_FCM_SETTINGS** 注册表变量的 FCM_CFG_BASE_AS_FLOOR 选项设置为 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。

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_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。

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。

并行处理

intra_parallel -“启用分区内并行性”

此参数指定在缺省情况下数据库连接是否将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。

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。

第 5 部分 管理 API, 命令和 SQL 语句

第 29 章 管理 API

sqleaddn - 将数据库分区添加至分区数据库环境

将数据库分区添加至数据库分区服务器。

作用域

此 API 只影响对其执行该 API 的数据库分区服务器。

权限

下列权限中的一项:

- SYSADM
- SYSCTRL

必需的连接

无

API 包含文件

```
sqlenv.h
```

API 和数据结构语法

```
SQL_API_RC SQL_API_FN
  sqleaddn (
        void * pAddNodeOptions,
           struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgaddn (
        unsigned short addnOptionsLen,
        struct sqlca * pSqlca,
        void * pAddNodeOptions);
```

sqleaddn API 参数

pAddNodeOptions

输入。指向可选的 sqle_addn_options 结构的指针。对于要创建的所有数据库分区，此结构用来指定系统临时表空间定义的源数据库分区服务器（如果有的话）。如果未指定（即，指定了 NULL 指针），那么系统临时表空间定义将与目录分区的那些表空间定义相同。

pSqlca

输出。指向 sqlca 结构的指针。

特定于 sqlgaddn API 的参数

addnOptionsLen

输入。一个 2 字节的无符号整数，表示可选的 sqle_addn_options 结构的长度（以字节计）。

使用说明

仅当数据库分区服务器已添加至具有一个数据库的环境并且在添加分区操作时该数据库未进行编目，才应该使用此 API。在此情况下，因为该数据库未进行编目，所以添加分区操作不会识别该数据库，并且不会在新的数据库分区服务器上为其创建数据库分区。在新的数据库分区服务器上进行任何连接数据库分区的尝试都会导致错误。必须首先对该数据库进行编目，才能使用 sqleaddn API 来在新的数据库分区服务器上为该数据库创建数据库分区。

如果环境中具有多个数据库，并且至少其中一个数据库在添加分区操作时已进行编目，那么不应该使用此 API。在此情况下，请使用 sqlecran API 来为在添加分区操作时未编目的每个数据库都创建数据库分区。必须首先对未编目的每个数据库进行编目，才能使用 sqlecran API 来在新的数据库分区服务器上为该数据库创建数据库分区。

在添加新的数据库分区之前，请确保具有足够的存储空间用于必须创建的容器。

“添加节点”操作在新数据库分区服务器上为存在于实例中的每个数据库创建空的数据库分区。将新数据库分区的配置参数设置为缺省值。

注: 当添加新数据库分区时，不会识别任何未编目的数据库。在新数据库分区上将不存在未编目的数据库。尝试与新数据库分区上的数据库进行连接会返回错误消息 SQL1013N。

如果在本地创建数据库分区时“添加节点”操作失败，那么它会进入清除阶段，以本地方式删除已创建的所有数据库。这意味着仅从添加的数据库分区服务器（即，本地数据库分区服务器）除去数据库分区。现有数据库分区在所有其他数据库分区服务器上不受影响。如果此操作失败，那么将不执行进一步清除，并且返回错误。

直到使用 ALTER DATABASE PARTITION GROUP 语句将数据库分区服务器添加到数据库分区组后，才可以使用新数据库分区服务器上的数据库分区来包含用户数据。

如果创建数据库或删除数据库操作正在进行中，那么此 API 将会失败。当操作完成时，可再次调用该 API。

当 sqleaddn API 必须与实例中每个数据库的目录分区通信，要检索存储器组存储路径定义。同样，如果要使用数据库分区创建系统临时表空间，那么 sqleaddn API 就可能必须与分区数据库环境中的另一数据库分区服务器通信，以便检索表空间定义。**start_stop_time** 数据库管理器配置参数用于指定以下时间：其他数据库分区服务器必须使用自动存储器和表空间定义来进行响应的时间（以分钟计）。如果超出此时间，那么 API 失败。请增大 **start_stop_time** 的值，然后再次调用该 API。

REXX API 语法

可以通过 SQLDB2 接口从 REXX 调用此 API。

---

sqlecran - 在数据库分区服务器上创建数据库

仅在调用 API 的数据库分区服务器上创建数据库。

此 API 不准备用于一般用途。例如，如果数据库分区服务器上的数据库分区受损且必须重新创建，那么它应该与 db2Restore 配合使用。错误使用此 API 可能会导致系统中出现不一致情况，因此应谨慎使用。

注: 如果此 API 用于重新创建已删除的数据库分区（因为它已损坏），那么此数据库分区服务器上的数据库将处于复原暂挂状态。在重新创建数据库分区后，必须在此数据库分区服务器上立即复原该数据库。

作用域

此 API 只影响在其上调用该 API 的数据库分区服务器。

权限

下列权限中的一项:

- SYSADM
- SYSCTRL

必需的连接

实例。要在另一数据库分区服务器上创建数据库，首先必须连接至该数据库分区服务器。此 API 会在处理期间临时建立数据库连接。

API 包含文件

```
sqlenv.h
```

API 和数据结构语法

```
SQL_API_RC SQL_API_FN
  sqlecran (
        char * pDbName,
        void * pReserved,
           struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgcran (
        unsigned short reservedLen,
        unsigned short dbNameLen,
        struct sqlca * pSqlca,
        void * pReserved,
        char * pDbName);
```

sqlecran API 参数

pDbName
输入。包含要创建的数据库名的字符串。不能为 NULL。

pReserved
输入。被设置为 null 或指向零的备用指针。保留以备将来使用。

pSqlca
输出。指向 sqlca 结构的指针。

特定于 sqlgcran API 的参数

reservedLen
输入。保留用于 **pReserved** 的长度。

dbNameLen
输入。一个 2 字节的无符号整数，表示数据库名的长度（以字节计）。

使用说明

当成功创建数据库时，它被置于复原暂挂状态。在可以使用数据库之前，必须在此数据库分区服务器上复原该数据库。

REXX API 语法

可以通过 SQLDB2 接口从 REXX 调用此 API。

---

sqledpan - 删除数据库分区服务器上的数据库

删除指定的数据库分区服务器上的数据库。只能在分区数据库环境中运行。

作用域

此 API 只影响在其上调用该 API 的数据库分区服务器。

权限

下列权限中的一项:

- SYSADM
- SYSCTRL

必需的连接

无。在调用期间建立实例连接。

API 包含文件

```
sqlenv.h
```

API 和数据结构语法

```
SQL_API_RC SQL_API_FN
  sqledpan (
        char * pDbAlias,
        void * pReserved,
           struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdpan (
        unsigned short Reserved1,
        unsigned short DbAliasLen,
        struct sqlca * pSqlca,
        void * pReserved2,
        char * pDbAlias);
```

sqledpan API 参数

pDbAlias

输入。包含要删除的数据库的别名的字符串。此名称用于引用系统数据库目录中的实际数据库名。

pReserved

保留参数。应为 NULL。

pSqlca

输出。指向 sqlca 结构的指针。

特定于 sqlgdpan API 的参数

Reserved1
保留以备将来使用。

DbAliasLen
输入。一个 2 字节的无符号整数，表示数据库别名的长度（以字节计）。

pReserved2
被设置为 null 或指向零的备用指针。保留以备将来使用。

使用说明

错误使用此 API 可能会导致系统中出现不一致情况，因此应谨慎使用。

REXX API 语法

可以通过 SQLDB2 接口从 REXX 调用此 API。

---

sqledrpn - 检查是否可以删除数据库分区服务器

验证数据库是否正在使用数据库分区服务器。将会返回消息，指示能否删除数据库分区服务器。

作用域

此 API 只影响在其上发出 API 的数据库分区服务器。

权限

下列权限中的一项:

- SYSADM
- SYSCTRL

API 包含文件

```
sqlenv.h
```

API 和数据结构语法

```
SQL_API_RC SQL_API_FN
  sqledrpn (
        unsigned short Action,
        void * pReserved,
           struct sqlca * pSqlca);

SQL_API_RC SQL_API_FN
  sqlgdrpn (
        unsigned short Reserved1,
        struct sqlca * pSqlca,
        void * pReserved2,
        unsigned short Action);
```

sqledrpn API 参数

操作 所请求的操作。有效值为: SQL_DROPNODE_VERIFY

pReserved
保留参数。应该是 NULL。

pSqlca
输出。指向 sqlca 结构的指针。

特定于 sqlgdrpn API 的参数

Reserved1
保留用于 **pReserved2** 的长度。

pReserved2
被设置为 NULL 或指向 0 的备用指针。保留以供将来使用。

使用说明

如果返回消息，指示数据库分区服务器未在使用中，那么将 **db2stop** 命令与 **DROP NODENUM** 配合使用，以便从 db2nodes.cfg 文件中除去数据库分区服务器的条目，它将从分区数据库环境中除去数据库分区服务器。

如果返回消息，指示数据库分区服务器正在使用中，那么应采取下列操作:

1. 对于实例中的每个数据库，要删除的数据库分区服务器将在其上具有数据库分区。如果任何这些数据库分区包含数据，那么重新分发那些使用这些数据库分区的数据库分区组。重新分发数据库分区组，以将数据移至存在于数据库分区服务器的尚未被删除的数据库分区。
2. 在重新分发数据库分区组之后，从使用数据库分区的每个数据库分区组中将其删除。要从数据库分区组中除去数据库分区，您可以使用 sqludrdt API 的删除节点选项或 ALTER DATABASE PARTITION GROUP 语句。
3. 删除在数据库分区服务器上定义的任何事件监视器。
4. 重新运行 sqledrpn 以确保数据库分区服务器上的数据库分区不再处于使用中。

REXX API 语法

可以通过 SQLDB2 接口从 REXX 调用此 API。

---

sqlugrpn - 为行获取数据库分区服务器号

从 V9.7 开始，建议不要使用此 API。请使用 db2GetRowPartNum（为行获取数据库分区服务器号）API 来返回行的数据库分区号和数据库分区服务器号。

如果调用 sqlugrpn API 且 **DB2_PMAP_COMPATIBILITY** 注册表变量设置为 OFF，那么会返回错误消息 SQL2768N。

基于分发键值返回数据库分区号和数据库分区服务器号。应用程序可以使用此信息来确定将特定的表行存储在哪个数据库分区服务器上。

分区数据结构 sqlupi 是此 API 的输入。该结构可由 sqlugtpi API 返回。另一个输入是相应分发键值的字符表示法。输出是由分布策略以及来自分发映射的相应数据库分区服务器号生成的数据库分区号。如果未提供分发映射信息，那么只返回数据库分区号。这在分析数据分布时非常有用。

在调用此 API 时，数据库管理器不需要处于运行状态。

作用域

必须从 db2nodes.cfg 文件中的数据库分区服务器调用此 API。由于客户机与服务器之间在代码页和尾数法方面存在差别，所以您不应该从客户机调用此 API，否则可能导致返回错误的数据库分区信息。

权限

无

API 包含文件

sqlutil.h

API 和数据结构语法

```
SQL_API_RC SQL_API_FN
  sqlugrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_ctrycode,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);

SQL_API_RC SQL_API_FN
  sqlggrpn (
   unsigned short num_ptrs,
   unsigned char ** ptr_array,
   unsigned short * ptr_lens,
   unsigned short territory_code,
   unsigned short codepage,
   struct sqlupi * part_info,
   short * part_num,
   SQL_PDB_NODE_TYPE * node_num,
   unsigned short chklvl,
   struct sqlca * sqlca,
   short dataformat,
   void * pReserved1,
   void * pReserved2);
```

sqlugrpn API 参数

num_ptrs

ptr_array 中的指针数目。该值必须与为 **part_info** 参数指定的值（即 **part_info->sqld**）相同。

ptr_array

指针数组，它指向 **part_info** 中指定的分布键的每个部分相应值的字符表示法。如果需要空值，那么将相应的指针设置为 null。对于生成列，此函数不会生成行值。用户负责提供将导致行正确分区的值。

ptr_lens

无符号整数的数组，它包含 **part_info** 中指定的分区键的每个部分相应值的字符表示法的长度。

territory_ctrycode
目标数据库的国家/地区代码。还可以使用 **GET DATABASE CONFIGURATION** 命令从数据库配置文件中获取此值。

codepage
目标数据库的代码页。您也可以使用 **GET DATABASE CONFIGURATION** 命令从数据库配置文件中获取此值。

part_info
指向 sqlupi 结构的指针。

part_num
指向用于存储数据库分区号的双字节有符号整数的指针。

node_num
指向用于存储节点号的 SQL_PDB_NODE_TYPE 字段的指针。如果指针为空，那么将不返回节点号。

chklvl 指定对输入参数进行的检查级别的无符号整数。如果指定的值为零，那么不进行检查。如果指定了任何非零的值，那么将检查所有输入参数。

sqlca 输出。指向 sqlca 结构的指针。

dataformat
指定分发键值的表示法。有效值为:

SQL_CHARSTRING_FORMAT
所有分发键值由字符串表示。这是缺省值。

SQL_IMPLIEDDECIMAL_FORMAT
隐含的小数点所在的位置由列定义来确定。例如，如果列定义为 DECIMAL(8,2)，那么值 12345 被处理为 123.45。

SQL_PACKEDDECIMAL_FORMAT
所有十进制列分发键值采用压缩十进制格式。

SQL_BINARYNUMERICS_FORMAT
所有数字分发键值采用大尾数法二进制格式。

pReserved1
保留以备将来使用。

pReserved2
保留以备将来使用。

使用说明

操作系统上支持的数据类型与可定义为分发键的那些数据类型相同。

注: 必须先将 CHAR、VARCHAR、GRAPHIC 和 VARGRAPHIC 数据类型转换为数据库代码页，然后才能调用此 API。

对于数字和日期时间数据类型，字符表示法必须在调用 API 的相应系统的代码页上。

如果 **node_num** 不为空，那么必须提供分布图；即，**part_info** 参数 (**part_info->pmaplen**) 中的 **pmaplen** 字段是 2 或 8192。否则，将返回 SQLCODE -6038。必须定义分布键；即，**part_info** 参数 (**part_info->sqld**) 中的 **sqld** 字段必须大于零。否则，将返回 SQLCODE -2032。

如果将空值分配给非空分区列，那么将返回 SQLCODE -6039。

对于 CHAR、VARCHAR、GRAPHIC 和 VARGRAPHIC 数据类型，只截掉尾部空格；对于除此之外的其他类型，输入字符串的所有前导空格和尾部空格都会被截掉。

第 30 章 命令

REDISTRIBUTE DATABASE PARTITION GROUP

在数据库分区组中的分区间重新分发数据。此命令影响数据库分区组中的所有对象，并且不能限制为只影响一个对象。

此命令只能从目录数据库分区发出。使用 **LIST DATABASE DIRECTORY** 命令，为每个数据库确定哪个数据库分区是目录数据库分区。

作用域

此命令会影响数据库分区组中的所有数据库分区。

权限

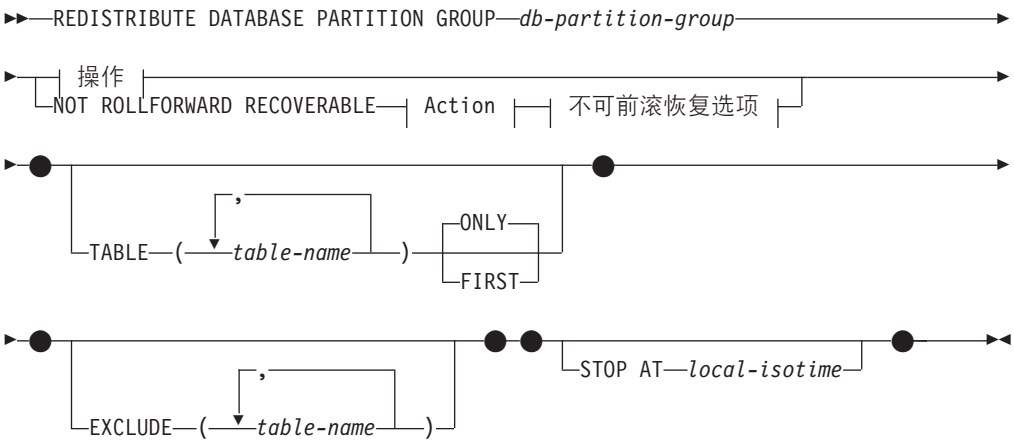
需要以下其中一项权限:

- SYSADM
- SYSCTRL
- DBADM

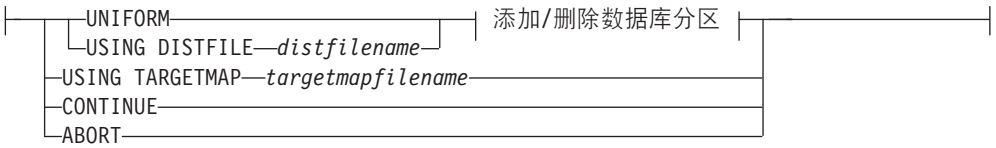
此外，还需要以下其中一组权限:

- 对正在重新分发的数据库分区组中所有表的 DELETE、INSERT 和 SELECT 特权
- DATAACCESS 权限

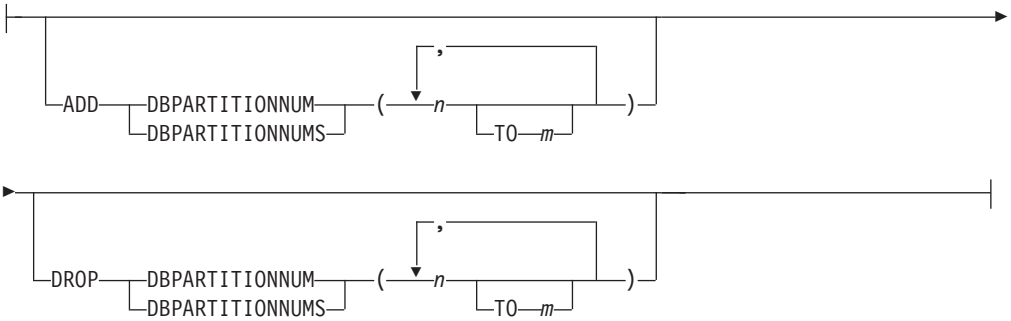
命令语法



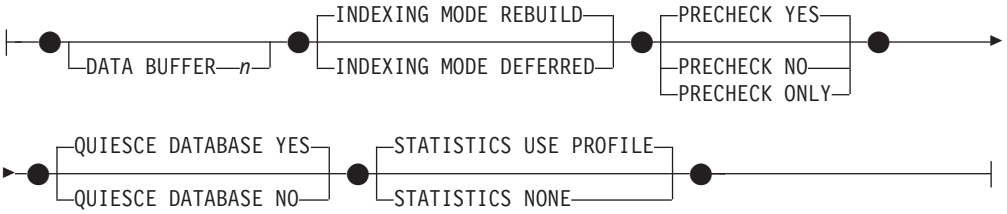
操作:



添加/删除数据库分区:



不可前滚恢复选项:



命令参数

DATABASE PARTITION GROUP *db-partition-group*
数据库分区组的名称。此单一部分名称标识了 SYSCAT.DBPARTITIONGROUPS 目录表中描述的数据库分区组。数据库分区组当前无法进行重新分发。

注: 无法将 IBMCATGROUP 和 IBMTEMPGROUP 数据库分区组中的表进行重新分发。

NOT ROLLFORWARD RECOVERABLE
如果使用此选项，那么 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令不可前滚恢复。

- 将成批移动数据，而不是通过内部插入和删除操作进行移动。这样会减少必须扫描和访问表的次数，从而提高性能。
- 对于每个插入和删除操作，不再需要日志记录。这意味着，当执行数据重新分发时，不再需要管理系统中的大量活动日志空间和日志归档空间。
- 如果使用带 **NOT ROLLFORWARD RECOVERABLE** 选项的 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令，那么重新分发操作对包含 XML 列的表使用 **INDEXING MODE DEFERRED** 选项。如果表未包含 XML 列，那么重新分发操作将使用在发出该命令时所指定的建立索引方式。

当*没有*使用此选项时，对所有行移动进行了大量记录，以致在有任何中断、错误或其他业务需要时可稍后恢复数据库。

UNIFORM
指定数据均匀分布在散列分区（即，假定每个散列分区均具有相同的行数），但相同数目的散列分区未映射至每个数据库分区。在重新分发之后，数据库分区组中的所有数据库分区均具有数目大致相同的散列分区。

USING DISTFILE *distfilename*
如果分发键值的分布不均匀，那么使用此选项以实现在数据库分区组的数据库分区之间均匀地重新分发数据。

使用 *distfilename* 以指示当前在 32 768 个散列分区之间的数据分布。

使用行计数、字节量或任何其他量度标准来指示每个散列分区表示的数据量。实用程序将与分区相关的整数值理解为该分区的权重。当指定 *distfilename* 时，实用程序会生成目标分发映射，用于在数据库分区组中的数据库分区之间尽可能均匀地重新分发数据。在重新分发之后，数据库分区组中每个数据库分区的权重将会大致相同（数据库分区的权重是映射至该数据库分区的所有散列分区的权重总和）。

例如，输入分布文件可能包含以下条目:

```
10223
1345
112000
0
100
...
```

在该示例中，散列分区 2 的权重为 112000，而权重为 0 的分区 3 根本没有映射至该分区的数据。

distfilename 应该包含 32 768 个字符格式的正整数值。这些值的总和应该小于或等于 4 294 967 295。

如果未指定 *distfilename* 的路径，那么将使用当前目录。

USING TARGETMAP *targetmapfilename*
使用在 *targetmapfilename* 中指定的文件作为目标分发映射。将根据此文件来重新分发数据。如果未指定路径，那么将使用当前目录。

targetmapfilename 应包含 32768 个整数，每个整数表示一个有效数据库分区号。任何行上的数字将散列值映射至数据库分区。这意味着如果行 *X* 包含值 *Y*，那么 HASHEDVALUE() 为 *X* 的每个记录将位于数据库分区 *Y* 上。

如果目标映射中包含的数据库分区不在该数据库分区组中，那么将返回错误。发出 ALTER DATABASE PARTITION GROUP ADD DBPARTITIONNUM 语句，然后运行 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令。

如果从目标映射中排除的数据库分区*位于*数据库分区组中，那么分区时将不包括该数据库分区。在使用 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令的前后，您都可以使用 ALTER DATABASE PARTITION GROUP DROP DBPARTITIONNUM 语句来删除此类数据库分区。

CONTINUE
继续先前失败或停止的 **REDISTRIBUTE DATABASE PARTITION GROUP** 操作。如果什么都未发生，那么将返回错误。

ABORT

中止先前失败或停止的 **REDISTRIBUTE DATABASE PARTITION GROUP** 操作。如果什么都未发生，那么将返回错误。

ADD

DBPARTITIONNUM *n*

TO *m*

n 或 *n* **TO** *m* 指定将添加至数据库分区组中的数据库分区编号列表。在数据库分区组中，任何指定的分区均必须尚未被定义（SQLSTATE 42728）。这相当于执行指定了 ADD DBPARTITIONNUM 子句的 ALTER DATABASE PARTITION GROUP 语句。

DBPARTITIONNUMS *n*

TO *m*

n 或 *n* **TO** *m* 指定将添加至数据库分区组中的数据库分区编号列表。在数据库分区组中，任何指定的分区均必须尚未被定义（SQLSTATE 42728）。这相当于执行指定了 ADD DBPARTITIONNUM 子句的 ALTER DATABASE PARTITION GROUP 语句。

注:

1. 使用此选项添加数据库分区时，表空间的容器将基于数据库分区组中最低编号的现有分区上相应表空间的容器。如果新分区与现有容器位于同一物理机器上，那么可能将导致容器之间发生命名冲突，倘若是这样，那么您不应使用此选项。反而，发出 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令之前，应将 ALTER DATABASE PARTITION GROUP 语句与 WITHOUT TABLESPACES 选项配合使用。然后，可以通过手动指定适当的名称来创建表空间容器。
2. 如果指定了 **ADD DBPARTITIONNUMS** 参数，那么数据重新分发可能会为所有新数据库分区创建表空间。

DROP

DBPARTITIONNUM *n*

TO *m*

n 或 *n* **TO** *m* 指定将从数据库分区组中删除的数据库分区编号列表。在数据库分区组中必须已定义所有指定的分区（SQLSTATE 42729）。这相当于执行指定了 DROP DBPARTITIONNUM 子句的 ALTER DATABASE PARTITION GROUP 语句。

DBPARTITIONNUMS *n*

TO *m*

n 或 *n* **TO** *m* 指定将从数据库分区组中删除的数据库分区编号列表。在数据库分区组中必须已定义所有指定的分区（SQLSTATE 42729）。这相当于执行指定了 DROP DBPARTITIONNUM 子句的 ALTER DATABASE PARTITION GROUP 语句。

TABLE *tablename*

为进行重新分发处理指定表顺序。

ONLY

如果表顺序后跟 **ONLY** 关键字（这是缺省值），那么将仅重新分发指定的表。余下的表稍后可由 **REDISTRIBUTE CONTINUE** 命令来处理。这是缺省值。

FIRST

如果表顺序后跟 **FIRST** 关键字，那么指定的表将采用给定顺序进行重新分发，而数据库分区组中余下的表将按照随机顺序重新分发。

EXCLUDE *tablename*

指定要在重新分发处理中省略的表。例如，可临时省略表直到您可配置它以满足数据重新分发的要求。省略的表稍后可由 **REDISTRIBUTE CONTINUE** 命令处理。

STOP AT *local-isotime*

当指定了此选项时，为每个表开始重新分发数据之前，会将 *local-isotime* 与当前的本地时间戳记比较。如果指定的 *local-isotime* 等于或早于当前的本地时间戳记，那么实用程序会停止，并且出现警告消息。正在进行的表数据重新分发处理在停止时将完成且不出现中断。不会开始任何新的表数据重新分发处理。可以使用 **CONTINUE** 选项来重新分发未处理的表。此 *local-isotime* 值可指定为时间戳记，即一个标识日期和时间组合的 7 部分字符串。格式为 *yyyy-mm-dd-hh.mm.ss.nnnnnn*（年、月、日、小时、分钟、秒和微秒），以本地时间表示。

DATA BUFFER *n*

指定要用作传送实用程序中数据的缓存空间的 4 KB 页数。仅当还指定了 **NOT ROLLFORWARD RECOVERABLE** 参数时，才能使用此命令参数。

如果所指定值小于最小受支持值，那么将使用最小受支持值，并不会返回警告。如果未指定 **DATA BUFFER** 值，那么在处理每个表的开始阶段实用程序将在运行时期间计算智能缺省值。特别是，缺省值将使用在开始重新分发表时实用程序堆中可用内存的 50%，并且也将各种表属性考虑在内。

此内存是直接从实用程序堆中分配的，可通过数据库配置参数 **util_heap_sz** 来修改此内存大小。如果系统中存在更多可用内存，那么 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令的 **DATA BUFFER** 参数值可以暂时超出 **util_heap_sz** 设置值。

INDEXING MODE

指定重新分发期间如何维护索引。仅当还指定了 **NOT ROLLFORWARD RECOVERABLE** 参数时，才能使用此命令参数。

有效值为:

REBUILD

将从头开始重建索引。索引不必有效即可使用此选项。使用此选项将导致索引页在磁盘上集群。

DEFERRED

重新分发将不会试图维护任何索引。索引将被标记为需要刷新。首次访问此类索引可能会强制执行重建，或者在重新启动数据库时可能会重建索引。

注: 对于非 MDC 表和非 ITC 表，如果表上存在无效的索引，那么 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令在您未指定 **INDEXING MODE DEFERRED** 的情况下将自动重建这些索引。对于 MDC 表或 ITC 表，由于实用程序需要组

合索引来处理 MDC 表或 ITC 表，所以即使您指定 **INDEXING MODE DEFERRED**，在开始重新分发表之前，仍会重建无效的组合索引。

PRECHECK
验证能否重新分发数据库分区组。仅当还指定了 **NOT ROLLFORWARD RECOVERABLE** 参数时，才能使用此命令参数。

YES
这是缺省值。仅当验证成功完成，重新分发操作才会开始。如果验证失败，那么该命令会终止并返回与第一次失败检查相关的错误消息。

NO 重新分发操作立即开始；不进行验证。

ONLY
此命令在执行验证后终止；不进行重新分发。缺省情况下，它不会停顿该数据库。如果 **QUIESCE DATABASE** 命令参数设置为 YES 或缺省为值 YES，那么数据库保持停顿。要复原与数据库的连接，请执行重新分发操作或发出 **UNQUIESCE DATABASE** 命令。

QUIESCE DATABASE
指定此项以强制所有用户离开该数据库并使该数据库进入停顿状态。仅当还指定了 **NOT ROLLFORWARD RECOVERABLE** 参数时，才能使用此命令参数。

YES
这是缺省值。只有带有 SYSADM、SYSMAINT 或 SYSCTRL 权限或已被授予 QUIESCE_CONNECT 权限的用户才能访问数据库或其对象。一旦重新分发成功完成，数据库就会取消停顿。

NO 重新分发操作不会停顿数据库；不会强制用户离开该数据库。

有关更多信息，请参阅 **QUIESCE DATABASE** 命令。

STATISTICS
指定实用程序应该收集具有统计信息概要文件的表的统计信息。仅当还指定了 **NOT ROLLFORWARD RECOVERABLE** 参数时，才能使用此命令参数。

在完成数据重新分发之后，指定此选项要比单独发出 **RUNSTATS** 命令更为高效率。

USE PROFILE
将为具有统计信息概要文件的表收集统计信息。对于没有统计信息概要文件的表，将不执行任何操作。这是缺省值。

NONE
将不为表收集统计信息。

示例

通过数据分布文件 distfile_for_dbpg_1 提供当前数据分布，对数据库分区组 DBPG_1 进行重新分发。将该数据移动到两个新数据库分区 6 和 7 上。

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_1
   USING DISTFILE /home/user1/data/distfile_for_dbpg_1
   ADD DATABASE PARTITION (6 TO 7)
```

重新分发数据库分区组 DBPG_2 以便:

- 重新分发是不可前滚恢复;
- 以统一方式在散列分区间分发数据;

- 从头开始重建索引;
- 不收集统计信息;
- 将 180000 个 4 KB 页用作传输数据的缓存空间。

```
REDISTRIBUTE DATABASE PARTITION GROUP DBPG_2
 NOT ROLLFORWARD RECOVERABLE
 UNIFORM
 INDEXING MODE REBUILD
 DATA BUFFER 180000
 STATISTICS NONE
```

此重新分发操作还会因为 **QUIESCE DATABASE** 和 **PRECHECK** 命令参数的缺省值而停顿数据库并执行预先检查。

使用说明

- 启动重新分发操作之前，请确保表处于正常状态而未处于“装入暂挂”状态或“重组暂挂”状态。可以使用 **LOAD QUERY** 命令来检查表状态。
- 如果指定了 **NOT ROLLFORWARD RECOVERABLE** 选项并且数据库是可恢复数据库，那么实用程序首次访问表空间时，它被置于“备份暂挂”状态。该表空间中的所有表将变为只读，一直到表空间被备份为止，只有在表空间中的所有表重新分发完毕时，您才可以执行该备份操作。
- 重新分发操作正在运行时，它会生成事件日志文件，其中包含了有关重新分发操作的一般信息以及所处理的每个表的开始与结束时间等信息。此事件日志文件被写入:
  - Linux 和 UNIX 操作系统上的 homeinst/sqllib/redist 目录（对子目录和文件名使用以下格式: *database-name.database-partition-group-name.timestamp*.log）。
  - Windows 操作系统上的 **DB2INSTPROF***instance*\redist 目录（其中 **DB2INSTPROF** 是 **DB2INSTPROF** 注册表变量的值）（对子目录和文件名使用以下格式: *database-name.database-partition-group-name.timestamp*.log）。
  - 时间戳记值是发出命令时的时间。
- 此实用程序在处理期间执行间歇性 COMMIT 命令。
- 使依赖正在进行重新分发的表的所有程序包无效。在完成重新分发数据库分区组操作之后，建议您以显式方式重新绑定此类程序包。以显式方式重新绑定可避免为无效包执行第一个 SQL 请求时的初始延迟。重新分发消息文件包含正在进行重新分发的所有表的列表。
- 缺省情况下，重新分发实用程序将更新具有统计信息概要文件的那些表的统计信息。对于没有统计信息概要文件的表，建议在完成重新分发操作后通过调用 db2Runstats API 或发出 **RUNSTATS** 命令，为这些表单独更新表和索引统计信息。
- 您无法重新分发包含复制的具体化查询表或使用 DATA CAPTURE CHANGES 定义的表的数据库分区组。
- 如果数据库分区组中存在具有现有已声明临时表或创建临时表的用户临时表空间，那么不允许执行重新分发操作。
- 在表上将忽略诸如 **INDEXING MODE** 之类不适用的选项，而不会出现警告。例如，在没有索引的表上将忽略 **INDEXING MODE**。

- 如果“添加数据库分区服务器”请求暂挂或正在进行，那么 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令可能失败（SQLSTATE 55071）。如果新数据库分区服务器是以联机方式添加至实例，并且并非所有应用程序都知道该新数据库分区服务器，那么此命令也可能失败（SQLSTATE 55077）。

兼容性

如果表包含的 XML 列使用 DB2 版本 9.5 或之前版本的 XML 记录格式，那么不能对这些表执行重新分发操作。使用 ADMIN_MOVE_TABLE 存储过程将该表迁移至新格式。

db2nchg - 更改数据库分区服务器配置

修改数据库分区服务器配置。这包括将数据库分区服务器从一台机器移动到另一台机器，更改机器的 TCP/IP 主机名，以及为该数据库分区服务器选择另一个逻辑端口号或另一个网络名。

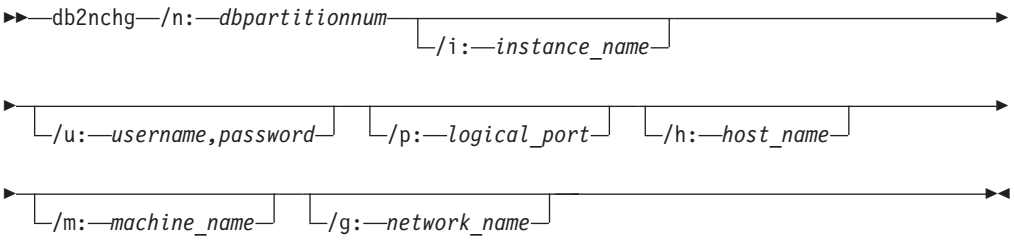
此命令仅在停止数据库分区服务器时才会使用。

此命令仅在 Windows 操作系统上可用。

权限

本地管理员

命令语法



命令参数

/n:*dbpartitionnum*

指定将要更改的数据库分区服务器配置的数据库分区号。

/i:*instance_name*

指定此数据库分区服务器所参与的实例。如果未指定参数，那么缺省值是当前实例。

/u:*username,password*

指定用户名和密码。如果未指定参数，那么将应用现有用户名和密码。

/p:*logical_port*

指定数据库分区服务器的逻辑端口。您必须指定此参数以便将数据库分区服务器移至另一机器。如果未指定参数，那么逻辑端口号将保持不变。

/h:*host_name*

指定由 FCM 用于内部通信的 TCP/IP 主机名。如果未指定此参数，那么主机名将保留不变。

/m:*machine_name*

指定数据库分区服务器将驻留的机器。仅当实例中没有现有数据库时，才可移动数据库分区服务器。

/g:*network_name*

更改数据库分区服务器的网络名。当机器上存在多个 IP 地址时，此参数可用于将特定 IP 地址应用于数据库分区服务器。您可以输入网络名或 IP 地址。

示例

要将分配给数据库分区 2（它参与实例 TESTMPP）的逻辑端口更改为逻辑端口 3，请输入下列命令:

```
db2nchg /n:2 /i:TESTMPP /p:3
```

db2ncrt - 将数据库分区服务器添加至实例

将数据库分区服务器添加至实例。

此命令仅在 Windows 操作系统上可用。

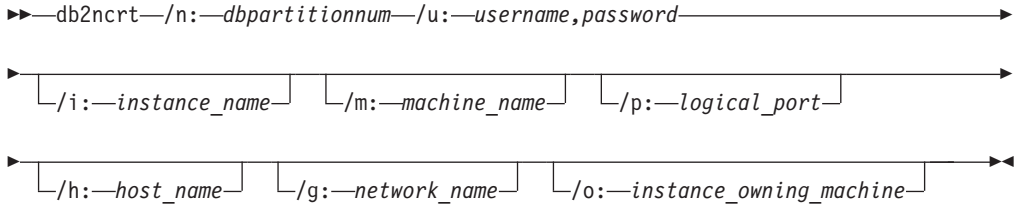
作用域

如果数据库分区服务器被添加至实例已经存在的计算机，那么数据库分区服务器将作为逻辑数据库分区服务器添加至计算机。如果数据库分区服务器被添加至不存在实例的计算机，那么将会添加实例，并且计算机变成新的物理数据库分区服务器。如果实例中有数据库，那么不应该使用此命令。而是应该发出带有 **ADD DBPARTITIONNUM** 选项的 **START DATABASE MANAGER** 命令。这确保可正确地将该数据库 添加至新的数据库分区服务器。您也可将数据库分区服务器添加至已创建数据库的实例。由于更改 db2nodes.cfg 文件可能导致分区数据库环境中出现不一致情况，所以不应该编辑该文件。

权限

对添加了新数据库分区服务器的计算机的本地管理员权限。

命令语法



命令参数

/n:*dbpartitionnum*

用于标识数据库分区服务器的唯一数据库分区号。所输入的编号范围在 1 至 999 之间。

/u:*username,password*

指定 DB2 的登录帐户名称和密码。

/i:*instance_name*

指定实例名。如果未指定参数，那么缺省值是当前实例。

/m:*machine_name*

指定数据库分区服务器驻留的 Windows 工作站的计算机名。如果在远程计算机上添加数据库分区服务器，那么此参数是必需的。

/p:*logical_port*

指定用于数据库分区服务器的逻辑端口号。如果未指定此参数，那么所分配的逻辑端口号将为 0。当创建逻辑数据库分区服务器时，必须指定此参数，并且必须选择未在使用中的逻辑端口号。但是，应注意下列限制:

- 每台计算机必须具有逻辑端口为 0 的数据库分区服务器。
- 端口号不能超过在 x:\winnt\system32\drivers\etc\ 目录中为 FCM 通信保留的端口范围。例如，如果为当前实例保留 4 个端口的范围，那么最大端口号为 3。端口 0 用于缺省逻辑数据库分区服务器。

/h:*host_name*

指定由 FCM 用于内部通信的 TCP/IP 主机名。当在远程计算机上添加数据库分区服务器时，此参数是必需的。

/g:*network_name*

指定数据库分区服务器的网络名。如果未指定参数，那么将使用系统上检测到的第一个 IP 地址。当计算机上存在多个 IP 地址时，此参数可用于将特定 IP 地址应用于数据库分区服务器。您可以输入网络名或 IP 地址。

/o:*instance_owning_machine*

指定实例拥有的计算机的计算机名。缺省值是本地计算机。当在任何非拥有实例的计算机上调用 **db2ncrt** 命令时，此参数是必须的。

示例

要将新的数据库分区服务器添加至实例拥有的计算机 SHAYER 上的实例 TESTMPP（其中新数据库分区服务器被称为数据库分区 2 且使用逻辑端口 1），请输入下列命令:

```
db2ncrt /n:2 /u:QBPAULZ\paulz,g1reeky /i:TESTMPP /m:TEST /p:1 /o:SHAYER /h:TEST
```

db2ndrop - 从实例中删除数据库分区服务器

从没有数据库的实例中删除数据库分区服务器。如果数据库分区服务器已删除，那么其数据库分区号可重新用于新的数据库分区服务器。

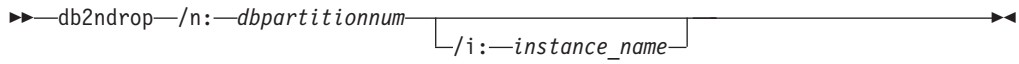
此命令仅在停止数据库分区服务器时才会使用。

此命令仅在 Windows 操作系统上可用。

权限

对将要删除数据库分区服务器的机器的本地管理员权限。

命令语法



命令参数

/n:*dbpartitionnum*
用于标识数据库分区服务器的唯一数据库分区号。

/i:*instance_name*
指定实例名。如果未指定参数，那么缺省值是当前实例。

示例

```
db2ndrop /n:2 /i=KMASCI
```

使用说明

如果从实例中删除拥有实例的数据库分区服务器（dbpartitionnum 0），那么该实例将变得不可用。要删除实例，请使用 **db2idrop** 命令。

如果在此实例中有数据库，那么不应该使用此命令。而应使用 **db2stop drop dbpartitionnum** 命令。这确保正确地从分区数据库环境中除去数据库分区服务器。您也可删除数据库存在于其中的实例中的数据库分区服务器。由于更改 db2nodes.cfg 文件可能导致分区数据库环境中出现不一致情况，所以不应该编辑该文件。

要从正在运行多个逻辑数据库分区服务器的机器中删除已分配给逻辑端口 0 的数据库分区服务器，必须先删除分配给其他逻辑端口的所有其他数据库分区服务器。每个数据库分区服务器均必须具有已分配给逻辑端口 0 的数据库分区服务器。

第 31 章 SQL 语言元素

数据类型

与数据库分区兼容的数据类型

在分发键的对应列的基本数据类型之间，定义了*数据库分区兼容性*。与数据库分区兼容的数据类型具有以下属性：两个变量分别属于两种类型，它们具有相同的值，由同一数据库分区功能映射至同一分发映射索引。

第 370 页的表 42 显示数据库分区中数据类型的兼容性。

数据库分区兼容性具有下列特征:

- 内部格式用于 DATE、TIME 和 TIMESTAMP。它们彼此都不兼容，且与字符或图形数据类型不兼容。
- 分区兼容性不受列的可空性影响。
- 分区兼容性受整理影响。除了忽略整理的强度（S）属性，区分语言环境的基于 UCA 的整理在整理时需要完全匹配。为确定分区兼容性，所有其他整理被视为等价。
- 仅当使用的整理并非区分语言环境的基于 UCA 的整理时，使用 FOR BIT DATA 定义的字符列才与未使用 FOR BIT DATA 定义的字符列兼容。
- 可兼容数据类型的空值以相同方式处理。可能对不可兼容数据类型的空值生成不同结果。
- UDT 的基本数据类型可用于分析数据库分区兼容性。
- 对分发键中值相同的时间戳记的处理完全相同，即使它们的时间戳记精度不同也是如此。
- 对分布键中值相同的小数的处理是完全相同的，即使它们的小数位和精度不同也是如此。
- 字符串（CHAR、VARCHAR、GRAPHIC 或 VARGRAPHIC）中的尾部空格会被系统提供的散列法功能忽略。
- 使用区分语言环境的基于 UCA 的整理时，CHAR、VARCHAR、GRAPHIC 和 VARGRAPHIC 是兼容的数据类型。当使用其他整理时，CHAR 和 VARCHAR 是兼容类型，而 GRAPHIC 和 VARGRAPHIC 是兼容类型，但 CHAR 和 VARCHAR 与 GRAPHIC 和 VARGRAPHIC 不是兼容类型。不同长度的 CHAR 或 VARCHAR 是兼容的数据类型。
- 即使精度不同，但相等的 DECFLOAT 值仍然会被同等对待。即使具有不同的有效数字数目，但数字相等的 DECFLOAT 值仍然会被同等对待。
- 对于数据库分区兼容性，作为分发键的一部分不受支持的数据类型不适用。这包括其数据类型为 BLOB、CLOB、DBCLOB、XML、基于任何这些数据类型的单值类型以及结构化类型的列。

表 42. 数据库分区兼容性

操作数	二进制整数	十进制数字	浮点	十进制浮点	字符串	图形字符串	日期	时间	时间戳记	单值类型
二进制整数	是	否	否	否	否	否	否	否	否	[1]
十进制数字	否	是	否	否	否	否	否	否	否	[1]
浮点	否	否	是	否	否	否	否	否	否	[1]
十进制浮点	否	否	否	是	否	否	否	否	否	[1]
字符串	否	否	否	否	是[2]	2 和 3	否	否	否	[1]
图形字符串	否	否	否	否	2 和 3	是[2]	否	否	否	[1]
日期	否	否	否	否	否	否	是	否	否	[1]
时间	否	否	否	否	否	否	否	是	否	[1]
时间戳记	否	否	否	否	否	否	否	否	是	[1]
单值类型	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]	[1]

注:

[1] 单值类型值是与单值类型的源数据类型或具有同一源数据类型的任何其他单值类型兼容的数据库分区。单值类型的源数据类型必须是作为分发键的一部分受支持的数据类型。用户定义的单值类型（UDT）值是与源类型的 UDT 或任何其他拥有数据库分区兼容源类型的 UDT 相兼容的数据库分区。单值类型不能基于 BLOB、CLOB、DBCLOB 或 XML。

[2] 当字符和图形字符串类型具有兼容的整理时它们兼容。

[3] 当与语言环境相关的基于 UCA 的整理生效时字符和图形字符串类型兼容。否则，它们不是兼容类型。

专用寄存器

CURRENT MEMBER

CURRENT MEMBER 专用寄存器指定为该语句标识协调程序 成员 的 INTEGER 值。

对于从应用程序发出的语句，协调程序是应用程序连接至的 成员。对于从例程发出的语句，协调程序是从中调用例程的 成员。

当在例程中的 SQL 语句中使用时，CURRENT MEMBER 永远不会继承自调用语句。

如果未将数据库实例定义为支持数据库分区或 IBM DB2 pureScale Feature，那么 CURRENT MEMBER 返回 0。如果没有 db2nodes.cfg 文件，那么不会将该数据库实例定义为支持这些环境。对于分区数据库或 DB2 pureScale 环境，db2nodes.cfg 文件存在，且包含数据库分区和 成员 定义。

可以通过 CONNECT 语句来更改 CURRENT MEMBER，但这仅限于在特定情况下执行。

为了与 DB2 的先前版本及其他数据库产品兼容，可以指定 NODE 来代替 MEMBER。

示例

示例 1: 将主机变量 APPL_NODE（整数）设置为应用程序连接至的 成员 的编号。

```
VALUES CURRENT MEMBER
  INTO :APPL_NODE
```

示例 2: 在分区数据库环境中的成员 0 和成员系统 4 上发出以下命令。此查询将检索当前连接的数据库成员编号。

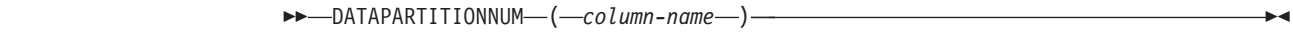
```
db2 "values current member"

1
-----------
          0
```

第 32 章 SQL 函数

DATAPARTITIONNUM

DATAPARTITIONNUM 函数返回该行驻留在其中的数据分区的序号（SYSDATAPARTITIONS.SEQNO）。



模式为 SYSIBM。

column-name

表中任何列的标准名称或非标准名称。由于返回行级别信息，所以无论指定哪一列，结果都是相同的。该列可以具有任何数据类型。

如果 *column-name* 引用视图中的列，那么该列的表达式必须引用底层的基本表的列，并且视图必须是可删除的。嵌套的或公共表表达式与视图一样遵循相同的规则。

数据分区按范围排序，序号从 0 开始。例如，DATAPARTITIONNUM 函数为具有最低范围的数据分区中的行返回 0。

结果的数据类型是 INTEGER 且永不为空。

注意

- 当创建用户定义的函数时，此函数不能用作源函数。因为该函数可接受任何数据类型作为参数，所以不必创建其他特征符来支持用户定义的单值类型。
- 不能在检查约束或生成列的定义中使用 DATAPARTITIONNUM 函数（SQLSTATE 42881）。不能在具体化查询表 (MQT) 定义中使用 DATAPARTITIONNUM 函数（SQLSTATE 428EC）。
- DATAPARTITIONNUM 函数不能作为 CREATE INDEX 语句中基于表达式的关键字的一部分使用。

示例

- *示例 1:* 检索 EMPLOYEE.EMPNO 行所在的数据分区的序号。

```
SELECT DATAPARTITIONNUM (EMPNO)
FROM EMPLOYEE
```

- *示例 2:* 要将 DATAPARTITIONNUM 返回的序号（例如 0）转换为可在其他 SQL 语句（例如，ALTER TABLE...DETACH PARTITION）中使用的数据分区名称，您可以查询 SYSCAT.DATAPARTITIONS 目录视图。包括从 WHERE 子句中的 DATAPARTITIONNUM 获取的 SEQNO，如下列示例中所示。

```
SELECT DATAPARTITIONNAME
   FROM SYSCAT.DATAPARTITIONS
   WHERE TABNAME = 'EMPLOYEE' AND SEQNO = 0
```

导致值 'PART0'。

DBPARTITIONNUM

DBPARTITIONNUM 函数为行返回数据库分区号。例如，如果在 SELECT 子句中使用，那么它会为结果集中的每个行返回数据库分区号。

模式为 SYSIBM。

column-name

表中任何列的标准名称或非标准名称。由于返回行级别信息，所以无论指定哪一列，结果都是相同的。该列可以具有任何数据类型。

如果 *column-name* 引用视图中的列，那么该列的表达式必须引用底层的基本表的列，并且视图必须是可删除的。嵌套的或公共表表达式与视图一样遵循相同的规则。

DBPARTITIONNUM 函数为特定行（和表）返回数据库分区号，该行（和表）由使用该函数的 SQL 语句的上下文来确定。

在转换变量和表上返回的数据库分区号从分发键列的当前转换值得来。例如，在“插入前”触发器中，该函数根据新转换变量的当前值返回映射的数据库分区号。然而，分发键列的值可能被后续的“插入前”触发器修改。因此，在将行插入数据库后，该行的最终数据库分区号可能与映射的值不同。

结果的数据类型是 INTEGER 且永不为空。如果没有 db2nodes.cfg 文件，那么结果为 0。

注意

- 不能在复制的表上、检查约束中或生成列的定义中使用 DBPARTITIONNUM 函数（SQLSTATE 42881）。
- 当创建用户定义的函数时，此 DBPARTITIONNUM 函数不能用作源函数。因为该函数可接受任何数据类型作为参数，所以不必创建其他特征符来支持用户定义的单值类型。
- DBPARTITIONNUM 函数不能作为 CREATE INDEX 语句中基于表达式的关键字的一部分。
- ***语法替换***：为了与 DB2 产品的先前版本兼容，函数名 NODENUMBER 是 DBPARTITIONNUM 的同义词。

示例

- *示例 1*：计算实例的数目，其中 EMPLOYEE 表中给定职员的行位于与 DEPARTMENT 表中职员部门的描述不同的数据库分区上。

```
SELECT COUNT(*) FROM DEPARTMENT D, EMPLOYEE E
  WHERE D.DEPTNO=E.WORKDEPT
  AND DBPARTITIONNUM(E.LASTNAME) <> DBPARTITIONNUM(D.DEPTNO)
```

- *示例 2*：连接 EMPLOYEE 表和 DEPARTMENT 表，以便两个表的行均在同一数据库分区上。

```
SELECT * FROM DEPARTMENT D, EMPLOYEE E
  WHERE DBPARTITIONNUM(E.LASTNAME) = DBPARTITIONNUM(D.DEPTNO)
```

- *示例 3*: 在 EMPLOYEE 表上使用前触发器，在名为 EMPINSERTLOG1 的表中记录职员编号和 EMPLOYEE 表中任何新行的映射的数据库分区号。

```
CREATE TRIGGER EMPINSLOGTRIG1
BEFORE INSERT ON EMPLOYEE
REFERENCING NEW AW NEWTABLE
FOR EACH ROW
INSERT INTO EMPINSERTLOG1
VALUES(NEWTABLE.EMPNO, DBPARTITIONNUM
(NEWTABLE.EMPNO))
```

第 33 章 SQL 语句

ALTER DATABASE PARTITION GROUP

使用 ALTER DATABASE PARTITION GROUP 语句可将一个或多个数据库分区添加到数据分区组中，也可以从数据库分区组删除一个或多个数据库分区。

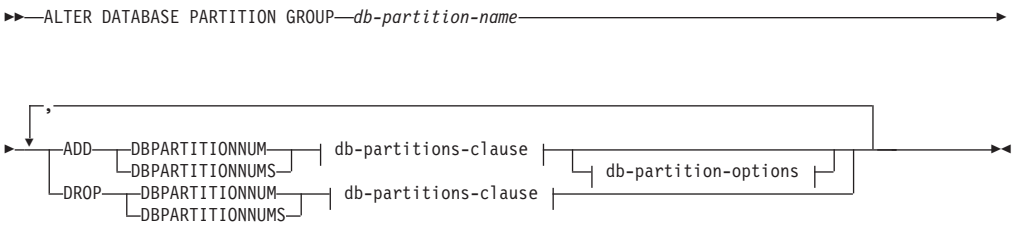
调用

此语句可嵌入应用程序中或者以交互方式发出。它是一个可执行语句，仅当 DYNAMICRULES 运行行为对于程序包有效时才能动态编译该语句（SQLSTATE 42509）。

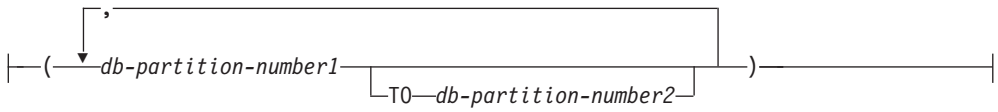
权限

语句的授权标识必须具有 SYSCTRL 或 SYSADM 权限。

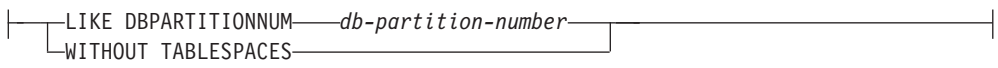
语法



db-partitions-clause:



db-partition-options:



描述

db-partition-name

为数据库分区组命名。这是单一部分名称。它是普通或定界 SQL 标识。它必须是目录中描述的数据库分区组。您不能指定 IBMCATGROUP 与 IBMTEMPGROUP（SQLSTATE 42832）。

ADD DBPARTITIONNUM
指定要添加至数据库分区组的特定数据库分区。DBPARTITIONNUMS 是 DBPARTITIONNUM 的同义词。在数据库分区组中，任何指定的数据库分区均必须尚未被定义（SQLSTATE 42728）。

DROP DBPARTITIONNUM
指定要从数据库分区组中删除的特定数据库分区。DBPARTITIONNUMS 是 DBPARTITIONNUM 的同义词。在数据库分区组中必须已定义所有指定的数据库分区（SQLSTATE 42729）。

db-partitions-clause
指定要添加或删除的数据库分区。

db-partition-number1
指定特定的数据库分区号。

TO *db-partition-number2*
指定数据库分区号的范围。*db-partition-number2* 的值必须大于或等于 *db-partition-number1* 的值（SQLSTATE 428A9）。

db-partition-options

LIKE DBPARTITIONNUM *db-partition-number*
指定数据库分区组中现有表空间的容器将与指定的 *db-partition-number* 上的容器相同。指定的数据库分区必须是在此语句之前就存在于数据库分区组中且未包括在同一语句的 DROP DBPARTITIONNUM 子句中的分区。

对于定义为使用自动存储器的表空间，即使用 CREATE TABLESPACE 语句的 MANAGED BY AUTOMATIC STORAGE 子句创建的表空间，或根本未为其指定 MANAGED BY 子句的表空间，容器将不必与来自指定分区的容器相匹配。数据库管理器将根据与数据库相关联的存储路径来自动指定容器，这可能会也可能不会导致使用同一个容器。每个表空间的大小基于创建表空间时指定的初始大小，它可能与指定分区上的表空间的当前大小不匹配。

WITHOUT TABLESPACES
指定不在新添加的数据库分区或分区上创建数据库分区组中现有表空间的容器。使用 *db-partitions-clause* 或 MANAGED BY AUTOMATIC STORAGE 子句的 ALTER TABLESPACE 语句必须用于定义要与在此数据库分区组上定义的表空间配合使用的容器。 如果未指定此选项，那么在新添加的数据库分区上为在数据库分区组上定义的每个表空间指定缺省容器。

对于已定义为使用自动存储器的表空间，即使用 CREATE TABLESPACE 语句的 MANAGED BY AUTOMATIC STORAGE 子句创建的表空间，或根本未为其指定 MANAGED BY 子句的表空间，将忽略此选项。没有办法来推迟为这些表空间创建容器。数据库管理器将根据与数据库相关联的存储路径来自动指定容器。每个表空间的大小将基于创建表空间时指定的初始大小。

规则

- 必须在 db2nodes.cfg 文件中定义由编号指定的每个数据库分区（SQLSTATE 42729）。
- *db-partitions-clause* 中列示的每个 *db-partition-number* 必须仅用于唯一的数据库分区（SQLSTATE 42728）。

- 有效的数据库分区号介于 0 和 999 之间（其中包括 0 和 999）（SQLSTATE 42729）。
- 数据库分区不能出现在 ADD 和 DROP 子句中（SQLSTATE 42728）。
- 数据库分区组中必须至少保留一个数据库分区。您不能从数据库分区组中删除最后一个数据库分区（SQLSTATE 428C0）。
- 如果在添加数据库分区时未指定 LIKE DBPARTITIONNUM 子句或 WITHOUT TABLESPACES 子句，那么缺省情况下将使用数据库分区组中现有数据库分区的最小数据库分区号（例如 *2*），然后继续进行操作，就好像已指定了 LIKE DBPARTITIONNUM *2* 一样。对于要用作缺省值的现有数据库分区，它必须具有为数据库分区组中的所有表空间定义的容器（SYSCAT.DBPARTITIONGROUPDEF 的列 IN_USE 不是“T”）。
- 如果“添加数据库分区服务器”请求暂挂或正在进行，那么 ALTER DATABASE PARTITION GROUP 语句可能失败（SQLSTATE 55071）。如果新数据库分区服务器是以联机方式添加至实例，并且并非所有应用程序都知道该新数据库分区服务器，那么此语句也可能失败（SQLSTATE 55077）。

注意

- 当数据库分区被添加至数据库分区组时，将为数据库分区建立目录条目（请参阅 SYSCAT.DBPARTITIONGROUPDEF）。分发映射会立即更改为包括新的数据库分区，同时出现指示器（IN_USE），表明在下列情况下数据库分区位于分发映射中:
  - 数据库分区组中未定义表空间，或者
  - 数据库分区组中定义的表空间中未定义表，并且未指定 WITHOUT TABLESPACES 子句。

  分发映射未更改，指示器（IN_USE）已设置为指示在下列任何一种情况下分发映射中未包括数据库分区:
  - 表存在于数据库分区组中的表空间中，或者
  - 表空间存在于数据库分区组中，并且 WITHOUT TABLESPACES 子句已被指定（除非所有表空间已定义为使用自动存储器，在此情况下，将忽略 WITHOUT TABLESPACES 子句）

  要更改分发映射，必须使用 REDISTRIBUTE DATABASE PARTITION GROUP 命令。这将重新分发数据、更改分发映射和更改指示器。如果指定了 WITHOUT TABLESPACES 子句，那么在试图重新分发数据之前，需要添加表空间容器。
- 当从数据库分区组中删除数据库分区时，将会更新该数据库分区的目录条目（请参阅 SYSCAT.DBPARTITIONGROUPDEF）。如果在数据库分区组中定义的表空间中未定义表，那么分发映射将立即更改为排除已删除的数据库分区，并且会删除该数据库分区组中数据库分区的条目。如果表存在，那么不会更改分发映射，并且指示器（IN_USE）被设置为指示数据库分区正在等待被删除。REDISTRIBUTE DATABASE PARTITION GROUP 命令必须用于重新分发数据，并且从数据库分区组中删除数据库分区的条目。
- ***语法替换***: 与先前版本的 DB2 及其他数据库产品的兼容性支持下列语法替换。这些替代语法并非标准语法，不应使用。
  - 可指定 NODE 来代替 DBPARTITIONNUM
  - 可指定 NODES 来代替 DBPARTITIONNUMS
  - 可指定 NODEGROUP 来代替 DATABASE PARTITION GROUP

示例

假设您具有的六分区数据库拥有下列数据库分区：0、1、2、5、7 和 8。两个数据库分区（3 和 6）被添加至系统。

- *示例 1*：假设您想要将数据库分区 3 和 6 添加至名为 MAXGROUP 的数据库分区组，并且具有表空间容器（如数据库分区 2 上的表空间容器）。该语句如下所示：

```
ALTER DATABASE PARTITION GROUP MAXGROUP
  ADD DBPARTITIONNUMS (3,6)LIKE DBPARTITIONNUM 2
```

- *示例 2*：假设您想要删除数据库分区 1，并且要将数据库分区 6 添加至数据库分区组 MEDGROUP。您将使用 ALTER TABLESPACE 为数据库分区 6 单独定义表空间容器。该语句如下所示：

```
ALTER DATABASE PARTITION GROUP MEDGROUP
  ADD DBPARTITIONNUM(6)WITHOUT TABLESPACES
  DROP DBPARTITIONNUM(1)
```

CREATE DATABASE PARTITION GROUP

CREATE DATABASE PARTITION GROUP 语句定义数据库中的新数据库分区组，将数据库分区指定给数据库分区组，以及在系统目录中记录数据库分区组定义。

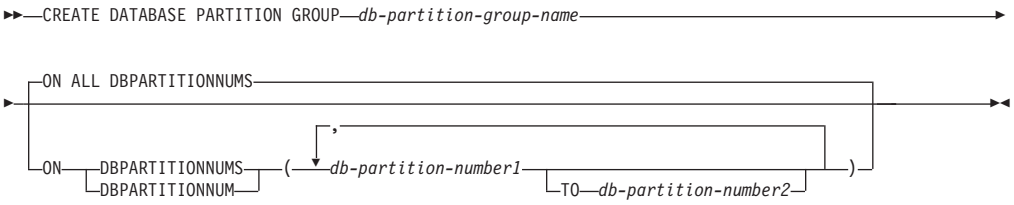
调用

此语句可嵌入应用程序中或者以交互方式发出。它是一个可执行语句，仅当 DYNAMICRULES 运行行为对于程序包有效时才能动态编译该语句（SQLSTATE 42509）。

权限

语句的授权标识所拥有的特权必须包括 SYSCTRL 或 SYSADM 权限。

语法



描述

db-partition-group-name

为数据库分区组命名。这是单一部分名称。它是普通或定界 SQL 标识。*db-partition-group-name* 必须未标识已存在于目录中的数据库分区组（SQLSTATE 42710）。*db-partition-group-name* 不得以字符“SYS”或“IBM”开头（SQLSTATE 42939）。

ON ALL DBPARTITIONNUMS

指定当创建数据库分区组时在定义给数据库的所有数据库分区上定义数据库分区组（db2nodes.cfg 文件）。

如果将数据库分区添加至数据库系统，那么应该发出 ALTER DATABASE PARTITION GROUP 语句，以将这个新的数据库分区包括在数据库分区组（包括

IBMDEFAULTGROUP）。此外，必须发出 REDISTRIBUTE DATABASE PARTITION GROUP 命令，以将数据移至数据库分区。

ON DBPARTITIONNUMS
指定位于数据库分区组中的数据库分区。DBPARTITIONNUM 是 DBPARTITIONNUMS 的同义词。

db-partition-number1
指定数据库分区号。（可以指定格式 NODE*nnnnn* 的 *node-name*，以便与先前版本兼容。）

TO *db-partition-number2*
指定数据库分区号的范围。*db-partition-number2* 的值必须大于或等于 *db-partition-number1* 的值（SQLSTATE 428A9）。介于指定的数据库分区号之间且包括这些分区号的所有数据库分区均包含在数据库分区组中。

规则

- 必须在 db2nodes.cfg 文件中定义由编号指定的每个数据库分区（SQLSTATE 42729）。
- ON DBPARTITIONNUMS 子句中列示的每个 *db-partition-number* 均必须最多仅出现一次 (SQLSTATE 42728)。
- 有效的 *db-partition-number* 介于 0 和 999 之间（包括 0 和 999）（SQLSTATE 42729）。
- 如果“添加数据库分区服务器”请求暂挂或正在进行，那么 CREATE DATABASE PARTITION GROUP 语句可能失败（SQLSTATE 55071）。如果新数据库分区服务器是以联机方式添加至实例，并且并非所有应用程序都知道该新数据库分区服务器，那么此语句也可能失败（SQLSTATE 55077）。

注意

- 此语句为数据库分区组创建分发映射。为每个分发映射生成了分发映射标识（PMAP_ID）。此信息记录在目录中，并且可以从 SYSCAT.DBPARTITIONGROUPS 和 SYSCAT.PARTITIONMAPS 中检索。分发映射中的每个条目指定所有散列的行驻留的目标数据库分区。对于单一分区数据库分区组，相应的分发映射只有一个条目。对于多分区数据库分区组，相应的分发映射具有 32768 个条目，其中数据库分区号在缺省情况下按循环法被分配给该图的条目。
- ***语法替换***: 与先前版本的 DB2 及其他数据库产品的兼容性支持下列语法替换。这些替代语法并非标准语法，不应使用。
  - 可指定 NODE 来代替 DBPARTITIONNUM
  - 可指定 NODES 来代替 DBPARTITIONNUMS
  - 可指定 NODEGROUP 来代替 DATABASE PARTITION GROUP

示例

下列示例基于分区数据库拥有定义为 0、1、2、5、7 和 8 的六个数据库分区。

- *示例 1*: 假设您想要在所有六个数据库分区上创建名为 MAXGROUP 的数据库分区组。该语句如下所示:

```
CREATE DATABASE PARTITION GROUP MAXGROUP ON ALL DBPARTITIONNUMS
```

- *示例 2*: 假设您想要在数据库分区 0、1、2、5 和 8 上创建名为 MEDGROUP 的数据库分区组。该语句如下所示:

```
CREATE DATABASE PARTITION GROUP MEDGROUP
  ON DBPARTITIONNUMS( 0 TO 2, 5, 8)
```

- *示例 3*: 假设您想要在数据库分区 7 上创建单一分区数据库分区组 MINGROUP。该语句如下所示:

```
CREATE DATABASE PARTITION GROUP MINGROUP
  ON DBPARTITIONNUM (7)
```

第 34 章 受支持的 SQL 管理例程和视图

ADMIN_CMD 存储过程和关联的管理 SQL 例程

使用 ADMIN_CMD 过程的 GET STMM TUNING 命令

用于读取目录表以报告用户首选的自调整内存管理器 (STMM) 调整成员编号和当前 STMM 调整成员编号。

权限

语句授权标识拥有的特权必须至少包括以下其中一项权限或特权:

- DBADM
- SECADM
- SQLADM
- ACCESSCTRL
- DATAACCESS
- 对 SYSIBM.SYSTUNINGINFO 的 SELECT 权限或特权

必需的连接

数据库

命令语法

示例

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

下列是从此查询输出的示例。

```
Result set 1
  --------------

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

  1 record(s) selected.

  Return Status = 0
```

使用说明

- 用户首选的自调整内存管理器 (STMM) 调整成员编号 (USER_PREFERRED_NUMBER) 由用户设置，它指定用户想要在其上运行内存调整器的成员。当数据库正在运行时，每个小时将应用几次调整成员。结果，返回的 CURRENT_NUMBER 和

USER_PREFERRED_NUMBER 可能会在您更新用户首选的 STMM 成员之后处于不同步状态。要解决此问题，等待异步更新 CURRENT_NUMBER，或者停止和启动数据库以强制更新 CURRENT_NUMBER。

兼容性

要获得与先前版本的兼容性:

- **DBPARTITIONNUM** 可替代 **MEMBER**，除非将 **DB2_ENFORCE_MEMBER_SYNTAX** 注册表变量设置为 ON。

通过使用 ADMIN_CMD 过程执行的 UPDATE STMM TUNING 命令

更新在其中创建自调整内存功能管理器 (STMM) 的用户首选数据库的成员编号。

权限

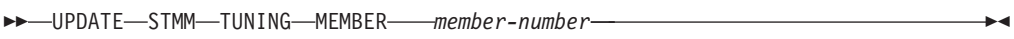
语句授权标识拥有的特权必须至少包括以下其中一项权限:

- DBADM
- DATAACCESS
- SQLADM

必需的连接

数据库

命令语法



命令参数

member-number

member-number 是一个整数。

在分区数据库环境中:

- 如果指定了有效成员编号，那么 DB2 数据库服务器在该成员上运行 STMM 内存调整器。
- 如果指定了 -1 或者指定的成员编号不存在，那么 DB2 数据库服务器将选择要在其上运行 STMM 内存调整器的相应成员。

在 DB2 pureScale 环境中:

- 如果指定了有效的成员编号，那么 DB2 数据库服务器将在该成员上运行 STMM 内存调整器。
- 如果指定了 -2，那么 DB2 数据库服务器将启用 STMM 调整器以在每个成员上单独运行和调整。
- 如果指定了 -1 或者指定的成员编号不存在，那么 DB2 数据库服务器将选择要在其上运行 STMM 内存调整器的相应成员。

示例

在分区数据库环境中，更新成员 3 的用户首选自调整内存管理器 (STMM) 调整成员编号。

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

使用说明

- STMM 调整进程定期检查用户首选的 STMM 调整成员数值的变化。如果 *member-number* 存在且为活动的成员，那么 STMM 调整进程将移至用户首选 STMM 调整成员。一旦此命令更改 STMM 调整成员编号，当前 STMM 调整成员编号就会立即更改。
- 命令执行状态从 **CALL** 语句产生的 SQLCA 中返回。
- 此命令落实其在 **ADMIN_CMD** 过程中的更改。

兼容性

要获得与先前版本的兼容性:

- **DBPARTITIONNUM** 可替代 **MEMBER**，除非将 **DB2_ENFORCE_MEMBER_SYNTAX** 注册表变量设置为 ON。

配置管理 SQL 例程和视图

DB_PARTITIONS

DB_PARTITIONS 表函数以表格式返回 db2nodes.cfg 文件的内容。

注: 不推荐使用此表函数，且已替换为 DB2_MEMBER 和 DB2_CF 管理视图及 DB2_GET_INSTANCE_INFO 表函数。

语法

模式为 SYSPROC。

权限

需要下列其中一项权限来执行该例程:

- 例程上的 EXECUTE 特权
- DATAACCESS 权限
- DBADM 权限
- SQLADM 权限

缺省 PUBLIC 特权

在非限制数据库中，自动创建函数时会将 EXECUTE 特权授予 PUBLIC。

表函数参数

函数没有输入参数。

示例

检索来自 4 成员 分区数据库实例的信息。

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

下列是从此查询输出的示例:

```
PARTITION_NUMBER HOST_NAME  PORT_NUMBER SWITCHNAME
---------------- --------- ----------- -------
               0 so1                 0 so1-ib0
               1 so2                 0 so2-ib0
               2 so3                 0 so3-ib0
               3 so4                 0 so4-ib0

    4 record(s) selected.
```

在 DB2 pureScale 环境中，检索来自 3 成员 和 1 集群高速缓存设施 DB2 pureScale 实例的信息。

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

下列是从此查询输出的示例:

```
PARTITION_NUMBER HOST_NAME  PORT_NUMBER SWITCHNAME
---------------- ---------- ----------- ----------
               0 so1                  0 so1-ib0
               0 so2                  0 so2-ib0
               0 so3                  0 so3-ib0

    3 record(s) selected.
```

使用说明

对于 DB2 Enterprise Server Edition 和在分区数据库环境中，DB_PARTITIONS 表函数会为 db2nodes.cfg 文件中的每个条目返回一行。

在 DB2 pureScale 环境中，DB_PARTITIONS 表函数会返回多个行，并在列中带有以下信息:

- PARTITION_NUMBER 列始终包含 0。
- 其余列显示当前 成员 的 db2nodes.cfg 文件中条目的信息。

返回的信息

表 43. DB_PARTITIONS 表函数返回的信息

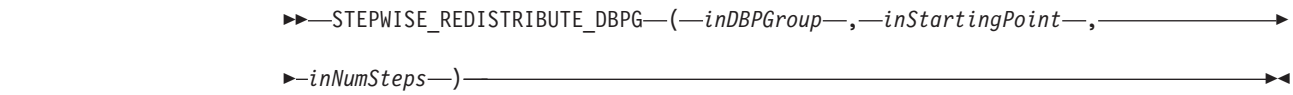
列名	数据类型	描述
PARTITION_NUMBER	SMALLINT	partition_number - 分区号监视器元素
HOST_NAME	VARCHAR(256)	host_name - 主机名监视器元素
PORT_NUMBER	SMALLINT	数据库分区服务器的端口号。
SWITCH_NAME	VARCHAR(128)	用于数据库分区通信的高速互连、交换机的名称。

按步骤重新分发管理 SQL 例程

STEPWISE_REDISTRIBUTE_DBPG 过程 - 重新分发部分数据库分区组

STEPWISE_REDISTRIBUTE_DBPG 过程根据为该过程指定的输入以及由 SET_SWRD_SETTINGS 过程创建或更新的设置文件来重新分发部分数据库分区组。

语法



模式为 SYSPROC。

过程参数

inDBPGroup
指定目标数据库分区组的名称的 VARCHAR (128) 类型的输入参数。

inStartingPoint
指定要使用的起始点的 SMALLINT 类型的输入参数。如果该参数被设置为正整数且不为 NULL，那么 STEPWISE_REDISTRIBUTE_DBPG 过程使用此值，而不是使用在设置文件中指定的 *nextStep* 值。当您想要从特定步骤重新运行 STEPWISE_REDISTRIBUTE_DBPG 过程时，这是非常有用的选项。如果该参数被设置为 NULL，那么将使用 *nextStep* 值。

inNumSteps
指定要运行的步骤编号的 SMALLINT 类型的输入参数。如果该参数被设置为正整数且不为 NULL，那么 STEPWISE_REDISTRIBUTE_DBPG 过程使用此值，而不是使用在设置文件中指定的 *stageSize* 值。当您想要使用与设置中指定的编号不同的步骤来重新运行 STEPWISE_REDISTRIBUTE_DBPG 过程时，这是非常有用的选项。例如，如果在预定的阶段中有五个步骤，并且重新分发过程在步骤 3 失败了，那么可以在纠正错误情况后调用 STEPWISE_REDISTRIBUTE_DBPG 过程来运行余下的三个步骤。如果该参数被设置为 NULL，那么将使用 *stageSize* 值。在此过程中，可以使用值 -2 来指示该编号是不受限制的。

注: 没有任何参数用于在 **REDISTRIBUTE DATABASE PARTITION GROUP** 命令中指定 **NOT ROLLFORWARD RECOVERABLE** 选项的等价选项。对于当使用 STEPWISE_REDISTRIBUTE_DBPG 过程时执行的行数据重新分发，始终会执行记录。

权限

- 对 STEPWISE_REDISTRIBUTE_DBPG 过程的 EXECUTE 特权
- SYSADM、SYSCTRL 或 DBADM

缺省 PUBLIC 特权

在非限制数据库中，自动创建过程时会将 EXECUTE 特权授予 PUBLIC。

示例

根据 SET_SWRD_SETTINGS 过程存储在注册表中的重新分发计划来重新分发数据库分区组“IBMDEFAULTGROUP”。它从步骤 3 开始重新分发数据，直至完成重新分发计划中的 2 个步骤。

```
CALL SYSPROC.STEPWISE_REDISTRIBUTE_DBPG('IBMDEFAULTGROUP', 3, 2)
```

有关逐步重新分发过程的完整用法示例，请参阅《*分区和集群指南*》中的『使用 STEPWISE_REDISTRIBUTE_DBPG 过程重新分发数据库分区组』。

使用说明

如果在开始执行 STEPWISE_REDISTRIBUTE_DBPG 过程后使用 SET_SWRD_SETTINGS 过程将 *processState* 的注册表值更新为 1，那么该进程会在下一步骤的开始阶段停止，并且返回警告消息。

由于重新分发进程调用 SQL COMMIT 语句，因此在 2 类连接下运行重新分发进程不受支持。

第 6 部分 附录

附录 A. 作为非 root 用户安装

作为非 root 用户安装 DB2 数据库服务器

可作为非 root 用户安装大多数 DB2 数据库产品。

开始之前

在作为非 root 用户安装任何 DB2 数据库产品之前，您应该了解 root 用户安装和非 root 用户安装之间的差别以及非 root 用户安装的局限性。有关非 root 用户安装的更多信息，请参阅『非 root 用户安装概述（Linux 和 UNIX）』。

作为非 root 用户安装 DB2 数据库产品的先决条件:

- 您必须能够安装该安装 DVD 或者自动安装。
- 您必须具有可用作 DB2 实例的所有者的有效用户标识。

  用户标识具有下列限制和要求:
  - 必须具有除 guests、admins、users 和 local 之外的主组
  - 可以包含小写字母 (a-z)、数字 (0-9) 和下划线字符 (_)
  - 长度不能超过八个字符
  - 不能以 IBM、SYS、SQL 或数字开头
  - 不能是 DB2 保留字（USERS、ADMINS、GUESTS、PUBLIC 或 LOCAL）或 SQL 保留字
  - 不能使用任何具有 root 用户特权的用户标识作为 DB2 实例标识、DAS 标识或受防护标识
  - 不能包含重音字符
  - 如果已指定现有用户标识，而不是创建新用户标识，那么确保该用户标识:
    - 未锁定
    - 不具有到期的密码
- 对于非 root 用户和 root 用户，您要安装的产品的硬件和软件先决条件都相同。
- 在 AIX 上，必须启用异步 I/O (AIO)。强烈建议系统启用 I/O 完成端口 (IOCP)。
- 您的主目录必须是有效的 DB2 路径。

  DB2 安装路径具有下列规则:
  - 可以包含小写字母 (a-z)、大写字母 (A-Z) 和下划线字符 (_)
  - 不能超过 128 个字符
  - 不能包含空格
  - 不能包含非英文字符

关于此任务

作为非 root 用户安装 DB2 数据库产品对于该非 root 用户是透明的。换而言之，除了作为非 root 用户登录之外，非 root 用户不需要执行特殊的操作就可以安装 DB2 数据库产品。

过程

为了执行非 root 用户安装:

1. 作为非 root 用户登录
2. 使用任何可用的方法来安装 DB2 数据库产品。 选项包括:
   - “DB2 安装”向导（GUI 安装）
   - **db2setup** 命令与响应文件（静默安装）

     注: 由于非 root 用户无法选择 DB2 数据库产品的安装目录，所以响应文件中的任何 **FILE** 关键字将被忽略。
3. 在安装 DB2 数据库产品后，您必须打开新登录会话以使用非 root 用户 DB2 实例。另外，如果使用 *$HOME*/sqllib/db2profile（对于 Bourne shell 和 Korn shell 用户）或 *$HOME*/sqllib/db2chsrc（对于 C shell 用户）来设置 DB2 实例环境（其中 *$HOME* 是非 root 用户的主目录），那么可以使用同一登录会话。

下一步做什么

在安装 DB2 数据库产品之后，请验证操作系统用户进程资源限制（ulimit）。如果不符合最小 ulimit 值，那么 DB2 引擎可能会遇到意外的操作资源不足错误。这些错误可能会导致 DB2 数据库系统停止运行。

附录 B. 使用备份

备份数据

使用 **BACKUP DATABASE** 命令来获取数据库数据副本并将其存储在另一介质上。然后在原始数据发生故障或损坏时使用此备份数据。

可以备份整个数据库，可以备份数据库分区，也可以只备份选择的表空间。

开始之前

不必连接到将要备份的数据库：备份数据库实用程序自动建立与指定数据库的连接，而此连接会在备份操作完成时终止。如果已连接到要备份的数据库，当发出 **BACKUP DATABASE** 命令时将断开连接，备份操作将继续进行。

数据库可以是本地数据库或远程数据库。备份映像保留在数据库服务器上，除非您使用的是存储管理产品，如 Tivoli Storage Manager (TSM) 或 DB2 高级副本服务 (ACS)。

如果要执行脱机备份并且已使用 **ACTIVATE DATABASE** 命令激活数据库，那么在运行脱机备份之前必须取消激活该数据库。如果存在与该数据库的活动连接，为了成功取消激活该数据库，具有 SYSADM 权限的用户必须连接至该数据库并发出下列命令：

```
CONNECT TO database-alias
QUIESCE DATABASE IMMEDIATE FORCE CONNECTIONS;
UNQUIESCE DATABASE;
TERMINATE;
DEACTIVATE DATABASE database-alias
```

在分区数据库环境中，可以使用 **BACKUP DATABASE** 命令来逐个备份数据库分区，使用 **ON DBPARTITIONNUM** 命令参数来一次性备份多个数据库分区或使用 **ALL DBPARTITIONNUMS** 参数来同时备份所有数据库分区。可以使用 **LIST DBPARTITIONNUMS** 命令来确定具有您想要备份的用户表的数据库分区。

除非您在使用单一系统视图（SSV）备份，否则，如果您要在分区数据库环境中执行*脱机*备份，那么应将目录分区独立于所有其他数据库分区进行备份。例如，可以先备份目录分区，然后备份所有其他数据库分区。此操作是必需的，因为备份操作可能需要在目录分区上进行独占数据库连接（在此期间不能连接其他数据库分区）。如果要执行*联机*备份，那么可以同时或以任何顺序备份所有数据库分区（包括目录分区）。

在分布式请求系统上，备份操作适用于分布式请求数据库和存储在数据库目录（包装器、服务器和昵称等等）中的元数据。不备份数据源对象（表和视图），除非它们也存储在分布式请求数据库中。

如果某个数据库是使用数据库管理器的前发行版创建的且尚未升级，那么必须先升级该数据库才能对其进行备份。

限制

以下限制适用于 BACKUP 实用程序：

- 表空间备份操作和表空间复原操作不能同时运行，即使涉及的是不同的表空间。

- 如果要能够在分区数据库环境中执行前滚恢复，那么必须定期备份节点列表上的数据库。还必须具有系统中余下节点（甚至未包含该数据库的用户数据的节点）的至少一个备份映像。下列两种情况都需要在不包含数据库的用户数据的数据库分区服务器中存在数据库分区的备份映像:
  - 在建立上一个备份之后已将一个数据库分区服务器添加到数据库系统，因此需要在此数据库分区服务器上执行正向恢复。
  - 使用时间点恢复，它要求系统中的所有数据库分区都处于前滚暂挂状态。
- DMS 表空间的联机备份操作与下列操作不兼容:
  - 装入
  - 重组（联机和脱机）
  - 删除表空间
  - 截断表
  - 创建索引
  - 最初未记录任何内容（与 CREATE TABLE 和 ALTER TABLE 语句配合使用）
- 如果尝试对当前活动的数据库执行脱机备份，那么将接收到错误。在运行脱机备份之前，可以通过发出 **DEACTIVATE DATABASE** 命令来确保数据库未处于活动状态。

过程

要调用 BACKUP 实用程序，请执行以下操作:

- 在命令行处理器 (CLP) 中发出 **BACKUP DATABASE** 命令。
- 运行带 BACKUP DATABASE 参数的 ADMIN_CMD 过程。
- 使用 db2Backup 应用程序编程接口 (API)。
- 在 IBM Data Studio 中对 **BACKUP DATABASE** 命令打开任务助手。

示例

以下是通过 CLP 发出的 **BACKUP DATABASE** 命令的示例:

```
db2 backup database sample to c:\DB2Backups
```

下一步做什么

如果执行了脱机备份，那么在备份完成后，必须重新激活该数据库:

```
ACTIVATE DATABASE sample
```

附录 C. 分区数据库环境目录视图

SYSCAT.BUFFERPOOLDBPARTITIONS

每一行表示缓冲池和成员的组合，其中该成员上的缓冲池大小与同一数据库分区组中其他成员的缓冲池缺省大小不同（如 SYSCAT.BUFFERPOOLS 中所述）。

表 44. SYSCAT.BUFFERPOOLDBPARTITIONS 目录视图

列名	数据类型	可空	描述
BUFFERPOOLID	INTEGER		内部缓冲池标识。
DBPARTITIONNUM	SMALLINT		成员编号。
NPAGES	INTEGER		此成员上的此缓冲池中的页数。

SYSCAT.DATAPARTITIONEXPRESSION

每一行均表示用于该部分表分区键的表达式。

表 45. SYSCAT.DATAPARTITIONEXPRESSION 目录视图

列名	数据类型	可空	描述
TABSCHEMA	VARCHAR (128)		分区表的模式名。
TABNAME	VARCHAR (128)		分区表的非标准名称。
DATAPARTITIONKEYSEQ	INTEGER		从 1 开始的表达式关键部分序列标识。
DATAPARTITIONEXPRESSION	CLOB (32K)		SQL 语法中该序列中此条目的表达式。
NULLSFIRST	CHAR (1)		• N = 此表达式 compare high 中的空值 • Y = 此表达式 compare low 中的空值

SYSCAT.DATAPARTITIONS

每一行均表示数据分区。请注意，如果在多个数据库分区上创建了表，那么数据分区统计信息描述一个数据库分区。

表 46. SYSCAT.DATAPARTITIONS 目录视图

列名	数据类型	可空	描述
DATAPARTITIONNAME	VARCHAR (128)		数据分区的名称。
TABSCHEMA	VARCHAR (128)		此数据分区所属的表的模式名。
TABNAME	VARCHAR (128)		此数据分区所属的表的非标准名称。
DATAPARTITIONID	INTEGER		数据分区的标识。
TBSPACEID	INTEGER	Y	存储此数据分区的表空间的标识。当 STATUS 为“I”时，此标识的值为 NULL。
PARTITIONOBJECTID	INTEGER	Y	表空间中数据分区的标识。
LONG_TBSPACEID	INTEGER	Y	存储长数据的表空间的标识。当 STATUS 为“I”时，此标识的值为 NULL。

表 46. SYSCAT.DATAPARTITIONS 目录视图 (续)

列名	数据类型	可空	描述
ACCESS_MODE	CHAR (1)		数据分区的访问限制状态。这些状态仅适用于处于集合完整性暂挂状态的对象或被 SET INTEGRITY 语句处理的对象。可能的值包括: • D = 没有数据移动 • F = 完全访问 • N = 无访问 • R = 只读访问
STATUS	VARCHAR(32)		• A = 新近连接数据分区 • D = 数据分区已断开连接，并且要以增量方式维护已断开连接的从属（对于此分区的内容） • I = 其条目在目录中的已拆离的数据分区仅在异步索引清除期间才会被维护；当引用已拆离分区的所有索引记录删除后，STATUS 值为“I”的行将会被除去 • L = 逻辑上数据库分区已断开连接 • 空字符串 = 数据分区可视（正常状态） 字节 2 至 32 被保留以供将来使用。
SEQNO	INTEGER		数据分区序号（从 0 开始）。
LOWINCLUSIVE	CHAR (1)		• N = 不包括低键值 • Y = 包括低键值
LOWVALUE	VARCHAR(512)		用于此数据分区的低键值（SQL 值的字符串表示法）。
HIGHINCLUSIVE	CHAR (1)		• N = 不包括高键值 • Y = 包括高键值
HIGHVALUE	VARCHAR(512)		用于此数据分区的高键值（SQL 值的字符串表示法）。
CARD	BIGINT		数据分区中的总行数；如果未收集统计信息，那么为 -1。
OVERFLOW	BIGINT		数据分区中的总溢出记录数；如果未收集统计信息，那么为 -1。
NPAGES	BIGINT		数据分区的行所在页面的总页数；如果未收集统计信息，那么为 -1。
FPAGES	BIGINT		数据分区中的总页数；如果未收集统计信息，那么为 -1。
ACTIVE_BLOCKS	BIGINT		数据分区中的总活动块数，或 -1。仅适用于多维集群 (MDC) 表。
INDEX_TBSPACEID	INTEGER		用于保存此数据分区所有分区索引的表空间的标识。
AVGROWSIZE	SMALLINT		此数据分区中压缩行和未压缩行的平均长度（以字节计）；如果未收集统计信息，那么为 -1。

表 46. SYSCAT.DATAPARTITIONS 目录视图 (续)

列名	数据类型	可空	描述
PCTROWSCOMPRESSED	REAL		数据分区中压缩行占总行数的百分比；如果未收集统计信息，那么为 -1。
PCTPAGESAVED	SMALLINT		作为行压缩的结果，保存在数据分区中的页面的大致百分比。此值包括用于数据分区中每个用户数据行的额外字节，但不包括由字典开销使用的空间；如果未收集统计信息，那么为 -1。
AVGCOMPRESSEDROWSIZE	SMALLINT		此数据分区中压缩行的平均长度（以字节计）；如果未收集统计信息，那么为 -1。
AVGROWCOMPRESSIONRATIO	REAL		对于数据分区中的压缩行，这是基于行的平均压缩率；即，平均未压缩行长度除以平均压缩行长度；如果未收集统计信息，那么为 -1。
STATS_TIME	TIMESTAMP	Y	最近一次对此对象的所记录统计信息进行了任何更改的时间。如果未收集统计信息，那么为 NULL。
LASTUSED	DATE		任何 DMS 语句或 LOAD 命令最近一次使用数据分区的日期。在 HADR 备用数据库上使用该数据分区时，不会更新此列。缺省值为“0001-01-01”。此值会异步更新，这样该值可能未反映最后 15 分钟的使用情况，并且在更新后 24 小时保持不变。

SYSCAT.DBPARTITIONGROUPDEF

每一行均表示数据库分区组中包含的数据库分区。

表 47. SYSCAT.DBPARTITIONGROUPDEF 目录视图

列名	数据类型	可空	描述
DBPGNAME	VARCHAR (128)		包含数据库分区的数据库分区组的名称。
DBPARTITIONNUM	SMALLINT		数据库分区组中包含的数据库分区的分区号。有效的分区号介于 0 和 999 之间（包括 0 和 999）。
IN_USE	CHAR (1)		数据库分区的状态。 • A = 新添加的数据库分区不在分发映射中，但数据库分区组中已创建表空间的容器；当重新分发数据库分区组操作成功完成后，会将数据库分区添加到分发映射中。 • D = 当重新分发数据库分区组操作成功完成后，将删除该数据库分区。 • T = 新添加的数据库分区不在分发映射中，并且它是使用 WITHOUT TABLESPACES 子句添加的；必须将容器添加至数据库分区组中的表空间。 • Y = 数据库分区在分发映射中。

SYSCAT.DBPARTITIONGROUPS

每一行均表示数据库分区组。

表 48. SYSCAT.DBPARTITIONGROUPS 目录视图

列名	数据类型	可空	描述
DBPGNAME	VARCHAR (128)		数据库分区组的名称。
OWNER	VARCHAR (128)		数据库分区组所有者的授权标识。
OWNERTYPE	CHAR (1)		• S = 系统的所有者 • U = 所有者是用户个体
PMAP_ID	SMALLINT		SYSCAT.PARTITIONMAPS 目录视图中分发映射的标识。
REDISTRIBUTE_PMAP_ID	SMALLINT		当前进行重新分发所使用的分发映射的标识；如果当前未在进行重新分发，那么它为 -1。
CREATE_TIME	TIMESTAMP		数据库分区组的创建时间。
DEFINER[1]	VARCHAR (128)		数据库分区组所有者的授权标识。
REMARKS	VARCHAR(254)	Y	用户提供的注释或 null 值。

注:

1. 包括 DEFINER 列以便实现向后兼容。请参阅 OWNER。

SYSCAT.PARTITIONMAPS

每一行均表示一个分发映射，后者用于根据散列表的分布来在数据库分区组中数据库分区之间分布该表的各行。

表 49. SYSCAT.PARTITIONMAPS 目录视图

列名	数据类型	可空	描述
PMAP_ID	SMALLINT		分发映射的标识。
PARTITIONMAP	BLOB（65536）		分发映射，用于多分区数据库分区组的 32768 个双字节整数的向量。对于单一分区数据库分区组，存在有一个条目，表明单一分区的分区号。

附录 D. 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 50. DB2 技术信息

书名	书号	是否提供印刷版	发布日期
Administrative API Reference	SC27-5506-00	是	2013 年 7 月 28 日
Administrative Routines and Views	SC27-5507-00	否	2013 年 7 月 28 日
Call Level Interface Guide and Reference Volume 1	SC27-5511-00	是	2013 年 7 月 28 日
Call Level Interface Guide and Reference Volume 2	SC27-5512-00	是	2013 年 7 月 28 日
Command Reference	SC27-5508-00	是	2013 年 7 月 28 日
数据库管理概念和配置参考	S151-1970-00	是	2013 年 7 月 28 日
数据移动实用程序指南和参考	S151-1992-00	是	2013 年 7 月 28 日
数据库监视指南和参考	S151-1971-00	是	2013 年 7 月 28 日
数据恢复和高可用性指南与参考	S151-1993-00	是	2013 年 7 月 28 日
数据库安全性指南	S151-1994-00	是	2013 年 7 月 28 日
DB2 工作负载管理指南与参考	S151-1986-00	是	2013 年 7 月 28 日
开发 ADO.NET 和 OLE DB 应用程序	S151-1973-00	是	2013 年 7 月 28 日
开发嵌入式 SQL 应用程序	S151-1974-00	是	2013 年 7 月 28 日
Developing Java Applications	SC27-5503-00	是	2013 年 7 月 28 日
Developing Perl, PHP, Python, and Ruby on Rails Applications	SC27-5504-00	否	2013 年 7 月 28 日
Developing RDF Applications for IBM Data Servers	SC27-5505-00	是	2013 年 7 月 28 日
开发用户定义的例程（SQL 和外部例程）	S151-1975-00	是	2013 年 7 月 28 日
数据库应用程序开发入门	G151-1976-00	是	2013 年 7 月 28 日
Linux 和 Windows 上的 DB2 安装和管理入门	G151-1978-00	是	2013 年 7 月 28 日
全球化指南	S151-1995-00	是	2013 年 7 月 28 日
安装 DB2 服务器	G151-1980-00	是	2013 年 7 月 28 日
安装 IBM Data Server Client	G151-1981-00	否	2013 年 7 月 28 日
消息参考第 1 卷	S151-1989-00	否	2013 年 7 月 28 日
消息参考第 2 卷	S151-1990-00	否	2013 年 7 月 28 日

表 50. DB2 技术信息 (续)

书名	书号	是否提供印刷版	发布日期
Net Search Extender 管理和用户指南	S151-1991-00	否	2013 年 7 月 28 日
分区和集群指南	S151-1996-00	是	2013 年 7 月 28 日
pureXML 指南	S151-1987-00	是	2013 年 7 月 28 日
Spatial Extender User's Guide and Reference	SC27-5525-00	否	2013 年 7 月 28 日
SQL 过程语言: 应用程序启用和支持	S151-1977-00	是	2013 年 7 月 28 日
SQL Reference Volume 1	SC27-5509-00	是	2013 年 7 月 28 日
SQL Reference Volume 2	SC27-5510-00	是	2013 年 7 月 28 日
Text Search Guide	SC27-5527-00	是	2013 年 7 月 28 日
故障诊断和调整数据库性能	S151-1972-00	是	2013 年 7 月 28 日
升级到 DB2 V10.5	S151-1979-00	是	2013 年 7 月 28 日
DB2 V10.5	S151-1985-00	是	2013 年 7 月 28 日
XQuery 参考	S151-1988-00	否	2013 年 7 月 28 日

表 51. 特定于 DB2 Connect 的技术信息

书名	书号	是否提供印刷版	发布日期
DB2 Connect 安装和配置 DB2 Connect Personal Edition	S151-1982-00	是	2013 年 7 月 28 日
DB2 Connect 安装和配置 DB2 Connect 服务器	S151-1983-00	是	2013 年 7 月 28 日
DB2 Connect 用户指南	S151-1984-00	是	2013 年 7 月 28 日

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://pic.dhe.ibm.com/infocenter/db2luw/v10r1。

对于 DB2 V9.8 主题，*DB2 信息中心* URL 是 http://pic.dhe.ibm.com/infocenter/db2luw/v9r8/。

对于 DB2 V9.7 主题，*DB2 信息中心* URL 是 http://pic.dhe.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。

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 E. 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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, 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]

F

G

H

I

L

M

N

R

S

T

U

V

W

X

[特别字符]

Printed in China

S151-1996-00

Spine information:

IBM DB2 10.5 for Linux, UNIX, and Windows

分区和集群指南

IBM