IBM® DB2 通用数据库™

管理指南：计划

版本 8

S152-0167-00

IBM® DB2 通用数据库™

管理指南：计划

版本 8

S152-0167-00

在使用本资料及其支持的产品之前，务必阅读*声明*中的一般信息。

本文档包含 IBM 的专利信息。它在许可证协议下提供，并受版权法保护。本出版物包含的信息不包括任何产品保证，且本手册提供的任何声明不应作如此解释。

可以在线方式或通过您当地的 IBM 代表订购 IBM 出版物。

- 要以在线方式订购出版物，可访问 IBM 出版物中心（IBM Publications Center），网址为 www.ibm.com/shop/publications/order。
- 要查找您当地的 IBM 代表，可访问 IBM 全球联系人目录（IBM Directory of Worldwide Contacts），网址为 www.ibm.com/planetwide。

在美国或加拿大，要从 “DB2 市场营销和销售中心” 订购 DB2 出版物，请致电 1-800-IBM-4YOU（426-4968）。

当您发送信息给 IBM 后，即授予 IBM 非专有权，IBM 对于您所提供的任何信息，有权利以任何它认为适当的方式使用或分发，而不必对您负任何责任。

目录

关于本书

“管理指南”的其中三卷提供使用和管理 DB2 关系数据库管理系统（RDBMS）产品所需的信息，并包括:

- 关于数据库设计的信息（可在*《管理指南: 计划》*中找到）
- 关于实现和管理数据库的信息（可在*《管理指南: 实现》*中找到）
- 关于配置和调整数据库环境以提高性能的信息（在*《管理指南: 性能》*中）

本书中描述的许多任务可以使用不同的接口来执行:

- **命令行处理器**，它允许您从图形界面访问和操纵数据库。从此接口，您也可执行 SQL 语句和 DB2 实用程序功能。本书中的大多数示例说明的是此接口的使用。有关使用命令行处理器的更多信息，参见 *Command Reference*。
- **应用程序编程接口**，它允许您在应用程序内执行 DB2 实用程序功能。有关使用应用程序编程接口的更多信息，参见 *Administrative API Reference*。
- **控制中心**，允许您使用图形用户界面来执行管理任务，如配置系统、管理目录、备份和恢复系统、调度作业和管理介质。“控制中心”还包含“复制管理”，它允许您设置系统间的数据复制。此外，“控制中心”使您能通过图形用户界面执行 DB2 实用程序功能。根据平台的不同，调用“控制中心”的方法也不同。例如，在命令行上使用 `db2cc` 命令，从 DB2 文件夹中选择 控制中心 图标，或在 Windows 平台上使用“开始”菜单。要获得介绍性帮助，从控制中心窗口的**帮助**下拉菜单中选择**入门**。**Visual Explain** 和**性能监视器**工具是从控制中心中调用的。

还可使用其它工具来执行管理任务。它们包括:

- 用来存储称为脚本的小应用程序的“脚本中心”。这些脚本可包含 SQL 语句、DB2 命令以及操作系统命令。
- 用来监视其它 DB2 操作导致的消息的“警告中心”。
- “健康中心”提供了一个工具来帮助 DB2 解决性能和资源分配问题。
- 用来更改“控制中心”、“警告中心”和“复制”的设置的“工具设置”。
- 用来调度要以无人照管方式运行的作业的“日志”。
- 用来管理仓库对象的“数据仓库中心”。

应使用本书的人员

本书主要面向需要设计、实现和维护要由本地或远程客户机存取的数据库的数据库管理员、系统管理员、安全性管理员和系统操作员。需要了解 DB2 关系数据库管理系统的管理和操作的程序员和其它用户也可使用本书。

本书的结构

本书包含关于下列主要主题的信息:

数据库概念

- 第 1 章, 『基本关系数据库概念』, 提供数据库对象和数据库概念的概述。
- 第 2 章, 『并行数据库系统』, 提供有关 DB2 可使用的并行性类型的介绍。
- 第 3 章, 『关于数据入库』, 提供数据入库和数据入库任务的概述。

数据库设计

- 第 4 章, 『逻辑数据库设计』, 讨论逻辑数据库设计的概念和准则。
- 第 5 章, 『物理数据库设计』, 讨论物理数据库设计（包括空间需求和表空间设计）的准则。
- 第 6 章, 『设计分布式数据库』, 讨论如何在单个事务中存取多个数据库。
- 第 7 章, 『针对 XA 兼容事务管理器进行设计』, 讨论如何在分布式事务处理环境中使用数据库。

附录

- 附录 A, 『发行版之间的不兼容性』, 展现版本 7 和版本 8 带来的不兼容性, 以及将来应注意的不兼容性。
- 附录 B, 『本地语言支持（NLS）』, 介绍 “DB2 本地语言支持”, 包括关于国家或地区、语言和代码页的信息。

“管理指南”其它卷的简要概述

管理指南: 实现

《*管理指南: 实现*》主要讨论数据库设计的实现。下面简要描述该卷中的特定章节和附录:

实现设计

- “在创建数据库之前”描述创建数据库之前的前提条件。
- “创建数据库”描述那些与数据库及相关数据库对象的创建相关联的任务。

- “改变数据库”讨论改变数据库之前必须完成的操作，以及与修改或删除数据库或相关数据库对象的相关联的任务。

数据库安全性

- “控制数据库存取”描述如何控制对数据库资源的存取。
- “审计 DB2 活动”描述如何检测和监视不想要或不希望进行的数据存取。

附录

- “命名规则”，提供在命名数据库和对象时要遵循的规则。
- “轻量级目录访问协议（LDAP）目录服务”提供关于如何使用“LDAP 目录服务”的信息。
- “对多个数据库分区发出命令”讨论如何使用 *db2_all* 和 *rah* 外壳程序脚本，以将命令发送至分区数据库环境中的所有分区。
- “Windows 管理规范（WMI）支持”描述 DB2 如何支持此管理基础结构标准将各种硬件和软件管理系统集成在一起。还讨论了 DB2 如何与 WMI 集成。
- “DB2 Windows NT 版如何使用 Windows NT 的安全性”描述 DB2 如何使用 Windows NT 安全性。
- “使用 Windows 性能监视器”提供关于向“Windows NT 性能监视器”注册 DB2 以及使用性能信息的信息。
- “使用 Windows 数据库分区服务器”提供关于可使用 Windows NT 或 Windows 2000 上的数据库分区服务器的实用程序的信息。
- “配置多个逻辑节点”描述如何在分区数据库环境中配置多个逻辑节点。
- “扩展控制中心”提供有关如何通过添加包含新操作的新工具栏按钮、添加新对象定义及添加新操作定义来扩展控制中心的信息。

注: 本书删除了两章。

有关移动数据的 DB2 实用程序以及 *Command Reference* 和 *Administrative API Reference* 中的类似主题的所有信息都被整合到 *Data Movement Utilities Guide and Reference* 中。

Data Movement Utilities Guide and Reference 是这些主题的主要的单一信息源。

要找到更多有关数据复制的信息，参见 *Replication Guide and Reference*。

有关备份和恢复数据的方法、以及 *Command Reference* 和 *Administrative API Reference* 中的类似主题的所有信息都被整合到*《数据恢复及高可用性指南与参考》*中。

《*数据恢复及高可用性指南与参考*》是这些主题的主要的单一信息源。

管理指南: 性能

《*管理指南: 性能*》主要讨论性能问题，即，那些主题和论点讨论建立、测试和改进应用程序及“DB2 通用数据库”产品本身的性能。下面简要描述该卷中的特定章节和附录:

性能介绍

- “性能介绍”介绍管理和改进 DB2 UDB 性能的概念和注意事项。
- “体系结构与进程”介绍基本的“DB2 通用数据库”体系结构和进程。

调整应用程序性能

- “应用程序注意事项”描述在设计应用程序时用来改进数据库性能的一些技巧。
- “环境注意事项”描述在设置数据库管理器用来改进数据库性能的一些技巧。
- “系统目录统计信息”描述如何收集并使用数据统计信息以确保最优性能。
- “了解 SQL 编译器”描述使用 SQL 编译器编译 SQL 语句的过程。
- “SQL 说明设施”描述“说明”设施，该设施允许您检查 SQL 编译器为存取数据所作的选择。

调整和配置系统

- “运行性能”概述数据库管理器如何使用内存以及影响运行时性能的其它注意事项。
- “使用控制器”介绍如何使用控制器来控制数据库管理的某些方面。
- “调整配置”描述与增加数据库系统的大小相关的一些注意事项和任务。
- “将数据重新分布到各分区中”讨论在分区数据库环境中将数据重新分布在各分区中所需的任务。
- “基准测试”提供有关基准测试和如何执行基准测试的概述。
- “配置 DB2”讨论数据库管理器和数据库配置文件，以及数据库管理器、数据库和 DAS 配置参数的值。

附录

- “DB2 注册表和环境变量”描述概要文件的注册表值和环境变量。
- “说明表和定义”描述“DB2 说明”设施使用的表以及如何创建那些表。
- “SQL 说明工具”描述如何使用 DB2 说明工具: db2expln 和 dynexpln。
- “db2exfmt — 说明表格式化工具”描述如何使用 DB2 说明工具来格式化说明表数据。

第 1 部分 数据库概念

第 1 章 基本关系数据库概念

数据库对象

实例:

实例（有时称为*数据库管理器*）是管理数据的 DB2® 代码。它控制可对数据执行的操作，并管理分配给它的系统资源。每一个实例都是一个完整的环境。它包含为给定并行数据库系统定义的所有数据库分区。一个实例有它自己的数据库（其它实例不能存取它），并且它的全部数据库分区共享相同的系统目录。它还有独立的安全性，不受同一机器（系统）上其它实例的影响。

数据库:

*关系数据库*将数据表示成表的集合。表由数目已定的列和任意数目的行组成。每个数据库都包括一组描述数据的逻辑和物理结构的系统目录表，一个包含为该数据库分配的参数值的配置文件以及一个带有正在进行的事务和可归档事务的恢复日志。

数据库分区组:

*数据库分区组*是一个或多个数据库分区的集合。想要为数据库创建表时，首先创建用来存储表空间的数据库分区组，然后创建用来存储表的表空间。

在较早版本的 DB2 中，*数据库分区组*称为*节点组*。

表:

关系数据库将数据表示成表的集合。表由逻辑排列的行和列数据组成。所有数据库和表数据都被存储在表空间中。表中的数据在逻辑上是相关的，且可以定义表与表之间的关系。根据数学规则和关系运算来查看和操纵数据。

表数据是通过“结构化查询语言”（SQL）存取的，SQL 是一种标准化语言，用于在关系数据库中定义和处理数据。应用程序或用户使用*查询*从数据库中检索数据。查询使用 SQL 来创建如下格式的语句

```
SELECT <data_name> FROM <table_name>
```

视图:

*视图*是高效率的数据呈示方法（无需维护数据）。视图不是实际的表，不需要永久性存储器。创建并使用一个“虚拟表”。

视图可以包括它所基于的表中的所有或某些列或行。例如，可以在视图中连接一个部门表和一个雇员表，以便可以列出特定部门中的所有雇员。

图 1 显示了表与视图之间的关系。

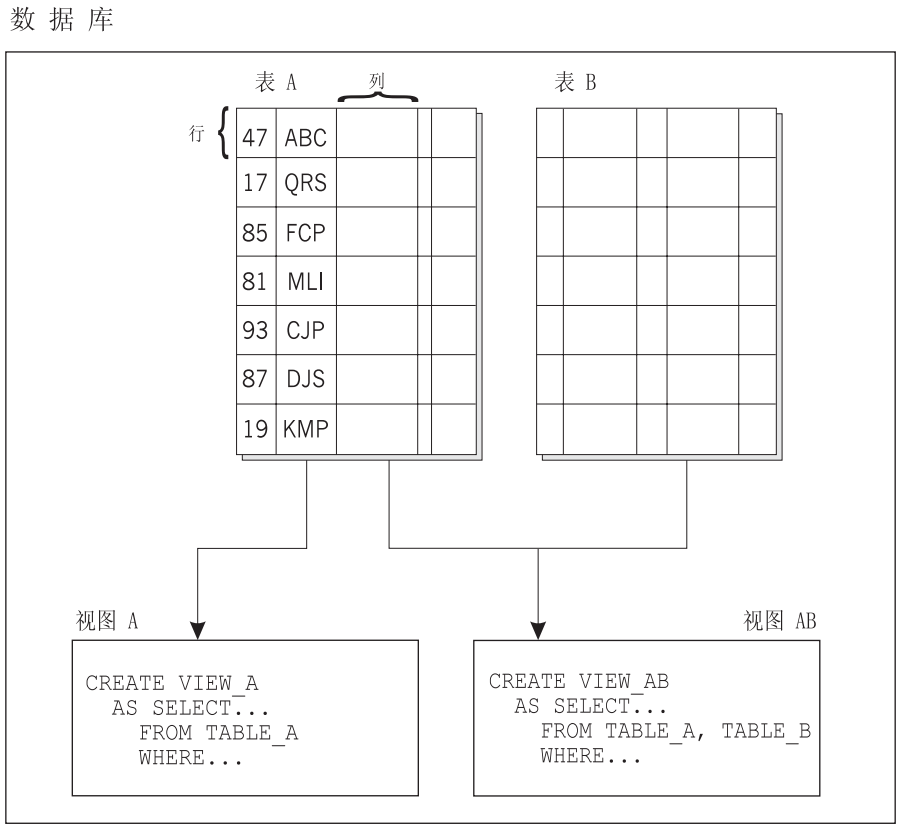

图 1. 表与视图之间的关系

索引:

*索引*是一组键，其每一个都指向一些表行。例如，在第 5 页的图 2 中，表 A 的一个索引基于表中的雇员号码。此键值提供指向表行的指针：雇员号码 19 指向雇员 KMP。通过使用指针创建指向数据的直接路径，索引使更有效率地存取表行成为可能。

SQL *优化器*自动选择最有效率的存取表中数据的方法。当确定最快速的数据存取路径时，优化器会将索引考虑在内。

可创建唯一索引以确保索引键的唯一性。*索引键*是定义了索引的一个列或一些列的有序集合。使用唯一索引将确保在编入索引的列中，每个索引键的值都是唯一的。

图 2 显示了索引与表之间的关系。

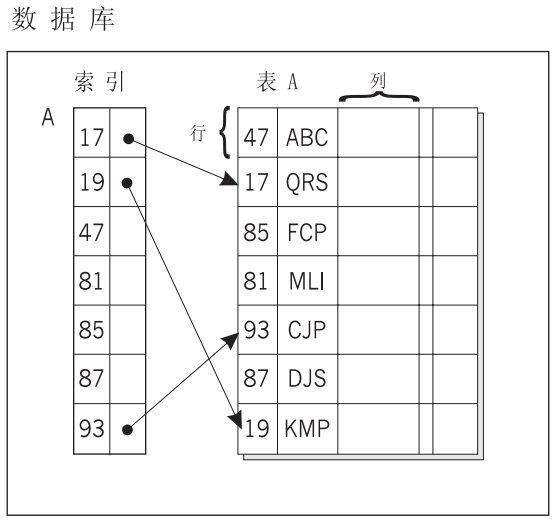

图 2. 索引与表之间的关系

第 6 页的图 3 说明一些数据库对象之间的关系。它还显示了表、索引和长型数据存储在表空间中的情况。

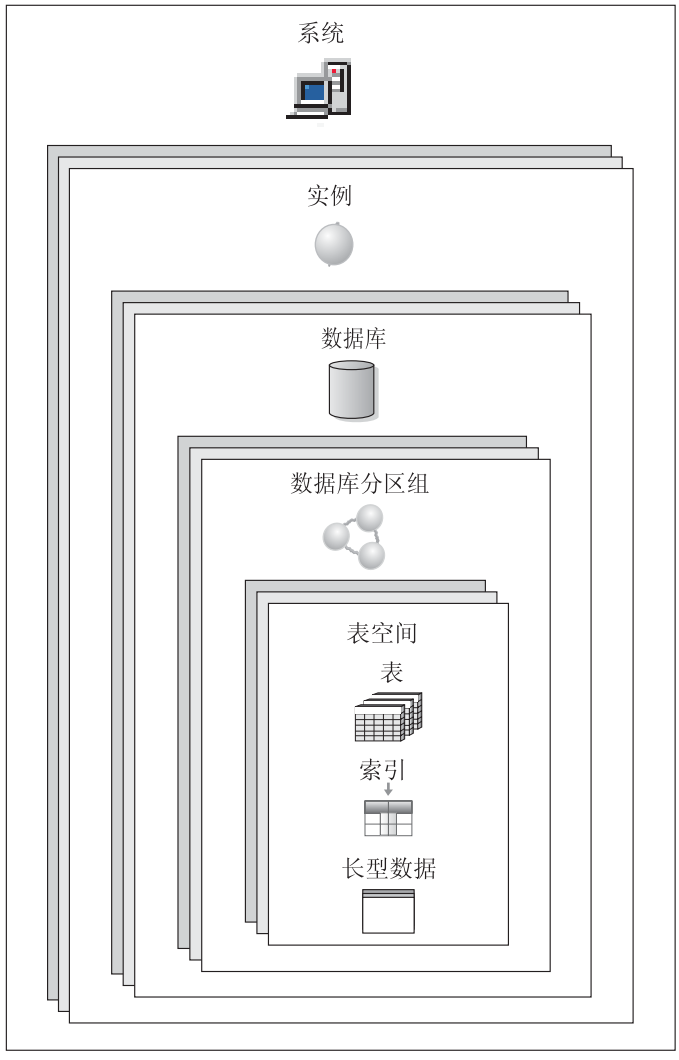

图 3. 一些数据库对象之间的关系

模式:

模式是一个标识符，如用户标识，它帮助分组表和其它数据库对象。模式可以归个人所有，所有者可以控制对数据以及其中的对象的存取。

模式也可以是数据库中的对象。它可以在创建模式中的第一个对象时自动创建。这样的对象可以是任何由模式名限定的对象，如表、索引、视图、程序包、单值类型、函数或触发器。若要自动创建模式，则您必须拥有 IMPLICIT_SCHEMA 权限，也可以隐式地创建模式。

模式名用作两部分对象名的第一部分。创建一个对象时，可将其分配给特定模式。若不指定模式，则它被分配给缺省模式，缺省模式通常是创建该对象的人员的用户标识。名称的第二部分是对象名。例如，名为 Smith 的用户可以有一个名为 SMITH.PAYROLL 的表。

系统目录表:

每个数据库都包括一组描述数据的逻辑和物理结构的*系统目录表*。DB2 为每个数据库维护一大组系统目录表。这些表包含有关数据库对象（例如，用户表、视图和索引）的定义的信息，以及用户对这些对象所拥有的权限的安全性信息。它们在数据库创建时被创建，并在常规操作期间得到更新。不能显式地创建或删除它们，但是可以使用目录视图查询和查看它们的内容。

表空间:

数据库由称为*表空间*的部件组成。表空间是用来存储表的位置。当创建表时，您可以决定将特定对象（如索引和大对象（LOB））数据与其余表数据分开存放。表空间也可以分布在一个或多个物理存储设备上。下图表显示了在表空间之间分布数据时具有的一些灵活性。

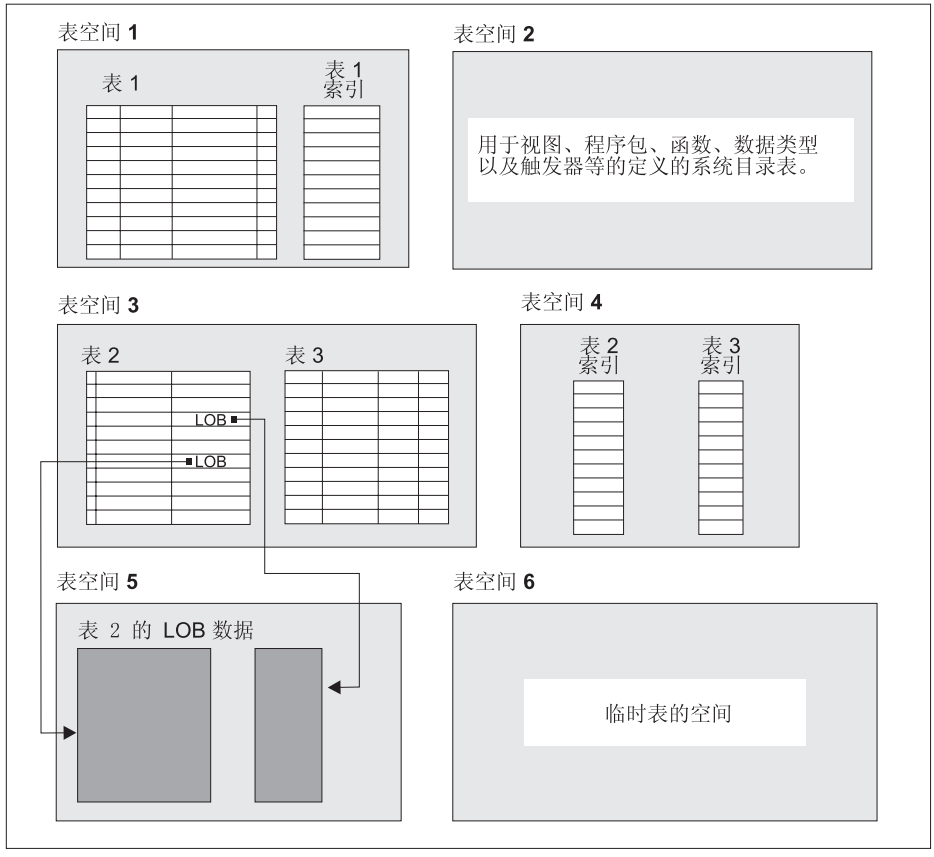

图 4. 表空间

表空间驻留在数据库分区组中。表空间定义和属性记录在数据库系统目录中。

将容器分配给表空间。容器是分配的物理存储器（如文件或设备）。

表空间可以是系统管理空间（SMS）或数据库管理空间（DMS）。对于 SMS 表空间，每个容器都是操作系统的文件空间中的一个目录，由操作系统的文件管理器控制存储空间。对于 DMS 表空间，每个容器或者是固定大小的预分配文件，或者是物理设备如磁盘，由数据库管理器控制存储空间。

第 9 页的图 5 举例说明了表、表空间以及两种类型的空间之间的关系。它还显示了表、索引和长型数据存储在表空间中的情况。

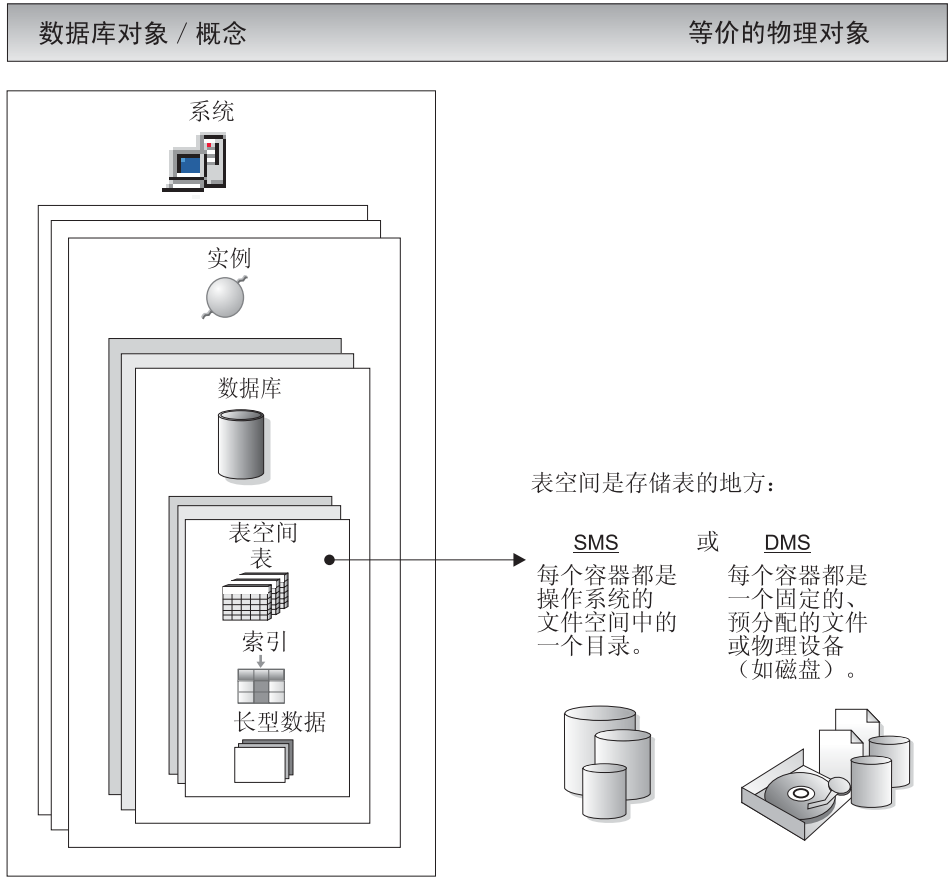

图 5. 表空间和表

第 10 页的图 6 显示了三种表空间类型: *常规*、*临时*和大（*型*）。

包含用户数据的表存放在规则表空间中。缺省用户表空间名为 USERSPACE1。系统目录表存放在规则表空间中。缺省系统目录表空间名为 SYSCATSPACE。

包含长型字段数据或长型对象数据（如多媒体对象）的表存在于大表空间或常规表空间中。这些列的基本列数据存储在常规表空间中，而长型字段或大对象数据可以存储在相同的常规表空间或指定大表空间中。

索引可以存储在常规表空间或大表空间中。

*临时表空间*分为系统临时表空间或用户临时表空间。*系统临时表空间*用来存储 SQL 操作（如排序、重组表、创建索引和连接表）期间所需的内部临时数据。虽然可以创建任意数目个系统临时表空间，但建议您只使用大多数表所使用的页大小创

建一个。缺省系统临时表空间名为 TEMPSPACE1。*用户临时表空间*用来存储已声明全局临时表（已声明全局临时表存储的是应用程序临时数据）。用户临时表空间*不是*在数据库创建时缺省创建的。

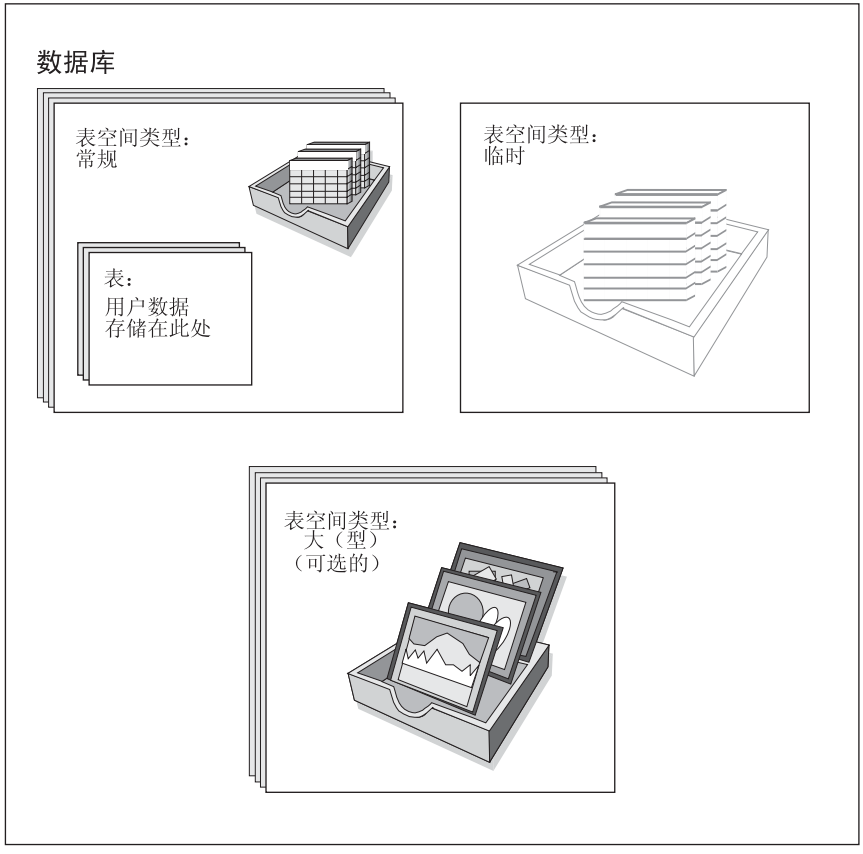

图 6. 三种表空间类型

容器:

*容器*是物理存储设备。它可以由目录名、设备名或文件名标识。

将为表空间分配容器。单个表空间可以横跨多个容器，但每个容器只能属于一个表空间。

第 11 页的图 7 举例说明了表与数据库中的表空间、相关联的容器和磁盘之间的关系。

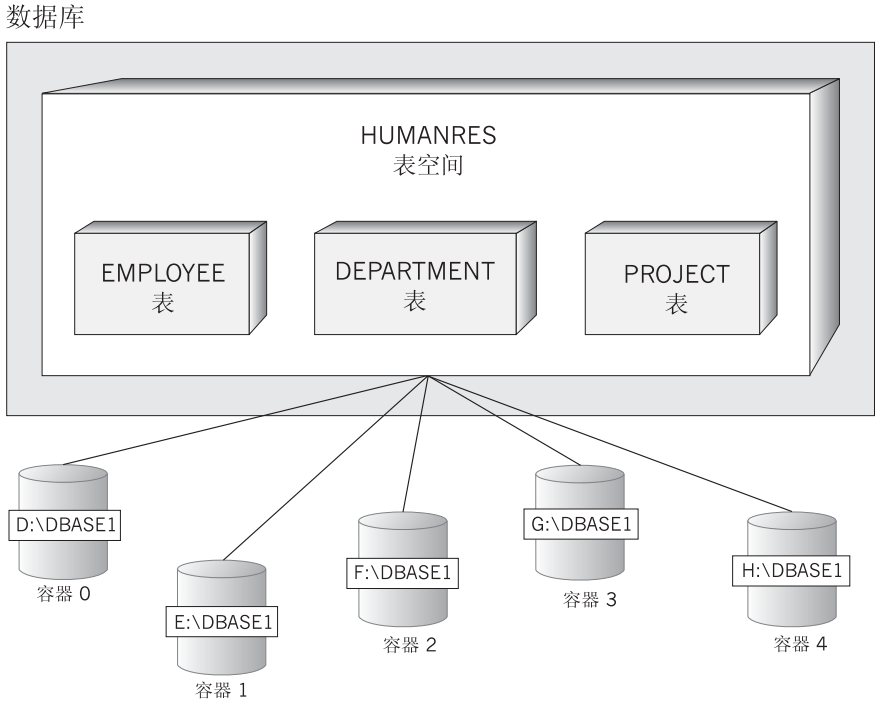

图 7. 数据库中的表空间和表

EMPLOYEE、DEPARTMENT 和 PROJECT 表在 HUMANRES 表空间中，该表空间横跨容器 0、1、2、3 和 4。此示例显示每个容器存在于不同的磁盘上。

任何表的数据都以循环方式存储在表空间中的所有容器中。这能在属于给定表空间的容器之间平衡数据。数据库管理器在使用另一个容器之前写入一个容器的页数称为*数据块大小*。

缓冲池:

*缓冲池*指的是从磁盘读取高速缓存表和索引数据页时或修改它们时分配给它们的主内存。缓冲池的目的是改进系统性能。从内存存取数据要比从磁盘存取数据快得多；因此，数据库管理器需要读写磁盘（I/O）的次数越少，性能也越好。（可以创建多个缓冲池，虽然在大多数情况下只需要一个。）

因为可以缩短慢速 I/O 所造成的延迟，所以缓冲池的配置是最为重要的调整项目。

第 12 页的图 8 举例说明了缓冲池与容器之间的关系。

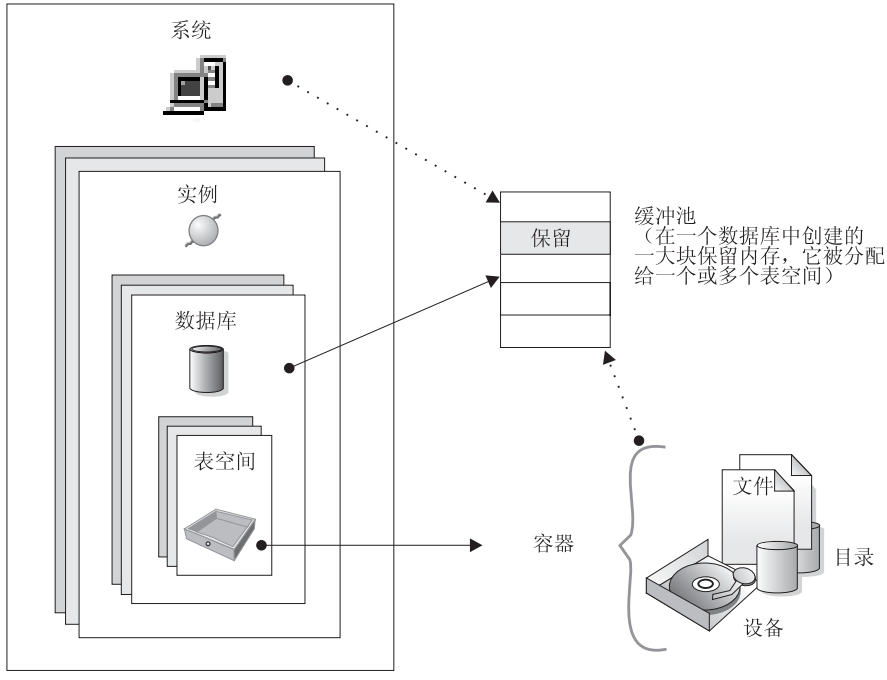

图 8. 缓冲池和容器

相关概念:

- 『Indexes』（*SQL Reference, Volume 1*）
- 『Tables』（*SQL Reference, Volume 1*）
- 『Relational databases』（*SQL Reference, Volume 1*）
- 『Schemas』（*SQL Reference, Volume 1*）
- 『Views』（*SQL Reference, Volume 1*）
- 『Table spaces and other storage structures』（*SQL Reference, Volume 1*）

配置参数

当创建 DB2® 实例或数据库时，将使用缺省参数值创建相应的配置文件。可修改这些参数值以改进性能。

*配置文件*包含特定的参数，这些参数定义诸如分配给 DB2 产品和个别数据库的资源以及诊断级别之类的值。有两种类型的配置文件:

- 每个 DB2 实例的数据库管理器配置文件
- 每个独立的数据库的数据库配置文件

*数据库管理器配置文件*是在创建 DB2 实例时创建的。它包含的参数在实例级影响系统资源，不受任何一个作为该实例一部分的数据库的影响。根据系统的配置，可将这些参数中的许多参数的值更改为非系统缺省值，以改进性能或增加容量。

每个客户机安装也有一个数据库管理器配置文件。此文件包含关于特定工作站的客户机使能器的信息。在可用于服务器的参数中，有一个子集可用于客户机。

数据库管理器配置参数存储在名为 db2systm 的文件中。当创建数据库管理器的实例时，会创建此文件。在基于 UNIX 的环境中，此文件在数据库管理器的实例的 sqllib 子目录中。在 Windows 中，此文件的缺省位置是 sqllib 目录的实例子目录。如果设置 DB2INSTPROF 变量，则文件在 DB2INSTPROF 变量指定的目录的实例子目录中。

在分区数据库环境中，此文件驻留在共享文件系统上，以便所有数据库分区服务器都具有对相同文件的存取权。数据库管理器的配置在所有数据库分区服务器上都相同。

大多数参数影响将分配到数据库管理器的单个实例的系统资源量，或它们基于环境考虑配置数据库管理器的设置和不同通信子系统。另外，存在仅供参考的其它参数，不能更改它们。所有这些参数都具有全局适用性，独立于存储在数据库管理器的该实例下的任何单个数据库。

*数据库配置文件*是在创建数据库时创建的，它驻留在数据库所驻留的地方。每个数据库都有一个配置文件。其参数指定要分配给该数据库的资源量以及其它事项。可更改许多参数的值，以改进性能或增加容量。根据特定数据库中活动类型的不同，可能需要进行不同的更改。

个别数据库的参数存储在名为 SQLDBCON 的配置文件中。此文件与 SQLnnnnn 目录中的数据库的其它控制文件存储在一起，其中 nnnnn 是创建数据库时指定的数字。每个数据库都有它自己的配置文件，并且文件中的大多数参数指定分配给该数据库的资源量。该文件还包含描述信息以及指示数据库的状态的标志。

在分区数据库环境中，对于每个分区数据库都存在一个单独的 SQLDBCON 文件。在每个数据库分区中，SQLDBCON 文件中的值可能相同或不同，但是建议数据库配置参数值在所有分区上相同。

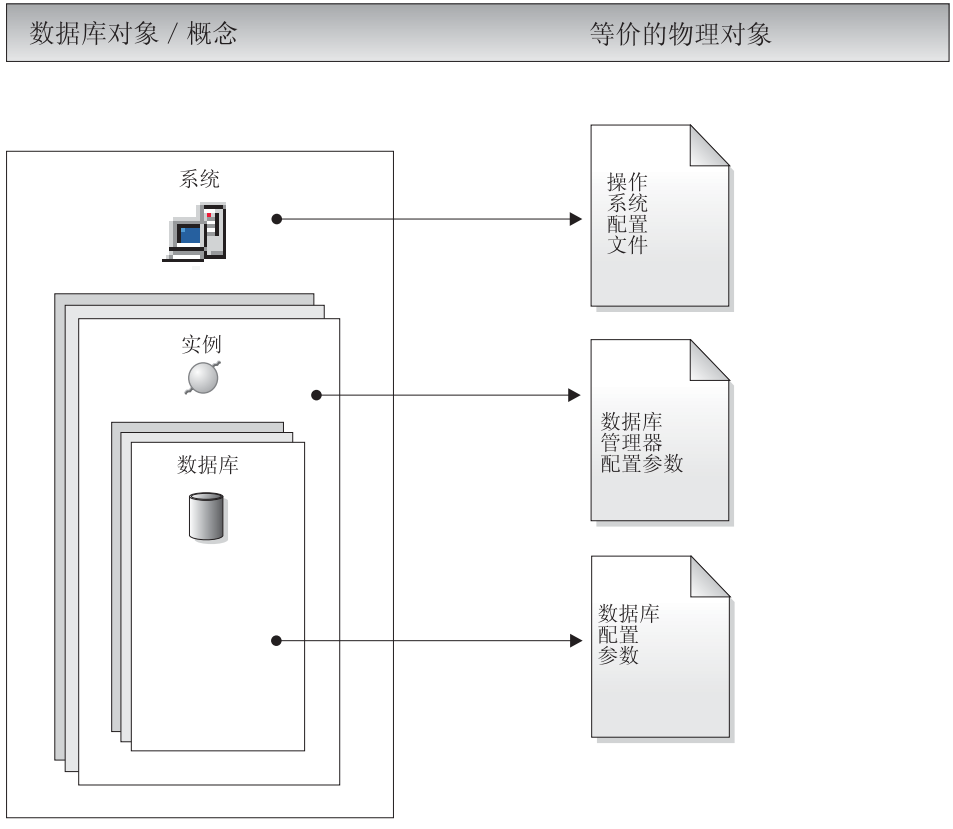

图 9. 配置参数文件

相关概念:

- 『配置参数调整』（*《管理指南: 性能》*）

相关任务:

- 『使用配置参数配置 DB2』（*《管理指南: 性能》*）

数据的商业规则

在任何业务中，数据通常必须符合特定限制或规则。例如，雇员号码必须是唯一的。DB2® 提供了*约束*作为实施这种规则的方法。

DB2 提供了下列约束类型:

- NOT NULL 约束
- 唯一约束

- 主键约束
- 外键约束
- 检查约束

NOT NULL 约束

NOT NULL 约束防止空值进入一个列。

唯一约束

唯一约束确保一组列中的值对于表中的所有行都是唯一的，且不为空。例如，DEPARTMENT 表中的典型唯一约束可以是：部门号是唯一的，且不为空。

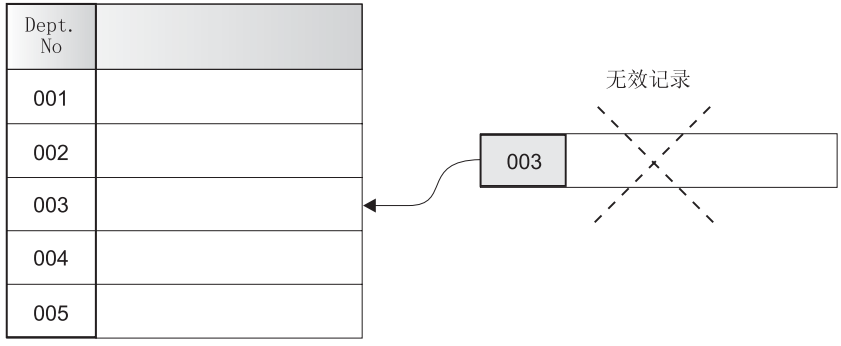

图 10. 唯一约束防止数据重复

数据库管理器在插入和更新操作期间强制执行此约束，以确保数据完整性。

主键约束

每个表都可以有一个主键。主键是与唯一约束具有相同特性的一个列或列的组合。可使用主键和外键约束来定义表之间的关系。

因为主键用来标识表中的一行，所以它应该是唯一的，并且只进行非常少的添加或删除。一个表不能有多个主键，但可以有多个唯一键。主键是可选的，可以在创建或改变表时定义。当导出或重组数据时，主键可以对数据进行排序，所以它们也是有益的。

在下面的表中，DEPTNO 和 EMPNO 是 DEPARTMENT 表和 EMPLOYEE 表的主键。

表 1. DEPARTMENT 表

DEPTNO（主键）	DEPTNAME	MGRNO
A00	Spiffy Computer Service Division	000010

表 1. DEPARTMENT 表 (续)

DEPTNO（主键）	DEPTNAME	MGRNO
B01	Planning	000020
C01	Information Center	000030
D11	Manufacturing Systems	000060

表 2. EMPLOYEE 表

EMPNO（主键）	FIRSTNAME	LASTNAME	WORKDEPT（外键）	PHONENO
000010	Christine	Haas	A00	3978
000030	Sally	Kwan	C01	4738
000060	Irving	Stern	D11	6423
000120	Sean	O'Connell	A00	2167
000140	Heather	Nicholls	C01	1793
000170	Masatoshi	Yoshimura	D11	2890

外键约束

外键约束（也称为引用完整性约束）使您能够定义表间以及表内必需的关系。

例如，典型的外键约束可能规定 EMPLOYEE 表中的每个雇员必须是一个现有部门的成员，该部门在 DEPARTMENT 表中定义。

要建立此关系，应将 EMPLOYEE 表中的部门号定义成外键，并将 DEPARTMENT 表中的部门号定义成主键。

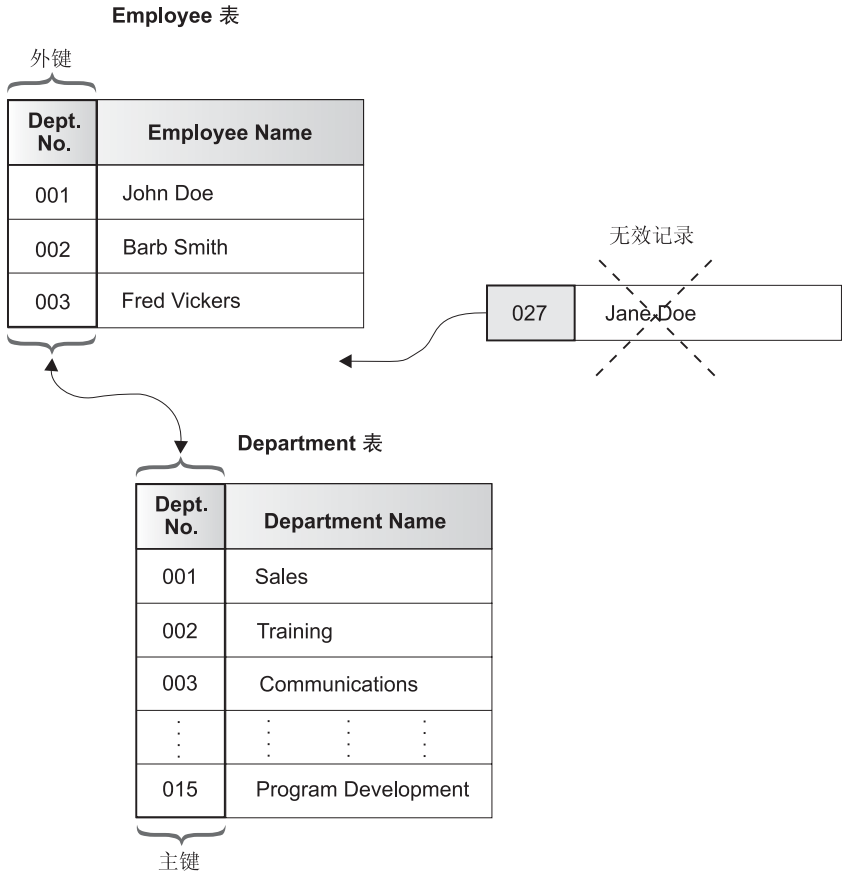

图 11. 外键约束和主键约束定义关键并保护数据

检查约束

检查约束是指定每个表行的一列或多列所允许的值的数据库规则。

例如，在 EMPLOYEE 表中，可将“工作类型”列定义为“推销员”、“经理”或“职员”。有了此约束，在“工作类型”列中具有不同的值的任何记录都是无效的，将被拒绝，并实施关于表中允许的数据类型的规则。

还可在数据库中使用*触发器*。与约束相比，触发器更为复杂，且潜在功能更加强大。它们定义一组操作，这组操作与用于指定基本表的 INSERT、UPDATE 或 DELETE 子句一起执行或由这些子句触发。可使用触发器支持常规格式的完整性或商业规则。例如，在接受订单之前，可使用触发器来检查客户的信用额度，也可以在银行业务应用程序中使用触发器，以便在帐户提款未符合客户的标准提款模式时提高警惕。

相关概念:

- 第 73 页的『约束』
- 第 77 页的『触发器』

开发备份与恢复策略

数据库可能会因为硬件和/或软件故障而不可用。您可能会不时遇到存储问题、电源中断、应用程序故障以及各种需要采取不同恢复措施的故障情况。如果已准备了一个进行过彻底演习的恢复策略，它可保护您的数据免被丢失。在制订恢复策略时，您需要回答的一些问题是：数据库是可恢复的吗？恢复数据库可能花费多长时间？在备份操作之间将花费多长时间？可以为备份副本和归档日志分配多少存储空间？表空间级别的备份是否足够？或是否需要进行完整的数据库备份？

数据库恢复策略应确保在数据库恢复操作需要时，所有信息都可用。它应包括一个进行数据库备份的固定调度表，在分区数据库系统中则包括缩放系统时（添加或删除数据库分区服务或节点时）的备份。完整的策略还应包括恢复命令脚本、应用程序、用户定义函数 （UDF）、操作系统库中的存储过程代码以及装入副本的过程。

在以下几节中还讨论了不同的恢复方法，您将发现最适用于您的商业环境的是哪种恢复方法。

数据库*备份*的概念与其它任何数据备份的概念一样：即，复制一份数据，然后将它存储在另一介质上，以防原始介质发生故障或毁坏。最简单的备份情况涉及关闭数据库（以确保不发生更多的事务），然后简单地对其进行备份。如果数据库已毁坏，就需要重构数据库。

数据库的重构称为*恢复*。*版本恢复*指的是使用备份操作期间创建的映象来复原数据库的先前版本。*前滚恢复*是指复原了数据库或表空间备份映象后，重新应用记录在数据库日志文件中的事务。

*崩溃恢复*是指在完成并落实所有更改（这些更改是一个或多个工作单元的一部分）或事务之前如果发生故障，会自动恢复数据库。这是通过回滚未完成的事务，并完成在发生崩溃时仍在内存中的已落实事务来实现的。

恢复日志文件和恢复历史记录文件是在创建数据库时自动创建的（第 19 页的图 12）。在需要恢复丢失或损坏的数据时，这些日志文件是很重要的。不能直接修改恢复日志文件或恢复历史文件；但是通过使用 PRUNE HISTORY 命令，可以从恢复历史文件中除去条目。还可以使用 *rec_his_retentn* 数据库配置参数来指定将保留恢复历史文件的天数。

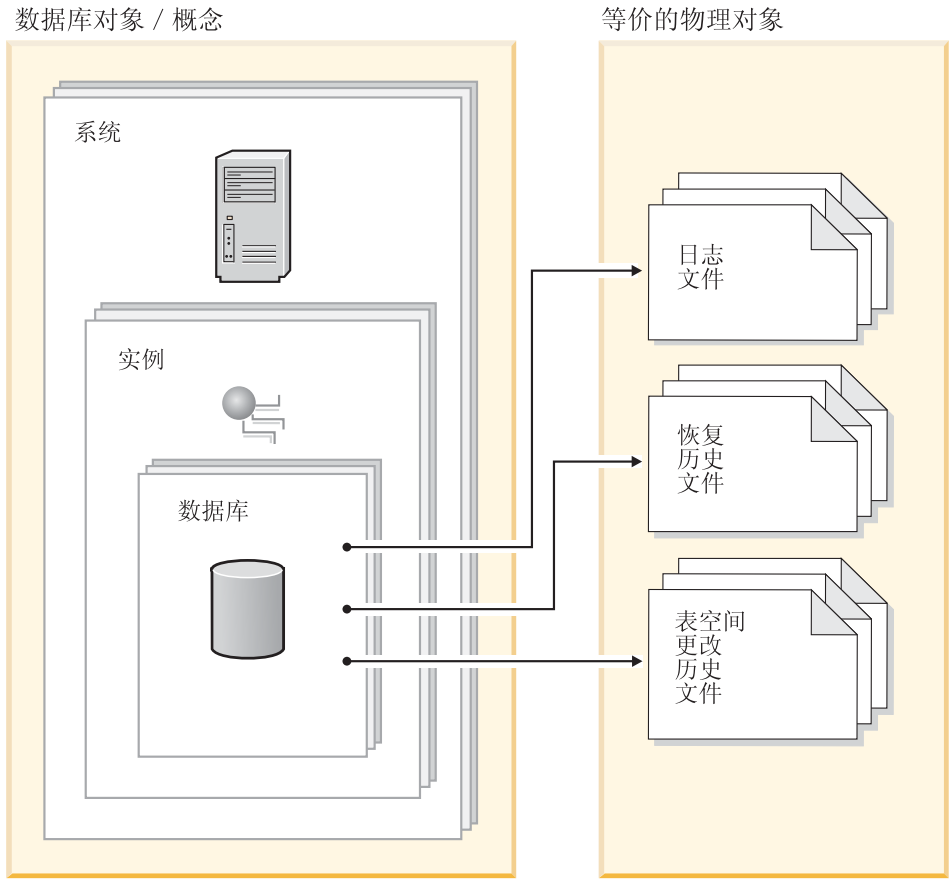

图 12. 恢复日志文件和恢复历史记录文件

每个数据库都包括*恢复日志*，它们用来从应用程序或系统错误中恢复。与数据库备份一起，它们用来将数据库的一致性恢复到出错时的时间点。

*恢复历史文件*包含可用来确定恢复选项的备份信息的总结，如果数据库的所有或部分必须恢复到给定时间点的话。恢复历史记录文件用来跟踪其它操作中与恢复相关的事件，如备份与复原操作。此文件位于数据库目录中。

表空间更改历史文件（它也位于数据库目录中）包含可用来确定哪些日志文件对特定表空间的恢复是必需的信息。

那些很容易重新创建的数据可存储在不可恢复的数据库中。这些数据包括：用于只读应用程序的外部源中的数据以及不常进行更新的表；由于对它们进行的日志记录量较小，如果在复原操作之后还要进行复杂的日志文件文件管理工作和前滚操作，就不太合理了。*不可恢复数据库*的 *logretain* 和 *userexit* 数据库配置参数都

已被禁用。这表明只有保存的日志才是崩溃恢复所必需的。这些日志称为*活动日志*，它们包含当前事务数据。使用*脱机*备份的版本恢复是解决不可恢复数据库的恢复问题的主要手段。（脱机备份表示当备份操作正在进行时，其它应用程序无法使用该数据库。）这样的数据库只能进行脱机复原。它被复原为进行备份映象且前滚恢复不受支持时的状态。

那些不容易重新创建的数据应存储在可恢复的数据库中。这些数据包括在装入后它的源已被破坏的数据、手工输入到表中的数据以及在装入数据库后由应用程序或用户修改的数据。*可恢复数据库*将 *logretain* 数据库配置参数设置为 ″RECOVERY″ 或启用了 *userexit* 数据库配置参数，或两种方法同时使用。活动日志仍可用于崩溃恢复，但您还有*已归档日志*，它包含已落实的事务数据。这样的数据库只能进行脱机复原。它被复原为创建备份映象时的状态。但对于前滚恢复，可通过使用活动日志和已归档日志来将数据库*前滚*（即，越过创建备份映象的时间）至特定的时间点，或者滚动至活动日志末尾。

可恢复数据库备份操作可以脱机执行，也可以*联机*执行（联机表示在备份操作期间其它应用程序可与该数据库连接）。数据库复原和前滚操作必须始终以脱机方式执行。联机备份操作期间，前滚恢复确保捕捉了*所有*表更改，且在复原该备份时重新应用这些更改。

若有一个可恢复数据库，则可备份、复原并将个别表空间前滚，而不必对整个数据库操作。当联机备份表空间时它仍然可用，同时发生的更新记录在日志中。当对表空间执行联机复原或前滚操作时，在该操作完成之前，该表空间本身不可用，但不阻止用户访问其它表空间中的表。

相关概念:

- 『崩溃恢复』（*《数据恢复及高可用性指南与参考》*）
- 『版本恢复』（*《数据恢复及高可用性指南与参考》*）
- 『前滚恢复』（*《数据恢复及高可用性指南与参考》*）
- 『Data Links server file backups』（*Post V8 GA*）
- 『Failure and recovery overview』（*DB2 Data Links Manager Administration Guide and Reference*）

相关参考:

- 『“恢复历史保留周期”配置参数 — rec_his_retentn』（*《管理指南: 性能》*）
- 『DB2 Data Links Manager system setup and backup recommendations』（*DB2 Data Links Manager Administration Guide and Reference*）

安全性

要保护与数据库服务器相关的数据和资源，DB2® 同时使用外部安全性服务与内部存取控制信息。要访问一个数据库服务器，必须在被允许存取数据库数据或资源之前通过一些安全性检查。数据库安全性中的第一步称为*认证*，在此步骤，您必须证明您的身份是真实的。第二步称为*授权*，在此步骤，数据库管理器决定是否允许已经验证的用户执行请求的操作或存取请求的数据。

相关概念:

- 第 21 页的『认证』
- 第 22 页的『授权』

认证

使用 DB2 之外的安全性设施来完成用户认证。此安全性设施可以是操作系统的一部分，可以是一个独立的产品，或者，在特定情况下，可能根本不存在。例如，在基于 UNIX® 的系统上，安全性设施包含在操作系统内部。

安全性设施需要两项来认证用户：用户标识和密码。用户标识向安全性设施标识用户。通过提供正确的密码（只有该用户和安全性设施才知道的信息）来验证该用户的身份（与该用户标识相对应）。

一旦认证:

- 必须使用 SQL 授权名或 *authid* 来向 DB2® 标识该用户。此名称可以与用户标识或一个映射值相同。例如，在基于 UNIX 的系统上，可将符合 DB2 命名约定的一个 UNIX 用户标识转换为大写字体来获得一个 DB2 *authid*。
- 获取用户所属的组的列表。在授权用户时，可使用组成员资格。组是安全性设施实体，它们也必须映射至 DB2 授权名。执行此映射的方法与映射用户标识的方法类似。

DB2 通过下列两种方式之一，使用安全性设施来认证用户:

- DB2 使用成功的安全性系统登录作为身份证明，并允许:
  - 使用本地命令存取本地数据。
  - 使用远程连接，在这种情况下服务器信赖客户机认证。
- DB2 接受用户标识和密码的组合。它使用安全性设施对此组合的成功验证作为身份证明，并允许:
  - 使用远程连接，在这种情况下服务器需要认证的证明。

– 使用操作，在这种情况下用户希望以某个不同于注册时所用的身份来运行命令。

AIX® 上的 DB2 UDB 可将操作系统的失败密码尝试次数计入日志，并检测客户机何时超出允许的登录尝试次数，该值由 LOGINRETRIES 参数指定。

相关概念:

• 第 21 页的『安全性』

授权

授权是 DB2® 获取有关已认证的 DB2 用户的信息的过程，此信息指示该用户可执行的数据库操作，以及可存取的数据对象。执行每个用户请求时，根据所涉及的对象和操作，可能有多个授权检查。

使用 DB2 设施来执行授权。DB2 表和配置文件用于记录与每个授权名相关的许可权。将一个已认证的用户的授权名和此用户所属的那些组的授权名与记录的许可权比较。根据这个比较，DB2 决定是否允许执行请求的存取。

DB2 记录了两种类型的许可权：特权和权限级。*特权*为授权名定义单个许可权，它使用户能够创建或存取数据库资源。特权存储在数据库目录中。*权限级*提供将特权分组的一个方法，并控制更高级别的数据库管理器的维护和实用程序操作。数据库特定的权限存储在数据库目录中；系统权限与组成员资格相关联，并存储在一个给定实例的数据库管理器配置文件中。

组提供了一个简便的方法来对一组用户执行授权，而不必单独对每个用户授予或撤消特权。除非另有指定，否则，组授权名可以用在为了授权而使用授权名的任何地方。通常，对于动态 SQL 和非数据库对象授权（如实例级命令和实用程序），考虑使用组成员资格，但对于静态 SQL，则不考虑使用它。在授予 PUBLIC 特权时则例外：在处理静态 SQL 时要考虑使用它们。DB2 文档中的适当地方提到了组成员资格不适用的特殊情况。

相关概念:

• 『Authorization and privileges』（*SQL Reference, Volume 1*）
• 『特权、权限和授权』（*《管理指南: 实现》*）
• 第 21 页的『安全性』

第 2 章 并行数据库系统

数据分区

DB2® 将数据库管理器扩展至并行多节点环境。*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。数据库分区有时称为节点或数据库节点。

*单分区数据库*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此分区中。对于这种情况，数据库分区组（如果存在的话）不提供任何附加功能。

*分区数据库*是有两个或更多个数据库分区的数据库。表可位于一个或多个数据库分区中。当表在由多个分区组成的数据库分区组中时，它的某些行存储在一个分区中，而其它行存储在其它分区中。

通常，每个物理节点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中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上，所以可在多个物理节点上使用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。

用户交互作用通过一个数据库分区发生，该数据库分区称为该用户的*协调程序节点*。协调程序与应用程序在相同的数据库分区上运行，或者对于远程应用程序，协调程序在该应用程序连接的数据库分区上运行。任何数据库分区都可用作协调程序节点。

DB2 支持分区存储模型，这允许您将数据存储在数据库的多个数据库分区中。这意味着该数据是以物理方式存储在多个数据库分区中，同时仍可以看作数据是位于同一个位置来进行存取。存取一个分区数据库中的数据的应用程序和用户不需要知道该数据的物理位置。

该数据在物理上是分离的，但在逻辑上将它作为一个整体来使用和管理。用户可以选择如何通过说明分区键来将数据分区。用户还可以通过选择表空间和应存储数据的相关数据库分区组，以确定表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。另外，将可更新的分区映射与散列算法一起使用，来指定分区键值至数据库分区的映射，以确定每行数据的位置和检索。因此，可以将大表的工作负荷分布在整个分区数据库上，并将较小的表存储在一个或多个数据库分区上。每个数据库分区具有它存储的数据的本地索引，这可以提高本地数据存取的性能。

并不是必须将所有表都分布到数据库中的所有数据库分区上。DB2 支持*部分撤消群集*，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上时，可考虑的替代方法是使用具体查询表，然后复制这些表。可创建包含所需信息的具体查询表，然后将它复制至每个节点。

并行性

任务（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确定 DB2® 将如何并行执行任务。这些注意事项是相关的，在决定数据库的物理和逻辑设计方案时，应当一起考虑它们。DB2 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，数据库管理器可以利用*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*是指多个应用程序同时查询一个数据库的能力。每个查询都独立于其它查询执行，但是 DB2 同时执行所有查询。DB2 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。一些 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的单个数据库操作（如索引创建、数据库装入或 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。

图 13 显示了一个查询，它被分为可并行运行的四个部分，返回结果的速度比按串行方式运行该查询的速度快得多。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。

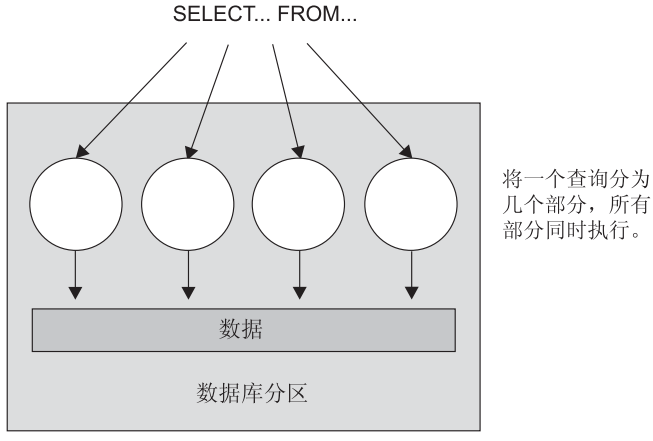

图 13. 分区内并行性

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。一些 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的单个数据库操作（如索引创建、数据库装入或 SQL 查询）细分成多个部分，其中的大部分或全部操作可以在*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 26 页的图 14 显示了一个查询，它被分为可并行运行的四个部分，返回结果的速度比在单个分区上按串行方式运行该查询的速度快得多。

并行度在很大程度上取决于您创建的分区数和您定义数据库分区组的方式。

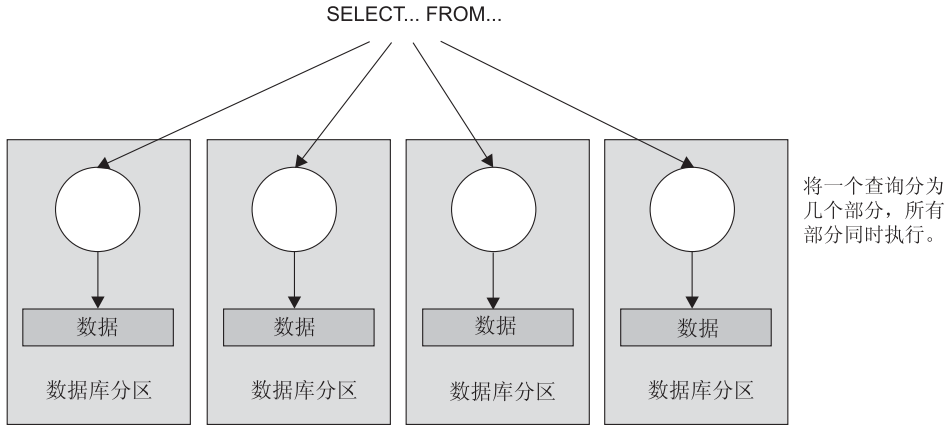

图 14. 分区间并行性

同时进行分区内并行性和分区间并行性

可以同时使用分区内并行性和分区间并行性。此组合提供两种并行性程度，这也使处理查询的速度显著加快。

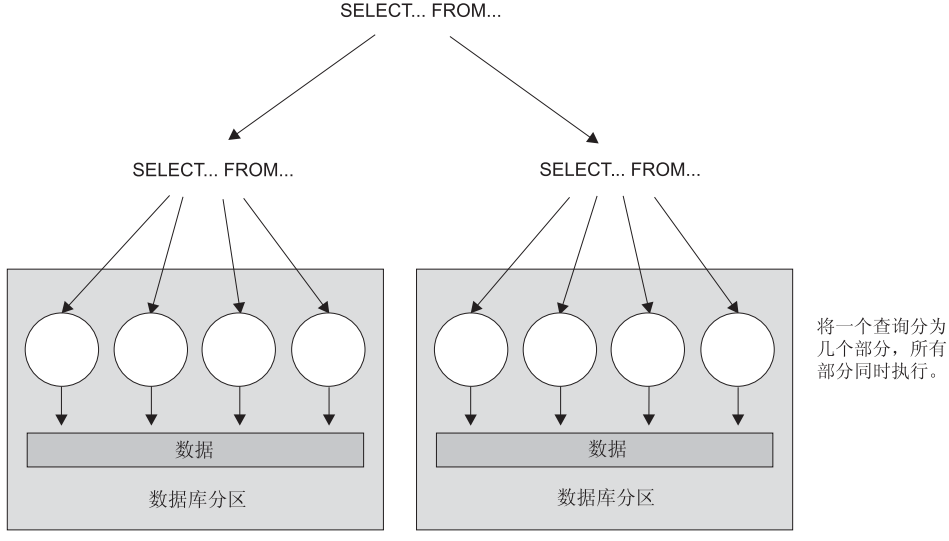

图 15. 同时使用分区间和分区内并行性

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，而实用程序在每个分区内并行执行。

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语法分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 命令通过在该表驻留的每个数据库分区上进行并行调用来利用分区内、分区间或 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 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会涉及到繁重的涉及 I/O 的任务。当执行备份和复原操作时，DB2 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。

相关概念:

- 第 27 页的『分区和处理器环境』

分区和处理器环境

本节对下列硬件环境提供了一个概述:

- 单处理器上的单分区
- 具有多个处理器的单分区 (SMP)
- 多分区配置
  - 具有一个处理器的分区 (MPP)
  - 具有多个处理器的分区（SMP 群集）
  - 逻辑数据库分区（在 DB2® 并行版 AIX® 版的版本 1 中，又称为“多逻辑节点”或 MLN）

下面讨论每种环境的容量和可伸缩性。*容量*是指能存取数据库的用户数和应用程序数。这很大程度上取决于内存、代理进程数、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。

单处理器上的单个分区

此环境由内存和磁盘组成，但仅包含一个 CPU（参见图 16）。可使用许多不同的名称来称呼它，包括独立数据库、客户机／服务器数据库、串行数据库、单处理器系统和单节点或非并行环境。

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。

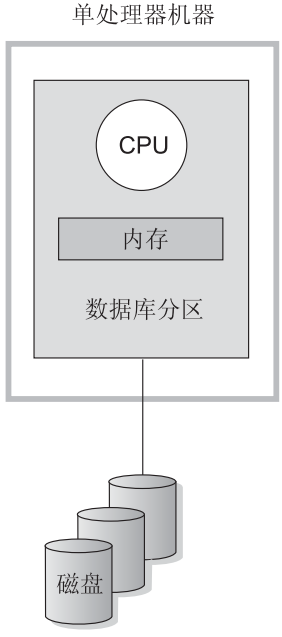

图 16. 单处理器上的单分区

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，则多个 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。无论可以添加的其它组件（如内存或磁盘）如何，随着工作负荷的增加，单个 CPU 可能无法更快地处理用户请求。若已达到最大容量或最大可伸缩性，则可考虑移到一个有多个处理器的单分区系统中。

具有多处理机的单个分区

此环境通常由同一台机器中几个能力相等的处理器组成（参见图 17），称为*对称多处理机（SMP）*系统。象磁盘空间和内存这类资源是*共享的*。

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 还可把单个查询的工作分布在可用的处理器中，以提高处理速度。其它数据库操作，如装入数据、备份和复原表空间以及在现有数据上创建索引，都可以利用多个处理器。

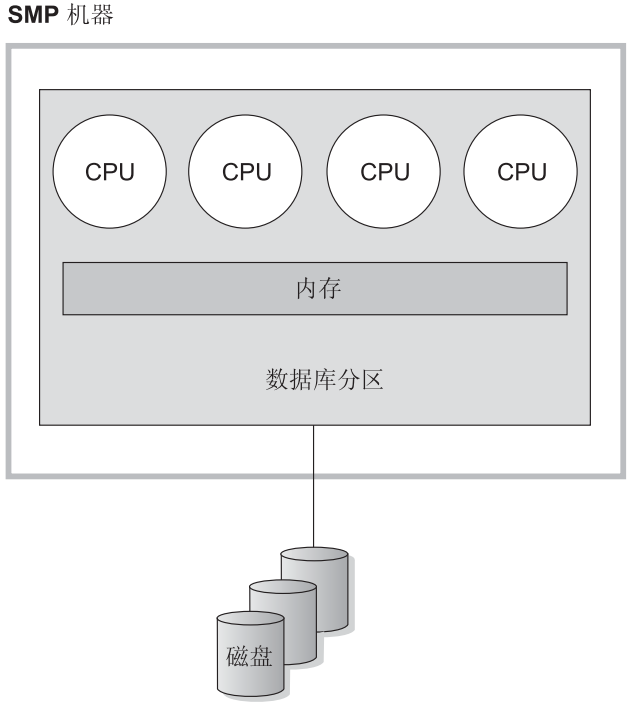

图 17. 单分区数据库对称多处理机系统

容量和可伸缩性

在此环境中可以添加更多的处理器。然而，由于不同的处理器可能会试图存取同一数据，所以随着您公司业务量的增加，此环境也可能遇到限制。利用共享内存和共享磁盘，可有效地共享所有数据库数据。

可以通过增加磁盘数来增加与处理器相关的数据库分区的 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，则多个 I/O 操作可同时执行。

若已达到最大容量或最大可伸缩性，可考虑移到有多个分区的系统中。

多分区配置

可以将一个数据库分布在多个分区中，每个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。本节描述下列分区配置:

- 具有一个处理器的系统上的分区
- 具有多个处理器的系统上的分区
- 逻辑数据库分区

带有一个处理器的分区

在此环境中，有许多数据库分区。每个分区都驻留在它自己的机器上，而且它有自己的处理器、内存和磁盘（第 31 页的图 18）。所有机器通过通信设施连接在一起。可使用许多不同的名称来称呼此环境，包括群集、单处理器群集、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 环境消除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逻辑上的整体性，但它实际上分布于多个分区中。数据是分区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。

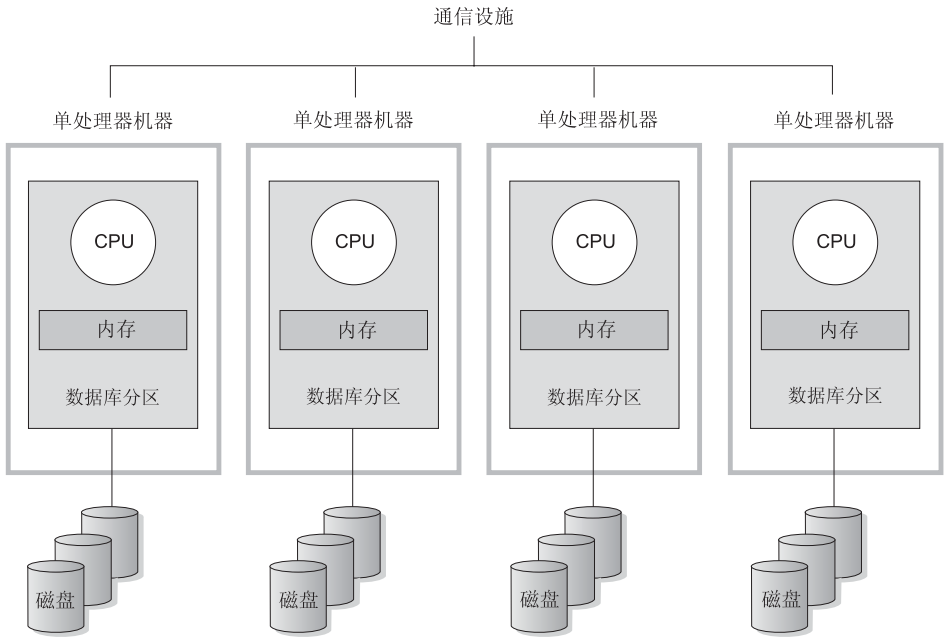

图 18. 大规模并行处理系统

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（节点）。在一些平台（如 RS/6000® SP）上，最大数目是 512 个节点。然而，在管理很多机器和实例时，可能存在实际的限制。

若已达到最大容量或最大可伸缩性，可考虑移到每个分区都有多个处理器的系统中。

带有多个处理器的分区

每个分区具有单处理器的配置的替代配置是，一个分区具有多个处理器的配置。这称为 *SMP 群集*（第 32 页的图 19）。

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个处理器的单个分区中执行。亦即一个查询可以用并行方式在多个分区中执行。

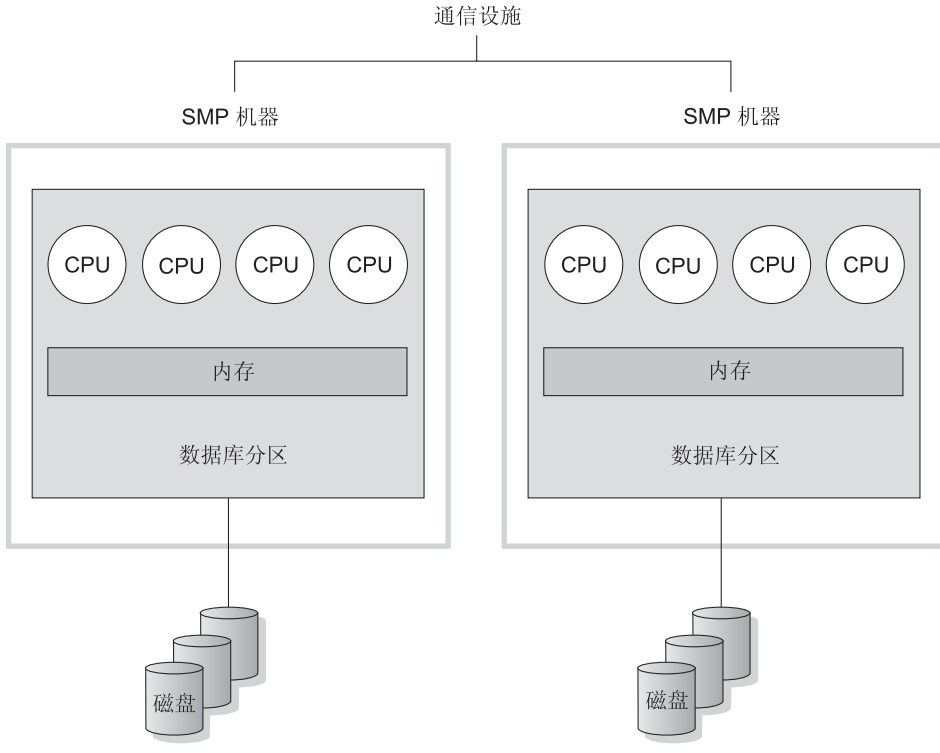

图 19. SMP 群集

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。在多个逻辑分区上运行的多个数据库管理器可以比一个单数据库管理器更能充分利用可用的资源。第 33 页的图 20 举例说明通过添加更多的分区可以在一台 SMP 机器上获得更大的可伸缩性；特别是对那些具有许多处理机的机器而言更是如此。通过对数据库分区，可以分别对每个分区进行管理和恢复。

大的 **SMP** 机器

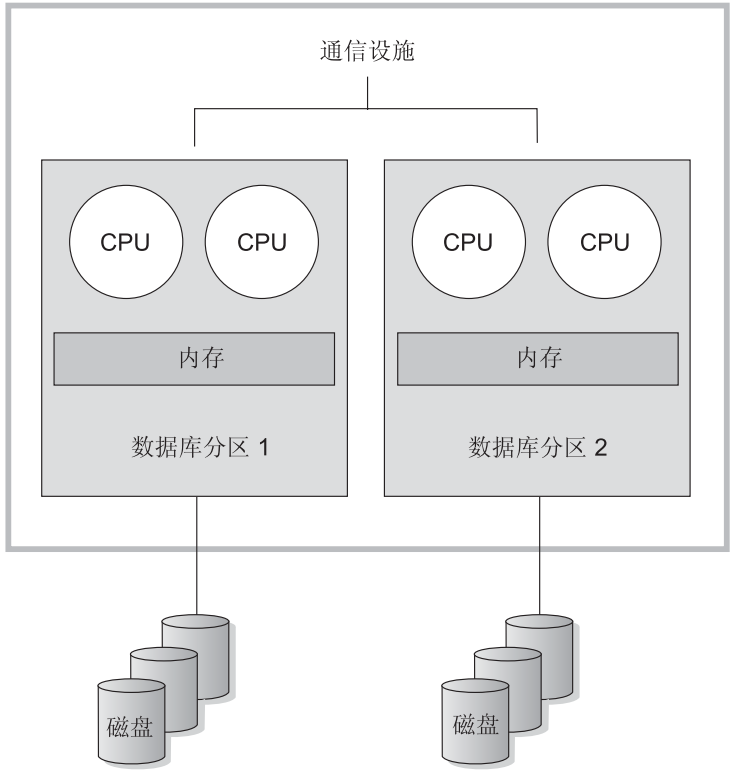

图 20. 分区数据库对称多处理机系统

第 34 页的图 21 举例说明可以扩大在图 20 中显示的配置以增强处理能力的事实。

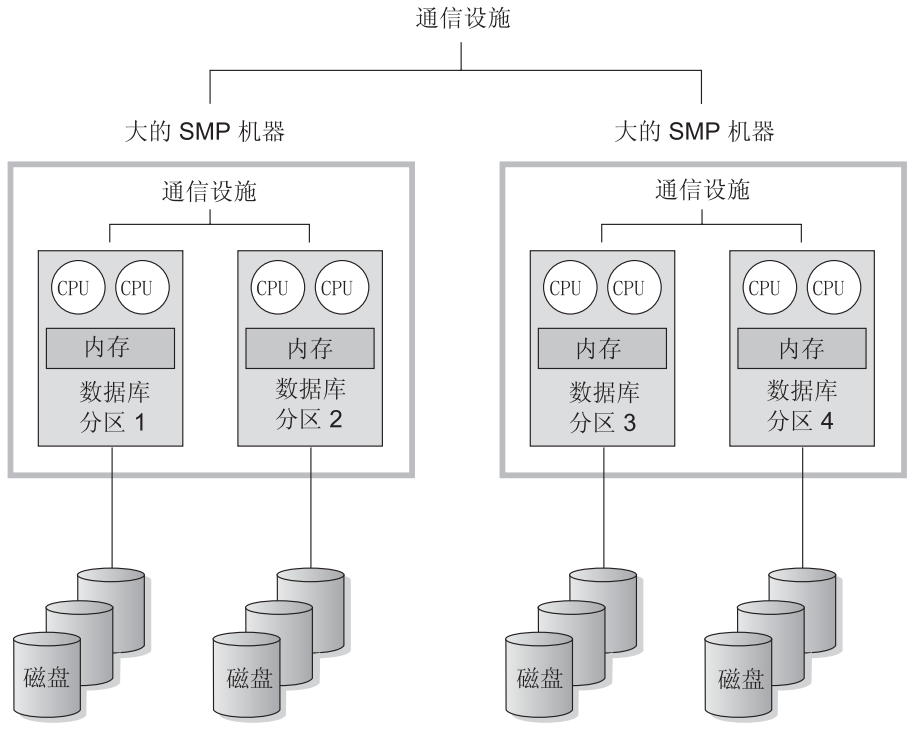

图 21. 分区数据库对称多处理机系统群集在一起

注: 两个以上的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恢复策略。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总结

下表总结了最适合于利用各种硬件环境的并行性的类型。

表 3. 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
单分区，单处理器	是	否 (1)	否
单分区，多个处理器（SMP）	是	是	否
多分区，一个处理器（MPP）	是	否 (1)	是
多个分区，多个处理器（SMP 群集）	是	是	是
逻辑数据库分区	是	是	是

表 3. 每种硬件环境中的可能并行性类型 (续)

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
注: (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，特别是在执行的查询并未充分利用 CPU 时（例如，若它们受 I/O 约束）。			

相关概念:

- 第 24 页的『并行性』

第 3 章 关于数据入库

什么是数据入库?

包含*操作数据*（运行公司日常事务的数据）的系统中有对商务分析员有用的信息。例如，分析员可以使用在年度的哪段时间在哪些地区销售了哪些产品之类的信息，来查找异常情况或制定将来的销售计划。

但是，当分析员直接存取操作数据时，可能会产生几个问题:

- 分析员可能没有查询可运作数据库的专门知识。例如，查询 IMS™ 数据库需要使用专用类型的数据操作语言的应用程序。具有查询可运作数据库的专门知识的程序员通常全职从事维护数据库及其应用程序的工作。
- 性能对于许多可运作数据库，如银行的数据库是至关重要的。系统不能处理用户进行特定查询的情况。
- 操作数据通常不是商务分析员所使用的最佳格式。例如，对于分析员来说，按产品、地区和季节汇总的销售数据比原始数据要有用得多。

数据入库解决了这些问题。在*数据入库*中，可创建*信息数据*的仓库。信息数据是从操作数据中抽取并进行变换以用于决策的数据。例如，数据入库工具可以从可运作数据库复制所有销售数据、清理该数据、执行计算以汇总该数据、然后将汇总的数据写入独立于操作数据的数据库的目标中。用户可以查询该独立数据库（*仓库*），而不影响可运作数据库。

数据仓库对象

以下各节描述将用于创建和维护数据仓库的对象。

主题区

主题区标识与业务的逻辑区相关的进程，并将进程分组。例如，若正构建市场营销和销售数据的仓库，则定义“销售”主题区和“市场营销”主题区。然后在“销售”主题区下添加与销售相关的进程。类似地，在“市场营销”主题区下添加与市场营销数据相关的定义。

仓库源

仓库源标识将为仓库提供数据的表和文件。“数据仓库中心”使用仓库源中的规范来存取数据。这些源几乎可以是与您的网络连接的任何关系或非关系源（表、视图、文件或预定义别名）、SAP R/3 源或 WebSphere® Site Analyzer 源。

仓库目标

仓库目标是包含已变换的数据的数据库表或文件。类似于仓库源，用户可以使用仓库目标向其它仓库目标提供数据。中央仓库可以向部门的服务器提供数据，或仓库中的主事实表可以向总结表提供数据。

仓库代理进程和代理点

仓库代理进程管理数据源和目标仓库之间的数据流。
AIX、Linux、iSeries、z/OS、Windows® NT、Windows 2000 和 Windows XP 操作系统及 Solaris 操作环境上都提供了仓库代理进程。代理进程使用“开放式数据库连接”（ODBC） 驱动程序或 DB2® CLI 来与不同的数据库通信。

有几种代理进程可以处理源与目标仓库之间的数据传送。使用的代理进程数取决于现有的连接配置和计划移至仓库的数据量。若需要同一代理进程的多个进程同时运行，则会生成该代理进程的附加实例。

代理进程可以是本地的或远程的。本地仓库代理进程是与仓库服务器安装在同一工作站上的代理进程。远程仓库代理进程是安装在与仓库服务器连接的另一工作站上的代理进程。

代理点是安装代理软件的工作站的逻辑名。代理点名称与 TCP/IP 主机名不同。单个工作站只能有一个 TCP/IP 主机名。然而，可以在单个工作站上定义多个代理点。逻辑名标识每个代理点。

缺省代理点，名为“缺省 DWC 代理点”，是在仓库控制数据库初始化期间，“数据仓库中心”定义的本地代理进程。

进程和步骤

进程包含为特定仓库用途执行数据变换和移动的一系列步骤。通常，进程将源数据移至仓库。然后，聚集并汇总该数据以供仓库使用。进程可以产生单个文本表和一组总结表。进程还可以执行某种特定类型的数据变换。

步骤是仓库内单个操作的定义。通过使用 SQL 语句或调用程序，步骤定义如何移动和变换数据。当运行步骤时，可能在仓库源和仓库目标之间进行数据传送，或对该数据进行任何变换。

步骤是“数据仓库中心”中的逻辑实体，它定义:

- 与其源数据的链接。
- 输出表或文件的定义和与输出表或文件的链接。
- 填充输出表或文件的机制（SQL 语句或程序）和定义。
- 填充输出表或文件所依据的处理选项和调度表。

假设想要“数据仓库中心”执行下列任务:

1. 从不同的数据库抽取数据。
2. 将数据转换为单一格式。
3. 将数据写入数据仓库中的表中。

应创建包含多个步骤的进程。每个步骤执行单独的任务，如从数据库抽取数据或将其转换为正确的格式。可能需要创建多个步骤，来彻底变换和格式化数据并将它放入最终的表中。

当步骤或进程运行时，它可能以下列方式影响目标:

- 将仓库目标中的所有数据替换为新数据
- 将新数据追加到现有数据
- 追加数据的一个单独版本
- 更新现有数据

可以按需求运行步骤，或者可以调度步骤在设置的时间运行。可以调度步骤只运行一次，或者调度它反复运行，如在每个星期五运行。还可以调度步骤按顺序运行，以便步骤紧接着上一个步骤运行。可以调度步骤根据另一个步骤的完成情况（成功或失败）来运行。若调度进程，该进程中的第一个步骤会在调度的时间运行。

下列各节描述将在“数据仓库中心”中发现的各种步骤。

SQL 步骤

SQL 步骤有两种类型。SQL Select 和 Insert 步骤使用 SQL SELECT 语句从仓库源中抽取数据，并生成 INSERT 语句以将数据插入到仓库目标表中。SQL Select 和 Update 步骤使用 SQL SELECT 语句从仓库源中抽取数据，并更新仓库目标表中的现有数据。

程序步骤

程序步骤有多种类型: “DB2 iSeries™ 版”程序、“DB2 z/OS™ 版”程序、“DB2 UDB 版”程序、Visual Warehouse™ 5.2 DB2 程序、OLAP Server 程序、“文件”程序和“复制”。这些步骤运行预定义的程序和实用程序。

变换器步骤

变换器步骤是存储过程和用户定义函数，它们指定可用来变换数据的统计变换器或仓库变换器。可使用变换器清除、反转和旋转数据；生成主键和周期表以及计算各种统计信息。

在变换器步骤中，指定统计变换器或仓库变换器中的一个。当运行该进程时，变换器步骤将数据写入一个或多个仓库目标。

用户定义程序步骤

用户定义程序步骤是“数据仓库中心”内的逻辑实体，它表示想要“数据仓库中心”启动的特定于企业的变换。因为每个企业都有独特的数据变换需求，所以企业可以选择编写他们自己的程序步骤，或使用一些工具，象 ETI 或 Vality 提供的那些工具。

例如，可以编写执行下列功能的用户定义程序:

1. 从表中导出数据。
2. 处理该数据。
3. 将该数据写入临时输出资源或仓库目标。

连接器

下列连接器是“DB2 仓库管理器”附带包括的，但是必须单独购买。这些连接器可以帮助您从电子商务库中抽取数据和元数据。

- DB2 仓库管理器 SAP R/3 连接器
- DB2 仓库管理器 Web 连接器

通过 SAP R/3 连接器，可将抽取的数据添加到数据仓库中、使用“DB2 数据仓库中心”来变换它或者使用 DB2 工具或其它供应商的工具来分析它。通过 Web 连接器，可以将 IBM® WebSphere Site Analyzer 中的“单击流”数据引入数据仓库中。

仓库任务

创建数据仓库涉及下列任务:

- 标识源数据（或操作数据）并定义它用作仓库源。
- 创建数据库以用作仓库并定义仓库目标。
- 为将在仓库中定义的进程组定义主题区。
- 通过在进程中定义步骤，指定如何移动源数据并将它变换为用于仓库数据库的格式。
- 测试定义的步骤并调度它们自动运行。

- 通过使用“正在运行”笔记本定义安全性和监视数据库的使用情况来管理仓库。
- 如果您具有“DB2® 仓库管理器”，则为仓库中的数据创建信息目录。信息目录是包含商业元数据的数据库。商业元数据帮助用户在该组织中标识和查找可供使用的数据和信息。可以将“数据仓库元数据”发布到信息目录中。可以搜索信息目录以确定仓库中提供了哪些数据。
- 为仓库中的数据定义星型模式模型。星型模式是一个专门的设计，由多个维度表和一个事实表组成，维度表描述业务的各个方面，事实表包含该业务的事实或计量。例如，对于制造业公司，维度表可以是产品、市场和时间。事实表包含有关每个地区按季节订购的产品的事务信息。

第 2 部分 数据库设计

第 4 章 逻辑数据库设计

设计数据库的目标是为您的环境建立一种表示，它易于理解且可以扩充。此外，您还希望数据库设计有助于维护数据的一致性和完整性。要达到此目的，可以设计一个减少冗余度并消除在更新数据库期间可能发生的异常的数据库。

数据库设计不是一个线性过程；进行该设计时，可能还会重新执行那些步骤。

要在数据库中记录的内容

数据库设计的第一步是标识要存储在数据库表中的数据的类型。一个数据库包括一个组织或企业中的*实体*及其它们之间的关系的信息。在关系数据库中，将*实体*表示为*表*。

一个*实体*是您想要存储其信息的一个人、一个对象或一个概念。样本表中描述的某些实体是雇员、部门和项目。

在 Employee 样本表中，实体“雇员”具有*属性*或特性，如雇员号、姓名、工作部门和薪水额。那些特性显示为 EMPNO、FIRSTNME、LASTNAME、WORKDEPT 和 SALARY *列*。

实体“雇员”的*具体值*由一个雇员的所有列中的值组成。每个雇员有一个唯一的雇员号（EMPNO），该号码可用来标识实体“雇员”的具体值。一个表中的每一行表示一个实体或关系的一个具体值。例如，在下表中，第一行中的值描述名为 Haas 的雇员。

表 4. 雇员实体及其属性的具体值

EMPNO	FIRSTNME	LASTNAME	WORKDEPT	JOB
000010	Christine	Haas	A00	President
000020	Michael	Thompson	B01	Manager
000120	Sean	O’Connell	A00	Clerk
000130	Dolores	Quintana	C01	Analyst
000030	Sally	Kwan	C01	Manager
000140	Heather	Nicholls	C01	Analyst
000170	Masatoshi	Yoshimura	D11	Designer

随着支持非传统数据库应用程序（如多媒体）的需要日益增长。可能要考虑一些属性以便支持多媒体对象，如文档、视频或混合媒体、图像和语音。

在一个表中，某一行的每一列都以某种方式与该行的所有其它列相关。样本表中表达的一些关系如下:

- 雇员被分配至部门
  - Dolores Quintana 被分配至部门 C01
- 雇员从事一个工作
  - Dolores 从事分析员的工作
- 雇员管理部门
  - Sally 管理部门 C01

“雇员”和“部门”是实体；Sally Kwan 是“雇员”的一个具体值的一部分，而 C01 是“部门”的一个具体值的一部分。相同的关系应用于一个表的每一行中的相同列。例如，一个表的某行表示的关系是 Sally Kwan 管理部门 C01；而另一行表示的关系是 Sean O’Connell 是部门 A00 中的职员。

一个表中包含的信息取决于要表示的关系、需要的灵活度和期望的数据检索速度。

除了标识企业中的实体关系外，还需标识其它类型的信息，如应用于该数据的商业规则。

相关概念:

- 第 46 页的『数据库关系』
- 第 49 页的『列定义』

数据库关系

可以在数据库中定义几种类型的关系。考虑雇员和部门之间的可能关系。

一对多和多对一关系

一个雇员只能在一个部门工作；对于雇员，此关系是*单值的*。另一方面，一个部门可有许多雇员；对于部门，此关系是*多值的*。雇员（单值的）和部门（多值的）之间的关系是*一对多*的关系。

要为每个一对多和每个多对一关系定义表:

1. 将关系的“多”方是相同实体的所有关系分组。
2. 为该组中的所有关系定义单个表。

在以下示例中，第一个和第二个关系的“多”方是“雇员”，因此我们定义一个雇员表 EMPLOYEE。

表 5. 多对一关系

实体	关系	实体
雇员	被分配到	部门
雇员	从事	工作
部门	报告给	（管理）部门

在第三个关系中，“部门”是“多”方，因此我们定义一个部门表 DEPARTMENT。

下面显示如何在表中表示这些关系:

EMPLOYEE 表:

EMPNO	WORKDEPT	JOB
000010	A00	President
000020	B01	Manager
000120	A00	Clerk
000130	C01	Analyst
000030	C01	Manager
000140	C01	Analyst
000170	D11	Designer

DEPARTMENT 表:

DEPTNO	ADMRDEPT
C01	A00
D01	A00
D11	D01

多对多关系

两个方向都是多值的关系是多对多关系。一个雇员可以开发多个项目，而一个项目可以有多个雇员。问题“Dolores Quintana 在做什么？”和“谁在开发项目 IF1000？”都得到多个答案。在一个表中，对每个实体（“雇员”和“项目”）使用一列，可以表示多对多关系，如下面示例所示。

下面显示如何在一个表中表示多对多关系（一个雇员可以参与多个项目，而一个项目可以有多个雇员在参与）：

雇员活动（EMP_ACT）表:

EMPNO	PROJNO
000030	IF1000
000030	IF2000
000130	IF1000
000140	IF2000
000250	AD3112

一对一关系

一对一关系在两个方向都是单值的。一个经理管理一个部门；一个部门只有一个经理。问题“谁是部门 C01 的经理？”和“Sally Kwan 管理哪个部门？”都有单个答案。可将该关系指定给 DEPARTMENT 表或 EMPLOYEE 表。因为所有部门都有经理，但不是所有雇员都是经理，因此将经理添加至 DEPARTMENT 表是最合乎逻辑的，如以下示例中所示。

下面显示如何在一个表中表示一对一关系:

DEPARTMENT 表:

DEPTNO	MGRNO
A00	000010
B01	000020
D11	000060

确保相等值表示同一实体

您可以有多个表，它们描述同一组实体的属性。例如，EMPLOYEE 表显示雇员被分配至的部门的编号，DEPARTMENT 表显示分配到每个部门的经理。要同时检索这两组属性，可用匹配的列将这两个表连接在一起，如以下示例中所示。WORKDEPT 和 DEPTNO 中的值表示相同的实体，并表示 DEPARTMENT 和 EMPLOYEE 表之间的*连接路径*。

DEPARTMENT 表:

DEPTNO	DEPTNAME	MGRNO	ADMRDEPT
D21	Administration Support	000070	D01

EMPLOYEE 表:

EMPNO	FIRSTNAME	LASTNAME	WORKDEPT	JOB
000250	Daniel	Smith	D21	Clerk

当从多个表检索一个实体的信息时，确保相等的值表示相同的实体。连接列可有不同的名称（如上一个示例中的 WORKDEPT 和 DEPTNO），也可以有相同的名称（如 Department 表和 Project 表中的 DEPTNO 列）。

相关概念:

- 第 45 页的『要在数据库中记录的内容』
- 第 49 页的『列定义』

列定义

要在一个关系表中定义一列:

1. 为该列选择一个名称。

   一个表的每一列都必须具有对于该表唯一的名称。

2. 说明对该列有效的数据类型。

   *数据类型*和*长度*指定对该列有效的数据类型和最大长度。可以从数据库管理器提供的那些类型中选择数据类型；也可以选择创建自己的用户定义类型。

   数据类型的示例有: 数字、字符串、双字节（或图形）字符串、日期时间和二进制字符串。

   大对象（LOB）数据类型支持多媒体对象，如文档、视频、图像和语音。这些对象是使用下列数据类型实现的:

   - *二进制大对象*（BLOB）字符串。BLOB 的示例是雇员的相片、语音和视频。
   - *字符大对象*（CLOB）字符串，它的字符序列可以是单字节字符或多字节字符，或这两者的组合。CLOB 的一个示例是雇员的简历。
   - *双字节字符大对象*（DBCLOB）字符串，它的字符序列是双字节字符。DBCLOB 的一个示例是日语简历。

用户定义类型（UDT），是从现有类型派生的一种类型。您可能需要定义从现有类型派生并与现有类型共享特性，但仍被视为单独的和不兼容的类型。

*结构化类型*是一种用户定义类型，其结构在数据库中定义。它包含一系列*属性*，每个属性都有一个数据类型。可将结构化类型定义为称为*超类型*的另一种结构化类型的*子类型*。子类型继承它的超类型的所有属性，并可定义附加属性。与一个公共超类型相关的结构化类型集合称为*类型层次结构*，没有任何超类型的超类型称为该类型层次结构的*根类型*。

可以将结构化类型用作表或视图的类型。结构化类型的属性的名称和数据类型，与对象标识符一起，成为这个*类型表*或*类型视图*的列的名称和数据类型。可以将类型表或类型视图的行当作结构化类型的实例。

不能将结构化类型用作表或视图的列的数据类型。也不支持将整个结构化类型的实例作为应用程序中的一个主机变量来检索。

*引用类型*是结构化类型的辅助类型。类似于单值类型，引用类型是一个标量类型，它与一种内置数据类型共享一个公共的表示法。在类型层次结构中的所有类型都共享这同一个表示法。引用类型的表示法是在创建类型层次结构的根类型时定义的。当使用引用类型时，将结构化类型指定为该类型的参数。此参数称为该引用的*目标类型*。

引用的目标始终是类型表或视图中的行。当使用引用类型时，可能需要定义*作用域*。作用域标识一个表（称为*目标表*）或视图（称为*目标视图*），它包含引用值的目标行。目标表或视图的类型必须与引用类型的目标类型相同。定义了作用域的引用类型的一个实例会唯一地标识类型表或类型视图中的一行，该行称为它的*目标行*。

由于许多原因，可能会使用*用户定义函数*（UDF），包括调用那些允许在用户定义类型之间比较或转换的例程。UDF 扩展并增强了内部 SQL 函数提供的支持，并可在可使用内置函数的任何地方使用。有两种类型的 UDF:

- 外部函数，它是用一种编程语言编写的
- 有源函数，它将用于调用其它 UDF

例如，两个数字数据类型是“欧洲鞋码”和“美国鞋码”。这两种类型表示的都是鞋码，但是它们不兼容，因为度量基数不同且不能比较。可调用用户定义函数将一种鞋码转换为另一种。

3. 说明哪些列可能需要缺省值。

   某些列不可能在所有行的值都是有意义的，因为:

   - 列值不适用于该行。

例如，包含雇员中间姓名首字母的一列不适用于没有中间姓名首字母的雇员。

- 值适用，但尚未知道该值。

  例如，MGRNO 列可能包含无效的经理号，因为该部门的上一个经理已调离，新经理尚未任命。

在这两种情况下，可以选择允许 NULL 值（指示列值是未知的或不适用的特殊值），或允许数据库管理器或应用程序指定的非 NULL 缺省值。

主键

*键*是一组可用来标识或存取特定行的列。在表、索引或引用约束的描述中标识键。同一个列可以是多个键的一部分。

*唯一键*是约束为其任何两个值都不相等的键。唯一键的列不能包含空值。例如，可将雇员号码列定义为唯一键，因为该列中的每个值只标识唯一一个雇员。任何两个雇员不能有相同雇员号码。

用来强制键唯一性的机制称为*唯一索引*。一个表的唯一索引是一列或若干列的有序集合，而每个值标识（在功能上确定）这些列的唯一行。唯一索引可以包含空值。

*主键*是在一个表上定义的唯一键中的一个，而且该键被选为最重要的键。一个表上只能有一个主键。

会自动为主键创建*主索引*。数据库管理器使用主索引来有效地存取表行，且主索引允许数据库管理器强制主键的唯一性。（也可以在非主键列上定义索引，以便在处理查询时高效率地存取数据。）

若一个表没有“自然”的唯一键，或者到达顺序是用于区分唯一行的方法，则使用时间戳记作为键的一部分可能有帮助。

某些样本表的主键是:

表	键列
Employee 表	EMPNO
Department 表	DEPTNO
Project 表	PROJNO

以下示例显示 PROJECT 表的一部分，包括其主键列。

表 6. PROJECT 表上的主键

PROJNO（主键）	PROJNAME	DEPTNO
MA2100	Weld Line Automation	D01
MA2110	Weld Line Programming	D11

若一个表中的每一列都包含重复的值，则不能只用一列来定义主键。具有多列的键是*组合键*。列值的组合应定义一个唯一实体。若定义组合键不太容易，可以考虑创建具有唯一值的新列。

以下示例显示包含多列的一个主键（组合键）：

表 7. EMP_ACT 表上的组合主键

EMPNO（主键）	PROJNO（主键）	ACTNO（主键）	EMPTIME	EMSTDATE（主键）
000250	AD3112	60	1.0	1982-01-01
000250	AD3112	60	.5	1982-02-01
000250	AD3112	70	.5	1982-02-01

标识候选键列

要标识候选键，选择定义唯一实体的最少数目的列。可能有多个候选键。在表 8 中，有多个候选键。EMPNO、PHONENO 和 LASTNAME 这三列都唯一地标识该雇员。

表 8. EMPLOYEE 表

EMPNO（主键）	FIRSTNAME	LASTNAME	WORKDEPT（外键）	PHONENO
000010	Christine	Haas	A00	3978
000030	Sally	Kwan	C01	4738
000060	Irving	Stern	D11	6423
000120	Sean	O’Connell	A00	2167
000140	Heather	Nicholls	C01	1793
000170	Masatoshi	Yoshimura	D11	2890

从一组候选键中选择一个主键的标准应是持久性、唯一性和稳定性。

- 持久性表示总是存在每一行的主键值。

- 唯一性表示每一行的键值与所有其它行不同。
- 稳定性表示主键值始终不变。

在该示例的三个候选键中，只有 EMPNO 全部满足这些标准。一个雇员在进入一家公司时，可能没有电话号码。姓可能改变，尽管它们在某个时候可能是唯一的，但不能保证始终如此。雇员号码列是主键的最佳选择。一个雇员只被分配一个唯一的号码一次，而且只要该雇员在该公司内任职，通常不会更新该号码。因为每个雇员必须有一个号码，所以雇员号码列中的值是持久的。

相关概念:

- 第 53 页的『身份列』

身份列

*身份列*向 DB2® 提供了一种为表中的每一行自动生成唯一数值的方法。一个表只能有一个定义了标识属性的列。身份列的示例包括订单号码、雇员号码、证券号码和事故号码。

身份列的值可以“始终生成”或“在缺省情况下生成”。

- DB2 保证定义为*始终生成*的身份列是唯一的。它的值总是由 DB2 生成；不允许应用程序提供显式的值。
- 定义为*在缺省情况下生成*的身份列向应用程序提供了显式地为身份列提供一个值的方法。若未给出任何值，则 DB2 生成一个，但在此情况下，不能保证该值的唯一性。DB2 只对它生成的一组值保证唯一行。“在缺省情况下生成”用于数据传播（复制现有表的内容），或用于一个表的卸装和重新装入。

身份列特别适合于生成唯一主键值这一任务。应用程序可使用身份列来避免当一个应用程序在数据库外部生成它自己的唯一计数器时可能会导致的并行性和性能问题。例如，一种常见的应用程序级实现是维护一个只有一行的表，它包含一个计数器。每个事务都锁定此表，增大该数字，然后落实；即，每次只有一个事务可以增大计数器。相反，若通过身份列维护该计数器，因为事务不锁定该计数器，所以可以获得更高级别的并行性。一个未落实的已增大该计数器的事务不会阻止后续事务也增大该计数器。

身份列的计数器的增大（或减小）独立于事务。若给定的事务两次增大一个标识计数器，则该事务可能会在生成的两个数字之间看到一个间隙，因为当前可能有其它事务正在增大同一标识计数器（即，将行插入同一个表中）。若一个应用程序必须要有连续范围的数字，则该应用程序应对带有身份列的表进行互斥锁定。因为会造成丢失并行性，所以必须对此决定作权衡。此外，有可能因为生成身份列

的值的事务已回滚，或因为存放值系列的数据库在指定所有存放的值之前被停用，从而导致给定的身份列出现为数字之间生成有间隙。

身份列生成的序号具有下列附加特性:

- 值可以是任何小数位为零的精确数字数据类型；即，小数位为零的 SMALLINT、INTEGER、BIGINT 或 DECIMAL。（单精度和双精度浮点类型被认为是适当的数字数据类型。）
- 连续值之间可以有任何指定的整数增量。缺省增量是 1。
- 身份列的计数器值是可恢复的。若发生故障，则从日志重新构造计数器值，因此可以保证继续生成唯一的值。
- 可以将身份列值存入高速缓存，以获得更好的性能。

相关概念:

- 第 51 页的『主键』

规范化

*规范化*帮助消除表数据中的冗余和不一致。它是将表精简为一组列的过程，在这组列中，所有非键列都依赖于主键列。若不是这样，则在更新期间该数据可能变得不一致。

本节简要回顾第一、第二、第三和第四范式的规则。表的第五范式在有关数据库设计的许多书籍中都有说明，在此就不再赘述。

格式　描述

第一种　表中的每一行位置和每一列位置均有一个值，永远不会是一组值。

第二种　不是键的一部分的每一列都依赖于该键。

第三种　每个非键列都独立于其它非键列，并依赖于该键。

第四种　没有一行包含有关实体的两个或更多个独立多值事实。

第一范式

若每个单元中都只有一个值，永远不会有一组值，则该表使用的是*第一范式*。使用第一范式的表不必满足更高的范式的标准。

例如，下表违反了第一范式，因为对于 PART 的每个具体值，WAREHOUSE 列都包含了几个值。

表 9. 违反第一范式的表

PART（主键）	WAREHOUSE
P0010	Warehouse A、Warehouse B 和 Warehouse C
P0020	Warehouse B 和 Warehouse D

以下示例显示使用第一范式的同一个表。

表 10. 符合第一范式的表

PART（主键）	WAREHOUSE（主键）	QUANTITY
P0010	Warehouse A	400®
P0010	Warehouse B	543
P0010	Warehouse C	329
P0020	Warehouse B	200
P0020	Warehouse D	278

第二范式

若每个不是键一部分的列都依赖于整个键，则该表使用的是第二范式。

当一个非键列依赖于组合键的一部分时，违反了第二范式，如以下示例所示:

表 11. 违反第二范式的表

PART（主键）	WAREHOUSE（主键）	QUANTITY	WAREHOUSE_ADDRESS
P0010	Warehouse A	400	1608 New Field Road
P0010	Warehouse B	543	4141 Greenway Drive
P0010	Warehouse C	329	171 Pine Lane
P0020	Warehouse B	200	4141 Greenway Drive
P0020	Warehouse D	278	800 Massey Street

主键是一个组合键，它由 PART 和 WAREHOUSE 列组成。因为 WAREHOUSE_ADDRESS 列只依赖于 WAREHOUSE 的值，因此该表违反了第二范式的规则。

此设计存在下列问题:

- 对于该仓库中存储的一个部件，在每个记录中重复了该仓库地址。

- 若一个仓库的地址变更，则必须更新引用存储在该仓库中的一个部件的每一行。
- 由于存在这种冗余，该数据可能变得不一致，表现为不同的记录对相同的仓库显示不同的地址。
- 若某个时候一个仓库中没有存储部件，就可能没有哪一行记录了该仓库地址。

解决方案是将该表分割成下面两个表:

表 12. 符合第二范式的 PART_STOCK 表

PART（主键）	WAREHOUSE（主键）	QUANTITY
P0010	Warehouse A	400
P0010	Warehouse B	543
P0010	Warehouse C	329
P0020	Warehouse B	200
P0020	Warehouse D	278

表 13. 符合第二范式的 WAREHOUSE 表

WAREHOUSE（主键）	WAREHOUSE_ADDRESS
Warehouse A	1608 New Field Road
Warehouse B	4141 Greenway Drive
Warehouse C	171 Pine Lane
Warehouse D	800 Massey Street

若有两个表使用第二范式，就需要考虑性能。根据部件位置生成报告的应用程序必须将这两个表连接起来以检索相关信息。

第三范式

若每个非键列都独立于其它非键列，且只依赖于键，则该表使用的是第三范式。

以下示例中的第一个表包含 EMPNO 列和 WORKDEPT 列。假定添加了 DEPTNAME 列（参见第 57 页的表 15）。这个新列依赖于 WORKDEPT，但主键是 EMPNO。该表现在违反了第三范式。更改雇员 John Parker 的 DEPTNAME 不会更改该部门中其它雇员的部门名。注意，现在部门号 E11 中有两个不同的部门名。产生的不一致显示在该表的更新版本中。

表 14. 更新前的不规范 EMPLOYEE_DEPARTMENT 表

EMPNO（主键）	FIRSTNAME	LASTNAME	WORKDEPT	DEPTNAME
000290	John	Parker	E11	Operations
000320	Ramlal	Mehta	E21	Software Support
000310	Maude	Setright	E11	Operations

表 15. 更新后的不规范 EMPLOYEE_DEPARTMENT 表.（表中的信息变得不一致）

EMPNO（主键）	FIRSTNAME	LASTNAME	WORKDEPT	DEPTNAME
000290	John	Parker	E11	Installation Mgmt
000320	Ramlal	Mehta	E21	Software Support
000310	Maude	Setright	E11	Operations

可创建一个含有 WORKDEPT 列和 DEPTNAME 列的新表，将该表规范化。诸如更改部门名之类的更新现在更加简单；只需要更新这个新表。

返回部门名和雇员名的 SQL 查询编写起来更复杂，因为它需要连接两个表。它运行的时间可能比对单个表查询的时间长。因为 WORKDEPT 列必须出现在两个表中，所以需要附加的存储空间。

下面的表被定义为规范化的结果。

表 16. 将 EMPLOYEE_DEPARTMENT 表规范化之后的 EMPLOYEE 表

EMPNO（主键）	FIRSTNAME	LASTNAME	WORKDEPT
000290	John	Parker	E11
000320	Ramlal	Mehta	E21
000310	Maude	Setright	E11

表 17. 将 EMPLOYEE_DEPARTMENT 表规范化之后的 DEPARTMENT 表

DEPTNO（主键）	DEPTNAME
E11	Operations
E21	Software Support

第四范式

若没有一行包含有关一个实体的两个或更多个独立的多值事实，则该表使用的是第四范式。

考虑下列实体：雇员、技能和语言。一个雇员可有几种技能，并通晓几种语言。有两个关系，一个关系是雇员和技能之间的关系，另一个关系是雇员和语言之间的关系。若一个表表示了这两种关系，则它未使用第四范式，如以下示例所示:

表 18. 违反第四范式的表

EMPNO（主键）	SKILL （主键）	LANGUAGE （主键）
000130	Data Modelling	英语
000130	Database Design	英语
000130	Application Design	英语
000130	Data Modelling	西班牙语
000130	Database Design	西班牙语
000130	Application Design	西班牙语

应改用两个表表示这种关系:

表 19. 符合第四范式的 EMPLOYEE_SKILL 表

EMPNO（主键）	SKILL （主键）
000130	Data Modelling
000130	Database Design
000130	Application Design

表 20. 符合第四范式的 EMPLOYEE_LANGUAGE 表

EMPNO（主键）	LANGUAGE （主键）
000130	英语
000130	西班牙语

但是，若这些属性是独立的（即，雇员只将特定的语言应用于特定的技能），则不应将该表分割。

设计数据库的一个良好策略是将所有数据安排在使用第四范式的表中，然后决定该结果是否提供了一个可接受的性能级别。如果没有，则可将数据重新安排在使用第三范式的表中，然后重新评价性能。

多维群集

多维群集（MDC）提供根据多个维对数据进行灵活的、连续的和自动的群集的极佳方法。这会使查询的性能显著提高，以及数据维护操作开销的显著减少，如重组和在插入、更新和删除操作期间的索引维护操作。多维群集主要用于数据入库和大型数据库环境，还可在联机事务处理（OLTP）环境中使用它。

多维群集允许同时根据多个键或维来对表进行物理群集。在版本 8 之前，DB2® 仅支持通过聚类索引对数据进行单维群集。通过使用聚类索引，在表中插入和更新记录时，DB2 会尝试以索引的键次序维护页上的数据的物理次序。聚类索引极大地提高具有包含聚类索引的键的谓词的范围查询的性能，因为，使用好的群集，只需要存取表的一部分，并且，当页是顺序的时候，可以执行更有效的预取。

MDC 将这些优点扩展至多维或群集键。对于查询性能，涉及表的指定维的任何一个或任何组合的范围查询将受益于群集。不仅这些查询只存取具有正确维值的记录，这些限定页还将根据范围分组。而且，尽管具有聚类索引的表经过一段时间由于表中的空间占满而变得不是群集的，但 MDC 表还是能够对所有维自动并持续地维护它的群集，这就避免了重组表来恢复数据的物理次序的必要。

创建表时，可以将一个或多个键指定为作为群集数据依据的维。每一个维都可以包括一列或多列，就象索引键一样。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存取数据。还将自动创建*组合块索引*，它包含所有维键列，并且将用来在插入和更新活动期间维护数据的群集。仅当单个维尚未包含所有的维键列时，才创建组合块索引。优化器还可使用组合块索引来有效地存取具有特定维值的数据。

在 MDC 表中，维值的每个唯一组合形成逻辑*单元*，它物理上由页块组成，其中一块是磁盘上的一组连续页。包含带有具有维块索引之一的特定键值的数据的页的块的集合称为*数据段*。表的每页是一块的部分，并且表的所有块由相同数目的页（*分块因子*）组成。分块因子相当于数据块大小，所以块边界与数据块边界对齐。

考虑记录本国零售商的销售数据的 MDC 表。该表根据维 YearAndMonth 和 Region 进行群集。表中的记录是以块为单位存储的，块包含相当于数据块大小的磁盘上的连续页。在第 60 页的图 22 中，块由矩形表示，并且根据表中分配的数据块的逻辑次序进行编号。图表中的网格表示这些块的逻辑分区，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数据段。例如，区域列中包含值“South-central”的所有记录都可在网格中的“South-central”列定义的数据段中包含的块中找到。实际上，此数据段中的每个块只包含在区域字段中具有

“South-central” 的记录。这样，当且仅当块包含在区域字段具有 “South-central” 的记录时，该块才包含在此网格的数据段或列中。

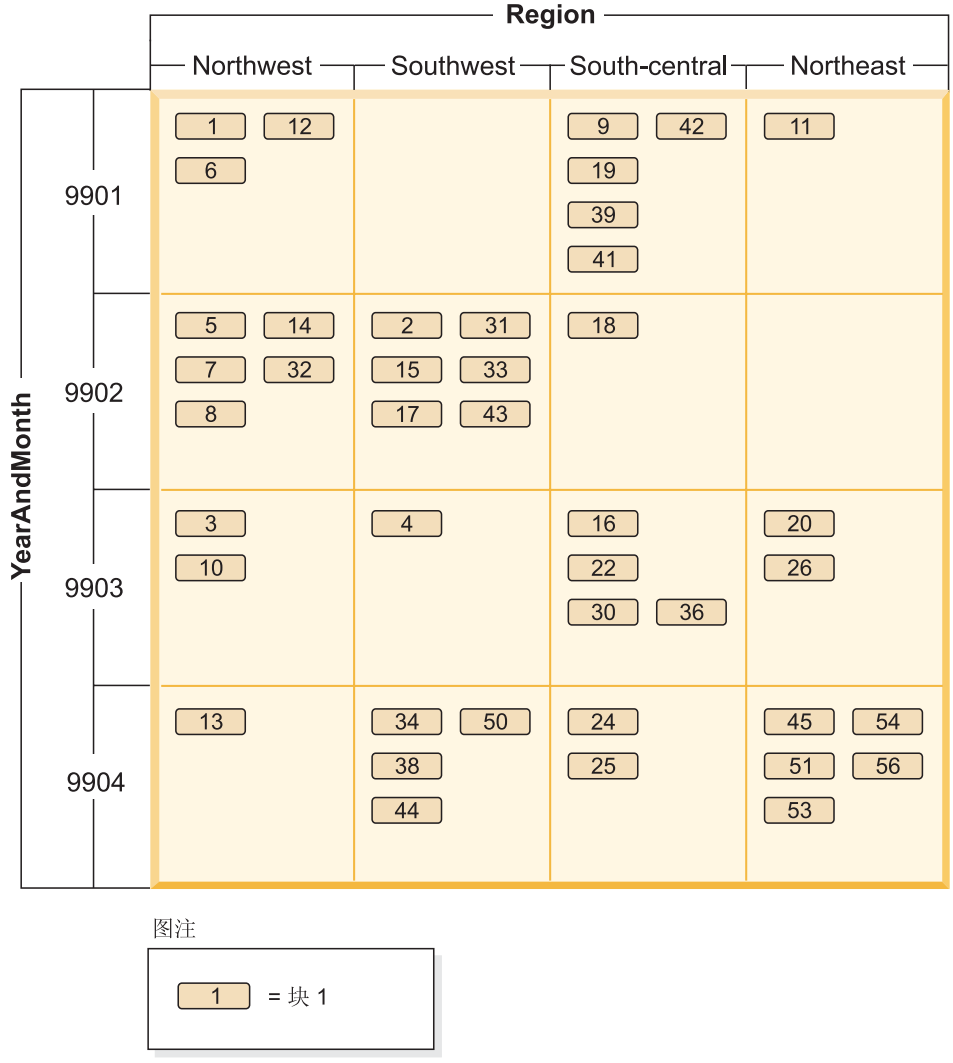

图 22. 带有维 Region 和 YearAndMonth 的多维表

为便于确定哪些块组成数据段，或哪些块包含具有特定维键值的所有记录，在创建表时自动为每个维创建维块索引。

在第 61 页的图 23 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，但在叶子级别，这些键指向块标识符（BID）而不是记录标识符

（RID）。因为每个块潜在地包含许多记录页，所以这些块索引比 RID 索引小得多，并且仅当需要新块并将新块添加至单元或清空现有块并且从单元中除去该块时，才需要更新块索引。

数据段，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 列表在相关维块索引中表示。

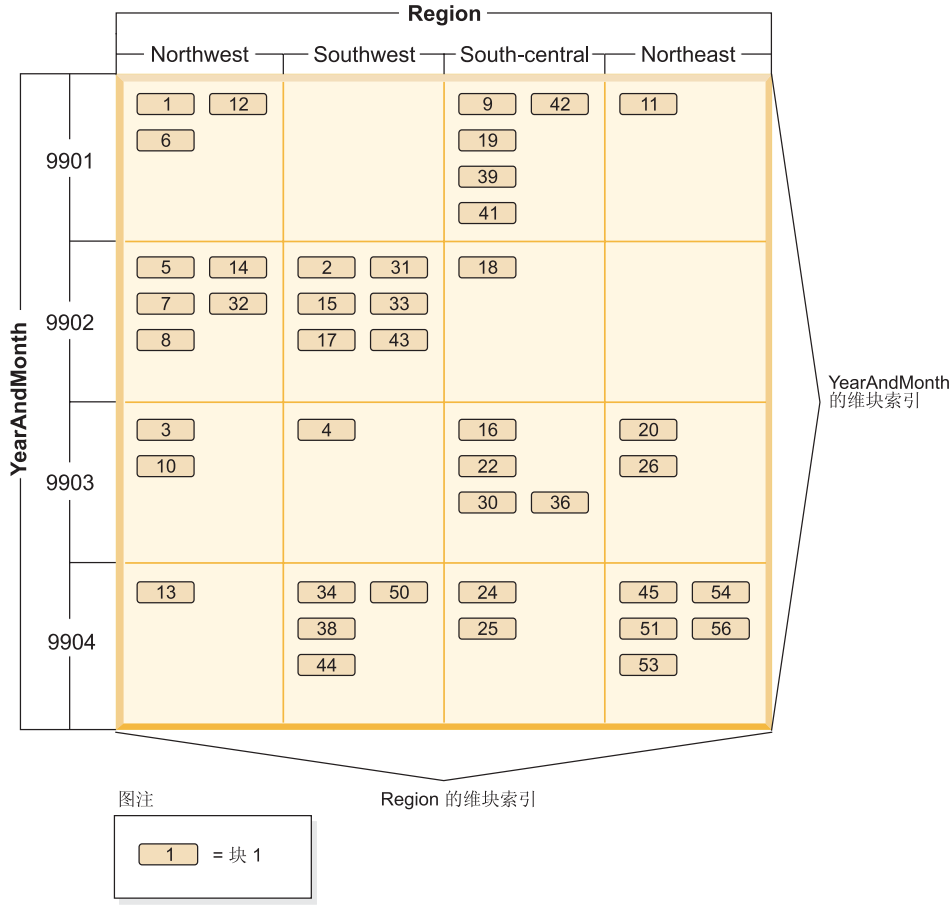

图 23. 显示维块索引且带有维 Region 和 YearAndMonth 的多维表

第 62 页的图 24 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。在第 62 页的图 24 中，列示的块编号与在 Sales 表的网格中发现的 “South-central” 数据段相同（参见 第 60 页的图 22）。

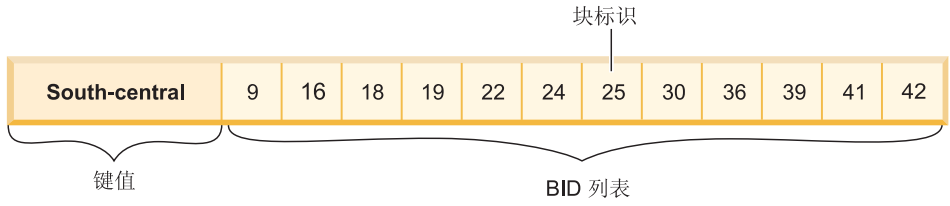

图 24. Region 的维块索引中的键

相似的，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，如图 25 中所示。

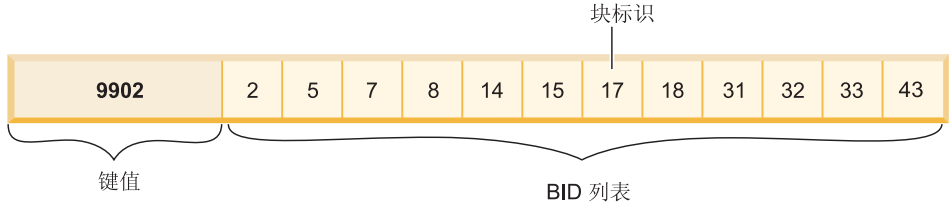

图 25. YearAndMonth 的维块索引中的键

将记录插入到 Sales 表中时，DB2 将确定针对其维值，是否存在单元。如果存在，则 DB2 会将记录插入到该单元的现有块中（如果可以的话），或将另一个块添加至该单元（如果当前块已满）。如果该单元尚未存在，则 DB2 将创建新的单元并向其添加块。此自动维护是使用附加块索引实现的，是在创建 MDC 表时创建的。此块索引基于表的所有维列，以便每个键值与该表中的特定单元相对应，并且它的 BID 列表与组成该单元的块的列表相对应（参见第 63 页的图 26）。此类型的索引是*组合块索引*。

对于包含记录的表的每个单元，此组合块索引中只有一个键。此块索引有助于快速有效地查找带有针对其维具有一组特定值的记录的那些块。组合块索引用于查询处理，以存取在表中具有特定维值的数据。它也用来在插入活动期间根据表维动态管理和维护数据的物理群集。

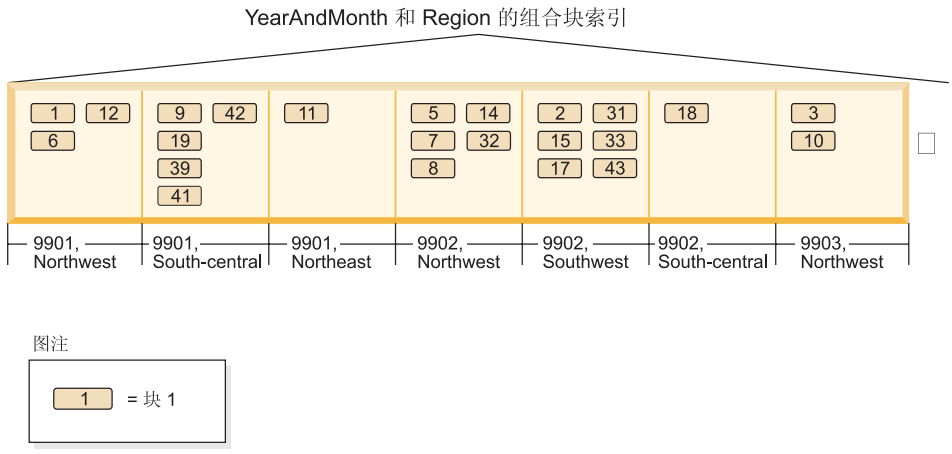

图 26. YearAndMonth 和 Region 的组合块索引

例如，如果想要查找 Sales 表中 Region 为 “Northwest” 且 YearAndMonth 为 “9903” 的所有记录，则 DB2 将在组合块索引中查找键值 9903,Northwest，如图 27 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 27. YearAndMonth 和 Region 的组合块索引中的键

要说明插入期间此索引的使用情况，以插入具有维值 9903 和 Northwest 的另一记录为例。DB2 将在组合块索引中查找此键值并找到块 3 和 10 的 BID。这些块包含具有这些维键值的仅有全部记录。如果这些块的页上还有空间，DB2 会将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，则 DB2 为表分配新块或使用表中先前清空的块。注意，在此示例中，表当前没有使用块 48。DB2 将记录插入到该块中的某页上，并通过将其 BID 添加至组合块索引及每个维块索引来将此块指定给此单元。参见第 64 页的图 28 以获取有关添加块 48 之后的维块索引的键的说明。

9903	3	4	10	16	20	22	26	30	36	48

Northwest	1	3	5	6	7	8	10	12	14	32	48

9903, Northwest	3	10	48

图 28. 添加块 48 之后维块索引的键

如果 Sales 表根据三个维进行群集，则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。如果表具有维 YearAndMonth、Region 和 Product，则可以认为它是逻辑多维数据，如第 65 页的图 29 中所示。

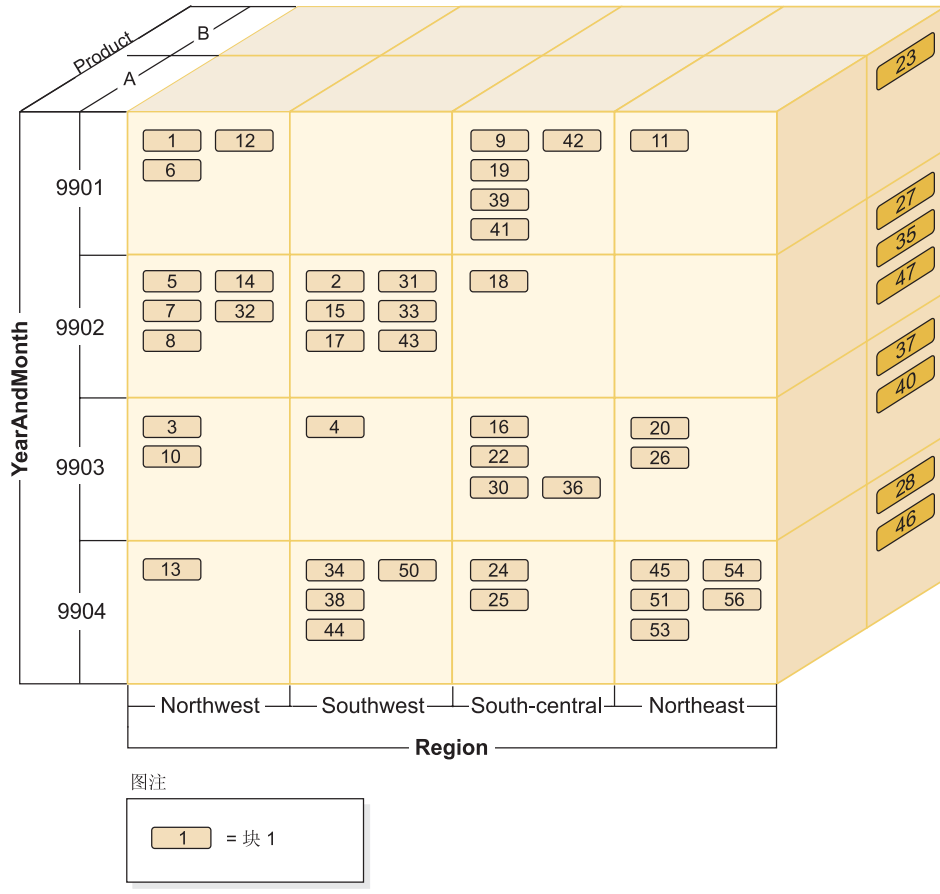

图 29. 具有维 Region、YearAndMonth 和 Product 的多维表

将对带有三个维: YearAndMonth、Region 和 Product 的 MDC 表创建四个块索引, 每个维一个; 并将带有所有维列的另一个块索引作为键。如果想要查找具有 Product=Product A 且 Region=Northeast 的所有记录, DB2 将通过在 Product 维块索引中查找 Product A 键来先确定包含具有 Product=Product A 的所有块的数据段。(参见图 30。)然后, DB2 通过在 Region 维块索引中查找 Northeast 键来确定包含具有 Region=Northeast 的所有记录的块。(参见第 66 页的图 31。)

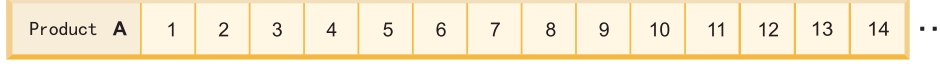

图 30. Product 的维块索引中的键

图 31. Region 的维块索引中的键

要查找包含具有这两个值的所有记录的一组块，必须找到这两个数据段的交集。可通过对两个 BID 列表进行索引 AND 操作来完成。常用的 BID 值有 11、20、26、45、54、51、53 和 56。

一旦具有要扫描的块列表，DB2 可以对每个块进行最小关系扫描。对于每个块，此操作仅涉及一个 I/O，因为每个块在磁盘上作为数据块存储并且可以作为一个单元读取到缓冲池中。DB2 只需要对该块中的某个记录重新应用这些谓词，因为块中的所有记录都保证具有相同的维键值。如果有其它的谓词，DB2 只需要对块中的余下记录检查这些谓词。

对于 MDC 表，还支持基于 RID 的常规索引，并且通过索引 AND 和 OR 操作将 RID 和块索引组合起来。索引顾问将帮助针对 MDC 表选择基于 RID 的索引，但是它对为 MDC 表选择维没有任何帮助。

或者将 MDC 表视作任何现有表；即，可根据这些表定义触发器、引用完整性、视图和具体查询表。

MDC 和数据库分区:

多维群集可以与数据库分区联合使用。实际上，MDC 可以作为数据库分区的补充。例如，MDC 表 Sales 是根据 YearAndMonth、Region 和 Product 群集的。还可以根据 Region 字段（或任何其它字段）在数据库分区上对该表进行分区。在根据 Region 进行分区的情况下，特定分区的所有块将在相同的数据库分区上，但是该分区可能包含多个区域值。每个分区都将包含逻辑多维数据的数据段与分区的 Region 分区值相对应的 MDC 表。例如，South-central 和 Northwest 数据段可能在一个分区上，而 Southwest 和 Northeast 在另一分区上。每个分区还将包含具有该特定分区上包含的块的键值的每一个块索引。

MDC 表的装入注意事项:

如果定期将数据装入到数据仓库中，则使用 MDC 表会很有帮助。在 MDC 表中，装入会先重新使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，一个月的旧数据），则可以使用装入实用程序来装入下一个月的数据，并且它可以重新使用（已落实）删除之后清空的块。

MDC 表的记录注意事项:

如果 RID 索引先前索引或另外索引的列现在是维并且因此由块索引来索引，则索引维护和记录将显著地减少。仅当删除了整个块中的最后一个记录，DB2 才需要从块索引中除去 BID 并将此索引操作计入日志。相似的，仅当将记录插入到新块中（如果它是单元的第一个记录或是对当前块已满的单元的插入）时，DB2 才需要将 BID 插入到块索引中并将该操作计入日志。因为块可以有 2 到 256 页的记录，所以此块索引维护和记录相对较小。仍将对表和 RID 索引的插入和删除计入日志。

MDC 表的块索引注意事项:

当为 MDC 表定义维时，会创建维块索引。另外，会为定义的所有维创建组合块索引。如果对 MDC 表仅定义了一个维，DB2 将仅创建一个块索引，它将充当维块索引和组合块索引。相似的，如果创建的 MDC 表具有针对列 A 和针对（列 A 和列 B）的维，DB2 将对列 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 上的维块索引也将充当组合块索引。

组合块索引也用于查询处理，以存取在表中具有特定维值的数据。注意组合块索引中的键部分的次序可能影响它对于查询处理的使用或适用性。它的键部分的次序由创建 MDC 表时使用的整个 ORGANIZE BY DIMENSIONS 子句中的列的次序确定。例如，如果使用以下语句创建表：`CREATE TABLE t1 (c1 int, c2 int, c3 int, c4 int) ORGANIZE BY DIMENSIONS (c1, c4, (c3,c1), c2)`，则将对列 (c1,c4,c3,c2) 创建组合块索引。注意，尽管在 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是使用了一次，并且是以第一次发现它的次序使用的。组块索引中的键部分的次序对于插入处理没有影响，但是对于查询处理有影响。因此，如果列次序 (c1,c2,c3,c4) 具有组合块索引更加合适，所以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)`。

相关概念:

- 『Indexes』（*SQL Reference, Volume 1*）
- 第 68 页的『选择 MDC 表维时的注意事项』
- 『MDC 表的表和索引管理』（《*管理指南: 性能*》）
- 『MDC 表的优化策略』（《*管理指南: 性能*》）
- 第 70 页的『创建 MDC 表时的注意事项』

相关参考:

- 『MDC 表的表和 RID 索引扫描的锁定方式』（《*管理指南: 性能*》）
- 『MDC 表的块索引扫描的锁定』（《*管理指南: 性能*》）

选择 MDC 表维时的注意事项

将受益于 MDC 的查询:

选择表维时的第一个注意事项是确定哪些查询将受益于块级别的群集。您将要考虑为涉及等同及范围查询的列创建维键，尤其是那些具有大量基数的列。您还要考虑为涉及与维度表进行星形连接的 MDC 事实表中的外键创建维。记住，对多个维进行自动持续的群集以及数据块或块级别的群集会提高性能。

可以利用多维群集的查询包括以下形式的查询。对于所有这些示例，假定存在一个 MDC 表 t1，具有维 c1、c2 和 c3。

示例 1:

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

此查询涉及单个维的范围谓词，所以可以在内部重写以使用 c3 的维块索引来存取该表。将扫描该索引以查找键值小于 5000 的块标识符（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 的维块索引来存取该表。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 t1 WHERE c2 > 100 AND c1 = '16/03/1999' AND c3 > 1000 and c3
< 5000
```

此查询涉及 c2 和 c3 的范围谓词和 c1 的同等谓词以及 AND 操作。可以内部重写该查询以根据每个维块索引来存取该表。将对 c2 块索引进行扫描以查找键值大于 100 的 BID; 将对 c3 块索引进行扫描以查找键值在 1000 和5000 之间的 BID; 将对 c1 块索引进行扫描以查找键值为 “16/03/1999” 的 BID。将对每个块扫描的结果 BID 进行索引 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 4:

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

此查询涉及 c1 维的范围谓词和 c2 维的 IN 谓词以及 OR 操作。可以内部重写该查询以根据维块索引 c1 和 c2 来存取该表。将对 c1 维块索引进行扫描以查找

小于 5000 的值并对 c2 维块索引进行另一扫描以查找值 1、2 和 3。将对每个块索引扫描的结果 BID 进行索引 OR 操作，然后对块的结果集应用最小关系扫描以查询实际记录。

示例 5:

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

此查询涉及 c1 维的等同谓词和不是维的列的另一范围谓词以及 AND 操作。可以内部重写该查询以存取 c1 的维块索引，以获取 c1 值为 15 的表的数据段的块列表。如果 c4 具有 RID 索引，则可以进行索引扫描以检索 c4 小于 12 的记录的 RID，然后可以对块的结果列表和此记录列表进行 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，则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 6:

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

此查询涉及维 c1 的范围谓词和非维列 c4 的等同谓词以及 OR 操作。如果 c4 列具有 RID 索引，则可以内部重写该查询以对 c1 的维块索引和 c4 的 RID 索引进行索引 OR 操作。如果 c4 没有索引，则可能选择表扫描作为替代，因为必须检查所有的记录。索引 OR 操作将涉及对 c1 的块索引扫描以查找小于 4 的值，以及对 c4 的 RID 索引扫描以查找值 100。当每个合格的块中的所有记录都合格时，将对这些块执行最小关系扫描，并且检索这些块之外的记录的任何附加 RID。

示例 7:

```
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'
```

此查询涉及星形连接。在此示例中，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）以及维度表。维度表不一定是 MDC 表。Region、year 和 product 是可以使用常规或块索引建立索引的各自维度表的列（如果维度表是 MDC 表的话）。当根据 c1、c2 和 c3 值存取事实表时，可以根据这些列对维块索引进行块索引扫描，然后对结果 BID 的执行索引 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:

设计多维表时的附加注意事项是表中的预期单元密度（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儿积。例如，如果根据维 Day、Region 和 Product 群集表且涵盖五年的数据，则表中可能具有 1821 days * 12 regions * 5 products = 109,260 个不同的单元。任何仅包含几个记录的任何单元仍需要分配给它的整块的页，以存储该单元的记录。如果块大小较大，则此表结果可能会比它实际需要的大得多。

要确保 MDC 表不大于它需要的大小，有几个选项。第一个选项是通过相应减小数据块的大小以减小每个块的大小。在此情况下，仅包含具有几个记录的一个块的每个单元包含的块较小，且浪费的空间也会少一些。但是，存在一个折衷，即对于那些具有许多记录的单元，将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符（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填充操作，因为会更快地清空和填充块。与数目较少的群集数据的较大组合相比较，它还会导致表中的群集数据的更多更小的组合，以装入更多填充单元值。

要确保 MDC 表具有适当的大小，则要考虑的第二个因素是考虑将一个或多个维上滚为较粗的粒度，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群集，但将维 Day 替换为维 YearAndMonth。这将给予 YearAndMonth、Region 和 Product 基数 60、12 和 5，可能的单元数目是 3600。每个单元将具有较大范围的值，并降低了只包含几个记录的可能性。还应考虑通常对涉及到的列使用的谓词，如使用的较多的是日期的月份、季度还是天。这将影响更改维的粒度的期望度。例如，如果大多数谓词是针对特定天，并且您根据月份群集表，则 DB2® 将能够使用 YearAndMonth 的块索引来快速缩小包含想要的天数的月份的范围并仅存取这些相关联的块。但是在扫描块时，必须应用天数谓词以确定哪些天合格。但是，如果根据天群集（并且天具有很高的基数），则可以使用天的块索引来确定扫描哪些块，并且仅需对合格的每个单元的第一个记录重新应用天谓词。在此情况下，在使用用户定义函数上滚 Region 列（它包含 12 个不同的值）至 Regions West、North、South 和 East 时最好考虑上滚其它维之一以增加单元的密度。

相关概念:

- 第 59 页的『多维群集』

创建 MDC 表时的注意事项

将数据从现有表移至 MDC 表:

要将数据从现有表移至 MDC 表，导出数据，删除原始表，创建 MDC 表（使用带有 ORGANIZE BY DIMENSIONS 子句的 CREATE TABLE 语句）并以您自己的数据装入 MDC 表。

SMS 表空间中的 MDC 表:

在 SMS 表空间中，缺省情况下文件每次扩展一页。对于多维群集，数据以块为单位群集并且分块因子与表空间的数据块大小相等。当扩展 MDC 表时，一次扩展一个块或一个数据块的页数。因此，如果将多维表存储在 SMS 表空间中，则必须启用多页文件分配。当启用多页文件分配时，SMS 文件作为一个整体精确地按照需要的大小扩展，因此显著地提高了性能。

运行 *db2empfa* 实用程序以启用多页文件分配。一旦启用了多页文件分配，就不能禁用它。

带有三个维的 MDC 表:

如果知道将在查询中大量使用特定谓词，则可以使用 ORGANIZE BY DIMENSIONS 子句根据涉及的列对表进行群集。

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多维数据（即，具有三个维）的三个本地列中的值群集。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段，以便涉及的关系运算符仅处理相应的数据段或单元中的块。注意，块的大小，即，块中的页数将是表的数据块大小。

带有多个列的维的 MDC 表:

每个维可以由一个或多个列组成。这样，可以创建根据包含两列的维群集的表。

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，表将根据两个维 c1 和 (c3, c4) 群集。这样，在查询处理期间，表可以根据 c1 维或组合 (c3, c4) 维逻辑分段。该表将与示例 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，将有三个维块索引，列 c1、c3 和 c4 各一个。在示例 2 中，将有两个维块索引，一个针对列 c1，而另一个针对 c3 和 c4。这两个方法的主要不同之处在于，在示例 1 中，仅涉及 c4 的查询可以使用 c4

的维块索引来快速直接地存取相关数据块。在示例 2 中，c4 是维块索引中的辅助键部分，因此仅涉及 c4 的查询涉及更多的处理。但是，在示例 2 中，DB2® 将少维护和存储一个块索引。

列表达式作为维的 MDC 表:

列表达式也可用于群集维。根据列表达式群集的功能对于将维上滚至更粗粒度非常有用，如将地址上滚为地理位置或区域，或将日期上滚为星期、月份或年。要以此方式实现维的上滚，可以使用生成的列。此类型的列定义将允许使用可表示维的表达式创建列。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群集的表。

示例 3:

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

在示例 3 中，列 c5 是基于列 c3 和 c4 的表达式，而列 c6 会将列 c1 上滚至更粗粒度。此语句将根据列 c2、c5 和 c6 中的值群集该表。

单调性:

列表达式可以包括任何有效的表达式，包括用户定义函数（UDF）。但是，要对维应用范围查询，即，要让范围扫描对基本块索引成为可能，表达式在性质上必须是单调的。非单调表达式将只允许对该维应用等同谓词。非单调函数的一个较好的示例是 MONTH( )，如示例 3 中的列 c6 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，则函数将返回范围为 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('99/01/05')) = 1
MONTH(date('99/02/08')) = 2
MONTH(date('99/03/24')) = 3
MONTH(date('99/04/30')) = 4
...
MONTH(date('99/12/09')) = 12
MONTH(date('00/01/18')) = 1
MONTH(date('00/02/24')) = 2
...
```

因此，尽管此示例中的日期连续增加，但 MONTH(date) 不会增加。更具体地说，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`。这可以以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，必须将年包括为月份的较高次序部分。DB2 提供对 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，可以分开查找年和月份的整数表示。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，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，类似于 INTEGER(date)。

只要可能，DB2 会在创建表的生成列或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，如 DATENUM( )、DAYS() 和YEAR( )。并且，列和常量的各种算术表达式，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，或如果它不能确定这一点，该维将仅支持对其基本列使用等同谓词。

相关概念:

- 『Export Overview』（*Data Movement Utilities Guide and Reference*）
- 第 59 页的『多维群集』
- 第 68 页的『选择 MDC 表维时的注意事项』

相关任务:

- 『对表定义维』（*《管理指南: 实现》*）

相关参考:

- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）
- 『db2empfa - Enable Multipage File Allocation Command』（*Command Reference*）

约束

*约束*是数据库管理器强制使用的规则。

有三种类型的约束:

- *唯一约束*是这样一个规则，它禁止一个表的一列或多列中出现重复值。唯一键和主键是受支持的唯一约束。例如，可对供应商表中的供应商标识符定义唯一约束以确保不会对两个供应商指定同一供应商标识符。
- *引用约束*是关于一个或多个表中的一列或多列中的值的一种逻辑规则。例如，一组表共享关于公司的供应商的信息。供应商的名称有时可能会更改。可定义一个引用约束，声明表中的供应商的标识必须与供应商信息中的供应商标识相匹配。此约束会阻止可能导致丢失供应商信息的插入、更新或删除操作。
- *表检查约束*对添加至特定表的数据设置限制。例如，表检查约束可确保每当在包含个人信息的表中添加或更新薪水数据时，雇员的薪水级别至少为 $20,000。

可打开或关闭引用约束和表检查约束。例如，通常会在将大量数据装入数据库时关闭约束。

唯一约束

*唯一约束*是这样一种规则，仅当某个键的值在表中是唯一的时候，这些值才有效。唯一约束是可选的，可在 CREATE TABLE 或 ALTER TABLE 语句中使用 PRIMARY KEY 子句或 UNIQUE 子句来定义。在唯一约束中指定的列必须定义为 NOT NULL。数据库管理器使用唯一索引在对唯一约束的各列进行更改时强制键的唯一性。

表可以有任意数目的唯一约束，最多将一个唯一约束定义为主键。对于同一组列，表不能有多个唯一约束。

引用约束的外键引用的唯一约束称为*父键*。

当在 CREATE TABLE 语句中定义唯一约束时，唯一索引是由数据库管理器自动创建的，且被指定为主索引或系统所需的唯一索引。

当在 ALTER TABLE 语句中定义唯一约束且同一组列存在索引时，该索引被指定为唯一的且是系统所需的。如果这样的索引不存在，数据库管理器会自动创建唯一索引，并将其指定为主索引或系统所需的唯一索引。

注意，定义唯一索引与创建唯一索引是有区别的。尽管都强制唯一性，但唯一索引允许可空列，且通常不能用作父键。

引用约束

*引用完整性*是数据库的一种状态，在该状态中，所有外键的所有值都有效。*外键*是表中的一列或一组列，它的值需要与其父表的行的至少一个主键或唯一键值相匹配。*引用约束*是这样一种规则，仅当满足下列其中一个条件时，外键的值才有效:

- 它们作为父键的值出现。
- 外键的某些组成部分为 null。

包含父键的表称为引用约束的*父表*，包含外键的表被认为是该表的*从属表*。

引用约束是可选的，可在 CREATE TABLE 语句或 ALTER TABLE 语句中定义它。引用约束在执行 INSERT、UPDATE、DELETE、ALTER TABLE、ADD CONSTRAINT 和 SET INTEGRITY 语句时由数据库管理器强制使用。

在强制使用所有其它引用约束之前，会先强制使用带有删除或更新规则 RESTRICT 的引用约束。在大部分情况下，带有删除或更新规则 NO ACTION 的引用约束的行为类似 RESTRICT。

注意，引用约束、检查约束和触发器可以组合使用。

引用完整性规则涉及下列概念和术语:

父键 引用约束的主键或唯一键。

父行 具有至少一个从属行的行。

父表 包含引用约束的父键的表。表可在任意数目的引用约束中充当父表。在引用约束中充当父表的表还可以是引用约束中的从属表。

从属表 在其定义中包含至少一个引用约束的表。表可在任意数目的引用约束中充当从属表。在引用约束中充当从属表的表还可以是引用约束中的父表。

派生表 作为表 T 的后代的表（如果它是 T 的从属表或是 T 的从属表的后代）。

从属行 具有至少一个父行的行。

后代行 作为行 r 的后代的行（如果它是 r 的从属行或是 r 的从属行的后代）。

引用循环
它是这样的一组引用约束，该组中的每个表都是它本身的后代。

自引用表
在同一引用约束中既充当父表又充当从属表的表。该约束称为*自引用约束*。

自引用行
充当本身的父行的行。

插入规则
引用约束的插入规则为: 外键的非 null 插入值必须与父表的父键的某些值相匹配。如果组合外键的值的任何组成部分为 null，则该值为 null。指定外键时，此规则是隐式的。

更新规则

引用约束的更新规则是在定义引用约束时指定的。选项有 NO ACTION 和 RESTRICT。在更新父表的某行或从属表的某行时应用更新规则。

如果是父行，更新父键的某列中的值时，下列规则适用:

- 如果从属表中的任何行与该键的原始值相匹配，在更新规则为 RESTRICT 的情况下，会拒绝更新。
- 如果在更新语句完成时从属表中的任意行没有相应的父键（排除 AFTER 触发器），当更新规则为 NO ACTION 时，会拒绝更新。

如果是从属行，当指定外键时，NO ACTION 更新规则是隐式的。NO ACTION 意味着更新语句完成时，外键的非 null 更新值必须与父表的父键的某些值相匹配。

如果组合外键的值的任何组成部分为 null，则该值为 null。

删除规则

引用约束的删除规则是在定义引用约束时指定的。选项有 NO ACTION、RESTRICT、CASCADE 或 SET NULL。仅当外键的某些列允许 null 值时，才能指定 SET NULL。

在删除父表的某行时应用引用约束的删除规则。更精确地说，当父表的某行是删除或传播删除操作（将在下面定义）的对象，且该行在引用约束的从属表中具有从属项时，该规则适用。考虑这样一个示例，其中 P 是父表，D 是从属表，而 p 是充当删除或传播删除操作的对象的父行。删除规则的工作方式如下:

- 对于 RESTRICT 或 NO ACTION，发生错误，且不会删除任何行。
- 对于 CASCADE，删除操作会传播至表 D 中的 p 的从属项。
- 对于 SET NULL，表 D 中的 p 的每个从属项的外键的每个可空列被设置为 null。

表在其中充当父表的每个引用约束有它自己的删除规则，所有适用的删除规则被用来确定删除操作的结果。因此，在带有删除规则 RESTRICT 或 NO ACTION 的引用约束中，如果行具有从属项，则不能删除该行，或在带有删除规则 RESTRICT 或 NO ACTION 的引用约束中，删除会级联至充当从属项的任何一个后代。

从父表 P 删除一行涉及其它表，可能会影响这些表的各行:

- 如果表 D 是 P 的从属项，且删除规则为 RESTRICT 或 NO ACTION，则该操作会涉及 D，但 D 不会受该操作影响。
- 如果 D 是 P 的从属项，且删除规则为 SET NULL，且该操作涉及 D，且在执行该操作期间会更新 D 的各行。

- 如果 D 是 P 的从属项且删除规则为 CASCADE，则该操作涉及 D，且 D 的各行会在执行该操作期间删除。

  如果 D 的各行被删除，且针对 P 的删除操作会被认为传播至 D。如果 D 也是父表，则从而此列表中描述的操作适用于 D 的从属项。

涉及针对 P 的删除操作的任何表都被认为是*删除连接*至 P。因此，如果表是 P 的从属项，或是 P 中的删除操作级联至的表的从属项，则该表删除连接至表 P。

表检查约束

*表检查约束*是这样一种规则，它指定表中每行的一列或多列中允许使用的值。约束是可选的，可使用 CREATE TABLE 或 ALTER TABLE 语句定义它。指定表检查约束是通过限制格式的搜索条件完成的。其中一个限制是表 T 的表检查约束中的列名必须标识表 T 中的某列。

表可以有任意数目的表检查约束。表检查约束是通过对插入或更新的每行应用其搜索条件强制使用的。如果搜索条件的结果对任何行都是 false，则发生错误。

当在 ALTER TABLE 语句中对带有现有数据的表定义一个或多个表检查约束时，会在 ALTER TABLE 语句完成前针对新条件检查现有数据。SET INTEGRITY 语句可用来将表置于*检查暂挂*状态，这允许 ALTER TABLE 语句继续进行而不检查数据。

相关参考:

- 『SET INTEGRITY statement』（*SQL Reference, Volume 2*）
- 『Interaction of triggers and constraints』（*SQL Reference, Volume 1*）

触发器

*触发器*定义这样一组操作，这些操作是在响应对指定表的插入、更新或删除操作时执行的。执行这样的 SQL 操作时，触发器被认为是已*激活的*。

触发器是可选的，且是使用 CREATE TRIGGER 语句定义的。

可将触发器与引用约束和检查约束配合使用，以强制使用数据完整性规则。还可使用触发器来导致对其它表的更新、自动生成或变换插入或更新的行的值或调用函数以执行如发出警告之类的任务。

对于定义或强制*事务性*商业规则，触发器是非常有用的机制，这些规则涉及数据的不同状态（例如，薪水增长不能超过 10%）。

使用触发器会设置逻辑以在数据库内强制使用商业规则。这表示应用程序不负责强制使用这些规则。对所有表强制使用的集中逻辑意味着更容易维护，因为在逻辑更改时，不需要更改应用程序。

下列各项是在创建触发器时指定的:

- *主题表*指定对其定义触发器的表。
- *触发事件*定义修改主题表的特定 SQL 操作。该事件可以是插入、更新或删除操作。
- *触发器激活时间*指定触发器应在触发事件发生之前还是之后激活。

导致触发器激活的语句包括*一组受影响的行*。这些行就是正对其进行插入、更新或删除操作的主题表的行。*触发器间隔尺寸*指定触发器的操作是对该语句执行一次，还是对每个受影响的行执行一次。

*触发操作*包括可选搜索条件和每次激活触发器时执行的一组 SQL 语句。仅当搜索条件求值为 true 时，才执行这些 SQL 语句。如果触发器激活时间是在触发事件之前，触发操作可包括用来选择、设置转换变量或发出 SQLstates 信号。如果触发器激活时间是在触发事件之后，触发操作可包括用来选择、插入、更新、删除或发出 SQLstates 信号。

触发操作可使用*转换变量*来引用一组受影响的行中的值。转换变量使用由指定的名称限定的主题表中的各个列名，以标识是引用旧值（更新前）还是引用新值（更新后）。在之前、插入或更新触发器中，还可使用 SET Variable 语句来更改新值。

引用一组受影响的行中的各个值的另一种方法是使用*转换表*。转换表同样使用主题表中的各个列名，但它指定一个名称，以允许将一整组受影响的行视作一个表。只能在之后触发器中使用转换表，且可对旧值和新值定义独立的转换表。

可对表、事件或激活时间的组合指定多个触发器。激活触发器的次序与创建它们的次序相同。因此，最新创建的触发器就是最后激活的触发器。

触发器的激活可能导致*触发器级联*，这是激活一个触发器的结果，它会执行导致另一触发器激活甚至是同一触发器再次激活的 SQL 语句。触发操作可能导致删除的引用完整性规则的应用程序造成的更新，从而可能导致激活附加触发器。借助触发器级联，可能会激活触发器和引用完整性删除规则链，从而由于单个 INSERT、UPDATE 或 DELETE 语句而导致对数据库的大幅度更改。

附加数据库设计注意事项

当设计一个数据库时，重要的是考虑用户应该能够存取哪些表。通过授权，授予或撤消对表的存取权。最高权限级别是系统管理权限（SYSADM）。具有 SYSADM 权限的用户可指定其它授权，包括数据库管理员权限（DBADM）。

为了进行*审计*，可能必须在指定的时间段内记录每次对数据所做的更新。例如，您可能希望在每次雇员工资改变时更新和审计表。若定义了适当的触发器，则可自动更新此表。还可通过 DB2® 审计设施执行审计活动。

考虑到性能方面的原因，可能只想存取选择的数据量，同时将基本数据作为*历史数据*维护。应在设计中加入对维护此历史数据的要求，如数据可以保留几个月或几年才可以清除。

您可能还想使用*总结*信息。例如，您可能有一个表，它具有所有雇员的信息。但是，您希望将此信息按分部或部门划分为单独的表。在这种情况下，基于原始表中的数据的每个分部或部门的具体查询表将非常有帮助。

在设计中还应标识隐含的*安全性*考虑事项。例如，您可能决定通过安全性表支持用户存取特定类型的数据。可定义各种数据的存取级别以及谁可存取此数据。机密数据，如雇员和工资单数据，将具有严格的安全性限制。

可以创建具有相关的*结构化类型*的表。利用这种类型表，可以使用表之间定义的关系建立一个层次结构，称为*类型层次结构*。该类型层次结构由单个的根类型、超类型和子类型组成。

*引用类型*表示法是在创建类型层次结构的根类型时定义的。引用的目标始终是类型表或视图中的行。

第 5 章 物理数据库设计

完成逻辑数据库设计后，关于数据库和表将驻留其中的物理环境，有几个问题需要考虑。这包括了解为支持和管理数据库而要创建的文件、了解存储数据需要多少空间以及确定如何使用存储数据所需的表空间。

数据库目录和文件

当创建一个数据库时，关于该数据库的信息（包括缺省信息）会保存在目录层次结构中。此分层目录结构的创建位置取决于您在 CREATE DATABASE 命令中提供的信息。如果您在创建数据库时未指定目录路径或驱动器的位置，则使用缺省位置。

我们建议您显式地指出您希望在何处创建数据库。

在您在 CREATE DATABASE 命令中指定的目录中，将使用实例名创建一个子目录。这个子目录可以确保在同一目录下的不同实例中创建的数据库不会使用相同的路径。在实例名子目录下面，将创建一个名为 NODE0000 的子目录。这个子目录可以区分逻辑分区数据库环境中的分区。在节点名目录下面，将创建一个名为 SQL00001 的子目录。这个子目录使用数据库记号进行命名，表示正在创建的数据库。SQL00001 中包含与第一个创建的数据库以及随后创建的具有更高编号的数据库（SQL00002等等）相关联的对象。这些子目录可以区分在 CREATE DATABASE 命令中指定目录下的实例中创建的数据库。

目录结构如下所示:

```
<your_directory>/<your_instance>/NODE0000/SQL00001/
```

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

- 文件 SQLBP.1 和 SQLBP.2 中包含缓冲池信息。这两个文件中具有相同的拷贝，从而提供备份。
- 文件 SQLSPCS.1 和 SQLSPCS.2 中包含表空间信息。这两个文件中具有相同的拷贝，从而提供备份。
- SQLDBCON 文件中包含数据库配置信息。切勿编辑此文件。要更改配置参数，请使用控制中心或命令行语句 UPDATE DATABASE MANAGER CONFIGURATION 和 RESET DATABASE MANAGER CONFIGURATION。
- DB2RHIST.ASC 历史记录文件及其备份 DB2RHIST.BAK 中包含关于备份、恢复、表装入、表重组、表空间更改和其它数据库更改的历史记录信息。

DB2TSCHNG.HIS 文件中包含某个日志文件级别上的表空间更改的历史记录。DB2TSCHG.HIS 中包含的有关每个日志文件的信息，有助于识别受该日志文件影响的表空间。表空间恢复程序在进行表空间恢复时，将使用此文件中的信息来确定要处理的日志文件。可以在文本编辑器中检查这两个历史记录文件中的内容。

- 日志控制文件 SQLOGCTL.LFH 和 SQLOGMIR.LFH 中包含有关活动日志的信息。

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

  注: 您应确保不要将日志子目录映射到用于存储数据的磁盘。这样，在磁盘发生问题时，只会影响到数据或日志，而不会同时影响到二者。由于日志文件与数据库容器不会争用同一磁盘磁头的移动，因此这会带来实实在在的性能好处。要更改日志子目录的位置，请更改 *newlogpath* 数据库配置参数。
- SQLINSLK 文件用于确保一个数据库只能由数据库管理器的一个实例使用。

SMS 数据库目录的附加信息

SQLT* 子目录中包含运作数据库所需的缺省“系统管理空间”（SMS）表空间。系统将创建三个缺省表空间:

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

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

此外，名为 SQL*.DAT 的文件中还包含有关子目录或容器中的每个表的信息。星号（*）将由唯一的数字组取代，用以识别每个表。对于每个 SQL*.DAT 文件，可能有一个或多个下列文件，这取决于表类型、表的重组状态或者表是否存在索引、LOB 或 LONG 字段:

- SQL*.BMP（如果是 MDC 表，则包含块分配信息）
- SQL*.LF（包含 LONG VARCHAR 或 LONG VARGRAPHIC 数据）
- SQL*.LB（包含 BLOB、CLOB 或 DBCLOB 数据）
- SQL*.LBA（包含关于 SQL*.LB 文件的分配和可用空间信息）
- SQL*.INX（包含索引表数据）
- SQL*.DTR（包含用于重组 SQL*.DAT 文件的临时数据）

- SQL*.LFR（包含用于重组 SQL*.LF 文件的临时数据）
- SQL*.RLB（包含用于重组 SQL*.LB 文件的临时数据）
- SQL*.RBA（包含用于重组 SQL*.LBA 文件的临时数据）

相关概念:

- 第 124 页的『SMS 和 DMS 表空间的比较』
- 『DMS 设备注意事项』（*《管理指南: 性能》*）
- 『SMS 表空间』（*《管理指南: 性能》*）
- 『DMS 表空间』（*《管理指南: 性能》*）
- 『DMS 表空间地址映射图表』（*《管理指南: 性能》*）
- 『了解恢复历史文件』（*《数据恢复及高可用性指南与参考》*）

相关参考:

- 『CREATE DATABASE Command』（*Command Reference*）

数据库对象的空间需求

数据库对象的大小估计不可能做到很精确。磁盘碎片、可用空间以及使用变长列所造成的开销都使大小估计变得十分困难，因为可能的列类型和行长度的范围实在是太广了。在最初估计数据库大小后，创建一个测试数据库，并用有代表性的数据填充它。

在“控制中心”中，可访问许多专门辅助您确定各种数据库对象的大小需求的实用程序:

- 可选择一个对象，然后使用“估计大小”实用程序。此实用程序可告诉您现有对象（例如，表）的当前大小。在更改该对象之后，该实用程序可以对该对象计算出新的估计值。该实用程序可以帮助您估计存储器需求（将未来的增长考虑在内）。它不仅仅是只能估计对象的大小。它还提供对象可能的大小范围: 最小大小（基于当前值）和可能的最大大小。
- 可通过使用“显示相关的”窗口来确定对象之间的关系。
- 可选择实例中的任何一个数据库对象，然后请求“生成 DLL”。此功能使用 **db2look** 实用程序来生成数据库的数据定义语句。

在这两种情况下，可使用“显示 SQL”或“显示命令”按钮。也可将生成的 SQL 语句或命令作为脚本文件保存，以便稍后使用。所有这些实用程序都有联机帮助可向您提供帮助。

在制订物理数据库需求计划时，应考虑这些实用程序。

估计数据库的大小时，必须考虑下列各项的影响:

- 系统目录表
- 用户表数据
- 长型字段数据
- 大对象（LOB）数据
- 索引空间
- 日志文件空间
- 临时工作区

不讨论与下列各项相关的空间需求:

- 本地数据库目录文件
- 系统数据库目录文件
- 操作系统所需的文件管理开销，包括:
  - 文件块大小
  - 目录控制空间

相关概念:

- 第 84 页的『系统目录表的空间需求』
- 第 85 页的『用户表数据的空间需求』
- 第 86 页的『长型字段数据的空间需求』
- 第 87 页的『大对象数据的空间需求』
- 第 88 页的『索引的空间需求』
- 第 90 页的『日志文件的空间需求』
- 第 92 页的『临时表的空间需求』

相关参考:

- 『db2look - DB2 Statistics and DDL Extraction Tool Command』（*Command Reference*）

系统目录表的空间需求

创建数据库时，会创建系统目录表。当将数据库对象和特权添加至该数据库时，这些系统表将增大。最初，这些系统表使用大约 3.5 MB 的磁盘空间。

为目录表分配的空间容量取决于表空间的类型和包含这些目录表的表空间的数据块大小。例如，若使用数据块大小为 32 的 DMS 表空间，则最初会分配给目录表空间 20MB 的空间。

注: 对于含多个分区的数据库，目录表只驻留在发出 CREATE DATABASE 命令的分区上。目录表的磁盘空间仅对于该分区才是必需的。

相关概念:

- 第 83 页的『数据库对象的空间需求』
- 『系统目录表的定义』（*《管理指南: 实现》*）

用户表数据的空间需求

在缺省情况下，表数据存储在 4KB 页上。每一页（不管页大小如何）都包含 68 个字节的数据库管理器开销。这将保留 4028 个字节来存放用户数据（或行），虽然 4 KB 页上的行的长度不能超过 4005 个字节。一行不会横跨多页。当使用 4KB 的页大小时，最多可有 500 列。

表数据页不包含用 LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB 或 DBCLOB 数据类型定义的列的数据。但是，一个表数据页中的行的确包含这些列的描述符。

通常，以“最先合适”次序将行插入到表中。（使用可用空间映射）搜索文件，查找大小足以存放该新行的第一个可用空间。当更新一行时，除非该页上所剩的空间不足以包含它，否则在原位置进行更新。若所剩空间不足以包含新行，则在原始行位置创建一个记录，以指向更新后的行在表文件中的新位置。

若调用 ALTER TABLE APPEND ON 语句，则将一直追加数据，且不保留关于数据页上任何可用空间的信息。

对于数据库中的每个用户表，可以通过如下计算公式来估计 4KB 页数:

```
ROUND DOWN(4028/(average row size + 10)) = records_per_page
```

然后，将结果插入:

```
(number_of_records/records_per_page) * 1.1 = number_of_pages
```

其中，平均行大小是平均列大小的总和，而因子“1.1”表示开销。

注: 此公式只是提供一个估计值。若记录长度因碎片和溢出记录而改变，则估计的准确性将降低。

也可以选择创建具有 8 KB、16 KB 或 32 KB 页大小的缓冲池或表空间。在特定大小的表空间中创建的所有表都将具有匹配的页大小。假设使用 32 KB 的页大

小，则单个表或索引对象的最大大小可达 512 GB。当使用 8 KB、16 KB 或 32 KB 的页大小时，最多可有 1012 列。对于 4 KB 的页大小，最大列数为 500。最大行宽也随页大小的不同而不同:

- 当页大小是 4 KB 时，行长度最大可为 4005 字节。
- 当页大小是 8 KB 时，行长度最大可为 8101 字节。
- 当页大小是 16 KB 时，行长度最大可为 16 293 字节。
- 当页大小是 32 KB 时，行长度最大可为 32 677 字节。

拥有更大的页大小有助于减小任何索引中的级别数。若使用执行随机行读写的 OLTP（联机事务处理）应用程序，则小一点的页大小会更好，这样，因意外的行而浪费的缓冲区空间更少。若使用一次存取大量连续行的 DSS（决策支持系统）应用程序，则大一点的页大小会更好，这样可以减少读取特定行数所需的 I/O 请求数。当行大小小于页大小除以 255 的值时，会发生异常。在这种情况下，每个页上都有浪费的空间。（每页最多只能有 255 行。）为减少浪费的空间，小一点的页大小可能更合适。

不能将备份复原为另一种页大小。

不能导入超过 755 列的 IXF 数据文件。

已声明临时表只能在它们自己的“用户临时”表空间类型的表空间中创建。没有缺省用户临时表空间。临时表不能含有 LONG 数据。当应用程序与数据库断开连接时，该表被隐式删除，估计它们的空间需求时应将这一点考虑在内。

相关概念:

- 第 83 页的『数据库对象的空间需求』

长型字段数据的空间需求

长型字段数据存储在单独的表对象中，它的结构与其它数据类型的存储空间不同。

数据存储在一些大小为 32 KB 的区域中，这些区域被分成大小为 512 字节的“2 的幂”倍的段。（因此，这些段可以是 512 字节、1024 字节 和 2048 字节，以此类推，直至 32 768 字节。）

长型字段数据类型（LONG VARCHAR 或 LONG VARGRAPHIC）以使可用空间易于收回的方式存储。有关分配和可用空间的信息存储在 4KB 分配页中，它在整个对象中不经常出现。

对象中未使用的空间量取决于长型字段数据的大小以及此大小是否在该数据的所有出现之处都是不变的。对于大于 255 字节的数据条目，这个未使用的空间最大可为该长型字段数据大小的 50%。

若该字符数据的长度小于页大小，且它适合含有该数据其余部分的记录，则应该使用 CHAR、GRAPHIC、VARCHAR 或 VARGRAPHIC 数据类型，而不要使用 LONG VARCHAR 或 LONG VARGRAPHIC。

相关概念:

- 第 83 页的『数据库对象的空间需求』

大对象数据的空间需求

大对象（LOB）数据存储在两个单独的表对象中，这两个对象的结构与其它数据类型的存储空间不同。

要估计 LOB 数据所需的空间，需要考虑用来存储使用这些数据类型定义的数据的两个表对象:

- **LOB 数据对象**

  数据存储在一些大小为 64 MB 的区域中，这些区域被分成大小为 1024 字节的“2 的幂”倍的段。（因此，这些段可以是 1024 字节、2048 字节和 4096 字节，以此类推，直至 64MB。）

  要减少 LOB 数据所用的磁盘空间量，可在 CREATE TABLE 和 ALTER TABLE 语句上的 *LOB 选项*子句上使用 COMPACT 参数。COMPACT 选项将所需的磁盘空间量减至最小，方法是将 LOB 数据分成更小的段。此过程不涉及数据压缩，只是使用最接近 1 KB 边界的最小空间量。使用 COMPACT 选项可能导致在追加 LOB 值时性能下降。

  包含在 LOB 数据对象中的可用空间量将受到更新和删除活动量以及要插入的 LOB 值的大小的影响。

- **LOB 分配对象**

  有关分配和可用空间的信息存储在与实际数据分离的 4 KB 分配页中。这些 4KB 页的数目取决于数据量，包括为大对象数据分配的未使用空间。开销的计算如下: 每 64 GB 一个 4 KB 页加上每 8 MB 一个 4KB 页。

若该字符数据的长度小于页大小，且它适合含有该数据其余部分的记录，则应该使用 CHAR、GRAPHIC、VARCHAR 或 VARGRAPHIC 数据类型，而不要使用 BLOB、CLOB 或 DBCLOB。

相关概念:

- 第 83 页的『数据库对象的空间需求』

相关参考:

- 『Large objects (LOBs)』（*SQL Reference, Volume 1*）

索引的空间需求

对于每个索引，可以按如下公式估计所需的空间:

```
(平均索引键大小 + 9) * 行数 * 2
```

其中:

- “平均索引键大小”是索引键中每列的字节计数。（估计 VARCHAR 和 VARGRAPHIC 列的平均列大小时，使用当前数据大小的平均值加上两个字节。不要使用最大声明大小。）
- 因子“2”表示开销，如非叶子页和可用空间。

注: 对于允许 NULL 的每个列，添加一个额外的字节以表示该空值指示符。

创建索引时，临时空间是必需的。在创建索引期间所需的最大临时空间可以按如下公式估计:

```
(平均索引键大小 + 9) * 行数 * 3.2
```

其中，因子”3.2“表示索引开销以及索引创建期间进行排序所需的空间。

注: 对于非唯一索引，存储重复键条目只需五个字节。上面显示的估计是假定没有重复的条目。因此以上公式可能会过多地估计存储索引所需的空间。

可使用下面两个公式来估计叶子页的数目（第二个公式提供更准确的估计）。这些估计的准确度很大程度上取决于平均值反映实际数据的准确程度。

注: 对于 SMS 表空间，所需的最小空间为 12 KB。对于 DMS 表空间，最小值是一个数据块。

- 每个叶子页的平均键数的粗略估计是:

```
(.9 * (U - (M*2))) * (D + 1)
----------------------------
      K + 7 + (5 * D)
```

  其中:

  – U（一页上的可用空间）大约等于页大小减 100。对于 4096 的页大小，U 是 3996。

  – M = U / (9 + *最小键大小*)

– D = 每个键值的平均重复项数目
– K = *平均键大小*

记住，*最小键大小*和*平均键大小*必须有一个额外字节，表示每个可空键部分；还必须有两个额外字节，表示每个变长键部分的长度。

若存在包含列，则在计算*最小键大小*和*平均键大小*时应将它们考虑在内。

若在创建索引期间指定了非缺省值 10% 的一个可用百分比，则可以使用任何 (100 - pctfree)/100 替换 .9。

- 每个叶子页的平均键数的更准确估计是:

  ```
  L = 叶子页数 = X / (叶子页上的平均键数)
  ```

  其中，X 是表中的总行数。

  可按如下方法估算索引的原始大小:

  ```
  (L + 2L/(叶子页上的平均键数)) * 页大小
  ```

  对于 DMS 表空间，将一个表上所有索引的大小加在一起，然后四舍五入为该索引所在表空间的数据块大小的一个倍数。

  应该为 INSERT/UPDATE 活动所引起的索引增长提供附加空间，这种增长可能导致分页。

  使用下列计算方法来获得更精确的原始索引大小的估算值，以及该索引中级别数的估算值。（若索引定义中使用包含列，可能要引起特别注意。）每个非叶子页的平均键数大约是:

  ```
  (.9 * (U - (M*2))) * (D + 1)
  ----------------------------
        K + 13 + (9 * D)
  ```

  其中:

  – U（一页上的可用空间）大约等于页大小减 100。对于 4096 的页大小，U 是 3996。
  – D 是在非叶子页上每个键值的重复值的平均数目（这将比在叶子页上的小很多，您可能想将该值设置为 0 以便简化计算）。
  – M = U / (9 + 非叶子页的*最小键大小*)
  – K = 非叶子页的*平均键大小*

  只要没有包含列，非叶子页与叶子页的*最小键大小*和*平均键大小*将是相同的。包含列不存储在非叶子页上。

除非 (100 - pctfree)/100 大于 .9，否则不应使用它来替换 .9，因为在创建索引期间会在非叶子页上留下 10% 的可用空间。

可用如下所示的方法估算非叶子页数:

```
if L > 1 then {P++; Z++}
While (Y > 1)
{
   P = P + Y
   Y = Y / N
  Z++
}
```

其中:

– P 是页数（最初为 0）。

– L 是叶子页数。

– N 是每个非叶子页的键数。

– Y = L / N

– Z 是索引树中的级别数（最初为 1）。

总页数是:

```
T = (L + P + 2) * 1.0002
```

附加的 0.02% 表示开销，包括空间映射页。

创建索引所需的空间容量估算为:

```
T * 页大小
```

相关概念:

- 『Indexes』（*SQL Reference, Volume 1*）
- 第 83 页的『数据库对象的空间需求』
- 『索引清除和维护』（*《管理指南: 性能》*）

日志文件的空间需求

日志控制文件需要 32 KB 的空间。

至少还需要足够的空间以进行活动日志配置，可进行如下计算

```
(logprimary + logsecond) * (logfilsiz + 2 ) * 4096
```

其中:

- *logprimary* 是在数据库配置文件中定义的主日志文件数

- *logsecond* 是在数据库配置文件中定义的辅助日志文件的数目；在此计算中，不能将 *logsecond* 设置为 *-1*（当 *logsecond* 设置为 -1 时，您是在请求一个无限的活动日志空间。）
- *logfilsiz* 是在数据库配置文件中定义的每个日志文件中的页数
- 2 是每个日志文件所需的标题页的数目
- 4096 是一页中的字节数

如果已对数据库启用了循环记录，则此公式的结果将提供足够的磁盘空间。

若允许对该数据库执行前滚恢复，应该考虑特殊的日志空间需求:

- 当启用 *logretain* 配置参数时，日志文件将被归档在日志路径目录中。除非将日志文件移至另一个位置，否则，联机磁盘空间最终将会填满。
- 当启用 *userexit* 配置参数时，用户出口程序会将归档的日志文件移至另一个位置。要允许下列情况，附加的日志空间仍是必需的:
  - 等待用户出口程序移动的联机归档日志。
  - 要格式化以供将来使用的新日志文件。

如果已对数据库启用了无限记录（即，将 *logsecond* 设置为 *-1*），则必须启用 *userexit* 配置参数，以便具有相同的磁盘空间注意事项。DB2® 将保留至少由日志路径中的 *logprimary* 指定的数目的活动日志文件，所以上面公式中 *logsecond* 的值不应使用 －1。确保提供额外磁盘空间以允许归档日志文件导致的延迟。

如果在镜像日志路径，则需要将估计的日志文件空间需求加倍。

相关概念:

- 第 83 页的『数据库对象的空间需求』
- 『了解恢复日志』（*《数据恢复及高可用性指南与参考》*）
- 『日志镜像』（*《数据恢复及高可用性指南与参考》*）

相关参考:

- 『“日志文件大小”配置参数 — logfilsiz』（*《管理指南: 性能》*）
- 『“主日志文件的数目”配置参数 — logprimary』（*《管理指南: 性能》*）
- 『“辅助日志文件数目”配置参数 — logsecond』（*《管理指南: 性能》*）
- 『“镜像日志路径”配置参数 — mirrorlogpath』（*《管理指南: 性能》*）

临时表的空间需求

某些 SQL 语句需要临时表来进行处理（如使用一个工作文件来进行不能在内存中执行的排序）。这些临时表需要磁盘空间；所需的空间量取决于查询以及返回的表的大小，这些无法估计。

可以使用数据库系统监视器和查询表空间 API 来跟踪在正常操作期间所用的工作空间量。

相关概念:

- 第 83 页的『数据库对象的空间需求』

相关参考:

- 『sqlbmtsq - Table Space Query』（*Administrative API Reference*）

数据库分区组

数据库分区组是一个或多个数据库分区的集合。想要为数据库创建表时，首先创建用来存储表空间的数据库分区组，然后创建用来存储表的表空间。

可以在数据库中定义一个或多个数据库分区组成的命名子集。您定义的每个子集称为*数据库分区组*。包含多个数据库分区的每个子集称为*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定义。

第 93 页的图 32 给出了一个含五个分区的数据库示例，在这个示例中:

- 数据库分区组横跨除一个数据库分区外的所有其它分区（数据库分区组 1）。
- 数据库分区组包含一个数据库分区（数据库分区组 2）。
- 数据库分区组包含两个数据库分区（数据库分区组 3）。
- 数据库分区组 2 中的数据库分区与数据库分区组 1 共享并与之相交。
- 数据库分区组 3 中存在单个数据库分区，该分区与数据库分区组 1 共享并与之相交。

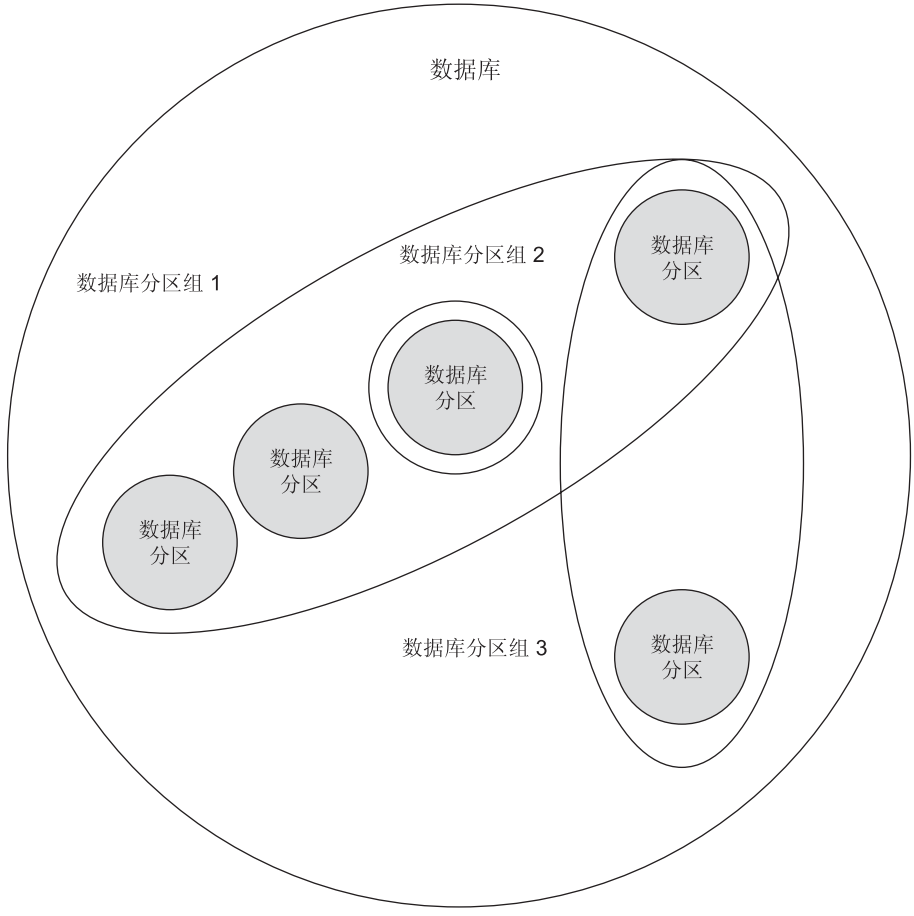

图 32. 数据库中的数据库分区组

使用 CREATE DATABASE PARTITION GROUP 语句创建新的数据库分区组。可以使用ALTER DATABASE PARTITION GROUP 语句修改它。数据分布在数据库分区组中的所有分区上，并且可以从数据库分区组添加或删除一个或多个数据库分区。如果正在使用多分区数据库分区组，则必须查看几个数据库分区组设计注意事项。

作为数据库系统配置一部分的每个数据库分区，必须已在称为 *db2nodes.cfg* 的分区配置文件中定义。数据库分区组可以只包含一个数据库分区，也可以包含为该数据库系统定义的所有数据库分区。

创建或修改数据库分区组时，*分区映射*与它是相关联的。数据库管理器将分区映射和*分区键*及散列算法联合使用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。

使用非分区数据库时，不需要任何分区键或分区映射。数据库分区是数据库的一部分，与用户数据、索引、配置文件和事务日志相辅相成。创建数据库时创建的缺省数据库分区组由数据库管理器使用。IBMCATGROUP 是包含系统目录的表空间的缺省数据库分区组。IBMTEMPGROUP 是系统临时表空间的缺省数据库分区组。IBMDEFAULTGROUP 是包含可选择将其放置其中的用户定义表的表空间的缺省数据库分区组。已声明临时表的用户临时表空间可在 IBMDEFAULTGROUP 或用户创建的任何数据库分区组中创建，但不能在 IBMTEMPGROUP 中创建。

相关概念:

- 第 94 页的『数据库分区组设计』
- 第 95 页的『分区映射』
- 第 96 页的『分区键』

相关参考:

- 『ALTER DATABASE PARTITION GROUP statement』（*SQL Reference, Volume 2*）
- 『CREATE DATABASE PARTITION GROUP statement』（*SQL Reference, Volume 2*）

数据库分区组设计

若使用非分区数据库，则不考虑数据库分区组设计。

若使用多分区数据库分区组，应考虑下列设计要点:

- 在多分区数据库分区组中，若唯一索引是该分区键的超集，则只能创建唯一索引。
- 根据该数据库中的数据库分区的数目，可能有一个或多个单分区数据库分区组，且存在一个或多个多分区数据库分区组。
- 必须给每个数据库分区指定唯一的分区号。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，则须避免将用户表放在同一个数据库分区上。这可通过将用户表放在不包括 IBMCATGROUP 数据库分区组中的那些数据库分区的数据库分区组中来实现。

除非要与一个更大的表*并置*，否则应该将小表放在单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。并置的表允许 DB2® 使用更有效的连接策略。并置的表可以驻留在单分区数据库分区组中。若表驻留在多分区数据库分区组中，且在分区键中具有相同数目的列，对应列的数据类型

是分区兼容的，则将这些表视为并置。在并置的表中具有相同分区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。

应该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 16 个分区的数据库分区组中可能比在 32 个分区的数据库分区组中执行得更好。

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，以确保不会对 OLTP 事务的性能产生负面影响。

相关概念:

- 第 92 页的『数据库分区组』
- 第 95 页的『分区映射』
- 第 96 页的『分区键』
- 第 98 页的『表整理』
- 第 99 页的『分区兼容性』
- 第 100 页的『复制具体查询表』

分区映射

在分区数据库环境中，数据库管理员必须设法弄清一个表的哪些行存储哪些数据库分区上。数据库管理器必须知道到哪里去查找所需的数据，并使用一个称为*分区映射*的映射来查找数据。

分区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 4 096 个条目，对于单分区数据库分区组，只包含一个条目。对于单分区数据库分区组，分区映射只有一个条目，该条目包含存储数据库表的所有行的数据库分区的分区号。对于多分区数据库分区组，以循环方式指定数据库分区组的分区号。正如使用网格将城市地图划分为区一样，数据库管理员使用*分区键*来确定存储数据的位置（数据库分区）。

例如，假定您将一个数据库创建在四个数据库分区（编号为 0–3）上。此数据库的 IBMDEFAULTGROUP 数据库分区组的分区映射将是:

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

若已使用数据库分区 1 和 2 在该数据库中创建了一个数据库分区组，则该数据库分区组的分区映射将是:

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

若要装入数据库的一个表的分区键是一个可能在范围 1 至 500 000 之间的整数，则会将分区键散列至 0 至 4 095 之间的一个分区号。将该编号用作分区映射中的索引，以选择用于该行的数据库分区。

图 33 显示如何将具有分区键值 (c1, c2, c3) 的行映射至分区 2，然后引用数据库分区 n5。

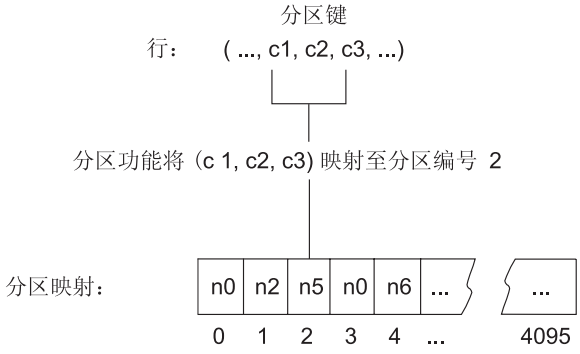

图 33. 使用分区映射的数据分布

分区映射可以灵活地控制将数据存储在分区数据库中的哪个位置。若将来需要更改数据库中各数据库分区上的数据分布，可以使用数据重新分布实用程序。此实用程序允许重新协调或调整数据分布。

可以使用“获取表分区信息”（**sqlugtpi**）API来获取您可以查看的分区映射的副本。

相关概念:

- 第 92 页的『数据库分区组』
- 第 94 页的『数据库分区组设计』
- 第 96 页的『分区键』

相关参考:

- 『sqlugtpi - Get Table Partitioning Information』（*Administrative API Reference*）

分区键

*分区键*是一列（或一组列），用于确定将某行数据存储在什么分区。分区键是使用 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区键，在缺省情况下将会根据主键的第一列创建分区键。若未指定主键，则缺省分区键是在该表中定义的第一个非长型字

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）。如果正在与单分区数据库分区组相关的表空间中创建表，且您希望有分区键，则必须显式定义该分区键。缺省情况下，不会创建它。

若没有列满足缺省分区键的要求，则会不带键创建该表。只有在单分区数据库分区组中才允许不带分区键的表。以后可以使用 ALTER TABLE 语句添加或删除分区键。只能对其表空间与单分区数据库分区组相关联的表改变分区键。

选择好的分区键很重要。应考虑下列各项:

- 如何存取表
- 查询工作负荷的性质
- 数据库系统所使用的连接策略

若并置不是主要的注意事项，则好的表分区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区键。与数据库分区组相关联的表空间中每个表的分区键确定这些表是否已并置。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
- 每个表中的分区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区键值的那些行位于同一个分区上。

不适当的分区键会导致数据分布不均匀。数据分布不均匀的列和含有少数特异值的列不应选作分区键。特异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。应用分区散列算法的成本与分区键的大小是成正比的。分区键不能超过 16 列，而且列越少，性能越好。不应将不需要的列包括在分区键中。

当定义分区键时，应该考虑下列几点:

- 不支持创建只包含长型数据类型（LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB 或 DBCLOB）的多分区表。
- 不能改变分区键定义。
- 分区键应该包括最频繁连接的列。
- 分区键应该由经常参与 GROUP BY 子句的列组成。
- 任何唯一键或主键必须包含所有分区键列。
- 在联机事务处理（OLTP）环境中，分区键中的所有列都应该使用带常量或主机变量的等于（=）谓词来参与该事务。例如，假定有一个在事务中经常使用的雇员号 *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区键。

*散列分区*是确定分区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区键的值，并生成介于 0 与 4095 之间的分区号。
2. 创建数据库分区组时将创建分区映射。每个分区号依次以循环方式重复，以填写该分区映射。
3. 该分区号用作分区映射中的一个索引。分区映射中该位置处的编号是存储该行的数据库分区的编号。

相关概念:

- 第 92 页的『数据库分区组』
- 第 94 页的『数据库分区组设计』
- 第 95 页的『分区映射』

相关参考:

- 『ALTER TABLE statement』（*SQL Reference, Volume 2*）

表整理

您可能会发现，作为对特定查询的响应，两个或多个表频繁地提供数据。在此情况下，您会希望这样的表中的相关数据的位置尽可能地靠近。在数据库被物理地划分为两个或多个数据库分区的环境中，必须有一种方法可将划分的表的相关碎片尽可能地靠近。完成此过程的功能称为*表并置*。

当表存储在同一个数据库分区组中且它们的分区键兼容时，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区映射。这些表可能位于不同的表空间，但是这些表空间必须与相同数据库分区组相关联。每个分区键中对应列的数据类型必须是*分区兼容*的。

当存取用于连接或子查询的多个表时，DB2® 能够识别要连接的数据是否位于相同数据库分区上。当发生这种情况时，DB2 可以选择在存储该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。这种在数据库分区上执行连接或子查询的能力具有显著的性能优点。

相关概念:

- 第 92 页的『数据库分区组』
- 第 94 页的『数据库分区组设计』

- 第 96 页的『分区键』
- 第 99 页的『分区兼容性』

分区兼容性

对分区键的对应列的基本数据类型进行比较，并可将它们声明为是*分区兼容*的。分区兼容的数据类型具有如下特性：具有相同值但有不同类型的两个变量会按相同的分区算法映射至同一个分区号。

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，且都不与 CHAR 兼容。
- 分区兼容性不受带有 NOT NULL 或 FOR BIT DATA 定义的列的影响。
- 对兼容数据类型的 NULL 值的处理是完全相同的；对不兼容数据类型的 NULL 值的处理可能不相同。
- 用户定义类型的基本数据类型用于分析分区兼容性。
- 对分区键中值相同的小数的处理是完全相同的，即使它们的标度和精度不同。
- 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。
- REAL 和 FLOAT 是兼容的数据类型。
- 不同长度的 CHAR 和 VARCHAR 是兼容的数据类型。
- GRAPHIC 和 VARGRAPHIC 是兼容的数据类型。
- 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区键。

相关概念:

- 第 92 页的『数据库分区组』
- 第 94 页的『数据库分区组设计』
- 第 96 页的『分区键』

复制具体查询表

*具体查询表*是由查询定义的表，也用于确定表中的数据。具体查询表可用来改进查询的性能。若 DB2® 确定查询的一部分可使用具体查询表来解决，则数据库管理器可重写该查询以使用具体查询表。

在分区数据库环境中，可复制具体查询表。可使用*复制具体查询表*来改进查询性能。复制具体查询表基于可能已在单分区数据库分区组中创建，但您想要在数据库分区组中的所有数据库分区中进行复制的表。要创建复制具体查询表，调用带 REPLICATED 关键字的 CREATE TABLE 语句。

通过使用复制具体查询表，可将平时未并置的表并置。对于大事实表和小维度表的连接，复制具体查询表特别有用。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: 复制的一次性成本可通过并置获得的性能效益来抵消。

通过在定义复制表所用的 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。

相关概念:

- 第 94 页的『数据库分区组设计』

相关任务:

- 『创建具体查询表』（*《管理指南: 实现》*）

相关参考:

- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）

表空间设计

表空间是一个存储结构，包含表、索引、大对象和长型数据。表空间驻留在数据库分区组中。它们允许将数据库和表数据的位置直接指定到容器上。（容器可以是目录名、设备名或文件名。）这可以提供改善的性能和更灵活的配置。

因为表空间驻留在数据库分区组中，所以选择保存表的表空间定义如何将该表的数据分布至数据库分区组中的数据库分区。单个表空间可跨多个容器。在同一个物理磁盘（即驱动器）上创建多个容器（一个或多个表空间）是可能的。为提高性能，每个容器应使用不同的磁盘。第 101 页的图 34 举例说明了一个数据库内的表和表空间与该数据库相关的容器之间的关系。

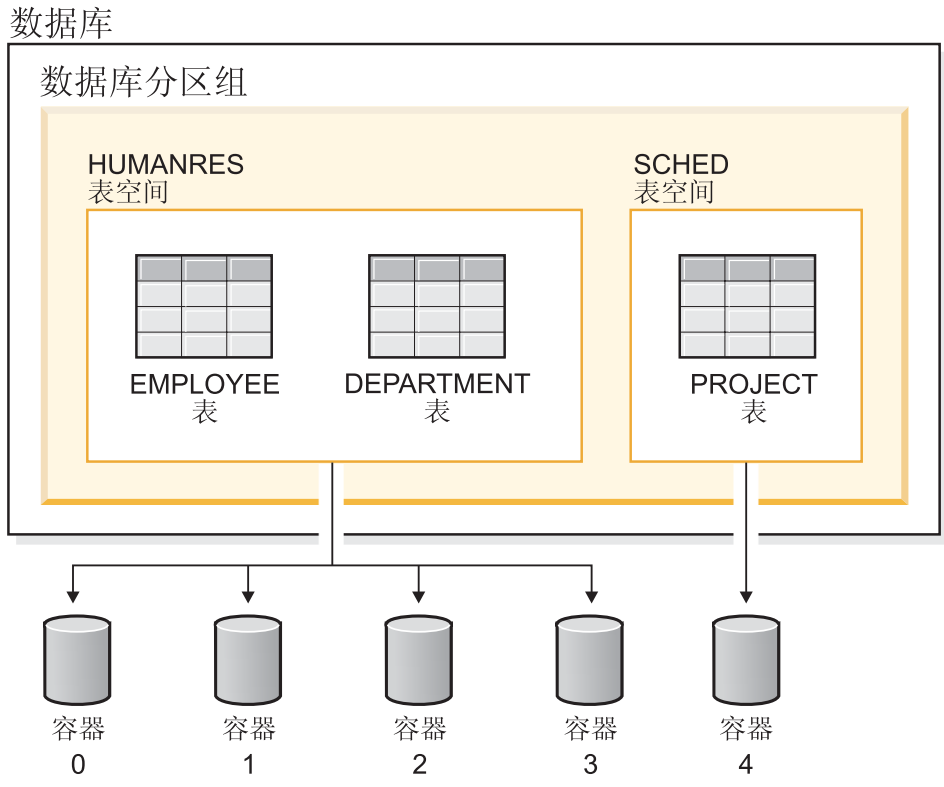

图 34. 数据库中的表空间和表

EMPLOYEE 和 DEPARTMENT 表在 HUMANRES 表空间中，该表空间跨越容器 0、1、2 和 3。PROJECT 表在容器 4 的 SCHED 表空间中。此示例显示每个容器存在于不同的磁盘上。

数据库管理器会尝试平衡分布在所有容器中的数据负荷。因此，所有容器都将用于存储数据。数据库管理器在使用另一个容器之前写入一个容器的页数称为*数据块大小*。数据库管理器并非始终从第一个容器开始存储表数据。

第 102 页的图 35 显示具有两个 4KB 页数据块大小的 HUMANRES 表空间，它有四个容器，每个容器有少量已分配的数据块。DEPARTMENT 和 EMPLOYEE 表都有 7 页，且跨所有四个容器。

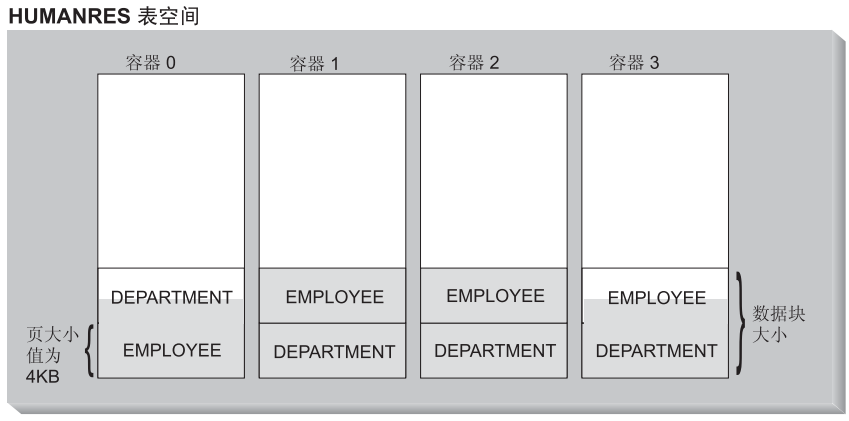

图 35. 容器和数据块

一个数据库至少必须包含三个表空间:

- 一个*目录表空间*，它包含该数据库的所有系统目录表。此表空间称为 SYSCATSPACE，它不能被删除。IBMCATGROUP 是此表空间的缺省数据库分区组。
- 一个或多个*用户表空间*，它们包含所有用户定义表。缺省情况下，会创建一个表空间 USERSPACE1。IBMDEFAULTGROUP 是此表空间的缺省数据库分区组。

  当创建一个表时，应指定表空间名，否则，结果可能不是您所期望的。

  表的页大小或者由行大小确定，或者由列数确定。一行中允许的最大长度取决于创建此表所在的表空间的页大小。可能的页大小的值为 4 KB（缺省值）、8 KB、16 KB 和 32 KB。可使用具有一种页大小的表空间作为基本表，而使用具有另一种页大小的另一个表空间来存储长型数据或 LOB 数据。（记住，SMS 不支持跨表空间的表，而 DMS 却支持。）若列数或行大小超过表空间页大小的限制，则返回一个错误（SQLSTATE 42997）。
- 一个或多个*临时表空间*，它们包含临时表。临时表空间可以是*系统临时表空间*或*用户临时表空间*。数据库必须有至少一个系统临时表空间；在缺省情况下，创建数据库时会创建一个名为 TEMPSPACE1 的系统临时表空间。IBMTEMPGROUP 是此表空间的缺省数据库分区组。用户临时表空间*并非*在创建数据库时缺省创建的。

  如果数据库使用多个临时表空间，并且需要新的临时对象，则优化器将为此对象选择相应的页大小。然后将把该对象分配到具有相应页大小的临时表空间中。若存在多个临时表空间具有该页大小，则将以循环方式选择表空间。在大多数情况下，建议对于任何一个页大小，不要使用多个临时表空间。

如果对用 4 KB（缺省值）以上的页大小定义的表空间中的表运行查询（例如，对 1012 列使用 ORDER BY），则某些查询可能会失败。若没有使用更大页大小定义的临时表空间，则会发生这种情况。可能需要创建具有更大页大小（8 KB、16 KB 或 32 KB ）的临时表空间。若没有与用户表空间中的最大页大小具有相同页大小的临时表空间，任何一种 DML（数据操作语言）语句都可能失败。

应定义一个 SMS 临时表空间，使它的页大小等于大多数用户表空间所使用的页大小。这对于典型的环境和工作负荷应已足够。

在分区数据库环境中，目录节点将包含全部三个缺省表空间，而其它每个数据库分区将只包含 TEMPSPACE1 和 USERSPACE1。

有两种类型的表空间，它们都可以在单个数据库中使用:

- 系统管理空间，其中操作系统的文件管理器控制存储空间。
- 数据库管理空间，其中数据库管理器管理器控制存储空间。

相关概念:

- 『Table spaces and other storage structures』（*SQL Reference, Volume 1*）
- 第 104 页的『系统管理空间』
- 第 106 页的『数据库管理空间』
- 第 124 页的『SMS 和 DMS 表空间的比较』
- 第 125 页的『表空间磁盘 I/O』
- 第 126 页的『表空间设计中的工作负荷注意事项』
- 第 128 页的『数据块大小』
- 第 129 页的『表空间和缓冲池之间的关系』
- 第 130 页的『表空间和数据库分区组之间的关系』
- 第 130 页的『临时表空间设计』
- 第 132 页的『目录表空间设计』

相关任务:

- 『创建表空间』（*《管理指南: 实现》*）
- 第 132 页的『当数据在 RAID 设备上时优化表空间性能』

相关参考:

- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）
- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）

系统管理空间

在 SMS（系统管理空间）表空间中，操作系统的文件系统管理器分配和管理用于存储表的空间。该存储模型通常由存储在文件系统空间中的多个文件组成，这些文件表示表对象。用户决定文件的位置，DB2® 控制它们的名称，而文件系统则负责管理它们。通过控制写入每个文件的数据量，数据库管理器均匀地将数据分布在所有表空间容器中。缺省情况下，在创建数据库时创建的初始表空间是 SMS。

每个表至少有一个与它相关的 SMS 物理文件。

在 SMS 表空间中，当对象增大时，文件每次扩展一页。若需要提高插入性能，可以考虑启用多页文件分配。这允许系统一次为文件分配或扩展多页。由于性能的原因，如果您将多维（MDC）表存储在 SMS 表空间中，则应启用多页文件分配。运行 **db2empfa**，以启用多页文件分配。在分区数据库环境中，必须对每个数据库分区运行此实用程序。一旦启用了多页文件分配，就不能禁用它。

SMS 表空间是使用 CREATE DATABASE 命令上或 CREATE TABLESPACE 语句上的 MANAGED BY SYSTEM 选项定义的。当设计 SMS 表空间时，必须考虑两个关键要素:

- 表空间的容器。

  必须指定要用于表空间的容器数。标识要使用的所有容器是非常重要的，因为您不能在创建了 SMS 表空间之后添加或删除容器。在分区数据库环境中，在将新分区添加至 SMS 表空间的数据库分区组时，可以使用 ALTER TABLESPACE 语句为新的分区添加容器。

  用于一个 SMS 表空间的每个容器都标识一个绝对或相对的目录名。其中每一个目录都可以位于不同的文件系统（物理磁盘）上。表空间的最大大小可以按以下方法估计:

  ```
  容器数 * (操作系统支持的最大文件系统大小)
  ```

  此公式假定有一个唯一的文件系统映射至每个容器，且每个文件系统都具有最大量的可用空间。实际上，情况可能不是这样，表空间最大大小可能小得多。对于数据库对象的大小也有 SQL 限制，它可能影响表空间的最大大小。

  注: 定义容器时必须很小心。若容器上已有文件或目录，将返回一个错误（SQL0298N）。

- 表空间的数据块大小。

  数据块大小只能在创建表空间时指定。因为以后不能更改它，因此为数据块大小选择一个适当的值就很重要。

创建表空间时，如果不指定数据块大小，数据库管理器将使用缺省数据块大小来创建表空间，该缺省大小由 *dft_extent_sz* 数据库配置参数定义。此配置参数最初是根据创建该数据库时提供的信息设置的。若未在 CREATE DATABASE 命令上指定 *dft_extent_sz* 参数，则会将缺省数据块大小设置为 32。

要为表空间的容器数和数据块大小选择适当的值，必须了解:

- 操作系统对逻辑文件系统的大小施加的限制。

  例如，某些操作系统有 2GB 的限制。因此，若想要一个 64GB 的表对象，则在此类型的系统上将需要至少 32 个容器。

  当创建该表空间时，可以指定驻留在不同文件系统上的容器，以便增加可以存储在该数据库中的数据量。

- 数据库管理器如何管理与一个表空间相关的数据文件和容器。

  在为该表空间指定的第一个容器中创建第一个表数据文件（SQL00001.DAT），并允许此文件增大至该数据块大小。当它达到此大小之后，数据库管理器将数据写入下一个容器中的 SQL00001.DAT。此过程会继续，直到所有容器都包含 SQL00001.DAT 文件为止，在那时，数据库管理器会返回至第一个容器。此过程（称为*划分带区*）会继续在容器目录中运行，直到一个容器装满为止（SQL0289N）或操作系统中不再有空间可分配为止（磁盘已满错误）。划分带区也用于索引（SQL*nnnnn*.INX）、长型字段（SQL*nnnnn*.LF）和 LOB（SQL*nnnnn*.LB 和 SQL*nnnnn*.LBA）文件。

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

  为了有助于将数据更加均匀地分布至这些容器中，数据库管理器根据将表标识符（以上示例中为 1）除以容器数所得的系数来确定首先使用的容器。容器从 0 开始依次编号。

相关概念:

- 第 100 页的『表空间设计』
- 第 106 页的『数据库管理空间』
- 第 124 页的『SMS 和 DMS 表空间的比较』

相关参考:

- 『db2empfa - Enable Multipage File Allocation Command』（*Command Reference*）

数据库管理空间

在 DMS（数据库管理空间）表空间中，数据库管理器控制存储空间。存储器模型由一定数目的设备或文件（其空间由 DB2 管理）组成。数据库管理员决定使用哪些设备和文件并且 DB2® 管理这些设备和文件上的空间。此表空间实质上是为了最好地满足数据库管理器的需要而设计的特殊目的的文件系统。

包含用户定义的表和数据的 DMS 表空间可以定义为:

- 存储所有表数据和索引数据（可选）的*常规*表空间。
- 存储长型字段或 LOB 数据或索引数据的大表空间。

在设计 DMS 表空间和容器时，应该考虑下列事项:

- 数据库管理器使用“拆开”来确保数据均匀地分布在所有容器上。
- 对于 4 KB 页，规则表空间的最大大小是 64 GB；对于 8 KB 页，是 128 GB；对于 16 KB 页，是 256 GB；对于 32 KB 页，是 512 GB。大表空间的最大大小是 2 TB。
- 与 SMS 表空间不同，组成一个 DMS 表空间的容器不必大小相同；然而，通常不建议这样做，因为会导致在容器间不均匀地进行划分带区，并会降低性能。若任何容器已满，DMS 表空间会使用其它容器中的可用空间。
- 因为空间是预分配的，所以在能够创建表空间之前，该空间必须是可用的。当使用设备容器时，该设备也必须有足够的空间以便存储容器定义。每个设备上只能定义一个容器。为避免浪费空间，设备的大小应该等于容器的大小。例如，若分配给该设备 5 000 页，而将设备容器定义为分配 3 000 页，则设备上的 2 000 页将是不可用的。
- 在缺省情况下，每个容器中都保留一个数据块作为开销。只使用整个数据块，因此为了对空间进行最优管理，可以使用如下公式来帮助您确定当分配容器时要使用的适当大小:

  ```
  extent_size * (n + 1)
  ```

  其中，*extent_size* 是表空间中每个数据块的大小，而 *n* 是您要在该容器中存储的数据块数。
- DMS 表空间的最小大小是五个数据块。试图创建小于五个数据块的表空间将产生错误（SQL1422N）。
  - 表空间中有三个数据块是保留给开销使用的。
  - 要存储任何用户表数据，至少需要两个数据块。（这些数据块是一个表的规则数据所必需的，但不是任何索引、长型字段或大对象数据所需的，它们需要自己的数据块。）
- 设备容器必须使用带“字符型特殊接口”的逻辑卷，而不是物理卷。

- 在 DMS 表空间中可以使用文件以代替设备。文件和设备之间不存操作性区别; 但是，由于存在与文件系统相关的运行时开销，文件效率会更低。在下列情况下，适合使用文件:
  - 设备不直接受支持
  - 设备不可用
  - 不需要最大性能
  - 不想设置设备
- 若工作负荷涉及 LOB 或 LONG VARCHAR 数据，则可通过文件系统高速缓存改进性能。注意，DB2 的缓冲池不缓冲 LOB 和 LONG VARCHAR。
- 某些操作系统允许拥有大小超过 2GB 的物理设备。应该考虑将该物理设备分区为多个逻辑设备，以使任何容器都不超过操作系统允许的大小。

相关概念:

- 第 100 页的『表空间设计』
- 第 104 页的『系统管理空间』
- 第 124 页的『SMS 和 DMS 表空间的比较』
- 第 107 页的『表空间映射』
- 第 111 页的『在 DMS 表空间中如何添加和扩展容器』

表空间映射

表空间映射是 DB2 的 DMS 表空间的内部表示，它描述表空间中页位置的逻辑至物理的转换。以下信息描述表空间映射为何有用，以及表空间映射中的信息从何而来。

在 DB2® 数据库中，DMS 表空间中的页从 0 到（N-1）逻辑编号，其中 N 是表空间中的可用页的数目。

DMS 表空间中的页组成数据块（基于数据块大小），并且从表空间管理的角度来看，所有对象分配都是以数据块为基础完成的。即，表可能仅使用数据块中的一半页，但是认为整个数据块都在使用中，并且由该对象所有。缺省情况下，使用一个数据块来存放容器标记，并且此数据块中的页不能用来存放数据。但是，如果 DB2_USE_PAGE_CONTAINER_TAG 注册表变量打开，则仅将一页用作容器标记。

因为容器中的空间是按数据块来分配的，所以不会使用未组成完整数据块的页。例如，如果您具有 205 页的容器，且数据块大小为 10，则 1 个数据块将用于存放标记，19 个数据块将用于存放数据，并且剩余的 5 页将被浪费。

如果 DMS 表空间包含单个容器，则从逻辑页编号到磁盘上的物理位置的转换是直接进行的过程，其中 0、1 和 2 将以磁盘上的相同次序定位。

在有多个容器并且每个容器大小相同的情况下，它也是相当直接的过程。表空间（包含页 0 到（数据块大小 -1））中第一个数据块将位于第一个容器中，第二个数据块将位于第二个容器中，如此类推。在最后一个容器之后，过程将以循环的样式重复，从第一个容器开始。

对于包含不同大小容器的表空间，不能使用简单循环方法，因为它不会利用大型容器中的额外空间。这就是引入表空间映射的位置 — 它规定如何在表空间内定位数据块，确保物理容器中的所有数据块都可用。

注: 在下列示例中，容器大小未将容器标记计算在内。容器大小很小，仅用于说明目的，它们不是建议的容器大小。这些示例显示表空间中不同大小的容器，但建议使用相同大小的容器。

示例 1:

表空间中有 3 个容器，每个容器包含 80 个可用的页，并且表空间的数据块大小是 20。因此，每个容器将有 4 个数据块（80 / 20），总数为 12 个数据块。这些数据块将位于磁盘上，如图 36 中所示。

图 36. 带有三个容器和 12 个数据块的表空间

要查看表空间映射，使用快照监视器获取表空间快照。在示例 1 中，其中三个容器大小相等，表空间映射为如下所示:

```
范围    分割    组合分割区    最大      最大    起始          结束          调节    容器
编号    集      偏移          数据块    页      组合分割区    组合分割区
 [0]     [0]             0        11   239               0             3       0   3（0、1 和 2）
```

*范围*是其中连续范围的组合分割区包含同一组容器的一段映射。在示例 1 中，所有的组合分割区（0 到 3）都包含同一组 3 个容器（0、1 和 2），因此认为它是单个范围。

表空间映射中的标题是“范围编号”、“分割集”、“组合分割区偏移”、“根据范围寻址的最大数据块编号”、“根据范围寻址的最大页编号”、“起始组合分割区”、“结束组合分割区”、“范围调节”和“容器列表”。对于示例 2，将会更详细地说明它们。

还可以对此表空间进行图解（如图 37 中所示），其中每条竖线对应一个容器，每条横线都称为*组合分割区*，并且每个单元编号对应一个数据块。

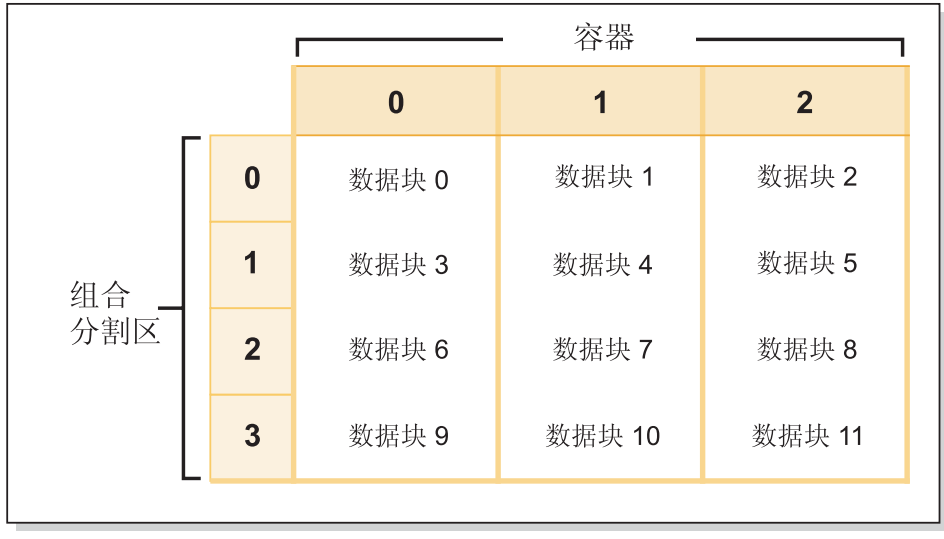

图 37. 带有三个容器和 12 个数据块，突出显示组合分割区的表空间

示例 2:

表空间中有两个容器：第一个大小是 100 页，第二个大小是 50 页并且数据块大小是 25。这意味着第一个容器具有四个数据块并且第二个容器具有两个数据块。可以对表空间进行图解，如第 110 页的图 38 中所示。

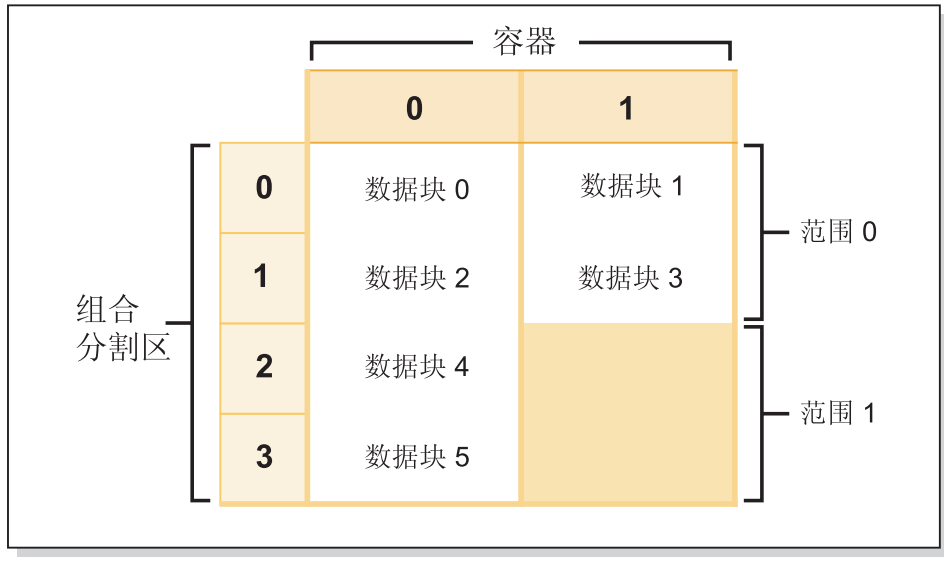

图 38. 带有两个容器，突出显示范围的表空间

组合分割区 0 和 1 包含两个容器（0 和 1），但是组合分割区 2 和 3 仅包含第一个容器（0）。这些分割集的每一个都是一个范围。表空间映射，如表空间快照中所示，类似如下:

```
范围    分割    组合分割区      最大      最大    起始          结束          调节    容器
编号    集      偏移          数据块    页      组合分割区    组合分割区
 [0]    [0]             0          3     99             0             1       0   2（0 和 1）
 [1]    [0]             0          5    149             2             3       0   1（0）
```

在第一个范围中有四个数据块，因此在此范围中寻址的最大数据块编号（最大数据块）为 3。每个数据块有 25 页，因此在第一个范围中有 100 页。因为页的最大编号也是从 0 开始，所以此范围中寻址的最大页编号（最大页）是 99。此范围中的第一个组合分割区（起始组合分割区）是 0 并且最后一个组合分割区（结束组合分割区）是 1。此范围中有两个容器并且它们是 0 和 1。组合分割区偏移是分割集中的第一个分割，因为只有一个分割集，所以在此情况下它是 0。范围调节（调节）是在表空间中再平衡数据时使用的偏移量。（当在表空间中添加或删除空间时会发生再平衡。）当没有再平衡时，它将始终为 0。

在第二个范围中有两个数据块并且因为先前范围中寻址的最大数据块编号是 3，所以此页中寻址的最大数据块编号是 5。在第二个范围中有 50 页（2 个数据块 * 25 页）并且由于在先前范围中寻址的最大页编号是 99，所以在此范围中寻址的最大页编号是 149。此范围从组合分割区 2 开始并且在组合分割区 3 结束。

相关概念:

- 第 106 页的『数据库管理空间』
- 『Snapshot monitor』（*System Monitor Guide and Reference*）
- 第 111 页的『在 DMS 表空间中如何添加和扩展容器』
- 第 121 页的『如何在 DMS 表空间中删除和缩小容器』

相关参考:

- 『GET SNAPSHOT Command』（*Command Reference*）

在 DMS 表空间中如何添加和扩展容器

创建表空间时，会创建其表空间映射并对齐所有初始容器，以使它们都从组合分割区 0 开始。这意味着数据将均匀分布在所有表空间容器上，直到个别容器已满。（参见第 112 页的示例 1。）

ALTER TABLESPACE 语句允许向现有表空间添加容器或扩展容器，以增加其存储容量。

添加比现有容器小的容器会导致数据分布不均匀。这可能导致并行 I/O 操作，如预取数据，它们的执行效率比对大小相同的容器执行的效率要低。

向表空间添加新容器或扩展现有容器时，*可能*发生表空间数据的再平衡。

再平衡

添加或扩展容器时的同志平衡的过程涉及将表空间数据块从一个位置移动到另一位置，这是通过试图保持数据在表空间内进行分割来完成的。

在再平衡期间不会限制对表空间的存取；可以象平常一样删除、创建和填充对象。但是，再平衡操作可能对性能有很大的影响。如果需要添加多个容器，并且计划再平衡容器，应在单个 ALTER TABLESPACE 语句中同时添加它们，以免数据库管理器不得不多次再平衡数据。

表空间上限标记在再平衡过程中起着关键作用。上限标记是表空间中的最高分配页的页号。例如，表空间有 1000 页，数据块大小为 10，则结果为 100 个数据块。如果第 42 个数据块是表空间中最高分配的数据块，则意味着上限标记是 42 * 10 = 420 页。这与已使用的页不同，因为可能已经释放了上限标记下的一些数据块以使它们可供重新使用。

在再平衡启动前，会根据所作的容器更改构建新的表空间映射。再平衡程序将把数据块从由当前映射确定的位置移至由新映射确定的位置。再平衡程序从数据块 0

开始，一次移动一个数据块，直到移动了持有上限标记的数据块为止。移动每个数据块时，当前映射每次改变一块以使其看起来与新映射相似。再平衡完成时，对于当前映射和新映射，一直到持有上限标记的组合分割区，看起来应完全相同。于是使当前映射与新映射看起来完全相同，再平衡过程就完成了。如果数据块在当前映射中的位置与它在新映射中的位置相同，则不移动该数据块，并且不发生 I/O。

当添加新容器时，该容器在新映射内的位置取决于其大小及其分割集中其它容器的大小。若容器足够大，以至于它可以从分割集中的第一个组合分割区开始，并在分割集中的最后一个组合分割区处（或以外）结束，则将它使用该方法放置（参见第 113 页的示例 2）。若容器不够大，无法做到这一点，则它将在映射中定位为在分割集的最后一个组合分割区结束（参见第 116 页的示例 4。）这样做是为了最小化需要再平衡的数据量。

注: 在下列示例中，容器大小未将容器标记计算在内。容器大小很小，仅用于说明目的，它们不是建议的容器大小。这些示例显示表空间中不同大小的容器，但建议使用相同大小的容器。

示例 1:

如果您创建的表空间有三个容器，数据块大小为 10，并且容器分别为60、40 和 80 页（6、4 和 8 个数据块），则将创建带有映射的表空间，可进行如第 113 页的图 39 中所示的图解。

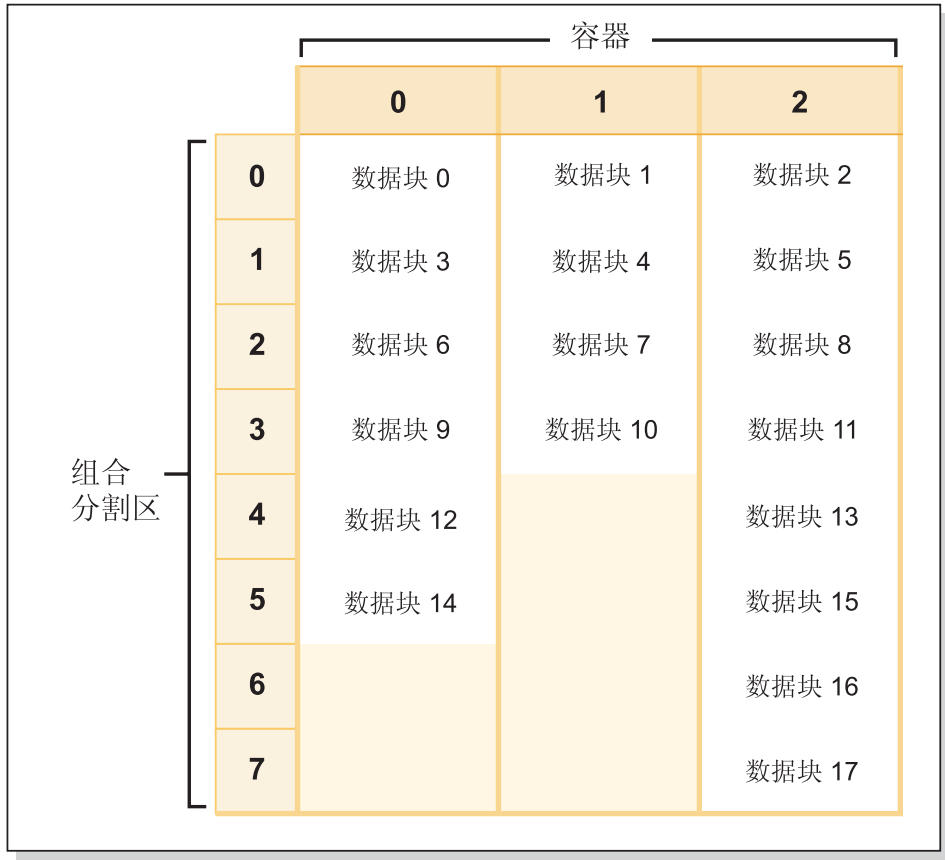

图 39.

相应的表空间映射，如表空间快照中所示，类似如下:

```
范围    分割    组合分割区    最大      最大    起始          结束          调节    容器
编号    集      偏移          数据块    页      组合分割区    组合分割区
 [0]     [0]             0       11     119             0             3     0   3 (1 和 2)
 [1]     [0]             0       15     159             4             5     0   2 (0 和 2)
 [2]     [0]             0       17     179             6             7     0   1 (2)
```

表空间映射中的标题是“范围编号”、“分割集”、“组合分割区偏移”、“根据范围寻址的最大数据块编号”、“根据范围寻址的最大页编号”、“起始组合分割区”、“结束组合分割区”、“范围调节”和“容器列表”。

示例 2:

如果在示例 1 中向表空间添加了一个 80 页的容器，容器就大到足以从第一个组合分割区（组合分割区 0）中开始，并在最后一个组合分割区（组合分割区 7）中结束。它被定位为从第一个组合分割区中开始。结果表空间可进行如图 40 中所示的图解。

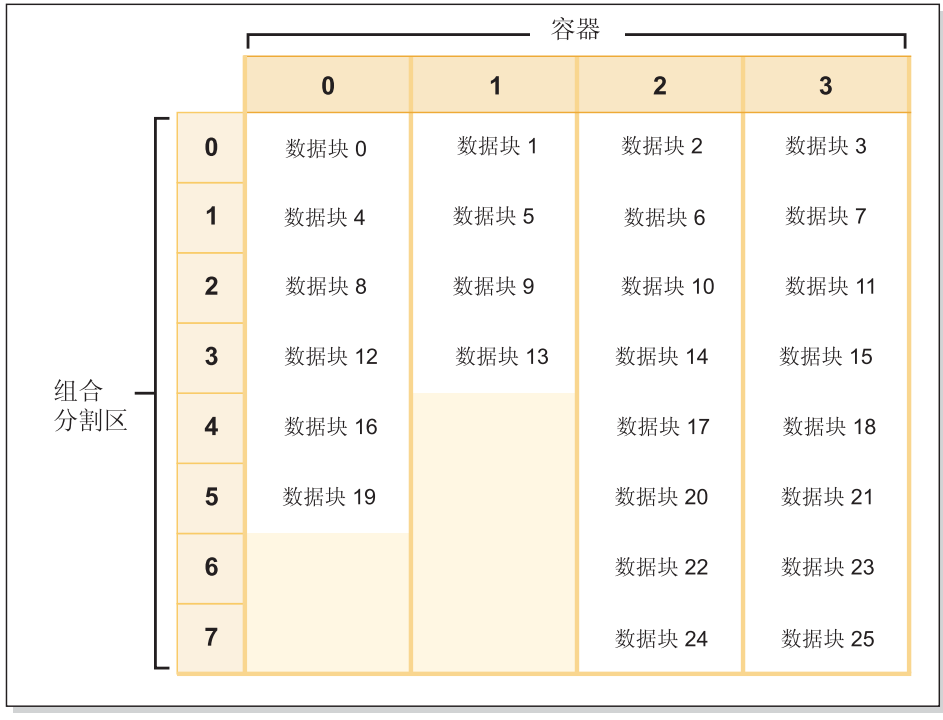

图 40.

相应的表空间映射，如表空间快照中所示，类似如下:

```
范围    分割    组合分割区    最大      最大    起始          结束          调节    容器
编号    集      偏移          数据块    页      组合分割区    组合分割区
 [0]    [0]           0        15    159             0             3       0    4 (0、1、2 和 3)
 [1]    [0]           0        21    219             4             5       0    3 (0、2 和 3)
 [2]    [0]           0        25    259             6             7       0    2 (2 和 3)
```

如果上限在标记在数据块 14 以内，则再平衡程序将从数据块 0 开始，并且将把所有数据块上移至 14（包括 14）。两个映射内的数据块 0 的位置相同，所以不必移动此数据块。数据块 1 和 2 的情况相同。需要移动数据块 3，所以从旧位置（容器 0 内的第二个数据块）读取该数据块并写至新位置（容器 3 内的第一个数

据块）。将移动此数据块后到数据块 14（包括数据块 14）的每个数据块。一旦移动了数据块 14，当前映射看起来会象新映射，并且再平衡程序将终止。

如果改变映射以使所有新添加的空间都在上限之后，则不需要再平衡并且所有的空间都立即可用。如果改变映射以使部分空间在上限标记之后，则组合分割区中在上限标记之上的空间将是可用的。余下部分直到再平衡完成才可用。

如果决定扩展容器，则再平衡程序的功能相似。如果扩展容器以使它超出了分割集中的最后一个分割，则将扩展该分割集以适应这种情况并将相应地移出其后的分割集。结果是容器不会扩展到其后的任何分割集中。

示例 3:

考虑示例 1 中的表空间。如果将容器 1 从 40 页扩展到 80 页，则新表空间将类似第 116 页的图 41。

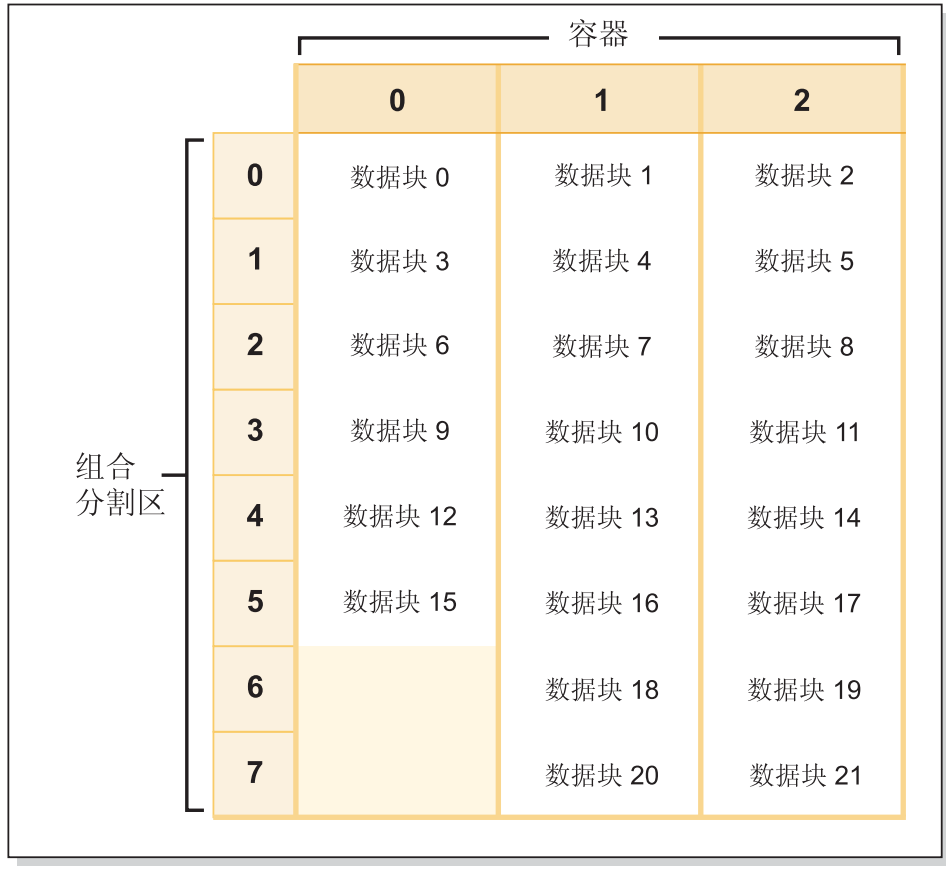

图 41.

相应的表空间映射，如表空间快照中所示，类似如下:

```
范围    分割    组合分割区    最大      最大    起始          结束          调节    容器
编号    集      偏移          数据块    页      组合分割区    组合分割区
 [0]     [0]             0        17    179             0             5       0    3（0、1 和 2）
 [1]     [0]             0        21    219             6             7       0    2（1 和 2）
```

示例 4:

考虑第 112 页的示例 1 中的表空间。若向它添加一个 50 页（5 个数据块）的容器，则将以如下方式将该容器添加至新映射。容器大小不足以从第一个组合分割

区（组合分割区 0）中开始，并在最后一个组合分割区（组合分割区 7）处或以外结束，因此将它定位为在最后一个组合分割区中结束。（参见图 42。）

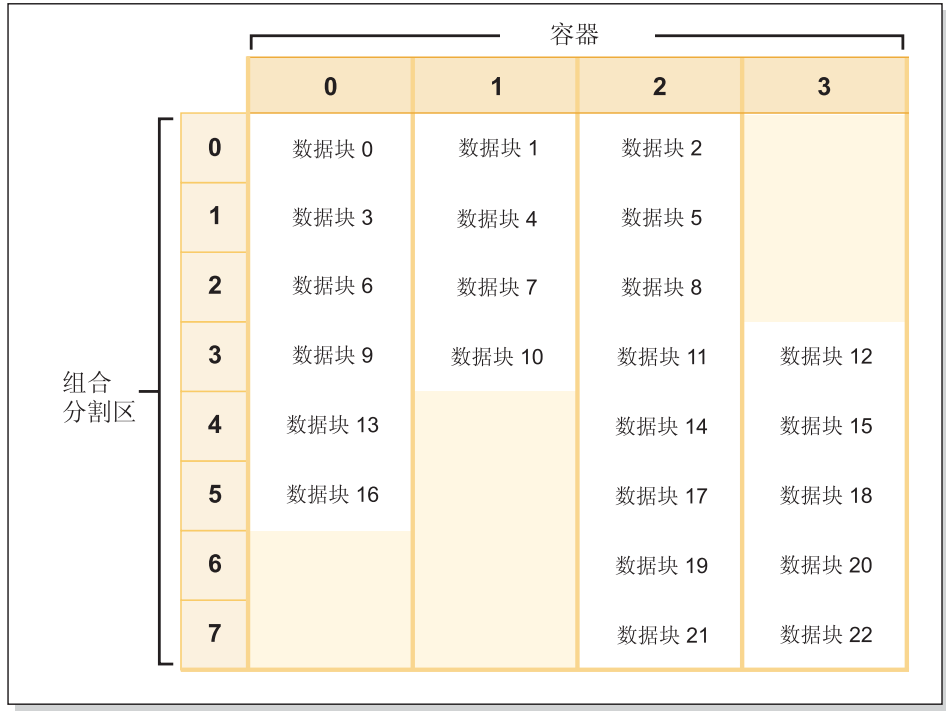

图 42.

相应的表空间映射，如表空间快照中所示，类似如下:

```
范围    分割    组合分割区    最大      最大    起始          结束          调节    容器
编号    集      偏移          数据块    页      组合分割区    组合分割区
[0]     [0]           0        8     89             0             2    0   3（0、1 和 2）
[1]     [0]           0       12    129             3             3    0   4（0、1、2 和 3）
[2]     [0]           0       18    189             4             5    0   3（0、2 和 3）
[3]     [0]           0       22    229             6             7    0   2（2 和 3）
```

要扩展容器，在 ALTER TABLESPACE 语句上使用 EXTEND 或 RESIZE 选项。要添加容器并再平衡数据，在 ALTER TABLESPACE 语句上使用 ADD 选项。如果正在向已经有多个分割集的表空间添加容器，则可以指定想要向哪个分割集添加容器。为此，在 ALTER TABLESPACE 语句上使用 ADD TO STRIPE SET 选项。如果不指定分割集，则缺省行为将是向当前分割集添加容器。当前分割集是最新创建的分割集，而不是最后向其添加空间的分割集。

对分割集的任何更改可能导致对该分割集及其后的任何其它分割集的再平衡。

可以通过使用表空间快照来监视再平衡的进度。表空间快照可以提供关于再平衡的信息，如再平衡的开始时间、已经移动了多少个数据块以及需要移动多少个数据块。

没有再平衡（使用分割集）

如果添加或扩展容器，并且添加的空间在表空间上限标记之上，则不会发生再平衡。

添加容器将总是在上限标记下添加空间。换句话说，添加容器时，再平衡通常是必要的。有一个选项可强制将新容器添加到上限标记之上，它允许您选择不对表空间的内容再平衡。此方法的一个优点是新容器可立即使用。不进行再平衡这一选项仅在添加容器时才适用，在扩展现有容器时不适用。扩展容器时，仅当添加的空间在上限标记之上时，才能避免再平衡。例如，如果有许多大小相同的容器，并且按相同的量扩展它们，则数据块的相对位置不会更改，并且不会发生再平衡。

通过添加新的*分割集*来添加容器而不进行再平衡。分割集是表空间中的一组容器，数据在其上进行分割，且独立于属于该表空间的其它容器。想要向表空间添加容器而不对数据进行再平衡时，则使用新分割集。现有分割集中的现有容器保持不变，并且正在添加的容器成为新分割集的一部分。

要添加容器而不进行再平衡，在 ALTER TABLESPACE 语句上使用 BEGIN NEW STRIPE SET 选项。

示例 5:

如果表空间有三个容器，数据块大小为 10，并且容器分别为 30、40 和 40 页（分别为 3、4 和 4 个数据块），则表空间可进行如第 119 页的图 43 中所示的图解。

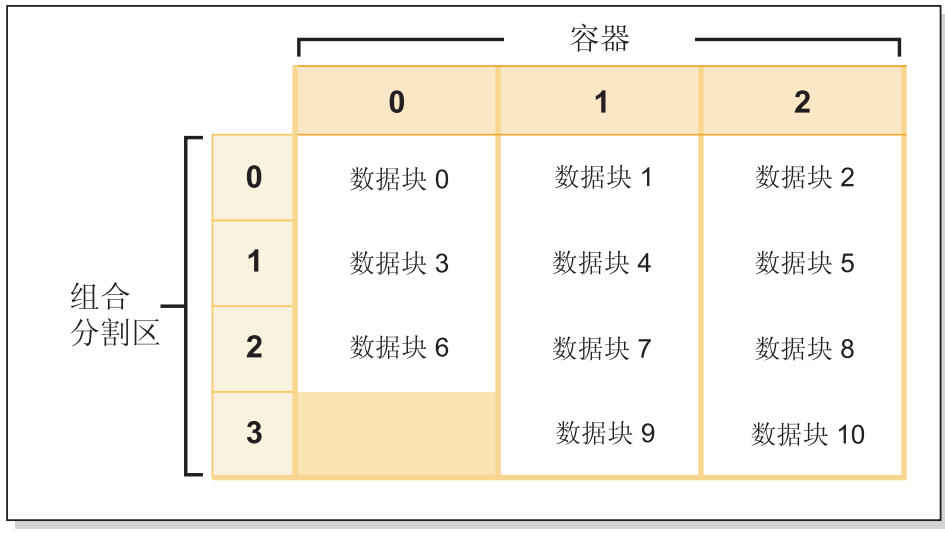

图 43.

相应的表空间映射，如表空间快照中所示，类似如下:

```
范围     分割     组合分割区     最大     最大     起始          结束          调节     容器
编号     集       偏移           数据块   页       组合分割区    组合分割区
 [0]      [0]             0         8    89              0             2        0   3（0、1 和 2）
 [1]      [0]             0        10   109              3             3        0   2（1 和 2）
```

示例 6:

在使用 BEGIN NEW STRIPE SET 选项添加 30 页和 40 页的两个新容器（分别为 3 和 4 个数据块）时，不会影响现有范围而是将创建一组新范围。这一组新范围是一个分割集，而最新创建的分割集称为当前分割集。添加了两个新的容器之后，表空间将类似第 120 页的图 44。

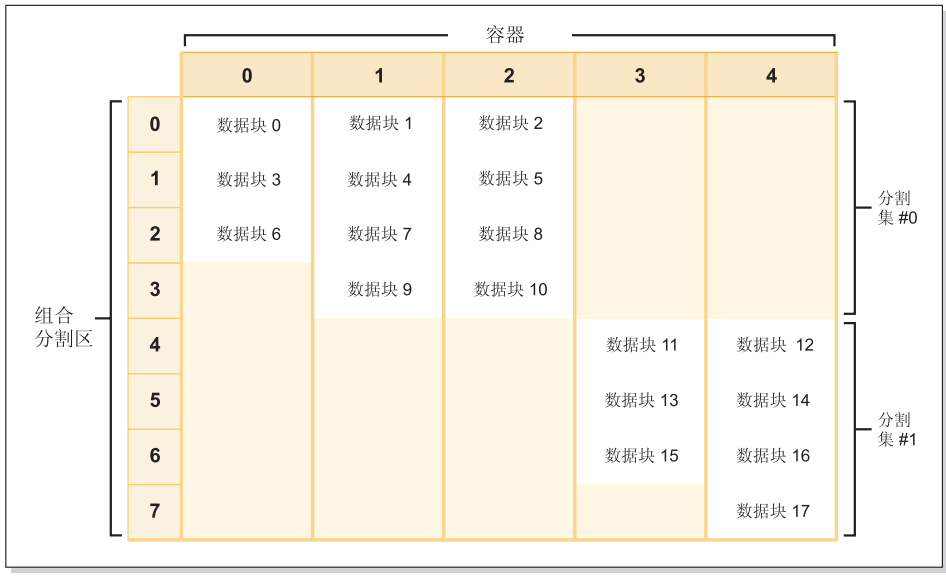

图 44.

相应的表空间映射，如表空间快照中所示，类似如下:

```
范围    分割    组合分割区    最大      最大    起始          结束          调节    容器
编号    集      偏移          数据块    页      组合分割区    组合分割区
 [0]     [0]            0        8     89              0             2       0    3（0、1 和 2）
 [1]     [0]            0       10    109              3             3       0    2（1 和 2）
 [2]     [1]            4       16    169              4             6       0    2（3 和 4）
 [3]     [1]            4       17    179              7             7       0    1（4）
```

如果向表空间添加新的容器，并且未将 TO STRIPE SET 选项与 ADD 子句配合使用，则将把容器添加至当前分割集（最高分割集）。可以使用 ADD TO STRIPE SET 子句来将容器添加到表空间中的任何分割集。必须指定有效的分割集。

DB2® 使用表空间映射跟踪分割集，并且添加新容器而不进行再平衡通常将导致映射比再平衡容器时增长得快。当表空间映射变得过大时，如果试图添加更多容器，将接收到错误 SQL0259N。

相关概念:

- 第 107 页的『表空间映射』

相关任务:

- 『将容器添加至 DMS 表空间』（*《管理指南: 实现》*）

- 『修改 DMS 表空间中的容器』（*《管理指南: 实现》*）

相关参考:

- 『ALTER TABLESPACE statement』（*SQL Reference, Volume 2*）
- 『GET SNAPSHOT Command』（*Command Reference*）
- 『tabel space activity data elements』（*System Monitor Guide and Reference*）

如何在 DMS 表空间中删除和缩小容器

对于 DMS 表空间，可以从表空间中删除容器或缩小容器的大小。使用 ALTER TABLESPACE 语句来完成此操作。

仅当该操作删除的数据块的数目小于或等于表空间中的上限标记之上的可用数据块的数目时，才允许删除或缩小容器。这是必须的，因为该操作不能更改页号，从而直到上限标记（包括上限标记）的所有数据块在表空间内必须处于相同的逻辑位置。因此，结果表空间必须具有足够的空间才能存放直到上限标记（包括上限标记）的所有数据。在没有足够的可用空间的情况下，执行语句时会立即接收到错误。

上限标记是表空间中的最高分配页的页号。例如，表空间有 1000 页，数据块大小为 10，则结果为 100 个数据块。如果第 42 个数据块是表空间中最高分配的数据块，则意味着上限标记是 42 * 10 = 420 页。这与已使用的页不同，因为可能已经释放了上限标记下的一些数据块以使它们可供重新使用。

删除或缩小容器时，如果数据驻留在正从表空间中删除的空间中，则将进行再平衡。在再平衡启动之前，会根据所作的容器更改构建新的表空间映射。再平衡程序将把数据块从由当前映射确定的位置移至由新映射确定的位置。再平衡程序从包含上限标记的数据块开始，一次移动一个数据块，直到移动了数据块 0 为止。移动每个数据块时，当前映射每次改变一块以使其看起来与新映射相似。如果数据块在当前映射中的位置与它在新映射中的位置相同，则不移动该数据块，并且不发生 I/O。因为再平衡从表空间中分配的最高数据块开始移动，并以表空间中的第一个数据块结束，所以又称为*逆向再平衡*（与在添加或扩展容器之后向表空间添加空间时发生的*正向再平衡*相反）。

删除容器时，余下的容器将重新编号，它们的容器标识从 0 开始每次加 1。如果删除了分割集中的所有容器，则将从映射除去该分割集，并且会下移映射中其后的所有分割集并对其重新编号，以便分割区集号码中没有间隔。

注: 在下列示例中，容器大小未将容器标记计算在内。容器大小很小，仅用于说明目的，它们不是建议的容器大小。这些示例显示表空间中不同大小的容器，但是这只是用于说明目的；建议使用相同大小的容器。

例如，考虑这样一个表空间，它有三个容器，数据块大小为 10。容器分别为 20、50 和 50 页（2、5 和 5 个数据块）。表空间的图表显示在图 45 中。

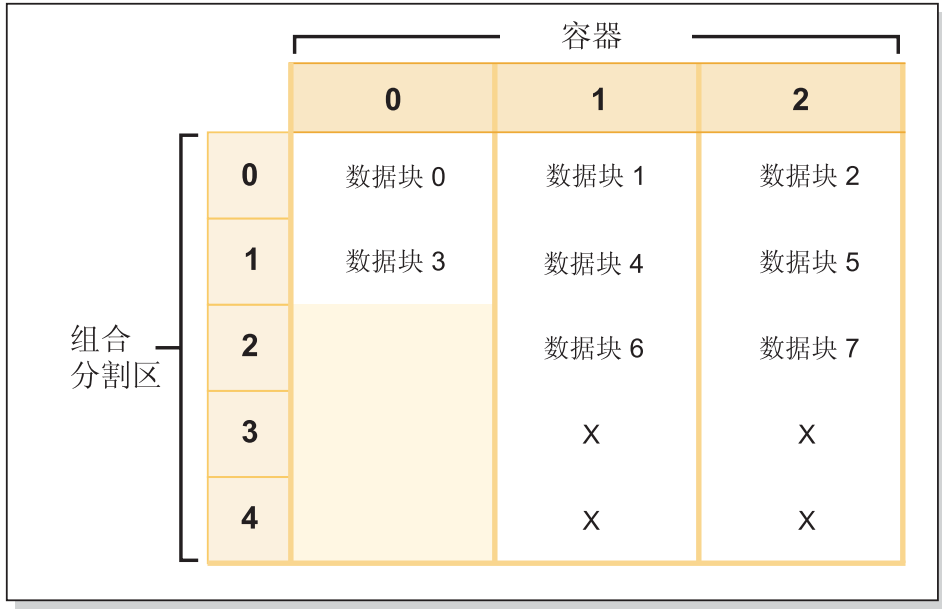

图 45.

X 指示存在数据块，但其中没有数据。

如果想要删除有两个数据块的容器 0，则上限标记之上必须至少具有两个可用数据块。上限标记在数据块 7 中，有四个可用数据块，因此可以删除容器 0。

相应的表空间映射，如表空间快照中所示，类似如下:

```
范围    分割    组合分割区    最大     最大     起始          结束          调节    容器
编号    集      偏移          数据块   页       组合分割区    组合分割区
[0]     [0]           0        5     59              0             1     0    3（0、1 和 2）
[1]     [0]           0       11    119              2             4     0    2（1 和 2）
```

删除之后，表空间将仅有容器 0 和容器 1。新的表空间图表显示在第 123 页的图 46 中。

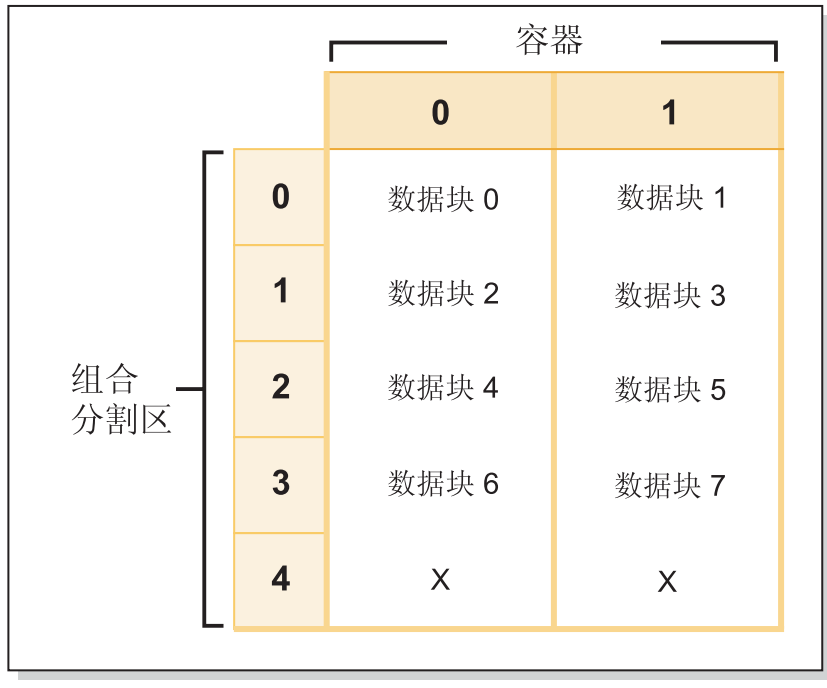

图 46.

相应的表空间映射，如表空间快照中所示，类似如下:

```
范围     分割     组合分割区     最大     最大     起始          结束          调节    容器
编号     集       偏移           数据块   页       组合分割区    组合分割区
[0]      [0]              0         9     99              0             4     0    2（0 和 1）
```

如果想要缩小容器，再平衡程序以相似的方式工作。

要缩小容器，在 ALTER TABLESPACE 语句上使用 REDUCE 或 RESIZE 选项。要删除容器，在 ALTER TABLESPACE 语句上使用 DROP 选项。

相关概念:

- 第 107 页的『表空间映射』

相关任务:

- 『修改 DMS 表空间中的容器』（*《管理指南: 实现》*）

相关参考:

- 『ALTER TABLESPACE statement』（*SQL Reference, Volume 2*）

- 『GET SNAPSHOT Command』（*Command Reference*）
- 『tabel space activity data elements』（*System Monitor Guide and Reference*）

SMS 和 DMS 表空间的比较

当确定应使用哪种类型的表空间存储数据时，有一些问题需要权衡一下。

SMS 表空间的优点:

- 直到需要时，系统才分配空间。
- 由于不必预定义容器，所以创建表空间需要的初始工作较少。

DMS 表空间的优点:

- 通过使用 ALTER TABLESPACE 语句，可添加或扩展容器来增加表空间的大小。现有数据可以自动在新的容器集合中再平衡，以保持最优 I/O 效率。
- 根据存储的数据的类型，一个表可以分布在多个表空间中:
  - 长型字段和 LOB 数据
  - 索引
  - 规则表数据

  可能出于性能方面的考虑而想要分隔表数据，或想要增加存储的表数据量。例如，您可能拥有具有 64GB 规则表数据、64GB 索引数据和 2TB 长型数据的一个表。若使用的是 8 KB 页，则表数据和索引数据可达 128 GB。若使用的是 16 KB 页，则表数据和索引数据可达 256 GB。若使用的是 32 KB 页，则表数据和索引数据可达 512 GB。
- 可控制数据在磁盘上的位置（若操作系统允许的话）。
- 若所有的表数据都位于单个表空间中，可以使用比删除和重新定义一个表更少的开销来删除和重新定义一个表空间。
- 通常，精心调整的一组 DMS 表空间的性能将优于 SMS 表空间。

注: 在 Solaris 上，强烈建议对性能非常关键的工作负荷使用具有原始设备的 DMS 表空间。

通常，小型个人数据库用 SMS 表空间管理最容易。另一方面，对于大的、增长中的数据库，您可能只希望使用 SMS 表空间用作临时表空间和目录表空间，而使用具有多个容器的单独的 DMS 表空间用于每个表。另外，您可能想将长型字段数据和索引存储在它们自己的表空间中。

若选择使用带设备容器的 DMS 表空间，您必须乐意调整和管理您的环境。

相关概念:

- 第 100 页的『表空间设计』
- 第 104 页的『系统管理空间』
- 第 106 页的『数据库管理空间』

表空间磁盘 I/O

表空间的类型和设计决定了对该表空间执行的 I/O 的效率。在进一步考虑关于表空间的设计和使用的问题之前，您应了解下列概念。

大块读取 在单个请求中检索多页（通常为一个数据块）的一种读取。一次读取几页比分别读取每页更有效。

预取 在一个查询引用页之前对那些页的读取。总的目的是为缩短响应时间。若页的预取可以与查询的执行异步发生，就能够达到此目的。当一个或多个 CPU 或 I/O 子系统以最大能力运行时达到最佳响应时间。

页清除 当读取和修改页时，它们会累积在数据库缓冲池中。当读入一页时，便将其读入到缓冲池页中。若该缓冲池已充满修改的页，则必须将这些修改的页之一写出至磁盘，然后才能再读入新的页。为避免缓冲池变满，页清除程序代理进程的任务就是写出修改的页，以保证缓冲池页可用于将来的读取请求。

无论何时，只要是有利，DB2® 就会执行大块读取。当检索顺序排列或本质上是部分顺序排列的数据时，通常会发生这种情况。在一个读取操作中读取的数据量取决于数据块大小 — 数据块大小越大，一次可以读取的页就越多。

如果可以将页从磁盘读入缓冲池内的连续页中，则可以进一步提高顺序预取的性能。因为缺省情况下缓冲池是基于页的，所以从磁盘的连续页中读取时，不能保证会找到一组连续页。基于块的缓冲池可以用于此目的，因为它们不仅包含页区域，还包含几组连续页的块区域。每一组连续页都命名为一个块，并且每个块都包含称为块大小的若干页。页和块区域的大小以及每个块中的页的数目都是可配置的。

数据块存储在磁盘上的方式影响 I/O 效率。在使用设备容器的 DMS 表空间中，数据往往在磁盘上是连续的，且可以在最短的搜索时间和磁盘延迟内进行读取。但是，若使用的是文件，则数据可能被文件系统分散，并存储在磁盘上的多个位置中。当使用一次将文件扩展一页的 SMS 表空间时（这使产生碎片的概率更高），最可能发生此情况。为了供 DMS 表空间使用而预分配的大文件往往在磁盘上是连续的，尤其当该文件分配在一个干净的文件空间中时更是这样。

可以通过更改 CREATE TABLESPACE 或 ALTER TABLESPACE 语句上的 PREFETCHSIZE 选项来更改预取的程度。（该数据库中所有表空间的缺省值由 *dft_prefetch_sz* 数据库配置参数设置。）PREFETCHSIZE 参数告诉 DB2 在触发一个预取时要读取的页数。通过在 CREATE TABLESPACE 语句上将 PREFETCHSIZE 设置为 EXTENTSIZE 参数的倍数，可以并行读取多个数据块。（该数据库中所有表空间的缺省值由 *dft_extent_sz* 数据库配置参数设置。）EXTENTSIZE 参数指定在跳至下一个容器之前将写入一个容器的 4 KB 大小的页数。

例如，假定有一个表空间使用三个设备。若将 PREFETCHSIZE 设置为 EXTENTSIZE 的三倍，则 DB2 可以用并行方式从每个设备中执行大块读取，从而显著增加 I/O 吞吐量。此情况假定每个设备是一个单独的物理设备，且控制器具有足够的带宽来处理来自每个设备的数据流。注意，DB2 可能需要根据查询速度、缓冲池的利用情况和其它因素在运行时动态调整预取参数。

某些文件系统使用它们自己的预取方法（如，AIX 上的“日志文件系统”）。在某些情况中，文件系统的预取设置得比 DB2 的预取更主动。这可能导致使用文件容器的 SMS 和 DMS 表空间的预取似乎比使用设备的 DMS 表空间的预取执行效率更高。但这是误导，因为它可能是在文件系统中发生的附加级别的预取的结果。DMS 表空间应该能够比任何等效配置的执行效率高。

为提高预取效率（甚至读取效率），必须存在足够数量的干净缓冲池页。例如，可能有一个并行预取请求要从一个表空间读取三个数据块，对于正在读取的每一页，从缓冲池写出经过修改的一页。该预取请求可能被拖慢，导致它跟不上查询的进展。应配置足够数量的页清除程序，以满足预取请求。

相关概念:

- 第 100 页的『表空间设计』
- 『将数据预取至缓冲池』（*《管理指南: 性能》*）

相关参考:

- 『ALTER TABLESPACE statement』（*SQL Reference, Volume 2*）
- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）

表空间设计中的工作负荷注意事项

在您的环境中由 DB2® 管理的主要工作负荷类型，会影响您选择要使用的表空间类型以及要指定的页大小。联机事务处理（OLTP）工作负荷的特征是：事务需要对数据进行随机存取，通常涉及频繁插入或更新活动和通常返回一小组数据的查询。假定存取是随机的，并且是存取一页或几页，则不太可能发生预取。

使用设备容器的 DMS 表空间在这种情况下表现得最好。若不需要最大性能，使用文件容器的 DMS 表空间或 SMS 表空间也适合用于 OLTP 工作负荷。若期望很少的顺序 I/O 或不期望它，则 CREATE TABLESPACE 语句上的 EXTENTSIZE 和 PREFETCHSIZE 参数的设置对于 I/O 的效率就显得不重要。但是，使用 *chngpgs_thresh* 配置参数设置足够数目的页清除程序是很重要的。

查询工作负荷的特征是，事务需要对数据进行顺序存取或部分顺序存取，并常常返回大的数据集。使用多个设备容器（每个容器都在单独的磁盘上）的 DMS 表空间，最有可能提供有效的并行预取。应该将 CREATE TABLESPACE 语句上的 PREFETCHSIZE 参数的值设置为 EXTENTSIZE 参数的值乘以设备容器数之积。这允许 DB2 以并行方式从所有的容器中预取。如果容器的数目更改，或需要使预取更多或更少，则可以使用 ALTER TABLESPACE 语句相应地更改 PREFETCHSIZE 值。

查询工作负荷的一个合理的替代方法是使用文件（若文件系统有自己的预取的话）。这些文件可以是使用文件容器的 DMS 类型或 SMS 类型。注意，若使用 SMS，则需要将目录容器映射至单独的物理磁盘，以实现 I/O 并行性。

混合工作负荷的目标是：对于 OLTP 工作负荷，使单个 I/O 请求尽可能有效；而对于查询工作负荷，最大程度地提高并行 I/O 的效率。

确定表空间页大小的注意事项如下所示:

- 对于执行随机行读写操作的 OLTP 应用程序，通常优先考虑小一点的页大小，因为它为存储不需要的行而浪费的缓冲池空间较少。
- 对于一次存取大量连续行的决策支持系统（DSS）应用程序，页大小大一些会比较好，因为它将减少读取特定数目的行所需的 I/O 请求数。但是，也有例外。若行大小小于:

  ```
  页大小 / 255
  ```

  则每页上都会有浪费的空间（每页最多有 255 行）。在这种情况下，更小一点的页大小可能更合适。
- 更大一点的页大小可允许减少索引中的级别数。
- 越大的页，支持的行越长。
- 在缺省的 4 KB 页上，一个表只能有 500 列，而更大的页大小（8 KB、16 KB 和 32 KB）支持 1012 列。
- 表空间的最大大小与表空间的页大小成正比。

相关概念:

- 第 104 页的『系统管理空间』
- 第 106 页的『数据库管理空间』

相关参考:

- 『 “已更改页阈值”配置参数 — chngpgs_thresh 』（ *《管理指南: 性能》* ）
- 『 ALTER TABLESPACE statement 』（ *SQL Reference, Volume 2* ）
- 『 CREATE TABLESPACE statement 』（ *SQL Reference, Volume 2* ）
- 『 SQL limits 』（ *SQL Reference, Volume 1* ）

数据块大小

表空间的数据块大小表示在将数据写入下一个容器之前，将写入当前容器的表数据的页数。当选择数据块大小时，应考虑:

- 表空间中表的大小和类型。

  将 DMS 表空间中的空间一次分配给表一个数据块。当填充该表而一个数据块变满时，会分配新的数据块。如果启用多页文件分配，则一次分配 SMS 表空间中的空间一个数据块。

  一个表由下列单独的表对象组成:

  – 数据对象。它是存储规则列数据的地方。
  – 索引对象。在表上定义的所有索引都存储在这里。
  – 长型字段对象。若表有一个或多个 LONG 列，则长型字段数据存储在此处。
  – 两个 LOB 对象。若表有一个或多个 LOB 列，则它们都存储在这两个表对象中:
    - 一个表对象用于存储 LOB 数据
    - 第二个表对象用于存储描述 LOB 数据的元数据
  – 多维表的块映射对象。

  每个表对象都是单独存储的，每个对象按需要分配新的数据块。每个表对象还与称为*数据块映射*的元数据对象配成一对，该元数据对象描述该表空间中属于该表对象的所有数据块。用于数据块映射的空间也是以一次一个数据块的方式分配。

  因此，一个表的空间的初始分配是每个表对象两个数据块。若您在一个表空间中有多个小表，您可能要分配相对大的空间来存储相对少量的数据。在这种情况下，应该指定小的数据块大小或使用一次分配一页的 SMS 表空间。

  另一方面，若您有一个增长速率高的非常大的表，且您使用具有较小数据块大小的 DMS 表空间，您可能会有与附加数据块的频繁分配相关的不需要的开销。
- 对这些表存取的类型。

若对表的存取包括许多查询或处理大量数据的事务，则从表中预取数据可以显著改善性能。

- 必需的最小数据块数。

  若容器中没有足够的空间以供表空间的五个数据块使用，则将无法创建表空间。

相关概念:

- 第 100 页的『表空间设计』

相关参考:

- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）
- 『db2empfa - Enable Multipage File Allocation Command』（*Command Reference*）

表空间和缓冲池之间的关系

每个表空间都与一个特定的缓冲池相关。缺省缓冲池是 IBMDEFAULTBP。若另一个缓冲池要与表空间相关，则该缓冲池必须存在（用 CREATE BUFFERPOOL 语句定义它），它必须具有相同的页大小，且在创建该表空间（使用 CREATE TABLESPACE 语句）时定义关联。表空间与缓冲池之间的关联可以使用 ALTER TABLESPACE 语句来更改。

若拥有多个缓冲池，则可以配置数据库使用的内存，以改善整体性能。例如，如果具有带有一个或多个用户可随机存取的大（大于可用内存）表的表空间，缓冲池的大小可能受到限制，因为高速缓存该数据页可能没有好处。用于联机事务应用程序的表空间可以与一个较大的缓冲池相关联，以便可以更长时间地高速缓存应用程序所使用的数据页，导致响应时间更快。配置新缓冲池时必须小心。

注: 若确定数据库需要 8 KB、16 KB 或 32 KB 的页大小，必须将使用其中一种页大小的每个表空间映射至具有相同页大小的缓冲池。

当启动数据库时，所有缓冲池必需的存储器对于数据库管理器必须是可用的。若 DB2® 无法获取必需的存储器，则数据库管理器将使用缺省缓冲池（具有 4 KB、8 KB、16 KB 和 32 KB 页大小的缓冲池各一个）启动，并发出警告。

在分区数据库环境中，可以为该数据库中的所有分区创建大小相同的一个缓冲池。也可以在不同的分区上创建不同大小的缓冲池。

相关概念:

- 『Table spaces and other storage structures』（*SQL Reference, Volume 1*）

相关参考:

- 『ALTER BUFFERPOOL statement』（*SQL Reference, Volume 2*）
- 『ALTER TABLESPACE statement』（*SQL Reference, Volume 2*）
- 『CREATE BUFFERPOOL statement』（*SQL Reference, Volume 2*）
- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）

表空间和数据库分区组之间的关系

在分区数据库环境中，每个表空间都与特定数据库分区组相关。这允许将表空间的特征应用于该数据库分区组中的每个分区。该数据库分区组必须存在（用 CREATE DATABASE PARTITION GROUP 语句定义它），且当使用 CREATE TABLESPACE 语句创建表空间时定义表空间与该数据库分区组之间的关联。

不能使用 ALTER TABLESPACE 语句来更改表空间与数据库分区组之间的关联。只能为数据库分区组内的个别分区更改表空间规范。在单分区环境中，每个表空间都与缺省数据库分区组相关。在定义表空间时，缺省数据库分区组为 IBMDEFAULTGROUP，除非在定义系统临时表空间，那时将使用 IBMTEMPGROUP。

相关概念:

- 『Table spaces and other storage structures』（*SQL Reference, Volume 1*）
- 第 92 页的『数据库分区组』
- 第 100 页的『表空间设计』

相关参考:

- 『CREATE DATABASE PARTITION GROUP statement』（*SQL Reference, Volume 2*）
- 『CREATE TABLESPACE statement』（*SQL Reference, Volume 2*）

临时表空间设计

建议定义一个 SMS 临时表空间，使它的页大小等于大多数规则表空间所使用的页大小。这应该适用于典型的环境和工作负荷。但最好用不同的临时表空间配置和工作负荷进行实验。应该考虑下列几点:

- 在大多数情况下，临时表空间是按顺序成批存取的。也就是说，插入一组行或顺序读取一组行。因此，当需要更少的逻辑或物理页 I/O 请求来读取给定的数据量时，页大小越大，通常所获得的性能越优。当临时表的平均行大小小于页大小除以 255 的值时，情况就不会总是这样。无论是哪种页大小，任何一页上最多可有 255 行。例如，若一个查询需要 15 字节一行的临时表，则用 4 KB 页

大小的临时表空间更合适，因为 255 行可以全部包含在一个 4 KB 页中。8 KB（或更大的）页大小将在临时表的每一页上浪费至少 4 KB（或更多）字节的空间，最好不要减少需要的 I/O 请求数。

- 若在数据库中有超过一半的规则表空间使用了相同的页大小，建议您定义具有相同页大小的临时表空间。这样做的原因是这种安排可以使临时表空间与大多数或全部的规则表空间共享同一个缓冲池空间。这样可简化缓冲池的调整。
- 当使用临时表空间重组表时，该临时表空间的页大小必须与该表的页大小匹配。由于这个原因，您应确保为现有表使用的每种不同的页大小定义了临时表空间，这样才可使用临时表空间重组这些表。

  也可不用临时表空间来重组，即直接在目标表空间中重组该表。当然，这种重组要求在目标表空间中有额外的空间来完成重组过程。
- 一般而言，当存在具有不同页大小的临时表空间时，优化器常常会选择具有最大缓冲池的临时表空间。在这种情况下，比较聪明的做法是给一个临时表空间分配一个足够大的缓冲池，而给其余临时表空间分配较小的缓冲池。这种缓冲池分配将有助于保证有效利用主内存。例如，若目录表空间使用 4 KB 页，而其余表空间使用 8 KB 页，则最佳临时表空间配置可能是：具有一个大缓冲池的一个 8 KB 临时表空间和一个具有小一些的缓冲池的一个 4 KB 表空间。

  注：目录表空间只能使用 4 KB 页大小。因此，数据库管理器始终强制保证 4 KB 系统临时表空间的存在，以允许目录表重组。
- 一般情况下，定义具有相同页大小的多个临时表空间没有什么好处。
- 对于临时表空间而言，SMS 几乎总是比 DMS 更合适，因为：
  - 使用 DMS 与使用 SMS 相比较而言，创建临时表需要更大的开销。
  - 在 SMS 中，磁盘空间是按需要分配的；而在 DMS 中必须预先分配它。预先分配可能比较困难：临时表空间保存着瞬时数据，这些数据在某个时候可能需要很大的存储器，但在正常情况下存储器需求却低得多。使用 DMS 时，必须预先分配存储器的峰值需求量；而使用 SMS 时，在高峰期间可以使用额外的磁盘空间来完成其它任务。
  - 数据库管理器尝试将临时表页保存在内存中，而不是将它们写出至磁盘。因此，DMS 的性能优点就没有那么突出。

相关概念:

- 第 100 页的『表空间设计』
- 第 104 页的『系统管理空间』

相关参考:

- 『REORG INDEXES/TABLE Command』（*Command Reference*）

目录表空间设计

建议使用 SMS 表空间来存储数据库目录，原因如下:

- 该数据库目录由许多大小不同的表组成。当使用 DMS 表空间时，对于每个表对象，分配最少两个数据块。根据选择的数据块大小，可能产生大量已分配而未使用的空间。当使用 DMS 表空间时，应选择小的数据块大小（二至四页）；否则，应使用 SMS 表空间。
- 目录表中存在大对象（LOB）列。LOB 数据不与其它数据一起保留在缓冲池中，而是每次需要时从磁盘中读取。从磁盘读取 LOB 降低了性能。因为文件系统通常有它自己的高速缓存，所以使用 SMS 表空间或在文件容器上构建的 DMS 表空间将避免可能的 I/O（若先前引用了该 LOB 的话）。

将这些因素考虑在内，SMS 表空间更适合用作目录。

另一个要考虑的因素是，将来是否需要扩大目录表空间。虽然某些平台支持扩大 SMS 容器的基本存储器，虽然可以使用重定向复原来扩大 SMS 表空间，但使用 DMS 表空间能简化添加新容器的工作。

相关概念:

- 『系统目录表的定义』（*《管理指南: 实现》*）
- 第 100 页的『表空间设计』
- 第 104 页的『系统管理空间』
- 第 106 页的『数据库管理空间』

当数据在 RAID 设备上时优化表空间性能

本节描述当将数据存放在“磁盘冗余阵列”（RAID）设备中时如何优化性能。

过程:

应对使用 RAID 设备的每个表空间执行下列操作:

- 为使用 RAID 设备的表空间定义一个容器。
- 使该表空间的 EXTENTSIZE 等于或数倍于 RAID 组合分割区大小。
- 确保该表空间的 PREFETCHSIZE 为:
  - RAID 组合分割区大小与 RAID 并行设备数之积（或此积的整数倍），并且是
  - EXTENTSIZE 的一个倍数。
- 使用 DB2_PARALLEL_IO 注册表变量对表空间启用并行 I/O。

DB2_PARALLEL_IO:

当对表空间容器读写数据时，若数据库中有多个容器，DB2 可以使用并行 I/O。但在某些情况下，对单个容器表空间启用并行 I/O 比较有利。例如，若该容器是在由多个物理磁盘组成的单个 RAID 设备上创建的，可发出并行读写调用。

要强制对只有一个容器的表空间执行并行 I/O，可使用 DB2_PARALLEL_IO 注册表变量。可将此变量设置为“*”（星号），表示每个表空间，或者可将它设置为由逗号分开的表空间标识的列表。例如:

```
db2set DB2_PARALLEL_IO=*        {对所有表空间打开并行 I/O}
db2set DB2_PARALLEL_IO=1,2,4,8  {对表空间 1、2、4 和表空间 8 打开并行 I/O}
```

在设置了注册表变量后，必须停止 DB2（**db2stop**），然后重新启动它（**db2start**），以使更改生效。

由于定义了多个容器，DB2_PARALLEL_IO 还会影响表空间。如果不设置注册表变量，则 I/O 并行性等于表空间中的容器的数目。如果设置注册表变量，则 I/O 并行性等于预取大小除以数据块大小的结果。如果表空间中的个别容器分布在多个物理磁盘上，则您可能想要设置注册表变量。

例如，表空间有两个容器，并且预取大小是数据块大小的四倍。如果没有设置注册表变量，则对此表空间的预取请求将分为两个请求（每个请求针对两个数据块）。如果预取程序可以工作，则两个预取程序可以并行处理这些请求。在设置注册表变量的情况下，对此表空间的预取请求将分为四个请求（每个请求针对一个数据块），且可以使用四个预取程序并行处理这些请求。

在此示例中，如果每一个容器都有专用的单个磁盘，则对此表空间设置注册表变量将导致对这些磁盘的争用，因为两个预取程序将立即存取每一个磁盘。但是，如果每个容器都分布在多个磁盘上，则设置注册表变量将潜在允许同时存取四个不同的磁盘。

DB2_USE_PAGE_CONTAINER_TAG:

缺省情况下，DB2 使用每个 DMS 容器（文件或设备）的第一个数据块来存储容器标记。容器标记是容器的 DB2 元数据。在较早版本的 DB2 中，使用第一页而不是第一个数据块来存储容器标记，并且因此该容器中用来存储该标记的空间较少。（在 DB2 的较早的版本中，DB2_STRIPED_CONTAINERS 注册表变量用来创建带有数据块大小标记的表空间。但是，因为这现在是缺省行为，所以此注册表变量不会再起作用。）

将 DB2_USE_PAGE_CONTAINER_TAG 注册表变量设置为 ON 时，创建的任何新 DMS 容器都带有一页标记，而不是一个数据块标记（缺省值）。这不会对在设置注册表变量前创建的现有容器产生任何影响。

除非有非常严格的空间约束或需要与版本 8 之前的数据块的行为保持一致，否则建议不要将此注册表变量设置为 ON。

如果 RAID 设备用于表空间容器，则将此注册表变量设置为 ON 可能对 I/O 性能有负面影响。当使用 RAID 设备作为表空间容器时，建议用等于或数倍于 RAID 组合分割区大小的数据块大小创建表空间。但是，如果将此注册表变量设置为 ON，则将使用一页容器标记并且数据块将不会与 RAID 组合分割区对齐。因此，在 I/O 请求期间可能需要存取比最优情况更多的物理磁盘。因此强烈建议用户不要设置此注册表变量。

要创建带有一页容器标记的容器，则将此注册表变量设置为 ON，然后停止并重新启动实例:

```
db2set DB2_USE_PAGE_CONTAINER_TAG=ON
db2stop
db2start
```

要停止创建带有一页容器标记的容器，复位此注册表变量，然后停止并重新启动实例。

```
db2set DB2_USE_PAGE_CONTAINER_TAG=
db2stop
db2start
```

“控制中心”、LIST TABLESPACE CONTAINERS 命令和 GET SNAPSHOT FOR TABLESPACES 命令不显示创建的容器具有一页标记还是数据块大小标记。它们使用标号“文件”或“设备”，这取决于该容器是如何创建的。要验证创建的容器是具有一页大小标记还是数据块大小标记，可以使用 DB2DART 的 /DTSF 选项来转储表空间和容器信息，然后查看要了解的容器的类型字段。查询容器 API（sqlbftcq 和 sqlbtcq）可以用来创建将显示类型的简单应用程序。

相关概念:

- 第 100 页的『表空间设计』

相关参考:

- 『系统环境变量』（*《管理指南: 性能》*）

为表选择表空间时的注意事项

当确定如何将表映射至表空间时，应考虑:

- 表的分区。

  至少应该确保选择的表空间位于具有您想要的分区的数据库分区组中。
- 表中的数据量。

若计划在一个表空间中存储许多小表，则考虑使用 SMS 充当该表空间。对于小表，DMS 表现在 I/O 和空间管理效率方面的优点就没有那么重要。一次分配一页空间的 SMS 的优点（且仅在需要时使用）却对小表更具吸引力。若一个表较大或者您需要更快地存取表中的数据，应考虑具有较小数据块大小的 DMS 表空间。

您可能希望对每个非常大的表都使用单独的表空间，而将所有的小表组合在单个表空间中。这种分隔还允许您根据表空间的使用选择适当的数据块大小。

- 表数据的类型。

  例如，有的表可能包含不经常使用的历史数据；最终用户可能愿意接受较长的响应时间，来等待对此数据执行的查询。在这种情况下，您可能会为历史表使用另一个表空间，并将此表空间分配给存取速率较低的较便宜的物理设备。

  或者，您也许能够标识某些表，这些表对于使数据快速可用以及您需要快速响应时间是必不可少的。可能要将这些表置于分配给一个快速物理设备的表空间中，这样将有助于支持这些重要的数据需要。

  通过使用 DMS 表空间，还可以将表数据分布在三个不同的表空间中：一个存储索引数据；一个存储 LOB 和长型字段数据；一个存储规则表数据。这允许您选择表空间特征和支持最适合该数据的那些表空间的物理设备。例如，可能会将索引数据置于可找到的最快的设备上，这样性能可显著提高。若将一个表分布在各 DMS 表空间中，如果启用前滚恢复，应考虑一起备份和复原那些表空间。SMS 表空间不支持以此方式将数据分布在所有表空间中。

- 管理问题。

  某些管理功能可以在表空间级执行，但不能在数据库或表级执行。例如，备份表空间（而不是数据库）可以帮助您更好地利用时间和资源。它允许频繁地备份带有大量更改的表空间，同时仅偶尔地备份带有少量更改的表空间。

  可以复原数据库或表空间。若不相关的表不共享表空间，就可以选择复原数据库一个较小的部分以降低成本。

  一种好方法是将相关的表编组在一组表空间中。这些表可以通过引用约束相关，也可以通过定义的其它商业约束相关。

  若经常需要删除并重新定义特定表，则应在它自己的表空间中定义该表，因为删除一个 DMS 表空间比删除一个表更有效率。

相关概念:

- 第 92 页的『数据库分区组』
- 第 104 页的『系统管理空间』
- 第 106 页的『数据库管理空间』
- 第 124 页的『SMS 和 DMS 表空间的比较』

第 6 章 设计分布式数据库

工作单元

事务在 DB2® 中通常称为*工作单元*。工作单元是应用程序进程内可恢复的操作序列。数据库管理器使用它来确保数据库处于一致状态。对数据库的任何读取或写入都在一个工作单元内执行。

例如，银行事务可能涉及将存款从储蓄帐户转至支票帐户。当应用程序从储蓄帐户中减去一定金额之后，这种不一致状态将一直保持，直到将该金额加入支票帐户为止。在两个步骤都完成后，便到达一致点。可以落实更改，并可使其用于其它应用程序。

当对数据库发出第一条 SQL 语句时，就启动了一个工作单元。应用程序必须发出 COMMIT 或 ROLLBACK 语句来结束该工作单元。COMMIT 语句使工作单元内的所有更改成为持久的。ROLLBACK 语句从数据库中除去这些更改。若应用程序正常结束，而未显式发出这两条语句中的任何一个，则自动落实该工作单元。若它在执行一个工作单元时异常结束，则自动回滚该工作单元。一旦发出 COMMIT 或 ROLLBACK，就不能停止它们。对于某些多线程应用程序或某些操作系统（如 Windows），若应用程序正常结束，而未显式发出这两条语句中的任何一个，则自动回滚工作单元。建议应用程序始终显式地落实或回滚，以完成工作单元。若部分工作单元未成功完成，则回滚更新，使参与的表保持该事务开始前的状态。这确保了请求既不会丢失，也不会重复。

相关参考:

- 『COMMIT statement』（*SQL Reference, Volume 2*）
- 『ROLLBACK statement』（*SQL Reference, Volume 2*）

在事务中更新单个数据库

事务的最简单形式是在单个工作单元中只对一个数据库读取和写入。此类数据库存取称为*远程工作单元*。

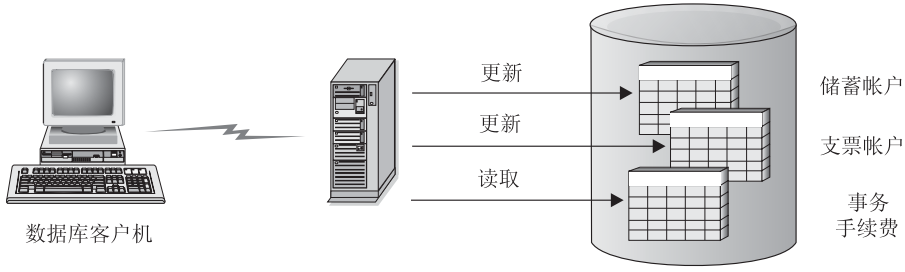

图 47. 在一个事务中使用单个数据库

图 47 显示了一个运行转帐应用程序的数据库客户机，该应用程序存取包含支票帐户和储蓄帐户表以及银行手续费表的数据库。该应用程序必须:

- 接受从用户界面指定的转帐金额
- 从储蓄帐户中减去该金额，然后确定新的余额
- 读取手续费表，以确定含有给定余额的储蓄帐户的事务手续费
- 从储蓄帐户中减去事务手续费
- 将转帐的金额添加至支票帐户
- 落实该事务（工作单元）

过程:

要设置这样的应用程序，必须:

1. 在同一个数据库中为储蓄帐户、支票帐户和银行业务手续费表创建表
2. 若物理上是远程的，则设置数据库服务器以使用相应的通信协议
3. 若物理上是远程的，则编目节点和数据库以便在数据库服务器上标识该数据库
4. 预编译应用程序以指定类型 1 连接；即在 PRECOMPILE PROGRAM 命令上指定 CONNECT 1（缺省值）

相关概念:

- 第 137 页的『工作单元』

相关任务:

- 第 139 页的『在多数据库事务中更新单个数据库』
- 第 140 页的『在事务中更新多个数据库』

相关参考:

- 『PRECOMPILE Command』（*Command Reference*）

在单个事务中使用多个数据库

当在单个事务中使用多个数据库时，对设置和管理环境的要求是不同的，这取决于在该事务中要更新的数据库的数量。

在多数据库事务中更新单个数据库

若您的数据分布在多个数据库上，您可能希望在从一个或多个数据库读取的同时更新一个数据库。可以在单个工作单元（事务）中执行此类存取。

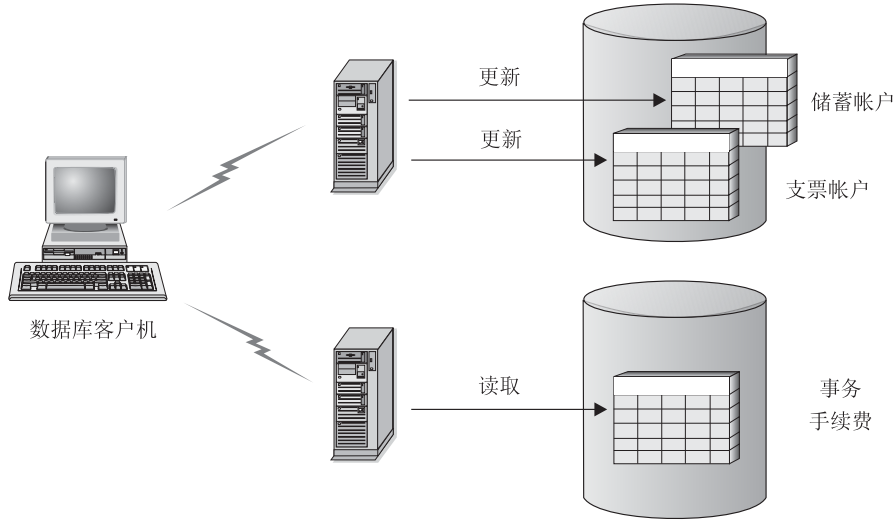

图 48. 在单个事务中使用多个数据库

图 48 显示了一个运行转帐应用程序的数据库客户机，该应用程序访问两个数据库服务器：一个包含支票和储蓄帐户，另一个包含银行业务手续费表。

过程:

要为此环境设置转帐应用程序，必须:

1. 在相应数据库中创建必须的表
2. 若物理上是远程的，则设置数据库服务器以使用相应的通信协议
3. 若物理上是远程的，则编目节点和数据库，以便在数据库服务器上标识这些数据库
4. 预编译应用程序以指定类型 2 连接（即在 PRECOMPILE PROGRAM 命令上指定 CONNECT 2）和一阶段落实（即在 PRECOMPILE PROGRAM 命令上指定 SYNCPOINT ONEPHASE）

若数据库位于主机或 iSeries 数据库服务器上，则需要 DB2 Connect 才能与这些服务器连接。

相关概念:

- 第 137 页的『工作单元』

相关任务:

- 第 137 页的『在事务中更新单个数据库』
- 第 140 页的『在事务中更新多个数据库』

相关参考:

- 『PRECOMPILE Command』（*Command Reference*）

在事务中更新多个数据库

若您的数据分布在多个数据库上，则您可能想在单个事务中读取和更新多个数据库。此类数据库存取称为*多站点更新*。

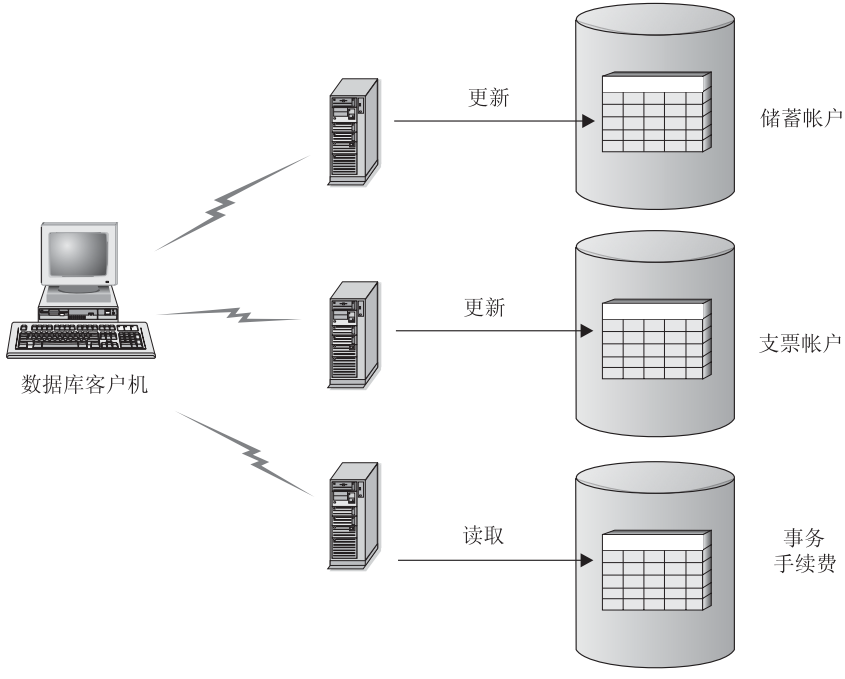

图 49. 在单个事务中更新多个数据库

第 140 页的图 49 显示了一个运行转帐应用程序的数据库客户机，该应用程序访问三个数据库服务器：一个包含支票帐户，另一个包含储蓄帐户，第三个包含银行业务手续费表。

过程:

要为此环境设置转帐应用程序，您有两种选择:

1. 使用 DB2 事务管理器（TM）:
   a. 在相应数据库中创建必须的表
   b. 若物理上是远程的，则设置数据库服务器以使用相应的通信协议
   c. 若物理上是远程的，则编目节点和数据库，以便在数据库服务器上标识这些数据库
   d. 预编译应用程序以指定类型 2 连接（即在 PRECOMPILE PROGRAM 命令上指定 CONNECT 2）和两阶段落实（即在 PRECOMPILE PROGRAM 命令上指定 SYNCPOINT TWOPHASE）
   e. 配置 DB2 事务管理器（TM）
2. 使用 XA 兼容事务管理器:
   a. 在相应数据库中创建必须的表
   b. 若物理上是远程的，则设置数据库服务器以使用相应的通信协议
   c. 若物理上是远程的，则编目节点和数据库，以便在数据库服务器上标识这些数据库
   d. 预编译应用程序以指定类型 2 连接（即在 PRECOMPILE PROGRAM 命令上指定 CONNECT 2）和一阶段落实（即在 PRECOMPILE PROGRAM 命令上指定 SYNCPOINT ONEPHASE）
   e. 配置 XA 兼容事务管理器以使用 DB2 数据库

相关概念:

- 第 137 页的『工作单元』
- 第 142 页的『DB2 事务管理器』

相关任务:

- 第 137 页的『在事务中更新单个数据库』
- 第 139 页的『在多数据库事务中更新单个数据库』

相关参考:

- 『PRECOMPILE Command』（*Command Reference*）

DB2 事务管理器

DB2® 事务管理器（TM）向事务指定标识符，监视它们的进度，并负责处理事务的完成和失败。DB2 通用数据库™（UDB）和 DB2 Connect™ 提供了事务管理器。DB2 TM 在指定的 TM 数据库中存储事务信息。

数据库管理器提供事务管理器功能，这些功能可用于协调在单个工作单元内几个数据库的更新。数据库客户机自动协调工作单元，并使用*事务管理器数据库*来注册每个事务并跟踪其完成状态。

可以将 DB2 事务管理器与 DB2 数据库配合使用。如果有不同于 DB2 数据库的资源想要参予两阶段落实事务，则可以使用 XA 兼容事务管理器。

相关概念:

- 第 137 页的『工作单元』
- 第 142 页的『DB2 事务管理器配置』
- 第 147 页的『两阶段落实』

DB2 事务管理器配置

如果正在使用 XA 兼容事务管理器，如 IBM® WebSphere、BEA Tuxedo 或 Microsoft® Transaction Server，则应遵循该产品的配置指示信息。

当对基于 UNIX 的系统或 Windows® 操作系统使用 DB2® UDB 协调事务时，则必须满足特定配置需求。若您只使用 TCP/IP 来进行通信，且您的事务仅涉及 DB2 UDB Unix 版、Windows 版、iSeries™ V5 版、z/OS™ 或 OS/390® 这些数据库服务器，配置会简单得多。

使用 TCP/IP 连接的 DB2 Unix 版和 Windows 版、DB2 z/OS 版、OS/390 版和 iSeries V5 版

若以下所有陈述都适合于您的环境，则多站点更新的配置步骤就会简化。

- 与远程数据库服务器（包括 DB2 UDB z/OS 版、OS/390 版和 iSeries V5 版）的所有通信都只使用 TCP/IP。
- 用于基于 UNIX® 的系统、Windows 操作系统、z/OS、OS/390 或 iSeries V5 的 DB2 UDB 是参与该事务的仅有数据库服务器。
- 未配置 DB2 Connect™ 同步点管理器（SPM）。

  DB2 Connect 同步点管理器是在创建 DB2 实例时自动配置的，它在下列情况下是必需的:

  – 主机或 iSeries 数据库服务器与 SNA 连接配合使用来进行多站点更新。

注: 应考虑切换至 TCP/IP，因为在将来发行版的 DB2 中可能不再支持 SNA。SNA 需要很多配置知识，并且配置进程本身经证明有错误倾向。TCP/IP 很容易配置，维护成本低并且提供很好的性能。

– XA 兼容事务管理器（如 IBM WebSphere ）正在协调两阶段落实。

这适用于与主机或 iSeries 数据库服务器的 SNA 和 TCP/IP 连接。若您的环境不需要 DB2 Connect 同步点管理器，则可以通过在 DB2 Connect 服务器上发出 `db2 update dbm cfg using spm_name NULL` 命令来将其关闭。然后停止并重新启动数据库管理器。

将用作事务管理器数据库的数据库由数据库客户机的数据库管理器配置参数 *tm_database* 确定。当设置此配置参数时，考虑下列因素:

- 事务管理器数据库可以是:
  - DB2 UDB UNIX 版或 Windows 版的版本 8 数据库
  - DB2 z/OS 版和 OS/390 版的版本 7 数据库或 DB2 OS/390 版的版本 5 或 6 数据库
  - DB2 iSeries V5 版数据库

    建议使用 DB2 z/OS 版、OS/390 版和 iSeries V5 版数据库服务器来作为事务管理器数据库。z/OS 版、OS/390 和 iSeries V5 系统通常比工作站服务器更安全，降低了意外断电和重新引导等的可能性。因此，在再同步时使用的恢复日志更安全。
- 若对 *tm_database* 配置参数指定了值 `1ST_CONN`，则与应用程序连接的第一个数据库被用作事务管理器数据库。

  当使用 `1ST_CONN` 时，必须小心。仅当能够比较容易地确保所有参与的数据库都正确编目时，才应使用此配置，即，如果:

  – 启动该事务的数据库客户机位于包含参与数据库（包括事务管理器数据库）的同一个实例中。

  注意，若您的应用程序试图与正用作事务管理器数据库的数据库断开连接，则将接收到警告消息，且该连接将被挂起，直到落实该工作单元为止。

其它环境

在您的环境中，如果:

- 与远程数据库服务器的通信不只使用 TCP/IP（例如，还使用 NETBIOS）
- 存取 DB2 AS/400® V4 版或 DB2 VM&VSE 版
- 使用 SNA 来存取 DB2 z/OS 版、OS/390 版或 iSeries V5 版
- 使用 DB2 Connect 同步点管理器访问主机或 iSeries 数据库服务器

多站点更新的配置步骤更为复杂。

将用作事务管理器数据库的数据库由数据库客户机的数据库管理器配置参数 *tm_database* 确定。如果使用“版本 8 DB2 运行时客户机”，则在配置此配置参数时考虑下列因素:

- 事务管理器数据库可以是 DB2 UDB UNIX 版、Windows 版、z/OS 版、OS/390 版或 iSeries V5 版数据库。事务管理器数据库不能是 DB2 UDB Unix 版、Windows 版或 OS/2® 版的版本 7 数据库。
- 使用的用户标识和密码将是存取的第一个数据库的用户标识和密码。
- 若对 *tm_database* 配置参数指定了值 1ST_CONN，则与应用程序连接的第一个数据库被用作事务管理器数据库。

  当使用 1ST_CONN 时，必须小心。仅当能够比较容易地确保所有参与的数据库都正确编目时，才应使用此配置，即，如果:

  – 启动该事务的数据库客户机位于包含参与数据库（包括事务管理器数据库）的同一个实例中。
  – 连接的第一个数据库是可接受用作事务管理器的数据库。

  注意，若您的应用程序试图与正用作事务管理器数据库的数据库断开连接，则将接收到警告消息，且该连接将被挂起，直到落实该工作单元为止。

配置参数

当您设置环境时，应该考虑下列配置参数。

数据库管理器配置参数

- *tm_database*

  此参数标识每个 DB2 实例的“事务管理器”（TM）数据库的名称。

- *spm_name*

  此参数向数据库管理器标识 DB2 Connect 同步点管理器实例的名称。为了使再同步成功，该名称在您的网络中必须是唯一的。

- *resync_interval*

  此参数标识一个时间间隔（以秒计），在该时间间隔之后，“DB2 事务管理器”、DB2 服务器数据库管理器以及 DB2 Connect 同步点管理器或 DB2 UDB 同步点管理器应该重新尝试恢复任何未完成的不确定事务。

- *spm_log_file_sz*

  此参数指定 SPM 日志文件的大小（以 4 KB 页计）。

- *spm_max_resync*

  此参数标识可同时执行再同步操作的代理进程数。

- *spm_log_path*

此参数标识 SPM 日志文件的日志路径。

数据库配置参数

- *maxappls*

  此参数指定允许的最大活动应用程序数。它的值必须等于或大于连接的应用程序数、可能正在同时完成一个两阶段落实或回滚的相同应用程序的数目以及预期任何时候可能存在的不确定事务的数目之和。

- *autorestart*

  此数据库配置参数指定在需要时是否自动调用 RESTART DATABASE 例程。缺省值是 YES（即，已启用）。

  包含不确定事务的数据库需要重新启动数据库操作才能启动。若断开与该数据库的上一个连接时未启用 *autorestart*，则下一个连接将失败并需要显式的 RESTART DATABASE 调用。这种情况将一直存在，直到事务管理器的再同步操作或管理员启动的试探性操作除去了不确定事务为止。当发出 RESTART DATABASE 命令时，若该数据库中有任何不确定事务，将返回一条消息。管理员就可以使用 LIST INDOUBT TRANSACTIONS 命令和其它的命令行处理器命令来获取有关那些不确定事务的信息。

相关概念:

- 第 142 页的『DB2 事务管理器』

相关任务:

- 第 170 页的『配置 IBM TXSeries CICS』
- 第 171 页的『配置 IBM TXSeries Encina』
- 第 172 页的『配置 BEA Tuxedo』
- 第 170 页的『配置 IBM WebSphere Application Server』

相关参考:

- 『“同步点管理器日志文件路径”配置参数 — spm_log_path』（*《管理指南: 性能》*）
- 『“自动重新启动启用”配置参数 — autorestart』（*《管理指南: 性能》*）
- 『“活动应用程序的最大数目”配置参数 — maxappls』（*《管理指南: 性能》*）
- 『“事务重新同步时间间隔”配置参数 — resync_interval』（*《管理指南: 性能》*）
- 『“事务管理器数据库名”配置参数 — tm_database』（*《管理指南: 性能》*）
- 『“同步点管理器名”配置参数 — spm_name』（*《管理指南: 性能》*）
- 『“同步点管理器日志文件大小”配置参数 — spm_log_file_sz』（*《管理指南: 性能》*）

- 『“同步点管理器重新同步代理进程限制”配置参数 — spm_max_resync』（*《管理指南: 性能》*）

从主机或 iSeries 客户机更新数据库

在主机或 iSeries 上执行的应用程序可以存取驻留在 DB2 UDB Unix 版和 Windows 版数据库服务器上的数据。此存取可以使用 TCP/IP 和 SNA。如果使用 SNA，则 DB2 UDB 服务器上需要有同步点管理器（SPM）。如果使用 TCP/IP，则不使用 SPM。

注: 如果正在使用 SNA，则应考虑切换至 TCP/IP，因为在将来发行版的 DB2 中可能不再支持 SNA。SNA 需要很多配置知识，并且配置进程本身经证明有错误倾向。TCP/IP 很容易配置，维护成本低并且提供很好的性能。

从主机或 iSeries 数据库客户机访问的数据库服务器不必在带有 DB2 同步点管理器的工作站本地。主机或 iSeries 数据库客户机可以连接至将 DB2 同步点管理器工作站用作中间网关的 DB2 UDB 服务器。这允许您将 DB2 同步点管理器工作站隔离在一个安全的环境中，而实际的 DB2 UDB 服务器位于您公司网络中的远程位置。这也允许 DB2 公共服务器版本 2 数据库参与从主机或 iSeries 数据库客户机发出的多站点更新。

过程:

在主机或 iSeries 应用程序将要直接访问的工作站上:

1. 安装“DB2 通用数据库企业版”以向主机或 iSeries 数据库客户机提供多站点更新支持。
2. 在同一个系统上创建数据库实例。例如，可以使用缺省实例 `DB2`，或使用如下命令来创建新实例:

   ```
   db2icrt myinstance
   ```

3. 按要求提供许可信息。
4. 确保注册表值 DB2COMM 包括值 `APPC` 和／或 `TCP/IP`。
5. 如果从主机或 iSeries 客户机使用 SNA，则必须配置同步点管理器:
   a. 按要求配置 SNA 通信。当使用受支持的 IBM SNA 产品时，将根据 *spm_name* 数据库管理器配置参数的值自动创建 DB2 同步点管理器所需的 SNA 概要文件。任何其它受支持的 SNA 堆栈都需要手工配置。
   b. 确定要对 *spm_name* 数据库管理器配置参数指定的值。此参数是在创建 DB2 实例时预先配置的，并且是从机器的 TCP/IP 主机名派生的。若它在您的环境中可接受且是唯一的，则不要更改它。

c. 必要时，使用 UPDATE DATABASE MANAGER CONFIGURATION 命令来在“DB2 通用数据库”服务器上更新 *spm_name*。

6. 配置此 DB2 工作站连接远程 DB2 UDB 服务器所需的通信（若有的话）。
7. 配置远程 DB2 UDB 服务器连接此 DB2 服务器所需的通信。
8. 在“DB2 通用数据库”服务器上停止并重新启动数据库管理器。

   应该能够从此 DB2 UDB 工作站与远程 DB2 UDB 服务器连接。

在主机或 iSeries 数据库客户机将要访问的每个远程 DB2 UDB 服务器上:

1. 配置带有 DB2 同步点管理器的远程 DB2 UDB 工作站与此 DB2 UDB 服务器连接所需的通信。
2. 停止并重新启动数据库管理器。

相关参考:

- 『“同步点管理器名”配置参数 — spm_name』（*《管理指南: 性能》*）
- 『通信变量』（*《管理指南: 性能》*）

两阶段落实

第 148 页的图 50 举例说明了多站点更新包括的步骤。若两阶段落实过程期间发生错误，则了解一个事务是如何管理的将有助于您解决问题。

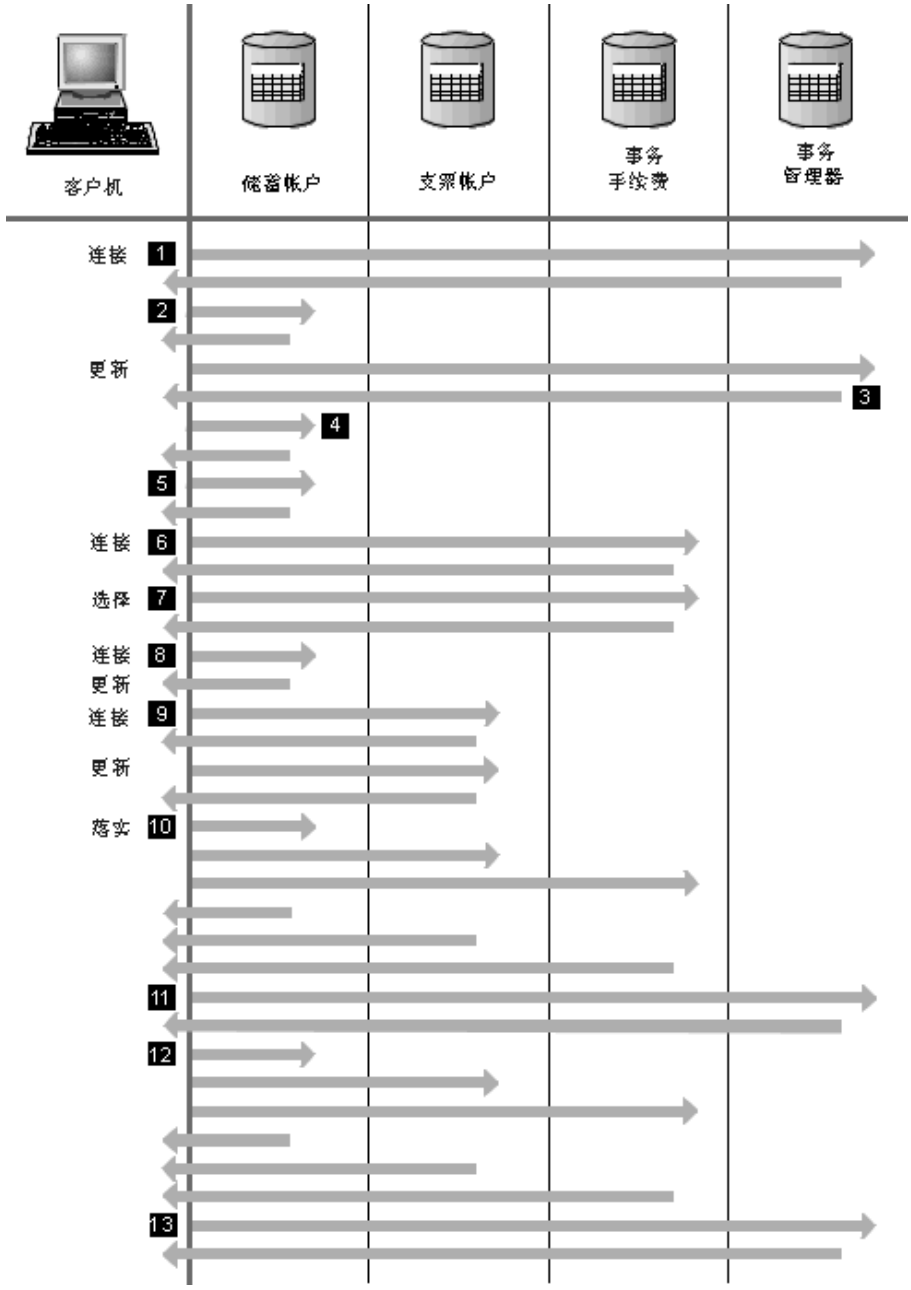

图 50. 更新多个数据库

0 为两阶段落实准备该应用程序。此过程可以通过预编译选项来完成。此过程还可以通过 DB2® CLI（调用层接口）配置来完成。

1 当数据库客户机要与 SAVINGS_DB 数据库连接时，它首先在内部与事务管理器（TM）数据库连接。该 TM 数据库将确认返回至数据库客户机。若将数据库管理器配置参数 *tm_database* 设置为 1ST_CONN，则 SAVINGS_DB 在此应用程序实例的持续时间内成为事务管理器数据库。

2 建立与 SAVINGS_DB 数据库的连接，并确认该连接。

3 数据库客户机开始更新 SAVINGS_ACCOUNT 表。这开始工作单元。TM 数据库提供该工作单元的一个事务标识，以应答数据库客户机。注意，当运行该工作单元中的第一条 SQL 语句时，而不是在建立连接期间，要对该工作单元进行注册。

4 当接收到该事务标识之后，数据库客户机向包含 SAVINGS_ACCOUNT 表的数据库注册该工作单元。将一个应答发送回客户机，以指示该工作单元已成功注册。

5 以正常方式处理对 SAVINGS_DB 数据库发出的 SQL 语句。当使用在一个程序中嵌入的 SQL 语句时，对每条语句的应答都会在 SQLCA 中返回。

6 当在工作单元内第一次存取该数据库期间，在包含 TRANSACTION_FEE 表的该 FEE_DB 数据库中注册该事务标识。

7 以正常方式处理对 FEE_DB 数据库发出的任何 SQL 语句。

8 可以适当设置该连接，来对 SAVINGS_DB 数据库运行附加的 SQL 语句。由于已向 SAVINGS_DB 数据库注册了该工作单元 **4**，因此数据库客户机不必再次执行注册步骤。

9 连接 CHECKING_DB 数据库，并遵循 **6** 和 **7** 中描述的规则来使用它。

10 当数据库客户机请求落实工作单元时，会将一条*准备*消息发送至参与该工作单元的所有数据库。每个数据库都将一个 “PREPARED” 记录写至它的日志文件，并答复数据库客户机。

11 在数据库客户机接收到来自所有数据库的肯定响应之后，它会向事务管理器数据库发送一条消息，以通知数据库现在已准备落实工作单元（PREPARED）。该事务管理器数据库则将一个 “PREPARED” 记录写入它的日志文件，并发送答复，以通知该客户机可以启动落实过程的第二阶段。

12 在落实过程的第二阶段期间，数据库客户机会向所有参与数据库发送一条消息，告知它们要落实。每个数据库则将一个 “COMMITTED” 记录写入它的日志文件，并释放为此工作单元挂起的锁定。当该数据库完成对更改的落实时，它会将一个答复发送至客户机。

13 在数据库客户机接收到来自所有参与数据库的肯定应答之后，它会向事务

管理器数据库发送一条消息，以通知数据库该工作单元已完成。于是，事务管理器数据库将一个“COMMITTED”记录写至其日志文件，指示该工作单元已完成，并答复该客户机，指示它已完成。

相关概念:

- 第 137 页的『工作单元』
- 第 142 页的『DB2 事务管理器』

两阶段落实期间的错误恢复

从错误状态恢复是一个与应用程序编程、系统管理、数据库管理和系统操作相关的正常任务。将数据库分布在几个远程服务器上会增加由于网络或通信故障而导致错误的可能性。要确保数据完整性，数据库管理器提供两阶段落实进程。下面解释数据库管理器如何在两阶段落实过程期间处理错误:

- **第一阶段的错误**

  若一个数据库应答“它准备落实工作单元失败”，则数据库客户机将在落实过程的第二阶段期间回滚该工作单元。在这种情况下，不会向事务管理器数据库发送准备消息。

  在第二个阶段期间，客户机会向所有在第一阶段成功地准备了落实的参与数据库发送一条回滚消息。于是，每个数据库将一个“ABORT”记录写入它的日志文件，并释放为此工作单元挂起的锁定。

- **第二阶段的错误**

  此阶段中的错误处理取决于第二阶段将落实还是回滚该事务。若第一阶段遇到错误，则第二阶段将只回滚该事务。

  若其中一个参与数据库落实工作单元失败（可能由于通信故障），则事务管理器数据库将对失败的数据库重试落实。然而，将通过 SQLCA 通知应用程序落实成功。DB2® 将确保落实数据库服务器中尚未落实的事务。数据库管理器配置参数 *resync_interval* 用来指定事务管理器数据库在尝试落实工作单元之间要等待的时间。在数据库服务器上挂起所有锁定，直到落实工作单元为止。

  若事务管理器数据库失败，它将在重新启动数据库时使该工作单元再同步。再同步过程将试图完成所有*不确定事务*；即，已完成第一阶段但未完成第二阶段落实过程的那些事务。与事务管理器数据库相关联的数据库管理器通过下列各项执行再同步:

  1. 与指示在落实过程的第一阶段期间“已准备”（PREPARED）落实的数据库连接。
  2. 试图落实那些数据库中的不确定事务。（若找不到不确定事务，则数据库管理器假定在落实过程的第二阶段期间该数据库已成功落实该事务。）

3. 当在参与数据库中落实了所有不确定事务之后，再落实事务管理器数据库中的不确定事务。

若其中一个参与数据库失败并重新启动，则此数据库的数据库管理器将查询事务管理器数据库，以获知此事务的状态，来确定是否应该回滚该事务。若在日志中找不到该事务，则数据库管理器假定该事务已被回滚，且该事务将回滚此数据库中的不确定事务。否则，该数据库等待事务管理器数据库发出落实请求。

若该事务已由事务处理监视器（XA 兼容的事务管理器）协调，则数据库将始终依靠 TP 监视器来启动该再同步。

若由于某种原因您不能等待事务管理器自动解决不确定事务，则您可以执行一些操作来以手工方式解决它们。此手工过程有时称为“作试探性决定”。

当 autorestart=off 时的错误恢复

如果 *autorestart* 数据库配置参数设置为 OFF，且 TM 或 RM 数据库中存在不确定事务，则需要执行 RESTART DATABASE 命令才能启动再同步进程。当从命令行处理器发出 RESTART DATABASE 命令时，将使用不同的会话。若您从同一个会话中重新启动另一个数据库，则先前重新启动数据库命令所建立的连接将被断开，并且必须再次重新启动。在 LIST INDOUBT TRANSACTIONS 命令不再返回更多不确定事务之后，发出 TERMINATE 命令来断开该连接。

相关概念:

- 第 147 页的『两阶段落实』

相关任务:

- 第 163 页的『手工解析不确定事务』

相关参考:

- 『“自动重新启动启用”配置参数 — autorestart』（*《管理指南: 性能》*）
- 『LIST INDOUBT TRANSACTIONS Command』（*Command Reference*）
- 『TERMINATE Command』（*Command Reference*）
- 『RESTART DATABASE Command』（*Command Reference*）

第 7 章 针对 XA 兼容事务管理器进行设计

若有除 DB2 数据库以外的资源，且希望它参与两阶段落实事务，您可能要在 XA 兼容事务管理器中使用您的数据库。如果事务仅存取 DB2 数据库，应使用 DB2 事务管理器，如第 140 页的『在事务中更新多个数据库』中所述。

下列主题将帮助您将数据库管理器与 XA 兼容事务管理器（如 IBM WebSphere 或 BEA Tuxedo）配合使用。

- 第 154 页的『X/Open 分布式事务处理模型』
- 第 157 页的『资源管理器设置』
- 第 158 页的『xa_open 字符串格式』
- 第 163 页的『手工解析不确定事务』
- 第 165 页的『XA 事务管理器的安全性注意事项』
- 第 166 页的『XA 事务管理器的配置注意事项』
- 第 167 页的『DB2 UDB 支持的 XA 功能』
- 第 169 页的『XA 接口问题确定』
- 第 170 页的『配置 IBM WebSphere Application Server』
- 第 170 页的『配置 IBM TXSeries CICS』
- 第 171 页的『配置 IBM TXSeries Encina』
- 第 172 页的『配置 BEA Tuxedo』

如果在查找关于 Microsoft Transaction Server 的信息，参见 *CLI Guide and Reference, Volume 1*。

若您使用的是 XA 兼容事务管理器，或正在实现该程序，请从我们的技术支持 Web 站点获取更多信息:

`http://www.ibm.com/software/data/db2/udb/winos2unix/support`

在此处选择 “DB2 通用数据库”，然后使用关键字 “XA” 在 Web 站点中搜索关于 XA 兼容事务管理器的最新可用信息。

X/Open 分布式事务处理模型

“X/Open 分布式事务处理”（DTP）模型包括三个相关的组件:

- 应用程序（AP）
- 事务管理器（TM）
- 资源管理器（RM）

图 51 举例说明了此模型，并显示了这些组件之间的关系。

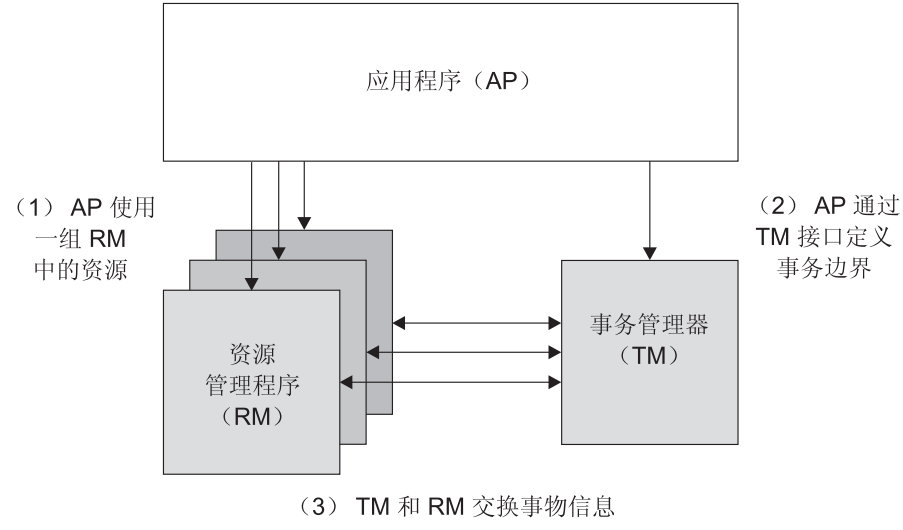

图 51. X/Open 分布式事务处理（DTP）模型

应用程序（AP）

应用程序（AP）定义事务边界，并定义构成该事务的应用程序特定的操作。

例如，CICS* 应用程序也许要访问资源管理器（RM），如数据库和 “CICSS®瞬时数据队列”，并使用编程逻辑来操纵数据。通过针对该资源管理器的函数调用，将每个存取请求传送至适当的 RM。对于 DB2，这些可能是由 DB2® 预编译器为每条 SQL 语句生成的函数调用，或是直接由程序员使用该 API 来编码的数据库调用。

事务管理器（TM）产品通常包括用于运行用户应用程序的事务处理（TP）监视器。TP 监视器提供 API 以允许应用程序启动和结束事务，在希望运行该应用程序

的多个用户之间执行应用程序的调度和负载平衡。分布式事务处理（DTP）环境中的应用程序实际上是用户应用程序和 TP 监视器的组合。

要实现有效的联机事务处理（OLTP）环境，TP 监视器会在启动时预先分配大量服务器进程，然后在多个用户事务之间调度和再利用它们。这样通过允许使用更少的服务器进程及其对应的 RM 进程来支持更多的并行用户，节省了系统资源。再利用这些进程还避免了为每个用户的事务或程序在 TM 和 RM 中启动一个进程的开销。（一个程序调用一个或多个事务。）这还意味着这些服务器进程对于 TM 和 RM 是真正的“用户进程”。它隐含安全性管理和应用程序编程。

从 TP 监视器可进行下列类型的事务:

- 非 XA 事务

  这些事务涉及由于未定义给 TM 而未在 TM 的两阶段落实协议下协调的 RM。若该应用程序需要访问不支持 XA 接口的 RM，这可能是需要的。TP 监视器只是提供有效的应用程序调度和负载平衡。由于 TM 不明确“打开”RM 来进行 XA 处理，所以 RM 将此应用程序作为在非 DTP 环境中运行的任何其它应用程序来处理。

- 全局事务

  这些事务涉及定义给 TM 并在 TM 的两阶段落实控制下的 RM。全局事务是可能涉及一个或多个 RM 的工作单元。*事务分支*是 TM 和 RM 之间支持全局事务的工作的一部分。当通过一个或多个由 TM 协调的应用程序进程来访问多个 RM 时，一个全局事务可能有多个事务分支。

  当多个应用程序进程中的每一个都访问 RM 时，它们就象在单独的全局事务中一样，存在松散耦合的全局事务，但是那些应用程序处于 TM 的协调下。每个应用程序进程在一个 RM 内将有它自己的事务分支。当 AP、TM 或 RM 中任何一个请求落实或回滚时，会一起完成事务分支。该应用程序负责确保在这些分支之间不发生资源死锁。（注意，DB2 事务管理器为使用 SYNCPOINT(TWOPHASE) 选项准备的应用程序执行的事务协调大致等效于这些松散耦合的全局事务。

  当多个应用程序进程轮流在一个 RM 中的相同事务分支下工作时，就会存在紧密耦合的全局事务。对于 RM，这两个应用程序进程是一个单个的实体。RM 必须确保在事务分支内不发生资源死锁。

事务管理器（TM）

事务管理器（TM）向事务指定标识符，监视它们的进程，并负责处理事务的完成和失败。事务分支标识符（称为 XID）由 TM 指定，以标识一个 RM 内的全局事务和特定分支。它是一个 TM 中的日志与一个 RM 中的日志之间的相关标记。两阶段落实或回滚需要 XID，以便在系统启动时执行*再同步*操作（也称为*再同步*（*resync*）），或在需要时允许管理员执行*试探*操作（也称为*手工干预*）。

在 TP 监视器启动后，它请求 TM 打开一组应用程序服务器已定义的所有 RM。TM 将 **xa_open** 调用传送至 RM，以便可以将它们初始化以进行 DTP 处理。作为此启动过程的一部分，TM 执行再同步以恢复所有的*不确定事务*。不确定事务是处于未确定状态的全局事务。当成功完成两阶段落实协议的第一阶段（即，准备阶段）之后 TM（或至少一个 RM）成为不可用时，就会出现此情况。在 TM 可以使自己的日志与再次变成可用的 RM 的日志相符合之前，RM 不会知道要落实还是回滚它的事务分支。要执行再同步操作，TM 会一次或多次地将 **xa_recover** 调用发出至每个 RM，以标识所有不确定事务。TM 将这些回答与它自己的日志中的信息作比较，以确定应该通知 RM **xa_commit** 那些事务，还是 **xa_rollback** 那些事务。若通过管理员执行试探操作，RM 已经落实或回滚了它的不确定事务的分支，则 TM 向该 RM 发出 **xa_forget** 调用来完成再同步操作。

当用户应用程序请求落实或回滚时，它必须使用 TP 监视器或 TM 提供的 API，以便 TM 可以在所有参与的 RM 之间协调落实和回滚。例如，当 CICS 应用程序发出 CICS SYNCPOINT 请求以落实一个事务时，CICS XA TM（在 “Encina®服务器” 中实现）将发出 XA 调用，如 **xa_end**、**xa_prepare**、**xa_commit** 或 **xa_rollback**，以请求 RM 落实或回滚该事务。若只涉及一个 RM，或者若 RM 回答它的分支是只读的，则 TM 可选择使用一阶段落实以替代两阶段落实。

资源管理器（RM）

资源管理器（RM）提供对共享资源（如数据库）的存取。

作为数据库的资源管理器的 DB2 可以参与由 XA 兼容的 TM 协调的*全局事务*。根据 XA 接口的要求，数据库管理器提供 xa_switch_t 类型的 *db2xa_switch* 外部 C 变量，以将 XA 开关结构返回给 TM。此数据结构包含由该 TM 调用的各种 XA 例程的地址和 RM 的操作特征。

RM 可使用两种方法来对它参与每个全局事务进行注册：*静态注册*和*动态注册*：

- 静态注册要求 TM（为每个事务）将 **xa_start**、**xa_end** 和 **xa_prepare** 等一系列调用发出至为服务器应用程序定义的所有 RM，而不管给定的 RM 是否正由该事务使用。若并非每个事务都包括每个 RM，则这是低效的，并且低效的程度与已定义的 RM 的数目成比例。
- 动态注册（由 DB2 使用）灵活且高效。仅当 RM 接收到对其资源的请求时，该 RM 才使用 **ax_reg** 调用向 TM 注册。注意，即使只定义了一个 RM 时，或当每个事务都使用每个 RM 时，此方法也不存在性能方面的缺点，因为 **ax_reg** 和 **xa_start** 调用在 TM 中具有相似的路径。

XA 接口提供 TM 和 RM 之间的双向通信。它是两个 DTP 软件组件之间的系统级接口，而不是应用程序开发者编码的普通应用程序接口。但是，应用程序开发人员应该熟悉该 DTP 软件组件所利用的编程限制。

虽然 XA 接口是不变的，但是每个 XA 兼容的 TM 可能具有集成 RM 的产品特定方式。有关将 DB2 产品作为资源管理器与特定的事务管理器集成的信息，参见适当的 TM 产品文档。

相关概念:

- 第 165 页的『XA 事务管理器的安全性注意事项』
- 第 167 页的『DB2 UDB 支持的 XA 功能』
- 『X/Open XA Interface Programming Considerations』（*Application Development Guide: Programming Client Applications*）

相关任务:

- 第 140 页的『在事务中更新多个数据库』

资源管理器设置

将每个数据库定义为事务管理器（TM）的一个单独的资源管理器（RM），且必须用 xa_open 字符串标识该数据库。

当将一个数据库设置为资源管理器时，不需要 xa_close 字符串。若提供了此字符串，数据库管理器会忽略它。

数据库连接注意事项

存取分区数据库的事务

在分区数据库环境中，可将用户数据分布在各数据库分区中。存取该数据库的应用程序与其中一个数据库分区（协调程序节点）连接，并向其发送请求。不同的应用程序可以与不同的数据库分区连接，而同一个应用程序可以为不同的连接选择不同的数据库分区。

对于对分区数据库环境中的某个数据库执行的事务，所有的存取都必须通过*同*一个数据库分区。即，从启动该事务起，直到落实该事务为止（包括落实时的那一刻），都必须使用同一个数据库分区。

对分区数据库执行的任何事务都必须在断开连接之前落实。

相关概念:

- 第 154 页的『X/Open 分布式事务处理模型』

相关参考:

- 第 158 页的『xa_open 字符串格式』

DB2 版本 7 和更新版本的 xa_open 字符串格式

以下是 xa_open 字符串的格式:

```
parm_id1 = <parm value>,parm_id2 = <parm value>, ...
```

以何次序指定这些参数并不重要。下表描述了 *parm_id* 的有效值。

表 21. parm_id 的有效值

参数名	值	是否必要?	是否区分大小写?	缺省值
DB	数据库别名	是	否	无
应用程序用来存取数据库的数据库别名。				
UID	用户标识	否	是	无
有权与数据库连接的用户标识。若指定密码，则是必需的。				
PWD	密码	否	是	无
与用户标识相关联的密码。若指定用户标识，则是必需的。				
TPM	事务处理监视器名	否	否	无
正在使用的 TP 监视器的名称。有关受支持的值，参见下一个表。可指定此参数以允许多个 TP 监视器使用单一 DB2 实例。指定的值将覆盖 *tp_mon_name* 数据库管理器配置参数中指定的值。				
AXLIB	包含 TP 监视器的 **ax_reg** 和 **ax_unreg** 函数的库。	否	是	无
DB2 使用此值来获取必需的 **ax_reg** 和 **ax_unreg** 函数的地址。可使用它来覆盖根据 TPM 参数假定的值，它也可以由未出现在 TPM 列表中的 TP 监视器使用。				
CHAIN_END	xa_end 链接标志。有效值是 T、F 或没有值。	否	否	F
XA_END 链接是 DB2 可用来减少网络流的优化。若 TP 监视器环境可以保证在调用 **xa_end** 之后，将立即在同一线程或进程中调用 **xa_prepare**，且 CHAIN_END 打开，则 xa_end 标志将与 **xa_prepare** 命令链接，从而消去一个网络流。值 T 表示 CHAIN_END 打开；值 F 表示 CHAIN_END 关闭；未指定值表示 CHAIN_END 打开。可使用此参数来覆盖从指定的 TPM 值派生的设置。				

表 21. parm_id 的有效值 (续)

参数名	值	是否必要?	是否区分大小写?	缺省值
SUSPEND_CURSOR	指定当挂起事务控制线程时是否保留游标。有效值是 T、F 或没有值。	否	否	F
挂起事务分支的 TP 监视器可以将暂挂的线程或进程重用于其它事务。如果 SUSPEND_CURSOR 关闭，则除了带有挂起属性的游标之外，将关闭所有游标。当恢复暂挂的事务时，应用程序必须再次获取游标。如果 SUSPEND_CURSOR 打开，则不关闭任何打开的游标，并且在恢复暂挂的事务时对该事务是可用的。值 T 表示 SUSPEND_CURSOR 打开；值 F 表示 SUSPEND_CURSOR 关闭；未指定值表示 SUSPEND_CURSOR 打开。可使用此参数来覆盖从指定的 TPM 值派生的设置。				
HOLD_CURSOR	指定是否跨事务落实保持游标。有效值是 T、F 或没有值。	否	否	F
通常，TP 监视器将线程或进程重用于多个应用程序。要确保新装入的应用程序不会继承由先前应用程序打开的游标，游标在落实之后被关闭。如果 HOLD_CURSORS 为打开，则带有挂起属性的游标不会关闭，并且将在事务落实边界保持不变。当使用此选项时，全局事务必须在同一线程的控制下落实或回滚。如果 HOLD_CURSOR 关闭，则将拒绝打开任何带有挂起属性的游标。值 T 表示 HOLD_CURSOR 打开；值 F 表示 HOLD_CURSOR 关闭；未指定值表示 HOLD_CURSOR 打开。可使用此参数来覆盖从指定的 TPM 值派生的设置。				

TPM 和 tp_mon_name 值

xa_open 字符串 TPM 参数和 *tp_mon_name* 数据库管理器配置参数用来向 DB2 指示正在使用的 TP 监视器。*tp_mon_name* 值适用于整个 DB2 实例。TPM 参数仅适用于特定 XA 资源管理器。TPM 值覆盖 *tp_mon_name* 参数。TPM 和 *tp_mon_name* 参数的有效值如下所示:

表 22. TPM 和 tp_mon_name 的有效值

TPM 值	TP 监视器产品	内部设置
CICS	IBM TxSeries CICS	AXLIB=libEncServer（对于 Windows） =/usr/lpp/encina/lib/libEncServer （对于基于 UNIX 的系统） HOLD_CURSOR=T CHAIN_END=T SUSPEND_CURSOR=F

表 22. TPM 和 tp_mon_name 的有效值 (续)

TPM 值	TP 监视器产品	内部设置
ENCINA	IBM TxSeries Encina 监视器	AXLIB=libEncServer（对于 Windows） =/usr/lpp/encina/lib/libEncServer （对于基于 UNIX 的系统） HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
MQ	IBM MQSeries	AXLIB=mqmax （对于 Windows） =/usr/mqm/lib/libmqmax.a （对于 AIX） =/opt/mqm/lib/libmqmax.a （对于 Solaris） HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F
CB	IBM Component Broker	AXLIB=somtrx1i （对于 Windows） =libsomtrx1 （对于基于 UNIX 的系统） HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
SF	IBM San Francisco	AXLIB=ibmsfDB2 HOLD_CURSOR=F CHAIN_END=T SUSPEND_CURSOR=F
TUXEDO	BEA Tuxedo	AXLIB=libtux HOLD_CURSOR=F CHAIN_END=F SUSPEND_CURSOR=F
MTS	Microsoft 事务处理服务器	并不一定要为 MTS 配置 DB2。DB2 的 ODBC 驱动程序自动检测 MTS。
JTA	Java 事务 API	并不一定要为“企业 Java 服务器”（EJS）（例如 IBM WebSphere）配置 DB2。DB2 的 JDBC 驱动程序自动检测此环境。因此，忽略此 TPM 值。

较早版本的 xa_open 字符串格式

较早版本的 DB2 使用这里描述的 xa_open 字符串格式。为了进行兼容，仍支持此格式。应在可能的时候将应用程序迁移至新格式。

将每个数据库定义为事务管理器（TM）的一个单独的资源管理器（RM），且必须用 xa_open 字符串标识该数据库，该字符串的语法如下:

```
"database_alias<,userid,password>"
```

database_alias 用于指定数据库的别名。除非您在创建数据库之后显式地编目了一个别名，否则，别名与数据库名相同。用户名和密码是可选的（这取决于认证方法），它们将用于向数据库提供认证信息。

示例

1. 您正在 Windows NT 上使用 IBM TxSeries CICS。TxSeries 文档指示您需要使用值 `libEncServer:C` 来配置 *tp_mon_name*。这仍是可接受的格式；然而，对于 DB2 UDB 或 DB2 Connect 版本 7，您可以选择:
   - 指定值为 `CICS` 的 *tp_mon_name*（建议用于此方案）:

     ```
     db2 update dbm cfg using tp_mon_name CICS
     ```

     对于向 CICS 定义的每个数据库，在“区域” —> “资源” —> “产品” —> XAD —> “资源管理器初始化字符串”中，指定:

     ```
     db=dbalias,uid=userid,pwd=password
     ```

   - 对于向 CICS 定义的每个数据库，在“区域” —> “资源” —> “产品” —> XAD —> “资源管理器初始化字符串”中，指定:

     ```
     db=dbalias,uid=userid,pwd=password,tpm=cics
     ```

2. 您正在 Windows NT 上使用 IBM MQSeries。MQSeries 文档指示您需要使用值 `mqmax` 来配置 *tp_mon_name*。这仍是可接受的格式；然而，对于 DB2 UDB 或 DB2 Connect 版本 7，您可以选择:
   - 指定值为 `MQ` 的 *tp_mon_name*（建议用于此方案）:

     ```
     db2 update dbm cfg using tp_mon_name MQ
     ```

     对于向 CICS 定义的每个数据库，在“区域” —> “资源” —> “产品” —> XAD —> “资源管理器初始化字符串”中，指定:

     ```
     uid=userid,db=dbalias,pwd=password
     ```

   - 对于向 CICS 定义的每个数据库，在“区域” —> “资源” —> “产品” —> XAD —> “资源管理器初始化字符串”中，指定:

     ```
     uid=userid,db=dbalias,pwd=password,tpm=mq
     ```

3. 您正在 Windows NT 上同时使用 IBM TxSeries CICS 和 IBM MQSeries。正在使用单个 DB2 实例。在此方案中，应按如下方法进行配置:
   a. 对于向 CICS 定义的每个数据库，在“区域” —> “资源” —> “产品” —> XAD —> “资源管理器初始化字符串”中，指定:

      ```
      pwd=password,uid=userid,tpm=cics,db=dbalias
      ```

   b. 对于在队列管理器特性中定义成资源的每个数据库，将 XaOpenString 指定为:

      ```
      db=dbalias,uid=userid,pwd=password,tpm=mq
      ```

4. 您正在 Windows NT 上开发您自己的 XA 兼容事务管理器（XA TM），您想要告知 DB2 库“myaxlib”有必需的函数 **ax_reg** 和 **ax_unreg**。库“myaxlib”在 PATH 语句中指定的目录中。您可以选择:
   - 指定值为 myaxlib 的 *tp_mon_name*:

     ```
     db2 update dbm cfg using tp_mon_name myaxlib
     ```

     并且，对于向 XA TM 定义的每个数据库，指定 xa_open 字符串:

     ```
     db=dbalias,uid=userid,pwd=password
     ```

   - 对于向 XA TM 定义的每个数据库，指定 xa_open 字符串:

     ```
     db=dbalias,uid=userid,pwd=password,axlib=myaxlib
     ```

5. 您正在 Windows NT 上开发您自己的 XA 兼容事务管理器（XA TM），您想要告知 DB2 库“myaxlib”有必需的函数 **ax_reg** 和 **ax_unreg**。库“myaxlib”在 PATH 语句中指定的目录中。您还想启用 XA END 链接。您可以选择:
   - 对于向 XA TM 定义的每个数据库，指定 xa_open 字符串:

     ```
     db=dbalias,uid=userid,pwd=password,axlib=myaxlib,chain_end=T
     ```

   - 对于向 XA TM 定义的每个数据库，指定 xa_open 字符串:

     ```
     db=dbalias,uid=userid,pwd=password,axlib=myaxlib,chain_end
     ```

相关概念:

- 第 154 页的『X/Open 分布式事务处理模型』

相关参考:

- 『“事务处理器监视器名”配置参数 — tp_mon_name』（*《管理指南: 性能》*）

使用 XA 兼容事务管理器更新主机或 iSeries 数据库服务器

主机和 iSeries 数据库服务器可能是可更新的，这取决于“XA 事务管理器”的体系结构。

过程:

要支持来自不同进程的落实序列，必须启用 DB2 Connect 连接集中器。要启用 DB2 Connect 连接集中器，将数据库管理器配置参数 *max_connections* 设置为大于 *maxagents* 的值。注意，DB2 Connect 连接集中器需要 DB2 版本 7.1 客户机或更新版本才能支持来自不同进程的 XA 落实序列。

还需要配置了 DB2 同步点管理器（SPM）的 DB2 Connect。

相关参考:

- 『“代理进程的最大数目”配置参数 — maxagents』（*《管理指南: 性能》*）
- 『“客户机连接的最大数目”配置参数 — max_connections』（*《管理指南: 性能》*）

手工解析不确定事务

XA 兼容事务管理器（事务处理监视器）使用类似于 DB2 事务管理器使用的两阶段落实过程。两个环境之间的主要区别是，TP 监视器提供记录和控制该事务的功能，而不是 DB2 事务管理器和事务管理器数据库来提供此功能。

当使用 XA 兼容事务管理器时，可能发生类似于 DB2 事务管理器发生的错误。类似于 DB2 事务管理器，XA 兼容的事务管理器将试图使不确定事务再同步。

若由于某种原因您不能等待事务管理器自动解决不确定事务，则您可以执行一些操作来以手工方式解决它们。此手工过程有时称为“作试探性决定”。

LIST INDOUBT TRANSACTIONS 命令（使用 WITH PROMPTING 选项）或相关的一组 API 允许您查询、落实和回滚不确定事务。另外，它还允许您除去日志记录并释放日志空间，以“忘记”通过试探方式落实或回滚的事务。

限制:

*务请极为谨慎*地使用这些命令（或相关的 API），并且只将其作为最后的手段。最好的策略是等待事务管理器驱动再同步过程。若您在其中一个参与数据库中以手工方式落实或回滚一个事务，并对另一个参与的数据库执行相反的操作，可能会遇到数据完整性问题。校正数据完整性问题要求您了解该应用程序的逻辑，标识已更改或回滚的数据，然后执行数据库的时间点恢复，或以手工方式撤销或重新执行更改。

若您不能等待事务管理器启动再同步过程，且您必须释放不确定事务占用的资源，则必须执行试探性操作。若事务管理器无法在延长期内执行再同步，且该不确定事务占用着急需的资源，则可能发生这种情况。不确定事务占用着事务管理器或资源管理器成为不可用之前与此事务相关的资源。对于数据库管理器，这些

资源包括对该事务占用的表和索引、日志空间和存储器的锁定。每个不确定事务还会递减（减一）数据库可以处理的并行事务的最大数目。而且，除非解析了所有不确定事务，否则不能进行脱机备份。

在下列情况下，需要试探性忘记功能:

- 当试探性落实或回滚事务导致日志已满情况（会在 LIST INDOUBT TRANSACTIONS 命令的输出中指示此情况）时
- 要进行脱机备份时

试探性忘记功能释放由不确定事务占用的日志空间。隐含情况为，若一个事务管理器最终对此不确定事务执行了再同步操作，则可能会作出错误的决定来落实或回滚其它资源管理器，因为在此资源管理器中没有该事务的日志记录。通常，“丢失”日志记录意味着该资源管理器已经回滚该事务。

过程:

尽管没有简单明了的方法来执行试探性操作，但下面还是提供了一些常规准则:

1. 与您要完成其所有事务的数据库连接。
2. 使用 LIST INDOUBT TRANSACTION 命令显示不确定事务。*xid* 表示全局事务标识，它与事务管理器和参与事务的其它资源管理器使用的 *xid* 完全相同。
3. 对于每个不确定事务，使用您掌握的有关该应用程序和操作环境的知识，来确定其它参与的资源管理器。
4. 确定事务管理器是否可用:
   - 若该事务管理器是可用的，且资源管理器中的不确定事务是由于在第二个落实阶段期间或更早的再同步过程期间资源管理器不可用导致的，则您应该检查该事务管理器的日志，以确定对其它资源管理器执行过什么操作。然后您应该对该数据库执行相同的操作；即使用 LIST INDOUBT TRANSACTIONS 命令，试探性落实或试探性回滚该事务。
   - 若事务管理器不可用，您将需要使用其它参与资源管理器中该事务的状态，以确定您应该执行什么操作:
     - 若其它资源管理器中至少有一个已经落实该事务，则您应该在所有资源管理器中试探性落实该事务。
     - 若其它资源管理器中至少有一个已经回滚该事务，则您应该试探性回滚该事务。
     - 若该事务在所有参与资源管理器中都处于“已准备好”（未确定）状态，则您应该试探性回滚该事务。
     - 若其它资源管理器中有一个或多个不可用，则您应该试探性回滚该事务。

要从 UNIX 或 Windows 上的 DB2 UDB 中获取不确定事务信息，连接至数据库，并发出 LIST INDOUBT TRANSACTIONS WITH PROMPTING 命令或等效 API。

对于与主机或 iSeries 数据库服务器相关的不确定事务信息，您有两个选择:

- 可以直接从主机或 iSeries 服务器获取不确定信息。

  要直接从 DB2 z/OS 和 OS/390 版获取不确定信息，调用 DISPLAY THREAD TYPE(INDOUBT) 命令。使用 RECOVER 命令来作试探性决定。要直接从 DB2 iSeries 版获取不确定信息，调用 **wrkcmtdfn** 命令。
- 可以从用来访问主机或 iSeries 数据库服务器的 DB2 Connect 服务器获取不确定信息。

  要从 DB2 Connect 服务器获取不确定信息，首先通过连接由 *spm_name* 数据库管理器配置参数的值所表示的 DB2 实例来与 DB2 同步点管理器连接。然后发出 LIST DRDA INDOUBT TRANSACTIONS WITH PROMPTING 命令来显示不确定事务并作试探性决定。

相关概念:

- 第 147 页的『两阶段落实』

相关参考:

- 『LIST INDOUBT TRANSACTIONS Command』（*Command Reference*）
- 『LIST DRDA INDOUBT TRANSACTIONS Command』（*Command Reference*）

XA 事务管理器的安全性注意事项

TP 监视器预分配一组服务器进程，并由在这些服务器进程标识下的不同用户运行事务。对于该数据库，每个服务器进程表现为一个有许多工作单元的大应用程序，所有工作单元都在与该服务器进程相关的相同标识下运行。

例如，在使用 CICS 的 AIX® 环境中，当启动 TXSeries™ CICS® 区域时，该区域与用于定义它的 AIX 用户名相关。所有 “CICS 应用程序服务器” 进程也都在 TXSeries CICS 主标识下运行，该标识通常被定义为 “cics”。CICS 用户可以在他们的 DCE 登录标识下调用 CICS 事务，且在 CICS 中时，他们还可以使用 CESN 注册事务来更改他们的标识。在任何一种情况中，RM 都无法使用最终用户的标识。因此，一个 “CICS 应用程序进程” 可能正在代表许多用户运行事务，但是，对于 RM 来说，它就象是一个带有许多工作单元的单一程序，而这些工作单元具有相同的 “cics” 标识。您可以选择在 xa_open 字符串中指定用户标识和密码，并将使用该用户标识来代替 “cics” 标识以与数据库连接。

对于静态 SQL 语句，没有很大影响，因为是使用绑定程序的特权而非最终用户的特权来存取数据库。但是，这意味着必须将数据库程序包的 EXECUTE 特权授予服务器标识，而不是最终用户标识。

对于动态语句，它们在运行时执行存取认证，必须将数据库对象的存取特权授予服务器标识，而非那些对象的实际用户。此时必须依靠 TP 监视器系统来确定允许用户运行的程序，而不是依靠数据库来控制特定用户的存取。必须授予服务器标识它的 SQL 用户需要的所有特权。

要确定谁存取了数据库表或视图，您可以执行下列步骤:

1. 从 SYSCAT.PACKAGEDEP 目录视图，获取取决于该表或视图的所有程序包的列表。
2. 通过安装期间使用的命名约定，确定与这些程序包对应的服务器程序（例如，CICS 程序）的名称。
3. 确定可能调用这些程序的客户机程序（例如，CICS 事务标识），然后使用 TP 监视器的日志（例如，CICS 日志）来确定谁运行了这些事务或程序以及是何时运行的。

相关概念:

- 第 154 页的『X/Open 分布式事务处理模型』

XA 事务管理器的配置注意事项

当您设置 TP 监视器环境时，应该考虑下列配置参数:

- *tp_mon_name*

  此数据库管理器配置参数标识正在使用的 TP 监视器产品的名称（例如，“CICS”或“ENCINA”）。

- *tpname*

  此数据库管理器配置参数标识远程事务程序的名称，当数据库客户机使用 APPC 通信协议向数据库服务器发出分配请求时，必须使用该事务程序。此值是在服务器的配置文件中设置的，且必须与在 SNA 事务程序中配置的事务处理器（TP）的名称相同。

- *tm_database*

  因为 DB2® 不会协调 XA 环境中的事务，所以不对协调 XA 的事务使用此数据库管理器配置参数。

- *maxappls*

此数据库配置参数指定允许的活动应用程序的最大数目。此参数的值必须等于或大于连接的应用程序数，加上可能正在同时完成一个两阶段落实或回滚的这些应用程序的数目之和。此和随后应加上在任何时刻可能存在的预期不确定事务数。

对于 TP 监视器环境（例如，TXSeries™ CICS），可能需要增加 *maxappls* 参数的值。这有助于确保可以处理所有 TP 监视器进程。

- *autorestart*

  此数据库配置参数指定在需要时是否自动调用 RESTART DATABASE 例程。缺省值是 YES（即，启用）。

  包含不确定事务的数据库需要重新启动数据库操作才能启动。若断开与该数据库的上一个连接时未启用 *autorestart*，则下一个连接将失败并需要显式的 RESTART DATABASE 调用。这种情况将一直存在，直到事务管理器的再同步操作或管理员启动的试探性操作除去了不确定事务为止。当发出 RESTART DATABASE 命令时，若该数据库中有任何不确定事务，将返回一条消息。管理员就可以使用 LIST INDOUBT TRANSACTIONS 命令和其它的命令行处理器命令来获取有关那些不确定事务的信息。

相关概念:

- 第 154 页的『X/Open 分布式事务处理模型』

相关参考:

- 『“自动重新启动启用”配置参数 — autorestart』（*《管理指南: 性能》*）
- 『“APPC 事务程序名称”配置参数 — tpname』（*《管理指南: 性能》*）
- 『“活动应用程序的最大数目”配置参数 — maxappls』（*《管理指南: 性能》*）
- 『“事务管理器数据库名”配置参数 — tm_database』（*《管理指南: 性能》*）
- 『“事务处理器监视器名”配置参数 — tp_mon_name』（*《管理指南: 性能》*）
- 『LIST INDOUBT TRANSACTIONS Command』（*Command Reference*）
- 『RESTART DATABASE Command』（*Command Reference*）

DB2 UDB 支持的 XA 功能

“DB2® 通用数据库”支持在 *X/Open CAE 规范分布式事务处理: XA 规范*中定义的 XA91 规范，但是下列情况除外:

- 异步服务

  XA 规范允许该接口使用异步服务，这样可以在稍后的某个时间检查请求的结果。数据库管理器要求以同步方式调用请求。

- 静态注册

XA 接口允许以两种方式来注册 RM: 静态注册和动态注册。 “DB2 通用数据库™ 只支持动态注册，此方式更高级且更有效率。

- 关联迁移

  DB2 通用数据库不支持控制的线程之间的事务迁移。

XA 开关使用情况和位置

根据 XA 接口的要求，数据库管理器提供 xa_switch_t 类型的 *db2xa_switch* 外部 C 变量，以将 XA 开关结构返回至该 TM。不是返回各种 XA 函数的地址，而是返回下列字段:

字段	值
名称	数据库管理器的产品名。例如，DB2 AIX 版。
标志	TMREGISTER | TMNOMIGRATE 明确声明 “DB2 通用数据库” 使用动态注册，且 TM 不应该使用关联迁移。明确声明不支持异步操作。
版本	必须为零。

使用 DB2 通用数据库 XA 开关

XA 体系结构要求 “资源管理器” （RM）提供*开关*，该开关使 “XA 事务管理器” （TM）可以访问 RM 的 **xa**_ 例程。RM 开关使用称为 xa_switch_t 的结构。该开关包含 RM 的名称、指向 RM 的 XA 入口点的非 NULL 指针、标志和版本号。

基于 UNIX 的系统

DB2 UDB 的开关可以通过下面两种方法之一获得:

- 通过更高一级的间接方法。在 C 程序中，可在使用 *db2xa_switch* 之前定义以下的宏来完成:

  ```
  #define db2xa_switch (*db2xa_switch)
  ```

- 通过调用 **db2xacic**

  DB2 UDB 提供了此 API，它返回 *db2xa_switch* 结构的地址。此函数的原型为:

  ```
  struct xa_switch_t * SQL_API_FN  db2xacic( )
  ```

不管使用哪种方法，必须将应用程序与 libdb2 链接。

Windows NT

指向 *xa_switch* 结构的指针 *db2xa_switch* 被作为 DLL 数据导出。这意味着使用此结构的 Windows® NT 应用程序必须以下列三种方式之一来引用它:

- 通过更高一级的间接方法。在 C 程序中，可在使用 *db2xa_switch* 之前定义以下的宏来完成:

  ```
  #define db2xa_switch (*db2xa_switch)
  ```

- 若使用 Microsoft® Visual C++ 编译器，则可以将 *db2xa_switch* 定义为:

  ```
  extern __declspec(dllimport) struct xa_switch_t db2xa_switch
  ```

- 通过调用 **db2xacic**

  DB2 UDB 提供了此 API，它返回 *db2xa_switch* 结构的地址。此函数的原型为:

  ```
  struct xa_switch_t * SQL_API_FN  db2xacic( )
  ```

使用这些方法中任何一种，必须将应用程序与 db2api.lib 链接。

C 代码示例

下列代码举例说明在任何 DB2 UDB 平台上通过 C 程序访问 *db2xa_switch* 的不同方法。务必将应用程序与适当的库连接。

```
#include <stdio.h>
#include <xa.h>

struct xa_switch_t * SQL_API_FN  db2xacic( );

#ifdef DECLSPEC_DEFN
extern __declspec(dllimport) struct xa_switch_t db2xa_switch;
#else
#define db2xa_switch (*db2xa_switch)
extern struct xa_switch_t db2xa_switch;
#endif

main( )
   {
      struct xa_switch_t *foo;
      printf ( "%s \n", db2xa_switch.name );
      foo = db2xacic();
      printf ( "%s \n", foo->name );
      return ;
   }
```

相关概念:

- 第 154 页的『X/Open 分布式事务处理模型』

XA 接口问题确定

若从 TM 发出 XA 请求期间检测到错误，则应用程序可能无法从 TM 获取错误代码。若您的程序异常终止，或从 TP 监视器或 TM 获得加密的返回代码，则应该检查“首次故障服务日志”，当诊断级别 3 或更高级别生效时，它将报告 XA 错误信息。

您还应该参考控制台消息、TM 错误文件或关于所用的外部事务处理软件的其它产品特定消息。

数据库管理器将所有 XA 特定错误写入“首次故障服务日志”，并使用 SQLCODE -998（事务或试探性错误）和适当的原因码。以下是一些更为常见的错误:

- xa_open 字符串中的语法无效。
- 由于下列原因之一，未能与打开字符串中指定的数据库连接:
  - 数据库未编目。
  - 数据库未启动。
  - 服务器应用程序的用户名或密码无权与该数据库连接。
- 通信错误。

相关概念:

- 第 154 页的『X/Open 分布式事务处理模型』

相关参考:

- 第 158 页的『xa_open 字符串格式』

XA 事务管理器配置

配置 IBM WebSphere Application Server

IBM WebSphere Application Server 是基于 Java 的应用程序服务器。它可以通过 DB2 JDBC 驱动程序提供的“Java 事务 API”（JTA）使用 DB2 的 XA 支持。参考关于如何将“Java 事务 API”与 WebSphere Application Server 配合使用的 IBM WebSphere 文档。可联机查看 WebSphere Application Server 文档，网址为 http://www-4.ibm.com/software/webservers/appserv/infocenter.html。

配置 IBM TXSeries CICS

有关如何配置 IBM TXSeries CICS 以将 DB2 用作资源管理器的信息，参考 *IBM TXSeries CICS Administration Guide*。TXSeries 文档可以通过访问如下站点联机查看：
http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/。

主机和 iSeries 数据库服务器可以参与协调 CICS 的事务。

配置 IBM TXSeries Encina

以下是当通过 DB2 Connect 访问时，将 “Encina 监视器” 与 “DB2 通用数据库” 服务器或 DB2 z/OS 版和 OS/390 版、DB2 iSeries 版或 DB2 VSE 版和 VM 版集成时需要的各种 API 和配置参数。TXSeries 文档可以通过访问如下站点联机查看：
http://www.transarc.com/Library/documentation/websphere/WAS-EE/en_US/html/。

主机和 iSeries 数据库服务器可以参与协调 Encina 的事务。

配置 DB2

要配置 DB2:

1. 必须在 DB2 数据库目录中定义每个数据库的名称。若该数据库是远程数据库，则还必须定义节点目录项。您可以使用 “配置助手” 或 DB2 命令行处理器（CLP）来执行该配置。例如:

   ```
   DB2 CATALOG DATABASE inventdb AS inventdb AT NODE host1 AUTH SERVER
   DB2 CATALOG TCPIP NODE host1 REMOTE hostname1 SERVER svcname1
   ```

2. 若 DB2 客户机知道它处理的是 Encina，则它可以针对 Encina 优化它的内部处理。您可以将 *tp_mon_name* 数据库管理器配置参数设置为 ENCINA 来指定此选项。缺省行为是不进行特殊的优化。若设置了 *tp_mon_name*，则该应用程序必须确保执行该工作单元的线程在结束之后也立即落实该工作。不能启动任何其它工作单元。若您的环境不是这样的，则确保 *tp_mon_name* 值是 NONE（或者，通过 CLP 将此值设置为 NULL）。可以通过 “控制中心” 或 CLP 更新此参数。CLP 命令是:

   ```
   db2 update dbm cfg using tp_mon_name ENCINA
   ```

为每个资源管理器配置 Encina

要为每个资源管理器（RM）配置 Encina，管理员必须为作为资源管理器的每个 DB2 数据库定义 “打开字符串”、“关闭字符串” 和 “控制线程协议”，然后才能为应用程序中的事务注册该资源管理器。可以使用 Enconcole 全屏幕界面或 Encina 命令行界面来执行该配置。例如:

```
monadmin create rm inventdb -open "db=inventdb,uid=user1,pwd=password1"
```

每个 DB2 数据库都有一个资源管理器配置，且每个资源管理器配置必须具有 rm 名称（“逻辑 RM 名称”）。为了简化情况，您应该使它与数据库名相同。

xa_open 字符串包含建立与数据库的连接所需的信息。该字符串的内容是 RM 所独有的。DB2 UDB 的 xa_open 字符串包含要打开的数据库的别名，以及与该连接相关的可选的用户标识和密码。注意，此处定义的数据库名也必须在所有数据库存取都必需的规则数据库目录中编目。

DB2 不使用 xa_close 字符串。

“控制线程协议”确定一个应用程序的代理进程线程是否一次可以处理多个事务。

如果在存取 DB2 z/OS 版和 OS/390 版、DB2 iSeries 版或 DB2 VSE 版和 VM 版，则必须使用DB2 同步点管理器。

从 Encina 应用程序中引用 DB2 数据库

要从 Encina 应用程序中引用 DB2 数据库:

1. 使用 “Encina 调度策略 API”，来指定可以从单个 TP 监视器的应用程序进程中运行多少个应用程序代理进程。例如:

   ```
   rc = mon_SetSchedulingPolicy (MON_EXCLUSIVE)
   ```

2. 使用 “Encina RM 注册 API” 来提供 XA 开关和逻辑 RM 名称，以供 Encina 在应用程序进程中引用 RM 时使用。例如:

   ```
   rc = mon_RegisterRmi ( &db2xa_switch,   /* xa switch */
                          "inventdb",      /* logical RM name */
                          &rmiId );        /* internal RM ID */
   ```

   XA 开关包含 RM 中 TM 可以调用的 XA 例程的地址，它还指定由 RM 提供的功能。DB2 通用数据库的 XA 开关是 db2xa_switch，它驻留在 DB2 客户机库（在 Windows 操作系统上是db2app.dll，在基于 UNIX 的系统上是 libdb2）中。

   逻辑 RM 名是 Encina 使用的那个名称，而不是在 Encina 下运行的 SQL 应用程序使用的实际数据库名。实际数据库名是在 “Encina RM 注册表” API 的 xa_open 字符串中指定的。逻辑 RM 名设置为与本示例中的数据库名相同。

   第三个参数返回 TM 引用此连接所用的内部标识符或句柄。

相关概念:

- 『DB2 Connect 和事务处理监视器』（*《DB2 Connect 用户指南》*）

相关参考:

- 『 “事务处理器监视器名” 配置参数 — tp_mon_name』（*《管理指南: 性能》*）
- 第 158 页的『xa_open 字符串格式』

配置 BEA Tuxedo

过程:

要配置 Tuxedo 以将 DB2 用作资源管理器，须执行下列步骤:

1. 按该产品的文档中指定的步骤，安装 Tuxedo。确保执行所有基本的 Tuxedo 配置，包括日志文件和环境变量。

   您还需要一个编译程序和 DB2 应用程序开发客户机。需要时安装它们。

2. 在 Tuxedo 服务器标识中，设置 DB2INSTANCE 环境变量，以引用包含您希望 Tuxedo 使用的数据库的实例。将 PATH 变量设置为包括 DB2 程序目录。确认 Tuxedo 服务器标识可以与 DB2 数据库连接。
3. 使用值 TUXEDO 来更新 *tp_mon_name* 数据库管理器配置参数。
4. 将 DB2 的定义添加至 Tuxedo 资源管理器的定义文件。在下面的示例中，UDB_XA 是为 DB2 在本地定义的 Tuxedo 资源管理器名，而 *db2xa_switch* 是类型为 xa_switch_t 的一个结构的 DB2 定义的名称:

   - 对于 AIX。在文件 ${TUXDIR}/udataobj/RM 中，添加定义:

     ```
     # DB2 UDB
     UDB_XA:db2xa_switch:-L${DB2DIR} /lib -ldb2
     ```

     其中，{TUXDIR} 是安装了 Tuxedo 的目录，而 {DB2DIR} 是 DB2 实例目录。

   - 对于 Windows NT。在文件 %TUXDIR%\udataobj\rm 中，添加定义:

     ```
     # DB2 UDB
     UDB_XA;db2xa_switch;%DB2DIR%\lib\db2api.lib
     ```

     其中，%TUXDIR% 是安装了 Tuxedo 的目录，而 %DB2DIR% 是 DB2 实例目录。

5. 为 DB2 构建 Tuxedo 事务监视器服务器程序:

   - 对于 AIX:

     ```
     ${TUXDIR}/bin/buildtms -r UDB_XA -o ${TUXDIR}/bin/TMS_UDB
     ```

     其中，{TUXDIR} 是安装了 Tuxedo 的目录。

   - 对于 Windows NT:

     ```
     %TUXDIR%\bin\buildtms -r UDB_XA -o %TUXDIR%\bin\TMS_UDB
     ```

6. 构建应用程序服务器。在下面的示例中，-r 选项指定资源管理器名，-f 选项（使用了一次或多次）指定包含应用程序服务的文件，-s 选项指定此服务器的应用程序服务名，而 -o 选项指定输出服务器文件名:

   - 对于 AIX:

     ```
     ${TUXDIR}/bin/buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
        -o UDBserver
     ```

     其中，{TUXDIR} 是安装了 Tuxedo 的目录。

   - 对于 Windows NT:

     ```
     %TUXDIR%\bin\buildserver -r UDB_XA -f svcfile.o -s SVC1,SVC2
        -o UDBserver
     ```

其中，%TUXDIR% 是安装了 Tuxedo 的目录。

7. 设置 Tuxedo 配置文件以引用 DB2 服务器。在 UDBCONFIG 文件的 *GROUPS 小节中，添加类似如下的条目:

```
UDB_GRP   LMID=simp GRPNO=3
  TMSNAME=TMS_UDB TMSCOUNT=2
  OPENINFO="UDB_XA:db=sample,uid=db2_user,pwd=db2_user_pwd"
```

   其中，TMSNAME 参数指定您先前构建的事务监视器的服务器程序，而 OPENINFO 参数指定资源管理器名。其后是数据库名和 DB2 用户和密码（用于认证）。

   您先前构建的应用程序服务器在 Tuxedo 配置文件的 *SERVERS 一节中被引用。

8. 若应用程序正在存取驻留在 DB2 z/OS 版和 OS/390 版、DB2 iSeries 版或 DB2 VM 版和 VSE 版上的数据，则将需要 DB2 Connect XA 集中器。

9. 启动 Tuxedo:

```
tmboot -y
```

   在该命令完成之后，Tuxedo 消息应指示服务器已启动。另外，若您发出 DB2 命令 LIST APPLICATIONS ALL，您应该看到两个连接（在这种情况下），它们由 Tuxedo 配置文件 UDBCONFIG 中的 UDB 组的 TMSCOUNT 参数指定。

相关概念:

- 『DB2 Connect 和事务处理监视器』（*《DB2 Connect 用户指南》*）

相关参考:

- 『“事务处理器监视器名”配置参数 — tp_mon_name』（*《管理指南: 性能》*）
- 『LIST APPLICATIONS Command』（*Command Reference*）

第 3 部分 附录

附录 A. 发行版之间的不兼容性

本节标识存在于 DB2 通用数据库和 DB2 的先前发行版之间的不兼容性。

*不兼容性*是DB2 通用数据库的一部分，行为方式与它在先前发行版的 DB2 中的行为方式不同。若在现有应用程序中使用，将生成预期不到的结果，使更改应用程序成为必要，或降低性能。在本上下文中，“应用程序”是指:

- 应用程序代码
- 第三方实用程序
- 交互式 SQL 查询
- 命令或 API 调用

下面描述 DB2 通用数据库版本 7 和版本 8 带来的不兼容性。它们根据下列类别分组:

- 系统目录信息
- 应用程序编程
- SQL
- 数据库安全性和调整
- 实用程序和工具
- 连通性和共存
- 消息
- 配置参数

每个不兼容小节都包括对不兼容性的描述、不兼容性的症状或造成的影响，以及可能的解决方案。每个不兼容性描述的开头都有一个指示符，标识不兼容性适用的操作系统:

WIN DB2 支持的 Microsoft Windows 平台

UNIX DB2 支持的基于 UNIX 的平台

OS/2 OS/2（仅适用于版本 7）

DB2 通用数据库计划不兼容性

本节描述“DB2 通用数据库”用户在编码新应用程序或修改现有应用程序时应该记住的将来的不兼容性。这将简化面向将来版本的 DB2 UDB 的迁移。

系统目录信息

将来版本的 DB2 通用数据库中的 PK_COLNAMES 和 FK_COLNAMES

WIN	UNIX

更改: SYSCAT.REFERENCES 的列 PK_COLNAMES 和 FK_COLNAMES 不再可用。

症状: 列不存在，并返回错误。

说明: 编码的工具或应用程序使用过时的 PK_COLNAMES 和 FK_COLNAMES 列。

解决方案: 更改该工具或应用程序，以改用 SYSCAT.KEYCOLUSE 视图。

将来版本的 DB2 通用数据库中的 COLNAMES 不再可用

WIN	UNIX

更改: SYSCAT.INDEXES 的列 COLNAMES 不再可用。

症状: 列不存在，并返回错误。

说明: 编码的工具或应用程序使用过时的 COLNAMES 列。

解决方案: 更改该工具或应用程序，以改用 SYSCAT.INDEXCOLUSE 视图。

实用程序和工具

将除去对类型 1 索引的重新创建的支持

WIN	UNIX

更改: 在版本 8 中介绍了新类型的索引，称为类型 2 索引。在类型 1 索引中，索引是在版本 8 之前创建的，作为表行的删除或更新的一部分，已从叶子页中物理除去了键。在 类型 2 中，当删除或更新行时，键标记为已删除，但在落实删除或更新前不会物理删除它们。当除去对类型 1 索引的重新创建的支持后，将不必手工重新构建索引。类型 1 索引将继续正常工作。由索引的重新创建导致的所有操作将自动将类型 1 索引转换为类型 2 索引。在将来的版本中，将除去对类型 1 索引的支持。

说明: 类型 2 索引与类型 1 索引相比较，具有以下优点:

- 可对长度大于 255 字节的列创建类型 2 索引。
- 将对下一个键锁定的使用减至最小，以提高并行性。

解决方案: 开发一个计划以在一定时间内将现有索引转换为类型 2 索引。当最小化可用性停止时，“联机索引重组”功能可以帮助完成此操作。如果需要的话，增加索引表空间大小。考虑在大表空间中创建新的索引，并将现有索引移至大表空间。

发行版之间的版本 8 不兼容性

系统目录信息

目录表中的 IMPLEMENTED 列

WIN	UNIX

更改: 在先前版本中，SYSIBM.SYSFUNCTIONS 和 SYSCAT.SYSFUNCTIONS 中的 IMPLEMENTED 列具有值 Y、M、H 和 N。在版本 8 中，值是 Y 和 N。

解决方案: 将应用程序程序编码以仅使用值 Y 和 N。

OBJCAT 视图重命名为 SYSCAT 视图

WIN	UNIX

更改: 已经将下列 OBJCAT 视图重命名为 SYSCAT 视图：TRANSFORMS、INDEXEXTENSIONS、INDEXEXTENSIONMETHODS、INDEXEXTENSIONDEP、INDEXEXTENSIONPARMS、PREDICATESPECS 和 INDEXEXPLOITRULES。

解决方案: 将应用程序程序编码以使用 SYSCAT 视图。

SYSCAT 视图现在是只读的

WIN	UNIX

更改: 对于版本 8，SYSCAT 视图是只读的。

症状: 对 SYSCAT 模式中的视图的 UPDATE 或 INSERT 操作现在失败。

说明: 建议使用 SYSSTAT 视图来更新系统目录信息。某些 SYSCAT 视图可能在无意中被设置为可更新，现在已得到修正。

解决方案: 更改应用程序来引用可更新的 SYSSTAT 视图作为替代。

应用程序编程

使用 VERSION 选项时，未返回 SQL0818N 错误

WIN	UNIX

更改: 如果在 PRECOMPILE、BIND、REBIND 和 DROP PACKAGE 命令上使用 VERSION 选项，则要执行的请求现在可能返回 SQL0805N 错误而不是 SQL0818N 错误。

症状: 编码为对 SQL0818N 错误作出反应的应用程序的行为可能与以前不同。

解决方案: 将应用程序重新编码以对 SQL0805N 和 SQL0818N 错误作出反应。

未定义主机变量时，不会对预编译器返回 SQL0306N 错误

WIN	UNIX

更改: 如果未在 BEGIN DECLARE 部分声明主机变量并且在 EXEC SQL 部分中使用它，则预编译器不会返回 SQL0306N。如果变量是在应用程序中的其它位置声明的，则应用程序运行时将返回 SQL0804N。如果未在应用程序中的任何位置声明该变量，则编译器会在编译时返回错误。

症状: 编码为在预编译时对 SQL0306N 错误作出反应的应用程序的行为可能与以前不同。

解决方案: 应在 BEGIN DECLARE 部分声明主机变量。如果主机变量是在不同于 BEGIN DECLARE 部分的部分中声明的，则应将应用程序重新编码以处理 SQL0804 返回代码。

与可滚动游标配合使用时不受支持的数据类型

WIN	UNIX

更改: 在版本 8 中，不支持使用 LONG VARCHAR、LONG VARGRAPHIC、DATALINK 和 LOB 类型以及上述任何一种类型的单值类型或结构类型的可滚动游标。不再支持版本 7 的可滚动游标的上述任何受支持的数据类型。

症状: 如果在可滚动游标的选择列表中指定了具有这些数据类型的任何列，将返回 SQL0270N 原因代码 53。

解决方案: 修改可滚动游标的选择列表，以便该列表不包括具有上述任何一种类型的列。

代码页转换表的欧洲版本

WIN	UNIX

更改: 版本 8 代码页转换表提供对欧元符号的支持，这与随 DB2 的先前版本提供的转换表稍有不同。

解决方案: 如果想要使用版本 8 之前的代码页转换表，它们在目录 `sqllib/conv/v7` 中。

在 LOB 定位器和 LOB 值之间切换

WIN	UNIX

更改: 在游标语句的向外绑定期间已经更改了大对象（LOB）定位器和 LOB 值之间切换的能力。当应用程序与 SQLRULES DB2 绑定在一起时（缺省行为），用户将不能在 LOB 定位器和 LOB 值之间切换。

解决方案: 如果想要在游标语句的向外绑定期间在 LOB 定位器和 LOB 值之间切换，则使用 SQLRULES STD 预编译应用程序。

UNIX 平台上的未落实工作单元

	UNIX

更改: 在版本 8 前，如果未使用显式或隐式上下文支持的基于 UNIX 的应用程序正常终止而不直接调用 CONNECT RESET、COMMIT 或 ROLLBACK 语句，该应用程序将落实未完成工作单元。如果 CLI、ODBC 和基于 Java 的应用程序（隐式上下文支持）和显式创建应用程序上下文的应用程序终止，则它们总是回滚未完成的工作单元。异常应用程序终止还将导致未完成工作单元的隐式回滚（ROLLBACK）。在版本 8 中，所有应用程序终止将隐式回滚未完成工作单元。基于 Windows 的应用程序将不会更改，因为它们已经对正常或异常应用程序终止执行了隐式回滚。

解决方案: 为了确保落实事务，应用程序在终止前应执行显式 COMMIT 或 CONNECT RESET。

对保存点命名的更改

WIN	UNIX

更改: 保存点名称不能再以“SYS”开始。

症状: 以“SYS”开始的名称创建保存点将会失败，返回错误 SQL0707N。

说明: 以“SYS”开始的保存点名称是保留为系统使用的。

解决方案: 将以“SYS”开始的所有保存点重命名为不以“SYS”开始的另一名称。

代码页转换错误和字节替换

WIN	UNIX

更改: 现在代码页转换将在绑定阶段执行（如果需要的话）。

症状: 代码页转换错误和字节替换可在新的情况下发生。例如，以下语句现在将主机变量 :hv 中的数据从应用程序代码页转换为数据库代码页（如果这些代码页不相同的话）：

```
SELECT :hv FROM table
VALUES :hv
```

说明: 没有错误或字节替换，不能将提供的数据转换为数据库代码页。先前未进行代码页转换，现在执行代码页转换。

解决方案: 更改数据、应用程序代码页或数据库代码页，以使代码页转换不会产生错误或字节替换，或编码应用程序以处理可能的代码页转换错误或字节替换。

主机变量的代码页转换

WIN	UNIX

更改: 现在代码页转换将在绑定阶段执行（如果需要的话）。

症状: 不同的结果。

说明: 既然总是会执行主机变量的代码页转换（必要时），谓词求值将总是在数据库代码页中进行，而不是在应用程序代码页中进行。例如，

```
SELECT * FROM table WHERE :hv1 > :hv2
```

将使用数据库代码页执行，而不是应用程序代码页。使用的整理仍然是数据库整理。

解决方案: 验证先前版本中的结果是否为真正想要的结果。如果答案是肯定的，则在使用数据库整理和代码页时，更改谓词以产生想要的结果。或者更改应用程序代码页或数据库代码页。

主机变量中数据的扩展和压缩

WIN	UNIX

更改: 现在代码页转换将在绑定阶段执行（如果需要的话）。

症状: 来自主机变量的数据的长度不同。

说明: 因为扩展或压缩可能在代码页转换期间发生，取决于主机变量中数据长度的操作可能产生不同的结果或错误情况。

解决方案: 更改数据、应用程序代码页或数据库代码页，以使代码页转换不产生转换的数据长度的更改，或编码应用程序以处理代码页转换可能导致的数据长度更改。

代码页转换后主机变量的长度

WIN	UNIX

更改: 代码页转换将不再因为扩充而导致主机变量或参数标记符的长度增加。

症状: 数据截断错误。

说明: 不再因为代码页转换的潜在扩充而增加对隐式类型参数标记确定的字符数据类型的长度。对于使用隐式类型参数标记的长度确定结果长度的操作，结果长度将更短。例如，如果 C1 是 CHAR(10) 列:

```
VALUES CONCAT (?, C1)
```

对于从应用程序代码页向数据库代码页转换时可以扩充 3 倍的数据库，结果数据类型和长度将不再是 CHAR(40)，而是结果数据类型和长度为 CHAR(20)。

解决方案: 使用 CAST 以对隐式类型参数标记指定想要的类型，或将确定隐式类型参数标记的类型的操作数更改为可容纳因代码页转换而造成的数据扩充的数据类型或长度。

对 DESCRIBE 语句的输出的更改

WIN	UNIX

更改: 代码页转换将不再因为扩充而导致主机变量或参数标记符的长度增加。

症状: 来自 DESCRIBE 语句的输出将会更改。

说明: 因为结果长度不会因代码页转换的潜在扩充而增加，描述这一结果长度的 DESCRIBE 语句的输出现在将会不同。

解决方案: 必要时更改应用程序以处理 DESCRIBE 语句返回的新值。

将 SUBSTR 函数与主机变量配合使用时出错

WIN	UNIX

更改: 代码页转换将不再因为扩充而导致主机变量或参数标记符的长度增加。

症状: SUBSTR 导致错误 SQL0138N。

说明: 考虑由于代码页转换导致的潜在扩充，方法是增加为主机变量保留的长度。这是允许的，例如，对长度为 10 的主机变量成功执行的 SUBSTR（:hv,19,1）。它将不再有效。

解决方案: 增加主机变量的长度以备转换数据的长度所用，或更改 SUBSTR 调用以指定主机变量长度内的位置。

Solaris 不再支持非线程安全库

	UNIX

更改: 非线程安全库 libdb2_noth.so 不再可用。

症状: 需要 libdb2_noth.so 的工具或应用程序将不再有效。

说明: 因为不再需要支持废弃的非线程安全库，DB2 UDB Solaris 版不包含 libdb2_noth.so 库。

解决方案: 更改工具或应用程序，以使用线程安全的 libdb2.so 库。使用 -mt 参数重新链接应用程序。

SQL

不允许函数和过程有相同特定名称

WIN	UNIX

更改: 已统一 SPECIFICNAME 的名称空间。先前版本的 DB2 将允许函数和过程具有相同的特定名称，但是版本 8 不允许这样做。

症状: 如果在将数据库迁移至版本 8，则 db2ckmig 实用程序将检查函数和过程是否具有相同特定名称。如果在迁移过程中遇到重复的名称，则迁移将会失败。

解决方案: 删除过程并以另一特定名称重新创建该过程。

对函数和过程的 EXECUTE 特权

WIN	UNIX

更改: 以前，用户只需创建例程以使其它用户能够使用它。现在，在创建例程后，用户必须对其授予执行权限（GRANT EXECUTE），其它用户才能使用它。

在先前版本中，对于过程没有权限检查，但是调用者必须具有对从该过程调用的任何程序包具有 EXECUTE 特权。在版本 8 中，对于使用 CALL_RESOLUTION IMMEDIATE 预编译的嵌入式应用程序，以及对于 CLI 编目的过程，调用者必须具有对过程的 EXECUTE 特权，而只有该过程的定义者才必须具有对所有程序包的 EXECUTE 特权。

症状:

1. 应用程序可能不能正常工作。
2. 由多个程序包组成的现有过程（并且过程的定义者对其不具有对所有程序包的存取权）将不能正常工作。

解决方案:

1. 发出必需的 GRANT EXECUTE 语句。如果所有的例程都在单个模式中，则可以使用单个语句授予对每种类型的例程的特权，例如:

   ```
   GRANT EXECUTE ON FUNCTION schema1.* TO PUBLIC
   ```

2. 如果一个程序包对每个用户都可用，但是另一程序包被限制为只有几个有特权的用户可以使用，则使用这两个程序包的存储过程在尝试存取第二个程序包时，将会发现有在权限错误。如果它看到权限错误，则它将知道该用户不是特权用户并且该过程将绕过它的逻辑部分。

   可以以几种方式解决这种情况:

a. 当预编译程序时，应设置 CALL_RESOLUTION DEFERRED 以指示当预编译器无法解析 CALL 语句上的过程时，程序将作为废弃的 sqleproc() API 的调用来执行。
b. 可以将 CLI 关键字 UseOldStpCall 添加至 db2cli.ini 文件以控制调用过程的方式。它可以有两个值：值 *0* 意味着将不会使用旧的调用方法来调用过程，而值 *1* 意味将使用旧的调用方法调用过程。
c. 对执行程序包的所有用户授予 EXECUTE 特权。

对表添加外键约束

WIN	UNIX

更改: 在先前版本中，如果创建引用处于检查暂挂状态的表的外键约束，则还将使从属表处于检查暂挂状态。在版本 8 中，如果创建引用处于检查暂挂状态的表的外键约束，则有两种可能的结果:

1. 如果在创建从属表时添加外键约束，则表的创建和约束的添加操作将会成功，因为表创建时将为空，所以没有违反约束的行。
2. 如果向现有表添加外键，则将接收到错误 SQL0668N。

解决方案: 在添加引用表的外键前，使用 SET INTEGRITY ... IMMEDIATE CHECKED 语句来打开对于处于检查暂挂状态的表的完整性检查。

更改为 SET INTEGRITY ... IMMEDIATE CHECKED

WIN	UNIX

更改: 在先前发行版中，对其发出 SET INTEGRITY ... UNCHECKED 语句的表（ 即，SYSCAT.TABLES 的 const_checked 列带有一些 “U” 字节）缺省情况下将在执行下一个 SET INTEGRITY ... IMMEDIATE CHECKED 语句时得到完全处理，这意味着将检查所有记录的约束违规情况。必须显式指定 INCREMENTAL 以避免完全处理。

在版本 8 中，当发出 SET INTEGRITY ... IMMEDIATE CHECKED 语句时，缺省情况是仅进行增量处理，而让未检查数据保持原样（即，保留 “U” 字节）。（将返回警告，指示旧数据保持未验证状态。）

说明: 进行此更改以避免缺省行为（对所有记录执行约束检查），这通常会消耗较多的资源。

解决方案: 您将必须显式指定 NOT INCREMENTAL 以强制执行完全处理。

CHAR 函数的十进制分隔符

WIN	UNIX

更改: 在语言环境使用逗号作为十进制分隔符且 CHAR 函数的非限定调用包括类型为 REAL 或 DOUBLE 的自变量的服务器上运行的动态应用程序将在 CHAR(double) 函数的结果中返回句号作为分隔符。在版本 8 中重新创建象视图和触发器这样的对象或在显式重新绑定静态程序包时，也会出现这种不兼容性。

说明: 这是解析新 SYSIBM.CHAR(double) 函数特征符而不是 SYSFUN.CHAR(double) 特征符的结果。

解决方案: 要维护来自较早版本的 DB2 的行为，应用程序将需要使用 SYSFUN.CHAR 显式调用函数而不是允许函数解析选择 SYSIBM.CHAR 特征符。

对 CALL 语句的更改

WIN	UNIX

更改: 在版本 8 中，使用 CALL_RESOLUTION IMMEDIATE 预编译的应用程序和 CLI 编目的过程与先前版本相比较有几个关键不同之处:

- 主机变量支持已替换为对动态 CALL 的支持。
- 已除去对调用未编目存储过程的应用程序的编译的支持。在将来版本的 DB2 中，将完全除去对未编目存储过程的支持。
- 已经取消对可变自变量列表存储过程的支持。
- 装入存储过程库有几个不同的规则。

解决方案: 在版本 8 之前受支持的 CALL 语句仍然可用，并且可以在 PRECOMPILE PROGRAM 命令上使用 CALL_RESOLUTION DEFERRED 选项来进行存取。

现有应用程序（在版本 8 前构建的）将继续工作。如果重新预编译应用程序时未使用 CALL_RESOLUTION DEFERRED 选项，则可能需要更改源代码。

在将来的版本中，将除去对 CALL_RESOLUTION DEFERRED 语句的支持。

来自 UDF 的输出返回定长字符串

WIN	UNIX

更改: 可将 UDF（标量或表函数）定义为返回定长字符串（CHAR(n) 或 GRAPHIC(n)）。在先前版本中，如果返回的值包含嵌入式 null 字符，则结果将仅为 n 字节（式对于 GRAPHIC 数据类型为 2n 字节）,包括 null 字符和 null 字符右边的所有字节。在版本 8 中，DB2 查找 null 字符并从该位置（null 字符）到值的结束返回空格。

解决方案: 如果想要继续版本 8 之前的行为，则将 CHAR(n) 返回的值的定义更改为 CHAR(n) FOR BIT DATA。对于 GRAPHIC 数据，没有任何方法可继续执行版本 8 之前的行为。

数据库连接行为中的更改

WIN	UNIX

更改: 在版本 7 中，如果使用嵌入式 SQL 连接至数据库，然后试图连接至不存在的数据库，则试图连接至不存在的数据库将会失败，返回 SQL1013N。与第一个数据库的连接仍然存在。在版本 8 中，试图连接至不存在的数据库将导致与第一个数据库断开连接。这将导致应用程序处于没有任何连接的状态。

解决方案: 编码嵌入式 SQL 以在与另一数据库的连接尝试不成功后重新连接至初始数据库。

取消对程序包的控制特权（CONTROL）

WIN	UNIX

更改: 用户可以使用 CONTROL 特权授予对程序包的特权。在 DB2 版本 8 中，WITH GRANT OPTION 提供机制以确定用户将对包的特权授予其它用户的权限。使用此机制代替 CONTROL 以确定用户是否能够将特权授予其它用户。取消 CONTROL 特权时，用户将仍然能够将特权授予其它用户。

症状: 用户在取消 CONTROL 特权之后，仍能授予对程序包的特权。

解决方案: 如果用户应不再有权将对程序包的特权授予其它用户，则取消对程序包的所有特权并仅授予必需的特权。

将 FOR BIT DATA 字符串强制转型为 CLOB 时出错

WIN	UNIX

更改: 将定义为 FOR BIT DATA 的字符串强制转型为 CLOB（使用 CAST 规范或 CLOB 函数）现在返回错误（SQLSTATE 42846）。

症状: 强制转型为 CLOB 现在返回错误，而先前不会。

说明: CLOB 数据类型不支持 FOR BIT DATA。未定义在将 FOR BIT DATA 字符串指定为自变量时使用 CAST 规范或 CLOB 函数的结果。此情况现在作为错误捕获。

解决方案: 将自变量更改为 CAST 规范或 CLOB 函数，这样它就不是 FOR BIT DATA 字符串。可通过使用 CAST 规范将 FOR BIT DATA 字符串强制转型为 FOR SBCS DATA 字符串或 FOR MIXED DATA 字符串来完成此操作。例如，在非 DBCS 数据库中，若 C1FBD 是声明为 FOR BIT DATA 的 VARCHAR(20) 列，则以下将是 CLOB 函数的有效自变量:

```
CAST (C1FBD AS VARCHAR(20) FOR SBCS DATA)
```

数据库安全性和调节

对 CREATE FUNCTION、CREATE METHOD 和 CREATE PROCEDURE 语句的权限

WIN	UNIX

更改: 在版本 8 中引入了 CREATE_EXTERNAL_ROUTINE 权限。

症状: 带有 EXTERNAL 选项的 CREATE FUNCTION、CREATE METHOD 和 CREATE PROCEDURE 语句可能会失败。

解决方案: 对发出带有 EXTERNAL 选项的 CREATE FUNCTION、CREATE METHOD 和 CREATE PROCEDURE 语句的用户授予 CREATE_EXTERNAL_ROUTINE 权限。

实用程序和工具

不支持低级 CREATE DATABASE 和 DROP DATABASE

WIN	UNIX

更改: 在版本 8 中，不支持来自下级客户机或向下级服务器发出的 CREATE DATABASE 和 DROP DATABASE 命令。

症状: 发出其中一个命令时，将接收到错误 SQL0901N。

说明: 仅支持将 CREATE DATABASE 和 DROP DATABASE 命令从版本 8 客户机发送至版本 8 服务器。不能从版本 6 或版本 7 客户机向版本 8 服务器发出这些命令。不能从版本 8 客户机向版本 7 服务器发出这些命令。

解决方案: 从版本 8 客户机创建或删除版本 8 数据库。从版本 6 或版本 7 客户机创建或删除版本 7 数据库。

装入后方式更改为表

WIN	UNIX

更改: 在先前的版本中，对于使用 INSERT 选项装入且具有即时具体查询表（又称为总结表）的表，在对其执行后继 SET INTEGRITY IMMEDIATE CHECKED 语句之后，该表将处于“常规”（完整存取）状态。在版本 8 中，表在执行 SET INTEGRITY IMMEDIATE CHECKED 语句后将处于“无数据移动”方式。

说明: 对处于“无数据移动”方式的表的存取与对处于“常规”（完整存取）方式的表的存取非常相似，但涉及该表内部的数据移动的一些语句和实用程序除外。

解决方案: 可以通过对基本表发出 SET INTEGRITY ... IMMEDIATE CHECKED FULL ACCESS 语句来强制已经装入且具有从属即时总结表的基本表绕过“无数据移动”方式并直接进入“完整存取”方式。但是，建议不要使用此选项，因为它将强制从属即时具体查询表（又称为总结表）的完全刷新。

处于插入或替换方式的装入实用程序

WIN	UNIX

更改: 在先前版本中，当使用处于插入或替换方式的装入实用程序时，若完整性检查关闭，则缺省选项是 CASCADE IMMEDIATE；当表置于检查暂挂状态时，还会立即将该表的所有从属外键表和从属具体查询表（又称为总结表）置于检查暂挂状态。

对于版本 8，当使用处于插入或替换方式的装入实用程序时，如果已经关闭了完整性检查，则缺省值是 CASCADE DEFERRED。

解决方案: 可以使用 LOAD 命令的 CHECK PENDING CASCADE IMMEDIATE 选项来将从属外键表和从属具体查询表以及它们的父表一起置于检查暂挂状态。

连接和共存

下级服务器支持

WIN	UNIX

更改: 在将环境从版本 7 移至版本 8 时，如果在将所有服务器迁移至版本 8 之前将客户机迁移至版本 8，则有几个限制。这些限制与 DB2 Connect 无关；也与 zSeries、OS/390 或iSeries 数据库服务器无关。

解决方案: 对于版本 8 客户机使用版本 7 服务器的情况，您需要在版本 7 服务器上配置／启用 “DRDA 应用程序服务器” 功能。有关如何完成此任务的信息，参考版本 7 *《安装和配置补遗》*。

为避免已知限制，在将任何客户机迁移至版本 8 之前，应将所有服务器迁移至版本 8。如果做不到这一点，则您应该知道，当从版本 8 客户机访问版本 7 服务器时，将不支持下列各项:

- 某些数据类型:
  - 大对象（LOB）数据类型。
  - 用户定义单值类型（UDT）。
  - DATALINK 数据类型。

    DATALINK 数据类型允许管理非关系存储器中的外部数据。DATALINK 数据类型引用物理上驻留在 “DB2 通用数据库” 外部的文件系统上的文件。
- 某些安全性功能:
  - 认证类型 SERVER_ENCRYPT。

    SERVER_ENCRYPT 是用来加密密码的方法。加密密码与用户标识配合使用以认证用户。
  - 更改密码。

    您不能从版本 8 客户机更改版本 7 服务器上的密码。
- 某些连接和通信协议:
  - 需要 ATTACH 而不是需要连接的实例请求。

    不支持从版本 8 客户机向版本 7 服务器执行 ATTACH。
  - 唯一受支持的网络协议是 TCP/IP。

    不支持其它网络协议，如 SNA、NetBIOS、IPX/SPX 及其它协议。
- 某些应用程序功能部件和任务:
  - 不支持 DESCRIBE INPUT 语句，但 ODBC/JDBC 应用程序例外。

为支持运行 ODBC/JDBC 应用程序的版本 8 客户机访问版本 7 服务器，对所有需要此类型访问的版本 7 服务器，必须应用对 DESCRIBE INPUT 支持修订。此修订与 APAR IY30655 相关联，将在“版本 8 一般可用性”日期之前提供。使用任何“DB2 通用数据库”文档中的“与 IBM 联系”信息来了解如何获取与 APAR IY30655 相关联的修订。

DESCRIBE INPUT 语句是性能和可用性的增强，允许应用程序请求器在已准备语句中获取输入参数标记的描述。对于 CALL 语句，这包括与存储过程的 IN 和 INOUT 参数相关联的参数标记。

– 两阶段落实。

当使用涉及版本 8 客户机的协调事务时，不能将版本 7 服务器用作事务管理器数据库。版本 7 服务器也不能参与版本 8 服务器可能在其中充当事务管理器数据库的协调事务。

– XA 兼容事务管理器。

使用版本 8 客户机的应用程序不能将版本 7 服务器用作 XA 资源。这包括事务管理安排中的 WebSphere、Microsoft COM+/MTS、BEA WebLogic 及其它。

– 监视。

– 实用程序。

当客户机为版本 8 而服务器为版本 7 时，不支持那些可由客户机对服务器启动的实用程序。

– 大小超过 32 KB 的 SQL 语句。

除了对使用版本 7 服务器的版本 8 客户机的上述限制之外，对使用版本 7 服务器的版本 8 工具也存在类似的限制。

下列版本 8 工具仅支持版本 8 服务器:

- 控制中心
- 任务中心
- 日志
- 卫星管理中心
- 信息目录中心（包括此中心的 Web 版本）
- 健康中心（包括此中心的 Web 版本）
- 许可证中心
- Spatial Extender
- 工具设置

下列版本 8 工具支持版本 7 服务器（带有某些限制）和版本 8 服务器:

- 配置助手（此工具具有不同的组件，其中仅导入／导出配置文件可与版本 7 服务器配合使用；所有组件都可与版本 8 一起使用）
- 数据仓库中心
- 复制中心
- 命令中心（包括此中心的 Web 版本）
- SQL 助手
- 开发中心
- Visual Explain

一般地，仅从“控制中心”的导航树内启动的任何版本 8 工具，或基于这些工具的任何细节视图，对版本 7 和较早版本的服务器都将是不可用或不可访问的。当使用版本 7 或较早版本的服务器时，应考虑使用版本 7 工具。

可滚动游标支持

WIN	UNIX

更改: 在版本 8 中，对于从版本 8 DB2 UDB Unix 版和 Windows 版客户机至版本 7 DB2 UDB Unix 版和 Windows 版服务器，不支持可滚动游标功能。仅对于从版本 8 DB2 UDB Unix 版和 Windows 版客户机至 DB2 UDB Unix 版和 Windows 版的版本 8 服务器或至 DB2 UDB z/OS 版和 OS/390 版的版本 7 服务器，才支持可滚动游标。DB2 UDB Unix 版和 Windows 版的版本 7 客户机将仍然支持至版本 8 DB2 UDB Unix 版和 Windows 版服务器的现有可滚动游标功能。

解决方案: 将服务器升级为版本 8。

通过 DB2 Connect 版本 8 服务器访问版本 7 服务器

WIN	UNIX

更改: 在版本 8 中，将不支持通过版本 8 服务器从 DB2 UDB Unix 版和 Windows 版客户机访问版本 7 DB2 UDB 服务器，其中该功能是由“DB2 Connect 企业版的版本 8”或“DB2 UDB 企业服务器版的版本 8”提供的。

解决方案: 将服务器升级为版本 8。

消息

返回 DB2 Connect 消息而不是 DB2 消息

WIN	UNIX

更改: 在版本 8 中，在先前发行版中应返回 DB2 消息的情况现在可能返回 DB2 Connect 消息。

示例:

- 将返回 SQLCODE -30081 而不是返回 SQLCODE -1224
- 将返回 SQLCODE -30082 而不是返回 SQLCODE -1403
- 将返回 SQLCODE -30104 而不是返回 SQLCODE -4930

症状: 编码为对 DB2 消息作出反应的应用程序的行为可能与以前不同。

配置参数

废弃的数据库管理器配置参数

WIN	UNIX

更改: 下列数据库管理器配置参数已废弃:

- *backbufsz*: 在先前版本中，可以使用缺省缓冲区大小执行备份操作，且 *backbufsz* 的值将被用作缺省值。在版本 8 中，当您使用备份实用程序时，应显式指定备份缓冲区的大小。
- *dft_client_adpt*: 不再支持 DCE 目录服务
- *dft_client_comm*: 不再支持 DCE 目录服务
- *dir_obj_name*: 不再支持 DCE 目录服务
- *dir_path_name*: 不再支持 DCE 目录服务
- *dir_type*: 不再支持 DCE 目录服务
- *dos_rqrioblk*
- *drda_heap_sz*
- *fcm_num_anchors*、*fcm_num_connect* 和 *fcm_num_rqb*: DB2 现在将动态并自动调整消息锚点、连接条目和请求块，所以您不必调整这些参数。
- *fileserver*: 不再支持 IPX/SPX
- *initdari_jvm*: Java 存储过程现在在缺省情况下将以多线程方式运行，并在不同于其它语言例程的进程中运行，因此此参数不再受支持。

- *ipx_socket*: 不再支持 IPX/SPX
- *jdk11_path*: 已被 *jdk_path* 数据库管理器配置参数替换
- *keepdari*: 已被 *keepfenced* 数据库管理器配置参数替换
- *max_logicagents*: 已被 *max_connections* 数据库管理器配置参数替换
- *maxdari*: 已被 *fenced_pool* 数据库管理器配置参数替换
- *num_initdaris*: 已被 *num_initfenced* 数据库管理器配置参数替换
- *objectname*: 不再支持 IPX/SPX
- *restbufsz*: 在先前版本中，可以使用缺省缓冲区大小执行复原操作，并且可将 *restbufsz* 的值用作缺省值。在版本 8 中，当使用复原实用程序时，应显式指定复原缓冲区的大小。
- *route_obj_name*: 不再支持 DCE 目录服务
- *ss_logon*: 这是 OS/2 参数，并且不再支持 OS/2
- *udf_mem_sz*: UDF 不再在共享内存中传送数据，所以不再支持此参数

解决方案: 从应用程序中除去对这些参数的所有引用。

废弃的数据库配置参数

WIN	UNIX

更改: 下列数据库配置参数已过时:

- *buffpage*: 在先前版本中，可以使用缺省大小创建或改变缓冲池，并且将 *buffpage* 的值用作缺省值。在版本 8 中，应在 ALTER BUFFERPOOL 或 CREATE BUFFERPOOL 语句中使用 SIZE 关键字显式指定缓冲池的大小。
- *copyprotect*
- *indexsort*

解决方案: 从应用程序中除去对这些参数的所有引用。

发行版之间的版本 7 不兼容性

应用程序编程

Query Patroller 通用客户机

WIN	UNIX	OS/2

更改: 因为有新的存储过程，所以这个新版本的客户机应用程序使能器（CAE）只能使用 “Query Patroller 服务器” 版本 7。CAE 是 DB2 的应用程序接口，所有应用程序最终必须通过它来存取数据库。

症状: 若对下级服务器运行此 CAE，则返回消息 SQL29001。

对象变换函数和结构化类型

WIN	UNIX	OS/2

更改: 在版本 7 之前的客户机与版本 7 服务器之间有一个次要的且可能性极小的不兼容性，此不兼容性与对 SQLDA 所作的更改相关。除了值 X’00’ 和 X’01’ 之外，第二个 SQLVAR 的字节 8 的值现在还可采用 X’12’。不期望新值的应用程序可能会受此扩展影响。

解决方案: 因为将来的发行版可能会对此字段进行其它扩展，所以建议开发者仅测试显式定义的值。

JVM 使用的类和 Jar 文件的版本

WIN	UNIX	OS/2

更改: 以前，在 Java 存储过程或用户定义函数（UDF）启动后，“Java 虚拟机”（JVM）便锁定 CLASSPATH 中给出的所有文件（包括 `sqllib/function` 中的那些文件）。在数据库管理器停止之前，JVM 使用这些文件。根据运行存储过程或 UDF 的环境的不同（即，根据 *keepdari* 数据库管理器配置参数的值，以及该存储过程是否受保护），刷新类允许您替换类和 jar 文件，而不会停止数据库管理器。这与先前行为不同。

“安装 Jar”、“替换 Jar”和“除去 Jar”命令更改了功能

WIN	UNIX	OS/2

更改: 以前，安装 jar 会导致清除所有 DARI（数据库应用程序远程接口）进程。这样，保证在下一次调用时挑选新的存储过程类。目前，不会有 jar 命令来刷新 DARI 进程。要确保挑选新安装或替换的 jar 中的类，必须显式发出 SQLEJ.REFRESH_CLASSES 命令。

由于不清除 DARI 进程而引入的另一不兼容性是这样一个事实: 对于受保护存储过程，当 *keepdari* 数据库管理器配置参数的值设置为 “YES” 时，客户机可能会获得不同版本的 jar 文件。考虑以下方案:

1. 用户 A 替换一个 jar，但未刷新类。
2. 然后，用户 A 从该 jar 调用一个存储过程。假定此调用使用同一 DARI 进程，则用户 A 将得到旧版本的 jar 文件。
3. 用户 B 调用同一存储过程。此调用使用一个新的 DARI，这表示新创建的类装入程序将挑选新版本的 jar 文件。

换而言之，若在 jar 操作后不刷新类，则可能会调用不同版本的 jar 的存储过程，这取决于正在使用的 DARI 进程。这与先前行为不同，先前行为（通过清除 DARI 进程）确保始终使用新类。

32 位应用程序不兼容性

	UNIX	

更改: 32 位可执行文件（DB2 应用程序）不能对新的 64 位数据库引擎运行。

症状: 应用程序连接失败。当尝试对 64 位 DB2 应用程序库连接 32 位目标时，显示操作系统连接程序错误消息。

解决方案: 必须将该应用程序重新编译为 64 位可执行程序，并对新的 64 位 DB2 库重新连接。

更改暂存区的长度字段

WIN	UNIX	OS/2

更改: 现在，任何更改传送给用户定义函数（UDF）的暂存区的长度字段的 UDF 都将接收到 SQLCODE -450。

症状: 更改暂存区的长度字段的 UDF 失败。调用语句接收到 SQLCODE -450，其中填入了模式和特定函数名。

解决方案: 重新编写 UDF 主体，不更改暂存区的长度字段。

SQL

使用由模式 SESSION 限定的规则表的应用程序

WIN	UNIX	OS/2

更改: 临时表只允许 SESSION 模式，现在，DB2 使用此模式来指示 SESSION 限定的表可引用临时表。然而，SESSION 不是一个为临时表保留的关键字，可以

用作规则基本表的模式。因此，应用程序会发现同时存在一个 SESSION.T1 实表和一个 SESSION.T1 已声明临时表。当绑定程序包时，若遇到包含由 ″SESSION″（显式或隐式）限定的表引用的静态语句，则既不将此语句的一节也不将此语句的相关性存储在目录中。而是，需要在运行时以增量方式绑定此节。这将把该节的一个副本放在动态 SQL 高速缓存中，高速缓存的副本供该应用程序的唯一实例专用。在运行时，若存在与该表名相匹配的已声明临时表，则即使存在同名的持久基本表，也使用该已声明临时表。

症状: 在版本 6（和更早版本）中，任何带有包含由 SESSION 限定的表的静态语句的程序包都始终引用持久基本表。当绑定该程序包时，将该语句的一节以及相应相关性记录保存在目录中。在版本 7 中，这些语句不在绑定时绑定，可以在运行时将它们解析为同名的已声明临时表。因此，会发生下列情况:

- 从版本 5 迁移。若版本 5 中存在这样的程序包，则将在版本 6 中再次绑定它，静态语句现在将以增量方式绑定。这会影响性能，因为除了不能在其它应用程序（甚至是同一应用程序可执行程序的不同实例）之间共享高速缓存的动态节外，这些以增量方式绑定的节的行为与高速缓存的动态 SQL 相同。
- 从版本 6 迁移至版本 7。即使版本 6 中存在这样的程序包，也无需在版本 7 中再次绑定它。而是，那些语句仍作为常规静态 SQL 执行，使用在原始绑定时保存在目录中的节。然而，若重新绑定此程序包（无论是隐式绑定还是显式绑定），则不再存储程序包中具有 SESSION 限定的表引用的语句，它们将需要增量绑定。这会降低性能。

总而言之，在版本 7 中绑定的具有引用 SESSION 限定表的静态语句的任何程序包，都不再象静态 SQL 那样执行，因为它们需要增量绑定。事实上，若应用程序进程向与现有 SESSION 限定表、视图或别名同名的一个表发出 DECLARE GLOBAL TEMPORARY TABLE 语句，则对那些对象的引用始终被改为引用已声明临时表。

解决方案: 有可能的话，更改持久表的模式名，使它们不是 ″SESSION″。否则，没有其它方法，性能会受到影响，并可能与已声明临时表发生冲突。

可使用以下查询来标识在应用程序使用临时表时可能会受影响的表、视图和别名:

```
select tabschema, tabname from SYSCAT.TABLES where tabschema = 'SESSION'
```

可使用以下查询来标识在目录中存储了静态节、并且在重新绑定程序包时其行为可能会更改的版本 7 绑定程序包（仅当从版本 6 移至版本 7 时才相关）:

```
select pkgschema, pkgname, bschema, bname from syscat.packagedep
   where bschema = 'SESSION' and btype in ('T', 'V', 'I')
```

实用程序和工具

AIX 和 Solaris 上的 db2set

	UNIX	

更改: 命令“db2set -ul（用户级别）”及其相关函数未移植至 AIX 或 Solaris。

连通性和共存

32 位客户机不兼容性

WIN	UNIX	OS/2

更改: 32 位客户机不能连接实例，也不能与 64 位服务器上的数据库连接。

症状: 若客户机和服务器都在运行版本 7 代码，则返回 SQL1434N；否则，连接失败，并返回 SQLCODE -30081。

解决方案: 使用 64 位客户机。

附录 B. 本地语言支持（NLS）

本节包含有关 DB2 提供的本地语言支持（NLS）的信息，包括有关国家或地区、语言和受支持的代码页（代码集）的信息，并介绍如何在数据库和应用程序中配置和使用 DB2 NLS 功能部件。

本地语言版本

DB2 版本 8 有简体中文、繁体中文、丹麦语、英语、芬兰语、法语、德语、意大利语、日语、韩国语、挪威语、波兰语、巴西葡萄牙语、俄语、西班牙语和瑞典语等版本。

DB2 运行时客户机有下列附加语言版本: 阿拉伯语、保加利亚语、克罗地亚语、捷克语、荷兰语、希腊语、希伯莱语、匈牙利语、葡萄牙语、罗马尼亚语、斯洛伐克语、斯洛文尼亚语和土耳其语。

相关参考:

- 第 201 页的『受支持的国家或地区代码和代码页』

受支持的国家或地区代码和代码页

第 202 页的表 23 显示了受数据库服务器支持的语言和代码集，并显示了如何将这些值映射为数据库管理器使用的国家或地区代码或代码页值。

以下是该表每一列的说明:

- **代码页**显示 IBM 定义的代码页，它是从操作系统代码集映射而来的。
- **组**显示代码页是单字节（“S”）、双字节（“D”）还是中性（“N”）。“-n”是一个用于创建字母 — 数字组合的数。匹配组合显示 DB2 在何处允许连接和转换。例如，所有的“S-1”个组都可一起工作。但是，如果该组是中性的，则允许使用列示的任何其它代码页进行连接和转换。
- **代码集**显示与受支持的语言相关的代码集。代码集被映射为 DB2 代码页。
- **国家或地区标识**显示国家或地区标识符。
- **国家或地区代码**显示数据库管理器为提供特定于区域的支持在内部使用的代码。
- **语言环境**显示数据库管理器所支持的语言环境值。

- **操作系统**显示支持此语言和代码集的操作系统。
- **国家或地区**显示国家或地区的名称。

表 23. 受支持的语言和代码集

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
1200		16 位 Unicode			-	任意	任意
1208		UTF-8 编码的 Unicode				任意	任意
819	S-1	ISO8859-1	AL	355	sq_AL	AIX	阿尔巴尼亚
850	S-1	IBM-850	AL	355	-	AIX	阿尔巴尼亚
923	S-1	ISO8859-15	AL	355	sq_AL.8859-15	AIX	阿尔巴尼亚
1208	N-1	UTF-8	AL	355	SQ_AL	AIX	阿尔巴尼亚
37	S-1	IBM-37	AL	355	-	HOST	阿尔巴尼亚
1140	S-1	IBM-1140	AL	355	-	HOST	阿尔巴尼亚
819	S-1	iso88591	AL	355	-	HP	阿尔巴尼亚
923	S-1	iso885915	AL	355	-	HP	阿尔巴尼亚
1051	S-1	roman8	AL	355	-	HP	阿尔巴尼亚
437	S-1	IBM-437	AL	355	-	OS2	阿尔巴尼亚
850	S-1	IBM-850	AL	355	-	OS2	阿尔巴尼亚
819	S-1	ISO8859-1	AL	355	-	Sun	阿尔巴尼亚
923	S-1	ISO8859-15	AL	355	-	Sun	阿尔巴尼亚
1252	S-1	1252	AL	355	-	WIN	阿尔巴尼亚
1046	S-6	IBM-1046	AA	785	Ar_AA	AIX	阿拉伯国家或地区
1089	S-6	ISO8859-6	AA	785	ar_AA	AIX	阿拉伯国家或地区
1208	N-1	UTF-8	AA	785	AR_AA	AIX	阿拉伯国家或地区
420	S-6	IBM-420	AA	785	-	HOST	阿拉伯国家或地区
425	S-6	IBM-425	AA	785	-	HOST	阿拉伯国家或地区
1089	S-6	iso88596	AA	785	ar_SA.iso88596	HP	阿拉伯国家或地区
864	S-6	IBM-864	AA	785	-	OS2	阿拉伯国家或地区
1256	S-6	1256	AA	785	-	WIN	阿拉伯国家或地区
819	S-1	ISO8859-1	AU	61	en_AU	AIX	澳大利亚

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
850	S-1	IBM-850	AU	61	-	AIX	澳大利亚
923	S-1	ISO8859-15	AU	61	en_AU.8859-15	AIX	澳大利亚
1208	N-1	UTF-8	AU	61	EN_AU	AIX	澳大利亚
37	S-1	IBM-37	AU	61	-	HOST	澳大利亚
1140	S-1	IBM-1140	AU	61	-	HOST	澳大利亚
819	S-1	iso88591	AU	61	-	HP	澳大利亚
923	S-1	iso885915	AU	61	-	HP	澳大利亚
1051	S-1	roman8	AU	61	-	HP	澳大利亚
437	S-1	IBM-437	AU	61	-	OS2	澳大利亚
850	S-1	IBM-850	AU	61	-	OS2	澳大利亚
819	S-1	ISO8859-1	AU	61	en_AU	SCO	澳大利亚
819	S-1	ISO8859-1	AU	61	en_AU	Sun	澳大利亚
923	S-1	ISO8859-15	AU	61	-	Sun	澳大利亚
1252	S-1	1252	AU	61	-	WIN	澳大利亚
819	S-1	ISO8859-1	AT	43	-	AIX	奥地利
850	S-1	IBM-850	AT	43	-	AIX	奥地利
923	S-1	ISO8859-15	AT	43	-	AIX	奥地利
1208	N-1	UTF-8	AT	43	-	AIX	奥地利
37	S-1	IBM-37	AT	43	-	HOST	奥地利
1140	S-1	IBM-1140	AT	43	-	HOST	奥地利
819	S-1	iso88591	AT	43	-	HP	奥地利
923	S-1	iso885915	AT	43	-	HP	奥地利
1051	S-1	roman8	AT	43	-	HP	奥地利
819	S-1	ISO-8859-1	AT	43	de_AT	Linux	奥地利
923	S-1	ISO-8859-15	AT	43	de_AT@euro	Linux	奥地利
437	S-1	IBM-437	AT	43	-	OS2	奥地利
850	S-1	IBM-850	AT	43	-	OS2	奥地利
819	S-1	ISO8859-1	AT	43	de_AT	SCO	奥地利
819	S-1	ISO8859-1	AT	43	de_AT	Sun	奥地利
923	S-1	ISO8859-15	AT	43	de_AT.ISO8859-15	Sun	奥地利
1252	S-1	1252	AT	43	-	WIN	奥地利
915	S-5	ISO8859-5	BY	375	be_BY	AIX	白俄罗斯
1208	N-1	UTF-8	BY	375	BE_BY	AIX	白俄罗斯
1025	S-5	IBM-1025	BY	375	-	HOST	白俄罗斯
1154	S-5	IBM-1154	BY	375	-	HOST	白俄罗斯
915	S-5	ISO8859-5	BY	375	-	OS2	白俄罗斯
1131	S-5	IBM-1131	BY	375	-	OS2	白俄罗斯
1251	S-5	1251	BY	375	-	WIN	白俄罗斯

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
819	S-1	ISO8859-1	BE	32	fr_BE	AIX	比利时
819	S-1	ISO8859-1	BE	32	nl_BE	AIX	比利时
850	S-1	IBM-850	BE	32	Fr_BE	AIX	比利时
850	S-1	IBM-850	BE	32	Nl_BE	AIX	比利时
923	S-1	ISO8859-15	BE	32	fr_BE.8859-15	AIX	比利时
923	S-1	ISO8859-15	BE	32	nl_BE.8859-15	AIX	比利时
1208	N-1	UTF-8	BE	32	FR_BE	AIX	比利时
1208	N-1	UTF-8	BE	32	NL_BE	AIX	比利时
274	S-1	IBM-274	BE	32	-	HOST	比利时
500	S-1	IBM-500	BE	32	-	HOST	比利时
1148	S-1	IBM-1148	BE	32	-	HOST	比利时
819	S-1	iso88591	BE	32	-	HP	比利时
923	S-1	iso885915	BE	32	-	HP	比利时
819	S-1	ISO-8859-1	BE	32	fr_BE	Linux	比利时
819	S-1	ISO-8859-1	BE	32	nl_BE	Linux	比利时
923	S-1	ISO-8859-15	BE	32	fr_BE@euro	Linux	比利时
923	S-1	ISO-8859-15	BE	32	nl_BE@euro	Linux	比利时
437	S-1	IBM-437	BE	32	-	OS2	比利时
850	S-1	IBM-850	BE	32	-	OS2	比利时
819	S-1	ISO8859-1	BE	32	fr_BE	SCO	比利时
819	S-1	ISO8859-1	BE	32	nl_BE	SCO	比利时
819	S-1	ISO8859-1	BE	32	fr_BE	Sun	比利时
819	S-1	ISO8859-1	BE	32	nl_BE	Sun	比利时
923	S-1	ISO8859-15	BE	32	fr_BE.ISO8859-15	Sun	比利时
923	S-1	ISO8859-15	BE	32	nl_BE.ISO8859-15	Sun	比利时
1252	S-1	1252	BE	32	-	WIN	比利时
915	S-5	ISO8859-5	BG	359	bg_BG	AIX	保加利亚
1208	N-1	UTF-8	BG	359	BG_BG	AIX	保加利亚
1025	S-5	IBM-1025	BG	359	-	HOST	保加利亚
1154	S-5	IBM-1154	BG	359	-	HOST	保加利亚
915	S-5	iso88595	BG	359	bg_BG.iso88595	HP	保加利亚
855	S-5	IBM-855	BG	359	-	OS2	保加利亚
915	S-5	ISO8859-5	BG	359	-	OS2	保加利亚
1251	S-5	1251	BG	359	-	WIN	保加利亚
819	S-1	ISO8859-1	BR	55	pt_BR	AIX	巴西
850	S-1	IBM-850	BR	55	-	AIX	巴西
923	S-1	ISO8859-15	BR	55	pt_BR.8859-15	AIX	巴西
1208	N-1	UTF-8	BR	55	PT_BR	AIX	巴西
37	S-1	IBM-37	BR	55	-	HOST	巴西

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
1140	S-1	IBM-1140	BR	55	-	HOST	巴西
819	S-1	ISO8859-1	BR	55	-	HP	巴西
923	S-1	ISO8859-15	BR	55	-	HP	巴西
819	S-1	ISO-8859-1	BR	55	pt_BR	Linux	巴西
923	S-1	ISO-8859-15	BR	55	-	Linux	巴西
850	S-1	IBM-850	BR	55	-	OS2	巴西
819	S-1	ISO8859-1	BR	55	pt_BR	SCO	巴西
819	S-1	ISO8859-1	BR	55	pt_BR	Sun	巴西
923	S-1	ISO8859-15	BR	55	-	Sun	巴西
1252	S-1	1252	BR	55	-	WIN	巴西
819	S-1	ISO8859-1	CA	1	fr_CA	AIX	加拿大
850	S-1	IBM-850	CA	1	Fr_CA	AIX	加拿大
923	S-1	ISO8859-15	CA	1	fr_CA.8859-15	AIX	加拿大
1208	N-1	UTF-8	CA	1	FR_CA	AIX	加拿大
37	S-1	IBM-37	CA	1	-	HOST	加拿大
1140	S-1	IBM-1140	CA	1	-	HOST	加拿大
819	S-1	iso88591	CA	1	fr_CA.iso88591	HP	加拿大
923	S-1	iso885915	CA	1	-	HP	加拿大
1051	S-1	roman8	CA	1	fr_CA.roman8	HP	加拿大
819	S-1	ISO-8859-1	CA	1	en_CA	Linux	加拿大
923	S-1	ISO-8859-15	CA	1	-	Linux	加拿大
850	S-1	IBM-850	CA	1	-	OS2	加拿大
863	S-1	IBM-863	CA	2	-	OS2	加拿大（法语）
819	S-1	ISO8859-1	CA	1	en_CA	SCO	加拿大
819	S-1	ISO8859-1	CA	1	fr_CA	SCO	加拿大
819	S-1	ISO8859-1	CA	1	en_CA	Sun	加拿大
923	S-1	ISO8859-15	CA	1	-	Sun	加拿大
1252	S-1	1252	CA	1	-	WIN	加拿大
1383	D-4	IBM-eucCN	CN	86	zh_CN	AIX	中国（PRC）
1386	D-4	GBK	CN	86	Zh_CN.GBK	AIX	中国（PRC）
1208	N-1	UTF-8	CN	86	ZH_CN	AIX	中国（PRC）
935	D-4	IBM-935	CN	86	-	HOST	中国（PRC）
1388	D-4	IBM-1388	CN	86	-	HOST	中国（PRC）
1383	D-4	hp15CN	CN	86	zh_CN.hp15CN	HP	中国（PRC）
1383	D-4	GBK	CN	86	zh_CN.GBK	Linux	中国（PRC）
1381	D-4	IBM-1381	CN	86	-	OS2	中国（PRC）
1386	D-4	GBK	CN	86	-	OS2	中国（PRC）
1383	D-4	eucCN	CN	86	zh_CN	SCO	中国（PRC）
1383	D-4	eucCN	CN	86	zh_CN.eucCN	SCO	中国（PRC）

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
1383	D-4	gb2312	CN	86	zh	Sun	中国（PRC）
1208	N-1	UTF-8	CN	86	zh.UTF-8	Sun	中国（PRC）
1381	D-4	IBM-1381	CN	86	-	WIN	中国（PRC）
1386	D-4	GBK	CN	86	-	WIN	中国（PRC）
1392/5488	D-4		CN	86	-		中国（PRC）
参见“注”（第 220 页的1）。							
912	S-2	ISO8859-2	HR	385	hr_HR	AIX	克罗地亚
1208	N-1	UTF-8	HR	385	HR_HR	AIX	克罗地亚
870	S-2	IBM-870	HR	385	-	HOST	克罗地亚
1153	S-2	IBM-1153	HR	385	-	HOST	克罗地亚
912	S-2	iso88592	HR	385	hr_HR.iso88592	HP	克罗地亚
912	S-2	ISO-8859-2	HR	385	hr_HR	Linux	克罗地亚
852	S-2	IBM-852	HR	385	-	OS2	克罗地亚
912	S-2	ISO8859-2	HR	385	hr_HR.ISO8859-2	SCO	克罗地亚
1250	S-2	1250	HR	385	-	WIN	克罗地亚
912	S-2	ISO8859-2	CZ	421	cs_CZ	AIX	捷克共和国
1208	N-1	UTF-8	CZ	421	CS_CZ	AIX	捷克共和国
870	S-2	IBM-870	CZ	421	-	HOST	捷克共和国
1153	S-2	IBM-1153	CZ	421	-	HOST	捷克共和国
912	S-2	iso88592	CZ	421	cs_CZ.iso88592	HP	捷克共和国
912	S-2	ISO-8859-2	CZ	421	cs_CZ	Linux	捷克共和国
852	S-2	IBM-852	CZ	421	-	OS2	捷克共和国
912	S-2	ISO8859-2	CZ	421	cs_CZ.ISO8859-2	SCO	捷克共和国
1250	S-2	1250	CZ	421	-	WIN	捷克共和国
819	S-1	ISO8859-1	DK	45	da_DK	AIX	丹麦
850	S-1	IBM-850	DK	45	Da_DK	AIX	丹麦
923	S-1	ISO8859-15	DK	45	da_DK.8859-15	AIX	丹麦
1208	N-1	UTF-8	DK	45	DA_DK	AIX	丹麦
277	S-1	IBM-277	DK	45	-	HOST	丹麦
1142	S-1	IBM-1142	DK	45	-	HOST	丹麦
819	S-1	iso88591	DK	45	da_DK.iso88591	HP	丹麦
923	S-1	iso885915	DK	45	_	HP	丹麦
1051	S-1	roman8	DK	45	da_DK.roman8	HP	丹麦
819	S-1	ISO-8859-1	DK	45	da_DK	Linux	丹麦
923	S-1	ISO-8859-15	DK	45	-	Linux	丹麦
850	S-1	IBM-850	DK	45	-	OS2	丹麦
819	S-1	ISO8859-1	DK	45	da	SCO	丹麦
819	S-1	ISO8859-1	DK	45	da_DA	SCO	丹麦

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
819	S-1	ISO8859-1	DK	45	da_DK	SCO	丹麦
819	S-1	ISO8859-1	DK	45	da	Sun	丹麦
923	S-1	ISO8859-15	DK	45	da.ISO8859-15	Sun	丹麦
1252	S-1	1252	DK	45	-	WIN	丹麦
922	S-10	IBM-922	EE	372	Et_EE	AIX	爱沙尼亚
1208	N-1	UTF-8	EE	372	ET_EE	AIX	爱沙尼亚
1122	S-10	IBM-1122	EE	372	-	HOST	爱沙尼亚
1157	S-10	IBM-1157	EE	372	-	HOST	爱沙尼亚
922	S-10	IBM-922	EE	372	-	OS2	爱沙尼亚
1257	S-10	1257	EE	372	-	WIN	爱沙尼亚
819	S-1	ISO8859-1	FI	358	fi_FI	AIX	芬兰
850	S-1	IBM-850	FI	358	Fi_FI	AIX	芬兰
923	S-1	ISO8859-15	FI	358	fi_FI.8859-15	AIX	芬兰
1208	N-1	UTF-8	FI	358	FI_FI	AIX	芬兰
278	S-1	IBM-278	FI	358	-	HOST	芬兰
1143	S-1	IBM-1143	FI	358	-	HOST	芬兰
819	S-1	iso88591	FI	358	fi_FI.iso88591	HP	芬兰
923	S-1	iso885915	FI	358	-	HP	芬兰
1051	S-1	roman8	FI	358	fi-FI.roman8	HP	芬兰
819	S-1	ISO-8859-1	FI	358	fi_FI	Linux	芬兰
923	S-1	ISO-8859-15	FI	358	fi_FI@euro	Linux	芬兰
437	S-1	IBM-437	FI	358	-	OS2	芬兰
850	S-1	IBM-850	FI	358	-	OS2	芬兰
819	S-1	ISO8859-1	FI	358	fi	SCO	芬兰
819	S-1	ISO8859-1	FI	358	fi_FI	SCO	芬兰
819	S-1	ISO8859-1	FI	358	sv_FI	SCO	芬兰
819	S-1	ISO8859-1	FI	358	-	Sun	芬兰
923	S-1	ISO8859-15	FI	358	fi.ISO8859-15	Sun	芬兰
1252	S-1	1252	FI	358	-	WIN	芬兰
915	S-5	ISO8859-5	MK	389	mk_MK	AIX	马其顿共和国
1208	N-1	UTF-8	MK	389	MK_MK	AIX	马其顿共和国
1025	S-5	IBM-1025	MK	389	-	HOST	马其顿共和国
1154	S-5	IBM-1154	MK	389	-	HOST	马其顿共和国
915	S-5	iso88595	MK	389	-	HP	马其顿共和国
855	S-5	IBM-855	MK	389	-	OS2	马其顿共和国
915	S-5	ISO8859-5	MK	389	-	OS2	马其顿共和国
1251	S-5	1251	MK	389	-	WIN	马其顿共和国

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
819	S-1	ISO8859-1	FR	33	fr_FR	AIX	法国
850	S-1	IBM-850	FR	33	Fr_FR	AIX	法国
923	S-1	ISO8859-15	FR	33	fr_FR.8859-15	AIX	法国
1208	N-1	UTF-8	FR	33	FR_FR	AIX	法国
297	S-1	IBM-297	FR	33	-	HOST	法国
1147	S-1	IBM-1147	FR	33	-	HOST	法国
819	S-1	iso88591	FR	33	fr_FR.iso88591	HP	法国
923	S-1	iso885915	FR	33	-	HP	法国
1051	S-1	roman8	FR	33	fr_FR.roman8	HP	法国
819	S-1	ISO-8859-1	FR	33	fr_FR	Linux	法国
923	S-1	ISO-8859-15	FR	33	fr_FR@euro	Linux	法国
437	S-1	IBM-437	FR	33	-	OS2	法国
850	S-1	IBM-850	FR	33	-	OS2	法国
819	S-1	ISO8859-1	FR	33	fr	SCO	法国
819	S-1	ISO8859-1	FR	33	fr_FR	SCO	法国
819	S-1	ISO8859-1	FR	33	fr	Sun	法国
923	S-1	ISO8859-15	FR	33	fr.ISO8859-15	Sun	法国
1208	N-1	UTF-8	FR	33	fr.UTF-8	Sun	法国
1252	S-1	1252	FR	33	-	WIN	法国
819	S-1	ISO8859-1	DE	49	de_DE	AIX	德国
850	S-1	IBM-850	DE	49	De_DE	AIX	德国
923	S-1	ISO8859-15	DE	49	de_DE.8859-15	AIX	德国
1208	N-1	UTF-8	DE	49	DE_DE	AIX	德国
273	S-1	IBM-273	DE	49	-	HOST	德国
1141	S-1	IBM-1141	DE	49	-	HOST	德国
819	S-1	iso88591	DE	49	de_DE.iso88591	HP	德国
923	S-1	iso885915	DE	49	_	HP	德国
1051	S-1	roman8	DE	49	de_DE.roman8	HP	德国
819	S-1	ISO-8859-1	DE	49	de_DE	Linux	德国
923	S-1	ISO-8859-15	DE	49	de_DE@euro	Linux	德国
437	S-1	IBM-437	DE	49	-	OS2	德国
850	S-1	IBM-850	DE	49	-	OS2	德国
819	S-1	ISO8859-1	DE	49	de	SCO	德国
819	S-1	ISO8859-1	DE	49	de_DE	SCO	德国
819	S-1	ISO8859-1	DE	49	de	Sun	德国
923	S-1	ISO8859-15	DE	49	de.ISO8859-15	Sun	德国
1208	N-1	UTF-8	DE	49	de.UTF-8	Sun	德国
1252	S-1	1252	DE	49	-	WIN	德国
813	S-7	ISO8859-7	GR	30	el_GR	AIX	希腊

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
1208	N-1	UTF-8	GR	30	EL_GR	AIX	希腊
423	S-7	IBM-423	GR	30	-	HOST	希腊
875	S-7	IBM-875	GR	30	-	HOST	希腊
813	S-7	iso88597	GR	30	el_GR.iso88597	HP	希腊
813	S-7	ISO-8859-7	GR	30	el_GR	Linux	希腊
813	S-7	ISO8859-7	GR	30	-	OS2	希腊
869	S-7	IBM-869	GR	30	-	OS2	希腊
813	S-7	ISO8859-7	GR	30	el_GR.ISO8859-7	SCO	希腊
737	S-7	737	GR	30	-	WIN	希腊
1253	S-7	1253	GR	30	-	WIN	希腊
912	S-2	ISO8859-2	HU	36	hu_HU	AIX	匈牙利
1208	N-1	UTF-8	HU	36	HU_HU	AIX	匈牙利
870	S-2	IBM-870	HU	36	-	HOST	匈牙利
1153	S-2	IBM-1153	HU	36	-	HOST	匈牙利
912	S-2	iso88592	HU	36	hu_HU.iso88592	HP	匈牙利
912	S-2	ISO-8859-2	HU	36	hu_HU	Linux	匈牙利
852	S-2	IBM-852	HU	36	-	OS2	匈牙利
912	S-2	ISO8859-2	HU	36	hu_HU.ISO8859-2	SCO	匈牙利
1250	S-2	1250	HU	36	-	WIN	匈牙利
819	S-1	ISO8859-1	IS	354	is_IS	AIX	冰岛
850	S-1	IBM-850	IS	354	Is_IS	AIX	冰岛
923	S-1	ISO8859-15	IS	354	is_IS.8859-15	AIX	冰岛
1208	N-1	UTF-8	IS	354	IS_IS	AIX	冰岛
871	S-1	IBM-871	IS	354	-	HOST	冰岛
1149	S-1	IBM-1149	IS	354	-	HOST	冰岛
819	S-1	iso88591	IS	354	is_IS.iso88591	HP	冰岛
923	S-1	iso885915	IS	354	-	HP	冰岛
1051	S-1	roman8	IS	354	is_IS.roman8	HP	冰岛
819	S-1	ISO-8859-1	IS	354	is_IS	Linux	冰岛
923	S-1	ISO-8859-15	IS	354	-	Linux	冰岛
850	S-1	IBM-850	IS	354	-	OS2	冰岛
819	S-1	ISO8859-1	IS	354	is	SCO	冰岛
819	S-1	ISO8859-1	IS	354	is_IS	SCO	冰岛
819	S-1	ISO8859-1	IS	354	-	Sun	冰岛
923	S-1	ISO8859-15	IS	354	-	Sun	冰岛
1252	S-1	1252	IS	354	-	WIN	冰岛
806	S-13	IBM-806	IN	91	hi_IN	-	印度
1137	S-13	IBM-1137	IN	91	-	HOST	印度

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
1252	S-1	1252	ID	62	-	WIN	印度尼西亚
819	S-1	ISO8859-1	IE	353	-	AIX	爱尔兰
850	S-1	IBM-850	IE	353	-	AIX	爱尔兰
923	S-1	ISO8859-15	IE	353	-	AIX	爱尔兰
1208	N-1	UTF-8	IE	353	-	AIX	爱尔兰
285	S-1	IBM-285	IE	353	-	HOST	爱尔兰
1146	S-1	IBM-1146	IE	353	-	HOST	爱尔兰
819	S-1	iso88591	IE	353	-	HP	爱尔兰
923	S-1	iso885915	IE	353	-	HP	爱尔兰
1051	S-1	roman8	IE	353	-	HP	爱尔兰
819	S-1	ISO-8859-1	IE	353	en_IE	Linux	爱尔兰
923	S-1	ISO-8859-15	IE	353	en_IE@euro	Linux	爱尔兰
437	S-1	IBM-437	IE	353	-	OS2	爱尔兰
850	S-1	IBM-850	IE	353	-	OS2	爱尔兰
819	S-1	ISO8859-1	IE	353	en_IE.ISO8859-1	SCO	爱尔兰
819	S-1	ISO8859-1	IE	353	en_IE	Sun	爱尔兰
923	S-1	ISO8859-15	IE	353	en_IE.ISO8859-15	Sun	爱尔兰
1252	S-1	1252	IE	353	-	WIN	爱尔兰
856	S-8	IBM-856	IL	972	Iw_IL	AIX	以色列
916	S-8	ISO8859-8	IL	972	iw_IL	AIX	以色列
1208	N-1	UTF-8	IL	972	HE-IL	AIX	以色列
916	S-8	ISO-8859-8	IL	972	iw_IL	Linux	以色列
424	S-8	IBM-424	IL	972	-	HOST	以色列
862	S-8	IBM-862	IL	972	-	OS2	以色列
1255	S-8	1255	IL	972	-	WIN	以色列
819	S-1	ISO8859-1	IT	39	it_IT	AIX	意大利
850	S-1	IBM-850	IT	39	It_IT	AIX	意大利
923	S-1	ISO8859-15	IT	39	it_IT.8859-15	AIX	意大利
1208	N-1	UTF-8	IT	39	It_IT	AIX	意大利
280	S-1	IBM-280	IT	39	-	HOST	意大利
1144	S-1	IBM-1144	IT	39	-	HOST	意大利
819	S-1	iso88591	IT	39	it_IT.iso88591	HP	意大利
923	S-1	iso885915	IT	39	_	HP	意大利
1051	S-1	roman8	IT	39	it_IT.roman8	HP	意大利
819	S-1	ISO-8859-1	IT	39	it_IT	Linux	意大利
923	S-1	ISO-8859-15	IT	39	it_IT@euro	Linux	意大利
437	S-1	IBM-437	IT	39	-	OS2	意大利

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
850	S-1	IBM-850	IT	39	-	OS2	意大利
819	S-1	ISO8859-1	IT	39	it	SCO	意大利
819	S-1	ISO8859-1	IT	39	it_IT	SCO	意大利
819	S-1	ISO8859-1	IT	39	it	Sun	意大利
923	S-1	ISO8859-15	IT	39	it.ISO8859-15	Sun	意大利
1208	N-1	UTF-8	IT	39	it.UTF-8	Sun	意大利
1252	S-1	1252	IT	39	-	WIN	意大利
932	D-1	IBM-932	JP	81	Ja_JP	AIX	日本
943	D-1	IBM-943	JP	81	Ja_JP	AIX	日本
参见 “注” （第 220 页的2）。							
954	D-1	IBM-eucJP	JP	81	ja_JP	AIX	日本
1208	N-1	UTF-8	JP	81	JA_JP	AIX	日本
930	D-1	IBM-930	JP	81	-	HOST	日本
939	D-1	IBM-939	JP	81	-	HOST	日本
5026	D-1	IBM-5026	JP	81	-	HOST	日本
5035	D-1	IBM-5035	JP	81	-	HOST	日本
1390	D-1		JP	81	-	HOST	日本
1399	D-1		JP	81	-	HOST	日本
954	D-1	eucJP	JP	81	ja_JP.eucJP	HP	日本
5039	D-1	SJIS	JP	81	ja_JP.SJIS	HP	日本
954	D-1	EUC-JP	JP	81	ja_JP	Linux	日本
932	D-1	IBM-932	JP	81	-	OS2	日本
942	D-1	IBM-942	JP	81	-	OS2	日本
943	D-1	IBM-943	JP	81	-	OS2	日本
954	D-1	eucJP	JP	81	ja	SCO	日本
954	D-1	eucJP	JP	81	ja_JP	SCO	日本
954	D-1	eucJP	JP	81	ja_JP.EUC	SCO	日本
954	D-1	eucJP	JP	81	ja_JP.eucJP	SCO	日本
943	D-1	IBM-943	JP	81	ja_JP.PCK	Sun	日本
954	D-1	eucJP	JP	81	ja	Sun	日本
954	D-1	eucJP	JP	81	日语	Sun	日本
1208	N-1	UTF-8	JP	81	ja_JP.UTF-8	Sun	日本
943	D-1	IBM-943	JP	81	-	WIN	日本
1394	D-1		JP	81	-		日本
参见 “注” （第 220 页的3）。							
1251	S-5	1251	KZ	7	-	WIN	哈萨克斯坦
970	D-3	IBM-eucKR	KR	82	ko_KR	AIX	南韩
1208	N-1	UTF-8	KR	82	KO_KR	AIX	南韩

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
933	D-3	IBM-933	KR	82	-	HOST	南韩
1364	D-3	IBM-1364	KR	82	-	HOST	南韩
970	D-3	eucKR	KR	82	ko_KR.eucKR	HP	南韩
970	D-3	EUC-KR	KR	82	ko_KR	Linux	南韩
949	D-3	IBM-949	KR	82	-	OS2	南韩
970	D-3	eucKR	KR	82	ko_KR.eucKR	SGI	南韩
970	D-3	5601	KR	82	ko	Sun	南韩
1208	N-1	UTF-8	KR	82	ko.UTF-8	Sun	南韩
1363	D-3	1363	KR	82	-	WIN	南韩
819	S-1	ISO8859-1	Lat	3	-	AIX	拉丁美洲
850	S-1	IBM-850	Lat	3	-	AIX	拉丁美洲
923	S-1	ISO8859-15	Lat	3	-	AIX	拉丁美洲
1208	N-1	UTF-8	Lat	3	-	AIX	拉丁美洲
284	S-1	IBM-284	Lat	3	-	HOST	拉丁美洲
1145	S-1	IBM-1145	Lat	3	-	HOST	拉丁美洲
819	S-1	iso88591	Lat	3	-	HP	拉丁美洲
923	S-1	iso885915	Lat	3	-	HP	拉丁美洲
1051	S-1	roman8	Lat	3	-	HP	拉丁美洲
819	S-1	ISO-8859-1	Lat	3	-	Linux	拉丁美洲
923	S-1	ISO-8859-15	Lat	3	-	Linux	拉丁美洲
437	S-1	IBM-437	Lat	3	-	OS2	拉丁美洲
850	S-1	IBM-850	Lat	3	-	OS2	拉丁美洲
819	S-1	ISO8859-1	Lat	3	-	Sun	拉丁美洲
923	S-1	ISO8859-15	Lat	3	-	Sun	拉丁美洲
1252	S-1	1252	Lat	3	-	WIN	拉丁美洲
921	S-10	IBM-921	LV	371	Lv_LV	AIX	拉托维亚
1208	N-1	UTF-8	LV	371	LV_LV	AIX	拉托维亚
1112	S-10	IBM-1112	LV	371	-	HOST	拉托维亚
1156	S-10	IBM-1156	LV	371	-	HOST	拉托维亚
921	S-10	IBM-921	LV	371	-	OS2	拉托维亚
1257	S-10	1257	LV	371	-	WIN	拉托维亚
921	S-10	IBM-921	LT	370	Lt_LT	AIX	立陶宛
1208	N-1	UTF-8	LT	370	LT_LT	AIX	立陶宛
1112	S-10	IBM-1112	LT	370	-	HOST	立陶宛
1156	S-10	IBM-1156	LT	370	-	HOST	立陶宛
921	S-10	IBM-921	LT	370	-	OS2	立陶宛
1257	S-10	1257	LT	370	-	WIN	立陶宛

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
1252	S-1	1252	ID	60	-	WIN	马来西亚
819	S-1	ISO8859-1	NL	31	nl_NL	AIX	荷兰
850	S-1	IBM-850	NL	31	Nl_NL	AIX	荷兰
923	S-1	ISO8859-15	NL	31	nl_NL.8859-15	AIX	荷兰
1208	N-1	UTF-8	NL	31	NL_NL	AIX	荷兰
37	S-1	IBM-37	NL	31	-	HOST	荷兰
1140	S-1	IBM-1140	NL	31	-	HOST	荷兰
819	S-1	iso88591	NL	31	nl_NL.iso88591	HP	荷兰
923	S-1	iso885915	NL	31	_	HP	荷兰
1051	S-1	roman8	NL	31	nl_NL.roman8	HP	荷兰
819	S-1	ISO-8859-1	NL	31	nl_NL	Linux	荷兰
923	S-1	ISO-8859-15	NL	31	nl_NL@euro	Linux	荷兰
437	S-1	IBM-437	NL	31	-	OS2	荷兰
850	S-1	IBM-850	NL	31	-	OS2	荷兰
819	S-1	ISO8859-1	NL	31	nl	SCO	荷兰
819	S-1	ISO8859-1	NL	31	nl_NL	SCO	荷兰
819	S-1	ISO8859-1	NL	31	nl	Sun	荷兰
923	S-1	ISO8859-15	NL	31	nl.ISO8859-15	Sun	荷兰
1252	S-1	1252	NL	31	-	WIN	荷兰
819	S-1	ISO8859-1	NZ	64	-	AIX	新西兰
850	S-1	IBM-850	NZ	64	-	AIX	新西兰
923	S-1	ISO8859-15	NZ	64	-	AIX	新西兰
1208	N-1	UTF-8	NZ	64	-	AIX	新西兰
37	S-1	IBM-37	NZ	64	-	HOST	新西兰
1140	S-1	IBM-1140	NZ	64	-	HOST	新西兰
819	S-1	ISO8859-1	NZ	64	-	HP	新西兰
923	S-1	ISO8859-15	NZ	64	-	HP	新西兰
850	S-1	IBM-850	NZ	64	-	OS2	新西兰
819	S-1	ISO8859-1	NZ	64	en_NZ	SCO	新西兰
819	S-1	ISO8859-1	NZ	64	en_NZ	Sun	新西兰
923	S-1	ISO8859-15	NZ	64	-	Sun	新西兰
1252	S-1	1252	NZ	64	-	WIN	新西兰
819	S-1	ISO8859-1	NO	47	no_NO	AIX	挪威
850	S-1	IBM-850	NO	47	No_NO	AIX	挪威
923	S-1	ISO8859-15	NO	47	no_NO.8859-15	AIX	挪威
1208	N-1	UTF-8	NO	47	NO_NO	AIX	挪威
277	S-1	IBM-277	NO	47	-	HOST	挪威
1142	S-1	IBM-1142	NO	47	-	HOST	挪威

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
819	S-1	iso88591	NO	47	no_NO.iso88591	HP	挪威
923	S-1	iso885915	NO	47	-	HP	挪威
1051	S-1	roman8	NO	47	no_NO.roman8	HP	挪威
819	S-1	ISO-8859-1	NO	47	no_NO	Linux	挪威
923	S-1	ISO-8859-15	NO	47	-	Linux	挪威
850	S-1	IBM-850	NO	47	-	OS2	挪威
819	S-1	ISO8859-1	NO	47	no	SCO	挪威
819	S-1	ISO8859-1	NO	47	no_NO	SCO	挪威
819	S-1	ISO8859-1	NO	47	no	Sun	挪威
923	S-1	ISO8859-15	NO	47	-	Sun	挪威
1252	S-1	1252	NO	47	-	WIN	挪威
912	S-2	ISO8859-2	PL	48	pl_PL	AIX	波兰
1208	N-1	UTF-8	PL	48	PL_PL	AIX	波兰
870	S-2	IBM-870	PL	48	-	HOST	波兰
1153	S-2	IBM-1153	PL	48	-	HOST	波兰
912	S-2	iso88592	PL	48	pl_PL.iso88592	HP	波兰
912	S-2	ISO-8859-2	PL	48	pl_PL	Linux	波兰
852	S-2	IBM-852	PL	48	-	OS2	波兰
912	S-2	ISO8859-2	PL	48	pl_PL.ISO8859-2	SCO	波兰
1250	S-2	1250	PL	48	-	WIN	波兰
819	S-1	ISO8859-1	PT	351	pt_PT	AIX	葡萄牙
850	S-1	IBM-850	PT	351	Pt_PT	AIX	葡萄牙
923	S-1	ISO8859-15	PT	351	pt_PT.8859-15	AIX	葡萄牙
1208	N-1	UTF-8	PT	351	PT_PT	AIX	葡萄牙
37	S-1	IBM-37	PT	351	-	HOST	葡萄牙
1140	S-1	IBM-1140	PT	351	-	HOST	葡萄牙
819	S-1	iso88591	PT	351	pt_PT.iso88591	HP	葡萄牙
923	S-1	iso885915	PT	351	-	HP	葡萄牙
1051	S-1	roman8	PT	351	pt_PT.roman8	HP	葡萄牙
819	S-1	ISO-8859-1	PT	351	pt_PT	Linux	葡萄牙
923	S-1	ISO-8859-15	PT	351	pt_PT@euro	Linux	葡萄牙
850	S-1	IBM-850	PT	351	-	OS2	葡萄牙
860	S-1	IBM-860	PT	351	-	OS2	葡萄牙
819	S-1	ISO8859-1	PT	351	pt	SCO	葡萄牙
819	S-1	ISO8859-1	PT	351	pt_PT	SCO	葡萄牙
819	S-1	ISO8859-1	PT	351	pt	Sun	葡萄牙
923	S-1	ISO8859-15	PT	351	pt.ISO8859-15	Sun	葡萄牙
1252	S-1	1252	PT	351	-	WIN	葡萄牙

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
912	S-2	ISO8859-2	RO	40	ro_RO	AIX	罗马尼亚
1208	N-1	UTF-8	RO	40	RO_RO	AIX	罗马尼亚
870	S-2	IBM-870	RO	40	-	HOST	罗马尼亚
1153	S-2	IBM-1153	RO	40	-	HOST	罗马尼亚
912	S-2	iso88592	RO	40	ro_RO.iso88592	HP	罗马尼亚
912	S-2	ISO-8859-2	RO	40	ro_RO	Linux	罗马尼亚
852	S-2	IBM-852	RO	40	-	OS2	罗马尼亚
912	S-2	ISO8859-2	RO	40	ro_RO.ISO8859-2	SCO	罗马尼亚
1250	S-2	1250	RO	40	-	WIN	罗马尼亚
915	S-5	ISO8859-5	RU	7	ru_RU	AIX	俄罗斯
1208	N-1	UTF-8	RU	7	RU_RU	AIX	俄罗斯
1025	S-5	IBM-1025	RU	7	-	HOST	俄罗斯
1154	S-5	IBM-1154	RU	7	-	HOST	俄罗斯
915	S-5	iso88595	RU	7	ru_RU.iso88595	HP	俄罗斯
915	S-5	ISO-8859-5	RU	7	ru_RU	Linux	俄罗斯
866	S-5	IBM-866	RU	7	-	OS2	俄罗斯
915	S-5	ISO8859-5	RU	7	-	OS2	俄罗斯
915	S-5	ISO8859-5	RU	7	ru_RU.ISO8859-5	SCO	俄罗斯
1251	S-5	1251	RU	7	-	WIN	俄罗斯
915	S-5	ISO8859-5	SP	381	sr_SP	AIX	芒特尼格罗共和国
1208	N-1	UTF-8	SP	381	SR_SP	AIX	芒特尼格罗共和国
1025	S-5	IBM-1025	SP	381	-	HOST	芒特尼格罗共和国
1154	S-5	IBM-1154	SP	381	-	HOST	芒特尼格罗共和国
915	S-5	iso88595	SP	381	-	HP	芒特尼格罗共和国
855	S-5	IBM-855	SP	381	-	OS2	芒特尼格罗共和国
915	S-5	ISO8859-5	SP	381	-	OS2	芒特尼格罗共和国
1251	S-5	1251	SP	381	-	WIN	芒特尼格罗共和国
912	S-2	ISO8859-2	SK	422	sk_SK	AIX	斯洛文尼亚
1208	N-1	UTF-8	SK	422	SK_SK	AIX	斯洛文尼亚

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
870	S-2	IBM-870	SK	422	-	HOST	斯洛文尼亚
1153	S-2	IBM-1153	SK	422	-	HOST	斯洛文尼亚
912	S-2	iso88592	SK	422	sk_SK.iso88592	HP	斯洛文尼亚
852	S-2	IBM-852	SK	422	-	OS2	斯洛文尼亚
912	S-2	ISO8859-2	SK	422	sk_SK.ISO8859-2	SCO	斯洛文尼亚
1250	S-2	1250	SK	422	-	WIN	斯洛文尼亚
912	S-2	ISO8859-2	SI	386	sl_SI	AIX	斯洛文尼亚
1208	N-1	UTF-8	SI	386	SL_SI	AIX	斯洛文尼亚
870	S-2	IBM-870	SI	386	-	HOST	斯洛文尼亚
1153	S-2	IBM-1153	SI	386	-	HOST	斯洛文尼亚
912	S-2	iso88592	SI	386	sl_SI.iso88592	HP	斯洛文尼亚
912	S-2	ISO-8859-2	SI	386	sl_SI	Linux	斯洛文尼亚
852	S-2	IBM-852	SI	386	-	OS2	斯洛文尼亚
912	S-2	ISO8859-2	SI	386	sl_SI.ISO8859-2	SCO	斯洛文尼亚
1250	S-2	1250	SI	386	-	WIN	斯洛文尼亚
819	S-1	ISO8859-1	ZA	27	en_ZA	AIX	南非
850	S-1	IBM-850	ZA	27	En_ZA	AIX	南非
923	S-1	ISO8859-15	ZA	27	en_ZA.8859-15	AIX	南非
1208	N-1	UTF-8	ZA	27	EN_ZA	AIX	南非
285	S-1	IBM-285	ZA	27	-	HOST	南非
1146	S-1	IBM-1146	ZA	27	-	HOST	南非
819	S-1	iso88591	ZA	27	-	HP	南非
923	S-1	iso885915	ZA	27	-	HP	南非
1051	S-1	roman8	ZA	27	-	HP	南非
437	S-1	IBM-437	ZA	27	-	OS2	南非
850	S-1	IBM-850	ZA	27	-	OS2	南非
819	S-1	ISO8859-1	ZA	27	en_ZA.ISO8859-1	SCO	南非
819	S-1	ISO8859-1	ZA	27	-	Sun	南非
923	S-1	ISO8859-15	ZA	27	-	Sun	南非
1252	S-1	1252	ZA	27	-	WIN	南非
819	S-1	ISO8859-1	ES	34	es_ES	AIX	西班牙
819	S-1	ISO8859-1	ES	34	ca_ES	AIX	西班牙（加泰隆）
850	S-1	IBM-850	ES	34	Es_ES	AIX	西班牙
850	S-1	IBM-850	ES	34	Ca_ES	AIX	西班牙（加泰隆）
923	S-1	ISO8859-15	ES	34	es_ES.8859-15	AIX	西班牙

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
923	S-1	ISO8859-15	ES	34	ca_ES.8859-15	AIX	西班牙（加泰隆）
1208	N-1	UTF-8	ES	34	ES_ES	AIX	西班牙
1208	N-1	UTF-8	ES	34	CA_ES	AIX	西班牙（加泰隆）
284	S-1	IBM-284	ES	34	-	HOST	西班牙
1145	S-1	IBM-1145	ES	34	-	HOST	西班牙
819	S-1	iso88591	ES	34	es_ES.iso88591	HP	西班牙
923	S-1	iso885915	ES	34	-	HP	西班牙
1051	S-1	roman8	ES	34	es_ES.roman8	HP	西班牙
819	S-1	ISO-8859-1	ES	34	es_ES	Linux	西班牙
923	S-1	ISO-8859-15	ES	34	es_ES@euro	Linux	西班牙
437	S-1	IBM-437	ES	34	-	OS2	西班牙
850	S-1	IBM-850	ES	34	-	OS2	西班牙
819	S-1	ISO8859-1	ES	34	es	SCO	西班牙
819	S-1	ISO8859-1	ES	34	es_ES	SCO	西班牙
819	S-1	ISO8859-1	ES	34	es	Sun	西班牙
923	S-1	ISO8859-15	ES	34	es.ISO8859-15	Sun	西班牙
1208	N-1	UTF-8	ES	34	es.UTF-8	Sun	西班牙
1252	S-1	1252	ES	34	-	WIN	西班牙
819	S-1	ISO8859-1	SE	46	sv_SE	AIX	瑞典
850	S-1	IBM-850	SE	46	Sv_SE	AIX	瑞典
923	S-1	ISO8859-15	SE	46	sv_SE.8859-15	AIX	瑞典
1208	N-1	UTF-8	SE	46	SV_SE	AIX	瑞典
278	S-1	IBM-278	SE	46	-	HOST	瑞典
1143	S-1	IBM-1143	SE	46	-	HOST	瑞典
819	S-1	iso88591	SE	46	sv_SE.iso88591	HP	瑞典
923	S-1	iso885915	SE	46	-	HP	瑞典
1051	S-1	roman8	SE	46	sv_SE.roman8	HP	瑞典
819	S-1	ISO-8859-1	SE	46	sv_SE	Linux	瑞典
923	S-1	ISO-8859-15	SE	46	-	Linux	瑞典
437	S-1	IBM-437	SE	46	-	OS2	瑞典
850	S-1	IBM-850	SE	46	-	OS2	瑞典
819	S-1	ISO8859-1	SE	46	sv	SCO	瑞典
819	S-1	ISO8859-1	SE	46	sv_SE	SCO	瑞典
819	S-1	ISO8859-1	SE	46	sv	Sun	瑞典
923	S-1	ISO8859-15	SE	46	sv.ISO8859-15	Sun	瑞典
1208	N-1	UTF-8	SE	46	sv.UTF-8	Sun	瑞典
1252	S-1	1252	SE	46	-	WIN	瑞典

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
819	S-1	ISO8859-1	CH	41	de_CH	AIX	瑞士
850	S-1	IBM-850	CH	41	De_CH	AIX	瑞士
923	S-1	ISO8859-15	CH	41	de_CH.8859-15	AIX	瑞士
1208	N-1	UTF-8	CH	41	DE_CH	AIX	瑞士
500	S-1	IBM-500	CH	41	-	HOST	瑞士
1148	S-1	IBM-1148	CH	41	-	HOST	瑞士
819	S-1	iso88591	CH	41	-	HP	瑞士
923	S-1	iso885915	CH	41	-	HP	瑞士
1051	S-1	roman8	CH	41	-	HP	瑞士
819	S-1	ISO-8859-1	CH	41	de_CH	Linux	瑞士
923	S-1	ISO-8859-15	CH	41	-	Linux	瑞士
437	S-1	IBM-437	CH	41	-	OS2	瑞士
850	S-1	IBM-850	CH	41	-	OS2	瑞士
819	S-1	ISO8859-1	CH	41	de_CH	SCO	瑞士
819	S-1	ISO8859-1	CH	41	fr_CH	SCO	瑞士
819	S-1	ISO8859-1	CH	41	it_CH	SCO	瑞士
819	S-1	ISO8859-1	CH	41	de_CH	Sun	瑞士
923	S-1	ISO8859-15	CH	41	-	Sun	瑞士
1252	S-1	1252	CH	41	-	WIN	瑞士
950	D-2	big5	TW	88	Zh_TW	AIX	台湾
964	D-2	IBM-eucTW	TW	88	zh_TW	AIX	台湾
1208	N-1	UTF-8	TW	88	ZH_TW	AIX	台湾
937	D-2	IBM-937	TW	88	-	HOST	台湾
1371	D-2	IBM-1371	TW	88	-	HOST	台湾
950	D-2	big5	TW	88	zh_TW.big5	HP	台湾
964	D-2	eucTW	TW	88	zh_TW.eucTW	HP	台湾
950	D-2	BIG5	TW	88	zh_TW	Linux	台湾
938	D-2	IBM-938	TW	88	-	OS2	台湾
948	D-2	IBM-948	TW	88	-	OS2	台湾
950	D-2	big5	TW	88	-	OS2	台湾
950	D-2	big5	TW	88	zh_TW.BIG5	Sun	台湾
参见“注”（第 222 页的8）。							
964	D-2	cns11643	TW	88	zh_TW	Sun	台湾
1208	N-1	UTF-8	TW	88	zh_TW.UTF-8	Sun	台湾
950	D-2	big5	TW	88	-	WIN	台湾
874	S-20	TIS620-1	TH	66	th_TH	AIX	泰国
1208	N-1	UTF-8	TH	66	TH_TH	AIX	泰国
838	S-20	IBM-838	TH	66	-	HOST	泰国

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
1160	S-20	IBM-1160	TH	66	-	HOST	泰国
874	S-20	tis620	TH	66	th_TH.tis620	HP	泰国
874	S-20	TIS620-1	TH	66	-	OS2	泰国
874	S-20	TIS620-1	TH	66	-	WIN	泰国
920	S-9	ISO8859-9	TR	90	tr_TR	AIX	土耳其
1208	N-1	UTF-8	TR	90	TR_TR	AIX	土耳其
1026	S-9	IBM-1026	TR	90	-	HOST	土耳其
1155	S-9	IBM-1155	TR	90	-	HOST	土耳其
920	S-9	iso88599	TR	90	tr_TR.iso88599	HP	土耳其
920	S-9	ISO-8859-9	TR	90	tr_TR	Linux	土耳其
857	S-9	IBM-857	TR	90	-	OS2	土耳其
920	S-9	ISO8859-9	TR	90	tr_TR.ISO8859-9	SCO	土耳其
1254	S-9	1254	TR	90	-	WIN	土耳其
819	S-1	ISO8859-1	GB	44	en_GB	AIX	英国
850	S-1	IBM-850	GB	44	En_GB	AIX	英国
923	S-1	ISO8859-15	GB	44	en_GB.8859-15	AIX	英国
1208	N-1	UTF-8	GB	44	EN_GB	AIX	英国
285	S-1	IBM-285	GB	44	-	HOST	英国
1146	S-1	IBM-1146	GB	44	-	HOST	英国
819	S-1	iso88591	GB	44	en_GB.iso88591	HP	英国
923	S-1	iso885915	GB	44	-	HP	英国
1051	S-1	roman8	GB	44	en_GB.roman8	HP	英国
819	S-1	ISO-8859-1	GB	44	en_GB	Linux	英国
923	S-1	ISO-8859-15	GB	44	-	Linux	英国
437	S-1	IBM-437	GB	44	-	OS2	英国
850	S-1	IBM-850	GB	44	-	OS2	英国
819	S-1	ISO8859-1	GB	44	en_GB	SCO	英国
819	S-1	ISO8859-1	GB	44	en	SCO	英国
819	S-1	ISO8859-1	GB	44	en_GB	Sun	英国
923	S-1	ISO8859-15	GB	44	en_GB.ISO8859-15	Sun	英国
1252	S-1	1252	GB	44	-	WIN	英国
1124	S-12	IBM-1124	UA	380	Uk_UA	AIX	乌克兰
1208	N-1	UTF-8	UA	380	UK_UA	AIX	乌克兰
1123	S-12	IBM-1123	UA	380	-	HOST	乌克兰
1158	S-12	IBM-1158	UA	380	-	HOST	乌克兰
1125	S-12	IBM-1125	UA	380	-	OS2	乌克兰
1251	S-12	1251	UA	380	-	WIN	乌克兰

表 23. 受支持的语言和代码集 (续)

代码页	组	代码集	国家或地区标识	国家或地区代码	语言环境	操作系统	国家或地区
819	S-1	ISO8859-1	US	1	en_US	AIX	美国
850	S-1	IBM-850	US	1	En_US	AIX	美国
923	S-1	ISO8859-15	US	1	en_US.8859-15	AIX	美国
1208	N-1	UTF-8	US	1	EN_US	AIX	美国
37	S-1	IBM-37	US	1	-	HOST	美国
1140	S-1	IBM-1140	US	1	-	HOST	美国
819	S-1	iso88591	US	1	en_US.iso88591	HP	美国
923	S-1	iso885915	US	1	-	HP	美国
1051	S-1	roman8	US	1	en_US.roman8	HP	美国
819	S-1	ISO-8859-1	US	1	en_US	Linux	美国
923	S-1	ISO-8859-15	US	1	-	Linux	美国
437	S-1	IBM-437	US	1	-	OS2	美国
850	S-1	IBM-850	US	1	-	OS2	美国
819	S-1	ISO8859-1	US	1	en_US	SCO	美国
819	S-1	ISO8859-1	US	1	en_US	SGI	美国
819	S-1	ISO8859-1	US	1	en_US	Sun	美国
923	S-1	ISO8859-15	US	1	en_US.ISO8859-15	Sun	美国
1208	N-1	UTF-8	US	1	en_US.UTF-8	Sun	美国
1252	S-1	1252	US	1	-	WIN	美国
1129	S-11	IBM-1129	VN	84	Vi_VN	AIX	越南
1208	N-1	UTF-8	VN	84	VI_VN	AIX	越南
1130	S-11	IBM-1130	VN	84	-	HOST	越南
1164	S-11	IBM-1164	VN	84	-	HOST	越南
1129	S-11	IBM-1129	VN	84	-	OS2	越南
1258	S-11	1258	VN	84	-	WIN	越南

注:

1. CCSID 1392 和 5488（GB 18030）仅能与装入或导入实用程序一起使用以将数据从 CCSID 1392 和 5488 移至 DB2 Unicode 数据库，或从 DB2 Unicode 数据库导出至 CCSID 1392 或 5488。
2. 在 AIX 4.3 或更新版本上，代码页为 943。如果在使用 AIX 4.2 或较早版本，则代码页为 932。
3. 代码页 1394（Shift JIS X0213）仅能与装入或导入实用程序一起使用以将数据从代码页 1394 移至 DB2 Unicode 数据库，或从 DB2 Unicode 数据库导出至代码页 1394。
4. 下列各项映射为阿拉伯语国家 / 地区（AA）：
   - Arabic（沙特阿拉伯）

- Arabic（伊拉克）
- Arabic（埃及）
- Arabic（利比亚）
- Arabic（阿尔及利亚）
- Arabic（摩洛哥）
- Arabic（突尼斯）
- Arabic（阿曼）
- Arabic（也门）
- Arabic（叙利亚）
- Arabic（约旦）
- Arabic（黎巴嫩）
- Arabic（科威特）
- Arabic（阿拉伯联合酋长国）
- Arabic（巴林）
- Arabic（卡塔尔）

5. 下列各项映射为英语国家或地区（US）：
   - English（牙买加）
   - English（加勒比海）
6. 下列各项映射为拉丁美洲国家或地区（Lat）：
   - Spanish（墨西哥）
   - Spanish（危地马拉）
   - Spanish（哥斯达黎加）
   - Spanish（巴拿马）
   - Spanish（多米尼加共和国）
   - Spanish（委内瑞拉）
   - Spanish（哥伦比亚）
   - Spanish（秘鲁）
   - Spanish（阿根廷）
   - Spanish（厄瓜多尔）
   - Spanish（智利）
   - Spanish（乌拉圭）
   - Spanish（巴拉圭）
   - Spanish（玻利维亚）

7. 下列 Indic 脚本在 Unicode 中是受支持的: Hindi、Gujarati、Kannada、Konkani、Marathi、Punjabi、Sanskrit、Tamil 和 Telugu。
8. Microsoft 代码页 950 和 Solaris 代码页 950, 又称为 Big5 , 不支持 IBM950 中的下列字符:

代码范围	描述	Solaris Big5	Microsoft Big5	IBM Big5
X'C6A1'-X'C8FE'	符号	保留区域	用户定义字符	符号
X'F9D6'-X'F9FE'	ETen 扩展	保留区域	系统字体	UDC
X'F286'-X'F9A0'	IBM 选择的字符	保留区域	未使用	IBM 选择的字符

启用和禁用欧元符号支持

"DB2 通用数据库" 提供对欧元货币符号的支持。已将欧元符号添加至许多代码页。许多 DB2 UDB 内部代码页转换表和 sqllib/conv/ 目录中的许多外部代码页转换表文件已增强为支持欧元符号。

Microsoft ANSI 代码页已修改为在位置 X'80' 包括欧元货币符号。代码页 850 已修改为将字符 DOTLESS I (在位置 X'D5' 处) 替换为欧元货币符号。DB2 UDB 内部代码页转换例程将这些修订过的代码页定义用作缺省定义以提供欧元符号支持。

但是, 如果想要使用代码页转换表的非欧元定义, 则在安装完成之后遵循下面的过程。

先决条件:

要替换现有外部代码页转换表文件, 您可能想要在以非欧元版本覆盖当前文件之前备份这些文件。

文件位于目录 sqllib/conv/ 中。在 UNIX 上, sqllib/conv/ 与 DB2 的安装路径相链接。

过程:

要禁用欧元符号支持:

1. 从 ftp://ftp.software.ibm.com/ps/products/db2/info/vr8/conv/ 上下载相应的二进制格式转换表文件。此 ftp 服务器是匿名的, 所以, 如果正通过命令行连接, 则

以用户 “anonymous” 登录并使用电子邮件地址作为密码。在登录后，切换至转换表目录：`cd ps/products/db2/info/vr8/conv`

2. 将文件复制至 `sqllib/conv/` 目录。
3. 重新启动 DB2。

代码页 819 和 1047:

对于代码页 819（ISO 8859-1 Latin 1 ASCII）和 1047（Latin 1 Open System EBCDIC），欧元替换代码页 923（ISO 8859-15 Latin 9 ASCII）和 924（Latin 9Open System EBCDIC）分别包含欧元符号和几个新的字符。在缺省情况下，DB2 仍然使用这两个代码页和转换表的旧（非欧元）定义，即 819 和 1047。有两种方法来激活新 923/924 代码页和相关联的转换表:

- 创建使用新代码页的新数据库。例如,

  ```
  DB2 CREATE DATABASE dbname USINGCODESET ISO8859-15 TERRITORY US
  ```

- 将`sqllib/conv/` 目录中的 923 或 924 转换表分别重命名或复制为 819 或 1047。

相关概念:

- 『Character conversion』（*SQL Reference, Volume 1*）

相关参考:

- 第 233 页的『代码页 923 和 924 的转换表』
- 第 223 页的『启用欧元的代码页的转换表文件』

启用欧元的代码页的转换表文件

下列各表列示已增强为支持欧元货币符号的转换表。如果想要禁用欧元符号支持，则下载标题为 “转换表文件” 的列中指示的转换表文件。

阿拉伯语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
420, 16804	1046, 9238	04201046.cnv, IBM00420.ucs
420, 16804	1256, 5352	04201256.cnv, IBM00420.ucs
420, 16804	1200, 1208, 13488, 17584	IBM00420.ucs
864, 17248	1046, 9238	08641046.cnv, 10460864.cnv, IBM00864.ucs
864, 17248	1256, 5352	08641256.cnv, 12560864.cnv, IBM00864.ucs

数据库服务器 **CCSID/CPGID**	数据库客户机 **CCSID/CPGID**	转换表文件
864，17248	1200，1208，13488，17584	IBM00864.ucs
1046，9238	864，17248	10460864.cnv，08641046.cnv， IBM01046.ucs
1046，9238	1089	10461089.cnv，10891046.cnv， IBM01046.ucs
1046，9238	1256，5352	10461256.cnv，12561046.cnv， IBM01046.ucs
1046，9238	1200，1208，13488，17584	IBM01046.ucs
1089	1046，9238	10891046.cnv，10461089.cnv
1256，5352	864，17248	12560864.cnv，08641256.cnv， IBM01256.ucs
1256，5352	1046，9238	12561046.cnv，10461256.cnv， IBM01256.ucs
1256，5352	1200，1208，13488，17584	IBM01256.ucs

Baltic:

数据库服务器 **CCSID/CPGID**	数据库客户机 **CCSID/CPGID**	转换表文件
921，901	1257	09211257.cnv，12570921.cnv， IBM00921.ucs
921，901	1200，1208，13488，17584	IBM00921.ucs
1112，1156	1257，5353	11121257.cnv
1257，5353	921，901	12570921.cnv，09211257.cnv， IBM01257.ucs
1257，5353	922，902	12570922.cnv，09221257.cnv， IBM01257.ucs
1257，5353	1200，1208，13488，17584	IBM01257.ucs

Belarus:

数据库服务器 **CCSID/CPGID**	数据库客户机 **CCSID/CPGID**	转换表文件
1131，849	1251，5347	11311251.cnv，12511131.cnv
1131，849	1283	11311283.cnv

Cyrillic:

数据库服务器 **CCSID/CPGID**	数据库客户机 **CCSID/CPGID**	转换表文件
855, 872	866, 808	08550866.cnv, 08660855.cnv
855, 872	1251, 5347	08551251.cnv, 12510855.cnv
866, 808	855, 872	08660855.cnv, 08550866.cnv
866, 808	1251, 5347	08661251.cnv, 12510866.cnv
1025, 1154	855, 872	10250855.cnv, IBM01025.ucs
1025, 1154	866, 808	10250866.cnv, IBM01025.ucs
1025, 1154	1131, 849	10251131.cnv, IBM01025.ucs
1025, 1154	1251, 5347	10251251.cnv, IBM01025.ucs
1025, 1154	1200, 1208, 13488, 17584	IBM01025.ucs
1251, 5347	855, 872	12510855.cnv, 08551251.cnv, IBM01251.ucs
1251, 5347	866, 808	12510866.cnv, 08661251.cnv, IBM01251.ucs
1251, 5347	1124	12511124.cnv, 11241251.cnv, IBM01251.ucs
1251, 5347	1125, 848	12511125.cnv, 11251251.cnv, IBM01251.ucs
1251, 5347	1131, 849	12511131.cnv, 11311251.cnv, IBM01251.ucs
1251, 5347	1200, 1208, 13488, 17584	IBM01251.ucs

Estonia:

数据库服务器 **CCSID/CPGID**	数据库客户机 **CCSID/CPGID**	转换表文件
922, 902	1257	09221257.cnv, 12570922.cnv, IBM00922.ucs
922, 902	1200, 1208, 13488, 17584	IBM00922.ucs
1122, 1157	1257, 5353	11221257.cnv

希腊语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
423	869，9061	04230869.cnv
813，4909	869，9061	08130869.cnv，08690813.cnv，IBM00813.ucs
813，4909	1253，5349	08131253.cnv，12530813.cnv，IBM00813.ucs
813，4909	1200，1208，13488，17584	IBM00813.ucs
869，9061	813，4909	08690813.cnv，08130869.cnv
869，9061	1253，5349	08691253.cnv，12530869.cnv
875，4971	813，4909	08750813.cnv，IBM00875.ucs
875，4971	1253，5349	08751253.cnv，IBM00875.ucs
875，4971	1200，1208，13488，17584	IBM00875.ucs
1253，5349	813，4909	12530813.cnv，08131253.cnv，IBM01253.ucs
1253，5349	869，9061	12530869.cnv，08691253.cnv，IBM01253.ucs
1253，5349	1200，1208，13488，17584	IBM01253.ucs

希伯莱语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
424，12712	856，9048	04240856.cnv，IBM00424.ucs
424，12712	862，867	04240862.cnv，IBM00424.ucs
424，12712	916	04240916.cnv，IBM00424.ucs
424，12712	1255，5351	04241255.cnv，IBM00424.ucs
424，12712	1200，1208，13488，17584	IBM00424.ucs
856，9048	862，867	08560862.cnv，08620856.cnv，IBM0856.ucs
856，9048	916	08560916.cnv，09160856.cnv，IBM0856.ucs
856，9048	1255，5351	08561255.cnv，12550856.cnv，IBM0856.ucs
856，9048	1200，1208，13488，17584	IBM0856.ucs

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
862，867	856，9048	08620856.cnv，08560862.cnv，IBM00862.ucs
862，867	916	08620916.cnv，09160862.cnv，IBM00862.ucs
862，867	1255，5351	08621255.cnv，12550862.cnv，IBM00862.ucs
862，867	1200，1208，13488，17584	IBM00862.ucs
916	856，9048	09160856.cnv，08560916.cnv
916	862，867	09160862.cnv，08620916.cnv
1255，5351	856，9048	12550856.cnv，08561255.cnv，IBM01255.ucs
1255，5351	862，867	12550862.cnv，08621255.cnv，IBM01255.ucs
1255，5351	1200，1208，13488，17584	IBM01255.ucs

日语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
290，8482	850，858	02900850.cnv
954	1200，1208，13488，17584	0954ucs2.cnv，ucs20954.cnv
1027，5123	850，858	10270850.cnv

拉丁语 — 1:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
37，1140	437	00370437.cnv，IBM00037.ucs
37，1140	850，858	00370850.cnv，IBM00037.ucs
37，1140	860	00370860.cnv，IBM00037.ucs
37，1140	1051	00371051.cnv，IBM00037.ucs
37，1140	1252，5348	00371252.cnv，IBM00037.ucs
37，1140	1275	00371275.cnv，IBM00037.ucs
37，1140	1200，1208，13488，17584	IBM00037.ucs

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
273, 1141	437	02730437.cnv, IBM00273.ucs
273, 1141	850, 858	02730850.cnv, IBM00273.ucs
273, 1141	1051	02731051.cnv, IBM00273.ucs
273, 1141	1252, 5348	02731252.cnv, IBM00273.ucs
273, 1141	1275	02731275.cnv, IBM00273.ucs
273, 1141	1200, 1208, 13488, 17584	IBM00273.ucs
277, 1142	437	02770437.cnv, IBM00277.ucs
277, 1142	850, 858	02770850.cnv, IBM00277.ucs
277, 1142	1051	02771051.cnv, IBM00277.ucs
277, 1142	1252, 5348	02771252.cnv, IBM00277.ucs
277, 1142	1275	02771275.cnv, IBM00277.ucs
277, 1142	1200, 1208, 13488, 17584	IBM00277.ucs
278, 1143	437	02780437.cnv, IBM00278.ucs
278, 1143	850, 858	02780850.cnv, IBM00278.ucs
278, 1143	1051	02781051.cnv, IBM00278.ucs
278, 1143	1252, 5348	02781252.cnv, IBM00278.ucs
278, 1143	1275	02781275.cnv, IBM00278.ucs
278, 1143	1200, 1208, 13488, 17584	IBM00278.ucs
280, 1144	437	02800437.cnv, IBM00280.ucs
280, 1144	850, 858	02800850.cnv, IBM00280.ucs
280, 1144	1051	02801051.cnv, IBM00280.ucs
280, 1144	1252, 5348	02801252.cnv, IBM00280.ucs
280, 1144	1275	02801275.cnv, IBM00280.ucs
280, 1144	1200, 1208, 13488, 17584	IBM00280.ucs
284, 1145	437	02840437.cnv, IBM00284.ucs
284, 1145	850, 858	02840850.cnv, IBM00284.ucs
284, 1145	1051	02841051.cnv, IBM00284.ucs
284, 1145	1252, 5348	02841252.cnv, IBM00284.ucs
284, 1145	1275	02841275.cnv, IBM00284.ucs
284, 1145	1200, 1208, 13488, 17584	IBM00284.ucs
285, 1146	437	02850437.cnv, IBM00285.ucs
285, 1146	850, 858	02850850.cnv, IBM00285.ucs

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
285, 1146	1051	02851051.cnv, IBM00285.ucs
285, 1146	1252, 5348	02851252.cnv, IBM00285.ucs
285, 1146	1275	02851275.cnv, IBM00285.ucs
285, 1146	1200, 1208, 13488, 17584	IBM00285.ucs
297, 1147	437	02970437.cnv, IBM00297.ucs
297, 1147	850, 858	02970850.cnv, IBM00297.ucs
297, 1147	1051	02971051.cnv, IBM00297.ucs
297, 1147	1252, 5348	02971252.cnv, IBM00297.ucs
297, 1147	1275	02971275.cnv, IBM00297.ucs
297, 1147	1200, 1208, 13488, 17584	IBM00297.ucs
437	850, 858	04370850.cnv, 08500437.cnv
500, 1148	437	05000437.cnv, IBM00500.ucs
500, 1148	850, 858	05000850.cnv, IBM00500.ucs
500, 1148	857, 9049	05000857.cnv, IBM00500.ucs
500, 1148	920	05000920.cnv, IBM00500.ucs
500, 1148	1051	05001051.cnv, IBM00500.ucs
500, 1148	1114, 5210	05001114.cnv, IBM00500.ucs
500, 1148	1252, 5348	05001252.cnv, IBM00500.ucs
500, 1148	1254, 5350	05001254.cnv, IBM00500.ucs
500, 1148	1275	05001275.cnv, IBM00500.ucs
500, 1148	1200, 1208, 13488, 17584	IBM00500.ucs
850, 858	437	08500437.cnv, 04370850.cnv
850, 858	860	08500860.cnv, 08600850.cnv
850, 858	1114, 5210	08501114.cnv, 11140850.cnv
850, 858	1275	08501275.cnv, 12750850.cnv
860	850, 858	08600850.cnv, 08500860.cnv
871, 1149	437	08710437.cnv, IBM00871.ucs
871, 1149	850, 858	08710850.cnv, IBM00871.ucs
871, 1149	1051	08711051.cnv, IBM00871.ucs
871, 1149	1252, 5348	08711252.cnv, IBM00871.ucs
871, 1149	1275	08711275.cnv, IBM00871.ucs
871, 1149	1200, 1208, 13488, 17584	IBM00871.ucs

数据库服务器 **CCSID/CPGID**	数据库客户机 **CCSID/CPGID**	转换表文件
1275	850，858	12750850.cnv，08501275.cnv

拉丁语 — **2**:

数据库服务器 **CCSID/CPGID**	数据库客户机 **CCSID/CPGID**	转换表文件
852，9044	1250，5346	08521250.cnv，12500852.cnv
870，1153	852，9044	08700852.cnv，IBM00870.ucs
870，1153	1250，5346	08701250.cnv，IBM00870.ucs
870，1153	1200，1208，13488，17584	IBM00870.ucs
1250，5346	852，9044	12500852.cnv，08521250.cnv，IBM01250.ucs
1250，5346	1200，1208，13488，17584	IBM01250.ucs

简体中文:

数据库服务器 **CCSID/CPGID**	数据库客户机 **CCSID/CPGID**	转换表文件
837，935，1388	1200，1208，13488，17584	1388ucs2.cnv
1386	1200，1208，13488，17584	1386ucs2.cnv，ucs21386.cnv

繁体中文:

数据库服务器 **CCSID/CPGID**	数据库客户机 **CCSID/CPGID**	转换表文件
937，835，1371	950，1370	09370950.cnv，0937ucs2.cnv
937，835，1371	1200，1208，13488，17584	0937ucs2.cnv
1114，5210	850，858	11140850.cnv，08501114.cnv

泰国语:

数据库服务器 **CCSID/CPGID**	数据库客户机 **CCSID/CPGID**	转换表文件
838，1160	874，1161	08380874.cnv，IBM00838.ucs
838，1160	1200，1208，13488，17584	IBM00838.ucs

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
874, 1161	1200, 1208, 13488, 17584	IBM00874.ucs

土耳其语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
857, 9049	1254, 5350	08571254.cnv, 12540857.cnv
1026, 1155	857, 9049	10260857.cnv, IBM01026.ucs
1026, 1155	1254, 5350	10261254.cnv, IBM01026.ucs
1026, 1155	1200, 1208, 13488, 17584	IBM01026.ucs
1254, 5350	857, 9049	12540857.cnv, 08571254.cnv, IBM01254.ucs
1254, 5350	1200, 1208, 13488, 17584	IBM01254.ucs

乌克兰语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
1123, 1158	1124	11231124.cnv
1123, 1158	1125, 848	11231125.cnv
1123, 1158	1251, 5347	11231251.cnv
1124	1251, 5347	11241251.cnv, 12511124.cnv
1125, 848	1251, 5347	11251251.cnv, 12511125.cnv

Unicode:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
1200, 1208, 13488, 17584	813, 4909	IBM00813.ucs
1200, 1208, 13488, 17584	862, 867	IBM00862.ucs
1200, 1208, 13488, 17584	864, 17248	IBM00864.ucs

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
1200, 1208, 13488, 17584	874, 1161	IBM00874.ucs
1200, 1208, 13488, 17584	921, 901	IBM00921.ucs
1200, 1208, 13488, 17584	922, 902	IBM00922.ucs
1200, 1208, 13488, 17584	954	ucs20954.cnv, 0954ucs2.cnv
1200, 1208, 13488, 17584	1046, 9238	IBM01046.ucs
1200, 1208, 13488, 17584	1250, 5346	IBM01250.ucs
1200, 1208, 13488, 17584	1251, 5347	IBM01251.ucs
1200, 1208, 13488, 17584	1253, 5349	IBM01253.ucs
1200, 1208, 13488, 17584	1254, 5350	IBM01254.ucs
1200, 1208, 13488, 17584	1255, 5351	IBM01255.ucs
1200, 1208, 13488, 17584	1256, 5352	IBM01256.ucs
1200, 1208, 13488, 17584	1386	ucs21386.cnv, 1386ucs2.cnv

越南语:

数据库服务器 CCSID/CPGID	数据库客户机 CCSID/CPGID	转换表文件
1130, 1164	1258, 5354	11301258.cnv
1258, 5354	1129, 1163	12581129.cnv

相关概念:

- 『Character conversion』（*SQL Reference, Volume 1*）

相关任务:

- 第 222 页的『启用和禁用欧元符号支持』

代码页 923 和 924 的转换表

以下是与代码页 923 和 924 相关联的所有代码页转换表文件的列表。每个文件的格式都是 XXXXYYYY.cnv 或 ibmZZZZZ.ucs，其中 XXXXX 是源代码页号，而 YYYY 是目标代码页号。文件 ibmZZZZZ.ucs 支持代码页 ZZZZZ 和 Unicode 之间的转换。

要激活特定代码页转换表，将该转换表文件重命名或复制至其新名称（如在第二列中所示）。

例如，要在将 8859-1/15（Latin 1/9）客户机连接至 Windows 1252 数据库时支持欧元符号，则需要对 sqllib/conv/ directory 目录中的下列代码页转换表文件进行重命名或复制:

- 从 09231252.cnv 重命名或复制至 08191252.cnv
- 从 12520923.cnv 重命名或复制至 12520819.cnv
- 从 ibm00923.ucs 重命名或复制至 ibm00819.ucs

sqllib/conv/ 目录中的 923 和 924 转换表文件	**新名称**
00370923.cnv	00370819.cnv
02730923.cnv	02730819.cnv
02770923.cnv	02770819.cnv
02780923.cnv	02780819.cnv
02800923.cnv	02800819.cnv
02840923.cnv	02840819.cnv
02850923.cnv	02850819.cnv
02970923.cnv	02970819.cnv
04370923.cnv	04370819.cnv
05000923.cnv	05000819.cnv
08500923.cnv	08500819.cnv
08600923.cnv	08600819.cnv
08630923.cnv	08630819.cnv
08710923.cnv	08710819.cnv
09230437.cnv	08190437.cnv
09230500.cnv	08190500.cnv

sqllib/conv/ 目录中的 923 和 924 转换表文件	新名称
09230850.cnv	08190850.cnv
09230860.cnv	08190860.cnv
09230863.cnv	08190863.cnv
09231043.cnv	08191043.cnv
09231051.cnv	08191051.cnv
09231114.cnv	08191114.cnv
09231252.cnv	08191252.cnv
09231275.cnv	08191275.cnv
09241252.cnv	10471252.cnv
10430923.cnv	10430819.cnv
10510923.cnv	10510819.cnv
11140923.cnv	11140819.cnv
12520923.cnv	12520819.cnv
12750923.cnv	12750819.cnv
ibm00923.ucs	ibm00819.ucs

相关概念:

- 『Character conversion』（*SQL Reference, Volume 1*）

相关任务:

- 第 222 页的『启用和禁用欧元符号支持』

为数据库选择语言

创建数据库时，必须决定数据将以什么语言存储。创建数据库时，可指定国家或地区和代码集。国家或地区和代码集可以与当前操作系统设置不同。如果未在数据库创建时间显式选择国家或地区和代码集，则将使用当前语言环境创建数据库。如果正在选择代码集，则确保它可以对您将使用的语言中的所有字符进行编码。

另一选项是将数据存储在 Unicode 数据库中，这意味这不必选择特定的语言；Unicode 编码包括来自几乎世界上所有现有语言的字符。

DB2 管理服务器的语言环境设置

确保“DB2 管理服务器”实例的语言环境与 DB2 实例的语言环境兼容。否则，DB2 实例不能与“DB2 管理服务器”通信。

如果在“DB2 管理服务器”的用户概要文件中未设置 LANG 环境变量，则将使用缺省的系统语言环境来启动“DB2 管理服务器”。如果未定义缺省的系统语言环境，则将使用代码页 819 来启动“DB2 管理服务器”。如果 DB2 实例使用 DBCS 语言环境之一，而使用代码页 819 启动“DB2 管理服务器”，则该实例将不能与“DB2 管理服务器”通信。“DB2 管理服务器”的语言环境与 DB2 实例的语言环境必须兼容。

例如，在简体中文 Linux 系统中，应该在“DB2 管理服务器”的用户概要文件中设置 `LANG=zh_CN`。

启用双向支持

使用新的“编码字符集标识符”（CCSID）定义在 DB2 通用数据库中实现了双向格式转换。对于新的特定于双向的 CCSID，执行格式转换来代替代码页转换或在代码页转换之外执行它。要使用此支持，必须将 DB2BIDI 注册表变量设置为 YES。缺省情况下，不设置此变量。服务器对所有转换设置此变量，且只有在服务器启动后才能设置。因为存在附加检查和格式转换，将 DB2BIDI 设置为 YES 可能会影响性能。

限制:

下列限制适用:

- 若您选择对于您的客户机平台的代码页或字符串类型不适当的 CCSID，将得到意外的结果。若您选择了不兼容的 CCSID（例如，对于与阿拉伯语数据库的连接，选择了希伯来语 CCSID），或者若未为该服务器设置 DB2BIDI，当您尝试连接时会接收到错误消息。
- 对于 HOST EBCDIC 平台是客户机，而 DB2 UDB 是服务器的情况，不支持 CCSID 覆盖。

过程:

要在非 DRDA 环境中指定特定的双向 CCSID:

- 确保 DB2BIDI 注册表设置为 YES。
- 选择与客户机的特征相匹配的 CCSID，并将 DB2CODEPAGE 设置为该值。
- 若您已有与该数据库的一个连接，您必须发出 TERMINATE 命令，并再次连接以使 DB2CODEPAGE 的新设置生效。

对于 DRDA 环境，若 HOST EBCDIC 平台也支持这些双向的 CCSID，您只须设置 DB2CODEPAGE 值。然而，若 HOST 平台不支持这些 CCSID，则您还必须为将要连接的 HOST 数据库服务器指定一个 CCSID 覆盖。这是必须的，因为在 DRDA 环境中，代码页转换和格式转换由数据接收器执行。然而，若 HOST 服务器不支持这些双向 CCSID，它将不对从 DB2 UDB 接收的数据执行格式转换。若您使用 CCSID 覆盖，DB2 UDB 客户机一样对出站数据执行格式转换。

相关概念:

- 第 239 页的『DB2 Connect 的双向支持』
- 『处理 BiDi 数据』（*《DB2 Connect 用户指南》*）

相关参考:

- 第 236 页的『特定于双向的 CCSID』
- 『一般注册表变量』（*《管理指南: 性能》*）

特定于双向的 CCSID

下列双向属性是对不同平台上的双向数据作更正处理所需的:

- 文本类型
- 数字整形
- 定向
- 文本整形
- 对称交换

因为不同平台上的缺省值不一样，因此在将 DB2 数据从一个平台移到另一个平台时会出现问题。例如，Windows 操作系统使用 LOGICAL UNSHAPED 数据，而 z/OS 和 OS/390 通常使用 SHAPED VISUAL 数据。因而，若对双向属性没有任何支持，则从 DB2 通用数据库 OS/390 版和 z/OS 版发送至 Windows 32 位操作系统工作站上的 DB2 UDB 的数据将不能正确显示。

DB2 通过专门的双向“编码字符集标识符”（CCSID）来支持双向数据属性。已定义下列双向 CCSID，且对 DB2 UDB 实现了它们，如表 24 中所示。已定义 CDRA 字符串类型，如第 238 页的表 25 中所示。

表 24. 双向 CCSID

CCSID	代码页	字符串类型
420	420	4
424	424	4

表 24. 双向 CCSID (续)

CCSID	代码页	字符串类型
856	856	5
862	862	4
864	864	5
867	862	4
916	916	5
1046	1046	5
1089	1089	5
1255	1255	5
1256	1256	5
5351	1255	5
5352	1256	5
8612	420	5
8616	424	10
9048	856	5
9238	1046	5
12712	424	4
16804	420	4
17248	864	5
62208	856	4
62209	862	10
62210	916	4
62211	424	5
62213	862	5
62215	1255	4
62218	864	4
62220	856	6
62221	862	6
62222	916	6
62223	1255	6
62224	420	6
62225	864	6
62226	1046	6

表 24. 双向 CCSID (续)

CCSID	代码页	字符串类型
62227	1089	6
62228	1256	6
62229	424	8
62230	856	8
62231	862	8
62232	916	8
62233	420	8
62234	420	9
62235	424	6
62236	856	10
62237	1255	8
62238	916	10
62239	1255	10
62240	424	11
62241	856	11
62242	862	11
62243	916	11
62244	1255	11
62245	424	10
62246	1046	8
62247	1046	9
62248	1046	4
62249	1046	12
62250	420	12

表 25. CDRA 字符串类型

字符串类型	文本类型	数字整形	定向	文本整形	对称交换
4	可视	联通	LTR	已整形	关闭
5	隐式	阿拉伯数字	LTR	未整形	打开
6	隐式	阿拉伯数字	RTL	未整形	打开
7*	可视	联通	Contextual*	未整形连字	关闭
8	可视	联通	RTL	已整形	关闭

表 25. CDRA 字符串类型 (续)

字符串类型	文本类型	数字整形	定向	文本整形	对称交换
9	可视	联通	RTL	已整形	打开
10	隐式	阿拉伯数字	Contextual LTR	未整形	打开
11	隐式	阿拉伯数字	Contextual RTL	未整形	打开
12	隐式	阿拉伯数字	RTL	已整形	关闭

注: * 当第一个字母字符为拉丁字符时，字段方向为从左到右（LTR）；当第一个字母字符为双向（RTL）字符时，字段方向为从右到左（RTL）。字符未整形，但保留 LamAlef 字母结合，而没有将其分开。

相关概念:

- 第 239 页的『DB2 Connect 的双向支持』

相关任务:

- 第 235 页的『启用双向支持』

DB2 Connect 的双向支持

当在 DB2® Connect 和服务器上的数据库之间交换数据时，对入局数据执行转换的往往是接收方。除通常的代码页转换外，同样的约定通常也适用于双向格式转换。DB2 Connect™ 能够对它准备发送至服务器数据库的数据和从服务器数据库接收到的数据执行双向格式转换。

为了 DB2 Connect 能够对服务器数据库的出局数据执行双向格式转换，必须替换服务器数据库的双向 CCSID。为此，可在服务器数据库的 DCS 数据库目录条目的 PARMS 字段中使用 BIDI 参数。

注: 如果希望 DB2 Connect 对准备发送至 DB2 主机或 iSeries™ 数据库的数据执行格式转换，即使不必覆盖其 CCSID，也必须将 BIDI 参数添加至 DCS 数据库目录的 PARMS 字段。在这种情况下，应提供的 CCSID 是缺省 DB2 主机或 iSeries 数据库 CCSID。

将 BIDI 参数指定为 PARMS 字段中的第九个参数，同时指定双向 CCSID，希望替换缺省的服务器数据库双向 CCSID:

```
",,,,,,,,BIDI=xyz"
```

其中 *xyz* 是 CCSID 覆盖

注: 要使 BIDI 参数生效，必须将注册表变量 DB2BIDI 设置为 YES。

为了最好地描述如何使用此功能部件，举例如下。

假设您有一个运行 CCSID 62213（双向字符串类型 5）的希伯莱语版 DB2 的客户机，而您又想存取运行 CCSID 00424（双向字符串类型 4）的 DB2 主机或 iSeries 数据库。但您知道包含在 DB2 主机或 iSeries 数据库中的数据基于 CCSID 08616（双向字符串类型 6）。

这样，就会有两个问题: 第一个问题是 DB2 主机或 iSeries 数据库不能区分 CCSID 00424 和 08616 的双向字符串类型。第二个问题是 DB2 主机或 iSeries 数据库不识别 DB2 客户机 CCSID（62213）。它只支持 CCSID 00862，它与 CCSID 62213 基于相同的代码页。

需要确保发送至 DB2 主机或 iSeries 数据库的数据以双向字符串类型 6 格式开头，并且要让 DB2 Connect 知道它必须对从 DB2 主机或 iSeries 数据库接收到的数据执行双向转换。需要对 DB2 主机或 iSeries 数据库使用以下编目命令:

```
db2 catalog dcs database nydb1 as telaviv parms ",,,,,,,,BIDI=08616"
```

此命令告诉 DB2 Connect 使用 CCSID 08616 覆盖 DB2 主机或 iSeries 数据库的 CCSID 00424。此覆盖包括下列处理:

1. DB2 Connect 连接至使用 CCSID 00862 的 DB2 主机或 iSeries 数据库。
2. DB2 Connect 对它准备发送至 DB2 主机或 iSeries 数据库的数据执行双向格式转换。即从 CCSID 62213（双向字符串类型 5）转换为 CCSID 62221（双向字符串类型 6）。
3. DB2 Connect 对它从 DB2 主机或 iSeries 数据库*接收*到的数据执行双向格式转换。即从 CCSID 08616（双向字符串类型 6）转换为 CCSID 62213（双向字符串类型 5）。

注: 有些情况下，使用双向 CCSID 可能造成 SQL 查询本身被修改，结果使 DB2 服务器不能识别该 SQL 查询。特别是，当可以使用另一种字符串类型时，应避免使用 IMPLICIT CONTEXTUAL CCSID 和 IMPLICIT RIGHT-TO-LEFT CCSID。若 SQL 查询包括引用字符串，则 CONTEXTUAL CCSID 可能产生不可预测的结果。避免在 SQL 语句中使用引用字符串；尽可能使用主机变量。

若某个特定的双向 CCSID 导致不能按照下列建议改正的问题，则将 DB2BIDI 设置为 NO。

相关概念:

- 『处理 BiDi 数据』（*《DB2 Connect 用户指南》*）

相关参考:

- 第 236 页的『特定于双向的 CCSID』

整理顺序

数据库管理器使用*整理顺序*来比较字符数据这是对一组字符的排序，确定某个字符的 ASCII 码是大于、小于还是等于另一个字符的 ASCII 码。

注: 用 FOR BIT DATA 属性定义的字符串数据和 BLOB 数据使用二进制排序顺序进行排序。

例如，可使用整理顺序来指示特定字符的小写与大写的 ASCII 码相等。

数据库管理器允许以定制整理顺序创建数据库。以下各节帮助您确定和实现数据库的特定整理顺序。

数据库中的每个单字节字符在内部表示为 0 与 255 之间（在十六进制表示中，在 X'00' 与 X'FF' 之间）的唯一数字。此数字又称字符的*代码点*；将数字指定为字符的集合统称为*代码页*。整理顺序是代码点与排序顺序中每个字符的期望位置之间的映射。该位置的数字值在整理顺序中称为字符的*权重*。在最简单的整理顺序中，权重与代码点完全相同。这称为*标识顺序*。

例如，假定字符 B 和 b 分别具有代码点 X'42' 和 X'62'。如果（根据整理顺序表）它们都具有排序权重 X'42'（B），则它们的整理顺序相同。如果 B 的排序权重为 X'9E'，而 b 的排序权重为 X'9D'，则排序时 b 将在 B 之前。整理顺序表指定每个字符的权重。该表不同于代码页，后者指定每个字符的代码点。

考虑以下示例。ASCII 字符 A 到 Z 由 X'41' 到 X'5A' 表示。要描述用来对这些字符连续排序（没有干扰字符）的整理顺序，可以这样写: X'41'，X'42'，... X'59'，X'5A'。

多字节字符的十六进制值又用作权重。例如，假定双字节字符 A 和 B 的代码点分别为 X'8260' 和 X'8261'，则使用 X'82'、X'60' 和 X'61' 的整理权重来根据这两个字符的代码点来进行排序。

整理顺序中的权重不一定是唯一的。例如，可对同一字母的大小写指定相同权重。

如果整理顺序提供所有 256 个代码点的权重，指定整理顺序将会非常简单。每个字符的权重可使用字符的代码点来确定。

在所有情况下，DB2® 使用在创建数据库时指定的整理表。如果想要多字节字符以它们出现在代码点表中的次序进行排序，在创建数据库时必须将 IDENTITY 指定为整理顺序。

注：对于 GRAPHIC 字段中的双字节和 Unicode 字符，排序顺序始终是 IDENTITY。

一旦定义了整理顺序，就将用该整理顺序执行该数据库将来的所有字符比较。除了定义为 FOR BIT DATA 或 BLOB 数字的字符之外，将对所有 SQL 比较和 ORDER BY 子句使用整理顺序，在设置索引和统计信息时也会使用整理顺序。

下列情况下可能会出现潜在问题:

- 应用程序将数据库中已排序的数据与应用程序的数据合并在一起，应用程序数据是使用不同的整理顺序排序的。
- 应用程序将一个数据库中已排序的数据与另一个数据库中已排序的数据合并在一起，但这两个数据库的整理顺序不同。
- 应用程序对排序的数据所作的假定不符合相关的整理顺序。例如，数字排在字母之前对特定的整理顺序可能适用也可能不适用。

最后一点要记住的是，对字符代码点进行直接比较所得到的任何排序的结果将只与使用等同的整理顺序排序的查询结果匹配。

相关概念:

- 『Character conversion』（*SQL Reference, Volume 1*）
- 『Character Comparisons Based on Collating Sequences』（*Application Development Guide: Programming Client Applications*）

整理泰国语字符

泰国语包含特殊元音（“前导元音”）、音调符号和其它不按顺序排序的特殊字符。

限制:

必须以泰国语语言环境和代码集创建数据库。

过程:

创建数据库时，在 CREATE DATABASE 命令上使用 COLLATE USING NLSCHAR 子句。

相关概念:

- 第 241 页的『整理顺序』

相关参考:

- 『CREATE DATABASE Command』（*Command Reference*）

基于国家或地区代码的日期和时间格式

用字符串表示的日期和时间格式是与应用程序的国家或地区代码相关的日期时间值的缺省格式。当预编译程序或将其绑定至数据库时，可指定 DATETIME 格式选项来覆盖此缺省格式。

以下是日期和时间的输入和输出格式的描述:

- 输入时间格式
  - 无缺省输入时间格式。
  - 对于所有国家或地区代码，允许所有时间格式作为输入。
- 输出时间格式
  - 缺省输出时间格式等于本地时间格式。
- 输入日期格式
  - 无缺省输入日期格式。
  - 在日期的本地格式与 ISO、JIS、EUR 或 USA 日期格式冲突之处，将把本地格式作为日期输入格式。例如，查看表 26 中的 UK 条目。
- 输出日期格式
  - 在表 26 中显示了缺省输出日期格式。

    注: 表 26 还显示了各种国家或地区代码的字符串格式的列表。

表 26. 基于国家或地区代码的日期和时间格式

国家或地区代码	本地日期格式	本地时间格式	缺省输出日期格式	输入日期格式
355 阿尔巴尼亚	yyyy-mm-dd	JIS	LOC	LOC, USA, EUR, ISO
785 阿拉伯	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
001 澳大利亚（1）	mm-dd-yyyy	JIS	LOC	LOC, USA, EUR, ISO
061 澳大利亚	dd-mm-yyyy	JIS	LOC	LOC, USA, EUR, ISO

表 26. 基于国家或地区代码的日期和时间格式 (续)

国家或地区代码	本地日期格式	本地时间格式	缺省输出日期格式	输入日期格式
032 比利时	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
055 巴西	dd.mm.yyyy	JIS	LOC	LOC, EUR, ISO
359 保加利亚	dd.mm.yyyy	JIS	EUR	LOC, USA, EUR, ISO
001 加拿大	mm-dd-yyyy	JIS	USA	LOC, USA, EUR, ISO
002 加拿大（法语区）	dd-mm-yyyy	ISO	ISO	LOC, USA, EUR, ISO
385 克罗地亚	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
042 捷克共和国	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
045 丹麦	dd-mm-yyyy	ISO	ISO	LOC, USA, EUR, ISO
358 芬兰	dd/mm/yyyy	ISO	EUR	LOC, EUR, ISO
389 FYR 马其顿	dd.mm.yyyy	JIS	EUR	LOC, USA, EUR, ISO
033 法国	dd/mm/yyyy	JIS	EUR	LOC, EUR, ISO
049 德国	dd/mm/yyyy	ISO	ISO	LOC, EUR, ISO
030 希腊	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
036 匈牙利	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
354 冰岛	dd-mm-yyyy	JIS	LOC	LOC, USA, EUR, ISO
091 印度	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
972 以色列	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
039 意大利	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
081 日本	mm/dd/yyyy	JIS	ISO	LOC, USA, EUR, ISO
082 韩国	mm/dd/yyyy	JIS	ISO	LOC, USA, EUR, ISO
001 拉丁美洲 (1)	mm-dd-yyyy	JIS	LOC	LOC, USA, EUR, ISO
003 拉丁美洲	dd-mm-yyyy	JIS	LOC	LOC, EUR, ISO

表 26. 基于国家或地区代码的日期和时间格式 (续)

国家或地区代码	本地日期格式	本地时间格式	缺省输出日期格式	输入日期格式
031 荷兰	dd-mm-yyyy	JIS	LOC	LOC, USA, EUR, ISO
047 挪威	dd/mm/yyyy	ISO	EUR	LOC, EUR, ISO
048 波兰	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
351 葡萄牙	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
086 中国	mm/dd/yyyy	JIS	ISO	LOC, USA, EUR, ISO
040 罗马尼亚	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
007 俄罗斯	dd/mm/yyyy	ISO	LOC	LOC, EUR, ISO
381 塞尔维亚 / 蒙的内哥罗	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
042 斯洛伐克	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
386 斯洛文尼亚	yyyy-mm-dd	JIS	ISO	LOC, USA, EUR, ISO
034 西班牙	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
046 瑞典	dd/mm/yyyy	ISO	ISO	LOC, EUR, ISO
041 瑞士	dd/mm/yyyy	ISO	EUR	LOC, EUR, ISO
088 台湾	mm-dd-yyyy	JIS	ISO	LOC, USA, EUR, ISO
066 泰国 (2)	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
090 土耳其	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
044 英国	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO
001 美国	mm-dd-yyyy	JIS	USA	LOC, USA, EUR, ISO
084 越南	dd/mm/yyyy	JIS	LOC	LOC, EUR, ISO

注:

1. 将国家或地区代码 001 指定给使用缺省 C 语言环境的国家或地区。
2. 佛教纪元的 yyyy 等价于罗马教的 yyyy + 543 年（限于泰国）。

相关参考:

- 『BIND Command』（*Command Reference*）
- 『PRECOMPILE Command』（*Command Reference*）

Unicode 字符编码

Unicode 字符编码标准是定长的字符编码模式，它包含了世界上几乎所有现用语言的字符。

有关 Unicode 的更多信息可在最新版本的 Unicode Standard 一书中找到，并且可从 “Unicode 合作”（Unicode Consortium）Web 站点（www.unicode.org）中找到。

Unicode 使用两种编码格式：8 位和 16 位。缺省编码格式是 16 位，即每个字符是 16 位（两个字节）宽，并且通常显示为 U+hhhh，其中 hhhh 是字符的十六进制代码点。生成的 65 000+ 代码元素足以用于编码世界上主要语言的大多数字符，Unicode 标准还提供了一种扩展机制，允许编码一百多万个字符。扩展机制使用一对高位和低位代用字符来对扩展字符或补充字符进行编码。第一个（或高位）代用字符具有 U+D800 和 U+DBFF 之间的代码值，而第二个（或低位）代用字符具有 U+DC00 和 U+DFFF 之间的代码值。

UCS-2

“国际标准化组织”（ISO）和 “国际电工委员会”（IEC）标准 10646（ISO/IEC 10646）指定了 “通用多八位字节编码字符集”（UCS），该编码字符集有一个 16 位（双字节）版本 (UCS-2) 和一个 32 位（四字节）版本（UCS-4）。UCS-2 相当于没有代用字符的 Unicode 16 位格式。UCS-2 可以对 Unicode 版本 3.0 指令表中定义的所有（16 位）字符进行编码。两个 UCS-2 字符 — 一个高位后面跟随一个低位代理 — 需要对从 Unicode 版本 3.1 开始引入的每个新补充的字符进行编码。这些补充字符在原始的 16 位 “基本多语言平面”（BMP 或 平面 0）外部定义。

UTF-8

对于面向字节基于 ASCII 的应用程序和文件系统，16 位 Unicode 字符是引起问题的主要因素。例如，不明白 Unicode 的应用程序可能会将大写字符 “A”（U+0041）的 8 个前导零位误解为单字节 ASCII NULL 字符。

UTF-8（UCS 变换格式 8）是一种算法变换，它将定长 Unicode 字符变换为变长 ASCII 安全的字节字符串。在 UTF-8 中，ASCII 和控制字符由通常的单字节代码表示，但其它字符变为双字节或更多字节。UTF 8 可以对非补充和补充字符进行编码。

UTF 16

ISO/IEC 10646 还定义了一种扩展技术，以使用两个 UCS-2 字符来编码某些 UCS-4 字符。此扩展技术称为 UTF-16，它相当于有代用字符的 Unicode 16 位编码格式。总之，UTF-16 字符指令集由所有 UCS-2 字符和可通过代用字符存取的附加的一百万字符构成。

当将 16 位 Unicode 字符序列化为字节时，某些处理器将最重要的字节放置在初始位置（称为大尾数法排序），而其它的处理器则首先放置最不重要的字节（称为小尾数法排序）。Unicode 的缺省字节排序是大尾数法。

使用 UTF-8 格式的每个 UTF-16 字符的字节数可根据表 27 来确定。

表 27. UTF-8 的位分布

代码值（二进制）	UTF-16（二进制）	第一个字节（二进制）	第二个字节（二进制）	第三个字节（二进制）	第四个字节（二进制）
00000000 0xxxxxxx	00000000 0xxxxxxx	0xxxxxxx			
00000yyy yyxxxxxx	00000yyy yyxxxxxx	110yyyyy	10xxxxxx		
zzzzyyyy yyxxxxxx	zzzzyyyy yyxxxxxx	1110zzzz	10yyyyyy	10xxxxxx	
uuuuu zzzzyyyy yyxxxxxx	110110ww wwzzzzyy 110111yy yyxxxxxx	11110uuu (where uuuuu = wwww+1)	10uuzzzz	10yyyyyy	10xxxxxx

在以上每一项中，u、w、x、y 和 z 串都是字符的位表示。例如，U+0080 变换为二进制中的 11000010 10000000，而代用字符对 U+D800 U+DC00 变为二进制中的 11110000 10010000 10000000 10000000。

相关概念:

- 第 248 页的『DB2 中的 Unicode 实现』
- 第 250 页的『数据类型的 Unicode 处理』
- 第 252 页的『Unicode 文字』

相关任务:

- 第 251 页的『创建 Unicode 数据库』

DB2 中的 Unicode 实现

DB2® UDB 支持 UTF 8 和 UCS-2，即，没有代用字符的 Unicode。

当创建 Unicode 数据库时，CHAR、VARCHAR、LONG VARCHAR 和 CLOB 数据以 UTF-8 格式存储，而 GRAPHIC、VARGRAPHIC、LONG VARGRAPHIC 和 DBCLOB 数据以 UCS-2 格式存储。

在版本 7.2 修订包 4 之前的 DB2 UDB 版本中，DB2 UDB 将代用字符对中的两个字符看作两个独立的 Unicode 字符。因此，将该代用字符对从 UTF-16/UCS-2 变换为 UTF-8 会生成两个三字节序列。从 DB2 UDB 版本 7.2 修订包 4 开始，DB2 UDB 在 UTF-16/UCS-2 和 UTF-8 之间变换时识别代用字符对，因此一对 UTF-16 代用字符将会变为一个 UTF-8 四字节序列。在其它用法中，DB2 继续将代用字符对作为两个独立的 UCS 2 字符对待。只要知道如何区分补充字符与非补充字符，就可以安全地将补充字符存储在 DB2 Unicode 数据库中。

DB2 UDB 将每个 Unicode 字符看作一个单独字符，其中包括诸如 COMBINING ACUTE ACCENT 字符（U+0301）的那些（非空格）字符。因此，DB2 UDB 将认为字符 LATIN SMALL LETTER A WITH ACUTE（U+00E1）照规范来讲与后跟字符 COMBINING ACUTE ACCENT（U+0301）的字符 LATIN SMALL LETTER A（U+0061）不是等价的。

UCS-2 Unicode 数据库的缺省整理序列是 IDENTITY，它根据代码点对字符进行排序。因此，缺省情况下，将根据代码点来排序和比较所有 Unicode 字符。对于非补充 Unicode 字符，当使用 UTF-8 和 UCS-2 编码时，它们的二进制整理次序是相同的。但是，如果任何补充字符需要一对代用字符才能进行编码，则在以 UTF-8 编码时将把该字符向末尾整理，但是使用UCS-2 编码时将把同一字符向中间的某处整理，并且可以将它的两个代用字符分开。原因在于当以 UTF-8 编码时，扩展字符具有四字节二进制代码值 11110xxx 10xxxxxx 10xxxxxx 10xxxxxx，它大于 UTF-8 编码 U+FFFF，即 X'EFBFBF'。但是在 UCS 中，相同的补充字符被编码为一对 UCS-2 高位和低位代用字符，并且具有二进制格式 1101 1000 xxxx xxxx 1101 1100 xxxx xxxx，它小于 UCS-2 编码 U+FFFF。

还可以使用 IDENTITY_16BIT 整理选项创建 Unicode 数据库。IDENTITY_16BIT 与缺省 IDENTITY 整理选项不同，Unicode 数据库中的 CHAR、VARCHAR、LONG VARCHAR 和 CLOB 数据将使用 CESU-8 二进制次序而不是 UTF-8 二进制次序进行整理。CESU-8 即 *UTF-16 的兼容性编码模式:*

8 位，并且就此规范而言，它的规范包含在 Draft Unicode Technical Report #26 中，后者可在 Unicode Technical Consortium Web 站点（www.unicode.org）上获得。除 Unicode 补充字符之外，CESU-8 是与 UTF-8 完全相同的二进制字符，那些补充字符是在 16 位“基本多语言平面”（BMP 或平面 0）之外定义的。在 UTF-8 编码中，补充字符由一个四字节的序列表示，但 CESU-8 中的相同字符需要两个三字节的序列。使用 IDENTITY_16BIT 整理选项对字符和图形数据会生成相同的整理次序。注意，即使指定了 IDENTITY_16BIT 选项，CHAR、VARCHAR、LONG VARCHAR 和 CLOB 数据仍然以 UTF-8 格式存储。

所有与文化相关的参数，如日期或时间格式、十进制分隔符及其它参数等等，都是以客户机的当前国家或地区为基础。

Unicode 数据库允许与 DB2 UDB 支持的所有代码页连接。客户机代码页和 UTF-8 之间的代码页字符转换由数据库管理器自动执行。图形字符串类型的数据通常使用 UCS-2，不必通过代码页转换。命令行处理器（CLP）环境是个例外。若从 CLP 选择图形字符串（UCS-2）数据，则返回的图形字符串数据将从 UCS-2 转换为客户机环境的代码页。

任何客户机均受字符编码系统、输入方法和它的环境所支持的字体限制，但是 UCS-2 数据库本身接受并存储所有 UCS-2 字符。因此，每个客户机通常使用 UCS-2 字符的一个子集，但数据库管理器允许使用 UCS-2 字符的整个编码系统。

当字符由本地代码页转换为 UTF-8 时，字节数可能增多。ASCII 字符不会增多，但其它 UCS-2 字符将增大两三倍。

代码页 / CCSID 号码

在 IBM 内部，已将 UCS-2 代码页注册为代码页 1200，它具有不断增加的字符集，也就是说当向一个代码页添加新字符时，代码页号码不变。代码页 1200 始终引用 Unicode 的当前版本。

UCS 标准的特定版本由 Unicode 2.0 和 ISO/IEC 10646-1 定义，在 IBM®内部已将它注册为 CCSID 13488。此 CCSID 已在 DB2 UDB 内部使用，用于存储 euc-Japan 和 euc-Taiwan 数据库中的字符串数据。CCSID 13488 和代码页 1200 都引用 UCS-2，除了它们的“双字节”(DBCS) 空格的值不同外，其处理方式是相同的:

CP/CCSID	单字节（SBCS）空间	双字节（DBCS）空间
1200	N/A	U+0020
13488	N/A	U+3000

注意: 在 UCS-2 数据库中，U+3000 没有任何特殊意义。

关于转换表，由于代码页 1200 是 CCSID 13488 的超集，所以对它们使用了相同的（超集）表。

在 IBM 中，已将 UTF-8 注册为具有增长的字符集的 CCSID 1208（有时也称为代码页 1208）。当向标准集中添加新字符，此号码（1208）不会改变。

MBCS 代码页号码为 1208，它是数据库代码页号码以及该数据库中字符串的代码页。UCS-2 的双字节代码页号码为 1200，它是数据库中的图形字符串数据的代码页。

相关概念:

- 第 246 页的『Unicode 字符编码』
- 第 250 页的『数据类型的 Unicode 处理』
- 第 252 页的『Unicode 文字』

相关任务:

- 第 251 页的『创建 Unicode 数据库』

数据类型的 Unicode 处理

DB2® UDB 所支持的所有数据类型在 UCS-2 数据库中也受支持。特别是，UCS-2 数据库支持图形字符串数据，并以 UCS-2／Unicode 存储。每个客户机（包括 SBCS 客户机）与 UCS-2 数据库连接时，可使用 UCS-2／Unicode 格式的图形字符串数据类型。

UCS-2 数据库同任何 MBCS 数据库一样，其字符串数据以字节数计。当使用 UTF-8 格式的字符串数据时，不应当认为每个字符都是单字节。在多字节 UTF-8 编码中，每个 ASCII 字符都是单字节，但每个非 ASCII 字符为两个至四个字节。当定义 CHAR 字段时应考虑这点。取决于 ASCII 与非 ASCII 字符的比率，一个大小为 *n* 字节的 CHAR 字段可包含 *n*/3 到 *n* 之间的任意多个字符。

对图形字符串 UCS-2 数据类型使用字符串 UTF-8 编码也对总存储器需求有影响。对于大多数字符是 ASCII 但其间插入一些非 ASCII 字符的情况，存储 UTF-8 数据可能是比较好的替代方法，因为存储器需求接近每个字符一个字节。另一方面，在大多数字符是扩充为三字节或四字节 UTF-8 序列的非 ASCII 字符（例如，表意字符）的情况下，UCS-2 图形字符串格式可能是比较好的替代方法，因为每个三字节 UTF-8 序列变为一个 16 位 UCS-2 字符，而每个四字节 UTF-8 序列变为两个 16 位 UCS-2 字符。

在 MBCS 环境中处理字符串的 SQL 函数，如 LENGTH、SUBSTR、POSSTR、MAX 和 MIN 等，是对“字节”数而不是“字

符”数进行运算的。该行为在 UCS-2 数据库中也一样，但当为 UCS-2 数据库指定偏移量和长度时应格外小心，因为始终是在数据库代码页的上下文中定义这些值。也就是说，对于 UCS-2 数据库，应在 UTF-8 中定义这些值。因为一些单字节字符需要 UTF-8 格式的多字节，因此对单字节数据库有效的 SUBSTR 索引可能对 UCS-2 数据库无效。若指定了不正确的索引，则将返回 SQLCODE -191（SQLSTATE 22504）。

在用户程序中，（C 语言中的）char 数据类型支持 SQL CHAR 数据类型。在用户程序中，sqldbchar 支持 SQL GRAPHIC 数据类型。注意，对于 UCS-2 数据库，sqldbchar 数据始终使用大尾结构（高字节在前）格式。当将应用程序与 UCS-2 数据库连接时，由 DB2 UDB 在应用程序代码页和 UTF-8 之间转换字符串数据，而图形字符串数据始终是 UTF-2 格式。

相关概念:

- 第 246 页的『Unicode 字符编码』
- 第 248 页的『DB2 中的 Unicode 实现』

创建 Unicode 数据库

过程:

缺省情况下，用创建数据库的应用程序的代码页创建数据库。因此，若从 Unicode（UTF-8）客户机（例如，AIX 的 UNIVERSAL 语言环境或者若将客户机上的 DB2CODEPAGE 注册表变量设置为 1208）创建数据库，您的数据库将创建为 Unicode 数据库。或者，可显式地将 “UTF-8” 指定为 CODESET 名，并使用 DB2 UDB 支持的任何有效 TERRITORY 代码。

要使用美国国家或地区代码来创建 Unicode 数据库:

```
DB2 CREATE DATABASE dbname USING CODESET UTF-8 TERRITORY US
```

要使用 **sqlecrea** API 创建 Unicode 数据库，应在 *sqledbterritoryinfo* 中相应地设置值。例如，将 SQLDBCODESET 设置为 UTF-8，将 SQLDBLOCALE 设置为任何有效的区域代码（例如 US）。

相关概念:

- 第 248 页的『DB2 中的 Unicode 实现』

相关参考:

- 『sqlecrea - Create Database』（*Administrative API Reference*）
- 『CREATE DATABASE Command』（*Command Reference*）

Unicode 文字

可以用下列两种方式指定 Unicode 文字:

- 作为图形字符串常量，使用 G'...' 或 N'....' 格式。用这种方式指定的任何文字将由数据库管理器从应用程序代码页转换为 16 位 Unicode。
- 作为 Unicode 十六进制字符串，使用 UX'....' 或 GX'....' 格式。在大尾数法排序中，在 UX 或 GX 后面的引号内指定的常量必须是 4 个十六进制数字的倍数。每个四位组表示一个 16 位 Unicode 代码点。注意代用字符始终成对出现，因此需要两个四位组来表示高位和低位代用字符。

当使用命令行处理器（CLP） 时，若本地应用程序代码页中存在 UCS-2 字符，则第一种方法比较容易（例如，从使用代码页 850 的终端输入代码页 850 的任何字符）。对于在应用程序代码页编码系统之外的字符，应使用第二种方法（例如，从使用代码页 850 的终端指定日语字符）。

相关概念:

- 第 246 页的『Unicode 字符编码』
- 第 248 页的『DB2 中的 Unicode 实现』

相关参考:

- 『Constants』（*SQL Reference, Volume 1*）

Unicode 数据库中的字符串比较

模式匹配是现有的 MBCS 数据库的行为与 UCS-2 数据库的行为稍微不同的一个范畴。

对于 DB2® UDB 中的 MBCS 数据库，当前行为如下所示: 若匹配表达式包含 MBCS 数据，则模式可以包含 SBCS 和非 SBCS 字符。该模式中的特殊字符解释如下:

- SBCS 下划线字符是一个 SBCS 字符。
- DBCS 下划线字符是一个 MBCS 字符。
- 百分比字符（SBCS 或 DBCS）表示由零个或多个 SBCS 或非 SBCS 字符组成的字符串。

在 Unicode 数据库中，“单字节”与“双字节”字符之间没有任何真正的差别; 每个 16 位字符占用两个字节。尽管 UTF-8 格式是 Unicode 字符的“混合字节”编码，但 UTF-8 格式的 SBCS 和 MBCS 字符之间没有任何真正的区别。每个字符是一个 Unicode 字符，而与采用 UTF-8 格式的字节数无关。当指定字符串或图形

字符串表达式时，下划线表示一个 Unicode 字符，而百分比符号表示由零个或多个 Unicode 字符组成的字符串。需要两条下划线才能与一个扩展 GRAPHIC 字符相匹配，因为一个扩展 GRAPHIC 字符在 GRAPHIC 列中由两个 UCS-2 字符表示。

在客户机端，字符串表达式使用客户机的代码页，将由数据库管理器转换为 UTF-8。SBCS 客户机代码页没有 DBCS 百分比字符或 DBCS 下划线字符，但每个受支持的代码页都包含一个单字节百分比字符（与 U+0025 对应）和一个单字节下划线字符（与 U+005F 对应）。UCS-2 数据库的特殊字符解释如下:

- SBCS 下划线字符（与 U+0025 对应）在图形字符串表达式中是一个 UCS-2 字符，而在字符串表达式中是一个 UTF-8 字符。
- SBCS 百分比符号（与 U+005F 对应）在图形字符串表达式中表示由零个或多个 UCS-2 字符组成的字符串，而在字符串表达式中表示由零个或多个 UTF-8 字符组成的字符串。

DBCS 代码页还支持 DBCS 百分比符号（与 U+FF05 对应）和 DBCS 下划线字符（与 U+FF3F 对应）。对于 UCS-2 数据库，这些字符没有任何特殊的意义。

对于可选的“转义表达式”，它指定一个用来修改下划线字符和百分比字符特殊意义的字符，仅支持 ASCII 字符或扩充为两字节 UTF-8 序列的字符。若指定扩充为三字节 UTF-8 值的转义字符，则将返回错误消息（错误 SQL0130N 和 SQLSTATE 22019）。

相关概念:

- 第 246 页的『Unicode 字符编码』
- 第 248 页的『DB2 中的 Unicode 实现』

相关参考:

- 『Character strings』（*SQL Reference, Volume 1*）
- 『Graphic strings』（*SQL Reference, Volume 1*）

附录 C. “DB2 通用数据库”技术信息

“DB2 通用数据库”技术信息概述

可以下列格式获取“DB2 通用数据库”技术信息:

- 书籍（PDF 和硬拷贝格式）
- 主题树（HTML 格式）
- DB2 工具的帮助（HTML 格式）
- 样本程序（HTML 格式）
- 命令行帮助
- 教程

本节是有关所提供技术信息以及可如何访问这些信息的概述。

DB2 文档的修订包

IBM 可能会阶段性地提供文档修订包。文档修订包使您可以在新信息可供使用时更新从 *DB2 HTML 文档 CD* 中安装的信息。

注: 如果您安装了文档修订包，则您的 HTML 文档将包含比 DB2 的印刷或联机 PDF 手册更新的信息。

DB2 技术信息类别

DB2 技术信息是按下列标题分类的:

- 核心 DB2 信息
- 管理信息
- 应用程序开发信息
- 商务智能信息
- DB2 Connect 信息
- 入门信息
- 教程信息
- 可选组件信息
- 发行说明

对于 DB2 资料库中的每本书，下表描述了订购硬拷贝、打印或查看 PDF 或者找出该书的 HTML 目录所需的信息。DB2 资料库中每本书的完整描述可从 IBM 出版物中心（IBM Publications Center）获得，网址为 www.ibm.com/shop/publications/order。

HTML 文档 CD 的安装目录对于各个信息类别来说是不同的:

htmlcdpath/doc/htmlcd/*%L*/*category*

其中:

- *htmlcdpath* 是安装了 HTML CD 的目录。
- *%L* 是语言标识符。例如，en_US。
- *category* 是类别标识符。例如，core 表示核心 DB2 信息。

在下表中的 PDF 文件名列中，文件名第六个位置的字符指示书籍的语言版本。例如，文件名 db2d1e80 标识英文版本的《*管理指南: 计划*》，而文件名 db2d1g80 标识该书的德语版本。下列字母用在文件名的第六个字符处以指示语言版本:

语言	标识符
阿拉伯语	w
巴西葡萄牙语	b
保加利亚语	u
克罗地亚语	9
捷克语	x
丹麦语	d
荷兰语	q
英语	e
芬兰语	y
法语	f
德语	g
希腊语	a
匈牙利语	h
意大利语	i
日语	j
韩国语	k
挪威语	n
波兰语	p
葡萄牙语	v
罗马尼亚语	8
俄语	r
简体中文	c
斯洛伐克语	7
斯洛文尼亚语	l
西班牙语	z

瑞典语	s
繁体中文	t
土耳其语	m

无书号指示该书只有联机版本而没有印刷版本。

核心 DB2 信息

此类别中的信息包括对所有 DB2 用户都很重要的 DB2 主题。不管您是程序员、数据库管理员或您将使用 DB2 Connect、DB2 仓库管理器或其它 DB2 产品，都将会发现此类别中的信息很有用。

此类别的安装目录为 `doc/htmlcd/%L/core`。

表 28. 核心 DB2 信息

书名	书号	PDF 文件名
IBM DB2 Universal Database Command Reference	SC09-4828	db2n0x80
《IBM DB2 通用数据库词汇表》	无书号	db2t0c80
《IBM DB2 通用数据库主索引》	S152-0192	db2w0c80
《IBM DB2 通用数据库消息参考第 1 卷》	G152-0177	db2m1c80
《IBM DB2 通用数据库消息参考第 2 卷》	G152-0178	db2m2c80
《IBM DB2 通用数据库新增内容》	S152-0176	db2q0c80

管理信息

此类别中的信息包括有效地设计、实现和维护 DB2 数据库、数据仓库和联合系统所需的那些主题。

此类别的安装目录为 `doc/htmlcd/%L/admin`。

表 29. 管理信息

书名	书号	PDF 文件名
《IBM DB2 通用数据库管理指南: 计划》	S152-0167	db2d1c80

表 29. 管理信息 (续)

书名	书号	PDF 文件名
《IBM DB2 通用数据库管理指南: 实现》	S152-0165	db2d2c80
《IBM DB2 通用数据库管理指南: 性能》	S152-0166	db2d3c80
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0x80
IBM DB2 Universal Database Data Movement Utilities Guide and Reference	SC09-4830	db2dmx80
《IBM DB2 通用数据库数据恢复和高可用性指南与参考大全》	S152-0181	db2hac80
《IBM DB2 通用数据库数据仓库中心管理指南》	S152-0188	db2ddc80
IBM DB2 Universal Database Federated Systems Guide	GC27-1224	db2fpx80
《IBM DB2 通用数据库管理和开发 GUI 工具指南》	S152-0180	db2atc80
IBM DB2 Universal Database Replication Guide and Reference	SC27-1121	db2e0x80
《IBM DB2 安装和管理卫星环境》	G152-0272	db2dsc80
IBM DB2 Universal Database SQL Reference, Volume 1	SC09-4844	db2s1x80
IBM DB2 Universal Database SQL Reference, Volume 2	SC09-4845	db2s2x80
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847	db2f0x80

应用程序开发信息

此类别中的信息对于应用程序开发者或使用 DB2 的程序员特别有用。将可找到有关受支持的语言和编译器的信息，以及使用各种受支持的编程接口（如嵌入式 SQL、ODBC、JDBC、SQLj 和 CLI）访问 DB2 所需的文档。如果您联机查看 HTML 格式的此信息，则还可以访问一组 HTML 格式的 DB2 样本程序。

此类别的安装目录为 doc/htmlcd/%L/ad。

表 30. 应用程序开发信息

书名	书号	PDF 文件名
《IBM DB2 通用数据库应用程序开发指南: 构建和运行应用程序》	S152-0168	db2axc80
IBM DB2 Universal Database Application Development Guide: Programming Client Applications	SC09-4826	db2a1x80
IBM DB2 Universal Database Application Development Guide: Programming Server Applications	SC09-4827	db2a2x80
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1x80
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2x80
IBM DB2 Universal Database Data Warehouse Center Application Integration Guide	SC27-1124	db2adx80
IBM DB2 XML Extender Administration and Programming	SC27-1234	db2sxx80

商务智能信息

此类别中的信息描述如何使用将增强 “DB2 通用数据库” 的数据入库功能和分析功能的组件。

此类别的安装目录为 doc/htmlcd/%L/wareh。

表 31. 商务智能信息

书名	书号	PDF 文件名
IBM DB2 Warehouse Manager Information Catalog Center Administration Guide	SC27-1125	db2dix80
《IBM DB2 仓库管理器安装指南》	G152-0187	db2idc80

DB2 Connect 信息

此类别中的信息描述如何使用 “DB2 Connect 企业版” 或 “DB2 Connect 个人版” 来存取主机或 iSeries 数据。

此类别的安装目录为 doc/htmlcd/%L/conn。

表 32. DB2 Connect 信息

书名	书号	PDF 文件名
APPC, CPI-C, and SNA Sense Codes	无书号	db2apx80
IBM Connectivity Supplement	无书号	db2h1x80
《IBM DB2 Connect 快速入门, DB2 Connect 企业版》	G152-0271	db2c6c80
《IBM DB2 Connect 快速入门, DB2 Connect 个人版》	G152-0171	db2c1c80
《IBM DB2 Connect 用户指南》	S152-0172	db2c0c80

入门信息

安装和配置服务器、客户机以及其它 DB2 产品时，此类别中的信息非常有用。

此类别的安装目录为 doc/htmlcd/%L/start。

表 33. 入门信息

书名	书号	PDF 文件名
《IBM DB2 通用数据库快速入门, DB2 客户机版》	G152-0170	db2itc80
《IBM DB2 通用数据库快速入门, DB2 服务器版》	G152-0173	db2isc80
《IBM DB2 通用数据库快速入门, DB2 个人版》	G152-0175	db2i1c80
《IBM DB2 通用数据库安装与配置补遗》	G152-0174	db2iyc80
《IBM DB2 通用数据库快速入门, DB2 Data Links Manager 版》	G152-0169	db2z6c80

教程信息

教程信息介绍 DB2 功能部件并指导如何执行各种任务。

此类别的安装目录为 doc/htmlcd/%L/tutr。

表 34. 教程信息

书名	书号	PDF 文件名
《商务智能教程: 数据仓库简介》	无书号	db2tuc80
《商务智能教程: 数据入库扩展课程》	无书号	db2tac80
Development Center Tutorial for Video Online using Microsoft Visual Basic	无书号	db2tdx80
Information Catalog Center Tutorial	无书号	db2aix80
Video Central for e-business Tutorial	无书号	db2twx80
《Visual Explain 教程》	无书号	db2tvx80

可选组件信息

此类别中的信息描述如何使用可选 DB2 组件。

此类别的安装目录为 doc/htmlcd/%L/opt。

表 35. 可选组件信息

书名	书号	PDF 文件名
IBM DB2 Life Sciences Data Connect Planning, Installation, and Configuration Guide	GC27-1235	db2lsx80
IBM DB2 Spatial Extender User's Guide and Reference	SC27-1226	db2sbx80
IBM DB2 Database Data Links Manager Administration Guide and Reference	SC27-1221	db2z0x80
IBM DB2 Universal Database Net Search Extender Administration and Programming Guide **注**: 此文档的 HTML 不是从 HTML 文档 CD 安装的。	SH12-6740	N/A

发行说明

发行说明提供了特定于产品发行版和修订包级别的附加信息。它们还提供了并入到每个发行版和修订包中的文档更新的总结。

表 36. 发行说明

书名	书号	PDF 文件名
《DB2 发行说明》	参见“注”。	参见“注”。
《DB2 安装说明》	仅在产品 CD-ROM 上提供。	仅在产品 CD-ROM 上提供。

注: 发行说明的 HTML 版本可从“信息中心”或产品 CD-ROM 上获取。要在基于 UNIX 的平台上查看 ASCII 文件，参见 `Release.Notes` 文件。此文件位于 `DB2DIR/Readme/%L` 目录中，其中 `%L` 表示语言环境名称，`DB2DIR` 表示:

- `/usr/opt/db2_08_01`（在 AIX 上）
- `/opt/IBM/db2/V8.1`（在所有其它 UNIX 操作系统上）

相关任务:

- 第 262 页的『从 PDF 文件打印 DB2 书籍』
- 第 263 页的『订购打印的 DB2 书籍』
- 第 264 页的『访问联机帮助』
- 第 267 页的『通过从管理工具访问“DB2 信息中心”来查找产品信息』
- 第 268 页的『直接从 DB2 HTML 文档 CD 联机查看技术文档』

从 PDF 文件打印 DB2 书籍

可从 *DB2 PDF 文档* CD 上的 PDF 文件打印 DB2 书籍。通过使用 Adobe Acrobat Reader，可打印整本书或特定范围的那些页。

先决条件:

确保具有 Adobe Acrobat Reader。它可从 Adobe Web 站点获得，网址为 www.adobe.com。

过程:

要从 PDF 打印 DB2 书籍:

1. 插入 *DB2 PDF 文档* CD。在 UNIX 操作系统上，安装 DB2 PDF 文档 CD。有关如何在 UNIX 操作系统上安装 CD 的详细信息，请参考《*快速入门*》一书。

2. 启动 Adobe Acrobat Reader。
3. 从下列位置之一打开 PDF 文件:
   - 在 Windows 操作系统上:

     x:\doc*language* 目录，其中 *x* 表示 CD-ROM 盘符，*language* 表示两个字符的地区代码（它表示您所用的语言），例如，EN 表示英语。
   - 在 UNIX 操作系统上:

     CD-ROM 上的 */cdrom*/doc/*%L* 目录，其中 */cdrom* 表示 CD-ROM 的安装点而 *%L* 表示期望的语言环境的名称。

相关任务:

- 第 263 页的『订购打印的 DB2 书籍』
- 第 267 页的『通过从管理工具访问 “DB2 信息中心” 来查找产品信息』
- 第 268 页的『直接从 DB2 HTML 文档 CD 联机查看技术文档』

相关参考:

- 第 255 页的『 “DB2 通用数据库” 技术信息概述』

订购打印的 DB2 书籍

过程:

要订购打印的书籍:

- 与 IBM 授权经销商或市场营销代表联系。要查找您当地的 IBM 代表，查看 IBM 全球联系人目录（IBM Worldwide Directory of Contacts），网址为 www.ibm.com/planetwide。
- 在美国可致电 1-800-879-2755，在加拿大则可致电 1-800-IBM-4YOU。
- 访问 IBM 出版物中心（IBM Publications Center），网址为 www.ibm.com/shop/publications/order。

还可通过从您的 IBM 分销商订购您的 DB2 产品的文档包来获得印刷的 DB2 手册。文档包是 DB2 库中的手册的一个子集，它们被选择来帮助您使用您购买的 DB2 产品进行初步的操作。文档包中的手册与 *DB2 PDF 文档 CD* 中以 PDF 格式提供的手册相同，并包含与 *DB2 HTML 文档 CD* 中提供的文档相同的内容。

相关任务:

- 第 262 页的『从 PDF 文件打印 DB2 书籍』
- 第 265 页的『通过从浏览器访问 “DB2 信息中心” 来查找主题』
- 第 268 页的『直接从 DB2 HTML 文档 CD 联机查看技术文档』

相关参考:

- 第 255 页的『 “DB2 通用数据库”技术信息概述 』

访问联机帮助

所有 DB2 组件附带提供的联机帮助有三种类型:

- 窗口和笔记本帮助
- 命令行帮助
- SQL 语句帮助

窗口和笔记本帮助说明可在窗口或笔记本中执行的任务并描述各控件。此帮助有两种类型:

- 可从**帮助**按钮访问的帮助
- 弹出信息

帮助按钮让您可以访问概述和先决条件信息。弹出信息描述窗口或笔记本中的各控件。窗口和笔记本帮助可从具有用户界面的 DB2 中心和组件获得。

命令行帮助包括“命令”帮助和“消息”帮助。“命令”帮助说明命令行处理器中命令的语法。“消息”帮助描述产生错误消息的原因并描述为解决错误而应采取的任何操作。

SQL 语句帮助包括 SQL 帮助和 SQLSTATE 帮助。DB2 返回可作为 SQL 语句结果的条件的 SQLSTATE 值。SQLSTATE 帮助说明 SQL 语句（SQL 语句和类代码）的语法。

注: SQL 帮助对于 UNIX 操作系统不可用。

过程:

要访问联机帮助:

- 对于窗口和笔记本帮助，单击**帮助**或单击该控件，然后单击 **F1**。如果选择了**工具设置**笔记本**常规**页上的**自动显示弹出信息**复选框，则还可以通过将鼠标光标置于特定控件上来查看该控件的弹出信息。
- 对于命令行帮助，打开命令行处理器并输入:
  - 对于“命令”帮助:

    ```
    ? command
    ```

    其中 *command* 表示一个关键字或整条命令。

例如，`? catalog` 显示所有 CATALOG 命令的帮助，而 `? catalog database` 显示 CATALOG DATABASE 命令的帮助。

- 对于“消息”帮助:

  `? XXXnnnnn`

  其中 *XXXnnnnn* 表示有效消息标识符。

  例如，`? SQL30081` 将显示有关 SQL30081 消息的帮助。

- 对于 SQL 语句帮助，打开命令行处理器并输入:

  `? sqlstate` 或 `? 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。

相关任务:

- 第 265 页的『通过从浏览器访问“DB2 信息中心”来查找主题』
- 第 268 页的『直接从 DB2 HTML 文档 CD 联机查看技术文档』

通过从浏览器访问“DB2 信息中心”来查找主题

“DB2 信息中心”可从浏览器访问，从而使您能够访问为充分利用“DB2 通用数据库”和 DB2 Connect 所需的信息。“DB2 信息中心”还记录主要的 DB2 功能部件和组件，包括复制、数据入库、元数据和 DB2 extender。

从浏览器访问的“DB2 信息中心”包括下列主要元素:

导航树 导航树位于浏览器窗口左边的框架中。该树可展开和折叠以显示和隐藏主题、词汇表和“DB2 信息中心”中的主索引。

导航工具栏

导航工具栏位于浏览器窗口的右上边框架中。导航工具栏包含一些使您能够执行下列操作的按钮：搜索“DB2 信息中心”、隐藏导航树以及查找导航树中当前显示的主题。

内容框架

内容框架位于浏览器窗口的右下边框架中。当单击导航树中的链接、单击搜索结果或访问另一主题或主索引的链接时，内容框架会显示“DB2 信息中心”的主题。

先决条件:

要从浏览器访问“DB2 信息中心”，必须使用下列浏览器之一:

- Microsoft Explorer，版本 5 或更高版本
- Netscape Navigator，版本 6.1 或更高版本

限制:

“DB2 信息中心”只包含您选择从 *DB2 HTML 文档 CD* 安装的那些主题集。如果当您尝试访问指向某个主题的链接时 Web 浏览器返回找不到文件错误，则您必须安装 *DB2 HTML 文档 CD* 中的一个或多个附加的主题集。

过程:

要通过使用关键字进行搜索来查找主题:

1. 在导航工具栏中，单击**搜索**。
2. 在“搜索”窗口最上面的文本输入字段中，输入一个或多个与您感兴趣的领域相关的词条，并单击**搜索**。一个按准确度排列的主题列表将显示在**结果**字段中。每一单项旁的数字等级提供了匹配程度的指示（较大的数字表示较高的匹配程度）。

   输入较多的项会提高查询的精度，同时还会减少从查询返回的主题数目。
3. 在**结果**字段中，单击想要阅读的主题的标题。该主题将会显示在内容框架中。

要查找导航树中的主题:

1. 在导航树中，单击与您感兴趣的区域相关的主题类别的书籍图标。一个子类别列表将显示在该图标下面。
2. 继续单击书籍图标，直到找到包含您感兴趣的主题的类别为止。链接至主题的类别在您将光标移到类别标题上时将类别标题显示为带下划线的链接。导航树使用页图标来标识主题。
3. 单击主题链接。该主题会显示在内容框架中。

要查找主索引中的主题或项:

1. 在导航树中，单击“索引”类别。该类别展开，并在导航树中显示按字母顺序排列的链接列表。
2. 在导航树中，单击相应于与感兴趣主题相关的项的第一个字符的链接。具有该首字符的项列表将会显示在内容框架中。具有多个索引条目的项将由一个书籍图标标识。
3. 单击与您感兴趣的项相对应的书籍图标。一个子项和主题列表将显示在您单击的项下面。主题是由页图标标识的，其标题带有下划线。
4. 单击符合需要的主题的标题。该主题会显示在内容框架中。

相关概念:

- 第 273 页的『易使用性』
- 第 275 页的『从浏览器访问的 DB2 信息中心』

相关任务:

- 第 267 页的『通过从管理工具访问 “DB2 信息中心” 来查找产品信息』
- 第 269 页的『更新安装在机器上的 HTML 文档』
- 第 271 页的『对于使用 Netscape 4.x 搜索 DB2 文档进行故障诊断』
- 第 272 页的『搜索 DB2 文档』

相关参考:

- 第 255 页的『 “DB2 通用数据库” 技术信息概述』

通过从管理工具访问 “DB2 信息中心” 来查找产品信息

“DB2 信息中心” 提供了对 DB2 产品信息的快速访问且在可以使用 DB2 管理工具的所有操作系统上可用。

从工具访问的 “DB2 信息中心” 提供了六种类型的信息。

任务 可使用 DB2 执行的关键任务。

概念 DB2 的关键概念。

参考 DB2 参考信息，如关键字、命令以及 API。

故障诊断
帮助您解决常见 DB2 问题的错误消息和信息。

样本 随 DB2 提供的样本程序的 HTML 列表的链接。

教程 用来帮助您了解 DB2 功能部件的指导性辅助。

先决条件:

“DB2 信息中心” 中的某些链接指向因特网上的 Web 站点。要显示这些链接的内容，首先必须与因特网连接。

过程:

要通过从工具访问 “DB2 信息中心” 来查找产品信息:

1. 用下列方法之一启动 “DB2 信息中心”：
   - 从图形管理工具中，单击工具栏中的**信息中心**图标。还可从**帮助**菜单中选择它。
   - 在命令行中输入 **db2ic**。

2. 单击与试图查找的信息相关的信息类型的选项卡。
3. 浏览整个树并单击感兴趣的主题。“信息中心”将启动 Web 浏览器以显示信息。
4. 要查找信息而无须浏览列表，可单击列表右边的**搜索**图标。

   一旦“信息中心”启动了浏览器来显示信息，就可通过单击导航工具栏中的**搜索**图标来执行全文本搜索。

相关概念:

- 第 273 页的『易使用性』
- 第 275 页的『从浏览器访问的 DB2 信息中心』

相关任务:

- 第 265 页的『通过从浏览器访问“DB2 信息中心”来查找主题』
- 第 272 页的『搜索 DB2 文档』

直接从 DB2 HTML 文档 CD 联机查看技术文档

还可直接从 CD 读取可从 *DB2 HTML 文档 CD* 安装的所有 HTML 主题。因此，可查看文档而不必安装它。

限制:

由于“工具”帮助是从 DB2 产品 CD 而不是从 *DB2 HTML 文档 CD* 安装的，您必须安装 DB2 产品才能查看该帮助。

过程:

1. 插入 *DB2 HTML 文档* CD。在 UNIX 操作系统上，安装 *DB2 HTML 文档 CD*。有关如何在 UNIX 操作系统上安装 CD 的详细信息，参考《*快速入门*》一书。
2. 启动 HTML 浏览器并打开适当的文件:
   - 对于 Windows 操作系统:

     ```
     e:\program files\IBM\SQLLIB\doc\htmlcd\%L\index.htm
     ```

     其中 *e* 表示 CD-ROM 驱动器，%L 是想要使用的文档的语言环境，例如，**en_US** 表示英语。
   - 对于 UNIX 操作系统:

     ```
     /cdrom/program files/IBM/SQLLIB/doc/htmlcd/%L/index.htm
     ```

     其中 */cdrom/* 表示安装 CD 的地方，%L 是想要使用的文档的语言环境，例如，**en_US** 表示英语。

相关任务:

- 第 265 页的『通过从浏览器访问 “DB2 信息中心” 来查找主题』
- 第 270 页的『将文件从 DB2 HTML 文档 CD 复制到 Web 服务器』

相关参考:

- 第 255 页的『 “DB2 通用数据库” 技术信息概述』

更新安装在机器上的 HTML 文档

现在，就有可能在 IBM 进行了更新之后更新从 *DB2 HTML 文档 CD* 安装的 HTML。可用以下两种方法之一来完成:

- 使用 “信息中心” （如果安装了 DB2 管理 GUI 工具的话）。
- 通过下载和应用 DB2 HTML 文档修订包。

注: 这将不会更新 DB2 代码; 它只更新从 *DB2 HTML 文档 CD* 安装的 HTML 文档。

过程:

要使用 “信息中心” 来更新本地文档:

1. 用下列方法之一启动 “DB2 信息中心”:
   - 从图形管理工具中，单击工具栏中的**信息中心**图标。还可从**帮助**菜单中选择它。
   - 在命令行中输入 **db2ic**。
2. 确保您的机器对外部因特网具有访问权; 更新程序将从 IBM 服务器下载最新的文档修订包（如果需要的话）。
3. 从菜单中选择**信息中心** —> **更新本地文档**以启动更新。
4. 提供代理信息（如果需要的话）以连接至外部因特网。

“信息中心” 将下载并应用最新的文档修订包（如果有的话）。

要手工下载并应用文档修订包:

1. 确保机器已连接至因特网。
2. 在浏览器中打开 DB2 支持页，网址为: www.ibm.com/software/data/db2/udb/winos2unix/support。
3. 访问版本 8 的链接并查找 “文档修订包” （Documentation FixPaks）链接。
4. 通过将文档修订包级别与已安装的文档级别进行比较来确定本地文档的版本是否已过时。您机器上的此当前文档处于以下级别: **DB2 v8.1 GA**。

5. 如果有更新的文档版本，则下载适用于您的操作系统的修订包。有一个适用于所有 Windows 平台的修订包和一个适用于所有 UNIX 平台的修订包。
6. 应用修订包:
   - 对于 Windows 操作系统: 文档修订包是自解压 zip 文件。将下载的文档修订包置于一个空目录中并运行它。这将创建一个 **setup** 命令，可运行该命令来安装文档修订包。
   - 对于 UNIX 操作系统: 文档修订包是压缩的 tar.Z 文件。解压并解取该文件。这将创建一个带有称为 **installdocfix** 的脚本的名为 `delta_install` 的目录。运行此脚本来安装文档修订包。

相关任务:

- 第 270 页的『将文件从 DB2 HTML 文档 CD 复制到 Web 服务器』

相关参考:

- 第 255 页的『 “DB2 通用数据库” 技术信息概述』

将文件从 DB2 HTML 文档 CD 复制到 Web 服务器

在 *DB2 HTML 文档 CD* 上交付了整个 DB2 信息库，可将它安装在 Web 服务器上以更便于访问。将想要的语言的文档复制至 Web 服务器即可。

注: 当您通过低速连接从 Web 服务器访问 HTML 文档时，可能会遇到性能较低的情况。

过程:

要将文件从 *DB2 HTML 文档 CD* 复制到 Web 服务器，使用适当的源路径:

- 对于 Windows 操作系统:

  `E:\program files\IBM\SQLLIB\doc\htmlcd\%L\*.*`

  其中 *E* 表示 CD-ROM 驱动器，*%L* 表示语言标识符。
- 对于 UNIX 操作系统:

  `/cdrom/program files/IBM/SQLLIB/doc/htmlcd/%L/*.*`

  其中 *cdrom* 表示 CD-ROM 驱动器的安装点，*%L* 表示语言标识。

相关任务:

- 第 272 页的『搜索 DB2 文档』

相关参考:

- 『受支持的 DB2 界面语言、语言环境和代码页』（*《DB2 服务器快速入门》*）

- 第 255 页的『 “DB2 通用数据库”技术信息概述』

对于使用 Netscape 4.x 搜索 DB2 文档进行故障诊断

大多数搜索问题都与 web 浏览器提供的 Java 支持有关。此任务描述可能的解决办法。

过程:

一个 Netscape 4.x 常见问题是丢失和误置安全性类。尝试下列解决办法，尤其是当您在浏览器 Java 控制台中看到以下行时更应尝试此方法:

```
找不到类 java/security/InvalidParameterException
```

- 在 Windows 操作系统上:

  从 *DB2 HTML 文档 CD*，将提供的 *x*:program files\IBM\SQLLIB\doc\htmlcd*locale*\InvalidParameterException.class 文件复制到相对于 Netscape 浏览器安装的 java\classes\java\security\ 目录，其中 *x* 表示 CD-ROM 驱动器盘符，*locale* 表示期望的语言环境的名称。

  注: 可能必须创建 java\security\ 子目录结构。

- 在 UNIX 操作系统上:

  从 *DB2 HTML 文档 CD*，将提供的 */cdrom*/program files/IBM/SQLLIB/doc/htmlcd/*locale*/InvalidParameterException.class 文件复制到相对于 Netscape 浏览器安装的 java/classes/java/security/ 目录，其中 *cdrom* 表示 CD-ROM 的安装点，*locale* 表示期望的语言环境的名称。

  注: 可能必须创建 java/security/ 子目录结构。

如果 Netscape 浏览器仍无法显示搜索输入窗口，则尝试下列操作:

- 停止 Netscape 浏览器的所有实例以确保机器上无任何 Netscape 代码运行。然后，打开 Netscape 浏览器的新实例并再次尝试启动搜索。
- 清除浏览器的高速缓存。
- 尝试用 Netscape 的其它版本或另一种浏览器。

相关任务:

- 第 272 页的『搜索 DB2 文档』

搜索 DB2 文档

可搜索 DB2 文档库来定位所需的信息。单击 “DB2 信息中心” （从浏览器访问）导航工具栏中的搜索图标时，将打开一个弹出式搜索窗口。可能需要一分钟来装入搜索，取决于您的计算机和网络的速度。

先决条件:

需要 Netscape 6.1 或更高版本或者 Microsoft 的 Internet Explorer 5 或更高版本。确保启用了浏览器的 Java 支持。

限制:

使用文档搜索时，将存在下列限制:

- 搜索不能区分大小写。
- 不支持布尔搜索。
- 不支持通配符搜索和部分搜索。例如，对 *java**（或 *java*）的搜索将仅查找文字字符串 *java**（或 *java*），而找不到 *javadoc*。

过程:

要搜索 DB2 文档:

1. 在导航工具栏中，单击**搜索**图标。
2. 在 “搜索” 窗口最上面的文本输入字段中，输入一个或多个与您感兴趣的领域相关的词条（由空格分隔），并单击**搜索**。一个按准确度排列的主题列表将显示在**结果**字段中。每一单项旁的数字等级提供了匹配程度的指示（较大的数字表示较高的匹配程度）。

   输入较多的项会提高查询的精度，同时还会减少从查询返回的主题数目。
3. 在**结果**列表中，单击要阅读的主题的标题。主题将显示在 “DB2 信息中心” 的内容框架中。

注: 执行搜索时，第一个（最高级别的）结果自动装入到浏览器框架中。要查看其它搜索结果的内容，单击结果列表中的结果。

相关任务:

- 第 271 页的『对于使用 Netscape 4.x 搜索 DB2 文档进行故障诊断』

联机 DB2 故障诊断信息

在 DB2® UDB 版本 8 的发行版中，将不再提供 *Troubleshooting Guide*。曾经包含在此指南中的故障诊断信息都已集成到 DB2 出版物中，从而使我们能向您提供最新信息。要查找有关故障诊断实用程序和 DB2 功能的信息，可从任何工具访问“DB2 信息中心”。

如果您遇到问题且想要获取查找可能原因及解决方案的帮助，请参考 Online Support 站点。该支持站点包含了一个不断更新的大型数据库，数据库的内容涉及 DB2 出版物、技术说明、APAR（产品问题）记录、修订包和其它资源。可使用该支持站点来搜索此知识库并查找问题的可能解决方案。

访问 Online Support 站点（网址为 www.ibm.com/software/data/db2/udb/winos2unix/support），或通过单击“DB2 信息中心”中的**在线支持**按钮来访问它。现在，还可从此站点获取经常更改的信息，如内部 DB2 错误代码列表。

相关概念:

- 第 275 页的『从浏览器访问的 DB2 信息中心』

相关任务:

- 第 267 页的『通过从管理工具访问“DB2 信息中心”来查找产品信息』

易使用性

易使用性功能部件可帮助那些身体有某些缺陷（如活动不方便或视力不太好）的用户成功使用软件产品。以下是“DB2® 通用数据库版本 8”中主要的易使用性功能部件:

- 通过键盘即可对所有 DB2 功能部件进行操作，而不必使用鼠标。参见第 274 页的『键盘输入和导航』。
- DB2 允许您定制字体的大小和颜色。参见第 274 页的『界面显示的易使用性』。
- DB2 允许您接收可视或音频警告信号。参见第 274 页的『备用警告信号』。
- DB2 支持使用 Java™ Accessibility API 的易使用性应用程序。参见第 274 页的『与辅助技术的兼容性』。
- DB2 附带了以易使用的格式提供的文档。参见第 274 页的『可访问文档』。

键盘输入和导航

键盘输入

只使用键盘就可对“DB2 工具”进行操作。使用键或键组合就可执行使用鼠标完成的大多数操作。

键盘焦点

在基于 UNIX 的系统中，键盘焦点的位置是突出显示的，指示窗口的哪个区域处于活动状态且击键对何处会有影响。

界面显示的易使用性

“DB2 工具”中的功能部件增强了用户界面，使视力不太好的用户更易使用。这些易使用性方面的增强包括了对可定制字体特性的支持。

字体设置

“DB2 工具”允许您通过使用“工具设置”笔记本来选择菜单和对话框窗口中文本的颜色、大小和字体。

不依赖于颜色

不需要分辨颜色就可以使用此产品中的任何功能。

备用警告信号

可使用“工具设置”笔记本来指定是否想要通过音频或可视信号接收警告。

与辅助技术的兼容性

“DB2 工具”界面支持对屏幕阅读器启用 Java Accessibility API 并支持有某些缺陷的用户使用其它辅助技术。

可访问文档

DB2 产品系列的文档提供了 HTML 格式的版本。使您可根据浏览器中设置的显示首选项来查看文档。还允许您使用屏幕阅读器和其它辅助性技术。

DB2 教程

DB2® 教程帮助您了解“DB2 通用数据库”的各个方面。教程提供了开发应用程序、调整 SQL 查询性能、使用数据仓库、管理元数据和使用 DB2 开发 Web 服务等方面的课程，这些课程中还提供了逐步指示信息。

开始之前:

必须先从 *DB2 HTML 文档* CD 中安装教程，才能使用以下的链接来访问这些教程。

如果不想安装这些教程，则可直接从 *DB2 HTML 文档 CD* 查看这些教程的 HTML 版本。还可在 *DB2 PDF 文档 CD* 上获取这些教程的 PDF 版本。

某些教程课程使用了样本数据或代码。有关各个教程特定任务的任何先决条件的描述，参见每个教程的内容。

“DB2 通用数据库”教程:

如果从 *DB2 HTML 文档* CD 安装了教程，则可单击下表中的某个教程标题来查看该教程。

《商务智能教程: 数据仓库中心简介》
使用“数据仓库中心”来执行介绍性的数据入库任务。

《商务智能教程: 数据入库的扩展课程》
使用“数据仓库中心”来执行高级数据入库任务。

Development Center Tutorial for Video Online using Microsoft® Visual Basic
使用 Microsoft Visual Basic 的“开发中心加载件”来构建应用程序的各个组件。

Information Catalog Center Tutorial
使用“信息目录中心”来创建和管理信息目录以定位并使用元数据。

Video Central for e-business Tutorial
使用 WebSphere® 产品来开发和部署高级“DB2 Web 服务”应用程序。

《Visual Explain 教程》
使用 Visual Explain 来分析、优化和调整 SQL 语句以获取更好的性能。

从浏览器访问的 DB2 信息中心

“DB2® 信息中心”让您可以访问在您的业务中充分利用 DB2 通用数据库™ 和 DB2 Connect™ 所需的所有信息。“DB2 信息中心”文档还记录主要的 DB2 功能部件和组件，包括复制、数据入库、信息目录中心、Life Sciences Data Connect 和 DB2 extender。

从浏览器访问的“DB2 信息中心”具有以下功能部件（如果是在 Netscape Navigator 6.1 或更高版本或者 Microsoft Internet Explorer 5 或更高版本中查看）。某些功能部件需要您启用对 Java 或 JavaScript 的支持:

定期更新的文档

通过下载更新的 HTML，使您的主题保持为最新。

搜索 通过单击导航工具栏中的**搜索**来搜索安装在工作站上的所有主题。

集成的导航树

从一个导航树中就可找出 DB2 资料库中的任何主题。导航树是按信息类型组织的，如下所示:

- “任务”提供了有关如何完成目标的逐步指示信息。
- “概念”提供了主题的概述。
- “参考”主题提供了有关主题的详细信息，包括语句和命令语法、消息帮助以及需求。

主索引 从主索引访问从 *DB2 HTML 文档 CD* 中安装的信息。索引是按索引项以字母顺序组织的。

主词汇表

主词汇表定义在“DB2 信息中心”中使用的术语。词汇表是按词汇表术语以字母顺序组织的。

相关任务:

- 第 265 页的『通过从浏览器访问“DB2 信息中心”来查找主题』
- 第 267 页的『通过从管理工具访问“DB2 信息中心”来查找产品信息』
- 第 269 页的『更新安装在机器上的 HTML 文档』

附录 D. 声明

IBM 可能在其它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 代理咨询。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证。您可以用书面方式将许可证查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可证查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

本条款不适用于联合王国或任何这样的条款与当地法律不一致的国家或地区: 国际商业机器公司以“按现状”的基础提供本出版物，不附有任何形式的（无论是明示的，还是默示的）保证，包括（但不限于）对非侵权性、适销性和适用于某特定用途的默示保证。某些国家或地区在某些交易中不允许免除明示或默示的保证。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。该 Web 站点中的资料不是 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 可以它认为合适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它程序（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
Office of the Lab Director
8200 Warden Avenue
Markham, Ontario
L6G 1C7
CANADA

只要遵守适当的条件和条款，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。

本文档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 国际程序许可证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，在其它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、其出版说明或其它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、兼容性或任何其它关于非 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。

本资料中可能包含用于日常业务运作的数据和报表的示例。为了尽可能完整地说明问题，这些示例可能包含个人、公司、品牌和产品的名称。所有这些名称都是虚构的，如与实际商业企业所使用的名称和地址有雷同，纯属巧合。

版权许可证:

本资料中可能包含源语言的样本应用程序，它们举例说明了各种操作平台上的编程技术。为了开发、使用、营销或分发符合编写这些样本程序所针对操作平台的应用程序编程接口的应用程序，您可以以任何形式复制、修改和分发这些样本程

序，而不必向 IBM 付款。尚未在所有条件下彻底测试这些示例。因此，IBM 不能保证或默示这些程序的可靠性、适用性或功能。

这些样本程序或任何派生产品的每个副本或任何部分都必须包括如下版权声明:

商标

下列各项是国际商业机器公司在美国和／或其它国家或地区的商标，且已在 DB2 UDB 文档库中的至少一份文档中使用。

ACF/VTAM
AISPO
AIX
AIXwindows
AnyNet
APPN
AS/400
BookManager
C Set++
C/370
CICS
Database 2
DataHub
DataJoiner
DataPropagator
DataRefresher
DB2
DB2 Connect
DB2 Extender
DB2 OLAP Server
DB2 Universal Database
Distributed Relational Database Architecture
DRDA
eServer
Extended Services
FFST
First Failure Support Technology
IBM
IMS
IMS/ESA
iSeries
LAN Distance
MVS
MVS/ESA
MVS/XA
Net.Data
NetView
OS/390
OS/400
PowerPC
pSeries
QBIC
QMF
RACF
RISC System/6000
RS/6000
S/370
SP
SQL/400
SQL/DS
System/370
System/390
SystemView
Tivoli
VisualAge
VM/ESA
VSE/ESA
VTAM
WebExplorer
WebSphere
WIN-OS/2
z/OS
zSeries

下列各项是其它公司的商标或注册商标，且已在 DB2 UDB 文档库中的至少一份文档中使用:

Microsoft、Windows、Windows NT 和 Windows 徽标是 Microsoft Corporation 在美国和／或其它国家或地区的商标。

Intel 和 Pentium 是 Intel Corporation 在美国和／或其它国家或地区的商标。

Java 和所有基于 Java 的商标是 Sun Microsystems, Inc. 在美国和／或其它国家或地区的商标。

UNIX 是 The Open Group 在美国和其它国家或地区的注册商标。

其它公司、产品或服务名称可能是其它公司的商标或服务标记。

索引

[D]

[F]

[G]

[H]

[T]

[W]

[X]

[Y]

[Z]

B

C

D

I

L

M

N

R

S

T

U

X

与 IBM 联系

在美国，请致电下列其中一个号码以与 IBM 联系:

- 1-800-237-5511，可获取客户服务
- 1-888-426-4343，可了解所提供的服务项目
- 1-800-IBM-4YOU（426-4968），可获取有关 DB2 市场营销与销售的信息

在加拿大，请致电下列其中一个号码以与 IBM 联系:

- 1-800-IBM-SERV (1-800-426-7378)，可获取客户服务
- 1-800-465-9600，可了解所提供的服务项目
- 1-800-IBM-4YOU (1-800-426-4968)，可获取有关 DB2 市场营销与销售的信息

要查找您所在国家或地区的 IBM 营业处，可查看 IBM 全球联系人目录（IBM Directory of Worldwide Contacts），网址为 www.ibm.com/planetwide。

产品信息

有关 “DB2 通用数据库” 产品的信息，可打电话获取或通过万维网获取，网址为: www.ibm.com/software/data/db2/udb。

此站点包含有关技术库、订购书刊、客户机下载、新闻组、修订包、新闻和 Web 资源链接的最新信息。

您如果住在美国，请致电下列其中一个号码:

- 1-800-IBM-CALL（1-800-426-2255），可订购产品或获取一般信息。
- 1-800-879-2755，可订购出版物。

有关如何在美国以外的国家或地区与 IBM 联系的信息，请访问 IBM Worldwide 页面，网址为 www.ibm.com/planetwide。

部件号: CT17VSC

中国印刷

S152-0167-00

Spine information:

IBM® DB2 通用数据库™ 管理指南: 计划 *版本 8*