IBM® DB2 Universal Database™
DB2 通用数据库

管理指南：实现

版本 8.2

S152-0165-01

IBM® DB2 Universal Database™
DB2 通用数据库

管理指南：实现

版本 8.2

S152-0165-01

在使用本资料及其支持的产品之前，请务必阅读『*声明*』中的一般信息。

本文档包含 IBM 的专利信息。它是根据许可协议提供的，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。

可以用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。

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”在其三卷中提供了使用和管理 DB2 关系数据库管理系统（RDBMS）产品所需的信息，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（可在*《管理指南: 性能》*中找到）

本书中描述的许多任务可以使用不同的界面来执行:

- **命令行处理器**，允许您从图形界面存取和处理数据库。通过此界面，您还可以执行 SQL 语句和 DB2 实用程序函数。本书中的大多数示例说明的是此界面的用法。有关使用命令行处理器的更多信息，请参阅 *Command Reference*。
- **应用程序编程接口**，允许您在应用程序内执行 DB2 实用程序函数。有关使用应用程序编程接口的更多信息，请参阅 *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，例如，配置系统、管理目录、备份和恢复系统、安排作业和管理介质。“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”允许您通过图形用户界面执行 DB2 实用程序函数。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。要获取介绍性帮助，从“控制中心”窗口的**帮助**下拉菜单中选择**入门**。**Visual Explain** 工具是从控制中心调用的。

  在下面三个视图中提供了“控制中心”:

  – 基本。此视图显示基本对象（例如，数据库、表和存储过程）上的核心 DB2 UDB 功能。
  – 高级。此视图具有所有对象和可用的操作。如果您正在企业环境中工作，并且您想连接至 DB2 z/OS 或 IMS 版，则使用此视图。
  – 定制。此视图使您能够调整对象树和对象操作。

还可使用其它工具来执行管理任务。它们包括:

- “命令编辑器”，它取代了“命令中心”，可用来生成、编辑、运行和处理 SQL 语句；IMS 和 DB2 命令；使用获得的输出；以及查看已说明 SQL 语句的存取方案的图形表示法。
- “开发中心”，它提供了对本机 SQL 持久存储器模块（PSM）存储过程；Java 存储过程 iSeries 版 V5R3 和更新版本；用户定义的函数（UDF）以及结构化类型的支持。
- “健康中心”提供了一个工具来帮助 DBA 解决性能和资源分配问题。
- 用来更改“控制中心”、“健康中心”和“复制中心”的设置的“工具设置”。
- 用来安排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”。

本书的读者

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 Universal Database™（DB2 UDB）（DB2 通用数据库）关系数据库管理系统的管理和操作的程序员和其他用户也可使用本书。

本书的结构

本书包含关于下列主要主题的信息:

实现设计

- 第 1 章, 『创建数据库之前』, 描述在创建数据库和数据库中的对象之前所需的先决条件。
- 第 2 章, 『创建和使用 DB2 管理服务器（DAS）』, 讨论什么是 DAS、如何创建它以及如何使用它。
- 第 3 章, 『创建数据库』, 描述与创建数据库和数据库中的对象相关联的任务。
- 第 4 章, 『创建表和其它相关表对象』, 描述在实现数据库设计时如何创建具有特定特征的表。
- 第 5 章, 『改变数据库』, 描述与改变或删除数据库以及数据库中的对象相关联的先决条件和任务。
- 第 6 章, 『改变表和其它相关表对象』, 描述如何删除表或者如何修改与这些表相关联的特定特征。同时还描述了删除和修改相关的表对象。

数据库安全性

- 第 7 章, 『控制数据库存取』, 描述如何控制对数据库资源的存取权。
- 第 8 章, 『审计 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）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

- 附录 A, 『符合命名规则』, 介绍命名数据库和对象时需遵循的规则。
- 附录 B, 『使用自动客户机重新路由』, 讨论自动重新路由客户机应用程序以及如何启用此支持。
- 附录 C, 『使用 “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。
- 附录 D, 『向多个数据库分区发出命令』, 讨论使用 *db2_all* 和 *rah* shell 脚本将命令发送至一个分区数据库环境中的所有分区。
- 附录 E, 『使用 Windows 管理规范（WMI）支持』, 提供关于如何使用 WMI 管理 DB2 的信息。
- 附录 F, 『使用 Windows NT 安全性』, 描述DB2如何使用 Windows 安全性。
- 附录 G, 『使用 “Windows 性能监视器” 』, 描述如何使用 “Windows 性能监视器” 来收集 DB2 性能数据。
- 附录 H, 『使用 Windows 数据库分区服务器』, 描述 Windows 用来与分区数据库服务器配合工作的实用程序。
- 附录 I, 『配置多逻辑节点』, 描述如何在分区数据库环境中配置多逻辑节点。

- 附录 J, 『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。

已从*《管理指南: 实现》*手册移动具有标题“移动数据的实用程序”的章节。

注: 所有关于移动数据的 DB2 实用程序的信息和来自 *Command Reference* 和 *Administrative API Reference* 的类似主题已合并到 *Data Movement Utilities Guide and Reference* 中。

Data Movement Utilities Guide and Reference 是有关这些主题的主要且单一的信息源。

要了解有关数据复制的更多信息, 请参阅 *《IBM DB2 Information Integrator SQL 复制指南与参考》*。

已从*《管理指南: 实现》*手册移动具有标题“恢复数据库”的章节。

注: 所有关于备份与恢复数据的方法和工具的信息和来自 *Command Reference* 和 *Administrative API Reference* 的类似主题已合并到*《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》*是有关这些主题的主要且单一的信息源。

“管理指南”的其它卷的简要概述

管理指南: 计划

*《管理指南: 计划》*主要讨论数据库设计。它展现逻辑与物理设计问题和分布式事务问题。下面简要描述该卷中的特定章节和附录:

数据库概念

- “基本关系数据库概念”提供数据库对象（包括恢复对象、存储器对象和系统对象）的概述。
- “并行数据库系统”提供有关 DB2 可使用的并行性类型的介绍。
- “关于数据仓储”提供数据仓储和数据仓储任务的概述。

数据库设计

- “逻辑数据库设计”讨论逻辑数据库设计的概念和准则。
- “物理数据库设计”讨论物理数据库设计的准则, 包括与数据存储相关的注意事项。
- “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。
- “针对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

- “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。
- “本地语言支持（NLS）”描述“DB2 本地语言支持”, 包括关于地域、语言和代码页的信息。

- “在 64 位环境中启用大页支持（AIX）”讨论对 16 MB 页大小的支持以及如何启用此支持。

管理指南: 性能

*《管理指南: 性能》*主要讨论性能问题，即那些关于建立、测试和提高应用程序及 DB2 通用数据库产品本身的性能的主题和问题。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”介绍管理和提高 DB2 UDB 性能的概念和注意事项。
- “体系结构与进程”介绍底层的 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提高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时用来提高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

- “操作性能”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”描述与增大数据库系统的大小相关的一些注意事项和任务。
- “将数据再分发到各数据库分区中”讨论在分区数据库环境中将数据再分发在各分区中所需的任务。
- “基准程序测试”提供有关基准程序测试和如何执行基准程序测试的概述。
- “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 实现设计

第 1 章 创建数据库之前

在确定了数据库的设计之后，必须创建数据库和其中的对象。这些对象包括模式、数据库分区组、表空间、表、视图、包装器、服务器、昵称、类型映射、函数映射、别名、用户定义的类型（UDT）、用户定义的函数（UDF）、自动总结表（AST）、触发器、约束、索引和程序包。您可以在命令行处理器中使用 SQL 语句以及通过应用程序中的 SQL 语句来创建这些对象。

有关 SQL 语句的信息，请参阅 *SQL Reference* 手册。有关命令行处理器命令的信息，请参阅 *Command Reference* 手册。有关应用程序编程接口（API）的信息，请参阅 *Administrative API Reference* 手册。

创建数据库对象的另一种方法是通过“控制中心”来进行。可以使用“控制中心”，而不需要使用 SQL 语句、命令行处理器命令或 API。

在本章中，使用“控制中心”完成任务的方法放在框中，予以突出显示。后面紧跟着使用命令行的方法以供您比较，有时还带有示例。在某些情况下，任务只显示一种方法。当使用“控制中心”时，请您记住，您可以使用其中的帮助，它可以提供比此处的概述信息更多的详细信息。

本章主要讨论您在创建数据库及其所有对象之前应该了解的信息。在创建数据库之前，您必须了解一些先决条件概念、主题以及几项必须执行的任务。

下一章包含各种对象的简要讨论，那些对象可能会成为您的数据库设计实现的一部分。

此部分的最后一章先是讨论在改变数据库之前必须考虑的主题，然后说明了如何改变或删除数据库对象。

对于 DB2 通用数据库与操作系统相互影响的那些领域，本章及随后章节中的某些主题可能会讨论特定操作系统的差别。您也许能够利用本机操作系统的能力或差别，而不是 DB2 UDB 提供的那些能力或差别。有关细微差别，请参阅您的《*快速入门*》手册和操作系统文档。

例如，Windows 支持称为“服务”的应用程序类型。DB2 Windows 版将把每个 DB2 实例定义为服务。服务可以在系统引导时自动启动，也可由用户通过“服务”控制面板 applet 启动，或者通过使用服务功能的 Windows 32 位应用程序（包括在 Microsoft Windows 32 位应用程序编程接口（API）中）启动。服务可以在系统引导时自动启动，也可由用户通过“服务”控制面板 applet 启动，或者通过使用服务功能的 Windows 32 位应用程序（包括在 Microsoft Windows 32 位应用程序编程接口（API）中）启动。甚至当没有用户登录该系统时，服务也可以执行。

| 除非特别指出，否则在提到 Windows 9x 时，Windows 9x 指的是 Windows 98 和
| Windows ME。提到 Windows NT 时，Windows NT 指的是 Windows NT、Windows
| 2000、Windows XP 和 Windows Server 2003。在提到 Windows 时，Windows 表示所
| 有受支持的 Windows 操作系统。

使用实例

在实现数据库之前，应了解下列先决条件任务:

- 『在 UNIX 上启动 DB2 UDB』
- 第 5 页的『在 Windows 上启动 DB2 UDB』
- 第 5 页的『数据库管理器的多个实例』
- 第 7 页的『根据模式对对象分组』
- 第 7 页的『并行性』
- 第 10 页的『在数据库中启用数据分区』
- 第 11 页的『在 UNIX 上停止实例』

在 UNIX 上启动 DB2 UDB

在正常的业务活动期间，可能需要启动或停止 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）；例如，必须启动一个实例，然后才能执行下列任务:

- 在该实例中与一个数据库连接
- 预编译应用程序
- 将程序包与数据库绑定
- 存取主机数据库

先决条件:

要在系统上启动 DB2 UDB 实例:

1. 使用对该实例具有 SYSADM、SYSCTRL 或 SYSMAINT 权限的用户标识或名称进行登录；或者作为实例所有者登录。
2. 按如下所示运行启动脚本:

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

其中 INSTHOME 是您想要使用的实例的主目录。

过程:

使用这两种方法中的一种方法来启动实例:

1. 使用“控制中心”来启动实例:

1. 展开对象树，直到您看到**实例**文件夹为止。 2. 右键单击您想要启动的实例，并从弹出菜单中选择**启动**。

2. 使用命令行来启动实例，输入:

```
db2start
```

相关任务:

- 第 11 页的『在 UNIX 上停止实例』
- 第 22 页的『设置当前实例』
- 第 5 页的『在 Windows 上启动 DB2 UDB』

在 Windows 上启动 DB2 UDB

在正常业务活动期间，可能需要启动或停止 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）；例如，必须启动一个实例，然后才能执行下列任务:

- 在该实例中与一个数据库连接
- 预编译应用程序
- 将程序包与数据库绑定
- 存取主机数据库

先决条件:

要成功地从 **db2start** 启动 DB2 UDB（作为服务），用户帐户必须具有 Windows NT 操作系统定义的正确特权，才能启动 Windows 服务。用户帐户可以是“管理员”、“服务器操作员”或“高级用户”组的一个成员。

过程:

使用这两种方法中的一种方法来启动实例:

1. 使用“控制中心”来启动实例:

   1. 展开对象树，直到您看到**实例**文件夹为止。
   2. 右键单击您想要启动的实例，并从弹出菜单中选择**启动**。

2. 使用命令行来启动实例，输入:

   ```
   db2start
   ```

db2start 命令将 DB2 UDB 作为 Windows 服务来启动。通过在调用 **db2start** 时指定“/D”开关，仍可以在 Windows 上将 DB2 UDB 作为进程运行。使用“控制面板”或“NET START”命令也可以将 DB2 UDB 作为服务启动。

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

相关任务:

- 第 4 页的『在 UNIX 上启动 DB2 UDB』
- 第 11 页的『在 UNIX 上停止实例』
- 第 12 页的『在 Windows 上停止实例』

数据库管理器的多个实例

可在单台服务器上创建数据库管理器的多个实例。这意味着可以在一台物理机器上创建同一个产品的几个实例，并使它们同时运行。这在设置环境方面提供了灵活性。

您可能希望有多个实例来创建下列环境:

- 将开发环境与生产环境分离。
- 针对实例要服务的特定应用程序单独调整每一个实例。
- 保护敏感信息，使管理员存取不到它。例如，也许需要将工资单数据库保护在它自己的实例中，以使其它实例的所有者不能查看工资单数据。

注: （仅在 UNIX® 操作系统上: ）要防止两个或多个实例之间的环境冲突，应确保每个实例都有自己的主文件系统。如果共享主文件系统，将返回错误。

DB2® 通用数据库（DB2 UDB）程序文件以物理形式存储在特定机器上的某个位置中。创建的每个实例都指向此位置，这样不会为创建的每个实例复制程序文件。几个相关的数据库可以位于单个实例内。

在节点目录中将实例编目为本地的或远程的。缺省实例由 DB2INSTANCE 环境变量来定义。可以与其它实例连接（**ATTACH**），以便执行只能在实例级别执行的维护和实用程序任务，如创建数据库、强制断开应用程序、监视数据库或更新数据库管理器配置。当试图与不在缺省实例中的实例连接时，将使用该节点目录来确定如何与该实例通信。

相关概念:

- 第 17 页的『UNIX 操作系统上的多个实例』
- 第 17 页的『Windows 操作系统上的多个实例』

相关任务:

- 第 18 页的『创建附加实例』

相关参考:

- 『ATTACH Command』（*Command Reference*）

连接至数据库管理器的另一实例

要与另一个可能是远程的实例连接，使用 **ATTACH** 命令。

先决条件:

必须存在多个实例。

过程:

要使用“控制中心”连接至数据库管理器的另一实例:

1. 展开对象树，直到您看到**实例**文件夹为止。
2. 单击您想要连接的实例。
3. 右键单击选择的实例名。
4. 在“连接 DB2”窗口中，输入用户标识和密码，并单击**确定**。

要使用命令行来与实例连接，输入:

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

例如，要连接至节点目录中先前编目的称为 testdb2 的实例:

```
db2 attach to testdb2
```

在对 testdb2 实例执行维护活动之后，可通过运行以下命令从该实例拆离（**DETACH**）:

```
db2 detach
```

相关参考:

- 『ATTACH Command』（*Command Reference*）
- 『DETACH Command』（*Command Reference*）

根据模式对对象分组

数据库对象名可由单个标识组成，也可以是由两个标识组成的*模式限定对象*。模式限定对象的模式或高位部分提供了一种将数据库中的对象分类或分组的方法。当创建如表、视图、别名、单值类型、函数、索引、程序包或触发器之类的对象时，会给它分配一个模式。此赋值可显式或隐式地执行。

当在一条语句中引用对象时若使用了两部分对象名的高位部分，则显式使用了该模式。例如，用户 A 在模式 C 中发出 CREATE TABLE 语句，如下所示:

```
CREATE TABLE C.X (COL1 INT)
```

当不使用两部分对象名的高位部分时，即是隐式使用该模式。当发生这种情况时，CURRENT SCHEMA 专用寄存器用于标识完成对象名的高位部分所用的模式名。CURRENT SCHEMA 的初始值是当前会话用户的授权标识。若希望在当前会话期间更改它，可使用 SET SCHEMA 语句来将该专用寄存器设置为另一个模式名。

当创建数据库时，会在特定的模式内创建一些对象并将其存储在系统目录表中。

在动态 SQL 语句中，模式限定对象名隐式地使用 CURRENT SCHEMA 专用寄存器值来作为非限定对象名引用的限定符。在静态 SQL 语句中，QUALIFIER 预编译 / 绑定选项隐式地指定非限定数据库对象名的限定符。

在创建自己的对象之前，需要考虑是按自己的模式创建还是通过使用将对象按逻辑分组的另一种模式来创建。若正在创建将共享的对象，则使用不同的模式名将会非常有用。

相关概念:

- 第 61 页的『系统目录表的定义』

相关任务:

- 第 75 页的『创建模式』

相关参考:

- 『SET SCHEMA statement』（*SQL Reference, Volume 2*）
- 『CURRENT SCHEMA special register』（*SQL Reference, Volume 1*）

并行性

必须修改配置参数，以利用数据库分区内或非分区数据库内的并行性。例如，可以使用分区内并行性来利用对称多处理器（SMP）机器上的多个处理器。

启用分区内查询并行性

过程:

根据数据库分区的数目和数据在这些分区上的分布，分区间并行性自动生效。

相关概念:

- 『分区和处理器环境』（*《管理指南: 计划》*）
- 『数据分区』（*《管理指南: 计划》*）
- 『数据库分区组设计』（*《管理指南: 计划》*）
- 『分区数据库中的分区』（*《管理指南: 性能》*）

相关任务:

- 第 8 页的『对查询启用分区内并行性』
- 第 10 页的『在数据库中启用数据分区』
- 『在分区之间再分发数据』（*《管理指南: 性能》*）

对查询启用分区内并行性

过程:

可使用“控制中心”来了解或修改特定数据库中或数据库管理器配置文件中的各个条目的值。

还可以使用 **GET DATABASE CONFIGURATION** 和 **GET DATABASE MANAGER CONFIGURATION** 命令来了解特定数据库或数据库管理器配置文件中的各个条目的值。要修改特定数据库或数据库管理器配置文件中的各个条目，可分别使用 **UPDATE DATABASE CONFIGURATION** 和 **UPDATE DATABASE MANAGER CONFIGURATION** 命令。

影响分区内并行性的配置参数包括 *max_querydegree* 和 *intra_parallel* 数据库管理器参数以及 *dft_degree* 数据库参数。

要使分区内查询并行性可用，必须修改一个或多个数据库配置参数、数据库管理器配置参数、预编译或绑定选项或专用寄存器。

intra_parallel
: 用来指定数据库管理器是否可使用分区内并行性的数据库管理器配置参数。缺省值是不使用分区内并行性。

max_querydegree
: 指定用于在此实例上运行的任何 SQL 语句的最大程度分区内并行性的数据库管理器配置参数。当在某个分区内运行并行操作时，SQL 语句将不会使用超过此参数给出的数字。要使用 *max_querydegree* 中的值，必须将 *intra_parallel* 配置参数设置为“YES”。此配置参数的缺省值是 -1。此值表示系统使用优化器确定的并行度；否则，使用用户指定的值。

dft_degree
: 数据库配置参数。提供 DEGREE 绑定选项和 CURRENT DEGREE 专用寄存器的缺省值。缺省值为 1。值 ANY 表示系统使用优化器确定的并行度。

DEGREE
: 静态 SQL 的预编译或绑定选项。

CURRENT DEGREE
: 动态 SQL 的专用寄存器。

相关概念:

- 『应用程序的并行处理』（*《管理指南: 性能》*）
- 『并行处理信息』（*《管理指南: 性能》*）

相关任务:

- 『用配置参数配置 DB2』（*《管理指南: 性能》*）

相关参考:

- 『max_querydegree - 最大查询并行度配置参数』（*《管理指南: 性能》*）
- 『intra_parallel - 启用分区内并行性配置参数』（*《管理指南: 性能》*）
- 『dft_degree - 缺省并行度配置参数』（*《管理指南: 性能》*）
- 『BIND Command』（*Command Reference*）
- 『PRECOMPILE Command』（*Command Reference*）
- 『CURRENT DEGREE special register』（*SQL Reference, Volume 1*）

为实用程序启用分区内并行性

本节概述如何为下列实用程序启用分区内并行性:

- 装入
- 创建索引
- 备份数据库或表空间
- 复原数据库或表空间

根据数据库分区的数目，实用程序的分区间并行性自动生效。

启用并行性以装入数据: 装入实用程序自动利用并行性，也可在 **LOAD** 命令上使用下列参数:

- CPU_PARALLELISM
- DISK_PARALLELISM

在分区数据库环境中，当对多分区定义目标表时，用于数据装入的分区间并行性就会自动发生。用于数据装入的分区间并行性可通过指定 OUTPUT_DBPARTNUMBS 来覆盖。装入实用程序还会根据目标分区的大小智能化启用数据分区并行性。可使用 MAX_NUM_PART_AGENTS 来控制装入实用程序选择的最大并行度。可在指定 ANYORDER 的同时指定 PARTITIONING_DBPARTNUMS 以覆盖数据分区并行性。

相关概念:

- 『Load Overview』（*Data Movement Utilities Guide and Reference*）
- 『Partitioned database load - overview』（*Data Movement Utilities Guide and Reference*）

在创建索引时启用并行性: 要在创建索引时启用并行性:

- *intra_parallel* 数据库管理器配置参数必须为 ON
- 表必须足够大，以便能从并行性受益
- 在 SMP 机器上必须启用多个处理器

相关参考:

- 『intra_parallel - 启用分区内并行性配置参数』（*《管理指南: 性能》*）
- 『CREATE INDEX statement』（*SQL Reference, Volume 2*）

在备份数据库或表空间时启用 I/O 并行性: 要在备份数据库或表空间时启用 I/O 并行性:

- 使用多个目标媒体。
- 通过定义多个容器为表空间配置并行 I/O，或将单个容器与多个磁盘配合使用，并适当地使用 DB2_PARALLEL_IO 注册表变量。如果想要利用并行 I/O，则必须考虑在定义任何容器之前必须完成的操作。不能在您认为需要时才执行此操作；必须在需要备份数据库或表空间之前进行计划。
- 在 **BACKUP** 命令上使用 PARALLELISM 参数以指定并行度。
- 在 **BACKUP** 命令上使用 WITH num-buffers BUFFERS 参数以保证提供足够的缓冲区来满足该并行度。缓冲区数应比已有的目标媒体数与选择的并行度之和略大。

  同时使用满足以下条件的备份缓冲区大小:
  - 尽可能大。4 MB 或 8 MB（1024 或 2048 页）比较合适。
  - 至少等于要备份的表空间（块大小 * 容器数之积）最大的产品。

相关参考:

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

在复原数据库或表空间时启用 I/O 并行性: 要在复原数据库或表空间时启用 I/O 并行性:

- 使用多个源媒体。
- 为并行 I/O 配置表空间。在定义容器之前必须决定是否使用此选项。不能在您认为需要时才执行此操作；必须在需要复原数据库或表空间之前进行计划。
- 在 **RESTORE** 命令上使用 PARALLELISM 参数以指定并行度。
- 在 **RESTORE** 使用上使用 WITH num-buffers BUFFERS 参数以保证提供足够的缓冲区来满足该并行度。缓冲区数应比已有的目标媒体数与选择的并行度之和略大。

  同时使用满足以下条件的复原缓冲区大小:
  - 尽可能大。4 MB 或 8 MB（1024 或 2048 页）比较合适。
  - 至少等于要复原的表空间中（块大小 * 容器数之积）的最大者。
  - 等于备份缓冲区大小或是其偶数倍。

相关参考:

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

在数据库中启用数据分区

必须在创建数据库之前决定是否让数据库在分区环境中运行。在作出数据库设计决定时，必须确定是否应通过对数据库进行分区来作出性能改进。

下面是有关作出创建分区数据库的决定的一些注意事项。

过程:

当在分区数据库环境中运行时，可以使用 **CREATE DATABASE** 命令或 sqlecrea() 应用程序编程接口（API）从 db2nodes.cfg 文件中的任何节点创建数据库。

在创建分区数据库之前，必须选择哪个数据库分区将作为该数据库的目录节点。然后，可以从该分区直接创建数据库，或从连接至该分区的远程客户机创建数据库。您要连接并对其执行 CREATE DATABASE 命令的数据库分区成为该特定数据库的*目录节点*。

目录节点是用于存储所有系统目录表的数据库分区。对系统表的所有存取都必须通过此数据库分区进行。所有联合数据库对象（包装器、服务器以及昵称等）都存储在此节点的系统目录表中。

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

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

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

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

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

必须修改配置参数，才能利用数据分区。使用 **GET DATABASE CONFIGURATION** 和 **GET DATABASE MANAGER CONFIGURATION** 命令以了解特定数据库或数据库管理器配置文件中各个条目的值。要修改特定数据库或数据库管理器配置文件中的各个条目，可分别使用 **UPDATE DATABASE CONFIGURATION** 和 **UPDATE DATABASE MANAGER CONFIGURATION** 命令。

影响分区数据库的数据库管理器配置参数包括 *conn_elapse*、*fcm_num_anchors*、*fcm_num_buffers*、*fcm_num_connect*、*fcm_num_rqb*、*max_connretries*、*max_coordagents*、*max_time_diff*、*num_poolagents* 和 *stop_start_time*。

相关任务:

- 『用配置参数配置 DB2』（*《管理指南: 性能》*）

相关参考:

- 『sqlecrea - Create Database』（*Administrative API Reference*）
- 『CREATE DATABASE Command』（*Command Reference*）

在 UNIX 上停止实例

您可能需要停止数据库管理器的当前实例。

先决条件:

要在系统上停止实例，必须执行下列操作:

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

限制:

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

注: 若命令行处理器会话与一个实例连接，则在运行 **db2stop** 命令前必须运行 **terminate** 命令来结束每个会话。**db2stop** 命令停止由 DB2INSTANCE 环境变量定义的实例。

过程:

使用下列两个方法中的一个停止实例:

1. 要使用“控制中心”停止实例:

   1. 展开对象树，直到您找到**实例**文件夹为止。
   2. 单击您想要停止的每个实例。
   3. 右键单击选择的任何实例，并从弹出菜单中选择**停止**。
   4. 在“确认停止”窗口上，单击**确定**。

2. 使用命令行来停止实例，输入:

   ```
   db2stop
   ```

可以使用 db2stop 命令来停止或删除分区数据库环境中的各个分区。当在分区数据库中工作时，试图使用以下命令删除逻辑分区

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

必须确保没有用户在试图存取该数据库。如果有的话，将接收到错误消息 SQL6030N。

相关参考:

- 『db2stop - Stop DB2 Command』（*Command Reference*）
- 『TERMINATE Command』（*Command Reference*）

在 Windows 上停止实例

您可能需要停止数据库管理器的当前实例。

先决条件:

要在系统上停止实例，必须执行下列操作:

1. 停止 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）服务的用户帐户必须具有 Windows 操作系统定义的正确特权。用户帐户可以是“管理员”、“服务器操作员”或“高级用户”组的一个成员。
2. 显示与要停止的特定数据库连接的所有应用程序和用户。要确保没有关键性的或极重要的应用程序在运行，列出应用程序。为此，需要 SYSADM、SYSCTRL 或 SYSMAINT 权限。
3. 强制所有应用程序和用户与该数据库断开。需要 SYSADM 或 SYSCTRL 权限来强制用户。

限制:

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

注: 若命令行处理器会话与一个实例连接，则在运行 **db2stop** 命令前必须运行 **terminate** 命令来结束每个会话。**db2stop** 命令停止由 DB2INSTANCE 环境变量定义的实例。

过程:

要在系统上停止实例，使用下列其中一个方法:

- **db2stop**
- 使用“控制中心”停止服务。

1. 展开对象树，直到您找到**实例**文件夹为止。
2. 单击您想要停止的每个实例。
3. 右键单击选择的任何实例，并从弹出菜单中选择**停止**。
4. 在“确认停止”窗口上，单击**确定**。

- 使用“NET STOP”命令停止。
- 从应用程序中停止实例。

请记住，在分区数据库环境中使用 DB2 UDB 时，每个数据库分区服务器都是作为服务启动的。必须停止每个服务。

相关参考:

- 『db2stop - Stop DB2 Command』（*Command Reference*）

准备创建数据库

在实际创建数据库之前，应将许多概念和任务作为要完成的工作的一部分加以考虑。这些概念和任务包括设计数据库及建立使用数据库所需的实例、目录和其它支持文件。这些主题包括:

- 设计逻辑和物理数据库特征
- 实例创建
- 环境变量和概要文件注册表
- DB2 管理服务器
- 创建节点配置文件

- 创建数据库配置文件
- 快速通信管理程序（FCM）通信

设计逻辑和物理数据库特征

在创建数据库之前，必须决定如何设计逻辑数据库和物理数据库。要了解有关逻辑数据库和物理数据库设计的更多信息，请参阅*《管理指南: 计划》*。

实例创建

实例是一个逻辑数据库管理器环境，可在其中对数据库进行编目并设置配置参数。根据需要，可创建多个实例。可使用多个实例执行以下操作:

- 将一个实例用作开发环境，将另一个实例用作生产环境。
- 调整一个实例以用作特定的环境。
- 限制对敏感信息的存取。
- 控制每个实例中对 SYSADM、SYSCTRL 和 SYSMAINT 权限的指定。
- 优化每个实例的数据库管理器配置。
- 限制实例失败所带来的影响。若一个实例失败，则只影响一个实例。其它实例可继续正常运行。

应注意多个实例存在一些小缺点:

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

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

- 数据库管理器配置文件
- 系统数据库目录
- 节点目录
- 节点配置文件（`db2nodes.cfg`）
- 包含调试信息（例如异常或寄存器转储或用于 DB2® 通用数据库（DB2 UDB）进程的调用堆栈）的其它任何文件。

在 UNIX® 操作系统上，该实例目录位于 `INSTHOME/sqllib` 目录，其中 `INSTHOME` 是实例所有者的主目录。

在 Windows® 操作系统上，该实例目录位于 `/sqllib` 子目录中，即安装 DB2 UDB 的目录。

在分区数据库系统中，该实例目录是由属于该实例的所有数据库分区服务器共享的。因此，必须在该实例中的所有机器可以访问的一个网络共享驱动器上创建实例目录。

安装过程中，创建一个 DB2 UDB 的初始实例，称为 “DB2”。在 UNIX 上，可以随意命名该初始实例，只要符合命名规则。实例名用于设置目录结构。

要想立即使用此实例，在安装期间设置下列各项:

- 将环境变量 DB2INSTANCE 设置为 “DB2”。

- 将 DB2 注册表变量 DB2INSTDEF 设置为 “DB2”。

在 UNIX 上，只要符合命名规则，可以随意命名该缺省实例。

在 Windows 上，实例名与服务的名称相同，因此不应有冲突。必须具有正确权限才能创建服务。

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

在使用 DB2 UDB 之前，必须更新每个用户的数据库环境，以便该环境可存取实例并运行 DB2 UDB 程序。这适用于所有用户（包括管理类用户）。

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

样本脚本包含用于执行以下操作的语句:

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

相关概念:

- 第 17 页的『UNIX 操作系统上的多个实例』
- 第 17 页的『Windows 操作系统上的多个实例』

相关任务:

- 第 21 页的『添加实例』
- 第 19 页的『创建实例时的 UNIX 详细信息』
- 第 20 页的『创建实例时产生的 Windows 详细信息』
- 第 22 页的『设置当前实例』
- 第 22 页的『自动启动实例』
- 第 23 页的『并行运行多个实例』
- 第 21 页的『列出实例』
- 第 18 页的『创建附加实例』

在 UNIX 上自动设置 DB2 UDB 环境

在缺省情况下，创建实例时设置数据库环境的脚本仅在当前会话期间才会影响用户环境。可更改 .profile 文件，以便当用户使用 Bourne 或 Korn shell 程序登录时可自动运行 db2profile 脚本。对于 C shell 的用户，可更改 .login 文件以使它可运行 db2shrc 脚本文件。

过程:

将下列其中一条语句添加到 .profile 或 .login 脚本文件:

- 对于共享该脚本的一个版本的用户，添加:

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

其中 INSTHOME 是希望使用的实例的主目录。

- 对于在其主目录中存在该脚本的定制版本的用户，添加:

```
. USERHOME/db2profile         （对于 Bourne 或 Korn shell 程序）
  source USERHOME/db2cshrc      （在 C shell 中）
```

其中 USERHOME 是用户的主目录。

相关任务:

- 第 16 页的『在 UNIX 上自动设置 DB2 UDB 环境』

在 UNIX 上自动设置 DB2 UDB 环境

过程:

要选择您要使用的实例，在命令提示符处输入下列其中一条语句。句点（.）和空格是必需的。

- 对于共享该脚本的一个版本的用户，添加:

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

其中 INSTHOME 是希望使用的实例的主目录。

- 对于在其主目录中存在该脚本的定制版本的用户，添加:

```
. USERHOME/db2profile         （对于 Bourne 或 Korn shell 程序）
  source USERHOME/db2cshrc      （在 C shell 中）
```

其中 USERHOME 是用户的主目录。

若要同时使用多个实例，则在单独的窗口中对要使用的每个实例运行该脚本。例如，假设有两个实例 test 和 prod，它们的主目录是 /u/test 和 /u/prod。

在窗口 1 中:

- 在 Bourne 或 Korn shell 程序中，输入:

```
. /u/test/sqllib/db2profile
```

- 在 C shell 中，输入:

```
source /u/test/sqllib/db2cshrc
```

在窗口 2 中:

- 在 Bourne 或 Korn shell 程序中，输入:

```
. /u/prod/sqllib/db2profile
```

- 在 C shell 中，输入:

```
source /u/prod/sqllib/db2cshrc
```

用窗口 1 实现 test 实例，用窗口 2 实现 prod 实例。

注: 输入 **which db2** 命令，以保证搜索路径已正确设置。此命令返回 CLP 可执行文件的绝对路径。验证它位于该实例的 sqllib 目录下。

相关任务:

- 第 15 页的『在 UNIX 上自动设置 DB2 UDB 环境』

UNIX 操作系统上的多个实例

在 UNIX® 操作系统上可能有多个实例。但是，每次只能在 DB2® 通用数据库（DB2 UDB）的一个实例内工作。

注: 要防止两个或多个实例之间的环境冲突，应确保每个实例都有它自己的主文件系统。如果共享主文件系统，会返回错误。

实例所有者和“系统管理”（SYSADM）组与每个实例相关。实例所有者和 SYSADM 组是在创建实例期间指定的。一个用户标识或用户名只能用于唯一一个实例。该用户标识或用户名也称为*实例所有者*。

每个实例所有者必须有一个唯一的主目录。运行实例需要的所有文件都在该实例所有者的用户标识或用户名的主目录中创建。

若需要从系统中除去实例所有者的用户标识或用户名，可能会丢失与该实例相关的文件并失去对存储在此实例中的数据的存取权。因此，建议将要独占使用的实例所有者用户标识或用户名专用于运行 DB2 UDB。

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

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

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

相关任务:

- 第 19 页的『创建实例时的 UNIX 详细信息』

Windows 操作系统上的多个实例

可能在同一台机器上运行 DB2® 通用数据库（DB2 UDB）的多个实例。DB2 UDB 的每个实例维护其自己的数据库并具有其自己的数据库管理器配置参数。

DB2 UDB 的实例由下列内容组成:

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

  注: 不会为 Windows 98 、Windows ME 或客户机实例创建 Windows 服务。

- 实例目录。此目录包含数据库管理器配置文件、系统数据库目录、节点目录、DCS 数据库目录以及与实例相关联的所有诊断日志和转储文件。缺省情况下，实例目录是 SQLLIB 目录中的一个子目录，并具有与实例名相同的名称。例如，实例 “DB2” 的实例目录是 C:\SQLLIB\DB2，其中 C:\SQLLIB 是安装 DB2 UDB 的目录。可使用注册表变量 DB2INSTPROF 来更改实例目录的缺省位置。如果将 DB2INSTPROF 注册表变量设置为另一位置，则会在 DB2INSTPROF 指向的目录下创建实例目录。例如，如果 DB2INSTPROF=D:\DB2PROFS，则实例目录将为 D:\DB2PROFS\DB2。
- HKEY_LOCAL_MACHINE\SOFTWARE\IBM®\DB2\PROFILES\<instance_name> 下有一个注册表键。会在此处创建所有实例级别注册表变量。

可以同时运行多个 DB2 UDB 实例。要使用实例，在对该实例发出命令之前，需要将 DB2INSTANCE 环境变量设置为实例的名称。

要阻止实例存取另一实例的数据库，可在与实例同名的目录下为实例创建数据库文件。例如，在驱动器 C: 上为实例 DB2 创建数据库时，会在称为 C:\DB2 的目录中创建数据库文件。类似地，在驱动器 C: 上为实例 TEST 创建数据库时，会在称为 C:\TEST 的目录中创建数据库文件。

相关概念:

- 『高可用性』（*《数据恢复及高可用性指南与参考》*）

相关任务:

- 第 20 页的『创建实例时产生的 Windows 详细信息』

创建附加实例

虽然实例是作为 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）安装的一部分创建的，但是业务需求可能需要您创建其它实例。

先决条件:

如果在 Windows 上属于 “管理员” 组，或者在 UNIX 平台上具有 root 用户权限，就可以添加其它 DB2 UDB 实例。添加实例的机器成为 “实例拥有的机器”（节点零）。一定要在 DB2 管理服务器驻留的机器上添加实例。

过程:

要使用命令行添加实例，输入:

```
db2icrt <instance_name>
```

当使用 **db2icrt** 命令来添加 DB2 UDB 的另一实例时，应该提供实例所有者的登录名，并可选择是否指定该实例的认证类型。该认证类型适用于在该实例下创建的所有数据库。认证类型是对将在何处进行用户认证的说明。

可在 DB2PATH 中使用 DB2INSTPROF 环境变量更改实例目录的位置。需要该实例目录的写存取权。如果想要在不同于 DB2PATH 的路径中创建目录，在输入 **db2icrt** 命令*之前*，必须设置 DB2INSTPROF。

对于 DB2 通用数据库企业服务器版（ESE），还需要声明正在添加的新实例是分区数据库系统。另外，使用具有多分区的 ESE 实例，并使用 “快速通信管理程序”（FCM）

时，通过在创建实例时定义更多 TCP/IP 端口可以在分区间拥有多个连接。例如，对于 Windows 操作系统，使用具有 **-r <port range>** 参数的 **db2icrt** 命令。端口范围显示如下:

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

其中 base_port 是 FCM 可以使用的第一个端口，end_port 是 FCM 可以使用的端口号范围内的最后一个端口。

相关概念:

- 第 188 页的『服务器的认证方法』
- 第 192 页的『远程客户机的认证注意事项』

相关参考:

- 『db2icrt - Create Instance Command』（*Command Reference*）

创建实例时的 UNIX 详细信息

当使用 UNIX 操作系统时，**db2icrt** 命令具有下列可选的参数:

- –h 或 –?

  此参数用于显示此命令的帮助菜单。

- –d

  此参数设置在确定问题期间要使用的调试方式。

- –a AuthType

  此参数指定该实例的认证类型。有效的认证类型是 SERVER、SERVER_ENCRYPT 或 CLIENT。如果未指定此参数，且如果安装了 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）服务器，则缺省值为 SERVER。否则，将它设置为 CLIENT。

  注:

  1. 该实例的认证类型适用于实例拥有的所有数据库。
  2. 在 UNIX 操作系统上，认证类型 DCE 不是有效的选项。

- –u FencedID

  此参数是受防护的用户定义的函数（UDF）和存储过程执行期间所归属的用户。如果安装了 DB2 UDB 客户机或 “DB2 UDB 应用程序开发客户机”，则不需要此参数。对于其它 DB2 UDB 产品，这是必需参数。

  注: FencedID 不能是 “root” 或 “bin”。

- –p PortName

  此参数指定要使用的 TCP/IP 服务名称或端口号码。对于实例中的每个数据库，将在实例的数据库配置文件中设置此值。

- –s InstType

  允许创建不同类型的实例。有效的的实例类型是: `ese`、`wse`、`client` 和 `standalone`。

示例:

- 要为 DB2 UDB 服务器添加实例，您可以使用下列命令:

  ```
  db2icrt -u db2fenc1 db2inst1
  ```

- 若只安装了 DB2 Connect 企业版，也可将该实例名用作 Fenced ID:

  ```
  db2icrt -u db2inst1 db2inst1
  ```

- 要为 DB2 UDB 客户机添加实例，您可以使用下列命令:

  ```
  db2icrt db2inst1 –s client –u fencedID
  ```

当要让一个工作站与其它数据库服务器连接，且该工作站上不需要本地数据库时，创建 DB2 UDB 客户机实例。

相关参考:

- 『db2icrt - Create Instance Command』（*Command Reference*）

创建实例时产生的 Windows 详细信息

当使用 Windows 操作系统时，**db2icrt** 命令具有下列可选的参数:

- –s InstType

  允许创建不同类型的实例。有效的的实例类型是: ese、wse、client 和 standalone。

- –p:InstProf_Path

  这是一个可选的参数，用来指定另一实例概要文件路径。若不指定该路径，将在 SQLLIB 目录下创建实例目录，并将共享名称 DB2 与实例名并置后的名称作为其名称。自动将读写权限授予域中的每个人。可以更改许可权，以限制该目录的存取权。

  若指定另一个实例概要文件路径，必须创建一个共享的驱动器或目录。除非更改了许可权，否则这将允许域中的每个用户有机会存取实例目录。

- –u:username,password

  当创建分区数据库环境时，必须声明 DB2 通用数据库服务的域／用户帐户名和密码。

- –r:base_port,end_port

  这是一个可选的参数，用于指定“快速通信管理程序”（FCM）的 TCP/IP 端口范围。若指定 TCP/IP 端口范围，则必须确保该端口范围在分区数据库系统中的所有机器上都可用。

以下示例可以用于 DB2 通用数据库（DB2 UDB）企业服务器版 Windows 版:

```
db2icrt inst1 –s ese
      –p:\\machineA\db2mpp      –u:<user account name>,<password> –r:9010,9015
```

注: 如果更改服务帐户；即，如果不再使用产品安装期间创建第一个实例时创建的缺省服务，则必须将下列高级权限授予用于创建实例的域／用户帐户名:

- 充当操作系统的一部分
- 创建标记对象
- 增加份额
- 作为服务登录
- 替换进程级标记
- 锁定内存中的页

该实例需要这些用户权限才能存取共享驱动器、认证用户帐户和将 DB2 UDB 作为 Windows 服务运行。需要“锁定内存中的页”权限来支持“地址窗口扩展”（AWE）。

相关参考:

- 『db2icrt - Create Instance Command』（*Command Reference*）

添加实例

过程:

一旦创建了附加实例，就需要在“控制中心”中添加该实例的记录才能从“控制中心”使用该实例。

要添加另一实例，执行下列步骤:

1. 使用具有“管理”权限或属于本地“管理员”组的用户标识或名称登录。
2. 要添加实例，使用下列其中一种方法:

使用“控制中心”:

1. 展开对象树，直到找到您想要的系统的**实例**文件夹为止。
2. 右键单击实例文件夹，并从弹出菜单中选择**添加**。
3. 填写信息，并单击**应用**。

相关概念:

- 第 14 页的『实例创建』

相关任务:

- 第 21 页的『列出实例』

列出实例

过程:

要使用“控制中心”获得系统上可用的所有实例的列表:

1. 展开对象树，直到您看到**实例**文件夹为止。
2. 右键单击“实例”文件夹，并从弹出菜单中选择**添加**。
3. 在“添加实例”窗口上，单击**刷新**。
4. 单击下拉箭头来查看数据库实例的列表。
5. 单击**取消**以退出该窗口。

要使用命令行获得系统上可用的所有实例的列表，输入:

```
db2ilist
```

要确定哪一个实例适用于当前会话（在受支持的 Windows 平台上），使用:

```
set db2instance
```

相关参考:

- 『db2ilist - List Instances Command』（*Command Reference*）

设置当前实例

过程:

当运行命令来启动或停止实例的数据库管理器时，DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）将该命令应用于当前实例。DB2 UDB 确定当前实例，如下所示:

- 若为当前会话设置 DB2INSTANCE 环境变量，则其值为当前实例。要设置 DB2INSTANCE 环境变量，输入:

  ```
  set db2instance=<new_instance_name>
  ```

- 若没有为当前会话设置 DB2INSTANCE 环境变量，则 DB2 UDB 使用系统环境变量中 DB2INSTANCE 环境变量的设置。在 Windows NT 上，在“系统环境”中设置系统环境变量。在 Windows 9x 上，应在 autoexec.bat 文件中设置它们。
- 如果根本没有设置 DB2INSTANCE 环境变量，则 DB2 UDB 使用注册表变量 DB2INSTDEF。

  要在注册表的全局级别设置 DB2INSTDEF 注册表变量，输入:

  ```
  db2set db2instdef=<new_instance_name> -g
  ```

要确定哪一个实例适用于当前会话，输入:

```
db2 get instance
```

相关任务:

- 第 26 页的『声明注册表和环境变量』

自动启动实例

过程:

在 Windows 操作系统上，缺省情况下，安装期间创建的 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）实例设置为自动启动。使用 **db2icrt** 创建的实例设置为手工启动。要更改启动类型，需要转至“服务”面板并在其中更改 DB2 UDB 服务的属性。

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

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

其中 <instance name> 是实例的登录名。

在 UNIX 操作系统上，要阻止一个实例在每次系统重新启动后自动启动，输入以下命令:

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

其中 <instance name> 是实例的登录名。

相关概念:

- 第 14 页的『实例创建』

相关参考:

- 『db2iauto - Auto-start Instance Command』（*Command Reference*）

并行运行多个实例

过程:

要使用“控制中心”并行地运行多个实例:

1. 展开对象树，直到您找到**数据库**文件夹为止。
2. 右键单击一个实例，并从弹出菜单中选择**启动**。
3. 重复步骤 2，直到启动了要并行运行的所有实例为止。

（仅在 Windows 上: ）要使用命令行并行运行多个实例:

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

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

2. 输入 **db2start** 命令以启动该实例。

相关概念:

- 第 5 页的『数据库管理器的多个实例』

相关任务:

- 第 19 页的『创建实例时的 UNIX 详细信息』
- 第 20 页的『创建实例时产生的 Windows 详细信息』
- 第 18 页的『创建附加实例』

许可证管理

DB2® 通用数据库（DB2 UDB）产品的许可证管理主要是通过该产品的联机界面即“控制中心”内的“许可证中心”来进行的。从“许可证中心”可检查每个安装的产品的许可证信息、统计信息、注册的用户和当前用户。

当“控制中心”不能使用时，**db2licm** 许可管理工具命令执行基本许可证功能。使用此命令，可以添加、除去、列示和修改本地系统上安装的许可证和策略。

相关参考:

- 『db2licm - License Management Tool Command』（*Command Reference*）

环境变量和概要文件注册表

环境变量和注册表变量控制数据库环境。

可以使用“配置助手”（**db2ca**）来配置配置参数和注册表变量。

在引入 DB2® 通用数据库（DB2 UDB）概要文件注册表之前，在 Windows® 工作站上更改环境变量（举例说明）要求您更改环境变量并重新引导。现在，除少数例外情况，您的环境均由 DB2 UDB概要文件注册表中存储的注册表变量控制。UNIX® 操作系

统上，对于给定的实例来说，具有系统管理（SYSADM）权限的用户可更新该实例的注册表值。Windows 用户不需要 SYSADM 权限就可以更新注册表变量。使用 **db2set** 命令来更新注册表变量，而不需重新引导；此信息会立即存储在概要文件注册表中。该 DB2 UDB 注册表将更新的信息应用于更改后启动的 DB2 UDB 服务器实例和 DB2 UDB 应用程序。

当更新注册表时，更改不会影响当前正在运行的 DB2 UDB 应用程序或用户。在更新后启动的应用程序将使用新值。

注: 存在 DB2 UDB 环境变量 DB2INSTANCE 和 DB2NODE，它们可能未存储在 DB2 UDB 概要文件注册表中。在某些操作系统上，必须使用 **set** 才能更新这些环境变量。在下次重新引导系统之后，这些更改不再有效。在 UNIX 平台上，可以使用 **export** 命令来代替 **set** 命令。

使用概要文件注册表允许集中控制环境变量。现在通过不同的概要文件提供了不同级别的支持。当使用 DB2 管理服务器时，还提供了对环境变量的远程管理。

有四个概要文件注册表:

- DB2 UDB 实例级别概要文件注册表。大多数 DB2 UDB 环境变量都位于此注册表中。特定实例的环境变量设置保存在此注册表中。在此级别定义的值将覆盖在全局级别的对应设置。
- DB2 UDB 全局级别概要文件注册表。若未对特定的实例设置环境变量，则使用此注册表。此注册表具有在当前机器范围内有效的环境变量设置。在 DB2 UDB ESE 中，每台机器上都有一个全局级别概要文件。
- DB2 UDB 实例节点级别概要文件注册表。在多分区环境中，此注册表级别包含特定于分区（节点）的变量设置。在此级别定义的值将覆盖实例级别和全局级别的对应设置。
- DB2 UDB 实例概要文件注册表。此注册表包含此系统可识别的所有实例名的列表。可通过运行 db2ilist 查看系统上提供的所有实例的完整列表。

DB2 UDB 按下列顺序检查注册表值和环境变量并解析它们以配置操作环境:

1. 使用 set 命令设置的环境变量。（或 UNIX 平台上的 export 命令。）
2. 使用实例节点级别概要文件设置的注册表值（使用 **db2set -i <instance name> <nodenum>** 命令）。
3. 使用实例级别概要文件设置的注册表值（使用 **db2set -i** 命令）。
4. 使用全局级别概要文件设置的注册表值（使用 **db2set -g** 命令）。

实例级别概要文件注册表

在与分区数据库环境工作时，会出现一对 UNIX 和 WINDOWS 差异。这些差异如以下示例所示。

假设分区数据库环境有三个以“红色”、“白色”和“蓝色”标识的物理节点。在 UNIX 平台上，如果实例所有者从任一节点运行下述命令:

```
db2set -i FOO=BAR
```

或

```
db2set FOO=BAR        ('-i' is implied)
```

对于当前实例的所有节点（即“红色”、“白色”和“蓝色”节点），FOO 的值将是可视的。

在 UNIX 平台上，实例级别概要文件注册表存储在 sqllib 目录下的文本文件中。在分区数据库环境下，sqllib 目录位于所有物理节点共享的文件系统中。

在 Windows 平台上，如果用户从“红色”节点执行相同的命令，则 FOO 的值只对当前实例的“红色”节点可见。DB2 UDB 将实例级别概要文件注册表存储在 WINDOWS 注册表中。物理节点之间没有共享。要设置所有物理机器上的注册表变量，可使用以下“rah”命令:

```
rah db2set -i FOO=BAR
```

rah 将在“红色”、“白色”和“蓝色”节点远程运行 db2set 命令。

可以使用 DB2REMOTEPREG 将非实例所有机器上的注册表变量配置为引用实例所有机器上的注册表变量。这样可以有效创建环境以便将实例所有机器上的注册表变量共享给实例中所有的机器。

运用以上示例并假设“红色”节点为实例所有机器，即可通过下述操作将“白色”和“蓝色”机器上的 DB2REMOTEPREG 设置为共享“红色”机器上的注册表变量:

```
(on red) do nothing
(on white and blue) db2set DB2REMOTEPREG=\\red
```

DB2REMOTEPREG 的设置在设置后不必更改。

以下是 REMOTEPREG 的工作方法:

当 DB2 UDB 在 WINDOWS 上读取注册表变量时，它首先读取 DB2REMOTEPREG 值。如果已经设置 DB2REMOTEPREG，则打开 DB2REMOTEPREG 变量中指定了其名称的远程机器的注册表。将对指定的远程机器重定向注册表变量的后续读取和更新。

访问远程注册表需要在目标机器上运行“远程注册服务”。而且，用户登录帐户和所有 DB2 UDB 服务登录帐户对远程注册表有足够的存取权限。因此，要使用 DB2REMOTEPREG，需要在 Windows 域环境下操作，这样才能给域帐户授权必需的注册表存取权限。

Microsoft® Cluster Server (MSCS) 注意事项。不可在 MSCS 环境下使用 DB2REMOTEPREG。当运行于 MSCS 配置（在此配置下所有的机器属于同一 MSCS 群集）下时，注册表变量在群集注册表内维护。因此，它们已经在同一 MSCS 群集内的所有机器之间共享，在此情况下没有必要使用 DB2REMOTEPREG。

当运行于多分区故障转移环境（在此环境下，分区范围跨越多个 MSCS 群集）下时，因为实例所有机器的注册表变量驻留在群集注册表中，不能用 DB2REMOTEPREG 来指向实例所有机器。

相关概念:

- 『DB2 注册表和环境变量』（*《管理指南: 性能》*）

相关任务:

- 第 26 页的『声明注册表和环境变量』

声明注册表和环境变量

过程:

db2set 命令支持本地声明注册表变量（和环境变量）。

要显示该命令的帮助信息，使用:

```
db2set ?
```

要列出所有受支持的注册表变量的完整集合，使用:

```
db2set -lr
```

要列出当前实例或缺省实例的所有已定义的注册表变量，使用:

```
db2set
```

要列出概要文件注册表中所有定义的注册表变量，使用:

```
db2set -all
```

要显示一个注册表变量在当前实例或缺省实例中的值，使用:

```
db2set registry_variable_name
```

要显示一个注册表变量在所有级别的值，使用:

```
db2set registry_variable_name -all
```

要在当前实例或缺省实例中更改一个注册表变量，使用:

```
db2set registry_variable_name=new_value
```

要更改该实例中所有数据库的注册表变量缺省值，使用:

```
db2set registry_variable_name=new_value
      -i instance_name
```

要更改实例中特定分区的注册表变量缺省值，使用:

```
db2set registry_variable_name=new_value
      -i instance_name node_number
```

要更改系统中所有实例的注册表变量缺省值，使用:

```
db2set registry_variable_name=new_value -g
```

如果使用“轻量级目录访问协议”（LDAP），则可以使用下列命令在 LDAP 中设置注册表变量:

- 要在 LDAP 中设置用户级别的注册表变量，使用:

  ```
  db2set -ul
  ```

- 要在 LDAP 中设置全局级别的注册表变量，使用:

  ```
  db2set -gl user_name
  ```

当在 LDAP 环境中运行时，可以在 LDAP 中对 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）注册表变量值作如下设置: 作用域对属于某个目录分区或 Windows NT 域的所有机器和所有用户是全局的。当前，在 LDAP 全局级别只能设置两个 DB2 UDB 注册表变量: DB2LDAP_KEEP_CONNECTION 和 DB2LDAP_SEARCH_SCOPE。

例如，要在 LDAP 中的全局级别设置搜索范围值，使用:

```
db2set -gl db2ldap_search_scope = value
```

其中 *value* 可以是“local”（局部）、“domain”（域）或“global”（全局）。

注:

1. 通过 db2set 命令同时（即，同时或接近于同时）更新 DB2 UDB profile.env 文件时，profile.env 文件的大小减少为零。而且 `db2set -all` 的输出显示不一致的值。
2. 用于在机器全局级别设置 DB2 UDB 注册表变量的 `-g` 选项与特定于 LDAP 全局级别的 `-gl` 之间是有区别的。
3. 在 LDAP 环境中运行时，用户级别注册表变量仅在 Windows 上受支持。
4. 用户级别的变量设置包含用户特定变量设置。对用户级别的任何更改都会写入 LDAP 目录。
5. 在同一命令中不能同时使用参数“-i”、“-g”、“-gl”和“-ul”。
6. 某些变量将始终缺省为全局级别概要文件。不能在实例级别或节点级别概要文件（如 DB2SYSTEM 和 DB2INSTDEF）设置它们。
7. 在 UNIX 上，必须具有系统管理（SYSADM）权限，才能更改实例的注册表值。只有具有 root 用户权限的用户才能更改全局级别注册表中的参数。

要将实例的注册表变量复位回“全局概要文件注册表”中的缺省值，请使用:

```
db2set -r registry_variable_name
```

要将实例中节点的注册表变量复位回“全局概要文件注册表”中的缺省值，请使用:

```
db2set -r registry_variable_name node_number
```

要删除一个变量在特定级别的值，可使用相同的命令语法来设置该变量但不对该变量值指定任何内容。例如，要删除该变量在节点级别的设置，输入:

```
db2set registry_variable_name= -i instance_name
  node_number
```

要删除变量的值并限制其用途，如果它是在更高的概要文件级定义的，则输入:

```
db2set registry_variable_name= -null instance_name
```

此命令将删除指定参数的设置，并限制高级别的概要文件（在本示例中为 DB2 UDB 全局级别概要文件）更改此变量的值。但指定的变量仍可由低级别的概要文件（在本示例中为 DB2 UDB 节点级别概要文件）设置。

相关概念:

- 『DB2 注册表和环境变量』（*《管理指南: 性能》*）

相关任务:

- 第 28 页的『在 Windows 上设置环境变量』
- 第 29 页的『在 UNIX 系统上设置环境变量』
- 第 290 页的『搜索 LDAP 目录分区或域』
- 第 292 页的『在 LDAP 环境中设置用户级别的 DB2 注册表变量』

在 Windows 上设置环境变量

过程:

强烈建议所有特定的注册表变量都在 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）概要文件注册表中定义。如果 DB2 UDB 变量是在注册表外设置的，则不可能对这些变量进行远程管理，要使变量值生效，必须重新引导该工作站。

Windows 操作系统有一个系统环境变量 DB2INSTANCE，只能在概要文件注册表外设置它；但不要求您设置 DB2INSTANCE。可以在全局级别概要文件中设置 DB2 UDB 概要文件注册表变量 DB2INSTDEF，以指定要使用的实例名（如果未定义 DB2INSTANCE）。

Windows 上的“DB2 UDB 企业服务器版”服务器有两个系统环境变量 DB2INSTANCE 和 DB2NODE，这两个变量只能在概要文件注册表外进行设置。不要求您设置 DB2INSTANCE。可以在全局级别概要文件中设置 DB2 UDB 概要文件注册表变量 DB2INSTDEF，以指定要使用的实例名（如果未定义 DB2INSTANCE）。

DB2NODE 环境变量用于将请求路由至机器内的目标逻辑节点。必须在发出该应用程序或命令的会话中而不是在 DB2 UDB 概要文件注册表中设置此环境变量。如果未设置此变量，则目标逻辑节点缺省为在该机器上定义为零（0）的逻辑节点。

要确定环境变量的设置，使用 **echo** 命令。例如，要检查 DB2PATH 环境变量的值，输入:

```
echo %db2path%
```

要设置系统环境变量，执行下列操作:

在 Windows 9x 上: 编辑 `autoexec.bat` 文件，然后重新引导系统以使更改生效。

在 Windows 上: 可按如下所示设置 DB2 UDB 环境变量 DB2INSTANCE 和 DB2NODE（在此描述中使用 DB2INSTANCE）:

- （在 Windows NT 和 Windows 2000 上）选择**开始**、**设置**和**控制面板**。（在 Windows XP 和 Windows Server 2003 上）选择**开始** —> **控制面板**。
- （在 Windows NT 和 Windows 2000 上）双击**系统**图标。（在 Windows XP 和 Windows Server 2003 上）根据 Windows 主题和当前选择的视图类型，可能必须先选择**性能和维护**，然后才能选择**系统**图标。
- （在 Windows NT 上）在“系统控制面板”的“系统环境变量”部分中，执行下列操作:（在 Windows 2000、Windows XP 和 Windows Server 2003 上）在“系统属性”窗口中，必须选择**高级**选项卡，然后单击“环境变量”按钮并执行下列操作:
  1. 若 DB2INSTANCE 变量不存在:
     a. （在 Windows NT 上）选择任何系统环境变量。（在 Windows 2000、Windows XP 和 Windows Server 2003 上）单击**新建**按钮。
     b. （在 Windows NT 上），将变量字段中的名称更改为 DB2INSTANCE。（在 Windows 2000、Windows XP和 Windows Server 2003 上）用 DB2INSTANCE 填充*变量名*字段。

c. （在 Windows NT 上），将*值*字段更改为实例名，例如 db2inst。（在 Windows 2000、Windows XP 和 Windows Server 2003 上）用实例名填充*变量值*字段，例如 db2inst。

2. 若 DB2INSTANCE 变量已经存在，则追加新的值:
   a. 选择 DB2INSTANCE 环境变量。
   b. 将*值*字段更改为实例名，例如 db2inst。
3. （在 Windows NT 上）选择“设置”。（在 Windows 2000、Windows XP 和 Windows Server 2003 上）选择“确定”。
4. 选择“确定”。
5. 重新引导系统，以使这些更改生效。

注: 也可在会话（进程）级别设置环境变量 DB2INSTANCE。例如，如果要启动另一个 DB2 UDB 实例 TEST，在命令窗口中发出下列命令:

```
set DB2INSTANCE=TEST
    db2start
```

当在 C shell 中工作时，在命令窗口中发出下列命令:

```
setenv DB2INSTANCE TEST
```

概要文件注册表位置如下:

- “DB2 UDB 实例级别概要文件注册表”位于 Windows 操作系统注册表中，其路径为:

  \HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\PROFILES*instance_name*

  注: *instance_name* 是 DB2 UDB 实例的名称。
- “DB2 UDB 全局级别概要文件注册表”位于 Windows 注册表中，其路径为:

  \HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\GLOBAL_PROFILE
- “DB2 UDB 实例节点级别概要文件注册表”位于 Windows 注册表中，其路径为:

  ...\SOFTWARE\IBM\DB2\PROFILES*instance_name*\NODES*node_number*

  注: *instance_name* 和 *node_number* 特定于您正在使用的数据库分区。
- 不需要“DB2 UDB 实例概要文件注册表”。对于系统中的每个 DB2 UDB 实例，在下列路径中创建一个键:

  \HKEY_LOCAL_MACHINE\SOFTWARE\IBM\DB2\PROFILES*instance_name*

  实例列表可通过对 PROFILES 键中的键进行计数获取。

相关概念:

- 第 37 页的『DB2 管理服务器』

相关任务:

- 第 29 页的『在 UNIX 系统上设置环境变量』

在 UNIX 系统上设置环境变量

过程:

强烈建议所有特定的注册表变量都在 DB2 UDB 概要文件注册表中定义。如果 DB2 UDB 变量是在注册表外设置的，则不能对这些变量进行远程管理。

在 UNIX 操作系统上，必须设置系统环境变量 DB2INSTANCE。

提供脚本 db2profile（对于 Korn shell 程序）和 db2cshrc（对于 Bourne shell 或 C shell）作为示例，以帮助您设置数据库环境。可以在 insthome/sqllib 中找到这些文件，其中 insthome 是实例所有者的主目录。

这些脚本包括对下列各项的说明:

- 使用下列目录来更新用户的路径:
  - insthome/sqllib/bin
  - insthome/sqllib/adm
  - insthome/sqllib/misc
- 将 DB2INSTANCE 设置为用于执行的缺省本地 instance_name。

注: 除 PATH 和 DB2INSTANCE 外，其它所有支持的变量都必须在 DB2 UDB 概要文件注册表中设置。要设置 DB2 UDB 不支持的变量，请在脚本文件 userprofile 和 usercshrc 中定义它们。

实例所有者或 SYSADM 用户可以为一个实例的所有用户定制这些脚本。或者，用户可以复制和定制脚本，然后直接调用脚本或将它添加至它们的 .profile 或 .login 文件。

要更改当前会话的环境变量，发出类似于以下的命令:

- 对于 Korn shell 程序:

```
DB2INSTANCE=inst1
   export DB2INSTANCE
```

- 对于 Bourne shell:

```
export DB2INSTANCE=<inst1>
```

- 对于 C shell:

```
setenv DB2INSTANCE <inst1>
```

为了正确管理 DB2 UDB 概要文件注册表，在 UNIX 操作系统上必须遵循下列文件所有权规则。

- “DB2 UDB 实例级别概要文件注册表”文件位于:

  ```
  INSTHOME/sqllib/profile.env
  ```

  此文件的存取许可权和所有权应该是:

  ```
  -rw-rw-r-- <db2inst1> <db2iadm1> profile.env
  ```

  其中 <db2inst1> 是实例所有者，而 <db2iadm1> 是实例所有者的组。

  INSTHOME 是实例所有者的主路径。

- “DB2 UDB 全局级别概要文件注册表”位于:
  - /var/db2/<version_id>/default.env（对于 AIX、Solaris Operating Environment 和 Linux 操作系统，其中 <version_id> 是当前版本）。
  - /var/opt/db2/<version_id>/default.env（对于 HP-UX 操作系统，其中 <version_id> 是当前版本）。

  此文件的存取许可权和所有权应该是:

  ```
  -rw-rw-r-- <Instance_Owner> <Instance_Owner_Group> default.env
  ```

为了修改全局注册表变量，用户必须作为 root 用户登录。

- “DB2 UDB 实例节点级别概要文件注册表”位于:

  INSTHOME/sqllib/nodes/<node_number>.env

  该目录和此文件的存取许可权和所有权应该是:

  ```
  drwxrwsr-w <Instance_Owner> <Instance_Owner_Group> nodes

  -rw-rw-r-- <Instance_Owner> <Instance_Owner_Group> <node_number>.env
  ```

  INSTHOME 是实例所有者的主路径。

- “DB2 UDB 实例概要文件注册表”位于:
  - /var/db2/<version_id>/profiles.reg（对于 AIX、Solaris 和 Linux，其中 <version_id> 是当前版本）。
  - /var/opt/db2/<version_id>/profiles.reg（对于 HP-UX 操作系统，其中 <version_id> 是当前版本）。

  此文件的存取许可权和所有权应该是:

  ```
  -rw-r--r-- root system profiles.reg
  ```

相关概念:

- 第 37 页的『DB2 管理服务器』

相关任务:

- 第 28 页的『在 Windows 上设置环境变量』

创建节点配置文件

过程:

若数据库要在分区数据库环境中运行，必须创建一个称为 db2nodes.cfg 的节点配置文件。在可以使用多分区的并行能力来启动数据库管理器之前，此文件必须位于该实例的主目录 sqllib 子目录中。该文件包含一个实例中所有数据库分区的配置信息，并且它由该实例的所有数据库分区共享。

Windows 注意事项

如果正在 Windows 上使用 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）企业服务器版，则节点配置文件是在创建实例时创建的。您不应尝试手工修改节点配置文件。可使用 **db2ncrt** 命令来将数据库分区服务器添加至实例。可使用 **db2ndrop** 命令来删除实例的数据库分区服务器。可使用 **db2nchg** 命令来修改数据库分区服务器配置，包括将数据库分区服务器从一台机器移至另一台机器；更改 TCP/IP 主机名；或选择另一逻辑端口或网络名。

注: 不应该在不是 DB2 UDB 创建的 sqllib 子目录下创建文件或目录，以防止删除实例时丢失数据。但有两个例外情况。若系统支持存储过程，则将该存储过程应用程序置于 sqllib 子目录下的 function 子目录中。另一个例外是在已创建用户定义的函数（UDF）的情况下。允许 UDF 可执行程序位于同一个目录中。

对于属于一个实例的每个数据库分区，该文件都包含一行。每行的格式如下:

```
dbpartitionnum hostname [logical-port [netname]]
```

记号由空格定界。这些变量是:

dbpartitionnum

数据库分区号唯一地定义节点，可在 0 到 999 之间。数据库分区号必须以递增顺序排序。该顺序中可以有间隔。

一旦指定了数据库分区号，就不能更改它。（否则，指定将数据如何分区的分区映射中的信息将会被泄露。）

若删除一个节点，则它的数据库分区号可以再次用于添加的任何新节点。

数据库分区号用于在数据库目录中生成节点名。它的格式为:

NODE*nnnn*

nnnn 是数据库分区号，其左边以零填充。**CREATE DATABASE** 和 **DROP DATABASE** 命令也使用此数据库分区号。

hostname

用作分区间通信的 IP 地址的主机名。使用主机名的全限定名称。/etc/hosts 文件也应该使用全限定名称。如果未在 db2nodes.cfg 文件和 /etc/hosts 文件中使用全限定名称，则可能接收到错误消息 SQL30082N RC=3。

（指定 netname 时例外。在此情况下，netname 用于大多数通信，而 hostname 仅用于 **db2start**、**db2stop** 和 **db2_all**。）

logical-port

此参数是可选的，它指定该节点的逻辑端口号。此号码与数据库管理器实例名一起用来标识 etc/services 文件中的 TCP/IP 服务名称条目。

IP 地址和逻辑端口的组合用作公认地址，且在所有支持节点间通信连接的应用程序中是唯一的。

对于每个主机名，一个逻辑端口必须为 0（零）或空白（缺省为 0）。与此逻辑端口相关联的节点是与客户机连接的主机上的缺省节点。可以使用 db2profile 脚本中的 DB2NODE 环境变量或 sqlesetc() API 来覆盖它。

如果同一主机上具有多个节点（即，对于一个主机，有多个 dbpartitionnum），则应该从 0 开始无间隔地为逻辑节点指定 logical-port 编号。编号的顺序并不重要。

例如，以下设置是有效的:

```
0   cpaiss43.mach1.xxx.com   1
1   cpaiss43.mach1.xxx.com   0
2   cpaiss43.mach1.xxx.com   2
3   cpaiss44.mach1.xxx.com   0
```

netname

此参数是可选的，并且用于支持有多个活动 TCP/IP 接口的主机，每个接口有其自己的主机名。

以下示例显示 RS/6000 SP 系统的可能节点配置文件，在该系统上，SP2EN1 有多个 TCP/IP 接口和两个逻辑分区且使用 SP2SW1 作为 DB2 UDB 接口。它还显示了从 1 开始（而不是从 0 开始）的分区号，以及 dbpartitionnum 序列中的间隙:

表 1. 数据库分区号示例表。

dbpartitionnum	hostname	logical-port	netname
1	SP2EN1.mach1.xxx.com	0	SP2SW1
2	SP2EN1.mach1.xxx.com	1	SP2SW1

表 1. 数据库分区号示例表。 (续)

dbpartitionnum	hostname	logical-port	netname
4	SP2EN2.mach1.xxx.com	0	
5	SP2EN3.mach1.xxx.com		

可以使用选择的编辑器更新 db2nodes.cfg 文件。(例外情况: 不应在 Windows 上使用编辑器)。但是，由于数据分区要求不要更改数据库分区号，因此必须注意保护文件中信息的完整性。该节点配置文件是在发出 **db2start** 时锁定的，而在 **db2stop** 结束数据库管理器后解锁。文件被锁定时，**db2start** 命令可在必要时更新该文件。例如，可发出带有 RESTART 选项或 ADDNODE 选项的 **db2start**。

注: 如果 **db2stop** 命令不成功而又未解锁该节点配置文件，则发出 **db2stop FORCE** 来将其解锁。

相关概念:

- 『存储过程的准则』(*《管理指南: 性能》*)

相关参考:

- 『db2start - Start DB2 Command』(*Command Reference*)
- 『db2stop - Stop DB2 Command』(*Command Reference*)
- 『CREATE DATABASE Command』(*Command Reference*)
- 『DROP DATABASE Command』(*Command Reference*)
- 『db2nchg - Change Database Partition Server Configuration Command』(*Command Reference*)
- 『db2ncrt - Add Database Partition Server to an Instance Command』(*Command Reference*)
- 『db2ndrop - Drop Database Partition Server from an Instance Command』(*Command Reference*)

创建数据库配置文件

过程:

为每个数据库创建*数据库配置文件*。此文件的创建已完成。此文件包含影响数据库使用的各种*配置参数*的值，如:

- 当创建数据库时指定或使用的参数(例如，数据库代码页、整理顺序和 DB2 UDB 发行版级别)
- 指示数据库当前状态的参数(例如，备份暂挂标志、数据库一致性标志和前滚暂挂标志)
- 定义数据库操作可以使用的系统资源的数量的参数(例如，缓冲池大小、数据库记录和排序内存大小)。

不应手工更改配置文件中的参数。只应使用受支持的接口。

性能提示: 许多配置参数都带有缺省值，但是为达到数据库的最优性能，可能需要更新它们。

对于多分区: 当有一个分布在多分区上的数据库时，该配置文件在所有数据库分区上应是相同的。一致性是必需的，因为 SQL 编译器根据本地节点配置文件中的信息来编译分布的 SQL 语句，并创建一个存取方案以满足 SQL 语句的需要。根据预编译该语句的数据库分区，维护数据库分区上的不同配置文件可能产生不同的存取方案。使用 **db2_all** 来保持所有数据库分区上的配置文件同步。

相关概念:

- 第 317 页的『在分区数据库环境中发出命令』

相关任务:

- 『用配置参数配置 DB2』（《*管理指南: 性能*》）

快速通信管理程序（FCM）通信

在分区数据库环境中，数据库分区之间的大多数通信都是由“快速通信管理程序”（FCM）来处理。要在一个数据库分区上启用 FCM 并允许与其它数据库分区通信，必须在 etc 目录的 services 文件创建一个服务条目，如下所示。FCM 使用指定的端口来通信。若已在同一个主机上定义了多分区，则必须定义一个端口范围，如下所示。

Windows® 注意事项

如果在 Windows 环境中使用 DB2® 通用数据库（DB2 UDB）企业服务器版，则会通过下列程序自动将 TCP/IP 端口范围添加至 services 文件:

- 安装程序，在创建实例或添加新节点时
- **db2icrt** 实用程序，在创建新实例时
- **db2ncrt** 实用程序，在机器上添加第一个节点时

服务条目的语法如下所示:

```
DB2_instance port/tcp #comment
```

DB2_*instance*

instance 的值是数据库管理器实例的名称。该名称的所有字符必须为小写。假定实例名为 db2puser，则应指定 DB2_db2puser。

port/**tcp**

要为该数据库分区保留的 TCP/IP 端口。

#*comment*

想要与该条目关联的任何注释。注释之前必须加 # 符号。

如果 etc 目录的 services 文件是共享的，则必须确保在该文件中分配的端口的数目大于或等于该实例中多个数据库分区的最大数目。当分配端口时，还要确保考虑了可以用作备份的任何处理器。

如果 etc 目录的 services 文件不是共享的，则注意事项基本相同，但它有一个附加注意事项: 必须确保为 DB2 UDB 实例定义的条目在 etc 目录的所有 services 文件中都是相同的（而不适用于分区数据库的条目不必相同）。

若在一个实例中的相同主机上有多个数据库分区，则必须定义多个端口以供 FCM 使用。为此，在 etc 目录的 services 文件中包括两行，以指示正分配的端口的范围。第一行指定第一个端口，而第二行指示端口块的结束。在下列示例中，为实例 sales 分配了五个端口。这意味着该实例中不会有处理器具有多于五个的数据库分区。例如,

```
DB2_sales         9000/tcp
   DB2_sales_END     9004/tcp
```

注: 只能用大写字母来指定 END。还必须确保包括了两个下划线（_）字符。

相关概念:

- 『分区和处理器环境』（*《管理指南: 计划》*）

第 2 章 创建和使用 DB2 管理服务器（DAS）

DB2 管理服务器（DAS）用于辅助 DB2 服务器任务。

DB2 管理服务器

DB2® 管理服务器（DAS）是一种控制点，仅用于辅助 DB2 Universal Database™（DB2 通用数据库）DB2 UDB 服务器上的任务。如果要使用提供的工具（如“配置助手”、“控制中心”或“开发中心”），则必须有一个正在运行的 DAS。当执行下列管理任务时，DAS 会辅助“控制中心”和“配置助手”。

- 启用 DB2 UDB 服务器的远程管理。
- 为作业管理提供设施，包括安排 DB2 UDB 和操作系统命令脚本运行的能力。这些命令脚本是用户定义的。
- 使用“任务中心”来对 DAS 的远程或本地作业定义作业的安排、查看已完成作业的结果以及执行其它管理任务。
- 与 DB2 UDB 发现实用程序一起提供一种查找关于 DB2 UDB 实例、数据库及其它 DB2 管理服务器配置的信息的方法。“配置助手”和“控制中心”使用此信息来简化和自动执行客户机与 DB2 UDB 数据库的连接配置。

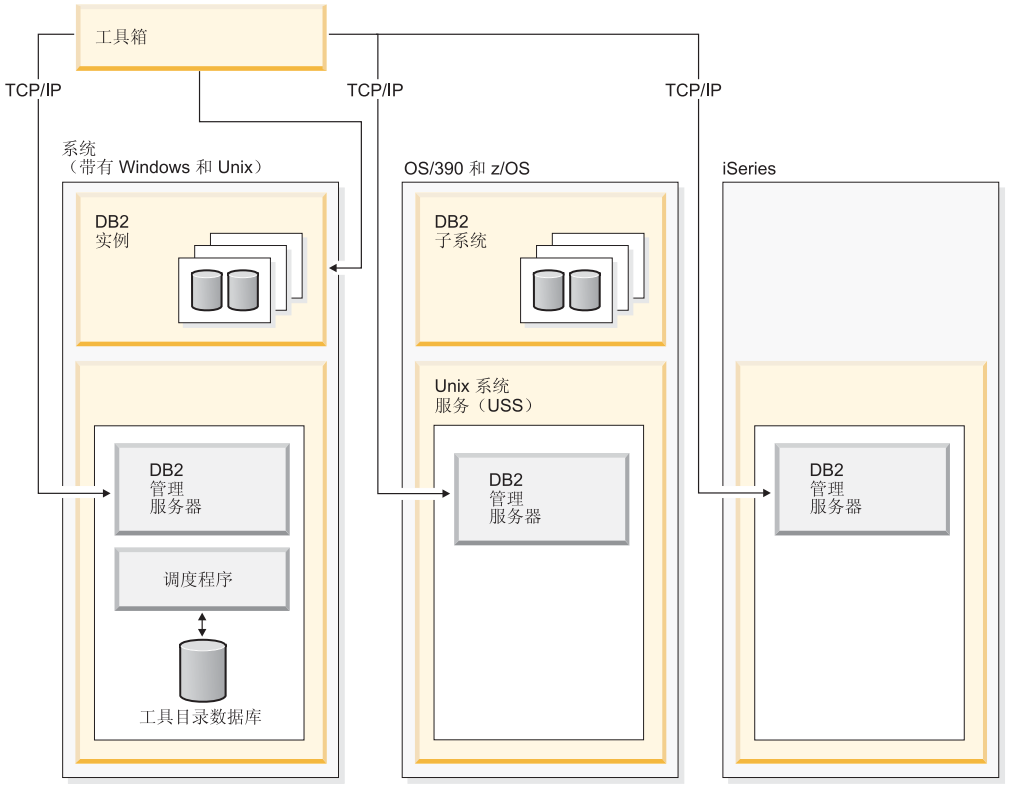


图 1. 在何处使用 DAS

在一台机器上只能有一个 DAS。在安装期间，将 DAS 配置为在引导操作系统时启动。

使用 DAS 在服务器系统和主机系统上代表来自“控制中心”、“配置助手”或任何其它可用工具的客户机请求执行远程任务。

DAS 在所有受支持的 Windows® 和 UNIX® 平台以及 zSeries®（仅适用于 OS/390 和 z/OS™）平台上可用。zSeries 上的 DAS 用于在管理任务中支持“控制中心”、“开发中心”和“复制中心”。

将 zSeries（仅适用于 OS/390 和 z/OS）上的 DB2 管理服务器作为 DB2 UDB “DB2 管理客户机”功能部件的一部分打包和交付。需要 DAS 的产品（如“控制中心”、“复制中心”和“开发中心”）需要安装 DAS 功能。有关操作系统上的 DAS 可用性的信息，与 IBM® 代表联系。

Windows 和 UNIX 上的 DAS 包括一个调度程序，以运行使用“任务中心”定义的任务（如 DB2 UDB 和操作系统命令脚本）。任务信息（例如，要运行的命令，与任务相关联的安排、通知和完成操作）以及运行结果存储在DB2 UDB 数据库中称为“工具目录”的一组表和视图中。“工具目录”是作为安装的一部分创建的。还可以通过“控制中心”或通过 CLP 使用 **CREATE TOOLS CATALOG** 命令创建和激活它。

虽然 zSeries（仅适用于 OS/390 和 z/OS）上未提供调度程序，但您可以使用“控制中心”中提供的“构建 JCL”和“创建 JCL”功能来生成 JCL（它保存在要使用系统调度程序运行的分区数据集中）。

相关概念:

- 第 47 页的『Windows 上的 DAS 安全性注意事项』
- 第 51 页的『企业服务器版（ESE）系统上的 DAS 配置』
- 第 52 页的『管理服务器、实例和数据库的发现』
- 第 55 页的『DB2 管理服务器首次故障数据捕获』

相关任务:

- 第 39 页的『创建 DB2 管理服务器』
- 第 39 页的『启动和停止 DAS』
- 第 40 页的『列示 DAS』
- 第 41 页的『配置 DAS』
- 第 47 页的『在 UNIX 上更新 DAS』
- 第 48 页的『除去 DAS』
- 第 49 页的『对企业服务器版（ESE）系统设置 DAS』
- 第 53 页的『隐藏服务器实例和数据库以免被发现』
- 第 53 页的『设置 discovery 参数』
- 第 54 页的『设置 DAS 以使用“配置助手”和“控制中心”』
- 第 55 页的『对发现更新 DAS 配置』
- 第 41 页的『工具目录数据库和 DAS 调度程序设置和配置』
- 第 45 页的『通知和联系人列表设置及配置』
- 第 46 页的『DAS Java 虚拟机设置』

创建 DB2 管理服务器

DB2 管理服务器为 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）工具（例如“控制中心”和“配置助手”）提供支持服务。

先决条件:

要创建 DAS，必须对 UNIX 平台具有 root 用户权限或使用具有创建服务的正确权限的帐户。

在 Windows 上，如果要标识特定用户，创建具有本地“管理员”权限的用户。输入 **db2admin create**。如果期望使用特定用户帐户，在发出 **db2admin create** 时必须使用“/USER:”和“/PASSWORD:”。

过程:

通常，DB2 UDB 安装期间，安装程序会在实例拥有的机器上创建一个 DAS。但是，若安装程序不能创建 DAS，可手工创建它。

当安装过程与 DAS 相关时，作为在安装过程期间发生的情况的一个概述，考虑下列事项:

- 在 Windows 平台上:

  使用具有创建服务的正确权限的帐户登录想要创建 DAS 的机器。

  当创建 DAS 时，可以选择是否提供用户帐户名和用户密码。若用户帐户名和密码有效，它们将标识该 DAS 的所有者。不要使用为 DAS 创建的用户标识或帐户名作为“用户帐户”。将该帐户名的密码设置为“密码永远不到期”。在创建了 DAS 之后，就可以使用 **db2admin setid** 命令提供一个用户帐户名和用户密码，来建立或修改它的所有权。
- 在 UNIX 平台上:
  1. 确保您具有 root 用户权限。
  2. 在命令提示符处，从 DB2 UDB 安装路径中的 `instance` 子目录发出以下命令:

     ```
     dascrt -u <DASUser>
     ```

     `<DASUser>` 是为 DB2 UDB 创建用户和组时创建的 DAS 用户的用户名。
     - 在 AIX 上:

       ```
       /usr/opt/db2_08_01/instance/
       dascrt -u <DASUser>
       ```
     - 在 HP-UX、Solaris Operating Environment 或 Linux 上:

       ```
       /opt/IBM/db2/V8.1/instance/
       dascrt -u <DASUser>
       ```

相关参考:

- 『db2admin - DB2 Administration Server Command』（*Command Reference*）

启动和停止 DAS

过程:

在 Windows 上，要手工启动或停止 DAS，必须首先使用属于“管理员”、“服务器操作员”或“高级用户”组的帐号或用户标识登录机器。在 Unix 上，要手工启动或停止 DAS，帐号或用户标识必须属于 *dasadm_group*。*dasadm_group* 是在 DAS 配置参数中指定的。

在 Windows 上，要启动或停止 DAS，使用 **db2admin start** 或 **db2admin stop** 命令。

当在任何 UNIX 操作系统的 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）上工作时，必须执行以下操作:

- 要启动 DAS:
  1. 作为 DAS 所有者登录。
  2. 使用下列其中一个命令运行启动脚本:

     ```
     . DASHOME/das/dasprofile     （对于 Bourne 或 Korn shell 程序）
       source DASHOME/das/dascshrc （对于 C shell）
     ```

     其中 DASHOME 是 DB2 管理服务器的主目录。
  3. 要启动 DAS，使用 **db2admin** 命令:

     ```
     db2admin start
     ```

  注: 在每次系统重新引导后自动启动 DAS。可使用 **dasauto** 命令来改变 DAS 的缺省启动行为。
- 要停止 DAS:
  1. 使用 *dasadm_group* 中的帐户或用户标识登录。
  2. 使用 **db2admin** 命令停止 DAS，如下所示:

     ```
     db2admin stop
     ```

注: 对于在 UNIX 下的这两种情况，使用这些命令的人必须先使用 DAS 所有者的授权标识登录。用户必须属于 *dasadm_group* 才能发出 **db2admin start** 或 **db2admin stop** 命令。

相关参考:

- 『db2admin - DB2 Administration Server Command』（*Command Reference*）
- 『dasadm_group - DAS 管理权限组名配置参数』（《*管理指南: 性能*》）

列示 DAS

过程:

要获取机器上的 DAS 的名称，输入:

```
db2admin
```

还可使用此命令来启动或停止 DAS、创建新用户和密码、删除 DAS 以及建立或修改与该 DAS 相关联的用户帐户。

相关参考:

- 『db2admin - DB2 Administration Server Command』（*Command Reference*）

配置 DAS

过程:

要查看与 DAS 相关的 DB2 管理服务器配置参数的当前值，输入:

```
db2 get admin cfg
```

它显示在安装该产品期间指定为缺省值的当前值，或在先前更新配置参数期间指定的那些值。

为了使用“命令行处理器”（CLP）和 UPDATE ADMIN CONFIG 来更新 DAS 配置文件，必须从与 DAS 具有相同安装级别的实例使用 CLP。要更新 DAS 配置文件中的各个条目，输入:

```
db2 update admin cfg using ...
```

要将配置参数复位为建议的缺省值，输入:

```
db2 reset admin cfg
```

在某些情况下，对 DAS 配置文件的更改仅在将更改装入内存后才生效（即在执行 **db2admin stop** 之后，再执行 **db2admin start** 时生效；或对于 Windows 平台，在停止并启动该服务时生效）。在其它情况下，配置参数是可联机配置的（即不必重新启动 DAS 就可以使更改生效）。

相关任务:

- 『用配置参数配置 DB2』（*《管理指南: 性能》*）

相关参考:

- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）

工具目录数据库和 DAS 调度程序设置和配置

工具目录数据库包含“任务中心”和“控制中心”创建的任务信息。这些任务是由 DB2 管理服务器的调度程序运行的。调度程序和工具目录数据库始终一起工作；缺一不可。调度程序是 DB2 管理服务器的一个特定段，该服务器充当代理进程以读取工具目录数据库，并在其各自的时间里运行任务。

先决条件:

必须安装 DB2 管理服务器。

过程:

目标是设置和配置工具目录数据库和 DAS 调度程序。

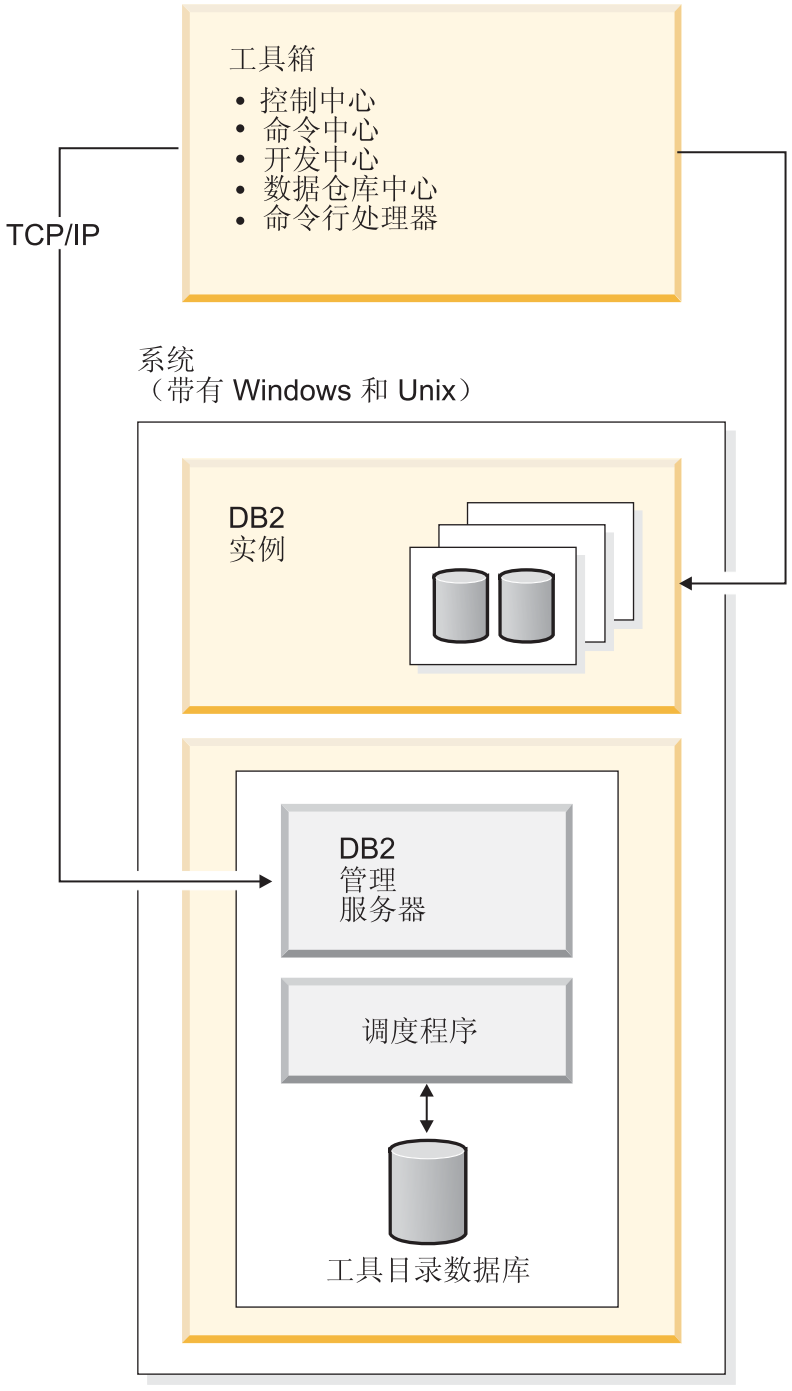


图 2. DAS 如何与 DB2 UDB 其它部分发生关系

DB2 管理服务器配置进程告诉调度程序工具目录数据库的位置以及是否应该启用调度程序。缺省情况下，当创建工具目录数据库时，更新其相应 DAS 配置。即，配置调度程序并准备使用新的工具目录；此时无需重新启动 DAS。

可以通过调度程序系统在本地或远程服务器上创建工具目录数据库。如果在远程服务器上创建工具目录，则必须在调度程序工具目录数据库实例（TOOLSCAT_INST）上

对工具目录进行编目。另外，必须使用命令 **db2admin setschedid** 设置调度程序用户标识，以便调度程序可以连接并认证远程目录。可在 *Command Reference* 中找到 **db2admin** 命令的完整语法。

DAS 调度程序需要 Java 虚拟机（JVM）来存取工具目录信息。JVM 信息是使用 DAS 的 jdk_path DB2 管理服务器配置参数指定的。

如果您正在对位于其中一个既支持 32 位实例又支持 64 位实例的平台（AIX、Sun 和 HP-UX）上的 64 位实例创建工具目录，则 jdk_64_path 配置参数是必需的。

“控制中心”和“任务中心”直接从客户机存取工具目录数据库。因此，在“控制中心”使用它之前，需要在客户机上对工具目录数据库进行编目。“控制中心”提供自动检索有关工具目录数据库的信息以及在本地节点目录或数据库目录中创建必需的目录条目的方法。此自动编目支持的唯一通信协议是 TCP/IP。

其中一个 DAS 配置参数称为 *exec_exp_task*。此参数指定调度程序是否执行过去已安排但尚未运行的任务。调度程序启动时仅检测到期的任务。

例如，如果已将作业安排为每个星期六运行，而在星期五关闭了调度程序，在星期一重新启动调度程序，安排为星期六运行的作业就是在过去安排的作业。如果将 *exec_exp_task* 设置为“是”，则在重新启动调度程序时运行星期六作业。

调度程序需要的其它 DAS 配置参数包括标识工具目录数据库和要用于通知的“简单电子邮件传输协议”（SMTP）服务器。

下列示例说明如何使用这些参数:

- 示例 Windows 服务器设置。
  1. 工具目录数据库可以是您想要的任何名称。在本示例中，工具目录数据库称为“CCMD”，且是在服务器 Host1（tcp/ip hostname Host1）上的 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）实例下创建的。使用模式名在给定数据库内唯一标识工具目录。出于本示例的目的，假定模式名为“CCADMIN”。
  2. 使用下列命令设置名为“DB2”的实例以使用端口号 50000 进行 TCP/IP 通信:

     ```
     db2set -i DB2 DB2COMM=TCPIP
     db2 update dbm cfg using svcename db2cDB2
     db2stop
      db2start
     ```

  3. db2cDB2 服务名称是在 %SystemRoot%\system32\drivers\etc\services 中定义的。即，在 services 中将找到以下行:

     ```
     db2cDB2     50000/tcp     #connection port for the DB2 instance DB2
     ```

  4. 工具目录是使用 CREATE TOOLS CATALOG 命令创建的。这将创建工具目录表和视图，并且其模式名与指定数据库中的目录名相对应。自动更新 DB2 管理服务器配置参数，并启用和启动调度程序。
  5. 假定用于电子邮件通知的 SMTP 服务器在机器 Host2（tcp/ip hostname Host2）上。然后，使用下列命令对 DB2 管理服务器指定此信息:

     ```
     db2 update admin cfg using smtp_server Host2
     ```

     这可以在安装过程中完成。如果以后才执行此操作，则需要通过 DB2 UDB V8 CLP 命令手工指定给 DAS，如上所示。

6. Windows 上的 IBM Software Development Kit (SDK) for Java 安装在 %DB2PATH%\java\jdk 下面。应已将它指定给 DAS。可以使用以下命令来验证并设置（必要时）它:

```
db2 update admin cfg using jdk_path c:\SQLLIB\java\jdk
```

这假定 DB2 UDB 已安装在 C:\SQLLIB 下面。

注: 如果将要使用 db2admin create 创建 DAS，则确保使用 /USER 和 /PASSWORD 选项。USER 帐户由调度程序进程使用。没有该帐户，将不能正确启动调度程序。USER 帐户应在工具目录实例中具有SYSADM 权限。

如果将要使用 db2admin create 创建 DAS，并且此时未指定 /USER 和 /PASSWORD 选项，则可在稍后时间更新 USER 信息。通过运行以下命令在 DAS 上完成此更新:

```
db2admin stop   db2admin setid <user account ID> <password>
db2admin start
```

- 示例 Windows 客户机设置。
  1. 假定“控制中心”正在客户机 C1 （tcp/ip hostname C1）上运行。
  2. 在本地节点目录中，使用“配置助手”或“控制中心”或使用以下命令将 DAS 作为管理服务器节点进行编目:

```
db2 catalog admin tcpip node Host1 remote Host1 system Host1 ostype NT
```

  3. 如果启动了“任务中心”且选择了系统 Host1，则“任务中心”试图在本地目录中查找工具目录数据库。（可使用“控制中心”替代“任务中心”。）如果找不到，则它使用下列命令尝试对节点和数据库进行编目:

```
db2 catalog tcpip node <unique-node name>
  remote Host1 server 50000
  remote_instance DB2 system Host1 ostype NT
db2 catalog db CCMD as <unique-db alias> at node <unique-node name>
```

如果自动编目不成功，可使用“配置助手”或“控制中心”来对数据库进行编目。然后，将由“任务中心”识别和使用数据库。

- 示例 AIX 服务器设置。
  1. 工具目录数据库可以是您想要的任何名称。在本示例中，工具目录数据库称为“CCMD”且是在服务器 Host1（tcp/ip hostname Host1）上的 db2inst1 实例下创建的。使用模式名在给定数据库内唯一标识工具目录。出于本示例的目的，假定模式名为“CCADMIN”。
  2. 通过使用下列命令设置实例 db2inst1 以使用端口号 50000 进行 TCP/IP 通信:

```
db2set -i DB2 DB2COMM=TCPIP
db2 update dbm cfg using svcename xdb2inst
db2stop
 db2start
```

  3. xdb2inst 服务名称是在 /etc/services 中定义的。即，在 services 中将找到以下行:

```
xdb2inst1     50000/tcp     #connection port for the DB2 instance db2inst1
```

  4. 工具目录是使用 CREATE TOOLS CATALOG 命令创建的。这将创建工具目录表和视图，并且其模式名与指定数据库中的目录名相对应。自动更新 DB2 管理服务器配置参数，并启用和启动调度程序。

5. 假定用于电子邮件通知的 SMTP 服务器在机器 Host2（tcp/ip hostname Host2）上。然后，使用下列命令对 DB2 管理服务器指定此信息:

   ```
   db2 update admin cfg using smtp_server Host2
   ```

   这可以在安装过程中完成。如果以后才执行此操作，则需要通过 DB2 UDB V8 CLP 命令手工指定给 DAS，如上所示。

6. AIX 上的 IBM Software Developer's Kit for Java (SDK) V1.3.1 安装在 /sqllib/java/jdk 下面。应已将它指定给 DAS。可以使用以下命令来验证并设置（必要时）它:

   ```
   db2 update admin cfg using jdk_path /sqllib/java/jdk
   ```

- 示例 AIX 客户机设置。
  1. 假定“控制中心”正在客户机 C1 （tcp/ip hostname C1）上运行。
  2. 在本地节点目录中，使用“配置助手”或“控制中心”或使用以下命令将 DAS 作为管理服务器节点进行编目:

     ```
     db2 catalog admin tcpip node Host1 remote Host1 system Host1
       ostype AIX
     ```

  3. 如果启动了“任务中心”且选择了系统 Host1，则“任务中心”试图在本地目录中查找工具目录数据库。（可使用“控制中心”替代“任务中心”。）如果找不到，则它使用下列命令尝试对节点和数据库进行编目:

     ```
     db2 catalog tcpip node <unique-node name>
       remote Host1 server 50000
       remote_instance DB2 system Host1 ostype AIX
     db2 catalog db CCMD as <unique-db alias> at node <unique-node name>
     ```

     如果自动编目不成功，可使用“配置助手”或“控制中心”来对数据库进行编目。然后，将由“任务中心”识别和使用数据库。

相关参考:

- 『svcename - TCP/IP 服务名称配置参数』（*《管理指南: 性能》*）
- 『sched_enable - 调度程序方式配置参数』（*《管理指南: 性能》*）
- 『toolscat_inst - 工具目录数据库实例配置参数』（*《管理指南: 性能》*）
- 『toolscat_db - 工具目录数据库配置参数』（*《管理指南: 性能》*）
- 『toolscat_schema - 工具目录数据库模式配置参数』（*《管理指南: 性能》*）
- 『smtp_server - SMTP 服务器配置参数』（*《管理指南: 性能》*）
- 『jdk_path - Software Developer's Kit for Java 安装路径 DAS 配置参数』（*《管理指南: 性能》*）
- 『exec_exp_task - 执行已到期的任务配置参数』（*《管理指南: 性能》*）

通知和联系人列表设置及配置

来自 DB2 管理服务器（DAS）的电子邮件和寻呼机通知可能是本地的或远程的。需要联系人列表以确保将通知发送至正确的主机名。

过程:

调度程序或健康监视器使用两个 DAS 配置参数来启用通知。

DAS 配置参数 *smtp_server* 用来标识“简单电子邮件传输协议”（SMTP）服务器，调度程序使用该服务器作为任务执行完成操作的一部分（象通过“任务中心”定义的那样）来发送电子邮件和寻呼机信息，或者，健康监视器使用该服务器来通过电子邮件或寻呼机发送报警通知。

DAS 配置参数 *contact_host* 指定调度程序和健康监视器用于通知的联系人信息的存储位置。将位置定义为 DB2 管理服务器的 TCP/IP 主机名。允许 *contact_host* 位于远程 DAS 上支持在多个 DB2 管理服务器间共享联系人列表。应对分区数据库环境设置此项以确保将公共联系人列表用于所有分区。联系人列表存储在 DAS 目录下的文本文件中。如果未指定 *contact_host*，则 DAS 假定联系人信息在本地。

相关参考:

- 『smtp_server - SMTP 服务器配置参数』（*《管理指南: 性能》*）
- 『contact_host - 联系人列表的位置配置参数』（*《管理指南: 性能》*）

DAS Java 虚拟机设置

过程:

jdk_path 配置参数指定安装 IBM Software Developer's Kit (SDK) for Java 的目录，IBM Software Developer's Kit (SDK) for Java 用于运行 DB2 管理服务器函数。Java 解释器使用的环境变量是根据此参数的值计算出来的。

调度程序需要 Java 虚拟机（JVM）才能使用工具目录数据库。必须进行此设置才能成功启动调度程序。

使用 UNIX 平台时，此参数没有缺省值。应在安装 IBM Software Developer's Kit (SDK) for Java 时指定此参数的值。

Windows 上的 IBM Software Developer's Kit (SDK) for Java 安装在 `%DB2PATH%\java\jdk`（在 Windows 平台上，这是此参数的缺省值）下面。应已将它指定给 DAS。可使用以下命令验证 *jdk_path* 的值:

```
db2 get admin cfg
```

此命令显示 DB2 管理服务器配置文件的值，其中 *jdk_path* 是一个配置参数。必要时，可使用以下命令设置该参数:

```
db2 update admin cfg using jdk_path 'C:\Program Files\IBM\SQLLIB'
```

这假定 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）安装在 `'C:\Program Files\IBM\SQLLIB'` 下面。

AIX 上的 IBM Software Developer's Kit (SDK) for Java 安装在 `/usr/java130` 下面。必要时，可使用以下命令设置该参数:

```
db2 update admin cfg using jdk_path /usr/java130
```

注: 如果正在对其中一个既支持 32 位实例又支持 64 位实例的平台（AIX、Sun 或 HP-UX）上的 64 位实例创建或使用工具目录，则使用 `jdk_64_path` 配置参数，而不使用 `jdk_path` 参数。此配置参数指定安装 64 位版本 IBM Software Develop's Kit (SDK) for Java 的目录。

相关参考:

- 『GET ADMIN CONFIGURATION Command』（*Command Reference*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）
- 『jdk_path - Software Developer's Kit for Java 安装路径 DAS 配置参数』（《*管理指南: 性能*》）

Windows 上的 DAS 安全性注意事项

可能需要更改 DAS 服务在 Windows® 上运行时使用的用户标识。

创建 DAS 后，可使用 **db2admin** 命令设置或更改登录帐户，如下所示:

```
db2admin setid <username> <password>
```

其中 `<username>` 和 `<password>` 是具有本地“管理员”权限的帐户的用户名和密码。在运行此命令之前，必须使用具有本地“管理员”权限的帐户或用户标识登录机器。

注: 记住密码是区分大小写的。允许大小写混合的密码，这意味着密码的大小写非常重要。

注: 在 Windows 上，不应使用**控制面板**中的**服务**实用程序来更改 DAS 的登录帐户，因为不会为该登录帐户设置某些必需的存取权。始终使用 **db2admin** 命令来设置或更改 DB2® 管理服务器（DAS）的登录帐户。

相关参考:

- 『db2admin - DB2 Administration Server Command』（*Command Reference*）

在 UNIX 上更新 DAS

过程:

在 UNIX 操作系统上，如果通过安装“程序临时性修订”（PTF）或代码补丁来更新 DB2，则应更新每个 DB2 管理服务器（DAS）和实例。要更新 DAS，使用 **dasupdt** 命令，可在特定于已安装的 DB2 版本和发行版的子目录下的 `instance` 子目录中找到该命令。

必须首先使用超级用户权限（通常作为“root”用户）登录机器。

按如下所示使用该命令:

```
dasupdt
```

此命令还有一些可选的参数:

- –h 或 –?

  显示此命令的帮助菜单。

- –d

  设置调试方式，用于问题分析。

- –D

将 DAS 从一个路径上的较高代码级别移至安装在另一个路径上的较低代码级别。

注: 在 Windows 上，更新 DAS 是安装过程的一部分。不需要任何用户操作。

示例:

DAS 正在版本 8 安装路径中运行版本 8.1.2 代码。如果修订包 3 安装在版本 8 的安装路径中，则从版本 8 安装路径中调用的以下命令将把 DAS 更新为修订包 3:

```
dasupdt
```

DAS 正在备用安装路径中运行版本 8.1.2 代码。如果修订包 1 安装在另一个备用安装路径中，则从修订包 1 备用安装路径中调用的以下命令将把 DAS 更新为修订包 1，从修订包 1 备用安装路径中运行:

```
dasupdt -D
```

相关概念:

- 第 37 页的『DB2 管理服务器』
- 第 47 页的『Windows 上的 DAS 安全性注意事项』

除去 DAS

过程:

要除去 DAS:

- 在 Windows 操作系统上:
  1. 使用具有除去服务的正确权限的帐户或用户标识登录机器。
  2. 使用 **db2admin stop** 来停止 DAS。
  3. 备份（若需要的话） `sqllib` 子目录下 `db2das00` 子目录中的所有文件。

     注: 本示例假设 `db2das00` 是要除去的 DAS 的名称。如果用户创建了名为 DB2DAS00 的 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）实例，则可能具有名称不是 DB2DAS00 的 DAS。在此情况下，会将 DAS 命名为 DB2DAS01（或者如果已采用该名称，则使用名称 DB2DAS02，依此类推）。应查找具有 “DB2DAS” 前缀的服务，以在可能存在的几个 DAS 的列表中标识特定 DAS。您可以不带任何选项使用 **db2admin** 命令来列示所有 DAS。
  4. 使用 **db2admin drop** 来删除 DAS。
- 在 UNIX 操作系统上:
  1. 作为具有 DASADM 权限的用户登录。
  2. 使用下列其中一个命令运行启动脚本:

     ```
     . DASHOME/das/dasprofile     （对于 Bourne 或 Korn shell 程序）
       source DASHOME/das/dascshrc （对于 C shell）
     ```

     其中 `DASHOME` 是 DAS 所有者的主目录。
  3. 使用 **db2admin** 命令停止 DAS，如下所示:

     ```
     db2admin stop
     ```
  4. 备份（如果需要的话）DAS 的主目录下 `das` 子目录中的所有文件。

5. 注销。
6. 作为 root 用户登录，并使用 **dasdrop** 命令除去 DAS，如下所示:

   ```
   dasdrop
   ```

   可在特定于已安装 DB2 UDB 版本和发行版的子目录下的 instance 子目录中找到 **dasdrop** 命令。

   注: **dasdrop** 命令除去 DB2 管理服务器（DAS）的主目录下的 das 子目录。

相关参考:

- 『db2admin - DB2 Administration Server Command』（*Command Reference*）
- 『dasdrop - Remove a DB2 Administration Server Command』（*Command Reference*）

对企业服务器版（ESE）系统设置 DAS

过程:

下列信息说明了配置 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）ESE 服务器（Linux、Solaris Operating Environment、Windows NT、Windows 2000、Windows Server 2003、HP-UX 和 AIX）以便使用“控制中心”进行远程管理所需的步骤。

在安装期间，安装程序会在实例拥有的机器上创建一个 DAS。您必须在其它机器上创建附加 DAS 才能允许“控制中心”或“配置助手”存取其它协调程序节点。然后可将工作（如管理协调程序节点）的开销分摊到实例中的多分区上。安装程序将在它在其上运行的所有节点上创建 DAS。仅当您不使用 **db2setup** 时，您才会需要手工执行此操作。

此处给出的指示只适用于分区 ESE 环境。如果您只在单一分区 ESE 系统上运行，则给出的指示不适用于您的环境。

要分发协调程序功能:

1. 在该分区数据库系统中所选的附加机器上创建一个新的 DAS。
2. 在“控制中心”或“配置助手”中将每个 DAS 作为单独的系统进行编目。
3. 在每个新的系统下对相同的实例进行编目，每次指定相同的机器名对该 DAS 进行编目。

配置具有两种用途: DB2 管理服务器（DAS）所需的配置和建议用于目标受管 DB2 UDB 实例的配置。在下面的三节中，有一节专门阐述这两个配置主题。在每个配置主题之前，都有一段描述假设的环境。

环境示例

产品 / 版本:

DB2 UDB ESE V8.1

安装路径:

install_path

TCP services 文件:

services

DB2 UDB 实例:

名称: db2inst

所有者标识:
db2inst

实例路径:
instance_path

节点: 3 个节点，db2nodes.cfg:

- 0 hostA 0 hostAswitch
- 1 hostA 1 hostAswitch
- 2 hostB 0 hostBswitch

DB 名称:
db2instDB

DAS:

名称: db2as00

所有者 / 用户标识:
db2as

实例路径:
das_path

安装 / 运行主机:
hostA

节点间通信端口:
16000 （hostA 和 hostB 的未使用端口）

注: 请用特定于站点的值替换上面的字段。例如，下表包含某些受支持的样本 ESE 平台的示例路径名:

表 2. 受支持的 ESE 平台的示例路径名

路径	DB2 UDB ESE AIX 版	DB2 UDB ESE Solaris Operating Environment 版	DB2 UDB ESE Windows 版
install_path	/usr/opt/<v_r_ID>	/opt/IBM/db2/<v_r_ID>	C:\sqllib
instance_path	/home/db2inst/sqllib	/home/db2inst/sqllib	C:\profiles\db2inst
das_path	/home/db2as/das	/home/db2as/das	C:\profiles\db2as
tcp_services_file	/etc/services	/etc/services	C:\winnt\system32 \drivers\etc\services

在表中，<v_r_ID> 是特定于平台的版本和发行版标识。例如，在版本 8 的 DB2 UDB ESE AIX 版中，<v_r_ID> 为 db2_08_01。

安装 DB2 UDB ESE 时，安装程序会在实例拥有的机器上创建 DAS。数据库分区服务器驻留在 DAS 所在的机器上，并是该实例的连接点。即，此数据库分区服务器是从“控制中心”或“配置助手”发出至该实例的请求的协调程序节点。

如果在每台物理机器上安装了 DAS，则每台物理机器都可充当协调程序节点。在“控制中心”或“配置助手”中，每台物理机器都都作单独的 DB2SYSTEM 出现。如果另一客户机使用另一系统来连接至分区数据库服务器，这将分发协调程序节点功能并帮助平衡入站连接。

相关概念:

- 第 37 页的『DB2 管理服务器』
- 第 51 页的『企业服务器版（ESE）系统上的 DAS 配置』

企业服务器版（ESE）系统上的 DAS 配置

DAS 是一个管理控制点，它代表一些工具执行特定的任务。每台物理机器最多只能有一个 DAS。对于由几台机器组成的 ESE 实例，所有机器都必须运行 DAS，以便“控制中心”可以管理该 ESE 实例。此 DAS（db2as）是由“控制中心”导航树中作为目标 DB2® 通用数据库（DB2 UDB）实例（db2inst）的父项出现的系统表示。

例如，db2inst 由分布在两个物理机器或主机上的三个节点组成。通过在 hostA 和 hostB 上运行 **db2as**，可以满足最低需求。

注:

1. 在 hostA 上存在的分区数对可在该主机上运行的 DAS 数没有任何影响。不管 hostA 的多逻辑节点（MLN）的配置如何，您只能在该主机上运行 DAS 的一个副本。
2. 每台机器或物理节点上都需要一个 DAS，必须使用 **dascrt** 命令分别创建 DAS。每台机器或物理节点上的 DAS 必须正在运行以便“任务中心”和“控制中心”可正常工作。db2as 标识必须存在于 hostA 和 hostB 上。db2as 标识的主目录不需要在这两个系统之间交叉安装。或者，可用不同的用户标识在 hostA 和 hostB 上创建 DAS。

在 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）企业服务器版 Windows® 版上，如果您正在使用“配置助手”或“控制中心”来自动配置与 DB2 UDB 服务器的连接，则与 DAS 位于相同机器上的数据库分区服务器将用作协调程序节点。这表示从客户至数据库的所有物理连接在路由至其它分区服务器之前将被引导至协调程序节点。

在 DB2 通用数据库（DB2 UDB）企业服务器版 Windows 版上，在其它机器上创建附加 DB2 管理服务器允许“配置助手”或“控制中心”使用 DB2 发现将其它系统配置为协调程序节点。

当使用 DB2 通用数据库（DB2 UDB）企业服务器版 Windows 版时，“DB2 UDB 远程命令服务”（**db2rcmd.exe**）会自动处理节点间管理通信。

“控制中心”使用 TCP 服务端口 523 来与 DAS 通信。此端口保留给 DB2 UDB 专用。因此，不需要将新条目插入到 TCP services 文件中。

相关任务:

- 第 39 页的『创建 DB2 管理服务器』

相关参考:

- 『db2admin - DB2 Administration Server Command』（*Command Reference*）

管理服务器、实例和数据库的发现

要配置与远程机器的连接，有两种方法：使用内置于“配置助手”中的发现服务；或者，使用现有目录服务，如“轻量级目录访问协议”（LDAP）。

“发现”服务是与“配置助手”和 DB2® 管理服务器集成在一起的。要配置与远程机器的连接，用户将登录客户机并运行“配置助手”（CA）。CA 将广播信号发送至网络上的所有机器。安装有 DAS 且已对发现作了配置的任何机器将通过送回包含所有实例以及有关该机器的数据库信息的程序包来响应来自 CA 的广播信号。然后，CA 使用此程序包中的信息来配置客户机连接。通过使用发现方法，可在本地数据库和节点目录中自动生成远程服务器的目录信息。

发现方法要求您登录每台客户机并运行 CA。如果您使用的环境中有大量客户机，这会很困难并且要花很多时间。在这种情况下，另一种方法是使用目录服务（如 LDAP）。

“已知发现”允许在系统上查找客户机知道的实例和数据库，并添加新系统，以便可找到它们的实例和数据库。“搜索发现”提供“已知发现”的所有设施，并添加选项以允许搜索本地网络，以查找其它 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）服务器。

要让系统支持“已知发现”，在 DAS 配置文件中将 *discover* 参数设置为 KNOWN。要让系统支持“已知发现”和“搜索发现”，则在 DAS 配置文件中将 *discover* 参数设置为 SEARCH（这是缺省情况）。要阻止某个系统及其所有实例和数据库的发现，将此参数设置为 DISABLE。在 DAS 配置文件中将 *discover* 参数设置为 DISABLE 将阻止系统的发现。

注： 由“搜索发现”返回给客户机的 TCP/IP 主机名与输入 **hostname** 命令时由 DB2 UDB 服务器系统返回的主机名相同。在该客户机上，此主机名所映射成的 IP 地址由客户机上配置的 TCP/IP 域名服务器（DNS）（若尚未配置 DNS）或客户机的 *hosts* 文件中的一个映射条目确定。如果在 DB2 UDB 服务器系统上配置了多个适配卡，必须确保在服务器上配置 TCP/IP 以返回正确的主机名，还必须确保 DNS 或本地客户机的 *hosts* 文件将主机名映射到期望的 IP 地址。

在客户机上，也可使用 *discover* 参数启用“发现”；但在这种情况下，要在客户机实例（或充当客户机的服务器）中设置 *discover* 参数，如下所示：

- **KNOWN**

  “配置助手”和“控制中心”使用 KNOWN 发现来检索与您的本地系统已知的系统相关联的实例和数据库信息。可使用这些工具中提供的**添加系统**功能来添加新系统。在将 *discover* 参数设置为 KNOWN 时，您将不能搜索网络。

- **SEARCH**

  启用“已知发现”的所有设施，并启用本地网络搜索。这表示所有搜索将限制在本地网络范围内。

  仅当选择了此选项时，“其它系统（搜索网络）”图标才会出现。这是缺省设置。

- **DISABLE**

  禁用“发现”。在这种情况下，在“添加数据库向导”中没有**搜索网络**选项。

注: *discover* 参数在所有客户机和服务器实例上缺省为 SEARCH。*discover* 参数在所有 DB2 管理服务器（DAS）上缺省设置为 SEARCH。

相关概念:

- 第 63 页的『“轻量级目录访问协议”（LDAP）目录服务』

相关任务:

- 第 53 页的『隐藏服务器实例和数据库以免被发现』
- 第 53 页的『设置 discovery 参数』

隐藏服务器实例和数据库以免被发现

过程:

可能在一个服务器系统上有多个实例，而在这些实例中又包含多个数据库。您可能想隐藏其中某些实例，以免“发现”进程发现。

要允许客户机发现一个系统上的服务器实例，在该系统上的每个服务器实例中将 *discover_inst* 数据库管理器配置参数设置为 ENABLE（这是缺省值）。将此参数设置为 DISABLE，可隐藏此实例及其数据库以免被“发现”发现。

要允许从客户机查找一个数据库，将 *discover_db* 数据库配置参数设置为 ENABLE（这是缺省值）。将此参数设置为 DISABLE，可隐藏该数据库以免被发现。

注: 如果想要发现实例，还应在 DAS 配置文件中将 discover 设置为 KNOWN 或 SEARCH。如果想要数据库被发现，则还必须在服务器实例中启用 *discover_inst* 参数。

相关参考:

- 『discover_inst - 发现服务器实例配置参数』（*《管理指南: 性能》*）
- 『discover_db - 发现数据库配置参数』（*《管理指南: 性能》*）

设置 discovery 参数

过程:

在服务器系统上的 DAS 配置文件及客户机上的数据库管理器配置文件中设置 *discover* 参数。使用“配置助手”或“控制中心”设置数据库管理器配置参数: *discover*、*discover_inst* 和 *discover_db*。按如下所示设置这两个参数:

- 在 DAS 上:

  在 DAS 配置文件中使用以下命令进程更新 *discover* 参数（作为示例）:

  ```
  update admin cfg using discover [ DISABLE | KNOWN |
        SEARCH ]
  ```

  DAS *discover* 配置参数是可联机配置的，这表示您不必停止并重新启动 DAS 就可让更改生效。

  注: “搜索发现”只能通过 TCP/IP 运行。

- 通过使用“配置助手”:

通过从命令行输入 **db2ca**（在所有平台上）或者从“开始”菜单（在 Windows 上）单击**开始 —> 程序 —> IBM DB2 —> 安装工具 —> 配置助手**来启动“配置助手”。

要使用“配置助手”来设置数据库管理器配置参数:

1. 单击**配置 —> DBM 配置**。
2. 单击想要修改的关键字。
3. 在“值”这一列中，单击想要修改的关键字的值，并单击**确定**。
4. 再次单击**确定**，将显示一条消息。单击**关闭**。

使用“控制中心”来设置 *discover_inst* 和 *discover_db* 参数。

还可以使用“配置助手”来更新配置参数。

相关参考:

- 『discover_inst - 发现服务器实例配置参数』（*《管理指南: 性能》*）
- 『discover_db - 发现数据库配置参数』（*《管理指南: 性能》*）
- 『UPDATE ADMIN CONFIGURATION Command』（*Command Reference*）
- 『discover - DAS 发现方式配置参数』（*《管理指南: 性能》*）

设置 DAS 以使用“配置助手”和“控制中心”

先决条件:

必须配置 **discover** 才能在网络上检索有关系统的信息。

限制:

DAS 必须驻留在每个物理分区上。在分区上创建 DAS 时，将 DB2SYSTEM 名配置为 TCP/IP 主机名，**discover** 设置缺省为 SEARCH。

过程:

“DB2 发现”是“配置助手”和“控制中心”使用的一个功能部件。配置此功能部件可能需要更新 DB2 管理服务器（DAS）配置和实例的数据库管理器配置，以确保“DB2 发现”检索正确的信息。

客户机从“配置助手”或“控制中心”发出发现请求时，启用了发现的每个 DAS 都会作出响应。在分区数据库环境中，每个物理分区将作为单独的 DB2SYSTEM 名响应。可管理的实际实例取决于该物理分区已知的实例。由于实例可跨越多分区，可通过不同的系统名潜在地管理同一实例。可使用此能力来帮助您平衡服务器实例上的负载。例如，如果实例“A”可通过系统“S1”和系统“S2”获取，则某些用户可使用“S1”对数据库进行编目，某些用户可使用“S2”对同一数据库进行编目。每个用户都可使用不同的协调程序数据库分区连接至服务器。

相关参考:

- 『db2ilist - List Instances Command』（*Command Reference*）

- 『db2ncrt - Add Database Partition Server to an Instance Command』（*Command Reference*）
- 『discover – DAS 发现方式配置参数』（《*管理指南: 性能*》）

对发现更新 DAS 配置

限制:

DAS 必须驻留在每个物理分区上。在分区上创建 DAS 时，将 DB2SYSTEM 名配置为 TCP/IP 主机名，*discover* 设置缺省为 SEARCH。

过程:

由“发现”检索到的系统名是 DB2 管理服务器（DAS）所在的系统。当建立连接时，发现将这些系统用作协调程序节点。

如果更新 DAS 配置，且想要能够从 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）系统列表中选择协调程序节点，则在每个 DB2 管理服务器的配置文件中设置 *discover*=SEARCH（这是缺省值）。

当分区数据库服务器环境中有多个 DAS 时，在“配置助手”或“控制中心”的界面上，同一实例可能出现在多个系统中；但每个系统使用不同的通信存取路径来存取实例。用户可选择不同的 DB2 UDB 系统作为通信的协调程序节点，因此重新分配工作负载。

相关参考:

- 『其它变量』（《*管理指南: 性能*》）

DB2 管理服务器首次故障数据捕获

首次故障数据捕获（FFDC）是一个普通术语，适用于发生错误时 DB2® 管理服务器自动捕获的诊断信息的集合。此信息减少了再现错误以获取诊断信息的需要。诊断信息包含在单个位置中。

DB2 管理服务器 FFDC 捕获的信息包括:

- 管理通知日志。

  当事件发生时，DB2 管理服务器将信息写入 DB2 管理服务器日志文件 db2dasdiag.log。
- 转储文件。

  对于某些错误情况，会将附加信息记录在以失败进程标识命名的外部二进制转储文件中。这些文件专供“DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）客户支持”使用。
- 陷阱文件。

  如果 DB2 管理服务器由于陷阱、分段违规或异常而不能继续处理，它就会生成陷阱文件。陷阱文件包含问题发生之前运行的最后步骤的函数流。

DB2 管理服务器首次故障数据捕获信息位置。

缺省情况下，DB2 管理服务器 FFDC 信息放在下列位置中:

- 在 Windows® 系统上:

  如果未设置 DB2INSTPROF 环境变量:

  ```
  db2path\DB2DAS00\dump
  ```

  其中 db2path 是 DB2PATH 环境变量中引用的路径，DB2DAS00 是 DAS 服务的名称。DAS 名称可通过不带任何自变量输入 **db2admin** 命令获取。

  如果设置了 DB2INSTPROF 环境变量:

  ```
  x:\db2instprof\DB2DAS00\dump
  ```

  其中 x: 是 DB2PATH 环境变量中引用的驱动器，db2instprof 是实例概要文件目录，而 DB2DAS00 是 DAS 服务的名称。

- 在基于 UNIX® 的系统上:

  ```
  $DASHOME/das/dump
  ```

  其中 $DASHOME 是 DAS 用户的主目录。

注: 应定期清理转储目录以防止它变得太大。

解释 DB2 管理服务器日志。

DB2 管理服务器日志文件（db2dasdiag.log）的格式类似于 DB2 FFDC 日志文件 db2diag.log 的格式。有关如何解释 db2dasdiag.log 文件的信息，请参阅《故障诊断》主题中『解释管理日志』的一节。

相关概念:

- 第 37 页的『DB2 管理服务器』

第 3 章 创建数据库

本章简要概述了各种对象，这些对象可能会成为您的数据库设计实现的一部分。

前一章主要讨论您在创建数据库之前需要了解的信息。该章还涉及几个主题以及在创建数据库之前必须执行的几项任务。

此部分的最后一章讨论您在改变数据库之前必须要考虑的事项。此外，该章还说明了如何改变或删除数据库对象。

创建数据库

先决条件:

创建数据库之前，应花足够多的时间来设计数据库的内容、布局、潜在增长和用途。

过程:

创建数据库时，为您完成了下列所有任务:

- 设置数据库所需的所有系统目录表
- 分配数据库恢复日志
- 创建数据库配置文件，设置缺省值
- 将数据库实用程序与数据库绑定

下列数据库特权被自动授予 PUBLIC：对系统目录视图的 CREATETAB、BINDADD、CONNECT、IMPLICIT_SCHEMA 和 SELECT 特权。

要使用“控制中心”创建数据库:

1. 展开对象树，直到您找到**数据库**文件夹为止。
2. 右键单击**数据库**文件夹，然后从弹出菜单中选择**创建 —> 标准**或**创建 —> 使用自动维护**。
3. 遵循步骤来完成此任务。

下列命令行处理器命令在缺省位置创建称为 personl 的一个数据库，并带有相关注释: ″Personnel DB for BSchiefer Co″。

```
CREATE DATABASE personl
   WITH "Personnel DB for BSchiefer Co"
```

创建数据库时，还可请求使用“配置顾问程序”来帮助配置数据库，而不是接受所有配置参数的缺省值。可通过在 **CREATE DATABASE** 命令上使用 AUTOCONFIGURE 选项来执行此任务:

```
CREATE DATABASE <database name>
   AUTOCONFIGURE
```

AUTOCONFIGURE 子句上有几个选项。在分区环境中创建数据库时不能使用 AUTOCONFIGURE 子句。

同时创建了数据库，还创建了详细的死锁事件监视器。同其它监视器一样，这个事件监视器将造成一些开销。如不需要详细死锁事件监视器，可运行以下命令来删除:

```
DROP EVENT MONITOR db2detaildeadlock
```

要限制此事件监视器消耗的磁盘空间，则在输出文件达到最大数时取消激活事件监视器，并将此消息写入到管理通知日志。将不再需要的输出文件除去即可在下次激活数据库时再次激活事件监视器。

您可以在另一个可能是远程的数据库管理器实例中创建数据库。在这种类型的环境中，还可以对不同于缺省实例的实例（包括远程实例）执行实例级别管理。

相关概念:

- 『要在数据库中记录的内容』（*《管理指南: 计划》*）
- 第 5 页的『数据库管理器的多个实例』
- 第 204 页的『数据库权限』
- 『附加数据库设计注意事项』（*《管理指南: 计划》*）

相关参考:

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

初始数据库分区组的定义

当最初创建数据库时，会为所有在 db2nodes.cfg 文件中指定的分区创建数据库分区。可使用 **ADD DBPARTITIONNUM** 和 **DROP DBPARTITIONNUM VERIFY** 命令来添加或除去其它分区。

定义了三个数据库分区组:

- 用于容纳 SYSCATSPACE 表空间的 IBMCATGROUP，它保存系统目录表
- 用于容纳 TEMPSPACE1 表空间的 IBMTEMPGROUP，它保存数据库处理期间创建的临时表
- 用于容纳 USERSPACE1 表空间的 IBMDEFAULTGROUP，缺省情况下它保存用户表和索引

相关概念:

- 『数据库分区组』（*《管理指南: 计划》*）

相关参考:

- 『ADD DBPARTITIONNUM Command』（*Command Reference*）
- 『DROP DBPARTITIONNUM VERIFY Command』（*Command Reference*）

定义初始表空间

当创建一个数据库时，要定义三个表空间:

- 用于系统目录表的 SYSCATSPACE
- 用于保存数据库处理期间创建的系统临时表的 TEMPSPACE1
- 用于保存用户定义的表和索引的 USERSPACE1

注: 当第一次创建一个数据库时，不创建用户临时表空间。

若未使用 **CREATE DATABASE** 命令指定任何表空间参数，则数据库管理器使用系统管理存储器（SMS）目录容器创建这些表空间。这些目录容器将在为该数据库创建的子目录中创建。这些表空间的扩展数据块大小被设置为缺省值。

先决条件:

必须创建数据库且必须具有创建表空间的权限。

过程:

要使用“控制中心”来定义初始表空间:

1. 展开对象树，直到您看到**数据库**文件夹为止
2. 右键单击**数据库**文件夹，然后从弹出菜单中选择**创建 —> 标准**或**创建 —> 使用自动维护**。
3. 遵循步骤来完成此任务。

要使用命令行来定义初始表空间，输入:

```
CREATE DATABASE <name>
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('<path>')
    EXTENTSIZE <value> PREFETCHSIZE <value>
  USER TABLESPACE
    MANAGED BY DATABASE USING (FILE'<path>' 5000,
                               FILE'<path>' 5000)
    EXTENTSIZE <value> PREFETCHSIZE <value>
  TEMPORARY TABLESPACE
    MANAGED BY SYSTEM USING ('<path>')
  WITH "<comment>"
```

若不希望使用这些表空间的缺省定义，可以在 **CREATE DATABASE** 命令上指定它们的特征。例如，可使用以下命令在 Windows 上创建数据库:

```
CREATE DATABASE PERSONL
  CATALOG TABLESPACE
    MANAGED BY SYSTEM USING ('d:\pcatalog','e:\pcatalog')
    EXTENTSIZE 16 PREFETCHSIZE 32
  USER TABLESPACE
    MANAGED BY DATABASE USING (FILE'd:\db2data\personl' 5000,
                               FILE'd:\db2data\personl' 5000)
    EXTENTSIZE 32 PREFETCHSIZE 64
  TEMPORARY TABLESPACE
    MANAGED BY SYSTEM USING ('f:\db2temp\personl')
  WITH "Personnel DB for BSchiefer Co"
```

在此示例中，每个初始表空间的定义是显式提供的。只需要为不希望使用缺省定义的那些表空间指定表空间定义。

注: 当您在分区数据库环境中工作时，不能创建容器或将容器指定给特定分区。首先，必须使用缺省用户和临时表空间创建数据库。然后应使用 CREATE TABLESPACE 语句来创建必需的表空间。最后，可删除缺省表空间。

CREATE DATABASE 命令上的 MANAGED BY 短语的编码与 CREATE TABLESPACE 语句上的 MANAGED BY 短语遵循同一格式。

相关概念:

- 第 61 页的『系统目录表的定义』
- 『表空间设计』（《*管理指南: 计划*》）

相关任务:

- 第 68 页的『创建表空间』

相关参考:

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

创建缓冲池

可以创建新的缓冲池以供数据库管理器使用。缓冲池可立即提高数据库系统性能。

为表空间指定的页大小将确定您为缓冲池选择的页大小。选择用于缓冲池的页大小是很重要的，这是因为创建缓冲池之后就不能改变页大小了。

先决条件:

语句的授权标识必须具有 SYSCTRL 或 SYSADM 权限。

在创建新的缓冲池之前，应解决下列问题:

- 使用什么缓冲池名称
- 是立即创建缓冲池，还是在下一次取消激活数据库并重新激活数据库之后创建缓冲池
- 是否想使缓冲池与组成数据库的所有数据库分区的子集相关联
- 您想使哪些值与用来控制缓冲池大小的参数（包括页大小和基于页数的缓冲池总大小）相关联
- 是想使用扩充存储器、基于块的支持还是这两者都不使用。

过程:

要创建新的缓冲池:

1. 使用 `SELECT BPNAME FROM SYSCAT.BUFFERPOOLS` 来获取数据库中已存在的缓冲池名称的列表。
2. 选择当前在结果列表中找不到的缓冲池名称。缓冲池名称一定不能以字符 “SYS” 或 “IBM” 开头。
3. 确定将创建的缓冲池的特征。
4. 确保您具有正确的授权标识来运行 CREATE BUFFERPOOL 语句。
5. 运行 CREATE BUFFERPOOL 语句。

相关任务:

- 第 146 页的『改变缓冲池』

相关参考:

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

系统目录表的定义

对于每个数据库，都会创建和维护一组系统目录表。这些表包含有关数据库对象（例如，表、视图、索引和程序包）的定义的信息，以及用户对这些对象所拥有的存取类型的安全性信息。这些表存储在 SYSCATSPACE 表空间中。

在一个数据库的操作期间，例如，创建一个表时，要更新这些表。不能显式地创建或删除这些表，但是可以查询和查看它们的内容。当创建该数据库时，除系统目录表对象外，在系统目录中还定义下列数据库对象:

- 模式 SYSIBM、SYSFUN 和 SYSPROC 中的一组例程（函数和过程）。
- 在 SYSCAT 模式中创建一组系统目录表的只读视图。
- 在 SYSSTAT 模式中创建一组可更新的目录视图。这些可更新的视图允许您更新特定的统计信息来调查假设数据库的性能，也可不使用 **RUNSTATS** 实用程序来更新统计信息。

在创建数据库之后，您也许希望限制对系统目录视图的存取。

相关概念:

- 『User-defined functions』（*SQL Reference, Volume 1*）
- 『Catalog views』（*SQL Reference, Volume 1*）
- 『Functions overview』（*SQL Reference, Volume 1*）

相关任务:

- 第 224 页的『保护系统目录视图』

相关参考:

- 『Functions』（*SQL Reference, Volume 1*）

数据库目录的定义

当建立或设置新数据库时，要使用三个目录。

- 本地数据库目录
- 系统数据库目录
- 节点目录

本地数据库目录

*本地数据库目录*文件存在于定义了数据库的每条路径（或 Windows® 操作系统的“驱动器”）中。对于可从该位置存取的每个数据库此目录都包含一个条目。每一个条目包含:

- 随 **CREATE DATABASE** 命令提供的数据库名称
- 数据库别名（若未指定别名，它与数据库名称相同）
- 随 **CREATE DATABASE** 命令提供的描述该数据库的注释
- 该数据库的根目录的名称
- 其它系统信息

相关参考:

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

系统数据库目录

对于数据库管理器的每个实例，都存在一个*系统数据库目录*文件，该文件对于针对此实例编目的每个数据库都包含一个条目。当发出 **CREATE DATABASE** 命令时将隐式地对数据库进行编目，也可以使用 **CATALOG DATABASE** 命令显式地对该数据库进行编目。

对于创建的每个数据库，都要将包含下列信息的一个条目添加至该目录:

- 随 **CREATE DATABASE** 命令提供的数据库名称
- 数据库别名（若未指定别名，它与数据库名称相同）
- 随 **CREATE DATABASE** 命令提供的数据库注释
- 本地数据库目录的位置
- 指示该数据库是*间接*数据库的指示符，表示数据库驻留在当前数据库管理器实例上
- 其它系统信息

在 UNIX® 平台和分区数据库环境中，必须确保所有数据库分区总是存取该实例主目录的 `sqldbdir` 子目录中的同一系统数据库目录文件 `sqldbdir`。如果系统数据库目录或同一 `sqldbdir` 子目录中的系统意向文件 `sqldbins` 是指向共享文件系统中的另一个文件的符号链接，可能会发生不可预测的错误。

相关任务:

- 第 10 页的『在数据库中启用数据分区』
- 第 66 页的『对数据库进行编目』

相关参考:

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

为数据库标识备用服务器

当服务器崩溃时，与服务器相连的每台客户机均会接收到通信错误，这会使导致应用程序错误的连接终止。如果可用性极其重要，则应实现冗余设置或具备将服务器故障转移到备用节点的能力。在上述任一情况下，DB2 Universal Database™ (DB2 UDB)（DB2 通用数据库）客户机代码尝试重新建立与原始服务器的连接，此服务器可能正在故障转移节点上运行（IP 地址也进行故障转移），或者建立与新服务器的连接。

过程:

要定义新服务器或备用服务器，请使用 UPDATE ALTERNATE SERVER FOR DATABASE 命令。此命令在系统数据库目录中更新数据库别名的备用服务器信息。

相关概念:

- 第 65 页的『自动客户机重新路由实现』
- 第 273 页的『自动客户机重新路由的描述和设置』

查看本地或系统数据库目录文件

您可能想查看与系统上的数据库相关联的某些信息。

先决条件:

在查看本地或系统数据库目录文件之前，必须先创建实例和数据库。

过程:

要查看本地数据库目录文件的内容，发出以下命令，其中 <location> 指定该数据库的位置:

```
LIST DATABASE DIRECTORY ON <location>
```

要查看系统数据库目录文件的内容，发出 **LIST DATABASE DIRECTORY** 命令而不指定该数据库目录文件的位置。

相关参考:

- 『LIST DATABASE DIRECTORY Command』（*Command Reference*）

节点目录

数据库管理器在对第一个数据库分区进行编目时会创建*节点目录*。要对数据库分区进行编目，使用 **CATALOG NODE** 命令。要列示本地节点目录的内容，使用 **LIST NODE DIRECTORY** 命令。在每个数据库客户机上都创建并维护节点目录。对于具有客户机可以存取的一个或多个数据库的每个远程工作站，该目录都包含一个条目。无论何时请求数据库连接或实例连接，DB2® 客户机都会使用该节点目录中的通信端点信息。

该目录中的条目还包含客户机与远程数据库分区通信要使用的通信协议的类型信息。对本地数据库分区进行编目将为驻留在同一台机器上的实例创建一个别名。

相关参考:

- 『CATALOG TCPIP NODE Command』（*Command Reference*）
- 『LIST NODE DIRECTORY Command』（*Command Reference*）
- 『CATALOG NETBIOS NODE Command』（*Command Reference*）
- 『CATALOG LOCAL NODE Command』（*Command Reference*）
- 『CATALOG NAMED PIPE NODE Command』（*Command Reference*）

“轻量级目录访问协议”（LDAP）目录服务

目录服务是一个关于分布式环境中的多个系统和服务的资源信息的资源库；它提供对这些资源的客户机和服务器访问。客户机和服务器将使用目录服务来找出如何存取其它资源。在分布式环境中，必须将有关这些其它资源的信息输入到目录服务库中。

“轻量级目录访问协议”（LDAP）是一个业界标准的访问目录服务的方法。每个数据库服务器实例都会将它的存在情况发布给 LDAP 服务器，并在创建数据库时向 LDAP 目录提供数据库信息。当客户机与数据库连接后，可从 LDAP 目录检索服务器的目录信

息。不再要求每个客户机将目录信息以本地方式存储在每台机器上。客户机应用程序搜索 LDAP 目录，查找连接数据库所需的信息。

作为 DB2® UDB 系统的管理员，您可以建立和维护目录服务。“配置助手”或“控制中心”可帮助维护此目录服务。可通过“轻量级目录访问协议”（LDAP）目录服务来使目录服务对 DB2 UDB 可用。要使用 LDAP 目录服务，首先必须有一个 DB2 支持的 LDAP 服务器，以便存储目录信息。

注: 当在 Windows® 2000 域环境中运行时，LDAP 服务器已经可用，因为它是与 Windows 2000 Active Directory 集成在一起的。因此，每台运行 Windows 2000 的机器都可使用 LDAP。

LDAP 目录在这样的企业环境中是非常有帮助的: 在该环境中，由于有数目庞大的客户机，在每台客户机上更新本地目录会非常困难。如果遇到此状况，建议将目录条目存储在 LDAP 服务器中，以便在一个位置（即 LDAP 服务器）完成目录条目的维护。购买和维护 LDAP 服务器的成本可能很高，因此仅应在有足够数目的客户机分担此成本时才考虑这样做。

相关概念:

- 第 52 页的『管理服务器、实例和数据库的发现』
- 第 279 页的『“轻量级目录访问协议”（LDAP）简介』

创建数据库分区组（节点组）

可使用 CREATE DATABASE PARTITION GROUP 语句创建数据库分区组。此语句指定表空间容器和表数据将驻留其上的一组数据库分区。此语句还可以:

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

先决条件:

机器和系统必须可用且能够处理分区数据库环境。已购买并安装了 DB2 通用数据库企业服务器版。该数据库必须存在。

过程:

要使用“控制中心”创建数据库分区组:

1. 展开对象树，直到您看到**数据库分区组**文件夹。
2. 右键单击**数据库分区组**文件夹，并从弹出菜单中选择**创建**。
3. 在“创建数据库分区组”窗口上，填写信息，使用箭头来将节点从**可用的节点**框移至**选择的数据库分区**框，并单击**确定**。

要使用命令行创建数据库分区组，输入:

```
CREATE DATABASE PARTITION GROUP <name> ON PARTITIONS (<value>,<value>)
```

假定要在数据库中一个数据库分区子集上装入一些表。应使用以下命令创建一个数据库分区组，它包含至少由三个（0 至 2）数据库分区组成的数据库中的两个数据库分区（1 和 2）：

```
CREATE DATABASE PARTITION GROUP mixng12 ON PARTITIONS (1,2)
```

CREATE DATABASE 命令或 sqlecrea() API 还会创建缺省系统数据库分区组 IBMDEFAULTGROUP、IBMCATGROUP 和 IBMTEMPGROUP。

相关概念:

- 『数据库分区组』（*《管理指南: 计划》*）
- 『分区映射』（*《管理指南: 计划》*）

相关参考:

- 『CREATE DATABASE PARTITION GROUP statement』（*SQL Reference, Volume 2*）
- 『sqlecrea - Create Database』（*Administrative API Reference*）
- 『CREATE DATABASE Command』（*Command Reference*）

数据库恢复日志的定义

*数据库恢复日志*保存对一个数据库所做的所有更改（包括新表的添加或对现有表的更新）的记录。此日志由大量*日志块*组成，每一个日志块包含在称为*日志文件*的一个单独的文件中。

数据库恢复日志可以用于确保故障（例如，系统断电或应用程序出错）不会使数据库处于不一致的状态。若发生故障，则回滚已进行而未落实的更改，重新执行可能未实际写入磁盘的所有已落实事务。这些操作确保了数据库的完整性。

相关概念:

- 『了解恢复日志』（*《数据恢复及高可用性指南与参考》*）

自动客户机重新路由实现

当 DB2® 通用数据库（DB2 UDB）客户机应用程序失去与 DB2 UDB 服务器的通信时，可能希望客户机能够恢复通信，无需您或另一管理员的干预。DB2 UDB 支持恢复客户机与 DB2 UDB 服务器之间的通信。通信失败前需要执行的操作是建立客户机连接知悉的备用位置。

使用 UPDATE ALTERNATE SERVER FOR DATABASE 命令来定义特定数据库上的备用服务器位置。备用主机名和端口号作为命令的部分提供。位置存储在服务器上系统数据库目录文件中。

在服务器实例的特定数据库上指定备用服务器位置之后，将备用服务器位置信息作为连接过程的一部分返回至客户机。如果客户机与服务器之间的通信因某种原因而丢失，则已编码的 DB2 UDB 客户机将使用备用服务器信息尝试重新建立连接。DB2 UDB 客户机将尝试与原始服务器和备用服务器重新连接，从两个服务器中选择一个。可能是非常快速的尝试，也可能是两次尝试之间时间间隔逐渐增加的尝试，所以这些尝试的计时不同。

一旦连接成功，则返回 SQLCODE -30108 以指示通信失败后已重新建立数据库连接。返回主机名 / IP 地址和服务名称 / 端口号。如果不能重新建立客户机与原服务器或备用服务器之间的通信，则客户机代码仅向应用程序返回原始通信故障的错误。

相关概念:

- 第 273 页的『自动客户机重新路由的描述和设置』
- 第 274 页的『自动客户机重新路由限制』

相关参考:

- 第 275 页的『自动客户机重新路由示例』

将实用程序绑定至数据库

当创建一个数据库时，数据库管理器试图将 db2ubind.lst 中的实用程序与该数据库绑定。此文件存储在 sqllib 目录的 bnd 子目录中。

绑定一个实用程序将创建一个*程序包*，程序包是这样一个对象，它包括处理单个源文件中特定 SQL 语句所需的所有信息。

注: 若希望从客户机使用这些实用程序，必须显式地绑定它们。

若由于某种原因需要将实用程序与数据库绑定或重新绑定，则使用命令行处理器发出下列命令:

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

注: 必须位于这些文件所驻留的目录中，才能在 sample 数据库中创建程序包。可在 sqllib 目录的 bnd 子目录中找到这些绑定文件。在此示例中，sample 是该数据库的名称。

相关任务:

- 第 57 页的『创建数据库』

相关参考:

- 『BIND Command』（*Command Reference*）

对数据库进行编目

当创建一个新数据库时，会在系统数据库目录文件中自动对它进行编目。还可以使用 **CATALOG DATABASE** 命令在系统数据库目录文件中显式地对数据库进行编目。**CATALOG DATABASE** 命令允许您使用另一别名对数据库进行编目，或对先前使用 **UNCATALOG DATABASE** 命令删除的数据库条目进行编目。

先决条件:

虽然数据库是在创建数据库时自动编目的，但仍需要对数据库进行编目。对数据库进行编目时，该数据库必须存在。

过程:

下列命令行处理器命令将 personl 数据库编目为 humanres:

```
CATALOG DATABASE personl AS humanres
      WITH "Human Resources Database"
```

此处，系统数据库目录条目将使 humanres 作为数据库别名，以便与数据库名称（personl）区分。

还可以在非缺省的实例上对数据库进行编目。在下列示例中，与数据库 B 的连接还连接至 INSTNC_C。在尝试此命令前，实例 instnc_c 必须已编目为本地节点。

```
CATALOG DATABASE b as b_on_ic AT NODE instnc_c
```

注: 在客户机节点上还会使用 **CATALOG DATABASE** 命令来对驻留在数据库服务器上的数据库进行编目。

注: 缺省情况下，可使用“目录高速缓存支持（*dir_cache*）”配置参数来在内存中高速缓存目录文件，包括数据库目录。当启用目录高速缓存时，另一个应用程序对目录所做的更改（例如，使用 **CATALOG DATABASE** 或 **UNCATALOG DATABASE** 命令）可能要在重新启动应用程序后才会生效。要刷新命令行处理器会话所用的目录高速缓存，发出 **db2 terminate** 命令。

在分区数据库中，在每个数据库分区上创建目录文件的高速缓存。

除应用程序级高速缓存外，数据库管理器级高速缓存也用于内部的数据库管理器查找。要刷新此“共享”高速缓存，发出 **db2stop** 和 **db2start** 命令。

相关任务:

- 第 67 页的『使用关于远程数据库服务器的信息来更新目录』

相关参考:

- 『dir_cache - 目录高速缓存支持配置参数』（*《管理指南: 性能》*）
- 『CATALOG DATABASE Command』（*Command Reference*）
- 『TERMINATE Command』（*Command Reference*）
- 『UNCATALOG DATABASE Command』（*Command Reference*）

使用关于远程数据库服务器的信息来更新目录

过程:

可使用“配置助手”（CA）解释器的“添加数据库向导”来创建目录条目。如果有 DB2 应用程序开发客户机，还可以创建应用程序来对条目进行编目。

注: 要对数据库进行编目，必须具有 SYSADM 或 SYSCTRL 权限；或者，必须将 *catalog_noauth* 配置参数设置为 YES。

要使用命令行处理器更新目录，执行以下操作:

1. 使用下列其中一个命令来更新节点目录:
   - 对于具有 APPC 连接的节点:

     ```
     db2 CATALOG APPC NODE <nodename>
       REMOTE <symbolic_destination_name> SECURITY <security_type>
     ```

例如:

```
db2 CATALOG APPC NODE DB2NODE REMOTE DB2CPIC SECURITY PROGRAM
```

- 对于具有 TCP/IP 连接的 DB2 通用数据库 z/OS 和 OS/390 版版本 5.1（或更新版本）或 DB2 通用数据库 AS/400 版版本 4.2（或更新版本）数据库:

```
db2 CATALOG TCPIP NODE <nodename>
   REMOTE <hostname> or <IP address>
   SERVER <service_name> or <port_number>
   SECURITY <security_type>
```

例如:

```
db2 CATALOG TCPIP NODE MVSIPNOD REMOTE MVSHOST SERVER DB2INSTC
```

在 DB2 OS/390 和 z/OS 版上，用于 TCP/IP 连接 TCP/IP 的缺省端口是 446。

2. 如果使用 DB2 Connect 连接，必须考虑使用 CATALOG DCS DATABASE 命令来更新 DCS 目录。

如果有远程客户机，还必须更新每台远程客户机上的目录。

相关概念:

- 『系统数据库目录值』（*《DB2 Connect 用户指南》*）
- 『DCS 目录值』（*《DB2 Connect 用户指南》*）

相关参考:

- 『CATALOG DATABASE Command』（*Command Reference*）
- 『CATALOG DCS DATABASE Command』（*Command Reference*）
- 『CATALOG APPC NODE Command』（*Command Reference*）
- 『CATALOG TCPIP NODE Command』（*Command Reference*）
- 『CATALOG NETBIOS NODE Command』（*Command Reference*）
- 『CATALOG APPN NODE Command』（*Command Reference*）

创建表空间

表空间建立数据库系统使用的物理存储设备与用来存储数据的逻辑容器或表之间的关系。

先决条件:

创建表空间时，必须知道将引用的容器的设备名或文件名。另外，必须知道与要分配给表空间的每个设备名或文件名相关联的空间量。

过程:

在一个数据库内创建表空间，会将容器分配到表空间，并在数据库系统目录中记录它的定义和属性。然后就可以在此表空间内创建表。

要使用“控制中心”来创建表空间:

1. 展开对象树，直到您看到**表空间**文件夹为止。
2. 右键单击**表空间**文件夹，并从弹出菜单中选择**创建 -> 使用向导创建表空间**。
3. 遵循向导中的步骤来完成该任务。

要使用命令行来创建 SMS 表空间，输入:

```
CREATE TABLESPACE <NAME>
   MANAGED BY SYSTEM
   USING ('<path>')
```

要使用命令行来创建 DMS 表空间，输入:

```
CREATE TABLESPACE <NAME>
   MANAGED BY DATABASE
   USING (FILE'<path>' <size>)
```

通过使用三个不同的驱动器上的三个目录，下列 SQL 语句在 Windows 上创建了一个 SMS 表空间:

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

以下 SQL 语句使用各自有 5,000 页的两个文件容器创建了一个 DMS 表空间:

```
CREATE TABLESPACE RESOURCE
   MANAGED BY DATABASE
   USING (FILE'd:\db2data\acc_tbsp' 5000,
          FILE'e:\db2data\acc_tbsp' 5000)
```

在前面两个示例中，为容器提供了显式的名称。但是，若指定相对容器名，则将在为该数据库创建的子目录中创建容器。

另外，若指定的路径名的一部分不存在，数据库管理器将会创建它。若数据库管理器创建了一个子目录，当删除该表空间时数据库管理器也可能将它删除。

在上述示例中，假定这些表空间与特定的数据库分区组无关。如果未在该语句中指定下列参数，将使用缺省数据库分区组 IBMDEFAULTGROUP:

```
IN database_partition_group_name
```

通过使用各有 10 000 页的三个逻辑卷，下列 SQL 语句在基于 UNIX 的系统上创建了一个 DMS 表空间，并指定它们的 I/O 特征:

```
CREATE TABLESPACE RESOURCE
      MANAGED BY DATABASE
      USING (DEVICE '/dev/rdblv6' 10000,
             DEVICE '/dev/rdblv7' 10000,
             DEVICE '/dev/rdblv8' 10000)
      OVERHEAD 12.67
      TRANSFERRATE 0.18
```

在此 SQL 语句中提到的 UNIX 设备必须已经存在，且实例所有者和 SYSADM 组必须能够写入它们。

以下示例将在 UNIX 分区数据库中称为 ODDGROUP 的数据库分区组上创建一个 DMS 表空间。ODDGROUP 必须是先前使用 CREATE DATABASE PARTITION GROUP 语

句创建的。在此示例中，假设 ODDGROUP 数据库分区组由编号为 1、3 和 5 的数据库分区组成。在所有数据库分区上都使用具有 10 000 个 4 KB 页的 /dev/hdisk0 设备。另外，还为每个数据库分区声明了包含 40 000 个 4 KB 页的设备。

```
CREATE TABLESPACE PLANS IN ODDGROUP
      MANAGED BY DATABASE
      USING (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n1hd01' 40000)
             ON DBPARTITIONNUM 1
            (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n3hd03' 40000)
             ON DBPARTITIONNUM 3
            (DEVICE '/dev/HDISK0' 10000, DEVICE '/dev/n5hd05' 40000)
             ON DBPARTITIONNUM 5
```

UNIX 设备分为两类: 字符串行设备和块结构化设备。对于所有文件系统设备，正常情况是每个块设备（或*已处理的*设备）都有一个相应的字符串行设备（或*原始*设备）。块结构化设备通常由类似于 “hd0” 或 “fd0” 的名称来指定。字符串行设备通常由类似于 “rhd0” 、 “rfd0” 或 “rmt0” 的名称来指定。这些字符串行设备存取速度比块设备快。在 CREATE TABLESPACE 命令上应使用字符串行设备名，而不应使用块设备名。

开销和传送速率有助于确定编译 SQL 语句时使用的最佳存取路径。当前缺省值为:

- OVERHEAD 12.67 ms
- TRANSFERRATE 0.18 ms

DB2 UDB 可使用顺序预取设施（此设施使用并行 I/O）来大大改进顺序 I/O 的性能。

您还可以创建一个表空间，它使用的页大小比缺省的 4 KB 大小更大。下列 SQL 语句在基于 UNIX 的系统上创建一个具有 8 KB 页大小的 SMS 表空间。

```
CREATE TABLESPACE SMS8K
      PAGESIZE 8192
      MANAGED BY SYSTEM
      USING ('FSMS_8K_1')
      BUFFERPOOL BUFFPOOL8K
```

注意相关联的缓冲池也必须具有相同的 8 KB 页大小。

创建的表空间只有在它所引用的缓冲池被激活后才能使用。

可以使用 ALTER TABLESPACE SQL 语句对 DMS 表空间添加、删除容器或调整容器的大小，并修改表空间的 PREFETCHSIZE、OVERHEAD 和 TRANSFERRATE 设置。应尽可能落实发出表空间语句的事务，以防止发生系统目录争用。

注: PREFETCHSIZE 应该是 EXTENTSIZE 的倍数。例如，如果 EXTENTSIZE 是 10，则 PREFETCHSIZE 应为 20 或 30。当创建表空间时，应该使用以下等式手工设置预取大小:

```
预取大小 =（容器数）X（每个容器的物理主轴数）X 扩展数据块大小
```

还应该考虑允许 DB2 UDB 自动确定预取大小。

相关概念:

- 『表空间设计』（*《管理指南: 计划》*）
- 『系统管理空间』（*《管理指南: 计划》*）
- 『数据库管理空间』（*《管理指南: 计划》*）
- 『顺序预取』（*《管理指南: 性能》*）

相关任务:

- 『在 64 位环境中启用大页支持（AIX）』（*《管理指南: 计划》*）

相关参考:

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

创建特定类型的表空间

数据库管理器、应用程序及用户使用不同类型的表空间。

创建系统临时表空间

虽然系统临时表空间是在创建数据库时缺省创建的，您可能想要分配独立的表空间以处理系统排序任务。

先决条件:

与系统临时表空间相关联的容器必须存在。

限制:

因为系统临时表只能存储在系统临时表空间中，所以数据库必须始终至少有一个这样的表空间。

过程:

系统临时表空间用来存储系统临时表。创建数据库时，定义的三个缺省表空间的其中一个表空间便是名为 “TEMPSPACE1” 的系统临时表空间。

可使用 CREATE TABLESPACE 语句来创建另一个系统临时表空间。例如，

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

对于每个页大小至少应具有一个表空间。

创建系统临时表空间时，只能指定数据库分区组 IBMTEMPGROUP。

相关任务:

- 第 71 页的『创建用户临时表空间』

相关参考:

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

创建用户临时表空间

用户临时表空间不是在创建数据库时缺省创建的。在应用程序处理数据库中的数据时，您可能需要使用临时表。如果是这样的话，则需要创建用户临时表。

过程:

用户临时表空间用来存储已声明临时表。

可使用 CREATE TABLESPACE 语句来创建用户临时表空间:

```
CREATE USER TEMPORARY TABLESPACE usr_tbsp
   MANAGED BY DATABASE
   USING (FILE 'd:\db2data\user_tbsp' 5000,
          FILE 'e:\db2data\user_tbsp' 5000)
```

与常规表空间一样，可在不同于 IBMTEMPGROUP 的任何数据库分区组中创建用户临时表空间。创建用户临时表空间时使用的缺省数据库分区组是 IBMDEFAULTGROUP。

DECLARE GLOBAL TEMPORARY TABLE 语句定义供在用户临时表空间中使用的已声明临时表。

相关任务:

- 第 89 页的『创建用户定义的临时表』

相关参考:

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

在数据库分区组中创建表空间

通过将表空间放置在多分区数据库分区组中，将该表空间内的所有表划分到或分区到该数据库分区组的每个分区中。将表空间创建到数据库分区组中。一旦表空间位于某个数据库分区组中，它就必须保留在该处；而不能更改至另一数据库分区组。CREATE TABLESPACE 语句用于将表空间与数据库分区组关联。

相关参考:

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

指定原始 I/O

使用容器存储数据时，DB2 通用数据库支持直接磁盘存取（原始 I/O）。此类型的支持允许您将直接磁盘存取（原始）设备连接至任何 DB2 通用数据库系统。（唯一的例外是 Windows 9x 操作系统。）

先决条件:

创建表空间时，必须知道准备引用的容器的设备名或文件名。必须知道与要分配给表空间的每个设备名或文件名相关联的空间量。

需要正确的许可权才能读写容器。

过程:

以下列表说明用于标识直接磁盘存取类型设备的物理方法和逻辑方法:

- 在 Windows 上，要指定物理硬盘驱动器，使用以下语法:

  \\.\PhysicalDriveN

其中 N 表示系统中的一个物理驱动器。在这种情况下，N 可以替换为 0、1、2 或任何其它正整数:

\\.\PhysicalDrive5

- 在 Windows 上，要指定逻辑驱动器（即，未格式化的分区），使用以下语法:

  \\.\N:

  其中 N: 表示系统中的一个逻辑驱动器盘符。例如，N: 可被 E: 或任何其它驱动器盘符替换。要克服使用盘符来标识驱动器所带来的局限性，可对逻辑驱动器使用全局唯一标识（GUID）。
- **注意**: 必须安装有带有 Service Pack 3 或更高版本的 Windows NT V4.0 才能写入设备。
- 在基于 UNIX 的平台上，逻辑卷对用户和应用程序可以单个相邻且可扩展的磁盘卷出现。尽管它以此方式显示，但是，它也可以驻留在不相邻的物理分区上，甚至可以出现在多个物理卷上。逻辑卷还必须包含在单个卷组中。每个卷组最多只能有 256 个逻辑卷。每个卷组最多只能有 32 个物理卷。可以使用 **mklv** 命令来创建附加的逻辑卷。此命令允许您指定逻辑卷的名称和定义它的特征，包括要为它分配的逻辑分区的数目和位置。

  在创建逻辑卷之后，可以使用 **chlv** 命令来更改它的名称和特征，并且可以使用 **extendlv** 命令来增加分配给它的逻辑分区数。在创建时，逻辑卷的缺省最大大小为 512 个逻辑分区，除非您将它指定为更大。**chlv** 命令可用来重设此限制。

  在 AIX 中，操作系统命令、库子例程以及其它允许您建立和控制逻辑卷存储器的工具的集合称为“逻辑卷管理器”（LVM）。LVM 通过在存储空间的简单而灵活的逻辑视图与实际的物理磁盘之间映射数据来控制磁盘资源。

  有关 **mklv** 和其它逻辑卷命令的更多信息，请参阅 *AIX 5L Version 5.2 System Management Concepts: Operating System and Devices*。

对于 Windows 2000 及更高版本的系统，有一个新方法可用来指定 DMS 原始表空间容器。卷（即基本磁盘分区或动态卷）是被创建时指定的全局唯一标识（GUID）。在表空间定义中指定容器时，可将 GUID 用作设备标识。GUID 在系统间是唯一的，这意味着在多分区数据库配置中，每个分区的 GUID 各不相同，即使磁盘分区定义相同也是如此。

工具 *db2listvolumes.exe* 可用来（仅在 Windows 操作系统上）使 Windows 系统上定义的所有磁盘卷的 GUID 显示起来更加容易。此工具在其运行的当前目录中创建两个文件。一个文件称为 `volumes.xml`，包含有关用 XML 编码的每个磁盘卷的信息，以易于在启用了 XML 的浏览器上进行查看。第二个文件称为 `tablespace.ddl`，包含指定表空间容器的必需语法。必须更新此文件才能填写表空间定义所需的余下信息。*db2listvolumes* 工具不需要任何命令行自变量。

相关任务:

- 第 74 页的『在 Linux 上设置原始 I/O』

在 Linux 上设置原始 I/O

使用容器存储数据时，DB2 通用数据库支持直接磁盘存取（原始 I/O）。此类型的支持允许您将直接磁盘存取（原始）设备连接至任何 DB2 通用数据库系统。在 Linux 环境中工作时有特定的信息。

先决条件:

创建表空间时，必须知道准备引用的容器的设备名或文件名。必须知道与要分配给表空间的每个设备名或文件名相关联的空间。

在 Linux 上设置原始 I/O 之前，需要下列条件:

- 一个或多个可用 IDE 或 SCSI 磁盘分区
- 名为 /dev/rawctl 或 /dev/raw 的裸设备控制器。若没有该设备，则创建一个符号链接:

  ```
  # ln -s /dev/your_raw_dev_ctrl /dev/rawctl
  ```

- 原始实用程序，通常通过 Linux 分发提供该实用程序

注: 对于当前支持原始 I/O 的分发，裸设备节点的命名各不相同:

表 3. Linux 分发支持原始 I/O。

分发	裸设备节点	裸设备控制器
RedHat 或 TurboLinux	/dev/raw/raw1 为 255	/dev/rawctl
SuSE	/dev/raw1 为 63	/dev/raw

DB2 支持上述裸设备控制器中的任何一种，并且支持裸设备节点的大多数其它名称。在 Linux/390 上 DB2 不支持裸设备。

过程:

Linux 具有一个裸设备节点池，必须将该池绑定到块设备上，然后才能对该池执行原始 I/O。有一个裸设备控制器，它充当裸设备至块设备绑定信息的中心库。使用名为 raw 的实用程序执行绑定，该程序通常由 Linux 发行人提供。

要在 Linux 上配置原始 I/O:

在本示例中，要使用的原始分区是 /dev/sda5。它应该不包含任何有用的数据。

步骤 1. 计算本分区中的页面数，每个页面的字节数为 4 096 个字节，必要时四舍五入。
例如:

```
# fdisk /dev/sda
Command (m for help): p

Disk /dev/sda: 255 heads, 63 sectors, 1106 cylinders
Units = cylinders of 16065 * 512 bytes
```

表 4. Linux 原始 I/O 计算。

设备引导	开始	结束	块	标识	系统
/dev/sda1	1	523	4200997	83	Linux
/dev/sda2	524	1106	4682947+	5	扩展
/dev/sda5	524	1106	4682947	83	Linux

```
Command (m for help): q
#
```

在 /dev/sda5 中的页数为

```
num_pages = floor( ((1106-524+1)*16065*512)/4096 )
num_pages = 11170736
```

步骤 2. 将未使用的裸设备节点绑定至此分区。每次重新引导机器时，需要执行此操作，且需要 root 用户存取权。使用 raw -a 来查看已经在使用哪些裸设备节点:

```
# raw /dev/raw/raw1 /dev/sda5
/dev/raw/raw1: bound to major 8, minor 5
```

步骤 3. 设置对裸设备控制器和磁盘分区的相应读许可权。设置对裸设备的相应读写权限。

步骤 4. 在 DB2 中创建表空间，指定裸设备，而不是磁盘分区。例如:

```
CREATE TABLESPACE dms1
      MANAGED BY DATABASE
USING (DEVICE '/dev/raw/raw1' 11170736)
```

DB2 所支持的所有其它页大小也支持裸设备上的表空间。

相关任务:

- 第 72 页的『指定原始 I/O』

创建模式

在将数据组织成表时，它可能还有助于将表和其它相关的对象编组在一起。为此，须使用 CREATE SCHEMA 语句来定义一个模式。有关该模式的信息保存在连接的数据库的系统目录表中。当创建其它对象时，就可以将它们置于此模式内。

先决条件:

数据库表及要组合在一起的其它对象必须存在。

限制:

此语句必须由具有 DBADM 权限的用户发出。

模式也可能是在用户具有 IMPLICIT_SCHEMA 权限时隐式创建的。使用此权限，无论何时用户使用未存在的模式名创建对象，都会隐式创建一个模式。

若用户不具有 IMPLICIT_SCHEMA 权限，则他们可以创建的唯一模式是具有与他们自己的授权标识同名的模式。

由于使用模式来强制数据库中的唯一性，所以不允许对模式内的对象的非限定存取。当考虑两个用户有可能使用同一名称创建两个表（或其它对象）时，这一点变得很清楚。没有模式来强制唯一性时，若第三个用户试图查询该表，将会造成不明确。没有进一步的限定时，则不可能确定要使用哪一个表。

系统目录中不能已经存在该新模式名，且该名称不能以“SYS”开头。

过程:

若用户具有 SYSADM 或 DBADM 权限，则用户可以使用任何有效的名称来创建模式。当创建数据库时，会将 IMPLICIT_SCHEMA 权限授予 PUBLIC（即，授予所有用户）。

作为 CREATE SCHEMA 语句的一部分创建的任何对象的定义者是模式所有者。此所有者可以授予和撤销其他用户的模式特权。

要允许另一用户来存取表而不必输入模式名作为表名的限定部分，需要为该用户建立视图。视图定义将定义包括用户模式的全限定表名；用户将只需要使用视图名进行查询。将通过作为视图定义一部分的用户模式对视图进行全限定。

要使用“控制中心”创建模式:

1. 展开对象树，直到您看到数据库中的**模式**文件夹为止。
2. 右键单击**模式**文件夹,并单击**创建**。
3. 填写新模式的信息，并单击**确定**。

要使用命令行来创建模式，输入:

```
CREATE SCHEMA <name> AUTHORIZATION <name>
```

以下是 CREATE SCHEMA 语句的一个示例，它为具有授权标识“joe”的各个用户创建模式:

```
CREATE SCHEMA joeschma AUTHORIZATION joe
```

相关概念:

- 第 7 页的『根据模式对对象分组』
- 第 206 页的『隐式模式权限（IMPLICIT_SCHEMA）注意事项』
- 第 206 页的『模式特权』

相关任务:

- 第 76 页的『设置模式』

相关参考:

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

有关创建模式的详细信息

模式用于在数据库中组织对象所有权。

设置模式

如果有几个模式，您也许想将其中一个指定为缺省模式，以用于在特定 DB2 连接内发出的动态 SQL 语句中的非限定对象引用。

过程:

建立缺省模式是通过将专用寄存器 CURRENT SCHEMA 设置为想要用作缺省模式的模式来完成的。任何用户都可以设置此专用寄存器：不需任何授权。

以下是如何设置 CURRENT SCHEMA 专用寄存器的一个示例:

```
SET CURRENT SCHEMA = 'SCHEMA01'
```

此语句可以在应用程序内使用或以交互方式发出。CURRENT SCHEMA 专用寄存器的值一旦被设置后，就用作动态 SQL 语句中非限定对象引用的限定符（模式），但存在对数据库对象的非限定引用的 CREATE SCHEMA 语句除外。

CURRENT SCHEMA 专用寄存器的初始值等于当前会话用户的授权标识。

相关概念:

- 『Schemas』（*SQL Reference, Volume 1*）

相关参考:

- 『SET SCHEMA statement』（*SQL Reference, Volume 2*）
- 『Reserved schema names and reserved words』（*SQL Reference, Volume 1*）
- 『CURRENT SCHEMA special register』（*SQL Reference, Volume 1*）

第 4 章 创建表和其它相关表对象

本章描述实现数据库设计时如何创建具有特定特征的表。

创建和填充表

表是在数据库中存储数据的主要的库。创建表并输入数据以填写表将在创建新数据库时进行。

先决条件:

必须花时间来设计和组织将用来保存数据的表。

过程:

在确定如何将数据组织成表之后，下一步就是使用 CREATE TABLE 语句来创建那些表。表描述存储在连接的数据库的系统目录中。

CREATE TABLE 语句给予该表一个名称（该名称可以是限定的或非限定的标识），并给它的每个列一个定义。可以将每个表存储在单独的表空间中，以使一个表空间只包含一个表。若一个表经常被删除和创建，一个更有效的方法是将它存储在单独的表空间中，然后删除该表空间而不是表。也可以将多个表存储在单个表空间中。在分区数据库环境中，选择的表空间还会定义用来存储表数据的数据库分区组和数据库分区。

最初，该表不包含任何数据。要将数据行添加至该表，使用下列其中一项:

- INSERT 语句
- LOAD 或 IMPORT 命令
- Autoloader 实用程序（如果是在分区数据库环境中工作）

可以在不记录更改的情况下向表添加数据。CREATE TABLE 语句上的 NOT LOGGED INITIALLY 子句阻止记录对表的更改。在创建该表的同一个工作单元中由 INSERT、DELETE、UPDATE、CREATE INDEX、DROP INDEX 或 ALTER TABLE 操作对该表所做的任何更改都不会被记录。记录在后续的工作单元中开始。

一个表由一个或多个列定义组成。可为一个表定义最多 500 列。列表示实体的属性。任何列中的值都是相同类型的信息。

注: 当使用 4 KB 页大小时，最多可有 500 列。当使用 8 KB、16 KB 或 32 KB 页大小时，最多可有 1012 列。

列定义包括*列名*、*数据类型*和任何需要的 *null 属性*或缺省值（由用户可选）。

列名描述列中包含的信息，它应是易于识别的。列名在表内必须是唯一的；但是，在其它表中可以使用与它相同的名称。

列的数据类型指示列值的长度和对该列有效的数据类型。数据库管理器使用字符串、数字、日期、时间和大对象数据类型。图形字符串数据类型只可用于使用多字节字符集的数据库环境。另外，可使用用户定义的单值类型来定义列。

缺省属性规范指示在未提供值的情况下，将使用什么值。可以指定缺省值，也可以使用系统定义的缺省值。可以为带或不带空值属性规范的列指定缺省值。

null 属性规范指示列是否可以包含空值。

要使用“控制中心”来创建表:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 右键单击**表**文件夹，并单击**创建**。
3. 遵循向导中的步骤来完成该任务。

要使用命令行来创建表，输入:

```
CREATE TABLE <NAME>
   (<column_name>  <data_type>  <null_attribute>)
   IN <TABLE_SPACE_NAME)
```

以下是使用 CREATE TABLE 语句在 RESOURCE 表空间中创建 EMPLOYEE 表的一个示例。此表是在样本数据库中定义的:

```
CREATE TABLE EMPLOYEE
   (EMPNO     CHAR(     NOT NULL PRIMARY KEY,
    FIRSTNME   VARCHAR(12)  NOT NULL,
    MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
    LASTNAME   VARCHAR(15)  NOT NULL,
    WORKDEPT   CHAR(3),
    PHONENO    CHAR(4),
    PHOTO     BLOB(10M)   NOT NULL)
IN RESOURCE
```

当创建一个表时，可根据结构化类型的属性选择该表的某些列。这种表称为“类型表”。

可定义一个类型表，以便从另一个类型表继承其某些列。这种表称为“子表”，而被继承的表称为其“超表”。一个类型表及其所有子表的组合称为“表层次结构”。该表层次结构中最上层的表（没有超表的表）称为该层次结构的“根表”。

要声明全局临时表，使用 DECLARE GLOBAL TEMPORARY TABLE 语句。

也可以创建根据查询结果定义的表。这种类型的表称为*具体查询表*。

相关概念:

- 『Import Overview』（*Data Movement Utilities Guide and Reference*）
- 『Load Overview』（*Data Movement Utilities Guide and Reference*）
- 『Moving data across platforms - file format considerations』（*Data Movement Utilities Guide and Reference*）
- 第 106 页的『用户定义的类型（UDT）』

相关任务:

- 第 112 页的『创建具体查询表』

相关参考:

- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）
- 『INSERT statement』（*SQL Reference, Volume 2*）
- 『DECLARE GLOBAL TEMPORARY TABLE statement』（*SQL Reference, Volume 2*）
- 『IMPORT Command』（*Command Reference*）
- 『LOAD Command』（*Command Reference*）

有关创建和填充表的详细信息

表包含您的所有数据。在创建表并将数据放入其中时，应考虑多种因素。

表的空间压缩的简介

将表存储在磁盘上时，有两种方法使表占用的空间较少:

- 如果列值为 NULL，则不要保留已定义的固定空间量。
- 在记录格式化和列抽取期间，如果列值易于识别或确定（如缺省值）且如果值可用于数据库管理器。

DB2® 通用数据库（DB2 UDB）具有一种可选记录格式，它允许以这种方式来节省空间。空间节省可在表级别和列级别进行。

相关概念:

- 『数据库对象的空间需求』（*《管理指南: 计划》*）
- 第 81 页的『新表的空间压缩』
- 第 149 页的『对现有表的空间压缩』

新表的空间压缩

创建表时，可使用可选 VALUE COMPRESSION 子句来指定表在使用表级别也可能是列级别的节省空间的行格式。

使用 VALUE COMPRESSION 时，不会将已指定给已定义的变长数据类型（VARCHAR、VARGRAPHICS、LONG VARCHAR、LONG VARGRAPHIC、BLOB、CLOB 和 DBCLOB）的 NULL 和零长度数据存储在磁盘上。只有与这些数据类型相关联的开销值才会占用磁盘空间。

如果使用了 VALUE COMPRESSION，则还可以使用可选 COMPRESS SYSTEM DEFAULT 选项来进一步减少磁盘空间的使用量。如果插入的或更新的值等于列的数据类型的系统缺省值，则使用的磁盘空间最少。缺省值将不会存储在磁盘上。支持 COMPRESS SYSTEM DEFAULT 的数据类型包括所有数字类型列、定长字符和定长图形字符串数据类型。这表示零和空格可以压缩。

相关参考:

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

大对象（LOB）列注意事项

在创建包含大对象列的表之前，需要作出下列决策:

1. 要记录 LOB 列的更改吗？

   若不想记录这些更改，当创建表时必须通过指定 NOT LOGGED 子句来将记录关闭:

   ```
   CREATE TABLE EMPLOYEE
      (EMPNO     CHAR(     NOT NULL PRIMARY KEY,
       FIRSTNME   VARCHAR(12)  NOT NULL,
       MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
       LASTNAME   VARCHAR(15)  NOT NULL,
       WORKDEPT   CHAR(3),
       PHONENO    CHAR(4),
       PHOTO     BLOB(10M)   NOT NULL  NOT LOGGED)
   IN RESOURCE
   ```

   若该 LOB 列大于 1 GB，则必须将记录关闭。（根据经验，可能不希望记录大于 10 MB 的 LOB 列。）与列定义中指定的其它选项一样，更改记录选项的唯一方法是重新创建该表。

   即使选择不记录更改，也会对 LOB 列*建立影子*，以允许回滚更改，不管该回滚是系统生成的错误导致的还是应用程序请求的结果。建立影子是一种恢复技术，使用该技术从不会覆盖当前存储器页的内容。即，旧的未修改的页保留为“影子”副本。当不再需要这些副本来支持事务的回滚时，就将它们废弃。

   注: 当使用 RESTORE 和 ROLLFORWARD 命令恢复数据库时，自上次备份以来“未记录的”和已写入的 LOB 数据会*被二进制零替换*。

2. 要最小化 LOB 列所需的空间吗？

   可以在 CREATE TABLE 语句中使用 COMPACT子句来使 LOB 列尽可能地小。例如:

   ```
   CREATE TABLE EMPLOYEE
      (EMPNO     CHAR(     NOT NULL PRIMARY KEY,
       FIRSTNME   VARCHAR(12)  NOT NULL,
       MIDINIT   CHAR(1)     NOT NULL WITH DEFAULT,
       LASTNAME   VARCHAR(15)  NOT NULL,
       WORKDEPT   CHAR(3),
       PHONENO    CHAR(4),
       PHOTO     BLOB(10M)   NOT NULL  NOT LOGGED  COMPACT)
   IN RESOURCE
   ```

   当追加至含有压缩的 LOB 列的表时，会存在*性能成本*，尤其是在 LOB 值的大小增加的情况下（因为必须调整存储器）。

   在不支持稀疏文件分配且将 LOB 置于 SMS 表空间中的平台上，应考虑使用 COMPACT 子句。稀疏文件分配涉及到操作系统如何使用物理磁盘空间。支持稀疏文件分配的操作系统存储 LOB 所用的物理磁盘空间比不支持稀疏文件分配的操作系统所用的空间小。无论是否支持稀疏文件分配，COMPACT 选项允许“节省”甚至更大的物理磁盘空间。因为使用 COMPACT 可以节省一些物理磁盘空间，所以若操作系统不支持稀疏文件分配，应该考虑使用 COMPACT。

   注: DB2® 系统目录使用 LOB 列，它占用的空间可能比先前版本要多。

3. 希望 LOB 列（包括在 DB2 系统目录中的那些 LOB 列）有更好的性能吗？

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。从磁盘中读取会降低涉及目录的 LOB 列的 DB2 的性能。因为一个文件系统常常有它自己存储（或高速缓存）数据的地方，所以使用 SMS 表空间或在文件容器上构建的 DMS 表空间，使得在先前引用 LOB 时避免 I/O 成为可能。

相关概念:

- 『大对象数据的空间需求』（*《管理指南: 计划》*）

相关参考:

- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）
- 『Large objects (LOBs)』（*SQL Reference, Volume 1*）

定义约束

本节讨论如何定义约束:

- 『定义唯一约束』
- 第 84 页的『定义引用约束』
- 第 87 页的『定义表检查约束』
- 第 88 页的『定义参考约束』.

有关约束的更多信息，请参阅*《管理指南: 计划》*中有关为约束强制作计划的章节，并参阅 *SQL Reference*。

定义唯一约束

*唯一约束*确保在指定键中的每个值都是唯一的。一个表可以有任意多个唯一约束，且至多将一个唯一约束定义为一个主键。

限制:

可能不能对子表定义唯一约束。

每个表只能有一个主键。

过程:

可在 CREATE TABLE 或 ALTER TABLE 语句中使用 UNIQUE 子句来定义唯一约束。唯一键可以由多个列组成。在一个表上允许多个唯一约束。

一旦建立了该约束，当 INSERT 或 UPDATE 语句修改表中的数据时，数据库管理器会自动实现该唯一约束。唯一约束通过唯一索引来实现。

当在 ALTER TABLE 语句中定义唯一约束且在该唯一键的同一组列上存在一个索引时，该索引就成为唯一索引且被该约束使用。

可以提取任何一个唯一约束，并将它用作*主键*。主键可以用作引用约束（以及其它唯一约束）中的父键。可在 CREATE TABLE 或 ALTER TABLE 语句中使用 PRIMARY KEY 子句来定义主键。主键可以由多个列组成。

主索引强制该主键的值为唯一的。当使用主键创建表时，数据库管理器会在该键上创建一个主索引。

用作唯一约束的索引的某些性能提示包括:

- 当初次装入带索引的空表时，LOAD 将提供比 IMPORT 更好的性能。不管是使用 LOAD 的 INSERT方式还是 REPLACE 方式，这种情况都一样。
- 当把大量数据追加到一个带索引的现有表中（使用 IMPORT INSERT 或 LOAD INSERT）时，LOAD 的性能只比 IMPORT 的稍好一点。
- 若要使用 IMPORT 命令来进行大量数据的初始装入，则要在导入或装入数据之后创建唯一键。这样避免了当装入该表时维护索引的额外开销。它还使索引使用最少量的存储器。
- 若正在以 REPLACE 方式使用装入实用程序，则在装入数据之前创建唯一键。在这种情况下，在装入期间创建索引比在装入之后使用 CREATE INDEX 语句更有效。

相关概念:

- 『Keys』（*SQL Reference, Volume 1*）
- 『Constraints』（*SQL Reference, Volume 1*）

相关参考:

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

定义引用约束

引用完整性是通过将引用约束添加到表和列定义来实现的。一旦对数据库管理器定义了引用约束，就会针对定义的约束检查对表和列中数据的更改。请求操作的完成取决于约束检查的结果。

过程:

引用约束是使用 CREATE TABLE 或 ALTER TABLE 语句中的 FOREIGN KEY 子句和 REFERENCES 子句建立的。创建引用约束之前，应考虑引用约束对类型表以及对作为类型表的父表的影响。

外键的标识在一个表的行内或两个表的行之间的值上施加约束。数据库管理器检查表定义中指定的约束，并相应地维持关系。目标是无论何时一个数据库对象引用另一个数据库对象都要维持完整性。

例如，主键和外键各有一个部门号列。对于 EMPLOYEE 表，该列名为 WORKDEPT，而对于 DEPARTMENT 表，该列名为 DEPTNO。这两个表之间的关系由下列约束定义:

- 对于 EMPLOYEE 表中的每个职员只有一个部门号，且该编号存在于 DEPARTMENT 表中。
- EMPLOYEE 表中的每一行都只与 DEPARTMENT 表中的一行相关。这两个表之间存在唯一的关系。
- 在 EMPLOYEE 表中具有 WORKDEPT 的非空值的每一行只与 DEPARTMENT 表的 DEPTNO 列中的一行相关。
- DEPARTMENT 表是父表，而 EMPLOYEE 表是从属表。

定义父表 DEPARTMENT 的 SQL 语句如下所示:

```
CREATE TABLE DEPARTMENT
      (DEPTNO    CHAR(3)     NOT NULL,
       DEPTNAME  VARCHAR(29) NOT NULL,
```

```
      MGRNO     CHAR(6),
      ADMRDEPT  CHAR(3)     NOT NULL,
      LOCATION  CHAR(16),
         PRIMARY KEY (DEPTNO))
   IN RESOURCE
```

定义从属表 EMPLOYEE 的 SQL 语句如下所示:

```
CREATE TABLE EMPLOYEE
      (EMPNO     CHAR(     NOT NULL PRIMARY KEY,
       FIRSTNME   VARCHAR(12)  NOT NULL,
       LASTNAME   VARCHAR(15)  NOT NULL,
       WORKDEPT   CHAR(3),
       PHONENO    CHAR(4),
       PHOTO     BLOB(10m)   NOT NULL,
          FOREIGN KEY DEPT (WORKDEPT)
          REFERENCES DEPARTMENT ON DELETE NO ACTION)
   IN RESOURCE
```

通过将 DEPTNO 列指定为 DEPARTMENT 表的主键，而将 WORKDEPT 指定为 EMPLOYEE 表的外键，就对 WORKDEPT 值定义了引用约束。此约束实现这两个表的值之间的引用完整性。在这种情况下，添加至 EMPLOYEE 表的任何职员必须具有一个可以在 DEPARTMENT 表中找到的部门号。

职员表中的引用约束的删除规则为 NO ACTION，这表示若一个部门中有任何职员，则不能将该部门从 DEPARTMENT 表中删除。

虽然先前的示例使用 CREATE TABLE 语句来添加引用约束，但是也可以使用 ALTER TABLE 语句。

另一个示例: 使用与先前示例所用的相同表定义。另外，在 EMPLOYEE 表之前创建 DEPARTMENT 表。每个部门有一个经理，且该经理在 EMPLOYEE 表中列出。DEPARTMENT 表的 MGRNO 实际是 EMPLOYEE 表的外键。因为此引用循环，此约束存在一个小小的问题。可在以后添加外键。还可以使用 CREATE SCHEMA 语句来同时创建 EMPLOYEE 和 DEPARTMENT 表。

相关概念:

- 第 85 页的『FOREIGN KEY 子句』
- 第 86 页的『REFERENCES 子句』

相关任务:

- 第 157 页的『添加外键』

相关参考:

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

FOREIGN KEY 子句

外键在同一个或另一个表中引用主键或唯一键。外键的指定指示将根据指定的引用约束来维持该引用完整性。可在 CREATE TABLE 或 ALTER TABLE 语句中使用 FOREIGN KEY 子句来定义外键。

外键中的列数必须等于父表的对应主约束或唯一约束（称为父键）中的列数。另外，键列定义的对应部分必须具有相同的数据类型和长度。可以赋予外键一个*约束名*。若未赋予一个名称，则会自动赋予一个。为便于使用，建议赋予一个*约束名*，而不要使用系统生成的名称。

若一个组合的外键每一列的值等于父键对应列的值，则该外键的值与该父键的值匹配。包含空值的外键不能与父键的值匹配，因为定义的父键不能有空值。但是，一个空的外键值始终是有效的，无论它的任何一个非空部分的值如何。

下列规则适用于外键定义:

- 一个表可以有多个外键
- 若任何部分都可为空，则该外键可为空
- 若任何部分为空，则该外键值为空

相关任务:

- 第 83 页的『定义唯一约束』
- 第 84 页的『定义引用约束』

相关参考:

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

REFERENCES 子句

REFERENCES 子句标识一个关系中的父表，并定义需要的约束。可以将它包括在列定义中，或作为一个单独的子句与 FOREIGN KEY 子句一起包括在 CREATE TABLE 或 ALTER TABLE 语句中。

若将 REFERENCES 子句指定为列约束，则隐式列列表由列出的一个或多个列名组成。记住，多个列可以有独立的 REFERENCES 子句，而单个列可以有多个 REFERENCES 子句。

REFERENCES 子句中包括的是删除规则。在我们的示例中，使用 ON DELETE NO ACTION 规则，它指示若有职员分配给部门，则不能删除该部门。其它删除规则包括 ON DELETE CASCADE、ON DELETE SET NULL 和 ON DELETE RESTRICT。

相关概念:

- 第 85 页的『FOREIGN KEY 子句』

相关参考:

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

实用程序操作的隐含意义

装入实用程序将关闭对自引用和从属表的约束检查，并将这些表置于检查暂挂状态。在装入实用程序完成之后，将需要对关闭了其约束检查的所有表打开约束检查。例如，若 DEPARTMENT 和 EMPLOYEE 表是唯一处于检查暂挂状态的表，可执行下列语句:

```
SET INTEGRITY FOR DEPARTMENT, EMPLOYEE IMMEDIATE CHECKED
```

引用约束以下列方式影响导入实用程序:

- 若该对象表有该表以外的其它对象从属于它，则不允许 REPLACE 和 REPLACE CREATE 函数。

  要使用这些函数，首先要删除该表为父表的所有外键。当导入完成时，使用 ALTER TABLE 语句重新创建这些外键。
- 导入带自引用约束的表的成功与否取决于这些行导入的顺序。

相关概念:

- 『Import Overview』（*Data Movement Utilities Guide and Reference*）
- 『Load Overview』（*Data Movement Utilities Guide and Reference*）
- 『Checking for integrity violations』（*Data Movement Utilities Guide and Reference*）

相关参考:

- 『SET INTEGRITY statement』（*SQL Reference, Volume 2*）

定义表检查约束

表检查约束指定搜索条件，对于定义了表检查约束的表的每一行都实现该搜索条件。一旦对数据库管理器定义了表检查约束，就会针对定义的约束检查对表中数据的插入或更新。请求操作的完成取决于约束检查的结果。

过程:

在创建或改变表时，通过将检查约束定义与表关联来对该表创建表检查约束。当 INSERT 或 UPDATE 语句修改该表中的数据时，就自动激活此约束。表检查约束对 DELETE 或 SELECT 语句没有影响。检查约束可以与类型表相关。

约束名不能与在同一个 CREATE TABLE 语句内指定的任何其它约束相同。若不指定约束名，系统会为该约束生成 18 个字符的唯一标识。

表检查约束用于实现键唯一性或引用完整性约束所未涵盖的数据完整性规则。在某些情况中，表检查约束可以用于实现域检查。在 CREATE TABLE 语句上发出的下列约束确保每个活动的开始日期不在同一个活动的结束日期之后:

```
CREATE TABLE EMP_ACT
   (EMPNO       CHAR(6)      NOT NULL,
    PROJNO     CHAR(6)      NOT NULL,
    ACTNO      SMALLINT     NOT NULL,
    EMPTIME    DECIMAL(5,2),
    EMSTDATE   DATE,
    EMENDATE   DATE,
    CONSTRAINT ACTDATES CHECK(EMSTDATE <= EMENDATE) )
IN RESOURCE
```

虽然上一个示例使用 CREATE TABLE 语句来添加表检查约束，但是也可以使用 ALTER TABLE 语句。

相关概念:

- 『Constraints』（*SQL Reference, Volume 1*）

相关任务:

- 第 158 页的『添加表检查约束』

相关参考:

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

定义参考约束

*参考约束*是一个规则，可由 SQL 编译器使用，但数据库管理器不会强制使用它。SQL 编译器包括一个重写查询阶段，它将 SQL 语句变换为可能是优化的格式并改进所需数据的存取路径。约束的目的不在于让数据库管理器对数据执行附加验证，而是为了改进查询性能。

过程:

使用 CREATE TABLE 或 ALTER TABLE 语句定义参考约束。在这些语句中添加引用完整性或检查约束。然后，将约束属性与它们相关联，指定是否想要数据库管理器强制使用约束；并指定是否想将约束用于查询优化。

相关概念:

- 『Constraints』（*SQL Reference, Volume 1*）
- 『SQL 编译器进程』（*《管理指南: 性能》*）
- 『查询重写方法和和示例』（*《管理指南: 性能》*）

相关参考:

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

对新表定义生成列

生成列在基本表中定义，在这些列中，存储的值是使用表达式计算得出的，而不是通过插入或更新操作指定。

过程:

当创建已知始终要使用特定表达式或谓词的表时，可对该表添加一个或多个生成列。通过使用生成列，有机会在查询表数据时改进性能。

例如，当性能很重要时，以下两种表达式求值方式成本很高:

1. 必须在查询期间进行许多次表达式求值。
2. 计算很复杂。

为了改进查询的性能，可定义一个附加列，它将包含该表达式的结果。然后，当发出包括同一表达式的查询时，可直接使用生成列，或者，优化器的查询重写组件可用生成列替换该表达式。

也有可能对生成列创建非唯一索引。

当查询涉及连接两个或多个表中的数据时，添加生成列允许优化器选择可能更好的连接策略。

以下是在 CREATE TABLE 语句上定义生成列的一个示例:

```
CREATE TABLE t1 (c1 INT,
                 c2 DOUBLE,
                 c3 DOUBLE GENERATED ALWAYS AS (c1 + c2)
                 c4 GENERATED ALWAYS AS
                    (CASE WHEN c1 > c2 THEN 1 ELSE NULL END))
```

在创建此表之后，可以使用生成列来创建索引。例如，

```
CREATE INDEX i1 ON t1(c4)
```

查询可以利用生成列。例如，

```
SELECT COUNT(*) FROM t1 WHERE c1 > c2
```

可以写成

```
SELECT COUNT(*) FROM t1 WHERE c4 IS NOT NULL
```

另一个示例:

```
SELECT c1 + c2 FROM t1 WHERE (c1 + c2) * c1 > 100
```

可以写成

```
SELECT c3 FROM t1 WHERE c3 * c1 > 100
```

将使用生成列来改进查询的性能。结果是，可能在创建和填充表之后添加生成列。

相关任务:

- 第 161 页的『对现有表定义生成列』

相关参考:

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

创建用户定义的临时表

当您编写应用程序以使用数据库中的数据时，需要用户定义的临时表。对数据进行处理产生的结果需要临时存储在表中。

先决条件:

在创建用户定义的临时表之前必须存在用户临时表空间。

限制:

此表的描述并不出现在系统目录中，使其对于其它应用程序而言不是持久的，也不能与其它应用程序共享此表。

当使用此表的应用程序终止或与数据库断开连接时，此表中的数据被删除，此表被隐式删除。

用户定义的临时表不支持:

- LOB 类型的列（或基于 LOB 的单值类型列）
- 用户定义的类型列

- LONG VARCHAR 列
- DATALINK 列

过程:

可使用 DECLARE GLOBAL TEMPORARY TABLE 语句来定义临时表。此语句需在应用程序中使用。只有在应用程序与数据库断开连接之前，用户定义的临时表才是持续的。

下面是定义临时表的一个示例:

```
DECLARE GLOBAL TEMPORARY TABLE gbl_temp
      LIKE empltabl
      ON COMMIT DELETE ROWS
      NOT LOGGED
      IN usr_tbsp
```

此语句创建一个名为 gbl_temp 的用户临时表。定义此用户临时表所使用的列的名称和描述与 empltabl 的列的名称和描述完全相同。隐式定义只包括列名、数据类型、可空性特征和列缺省值属性。未定义所有其它列属性，包括唯一约束、外键约束、触发器和索引。执行 COMMIT 操作时，若未对该表打开 WITH HOLD 游标，则该表中的所有数据都被删除。不记录对用户临时表所作的更改。用户临时表被放在指定的用户临时表空间中。此表空间必须存在，否则此表的声明将失败。

如果创建此表时指定了 ROLLBACK 或 ROLLBACK TO SAVEPOINT，则可以指定删除表中的所有行（DELETE ROWS，这是缺省情况），或者可以指定保留表中的各行（PRESERVE ROWS）。

相关任务:

- 第 71 页的『创建用户临时表空间』

相关参考:

- 『ROLLBACK statement』（*SQL Reference, Volume 2*）
- 『SAVEPOINT statement』（*SQL Reference, Volume 2*）
- 『DECLARE GLOBAL TEMPORARY TABLE statement』（*SQL Reference, Volume 2*）

对新表定义标识列

标识列 为 DB2 提供一种方法，可自动为添加至表的每一行生成唯一数字值。当创建一个表时，若您知道需要唯一标识将添加至该表的每一行，则可向该表添加一个标识列。要保证为添加至表的每一行提供唯一数字值，您应在标识列定义唯一索引，或将其声明为主键。

限制:

在创建之后，便不能将表描述改变为包括标识列。

如果将行插入到指定了显式标识列值的表中，则不会更新在内部生成的下一个值，并且可能会与该表中的现有值发生冲突。如果主键或在标识列定义的唯一索引在标识列强制执行值的唯一性，则重复值将生成一条错误消息。

过程:

CREATE TABLE 语句上的 AS IDENTITY 子句允许说明标识列。

以下是在 CREATE TABLE 语句上定义标识列的一个示例:

```
CREATE TABLE table (col1 INT,
                    col2 DOUBLE,
                    col3 INT NOT NULL GENERATED ALWAYS AS IDENTITY
                                      (START WITH 100, INCREMENT BY 5))
```

在此示例中，第三个列是标识列。还可以指定该列中用来在添加行时唯一标识每一行的值。在输入的第一行的列中具有值“100”；添加到该表中的每个后续行都具有相关联的值，这些值将依次增加五。

某些附加示例使用一个标识列，该标识列为订单号、职员编号、股票代码或者事故编号。标识列的值可以由 DB2: ALWAYS 或 BY DEFAULT 生成。

将对定义为 GENERATED ALWAYS 的标识列给予总是由 DB2 生成的值。不允许应用程序提供显式的值。定义成 GENERATED BY DEFAULT 的标识列使应用程序能够显式地为标识列提供值。如果应用程序不提供值，则 DB2 将生成一个值。因为由应用程序控制该值，所以 DB2 不能保证该值的唯一性。GENERATED BY DEFAULT 子句用于数据传播，其目的是复制现有表的内容；或者用于卸装和重新装入表。

相关概念:

- 第 92 页的『比较 IDENTITY 列和序列』

相关参考:

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

创建序列

*序列*是一个数据库对象，它允许自动生成值。序列特别适合于生成唯一键值这一任务。应用程序可以使用序列来避免由于在数据库外生成唯一计数器而引起可能的并发性和性能问题。

限制:

与标识列属性不同，未使序列与特定表列相关，也未将它绑定至唯一表列，只是仅可通过该表列存取。

若将包含一个或多个序列的数据库恢复至前一时间点，则可能会导致对某些序列生成重复值。要避免可能的重复值，不应该将具有序列的数据库恢复至前一时间点。

在可使用 NEXTVAL 或 PREVVAL 表达式的位置有几个限制。

过程:

可以创建或改变序列，以便序列以下列其中一种方式来生成值:

- 没有界限单调地递增或递减
- 单调地递增或递减至用户定义的限制并停止
- 单调地递增或递减至用户定义的限制并循环回至起点，然后再开始。

以下是创建序列对象的一个示例:

```
CREATE SEQUENCE order_seq
   START WITH 1
   INCREMENT BY 1
   NOMAXVALUE
   NOCYCLE
   CACHE 24
```

在此示例中，序列称为 order_seq。它将从 1 开始，并以 1 为增量增加，且没有上限。没有原因循环回至起点并从 1 重新开始，因为没有指定上限。与 CACHE 参数关联的数指定了序列值的最大数目，数据库管理器预分配此数目并将它保存在内存中。

生成的序列号具有下列属性:

- 值可以是小数位为零的任何精确数字数据类型。这样的数据类型包括: SMALLINT、BIGINT、INTEGER 和 DECIMAL。
- 连续值之间的差别可以为任何指定的整数增量。缺省递增值是 1。
- 计数器值是可恢复的。当需要恢复时，从日志中重建计数器值。
- 可以高速缓存值以改善性能。在高速缓存中预分配并存储值，可以在为序列生成值时减少对日志的同步 I/O。万一发生系统故障，则永远不会使用尚未落实的所有高速缓存值并认为它们已丢失。为 CACHE 指定的值是可能丢失的序列值的最大数目。

有两种表达式与序列一起使用。

对于当前应用程序进程中的先前语句，PREVVAL 表达式对指定序列返回最新生成的值。

NEXTVAL 表达式对指定序列返回下一个值。当 NEXTVAL 表达式指定序列的名称时，生成一个新的序列号。但是，若在查询中 NEXTVAL 表达式的多个实例指定同一序列名，则对于结果的每一行，序列计数器仅递增一次，且 NEXTVAL 的所有实例对结果的某一行返回同一个值。

通过对第一行使用 NEXTVAL 表达式引用序列号，而对任何附加行使用 PREVVAL 表达式引用序列号，相同序列号可以在两个不同的表中用作唯一键值。

例如:

```
INSERT INTO order (orderno, custno)
   VALUES (NEXTVAL FOR order_seq, 123456);
INSERT INTO line_item (orderno, partno, quantity)
   VALUES (PREVVAL FOR order_seq, 987654, 1)
```

相关概念:

- 第 92 页的『比较 IDENTITY 列和序列』

相关参考:

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

比较 IDENTITY 列和序列

虽然在 IDENTITY 列和序列之间存在相似之处，但是也存在差别。当设计数据库和应用程序时可以使用其各自的特征。

标识列具有下列特征:

- 仅当创建了表时，才可以将标识列定义为表的一部分。一旦创建了表，您就不能改变它来添加一个标识列。（但是，可以改变现有的标识列特征。）
- 标识列自动为单个表生成值。
- 当将标识列定义为 GENERATED ALWAYS 时，始终由数据库管理器生成所用的值。在修改表的内容期间，不允许应用程序来提供它们自己的值。

序列对象具有下列特征:

- 序列对象是未与任何一个表关联的数据库对象。
- 序列对象生成可在任何 SQL 语句中使用的顺序值。
- 由于任何应用程序可以使用序列对象，所以有两种表达式可用来控制如何检索指定序列中的下一个值和在正在执行的语句之前生成的值。对于当前会话中的先前语句，PREVVAL 表达式对指定序列返回最新生成的值。NEXTVAL 表达式对指定序列返回下一个值。使用表达式允许在几个表中的几个 SQL 语句中使用同一值。

虽然这些并非这两项的所有特征，但是这些特征将帮助您根据数据库设计和使用数据库的应用程序来确定使用哪一项。

相关任务:

- 第 88 页的『对新表定义生成列』
- 第 91 页的『创建序列』
- 第 161 页的『对现有表定义生成列』

范围群集表示例

下面是两个很简单的示例，它们演示了创建范围群集表的方法。这些示例说明了可以如何使用单列或多列作为一个表的键。另外，它们还说明了如何创建一个允许数据溢出的表和不允许数据溢出的表。

第一个示例说明了一个范围群集表，它使用 STUDENT_ID 来查找学生。每一条学生记录都包括下列信息:

- 学校标识
- 程序标识
- 学生编号
- 学生标识
- 学生的名
- 学生的姓
- 学生的学年平均成绩（GPA）

在本例中，我们想只根据 STUDENT_ID 来处理学生记录。STUDENT_ID 将用来添加、更新和删除学生记录。

注: 其它索引可以另外单独添加。但是，对于此示例，创建表时就定义了表的组织以及如何存取表的数据。

以下是此表所需要的语法:

```
CREATE TABLE STUDENTS
   (SCHOOL_ID      INT NOT NULL,
    PROGRAM_ID       INT NOT NULL,
    STUDENT_NUM    INT NOT NULL,
    STUDENT_ID     INT NOT NULL,
    FIRST_NAME     CHAR(30),
    LAST_NAME      CHAR(30),
    GPA            FLOAT)
   ORGANIZE BY KEY SEQUENCE
   (STUDENT_ID     STARTING FROM 1 ENDING AT 1000000)
    ALLOW OVERFLOW
;
```

每条记录的大小是所有列的总计。在本例中，包括 10 个字节的头 + 4 + 4 + 4 + 4 + 30 + 30 + 8 + 3 （用于可空列），总共为 97 个字节。页大小为 4 KB（或者 4096 个字节），除去开销之后还有 4038 个字节，具有足够的空间用于每页 42 条记录。如果允许具有一百万条学生记录，则将需要 23809.5 页（一百万条记录除以每页 42 条记录）。实际上将需要 23810 页（舍入）。添加四页用于表开销，添加三页用于数据块映射。因此，需要预分配 23817 页（每页大小为 4 KB）。（数据块映射假定使用单个容器来保存此表。因此，每个容器应该具有三页。）

在第二个示例中（此示例是第一个示例的变体）假定了学校管理局这样一个概念。在学校管理局中有 200 所学校，每所学校有 20 间教室，每间教室可容纳 35 个学生。此学校管理局最多可以容纳 140,000 名学生。

在本例中，我们想根据三个因素来处理学生记录：SCHOOL_ID、CLASS_ID 和 STUDENT_NUM 值。这三列中的每一列都将具有唯一值，并将这三列一起用来添加、更新和删除学生记录。

注: 与前一个示例一样，可以另外单独添加其它索引。

以下是此表所需要的语法:

```
CREATE TABLE STUDENTS
   (SCHOOL_ID      INT NOT NULL,
    CLASS_ID       INT NOT NULL,
    STUDENT_NUM    INT NOT NULL,
    STUDENT_ID     INT NOT NULL,
    FIRST_NAME     CHAR(30),
    LAST_NAME      CHAR(30),
    GPA            FLOAT)
   ORGANIZE BY KEY SEQUENCE
   (SCHOOL_ID      STARTING FROM 1 ENDING AT 200,
    CLASS_ID       STARTING FROM 1 ENDING AT 20,
    STUDENT_NUM    STARTING FROM 1 ENDING AT 35)
    DISALLOW OVERFLOW
;
```

在本例中，不允许溢出。这样做是有道理的，因为学校管理局的政策可能会限制每个班级所允许的学生人数。在本示例中，每个班级最多只能有 35 名学生。如果将此因素与教室数和学校数实际存在的限制联系起来考虑，就不难理解为何不允许学校管理局拥有的学生数发生溢出。

一些学校的教室数目可能会发生一些变化。如果是这种情况，当（使用 CLASS_ID）定义教室数的范围时，上限应该是将所有学校考虑在内时所得到的最大教室数。这可能意味着某些较小的学校（与最大的学校比较起来教室数更少的那些学校）将具有用于从未使用的学生记录的空间（除非，例如，为学校添加移动教室）。

与前一个示例中一样，使用相同的 4 KB 页大小和相同的学生记录大小，我们知道每一页可以具有 42 条记录。因此，140,000 条学生记录就将需要 3333.3 页，进行舍入之后就是 3334 页。另外，表信息还需要两页，数据块映射需要三页。因此，需要预分配 3339 页（每页大小为 4 KB）。

相关参考:

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

SQL 编译器如何使用范围群集表

SQL 编译器处理范围群集表（RCT）的方式与处理具有辅助 B+ 树索引的常规表的方式相似。RCT 并不是通过 B+ 树索引来确定记录的位置或记录标识（RID），而是使用涉及到范围定义中的记录键值和算法的功能查找。这与具有索引的情况相似，因为可以使用键值来快速获得 RID。

当确定必需数据的最佳存取路径时，SQL 编译器使用有关表的统计信息。索引统计信息是在发出 RUNSTATS 命令时在扫描表期间收集的。对于 RCT，将表模型化为常规表，将索引模型化为基于函数的索引。

当创建允许溢出的范围群集表时，不能保证表中的记录顺序。

相关概念:

- 『范围群集表』（*《管理指南: 计划》*）
- 第 95 页的『关于使用范围群集表的准则』

关于使用范围群集表的准则

使用 DB2® 通用数据库和范围群集表（RCT）时，应注意下列准则:

- 当定义键值的范围时，最小值是可选的；如果没有指定最小值，则缺省值是一（1）。允许最小值和最大值为负值。当使用负值时，必须明确声明最小值。例如，ORGANIZE BY KEY SEQUENCE (F1 STARTING FROM -100 ENDING AT -10)
- 不允许对用来定义范围群集表的相同键值创建常规索引。
- 某些 ALTER TABLE 选项不可对范围群集表使用。此时选项不会影响表的物理结构，允许使用该选项。
- 因为创建范围群集表的过程中就预分配了必需的磁盘空间，所以该空间必须是可用的，否则创建表就会失败。

相关概念:

- 『范围群集表』（*《管理指南: 计划》*）
- 第 93 页的『范围群集表示例』

对表定义维

*维*是表的群集键。可对一个表选择一个或多个维。一个表有多个维时，会认为该表是多维群集表。这种表是使用带有 ORGANIZE BY DIMENSIONS 子句的 CREATE TABLE 语句创建的。

限制:

在 ORGANIZE BY [DIMENSIONS] 子句中使用的一组列必须遵循 CREATE INDEX 语句的规则。这些列被视为用来维护数据在存储器中的物理顺序的键。

过程:

每个维都是在 CREATE TABLE 语句中使用 ORGANIZE BY [DIMENSIONS] 子句和一列或多列指定的。在维列表中使用圆括号来将与单个维相关联列组合到一起。

同时根据一个维或多个维（针对指定的所有维）对数据进行物理群集。数据是根据维行按数据块或“块”组织的。使用维谓词查询数据时，可将扫描限制为仅包含涉及的维值的表的数据块。另外，由于扩展数据块是磁盘上的一组顺序页，可以对这些扫描执行非常有效的预取。

虽然随着时间的推移，当表中的空间填满时，具有单个群集索引的表可能变成非群集的，但带有多个维的表能够自动且持续地维护所有维的群集。因此，不需要重组表就可以复原数据的顺序。

会自动地为指定的每个维创建维块索引。使用维块索引来根据维存取数据。维块索引指向数据块而不是各个行，因此，维块索引远远小于常规索引。可使用维块索引非常快速地只存取包含特定维值的表的数据块。

自动创建组合块索引以包含所有维键列。使用组合块索引在插入和更新活动期间维护数据群集。在查询处理期间使用组合块索引来存取具有特定维值的表中的数据。

注: 键部分在组合块索引中的顺序可能会影响它对查询处理的用法或适用性。键部分的顺序是创建 MDC 表时使用的整个 ORGANIZE BY [DIMENSIONS] 子句中的列的顺序确定的。例如，如果表是使用下列语句创建的:

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

则将对列 (c1,c4,c3,c2) 创建组合块索引。虽然 c1 在维子句中指定了两次，但它仅作为组合块索引的键部分使用了一次，并且是以第一次发现它的顺序使用的。组合块索引中的键部分的顺序对于插入处理没有影响，但对于查询处理可能是有影响的。因此，如果更期望组合块索引的列顺序为 (c1,c2,c3,c4)，应使用下列命令创建该表:

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

在指定维已包含组合块索引将具有的所有列的情况下，不会创建组合块索引。例如，不会为下表创建组合块索引:

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

相关概念:

- 『多维群集表』（*《管理指南: 计划》*）

相关参考:

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

创建层次结构表或类型表

层次结构表是与类型表层次结构的实现相关联的一个表。它与层次结构的根表同时创建。

作为创建结构化类型层次结构的一部分，将创建类型表。可以使用类型表来存储使用 CREATE TYPE 语句定义其特征的对象的实例。

先决条件:

将对其创建层次结构表或类型表的类型必须存在。

过程:

可以使用 CREATE TABLE 语句的变体来创建层次结构表或类型表。

相关概念:

- 『Typed tables』（*Application Development Guide: Programming Server Applications*）

相关任务:

- 第 97 页的『填充类型表』
- 第 111 页的『创建带类型视图』
- 『Creating a structured type hierarchy』（*Application Development Guide: Programming Server Applications*）
- 『Dropping typed tables』（*Application Development Guide: Programming Server Applications*）
- 『Creating typed tables』（*Application Development Guide: Programming Server Applications*）

相关参考:

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

填充类型表

在建立结构化类型层次结构时，将创建类型表。使用类型表来存储使用 CREATE TYPE 语句定义其特征的对象的实例。创建类型表之后，将需要在类型表中放置数据。

先决条件:

类型表必须存在。

过程:

在创建结构化类型并接着创建相对应的表和子表之后，可以填充类型表。

相关概念:

- 『Substitutability in typed tables』（*Application Development Guide: Programming Server Applications*）
- 『Typed tables』（*Application Development Guide: Programming Server Applications*）

相关任务:

- 第 97 页的『创建层次结构表或类型表』
- 『Storing objects in typed table rows』（*Application Development Guide: Programming Server Applications*）
- 『Dropping typed tables』（*Application Development Guide: Programming Server Applications*）
- 『Creating typed tables』（*Application Development Guide: Programming Server Applications*）

相关参考:

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

在多个表空间中创建表

表数据、表的索引以及与该表相关联的任何长整型列数据可以存储在同一表空间中。也可以将索引放在一个单独的表空间中，并将任何长整型列数据放在一个单独的表空间中，以便与用于存放其余表数据的表空间分隔开。

先决条件:

在运行 CREATE TABLE 语句之前，所有表空间都必须存在。

限制:

只能使用 DMS 表空间来分隔表的各个部分。

过程:

要使用“控制中心”来在多个表空间中创建一个表:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 右键单击**表**文件夹，并从弹出菜单中选择**创建**。
3. 输入表名并单击**下一步**。
4. 为该表选择列。
5. 在**表空间**页上，单击**使用单独的索引空间**和**使用单独的长型空间**，指定信息，并单击**完成**。

要使用命令行来在多个表空间中创建一个表，输入:

```
CREATE TABLE <name>
   (<column_name>  <data_type>  <null_attribute>)
    IN <table_space_name>
    INDEX IN <index_space_name>
    LONG  IN <long_space_name>
```

以下示例显示如何创建 EMP_PHOTO 表来将该表的不同部分存储在不同的表空间中:

```
CREATE TABLE EMP_PHOTO
   (EMPNO        CHAR(6)      NOT NULL,
    PHOTO_FORMAT VARCHAR(10)  NOT NULL,
```

```
    PICTURE      BLOB(100K) )
IN RESOURCE
INDEX IN RESOURCE_INDEXES
LONG  IN RESOURCE_PHOTO
```

此示例将使 EMP_PHOTO 数据按如下所示存储:

- 为 EMP_PHOTO 表创建的索引将存储在 RESOURCES_INDEXES 表空间中
- PICTURE 列的数据将存储在 RESOURCE_PHOTO 表空间中
- EMPNO 和 PHOTO_FORMAT 列的数据将存储在 RESOURCE 表空间中

相关参考:

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

在分区数据库中创建表

在分区数据库中跨越几个分区创建一个表在性能上有几个优点。与检索数据相关联的工作可分成几部分在各个数据库分区中进行。

先决条件:

在创建将以物理方式划分或分区的一个表之前，需要考虑下列事宜:

- 表空间可以横跨多个数据库分区。它们跨越的分区数取决于数据库分区组中的分区数。
- 可以通过如下方法来并置表：将表置于同一个表空间中，或置于另一个表空间中，该表空间与第一个表空间一起，与同一个数据库分区组相关联。

限制:

必须小心地选择适当的分区键，因为*以后再也不能更改它*。再者，必须将任何唯一索引（因此也是唯一键或主键）定义为分区键的一个超集。即，若定义了分区键，则唯一键和主键必须包括所有与分区键相同的列（它们可能有多列）。

表的一个分区大小不能超过 64 GB 和可用的磁盘空间中较小的那一个。（假设表空间具有 4 KB 的页大小。）该表的最大大小可以是 64 GB（或可用磁盘空间）乘以数据库分区数之积。若该表空间的页大小为 8 KB，则该表最大的大小可以为 128 GB（或可用的磁盘空间）乘以数据库分区数之积。若该表空间的页大小为 16 KB，则该表的最大大小可为 256 GB（或可用的磁盘空间）乘以数据库分区数之积。若该表空间的页大小为 32 KB，则该表的最大大小可为 512 GB（或可用的磁盘空间）乘以数据库分区数之积。

过程:

在创建表时，指定创建的表将成为若干数据库分区的一部分。当在分区数据库环境中创建表时，有一个附加选项：*分区键*。分区键是作为一个表定义的一部分的键。它确定用于存储每行数据的分区。

若不显式指定分区键，会使用下列缺省值。*确保缺省分区键是适当的*。

- 若在 CREATE TABLE 语句中指定了主键，则该主键的第一列会用作分区键。
- 若不存在主键，则使用非长型字段的第一列。
- 若没有列满足缺省分区键的需求，则会不带关键字创建该表（这只在单一分区数据库分区组中允许）。

以下是一个示例:

```
CREATE TABLE MIXREC (MIX_CNTL INTEGER NOT NULL,
                     MIX_DESC CHAR(20) NOT NULL,
                     MIX_CHR  CHAR(9) NOT NULL,
                     MIX_INT INTEGER NOT NULL,
                     MIX_INTS SMALLINT NOT NULL,
                     MIX_DEC DECIMAL NOT NULL,
                     MIX_FLT FLOAT NOT NULL,
                     MIX_DATE DATE NOT NULL,
                     MIX_TIME TIME NOT NULL,
                     MIX_TMSTMP TIMESTAMP NOT NULL)
                     IN MIXTS12
                     PARTITIONING KEY (MIX_INT) USING HASHING
```

在上一个示例中，表空间是 MIXTS12，而分区键是 MIX_INT。若未显式指定分区键，则它是 MIX_CNTL。（若未指定主键且未定义分区键，则分区键是该列表中的第一个非长型字段的列。）

表的一行和有关该行的所有信息始终驻留在同一个数据库分区上。

相关概念:

- 『数据库分区组』（*《管理指南: 计划》*）
- 『数据库分区组设计』（*《管理指南: 计划》*）
- 『表并置』（*《管理指南: 计划》*）

相关参考:

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

创建触发器

触发器定义一组操作，这组操作与用于指定基本表或类型表的 INSERT、UPDATE 或 DELETE 子句一起执行或由这些子句触发。触发器的某些用途如下:

- 验证输入数据
- 为新插入的行生成值
- 为交叉引用而从其它表中进行读取
- 为审计跟踪而向其它表写入

可使用触发器支持一般形式的完整性或商业规则。例如，在接受订单或更新总结数据表之前，触发器可以检查客户的信用额度。

使用触发器的优点有:

- 更快地开发应用程序: 因为触发器存储在数据库中，所以不必为它在每个应用程序中执行的操作进行编码。
- 更容易维护: 一旦定义了一个触发器，则当存取创建它所基于的表时，会自动调用它。
- 商业规则的全局实现: 若业务策略改变，只需更改触发器而不必更改每个应用程序。

限制:

不能使用具有昵称的触发器。

若触发器是一个 BEFORE 触发器，则由触发操作指定的列名可能不是生成列而是标识列。即，生成的标识值对 BEFORE 触发器可见。

当创建原子触发器时，必须认真对待语句结束字符。缺省情况下，数据库管理器将“;”当作是语句结束标记。您应该在脚本中手工编辑语句结束字符来创建原子触发器，以便使用一个非“;”的字符。例如，可以用另一个特殊字符（如“#”）替换“;”。

然后，您必须:

- 使用在“命令编辑器”（此编辑器替换“命令中心”）中选择的脚本选项卡，通过工具 -> 工具设置菜单更改定界符，然后运行脚本；或者，
- 从“命令行处理器”，使用:

  ```
  db2 -td <delimiter> -vf <script>
  ```

  其中 delimiter 是备用语句结束字符，而 <script> 是已修改的在其中使用新定界符的脚本。

过程:

要使用“控制中心”来创建触发器:

1. 展开对象树，直到您看到**触发器**文件夹为止。
2. 右键单击**触发器**文件夹，并从弹出菜单中选择**创建**。
3. 指定触发器的信息。
4. 指定要让触发器调用的操作，并单击**确定**。

要使用命令行来创建触发器，输入:

```
CREATE TRIGGER <name>
   <action> ON <table_name>
   <operation>
   <triggered_action>
```

下列 SQL 语句创建一个触发器，它在每次雇佣新人时会增加职员数，方法为每次向 EMPLOYEE 表添加一行时就在 COMPANY_STATS 表的职员数（NBEMP）列中加 1。

```
CREATE TRIGGER NEW_HIRED
      AFTER INSERT ON EMPLOYEE
      FOR EACH ROW
      UPDATE COMPANY_STATS SET NBEMP = NBEMP+1;
```

触发器主体可以包括下列 SQL 语句中的一个或多个: INSERT、搜索式 UPDATE、搜索式 DELETE、full-select、SET 转换变量和 SIGNAL SQLSTATE。可以在触发器引用的 INSERT、UPDATE 或 DELETE 语句之前或之后激活触发器。

相关概念:

- 第 102 页的『触发器相关性』
- 『INSERT, UPDATE, and DELETE triggers』（*Application Development Guide: Programming Server Applications*）
- 『Triggers in application development』（*Application Development Guide: Programming Server Applications*）

- 『Trigger creation guidelines』（*Application Development Guide: Programming Server Applications*）
- 第 102 页的『使用触发器更新视图内容』

相关任务:

- 第 172 页的『删除触发器』
- 『Creating triggers』（*Application Development Guide: Programming Server Applications*）
- 『Defining business rules using triggers』（*Application Development Guide: Programming Server Applications*）
- 『Defining actions using triggers』（*Application Development Guide: Programming Server Applications*）

相关参考:

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

触发器相关性

触发器与某个其它对象的所有相关性都记录在 SYSCAT.TRIGDEP 目录中。一个触发器可依赖许多个对象。这些对象和从属触发器在 DROP 语句中有更详尽的描述。

若删除这些对象中的一个，则该触发器就会不可用，但是它的定义仍保留在目录中。要重新激活此触发器，必须从目录中检索它的定义并提交新的 CREATE TRIGGER 语句。

若删除触发器，则它的描述会从 SYSCAT.TRIGGERS 目录视图中被删除，且它所有的相关性也从 SYSCAT.TRIGDEP 目录视图中被删除。所有与该触发器有 UPDATE、INSERT 或 DELETE 关系的程序包都会被停用。

若从属对象是视图且已使它不可用，则也会将该触发器标记为不可用。任何从属于已标记为不可用的触发器的程序包都会被停用。

相关概念:

- 第 102 页的『使用触发器更新视图内容』

相关任务:

- 第 100 页的『创建触发器』
- 第 172 页的『删除触发器』

相关参考:

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

使用触发器更新视图内容

可使用 INSTEAD OF 触发器来代表本质上不是可更新的视图执行删除、插入或更新请求。利用此类型触发器的应用程序能够对视图写入更新操作，就好象视图是表一样。

例如，可使用下列 SQL 语句来创建视图:

```
CREATE VIEW EMPV(EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO, HIREDATE,
      DEPTNAME)
   AS SELECT EMPNO, FIRSTNME, MIDINIT, LASTNAME, PHONENO, HIREDATE, DEPTNAME
      FROM EMPLOYEE, DEPARTMENT WHERE EMPLOYEE.WORKDEPT = DEPARTMENT.DEPTNO
```

由于加入了 EMPV 视图的主体，在添加下列语句之前，将不能使用该视图来更新基础表中的数据:

```
CREATE TRIGGER EMPV_INSERT INSTEAD OF INSERT ON EMPV
   REFERENCING NEW AS NEWEMP DEFAULTS NULL FOR EACH ROW
   INSERT INTO EMPLOYEE (EMPNO, FIRSTNME, MIDINIT, LASTNAME, WORKDEPT,
      PHONENO, HIREDATE)
   VALUES(EMPNO, FIRSTNME, MIDINIT, LASTNAME,
       COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
       WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
       RAISE_ERROR('70001', 'Unknown department name')),
       PHONENO, HIREDATE)
```

此 CREATE TRIGGER 语句将允许对 EMPV 视图执行 INSERT 请求。

```
CREATE TRIGGER EMPV_DELETE INSTEAD OF DELETE ON EMPV
   REFERENCING OLD AS OLDEMP FOR EACH ROW
   DELETE FROM EMPLOYEE AS E WHERE E.EMPNO = OLDEMP.EMPNO
```

此 CREATE TRIGGER 语句将允许对 EMPV 视图执行 DELETE 请求。

```
CREATE TRIGGER EMPV_UPDATE INSTEAD OF UPDATE ON EMPV
   REFERENCING NEW AS NEWEMP
               OLD AS OLDEMP
   DEFAULTS NULL FOR EACH ROW
   BEGIN ATOMIC
      VALUES(CASE WHEN NEWEMP.EMPNO = OLDEMP.EMPNO THEN 0
                  ELSE RAISE_ERROR('70002', 'Must not change EMPNO') END);
      UPDATE EMPLOYEE AS E
         SET (FIRSTNME, MIDINIT, LASTNAME, WORKDEPT, PHONENO, HIREDATE) =
             (NEWEMP.FIRSTNME, NEWEMP.MIDINIT, NEWEMP.LASTNAME,
              COALESCE((SELECT DEPTNO FROM DEPARTMENT AS D
                              WHERE D.DEPTNAME = NEWEMP.DEPTNAME),
              RAISE_ERROR('70001', 'Unknown department name')),
              NEWEMP.PHONENO, NEWEMP.HIREDATE)
         WHERE NEWEMP.EMPNO = E.EMPNO;
   END
```

此 CREATE TRIGGER 语句将允许对 EMPV 视图执行 UPDATE 请求。

相关任务:

- 第 100 页的『创建触发器』

相关参考:

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

创建用户定义的函数（UDF）或方法

用户定义的函数（UDF）扩展并添加了 SQL 的内置函数提供的支持，且可在可使用内置函数的任何地方使用。可按下列两种方式中任何一种创建 UDF:

- 外部函数，它是用一种编程语言编写的
- 有源函数，它的实现是从另一个现有函数继承来的

有三种类型的 UDF:

标量 每次调用它时，都返回一个单值答案。例如，内置函数 SUBSTR() 是一个标量函数。标量 UDF 可以是外部函数或有源函数。

列 从一组相似的值（一列）中返回单值答案。在 DB2® 中，它有时也称为聚集函数。列函数的一个示例为内置函数 AVG()。不能给 DB2 定义外部列 UDF，但是可以定义源于一个内置列函数的列 UDF。这对于单值类型是有用的。

例如，若存在用基本类型 INTEGER 定义的单值类型 SHOESIZE，则应可以定义源于内置函数 AVG(INTEGER) 的 UDF AVG(SHOESIZE)，且它应该是一个列函数。

表 将一个表返回至引用它的 SQL 语句。只能在 SELECT 语句的 FROM 子句中引用表函数。此类函数可用于将 SQL 语言处理能力应用于非 DB2 数据的数据，或将此类数据转换为 DB2 表。

例如，表函数可以提取一个文件并将它转换成表，将来自万维网的样本数据制成表，或存取 Lotus® Notes 数据库，并返回信息（如邮件消息的日期、发送方和文本）。此信息可以与该数据库中的其它表连接。

表函数只能是外部函数。它不能是有源函数。

有关现有 UDF 的信息记录在 SYSCAT.FUNCTIONS 和 SYSCAT.FUNCPARMS 目录视图中。系统目录不包含 UDF 的可执行代码。（因此，当创建备份和恢复计划时，应考虑如何管理 UDF 可执行程序。）

当编译 SQL 语句时，关于 UDF 性能的统计信息很重要。

相关概念:

- 『Scalar functions』（*SQL Reference, Volume 1*）
- 『User-defined functions』（*SQL Reference, Volume 1*）
- 『Table functions』（*SQL Reference, Volume 1*）
- 第 105 页的『创建函数映射』
- 『用户定义的函数的统计信息』（*《管理指南: 性能》*）
- 『用于手工更新目录统计信息的一般规则』（*《管理指南: 性能》*）
- 『DB2 User-Defined Functions and Methods』（*Application Development Guide: Programming Client Applications*）

相关任务:

- 第 105 页的『创建函数模板』

相关参考:

- 『Functions』（*SQL Reference, Volume 1*）
- 『CREATE FUNCTION statement』（*SQL Reference, Volume 2*）

有关创建用户定义的函数（UDF）或方法的详细信息

本节提供创建用户定义的函数或方法时要综合考虑的注意事项的信息。

创建函数映射

在一个联合数据库中，当需要在一个或多个数据源映射本地函数或具有函数的本地函数模板时，要创建函数映射。为许多数据源函数提供了缺省函数映射。

在下列情况下，函数映射很有用:

- 在一个数据源中出现了一个新的内置函数。
- 需要将一个数据源中的某个用户定义的函数映射为本地函数。
- 应用程序需要与缺省映射提供的缺省行为不同的行为。

用 CREATE FUNCTION MAPPING 语句定义的函数映射存储在联合数据库中。

函数（或函数模板）必须与数据源函数具有相同数目的输入参数。此外，联合端的输入参数数据类型应与数据源端的输入参数数据类型兼容。这些需求也适用于返回的值。

使用 CREATE FUNCTION MAPPING 语句创建函数映射。例如，要在 Oracle AVGNEW 函数与服务器 ORACLE1 上的等效 DB2® 函数之间创建函数映射:

```
CREATE FUNCTION MAPPING ORAVGNEW FOR SYSIBM.AVG(INT) SERVER ORACLE1
   OPTIONS (REMOTE_NAME 'AVGNEW')
```

必须对联合数据库具有 SYSADM 或 DBADM 权限，才可使用此语句。函数映射属性存储在 SYSCAT.FUNCMAPPINGS 中。

联合服务器将不绑定输入主变量，或检索 LOB、LONG VARCHAR/VARGRAPHIC、DATALINK、单值类型和结构化类型的结果。若输入参数或返回的值包括其中一个类型，则不能创建函数映射。

相关概念:

- 『Host language program mappings with transform functions』（*Application Development Guide: Programming Server Applications*）

相关任务:

- 第 105 页的『创建函数模板』

相关参考:

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

创建函数模板

在联合系统中，函数模板为函数映射提供了“锚点”。若对应的 DB2 函数在联合服务器中不存在，可使用函数模板来启用数据源函数的映射。函数映射需要一个函数模板或一个相似的现有 DB2 函数。

该模板只是一个函数外壳: 名称、输入参数和返回值。该函数没有本地可执行文件。

限制:

因为该函数没有本地可执行文件，所以即使可在数据源中使用该函数，但对该函数模板的调用仍有可能失败。例如，考虑如下查询:

```
SELECT myfunc(C1)
   FROM nick1
   WHERE C2 < 'A'
```

若 DB2 和包含 nick1 引用的对象的数据源没有相同的整理顺序，查询将失败，因为当该函数位于数据源时必须在 DB2 执行比较。若整理顺序相同，可在具有 myfunc 引用的基本函数的数据源中执行比较操作。

函数（或函数模板）必须与数据源函数具有相同数目的输入参数。联合端的输入参数数据类型应与数据源端的输入参数数据类型兼容。这些需求也适用于返回的值。

过程:

使用具有 AS TEMPLATE 关键字的 CREATE FUNCTION 语句创建函数模板。当创建该模板后，可使用 CREATE FUNCTION MAPPING 语句将模板映射为数据源。

例如，要为服务器 S1 上的函数 MYS1FUNC 创建一个函数模板和函数映射:

```
CREATE FUNCTION MYFUNC(INT) RETURNS INT AS TEMPLATE

   CREATE FUNCTION MAPPING S1_MYFUNC FOR MYFUNC(INT) SERVER S1 OPTIONS
   (REMOTE_NAME 'MYS1FUNC')
```

相关概念:

- 第 105 页的『创建函数映射』

相关参考:

- 『CREATE FUNCTION (Sourced or Template) statement』（*SQL Reference, Volume 2*）

用户定义的类型（UDT）

用户定义的类型（UDT）是由用户在数据库中创建的命名的数据类型。UDT 可以是单值类型，它与内部数据类型或结构化类型共享一个公共的表示法，结构化类型具有一个命名属性序列，其中每个属性都有一个类型。结构化类型可以是另一个定义类型层次结构的结构化类型（称为超类型）的子类型。

UDT 支持强类型，这表示即使它们与其它类型共享相同的表示，但一个给定 UDT 的值会被视为只与同一个类型层次结构中相同的 UDT 的值兼容。

SYSCAT.DATATYPES 目录视图允许查看已为数据库定义的 UDT。此目录视图还显示当创建该数据库时由数据库管理器定义的数据类型。

UDT 不能用作大多数系统提供的函数或内置函数的自变量。必须提供用户定义的函数来启用这些操作和其它操作。

仅在下列情况下，才可以删除 UDT:

- 现有表的列定义中未使用它。
- 未把它用作现有的类型表或带类型视图的类型。
- 在不能删除的 UDF 函数中未使用它。若视图、触发器、表检查约束或另一个 UDF 从属于某个 UDF，则不能删除该 UDF。

当删除一个 UDT 时，也会删除从属于它的任何函数。

相关概念:

- 第 108 页的『创建用户定义的结构化类型』

相关任务:

- 第 107 页的『创建用户定义的单值类型』

相关参考:

- 『User-defined types』（*SQL Reference, Volume 1*）
- 『Data types』（*SQL Reference, Volume 1*）

有关创建用户定义的类型（UDT）的详细信息

此处讨论的单值类型和结构化类型定义作为联合类型映射。

创建用户定义的单值类型

用户定义的单值类型是从现有类型（如整数、小数或字符类型）派生的数据类型。可使用 CREATE DISTINCT TYPE 语句创建单值类型。

限制:

若在 CREATE DISTINCT TYPE 语句中指定 WITH COMPARISONS 子句（如本示例），则具有相同单值类型的实例可以相互比较。若源数据类型是大对象、DATALINK、LONG VARCHAR 或 LONG VARGRAPHIC 类型，则不能指定 WITH COMPARISONS 子句。

具有单值类型的实例不能用作在源类型上定义的函数的自变量或操作的操作数。类似地，源类型不能用于为使用单值类型定义的自变量或操作数。

过程:

如下 SQL 语句将单值类型 t_educ 创建为 smallint:

```
CREATE DISTINCT TYPE T_EDUC AS SMALLINT WITH COMPARISONS
```

一旦创建了单值类型，就可以在 CREATE TABLE 语句中使用它来定义列:

```
CREATE TABLE EMPLOYEE
      (EMPNO        CHAR(6)      NOT NULL,
       FIRSTNME   VARCHAR(12)  NOT NULL,
       LASTNAME   VARCHAR(15)  NOT NULL,
       WORKDEPT   CHAR(3),
       PHONENO    CHAR(4),
       PHOTO      BLOB(10M)    NOT NULL,
       EDLEVEL    T_EDUC)
   IN RESOURCE
```

创建单值类型还会生成对在单值类型和源类型之间进行强制类型转型的支持。因此，类型为 T_EDUC 的值可强制类型转型为 SMALLINT 值，而 SMALLINT 值可强制类型转型为 T_EDUC 值。

可通过转换将 UDT 变换为基本数据类型，或将基本数据类型变换为 UDT。通过 CREATE TRANSFORM 语句创建变换函数。

也可通过 CREATE METHOD 语句以及 CREATE FUNCTION 语句的扩充语句来获得变换支持。

相关概念:

- 第 106 页的『用户定义的类型（UDT）』

相关参考:

- 『CREATE DISTINCT TYPE statement』（*SQL Reference, Volume 2*）
- 『CREATE TRANSFORM statement』（*SQL Reference, Volume 2*）
- 『CREATE METHOD statement』（*SQL Reference, Volume 2*）
- 『CREATE FUNCTION (Sourced or Template) statement』（*SQL Reference, Volume 2*）
- 『User-defined types』（*SQL Reference, Volume 1*）
- 『Data types』（*SQL Reference, Volume 1*）

创建用户定义的结构化类型

结构化类型是包含一个或多个指定属性的用户定义的数据类型。每个属性都具有它自己的名称和数据类型。属性（Attribute）是用来描述一种类型的实例的属性（property）。结构化类型可用作表的类型，在这种情况下，表中的每一列可从该结构化类型的一个属性中获取名称和数据类型。

相关概念:

- 『User-defined structured types』（*Application Development Guide: Programming Server Applications*）
- 『Structured type hierarchies』（*Application Development Guide: Programming Server Applications*）

相关任务:

- 『Creating structured types』（*Application Development Guide: Programming Server Applications*）
- 『Creating a structured type hierarchy』（*Application Development Guide: Programming Server Applications*）

相关参考:

- 『CREATE TYPE (Structured) statement』（*SQL Reference, Volume 2*）
- 『User-defined types』（*SQL Reference, Volume 1*）

创建类型映射

在联合系统中，类型映射允许将数据源表和视图中的特定数据类型映射为 DB2 单值数据类型。一个类型映射可以适用于一个数据源或一个范围内（类型和版本）的数据源。

为内部数据源类型和内置 DB2 类型提供了缺省数据类型映射。将在 SYSCAT.TYPEMAPPINGS 视图中列出您创建的新的数据类型映射。

限制:

不能为 LOB、LONG VARCHAR/VARGRAPHIC、DATALINK、结构化类型或单值类型创建类型映射。

过程:

用 CREATE TYPE MAPPING 语句创建类型映射。必须对联合数据库具有 SYSADM 或 DBADM 权限，才可使用此语句。

类型映射语句的一个示例是:

```
CREATE TYPE MAPPING MY_ORACLE_DEC FROM SYSIBM.DECIMAL(10,2)
   TO SERVER ORACLE1 TYPE NUMBER([10..38],2)
```

相关参考:

- 『CREATE TYPE MAPPING statement』（*SQL Reference, Volume 2*）
- 『Data Type Mappings between DB2 and OLE DB』（*Application Development Guide: Programming Client Applications*）

创建视图

视图可从一个或多个基本表、别名或视图中派生，且可以在检索数据时与基本表互换使用。当对视图中显示的数据进行更改时，该数据会在表中自行更改。

可以创建视图来限制对敏感数据的存取，同时又允许对其它数据进行更多的一般存取。

当插入到一个视图中，而该视图的视图定义的 SELECT 列表直接或间接地包括基本表的标识列的名称时，同一规则适用，就象 INSERT 语句直接引用基本表的标识列一样。

除按上述方式使用视图外，视图还可以用于:

- 改变表而不影响应用程序。这可通过创建一个基于基础表的视图来完成。使用基础表的应用程序不会因新视图的创建而受影响。新的应用程序可将创建的视图用于与那些使用基础表的应用程序不同的目的。
- 对一列中的值求和，选择最大值，或计算平均值。
- 存取一个或多个数据源中的信息。可在 CREATE VIEW 语句内引用别名，并可创建多个位置 / 全局视图（该视图可以连接位于不同系统中多个数据源的信息）。

  当使用标准的 CREATE VIEW 语法创建一个引用别名的视图时，将看到一个警告，它提醒您将用于存取数据源处的基本对象的是视图用户的认证标识而不是视图创建者的认证标识。使用 FEDERATED 关键字阻止此警告。

创建视图的另一种方法是使用嵌套的或公共表表达式，以减少目录查找并提高性能。

先决条件:

在创建视图之前，视图以其为基础的基本表、别名或视图必须已经存在。

限制:

可以创建在其定义中使用 UDF 的视图。但是，要更新此视图以使它包含最新的函数，必须删除它，然后重新创建它。若视图从属于 UDF，则不能删除该函数。

下列 SQL 语句创建一个在其定义中带有函数的视图:

```
CREATE VIEW EMPLOYEE_PENSION (NAME, PENSION)
  AS SELECT NAME, PENSION(HIREDATE,BIRTHDATE,SALARY,BONUS)
     FROM EMPLOYEE
```

UDF 函数 PENSION 根据涉及 HIREDATE、BIRTHDATE、SALARY 和 BONUS 的一个公式来计算一个职员应当得到的当前退休金。

过程:

要使用“控制中心”来创建视图:

1. 展开对象树，直到您看到**视图**文件夹为止。
2. 右键单击**视图**文件夹，并从弹出菜单中选择**创建**。
3. 填写信息，并单击**确定**。

要使用命令行来创建视图，输入:

```
CREATE VIEW <name> (<column>, <column>, <column>)
   SELECT <column_names> FROM <table_name>
   WITH CHECK OPTION
```

例如，EMPLOYEE 表中可能有工资信息，它不应对每个人都是可用的。但是，职员的电话号码应是一般人都可存取的。在此情况中，可以仅根据 LASTNAME 和 PHONENO 列创建一个视图。可将该视图的存取权授予 PUBLIC，而将整个 EMPLOYEE 表的存取权限制在具有查看工资信息授权的那些人范围内。

使用视图，可以使表数据的子集可用于一个应用程序，并验证要插入或更新的数据。视图可以有与原始表中对应列的名称不同的列名。

使用视图使程序和最终用户查询可以灵活地查看表数据。

下列 SQL 语句创建 EMPLOYEE 表的视图，它列示部门 A00 的所有职员及其职员姓名和电话号码:

```
CREATE VIEW EMP_VIEW (DA00NAME, DA00NUM, PHONENO)
     AS SELECT LASTNAME, EMPNO, PHONENO FROM EMPLOYEE
     WHERE WORKDEPT = 'A00'
     WITH CHECK OPTION
```

此语句的第一行对该视图命名并定义它的列。名称 EMP_VIEW 在 SYSCAT.TABLES 中的模式内必须是唯一的。尽管不包含数据，视图名看上去仍象一个表名。该视图将有称为 DA00NAME、DA00NUM 和 PHONENO 的三列，它们与 EMPLOYEE 表中的列 LASTNAME、EMPNO 和 PHONENO 相对应。列出的列名按一一对应的关系应用于 SELECT 语句的选择列表。若不指定列名，则视图使用与 SELECT 语句的结果表的列相同的名称。

第二行是描述要从数据库选择哪些值的 SELECT 语句。它可以包括子句: ALL、DISTINCT、FROM、WHERE、GROUP BY 和 HAVING。为视图提供列的数据对象的一个或多个名称必须跟在 FROM 子句后面。

WITH CHECK OPTION 子句指示必须根据该视图定义检查该视图的任何更新的行或插入的行，若它不符合，则拒绝它。这增强了数据完整性，但是需要附加的处理。若将此子句省略，则不会根据视图定义检查插入和更新。

以下 SQL 语句使用 SELECT AS 子句创建基于 EMPLOYEE 表的相同视图:

```
CREATE VIEW EMP_VIEW
      SELECT LASTNAME AS DA00NAME,
             EMPNO AS DA00NUM,
             PHONENO
        FROM EMPLOYEE
      WHERE WORKDEPT = 'A00'
      WITH CHECK OPTION
```

相关概念:

- 『Views』（*SQL Reference, Volume 1*）
- 第 208 页的『表和视图特权』
- 第 216 页的『使用视图控制对数据的存取』
- 第 102 页的『使用触发器更新视图内容』

相关任务:

- 第 111 页的『创建带类型视图』
- 第 153 页的『从表或视图除去行』
- 第 174 页的『改变或删除视图』
- 第 176 页的『恢复不可用视图』

相关参考:

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

有关创建视图的详细信息

带类型视图以预定义的结构化类型为基础。

创建带类型视图

过程:

可使用 CREATE VIEW 语句来创建带类型视图。

相关概念:

- 『Typed views』（*Application Development Guide: Programming Server Applications*）

相关任务:

- 『Creating typed views』（*Application Development Guide: Programming Server Applications*）
- 『Altering typed views』（*Application Development Guide: Programming Server Applications*）
- 『Dropping typed views』（*Application Development Guide: Programming Server Applications*）

相关参考:

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

创建具体查询表

*具体查询表*是以查询结果为基础所定义的一种表。因此，具体查询表通常包含预先计算的结果，这些结果是根据表定义中引用的一个或多个表中的现有数据计算而得。若 SQL 编译器确定查询在针对具体查询表运行时比针对一个或多个基本表运行时效率更高，将对具体查询表执行该查询，且能够更快地获得结果。

限制:

不要将定义为 REFRESH DEFERRED 的具体查询表用于优化静态 SQL。

设置 CURRENT REFRESH AGE 专用寄存器为一个非零的值时应小心。如果允许使用不能表现基本表的值的具体查询表来优化查询的处理，则查询的结果*不能*准确地表现基础表中的数据。当知道基础数据未更改时，或者当根据对数据的了解愿意接受结果中的错误程度时，这可能是合理的。

若要以任何有效的 *fullselect* 为基础创建一个新基本表，则在创建该表时指定 DEFINITION ONLY 关键字。当创建表操作完成时，不要将新表作为具体查询表而是作为基本表来处理。例如，可以创建用于 LOAD 和 SET INTEGRITY 的异常表，如下所示:

```
CREATE TABLE XT AS
   (SELECT T.*, CURRENT TIMESTAMP AS TIMESTAMP,CLOB(",32K)
   AS MSG FROM T) DEFINITION ONLY
```

以下是关于具体查询表的一些关键限制:

1. 不能改变具体查询表。
2. 如果基本表具有具体查询表，则不能改变该基本表的列的长度。
3. 不能将数据导入到具体查询表中。
4. 不能对具体查询表创建唯一索引。
5. 不能根据引用一个或多个别名的查询的结果创建具体查询表。

过程:

使用复制选项创建具体查询表这一方法可用来在分区数据库环境中复制所有节点上的表。它们称为“复制具体查询表”。

通常，如果具体查询表或复制具体查询表的隔离级别高于或等于查询的隔离级别，则使用该具体查询表或复制具体查询表来优化该查询。例如，如果查询在游标稳定性（CS）隔离级别下运行，则仅使用在 CS 或更高隔离级别下定义的具体查询表和复制具体查询表来进行优化。

要创建具体查询表，将 CREATE TABLE 语句与 AS *fullselect* 子句和 IMMEDIATE 或 REFRESH DEFERRED 选项配合使用。

您可以选择唯一标识具体查询表的列的名称。列名列表所包含的名称数必须与全选择的结果表中的列数相等。若全选择的结果表带有重复的列名或带有未命名的列，则必须给出列名列表。未命名的列是从使用选择列表的 AS 子句时未命名的常量、函数、表达式或设置操作派生的。若不指定列名列表，则表列继承全选择的结果集的列名。

创建具体查询表时，可选择指定是系统维护具体查询表还是用户维护具体查询表。缺省值是系统维护，可使用 MAINTAINED BY SYSTEM 子句显式指定。可使用 MAINTAINED BY USER 子句指定用户维护的具体查询表。

如果创建系统维护的具体查询表，还可以选择指定是在更改基本表时自动刷新具体查询表还是使用 REFRESH TABLE 语句来刷新它。要在更改基本表时自动刷新具体查询表，指定 REFRESH IMMEDIATE 关键字。在下列情况下，即时刷新很有帮助:

- 查询需要确保它们存取的数据是最新的
- 基本表不经常更改
- 刷新成本不高

在这种情况下，具体查询表可提供预先计算的结果。若想要延迟具体查询表的刷新，指定 REFRESH DEFERRED 关键字。指定为 REFRESH DEFERRED 的具体查询表将**不**反映对底层基本表的更改。如果不要求反映基本表的更改，应使用具体查询表。例如，如果运行 DSS 查询，应使用具体查询表以包含旧数据。

在下列情况下，可使用定义为 REFRESH DEFERRED 的具体查询表代替查询:

- 除下列情况外，符合即时刷新总结表的全选择的限制:
  - 不需要 SELECT 列表来包括 COUNT(*) 或 COUNT_BIG(*)
  - SELECT 列表可包括 MAX 和 MIN 列函数
  - 允许 HAVING 子句

使用 CURRENT REFRESH AGE 专用寄存器来指定定义为 REFRESH DEFERRED 的具体查询表在必须刷新之前可供动态查询使用的时间长短。要设置 CURRENT REFRESH AGE 专用寄存器的值，可使用 SET CURRENT REFRESH AGE 语句。

可将 CURRENT REFRESH AGE 专用寄存器设置为 ANY 或值 99999999999999，以允许在动态查询中使用延迟式具体查询。这串由 9 组成的数是此专用寄存器中允许的最大值，该值是数据类型为 DECIMAL(20,6) 的时间戳记持续时间值。值零（0）指示只有定义为 REFRESH IMMEDIATE 的具体查询表可用来优化查询的处理。在这种情况下，不要将定义为 REFRESH DEFERRED 的具体查询表用于优化。

定义为 REFRESH IMMEDIATE 的具体查询表适用于静态和动态查询，且不需要使用 CURRENT REFRESH AGE 专用寄存器。

在使用 ENABLE QUERY OPTIMIZATION 子句定义表之后，具体查询表就会让查询路由至它们，且如果是延迟式具体查询表，则会将 CURRENT REFRESH AGE 专用寄存器设置为 ANY。但是，对于用户维护的具体查询表，使用 CURRENT REFRESH AGE 专用寄存器并非控制查询的重新选择路由的最好方法。CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION 专用寄存器将指示哪种类型的高速缓存数据可用于路由。

由于活动会影响源数据，因此随着时间的推移，具体查询表将不会再包含准确的数据。将需要使用 REFRESH TABLE 语句。

相关概念:

- 『Isolation levels』（*SQL Reference, Volume 1*）

相关任务:

- 第 167 页的『改变具体查询表属性』
- 第 167 页的『刷新具体查询表中的数据』
- 第 176 页的『删除具体查询表或分级表』

相关参考:

- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）
- 『REFRESH TABLE statement』（*SQL Reference, Volume 2*）
- 『SET CURRENT REFRESH AGE statement』（*SQL Reference, Volume 2*）
- 『CURRENT REFRESH AGE special register』（*SQL Reference, Volume 1*）
- 『CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION special register』（*SQL Reference, Volume 1*）
- 『SET CURRENT MAINTAINED TABLE TYPES FOR OPTIMIZATION statement』（*SQL Reference, Volume 2*）

创建用户维护的具体查询表

用户维护的具体查询表（MQT）对于数据库系统很有用，总结数据表已存在于此系统中。维护此类总结表的定制应用程序是常用的应用程序。将现有总结表标识为用户维护的 MQT，将导致查询优化器使用现有总结表以针对基本表计算查询的结果集。

注: 为静态 SQL 查询选择存取方案时，查询优化器未使用用户维护的 MQT。

限制:

如果创建用户维护的具体查询表，与系统维护的具体查询表相关联的限制仍然适用，但下列情况除外:

- 允许对具体查询表执行 INSERT、UPDATE 和 DELETE 操作。但是，不会针对基本表执行有效性检验。数据的正确性由您自己负责。
- LOAD、EXPORT、IMPORT 和数据复制将使用此类型的具体查询表，但没有有效性检验时除外。
- 不允许您对此类型的具体查询表使用 REFRESH TABLE 语句。
- 不允许您对此类型的具体查询表使用 SET INTEGRITY ... IMMEDIATE CHECKED 语句。
- 必须将用户维护的具体查询表定义为 REFRESH DEFERRED。

有关其它限制，请参阅『创建具体查询表』主题。

过程:

要创建具体查询表，将 CREATE TABLE 语句与 AS *fullselect* 子句和 IMMEDIATE 或 REFRESH DEFERRED 选项配合使用。

创建具体查询表时，可选择指定是系统维护具体查询表还是用户维护具体查询表。缺省值是系统维护，可使用 MAINTAINED BY SYSTEM 子句显式指定。可使用 MAINTAINED BY USER 子句指定用户维护的具体查询表。

在大型数据库环境或数据仓库环境中，通常会有用来维护和装入用户维护的具体查询表的定制应用程序。

注: 要使优化器考虑用户维护的 MQT，查询优化级别必须设置为第 2 级，或者设置为大于或等于第 5 级的级别。

相关任务:

- 第 115 页的『填充用户维护的具体查询表』

相关参考:

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

填充用户维护的具体查询表

在创建表来容纳总结信息之后，您将会想要使用总结数据来填充具体查询表（MQT），在确定结果集时，您希望优化器使用这些总结数据。

先决条件:

确保容纳总结信息的表存在。

过程:

您可以使用触发器、插入操作或 LOAD、IMPORT 和 DB2 DataPropagator 实用程序来填充用户维护的 MQT。当初次填充用户维护的 MQT 时，可以使用 LOAD 或 IMPORT 实用程序避免记录开销。

下列步骤提供了一种填充用户维护的 MQT 的典型方法:

- 使基本表处于只读状态以避免创建新记录或修改现有记录。
- 从基本表中抽取所需数据和将数据写入外部文件。
- 将数据从外部文件导入或装入 MQT。您可以在 CHECK PENDING NO ACCESS 状态下对表使用 LOAD 或 IMPORT 实用程序。

  注: 如果要通过 SQL 插入操作来填充 MQT，则需要复位 PENDING NO ACCESS 状态。但是，必须首先使用 ALTER TABLE 语句的 SET MATERIALIZED QUERY 子句中的 DISABLE QUERY OPTIMIZATION 选项来禁用优化器，以确保动态 SQL 查询不会在建立 MQT 中的数据时意外地优化此 MQT。填充 MQT 之后，需要使用 ALTER TABLE 语句的 SET MATERIALIZED QUERY 子句中的 ENABLE QUERY OPTIMIZATION 选项来启用优化。
- 要对新的 MQT 发出 SQL 查询，请重新设置 CHECK PENDING NO ACCESS 状态。执行此操作则表示您承担了保证具体化视图数据完整性的责任。执行此操作的语句是:

  ```
  DB2 SET INTEGRITY FOR example ALL IMMEDIATE UNCHECKED
  ```

- 使基本表处于读 / 写状态。

注: 要使优化器考虑用户维护的 MQT，查询优化级别必须设置为第 2 级，或者设置为大于或等于第 5 级的级别。

相关参考:

- 『IMPORT Command』（*Command Reference*）
- 『LOAD Command』（*Command Reference*）

创建分级表

分级表允许对延迟式具体查询表的增量维护支持。分级表收集需要应用于具体查询表以使其与基础表的内容同步的更改。使用分级表消除了在请求对具体查询表的即时刷新时由于立即维护内容而引起的高锁定争用。另外，每次执行 REFRESH TABLE 时，不再需要完全重新生成具体查询表。

在改进复杂查询的响应时间方面，具体查询表是非常有效的方法，特别是针对可能需要下列某些操作的查询:

- 聚集涉及一个或多个维的数据
- 连接和聚集涉及一组表的数据
- 经常存取的数据的子集中的数据
- 在分区数据库环境中，表或表的一部分中的重新分区的数据

限制:

以下是关于分级表的一些关键限制:

1. 用来定义分级表的查询必须是可增量维护的；即，它必须与带有即时刷新选项的具体查询表遵守相同的规则。
2. 只有延迟式刷新才能有支持分级表。查询还定义与分级表相关联的具体查询表。具体查询表必须定义为 REFRESH DEFERRED。
3. 使用分级表进行刷新时，仅支持刷新至当前时间点。

过程:

不能使用不一致、不完整或处于暂挂状态的分级表来增量刷新相关联的具体查询表，除非执行了某些其它操作。这些操作将使分级表的内容与其相关联的具体查询表及其基础表保持一致，并使分级表脱离暂挂状态。刷新具体查询表之后，其分级表的内容会被清除并将分级表设置为正常状态。还可以使用带有相应选项的 SET INTEGRITY 语句有目的地修剪分级表。修剪会将分级表更改为不一致状态。例如，下列语句强制修剪称为 STAGTAB1 的分级表:

```
SET INTEGRITY FOR STAGTAB1 PRUNE;
```

创建分级表时，会将它置于暂挂状态并会出现一个指示符，显示该表对于基础表的内容及相关联的具体查询表是不一致或不完整的。需要使分级表脱离暂挂状态和不一致状态以便开始收集对其基础表的更改。处于暂挂状态时，试图对任何分级表的基础表进行修改都将失败，试图刷新相关联的具体查询表也会失败。

有几种方法可使分级表脱离暂挂状态；例如:

- SET INTEGRITY FOR <staging table name> STAGING IMMEDIATE UNCHECKED
- SET INTEGRITY FOR <staging table name> IMMEDIATE CHECKED

相关任务:

- 第 112 页的『创建具体查询表』
- 第 167 页的『改变具体查询表属性』
- 第 167 页的『刷新具体查询表中的数据』
- 第 176 页的『删除具体查询表或分级表』

相关参考:

- 『SET INTEGRITY statement』（*SQL Reference, Volume 2*）

创建别名

别名是引用表、别名或视图的间接方法，这样 SQL 语句可与该表或视图的限定名无关。仅当表名或视图名更改的情况下，才必须更改别名定义。可以在一个别名上创建另一个别名。别名可以在视图或触发器定义以及任何 SQL 语句中使用，但表检查约束定义除外，在该定义中可以引用现有的表名或视图名。

先决条件:

可以为定义时不存在的表、视图或别名定义别名。但是，当编译包含该别名的 SQL 语句时，它必须存在。

限制:

别名可以在任何可使用现有表名的地方使用，且在别名链中不存在循环引用或重复引用的情况下，可以引用另一个别名。

别名不能与现有的表、视图或别名同名，而只能引用同一个数据库中的一个表。在 CREATE TABLE 或 CREATE VIEW 语句中使用的表或视图的名称不能与相同模式中的别名相同。

除非别名所处的模式不是您当前的授权标识所拥有的模式（它需要 DBADM 权限），否则，创建别名不需要特权。

当删除一个别名或别名引用的对象时，从属于该别名的所有程序包就会标记为无效，而从属于该别名的所有视图和触发器则标记为不可用。

过程:

要使用“控制中心”来创建别名:

1. 展开对象树，直到您看到**别名**文件夹为止。
2. 右键单击**别名**文件夹，并从弹出菜单中选择**创建**。
3. 填写信息，并单击**确定**。

要使用命令行来创建别名，输入:

```
CREATE ALIAS <alias_name> FOR <table_name>
```

在编译语句时，别名被表名或视图名替换。若别名或别名链不能被解析为表名或视图名，则将导致错误。例如，若 WORKERS 是 EMPLOYEE 的一个别名，则在编译时:

```
SELECT * FROM WORKERS
```

就会使以下语句生效

```
SELECT * FROM EMPLOYEE
```

下列 SQL 语句为 EMPLOYEE 表创建别名 WORKERS:

```
CREATE ALIAS WORKERS FOR EMPLOYEE
```

注: DB2 OS/390 或 z/Series 版使用两种不同概念的别名: ALIAS 和 SYNONYM。这两种概念在 DB2 通用数据库中是有区别的，如下所示:

- DB2 OS/390 或 z/Series 版中的 ALIAS:
  - 要求它们的创建者具有特殊的权限或特权
  - 不能引用其它别名
- DB2 OS/390 或 z/Series 版中的 SYNONYM:
  - 只能被它们的创建者使用
  - 始终是非限定的
  - 删除引用的表时被删除
  - 不与表或视图共享名称空间

相关概念:

- 『Aliases』（*SQL Reference, Volume 1*）

相关参考:

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

索引、索引扩展或索引规范

索引是行位置的列表，按一个或多个指定列的内容来排序。索引通常用于加速对表的存取。但是，它们还可以为逻辑数据设计服务。例如，*唯一索引*不允许列中存在重复值的条目，从而保证了一个表中不会有两行相同。也可以创建索引，以指定一个列中值的递增顺序或降序顺序。

索引扩展是一个索引对象，它配合带有结构化类型或单值类型列的索引使用。

索引规范是一个元数据构造。它告诉优化器别名所引用的数据源对象（表或视图）是否存在索引。索引规范不包含行位置的列表 – 它只是索引的描述。优化器使用索引规范来改进对由别名表示的对象的存取。当第一次创建别名时，如果数据源中的基础表有索引且该索引使用 DB2® 可识别的格式，则会生成索引规范。

注: 需要的话，根据表别名或视图别名（对于视图基于一个表的情况）创建索引规范。

在下列情况下手工创建索引或索引规范:

- 它将改进性能。例如，若要鼓励优化器使用特定的表或别名作为嵌套循环连接的内部表，且不存在索引的话，则基于该连接列创建索引规范。
- 在创建基本表的别名后，添加了该表的索引。

当基本表上不存在任何索引时，可创建索引规范（当发出 CREATE INDEX 语句时，DB2 将不会检查是否存在远程索引）。即使指定了 UNIQUE 关键字，索引规范也不会强制行的唯一性。

“DB2 索引顾问程序”是一个向导，它辅助您选择一组最优的索引。可通过“控制中心”访问此向导。一个类似的实用程序称为 *db2advis*。

索引是由基本表中的列定义的。它可以由表的创建者或知道某些列需要直接访问的用户来定义。除非已经存在用户定义的索引，否则会根据主键来自动创建主索引键。

可在特定的基本表上定义任意个索引，且这些索引可对查询性能产生良好的影响。但是，存在的索引越多，在更新、删除和插入操作期间数据库管理器必须修改的索引就越多。为接收很多更新的表创建大量索引可能减慢请求的处理速度。因此，仅当频繁存取有明显有利之处时，才使用索引。

索引中的最大列数是 16。若是对类型表建立索引，则最大列数是 15。索引键的最大长度是 1024 字节。如前所述，表上的索引键太多会减慢请求的处理速度。同样，大型索引键也会减慢处理请求的速度。

*索引键*是定义了索引的一个列或一些列的集合，它决定索引的有用程度。虽然构成一个索引键的列的顺序不会给索引键的创建带来影响，但是当它决定是否使用索引时就可能影响优化器。

若要建立索引的表是空的，则仍创建索引，但是在未装入该表或插入行之前，不会建立任何索引条目。若该表不是空的，则数据库管理器在处理 CREATE INDEX 语句时会建立索引条目。

对于*群集索引*，将新行实际插入具有相似键值的现有行附近。这可改善查询性能，因为它导致数据页的存取模式更线性化，且产生更有效的预取。

若要让主键索引成为群集索引，不应在 CREATE TABLE 中指定主键。一旦创建了主键，就不能修改相关的索引。而是执行不带主键子句的 CREATE TABLE。然后，发出 CREATE INDEX 语句，并指定群集属性。最后，使用 ALTER TABLE 语句添加与刚创建的索引对应的主键。将把此索引用作主键索引。

通常，若群集索引是唯一的，则群集维护起来就更有效率。

不是唯一索引键的一部分但是要在该索引中存储 / 维护的列数据称为*包含列*。只能为唯一索引指定包含列。当用包含列创建索引时，仅对唯一键列排序并考虑其唯一性。当涉及到索引存取时，包含列的使用可提高数据检索的性能。

数据库管理器使用 B+ 树结构来存储索引，该结构的底层由叶节点组成。叶节点或叶子页是存储实际的索引键值之处。当创建索引时，允许联机合并那些索引叶子页。联机索引碎片整理用来防止以下情况：在进行大量删除和更新活动后以及某个索引的大部分叶子页只剩下少许索引键时。在此类情况下，如果不进行联机索引碎片整理，只能通过重组数据（包括或不包括索引）来回收空间。当决定是否创建有能力对索引页进行联机碎片整理的索引时，应考虑以下问题：每次从叶子页物理除去键时，因为检查要合并的空间而增加的性能成本以及完成合并（如果有足够的空间）的实际成本是否大于较好地利用索引空间所带来的好处，且小于执行重组以回收空间这一减少了的需要？

注:

1. 在联机索引碎片整理后释放的页仅可重新用于同一个表中的其它索引。对于完整重组，释放的那些页可用于其它对象（当使用“数据库管理存储器”时）或磁盘空间（当使用“系统管理存储器”时）。此外，联机索引碎片整理将不会释放索引的任何非叶子页，而完整重组通过减少非叶子页和叶子页以及索引层数以使索引尽可能地小。
2. 在版本 8 之前创建的索引中，键是在删除或更新表行时从叶子页中物理除去的。对于类型 2 索引，删除或更新某行时，只是将键标记为已删除。在落实删除或更新操作后完成清除之前，不会从页中物理除去它。这种清除可能由后续事务完成，该事

务正在更改键为标记为已删除的页。可使用 REORG INDEXES 实用程序的 CLEANUP ONLY [ALL | PAGES] 选项显式触发清除。

使用相同的 CREATE INDEX 语句来构建分区数据库中表的索引。根据该表的分区键来对其分区。一个表上的索引由数据库分区组中每个节点上该表的本地索引组成。注意，在一个多分区环境中定义的唯一索引必须是分区键的超集。

相关概念:

- 『Indexes』（*SQL Reference, Volume 1*）
- 第 121 页的『使用索引』
- 第 122 页的『CREATE INDEX 语句中的选项』
- 第 125 页的『创建用户定义扩展索引类型』
- 第 210 页的『索引特权』

相关任务:

- 第 9 页的『在创建索引时启用并行性』
- 第 120 页的『创建索引』
- 第 169 页的『重命名现有表或索引』
- 第 178 页的『删除索引、索引扩展或索引规范』

相关参考:

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

有关创建索引、索引扩展或索引规范的详细信息

您可以使用数据库管理器所维护的索引或者指定自己的索引。

创建索引

*索引*是一个或多个键的集合，每个键指向表中的一行。索引允许通过指针创建指向数据的直接路径更有效地存取表中的各行。

过程:

性能提示: 若要执行以下一系列任务:

1. 创建表
2. 装入表
3. 创建索引（没有 COLLECT STATISTICS 选项）
4. 执行 RUNSTATS

或者，如果准备执行下面的一系列任务:

1. 创建表
2. 装入表
3. 创建索引（带有 COLLECT STATISTICS 选项）

应该考虑按下列顺序执行任务:

1. 创建表
2. 创建索引
3. 装入请求了 statistics yes 选项的表

在创建索引之后，维护它们。随后，当应用程序使用一个键值来随机存取和处理表中的行时，就可以使用基于该键值的索引来直接存取行。这很重要，因为一个基本表中各行的物理存储器是无序的。

创建表时，可选择创建多维群集（MDC）表。通过创建此类型的表，会创建块索引。常规索引指向各个行；块索引指向块或数据块，且比常规索引稍小。块索引与常规索引一起存储在于同一表空间中。

当插入一行时，除非定义了一个群集索引，否则，将它置于可以容纳它的最方便的存储位置中。当搜索满足特定选择条件的表行且该表没有索引时，就会扫描整个表。索引优化了数据检索，而无需执行长时间的顺序搜索。

索引数据可以与表数据存储在相同的表空间中，或存储在包含索引数据的单独表空间中。用来存储索引数据的表空间是在创建表时确定的。

要使用“控制中心”来创建索引:

1. 展开对象树，直到您看到**索引**文件夹为止。
2. 右键单击**索引**文件夹，并从弹出菜单中选择**创建 -> 使用向导创建索引**。
3. 遵循向导中的步骤来完成该任务。

要使用命令行来创建索引，输入:

```
CREATE INDEX <name> ON <table_name> (<column_name>)
```

相关概念:

- 『Optimizing load performance』（*Data Movement Utilities Guide and Reference*）
- 第 121 页的『使用索引』
- 第 122 页的『CREATE INDEX 语句中的选项』
- 第 210 页的『索引特权』

相关任务:

- 第 169 页的『重命名现有表或索引』
- 第 178 页的『删除索引、索引扩展或索引规范』

相关参考:

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

使用索引

应用程序永远不会直接使用索引。决定是否使用索引和要使用哪些潜在可用的索引是优化器的责任。

表上最好的索引具有下列特点:

- 使用高速磁盘

- 是高度群集的
- 由少数窄列组成
- 使用具有高基数的列

相关概念:

- 『索引规划技巧』（*《管理指南: 性能》*）
- 『索引性能技巧』（*《管理指南: 性能》*）
- 『通过索引扫描的数据存取』（*《管理指南: 性能》*）
- 『标准表的表和索引管理』（*《管理指南: 性能》*）
- 『MDC 表的表和索引管理』（*《管理指南: 性能》*）

CREATE INDEX 语句中的选项

可以创建一个索引，它将允许重复值（非唯一索引）以便可按非主键的列来启用有效检索，也允许在已建立索引的一列或多列中存在重复值。

下列 SQL 语句根据 EMPLOYEE 表中的 LASTNAME 列创建称为 LNAME 且按递增顺序排序的非唯一索引:

```
CREATE INDEX LNAME ON EMPLOYEE (LASTNAME ASC)
```

以下 SQL 语句基于电话号码列创建唯一索引:

```
CREATE UNIQUE INDEX PH ON EMPLOYEE (PHONENO DESC)
```

唯一索引确保在已建立索引的一列或多列中不存在重复的值。在更新行或插入新行的 SQL 语句的结尾强制该约束。若一个或多个列组成的集合已经有重复的值，则不能创建此类型的索引。

关键字 ASC 按列的递增顺序放置这些索引条目，而 DESC 按列的降序放置它们。缺省值为递增顺序。

可以在两个列上创建一个唯一索引，其中的一列是包含列。在非包含列上定义主键。两列都在目录中作为同一表上的主键显示。通常每个表只能有一个主键。

INCLUDE 子句指定将附加列追加到一组索引键列。使用此子句所包含的任何列未用来强制唯一性。包含的列可能会通过仅存取索引而改进某些查询的性能。这些列必须与用来强制唯一性的列区分开（否则将会接收到错误消息 SQLSTATE 42711）。对于列数和总长度属性的限制适用于唯一键和索引中的所有列。

执行检查以确定现有的索引是否与主键定义匹配（忽略索引中的任何 INCLUDE 列）。若索引定义标识同一组列，则索引定义匹配，而不考虑列的顺序或方向（递增顺序或降序顺序）规范。若找到了匹配的索引定义，则更改索引的描述以表示它是主索引（这是系统所要求的），并且若它是非唯一索引，则将其更改为是唯一的（在确保唯一性之后）。

这就是如目录中所表示的同一个表中可能有多个主键的原因。

当使用结构类型时，可能需要创建用户定义的索引类型。这需要一种定义索引维护、索引搜索和索引利用功能的方法。

以下 SQL 语句在 EMPLOYEE 表的 LASTNAME 列上创建一个群集索引，称为 INDEX1:

```
CREATE INDEX INDEX1 ON EMPLOYEE (LASTNAME) CLUSTER
```

要有效地使用数据库的内部存储器，将群集索引与 ALTER TABLE 语句关联的 PCTFREE 参数配合使用，以便可将新数据插入到正确的页上。当将数据插入到正确的页上时，群集顺序保持不变。通常，表上的 INSERT 活动越多，为维护群集所需的（该表上的）PCTFREE 值也就越大。因为此索引确定数据在物理页面上放置的顺序，因此对任何特定的表只能定义一个群集索引。

例如，如果这些新行的索引键值总是新的大键值，则表的群集属性将尝试将其放置在表的末尾。其它页上有可用空间对保持群集没有什么作用。在这种情况下，将表置于追加方式可能优于群集索引，同时改变表以拥有大的 PCTFREE 值。可发出如下命令来将表设置为追加方式：ALTER TABLE APPEND ON。

以上讨论也适用于增加行大小的 UPDATE 所导致的新的“溢出”行。

可以向前或向后方向扫描单个索引，此索引是在 CREATE INDEX 语句上使用 ALLOW REVERSE SCANS 参数创建的。即，这种索引支持以创建索引时定义的方向进行扫描以及以相反方向进行扫描。语句可能类似如下:

```
CREATE INDEX iname ON tname (cname DESC) ALLOW REVERSE SCANS
```

在这种情况下，索引（iname）是根据给定列（cname）中的降序值（DESC）构成的。通过允许逆向扫描，虽然列的索引被定义为以降序扫描，但还是可以递增顺序（逆向）执行扫描。实际是否以两个方向使用索引不是由您控制的，而是由优化器在创建和考虑存取方案时控制的。

CREATE INDEX 语句的 MINPCTUSED 子句指定在索引叶子页上最小已用空间量的阈值。如果使用此子句，则对此索引启用联机索引碎片整理。一旦启用，可参照下列注意事项来确定是否执行联机索引碎片整理：从此索引的叶子页中物理除去键之后，且该页上已使用空间的百分比小于指定的阈值，则检查相邻的索引叶子页以确定是否可将两个叶子页上的键合并到单个索引叶子页中。

例如，以下 SQL 语句创建启用了联机索引碎片整理的索引:

```
CREATE INDEX LASTN ON EMPLOYEE (LASTNAME) MINPCTUSED 20
```

从此索引的索引页物理除去键时，若该索引页的其余键占用索引页上不超过百分之二十的空间，可尝试将此索引页的键与相邻索引页的键合并来删除该索引页。若组合的键可全部位于一页上，则执行此合并并删除其中一个索引页。

CREATE INDEX 语句允许您在创建索引的同时对基础表以及任何先前存在的索引进行读写存取。要在创建索引时限制对表的存取，则使用 LOCK TABLE 语句在创建索引之前锁定该表。新索引是通过扫描基础表创建的。创建索引时，就记录了对表所作的任何更改。一旦创建了新索引，就会将更改应用于该索引。要在创建索引期间更快地应用已记录的更改，在内存缓冲区空间中为更改保留了单独的副本，内存缓冲区空间是从实用程序堆中按需要分配的。这允许索引创建通过首先从内存中直接读取来处理更改，如果需要的话，在较长时间之后通过日志来读取。将所有更改应用于索引之后，在新索引变得可视时会停顿表。

创建唯一索引时，确保表中没有重复的键且创建索引期间的并发插入操作不会引入重复的键。索引创建使用延迟式唯一方案来检测重复的键，因此，在索引创建完全结束之前，检测不到任何重复的键，而最后索引创建将因为重复键而失败。

CREATE INDEX 语句的 PCTFREE 子句指定构建索引时每个索引页中要留作可用空间的百分比。在索引页上保留更多的可用空间将导致更少的页分割。这将减少为重新获得顺序索引页（此操作会增加预取）而重组表的需要。而预取是一个可提高性能的重要组件。此外，如果总是存在大键值，则要考虑减少 CREATE INDEX 语句的 PCTFREE 子句的值。这样会限制每个索引页上保留的浪费的空间。

LEVEL2 PCTFREE 子句指导系统在索引的第二级的每一页上保留指定百分比的可用空间。当创建索引时，指定一定百分比的可用空间来容纳将来的插入和更新。第二级就是刚好在叶级上面的那一级。缺省值是在所有非叶子页中保留最小值 10 和 PCTFREE 值。LEVEL2 PCTFREE 参数允许覆盖缺省值；如果在 CREATE INDEX 语句中使用 LEVEL2 PCTFREE 整数选项，则在第 2 级的中间页上保留整数百分比的可用空间。在第 3 级和更高级别的中间页上保留了最小值 10 和整数百分比的可用空间。通过在第二级保留更多可用空间，在索引的第二级产生的分页数就会减少。

当插入到索引中时，PAGE SPLIT SYMMETRIC、PAGE SPLIT HIGH 和 PAGE SPLIT LOW 子句允许在分页行为中进行选择。

PAGE SPLIT SYMMETRIC 子句是一种缺省分页行为，它大致在索引页的中部进行分割。当随机插入到索引中或者不遵循由 PAGE SPLIT HIGH 和 PAGE SPLIT LOW 子句提出的其中一种模式时，最好是使用这种缺省行为。

当以前在索引中增大了范围时，PAGE SPLIT HIGH 行为是很有用的。在下列情况下可能会发生在索引中增大范围:

- 有一个索引具有多个键部件并且有多个值（相当于多个索引页），而除了最后一个键部件之外的所有键部件都具有相同的值
- 对表进行的所有插入将由新值组成，新值将与除了最后一个键部件之外的所有键部件的现有键具有相同的值
- 所插入的值的最后一个键部件大于现有键的值

例如，如果在索引中具有下列键值;

```
(1,1),(1,2),(1,3), ... (1,n),
(2,1),(2,2),(2,3), ... (2,n),
...
(m,1),(m,2),(m,3), ...(m,n)
```

则要插入的下一个键将具有值 (x,y)，其中 1 <= x <= m 且 y > n。如果插入遵循这样一种模式，则可以使用 PAGE SPLIT HIGH 子句，以便页分割不会导致许多页的百分之五十都是空的。

类似地，当以前在索引中减小了范围时，可以使用 PAGE SPLIT LOW 来避免使许多页的百分之五十都是空的。

注: 如果想要添加主键或唯一键，并且想要底层索引使用 SPLIT HIGH、SPLIT LOW、PCTFREE、LEVEL2 PCTFREE、MINPCTUSED、CLUSTER 或 ALLOW REVERSE SCANS，则必须首先通过指定期望的键和参数来创建索引。然后使用 ALTER TABLE 语句来添加主键或唯一键。ALTER TABLE 语句将检取和重用您已经创建的索引。

可在创建索引时收集索引统计信息。在您使用 CREATE INDEX 语句时，会提供键值统计信息和物理统计信息。通过在执行 CREATE INDEX 语句时收集索引统计信息，您将不必在完成 CREATE INDEX 之后立即运行 RUNSTATS 实用程序。

例如，以下 SQL 语句会在创建索引时收集基本索引统计信息:

```
CREATE INDEX IDX1 ON TABL1 (COL1) COLLECT STATISTICS
```

如果有一个复制的总结表，则其基本表必须具有唯一索引，且该索引键列必须用在定义复制的总结表的查询中。

对于分区内并行性，通过使用多个处理器来处理创建索引期间执行的数据扫描和排序，以提高创建索引的性能。通过将 *intra_parallel* 设置为 YES(1) 或 ANY(-1) 来允许使用多个处理器。在索引创建期间使用的处理器的数目由系统确定，不受配置参数 *dft_degree* 或 *max_querydegree*、应用程序运行时等级或 SQL 语句编译等级影响。

在多分区数据库中，必须将唯一索引定义为分区键的超集。

相关概念:

- 『索引性能技巧』（*《管理指南: 性能》*）
- 『索引重组』（*《管理指南: 性能》*）
- 『标准表的表和索引管理』（*《管理指南: 性能》*）
- 『联机索引整理碎片』（*《管理指南: 性能》*）
- 『MDC 表的表和索引管理』（*《管理指南: 性能》*）

相关任务:

- 第 164 页的『更改表属性』

相关参考:

- 『max_querydegree - 最大查询并行度配置参数』（*《管理指南: 性能》*）
- 『intra_parallel - 启用分区内并行性配置参数』（*《管理指南: 性能》*）
- 『dft_degree - 缺省并行度配置参数』（*《管理指南: 性能》*）
- 『CREATE INDEX statement』（*SQL Reference, Volume 2*）

创建用户定义扩展索引类型

为了支持用户定义的索引类型，DB2® 通用数据库允许您对决定索引工作方式的主要组件创建并应用您自己的逻辑。那些可以替换的组件包括:

- 索引维护。这使得可以将索引列内容映射至索引键。这样的映射是通过用户定义的映射函数完成的。扩展索引中只能有一个结构化类型列。与普通索引不同，扩展索引每行可以有多个索引条目。每行有多个索引条目允许文本文档作为特定对象存储，对于该对象，文档中的每个关键字都有单独的索引条目。
- 索引利用。这使应用程序设计者能够将过滤条件（范围谓词）与用户定义的函数（UDF）相关联（否则，这些用户定义的函数对优化器而言将是不透明的）。这使 DB2 能够避免对每一行进行单独的 UDF 调用，因而避免了客户机与服务器之间的上下文相关切换，从而大幅改进性能。

注: 用户定义的函数定义必须是确定的，并且一定不能允许外部操作，这样才能由优化器利用。

还可指定可选的数据过滤函数。在对用户定义的函数求值之前，优化器对取到的元组使用过滤器。

只有结构化类型或单值类型列才能使用索引扩展来对这些对象创建用户定义扩展索引类型。用户定义扩展索引类型一定不能:

- 使用群集索引定义
- 具有 INCLUDE 列

相关概念:

- 第 126 页的『有关索引维护的详细信息』
- 第 126 页的『有关索引搜索的详细信息』
- 第 127 页的『有关索引利用的详细信息』
- 第 128 页的『定义索引扩展的方案』

有关创建用户定义的扩展索引类型的详细信息

本节讨论创建自己的扩展索引类型时所需的各个方面。

有关索引维护的详细信息

通过 CREATE INDEX EXTENSION 语句定义构成索引操作的其中两个组件。

索引维护是这样的一个过程: 将索引列内容（或源键）变换为目标索引键。变换过程由数据库中先前定义的表函数定义。

FROM SOURCE KEY 子句为此索引扩展所支持的源键列指定结构化数据类型或单值类型。给出了单个参数名和数据类型，它们与源键列相关联。

GENERATE KEY USING 子句指定用来生成索引键的用户定义表函数。必须在 TARGET KEY 子句规范中指定此函数的输出。也可将此函数的输出用作 FILTER USING 子句上指定的索引过滤函数的输入。

相关概念:

- 第 125 页的『创建用户定义扩展索引类型』

相关参考:

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

有关索引搜索的详细信息

索引搜索将搜索自变量映射至搜索范围。

CREATE INDEX EXTENSION 语句的 WITH TARGET KEY 子句指定目标键参数，它们是 GENERATE KEY USING 子句上指定的用户定义表函数的输出。给出了单个参数名和数据类型，它们与目标键列相关联。此参数与 GENERATE KEY USING 子句的用户定义表函数的 RETURNS 表的列相对应。

SEARCH METHODS 子句引入对该索引定义的一个或多个搜索方法。每个搜索方法都由一个方法名、搜索自变量、一个范围生成函数和一个可选的索引过滤函数组成。每

个搜索方法都定义用户定义的表函数生成底层用户定义的索引的索引搜索范围的方法。此外，每个搜索方法都定义如何通过用户定义的标量函数进一步限定特定搜索范围中的索引条目，以返回单个值。

- WHEN 子句将一个标号与一个搜索方法相关联。此标号是一个 SQL 标识，它与索引利用规则中指定的方法名（可在用户定义的函数的 PREDICATES 子句中找到）相关。给出了一个或多个参数名和数据类型，用作范围函数（包括或不包括索引过滤函数）中的自变量。WHEN 子句指定当 CREATE FUNCTION 语句的 PREDICATES 子句与入局查询相匹配时优化器可以执行的操作。
- RANGE THROUGH 子句指定一个用户定义外部表函数，该函数生成索引键范围。这使优化器能够在索引键落在键范围之外时避免调用相关联的 UDF。
- FILTER USING 子句是一种可选的指定用户定义外部表函数或 CASE 表达式的方法，该函数或表达式用来过滤范围生成函数所返回的索引条目。若索引过滤函数或情况表达式返回的值是 1，则从表中检索与索引条目相对应的行。若返回的值不是 1，则废弃该索引条目。当辅助过滤器的成本比求值原始方法的成本低，且辅助过滤器的选择性相对较低时，此功能非常有价值。

相关概念:

- 第 125 页的『创建用户定义扩展索引类型』
- 第 126 页的『有关索引维护的详细信息』
- 第 127 页的『有关索引利用的详细信息』

相关参考:

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

有关索引利用的详细信息

索引利用在对搜索方法进行求值时发生。

CREATE FUNCTION（外部标量）语句创建一个用户定义谓词，此谓词与对索引扩展定义的搜索方法配合使用。

PREDICATES 子句标识那些使用此函数并有可能利用索引扩展（且有可能使用该谓词的搜索条件的可选 SELECTIVITY 子句）的谓词。若指定 PREDICATES 子句，则必须将函数定义成 DETERMINISTIC，并指定 NO EXTERNAL ACTION。

- WHEN 子句引入谓词中使用比较运算符（=、>、< 及其它）和常量或表达式（使用 EXPRESSION AS 子句）定义的函数的特定使用。当谓词将此函数与同一比较运算符和给定的常量或表达式配合使用时，可使用过滤和索引利用。允许使用常量主要是为了包括布尔表达式（其结果类型是 1 或 0）。对于所有其它情况，EXPRESSION AS 子句是更好的选择。
- FILTER USING 子句标识可用来对结果表执行附加过滤的过滤函数。这是已定义函数（在谓词中使用）的备用且更快速的版本，它减少了为了确定行是否有资格而必须对其执行用户定义谓词的行数。如果索引生成的结果接近用户定义谓词所预期的结果，则此过滤函数的应用程序便可能是多余的。
- 可选择对索引扩展的每个搜索方法定义一组规则，来利用索引。也可以在索引扩展中定义一个搜索方法来描述搜索目标、搜索自变量以及如何使用它们来执行索引搜索。
  - SEARCH BY INDEX EXTENSION 子句标识索引扩展。

– 可选的 EXACT 子句指示索引查找正位于其谓词求值之中。此子句告知数据库在索引查找之后不要应用用户提供的原始谓词函数或过滤函数。若不使用索引查找，则必须应用原始谓词和过滤函数。若不使用 EXACT 子句，则在索引查找之后，应用用户提供的原始谓词。当索引查找返回与 EXACT 谓词相同的结果时，该谓词非常有用。这防止查询执行对从索引查找获取的结果应用用户定义谓词。若只期望索引提供谓词的近似值，则不要指定 EXACT 子句。

– WHEN KEY 子句定义搜索目标。只可对一个键指定一个搜索目标。WHEN KEY 子句后面给出的值标识正在定义的函数的参数名。当命名的参数的值是那些根据指定的索引扩展被索引包括的列时，此子句求值为真。

– USE 子句定义搜索自变量。搜索自变量标识将要使用索引扩展中定义的哪一种方法。此处给定的方法名必须与索引扩展中定义的方法相匹配。一个或多个参数值标识正在定义的函数的参数名，这些参数名一定不能与搜索目标中指定的任何参数名相同。参数值的数目和每个参数的数据类型必须与对索引扩展中该方法定义的参数相匹配。内置和单值数据类型必须精确匹配，且必须在相同的结构类型中。

相关概念:

- 第 125 页的『创建用户定义扩展索引类型』
- 第 126 页的『有关索引维护的详细信息』
- 第 126 页的『有关索引搜索的详细信息』
- 第 128 页的『定义索引扩展的方案』

相关参考:

- 『CREATE FUNCTION (External Scalar) statement』（*SQL Reference, Volume 2*）

定义索引扩展的方案

下面是一个定义索引扩展的方案:

1. （为形状）定义结构化类型。使用 CREATE TYPE 语句来定义一个类型层次结构，形状是超类型，空形状、点、线和多边形是子类型。这些结构化类型建立空间实体的模型。例如，商店的位置是一个点，河流的路径是一条线；而商业区的边界是一个多边形。最小定界矩形（mbr）是一个属性。gtype 属性标识相关联的实体是一个点、一条线还是一个多边形。地理边界通过 numpart、numpoint 和 geometry 属性建立模型。所有其它属性都被忽略，因为此方案对它们不感兴趣。
2. 创建索引扩展。
   - 使用 CREATE FUNCTION 语句来创建用于下列各项的函数：键变换（gridentry）、范围生成（gridrange）和索引过滤（checkduplicate 和 mbroverlap）。
   - 使用 CREATE INDEX EXTENSION 语句来创建所需的其余索引组件。
3. 创建与索引的索引维护组件相对应的键变换。

   ```
   CREATE INDEX EXTENSION iename (parm_name datatype, ...)
         FROM SOURCE KEY (parm_name datatype)
         GENERATE KEY USING table_function_invocation
         ...
   ```

   FROM SOURCE KEY 子句标识键变换的参数和数据类型。GENERATE KEY USING 子句标识一个特定函数，该函数用来映射带有从该函数生成的值的源键。
4. 定义与索引的索引搜索组件相对应的范围生成与索引过滤函数。

```
CREATE INDEX EXTENSION iename (parm_name datatype, ...)
      ...
      WITH TARGET KEY
         WHEN method_name (parm_name datatype, ...)
         RANGE THROUGH range_producing_function_invocation
         FILTER USING index_filtering_function_invocation
```

WITH TARGET KEY 子句标识搜索方法定义。WHEN 子句标识方法名。RANGE THROUGH子句标识用来限制要使用的索引作用域的函数。FILTER USING 子句标识一个特定函数，该函数用来从生成的索引值中消去不必需的项。

注: FILTER USING 子句可以标识一个情况表达式，而不是索引过滤函数。

5. 定义谓词来利用索引扩展。

```
CREATE FUNCTION within (x shape, y shape)
      RETURNS INTEGER
      ...
      PREDICATES
         WHEN = 1
            FILTER USING mbrWithin (x..mbr..xmin, ...)
            SEARCH BY INDEX EXTENSION grid_extension
            WHEN KEY (parm_name) USE method_name(parm_name)
```

PREDICATES 子句引入一个或多个谓词，这些谓词以 WHEN 子句开始。WHEN 子句以一个比较运算符开始指定谓词，然后指定常量或 EXPRESSION AS 子句。FILTER USING 子句标识可用来对结果表执行附加过滤的过滤函数。这是已定义函数（在谓词中使用）的廉价版本，它减少了为了确定有资格的行而必须对其执行用户定义谓词的行数。SEARCH BY INDEX EXTENSION 子句指定发生索引利用的位置。索引利用使用可用来利用索引的索引扩展的搜索方法来定义一组规则。WHEN KEY 子句指定利用规则。利用规则描述搜索目标和搜索自变量以及如何通过搜索方法使用它们来执行索引搜索。

6. 定义过滤器函数。

```
   CREATE FUNCTION mbrWithin (...)
```

创建此处定义的函数是为了在索引扩展的谓词中使用。

为了使查询优化器能够成功地利用创建的索引来改进查询性能，在函数调用上提供了 SELECTIVITY 选项。在您知道谓词可能返回的行的百分比的情况下，可在调用函数时使用 SELECTIVITY 选项来帮助 DB2® 优化器选择更有效的存取路径。

在以下示例中，within 用户定义的函数计算中心和半径（分别根据第一个和第二个参数），并用适当的选择性构建语句字符串:

```
SELECT * FROM customer
   WHERE within(loc, circle(100, 100, 10)) = 1 SELECTIVITY .05
```

在此示例中，指示的谓词（SELECTIVITY .05）过滤掉 customer 表中 95% 的行。

相关概念:

- 第 125 页的『创建用户定义扩展索引类型』
- 第 126 页的『有关索引维护的详细信息』
- 第 126 页的『有关索引搜索的详细信息』
- 第 127 页的『有关索引利用的详细信息』

相关参考:

- 『CREATE INDEX EXTENSION statement』（*SQL Reference, Volume 2*）
- 『CREATE FUNCTION (External Scalar) statement』（*SQL Reference, Volume 2*）

通过命令行处理器调用“配置顾问程序”

先决条件:

已经创建了数据库。

过程:

创建数据库之后，可以使用 AUTOCONFIGURE 命令来调用“配置顾问程序”。即使您在创建数据库时选择 AUTOCONFIGURE 选项，也可以使用此方法。

可以使用 AUTOCONFIGURE 的可用选项来定义几个配置参数的值，并确定这些参数的应用程序的作用域。作用域可为 NONE，表示无值适用；作用域为 DB ONLY，表示只有数据库配置和缓冲池值适用；或者作用域为 DB AND DBM，表示所有参数及其值都适用。

相关概念:

- 『配置参数』（*《管理指南: 性能》*）

相关参考:

- 『AUTOCONFIGURE Command』（*Command Reference*）

第 5 章 改变数据库

本章主要讨论在改变数据库之前必须考虑的事项以及如何改变或删除数据库对象。

改变实例

在实现了一个数据库设计的一段时间之后，可能需要更改数据库设计。您应该重新考虑以前设计中的主要设计问题。

在进行影响整个数据库的更改之前，应该查看所有逻辑和物理设计决策。例如，当改变表空间时，应该查看关于 SMS 或 DMS 存储器类型使用的设计决策。

在管理 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）产品的许可证时，可能会发现需要增加许可证的数目。可使用“控制中心”中的“许可证中心”来检查已安装产品的使用情况，并根据该使用情况来增加许可证数。

尤其应该特别注意下列各项:

- 『更改实例（仅适用于 UNIX）』
- 第 135 页的『更改节点和数据库配置文件』

更改实例（仅适用于 UNIX）

实例都设计为尽可能与以后产品的安装和除去所产生的影响无关。

在大多数情况下，现有实例自动继承或失去要安装或除去的产品功能的存取权。但是，若安装或除去了特定的可执行文件或组件，则现有实例不会自动继承新的系统配置参数或获得所有附加功能的存取权。必须更新该实例。

如果通过安装“程序临时性修订”（PTF）或补丁来更新 DB2® 通用数据库（DB2 UDB），则应使用 **db2iupdt** 命令更新全部现有 DB2 UDB 实例。

在尝试更改或删除实例前，应确保了解那些实例和实例中已有的数据库分区服务器。

相关概念:

- 第 14 页的『实例创建』

相关任务:

- 第 132 页的『在 UNIX 上更新实例配置』
- 第 134 页的『除去实例』

相关参考:

- 『db2iupdt - Update Instances Command』（*Command Reference*）

有关更改实例的详细信息

在更改实例之前，您应该列出全部现有的实例。

列出实例

过程:

要使用“控制中心”获取系统上可用的所有实例的列表:

1. 展开对象树，直到您看到**实例**文件夹为止。
2. 右键单击“实例”文件夹，并从弹出菜单中选择**添加**。
3. 在“添加实例”窗口上，单击**刷新**。
4. 单击下拉箭头来查看数据库实例的列表。
5. 单击**取消**以退出该窗口。

要使用命令行获得系统上可用的所有实例的列表，输入:

```
db2ilist
```

要确定哪一个实例适用于当前会话（在受支持的 Windows 平台上），使用:

```
set db2instance
```

在 UNIX 上更新实例配置

运行 **db2iupdt** 命令，并执行以下操作来更新指定的实例:

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

过程:

db2iupdt 命令可在 AIX 上的 /usr/opt/db2_08_01/instance/ 目录中找到。**db2iupdt** 命令可在 HP-UX、Solaris Operating Environment 或 Linux 上的 /opt/IBM/db2/V8.1/instance/ 目录中找到。

按如下所示使用该命令:

```
db2iupdt InstName
```

InstName 是实例所有者的登录名。

此命令还有其它可选的参数:

- –h 或 –?

  显示此命令的帮助菜单。

- –d

  设置在问题确定期间要使用的调试方式。

- –a AuthType

  指定实例的认证类型。有效的认证类型是 SERVER、SERVER_ENCRYPT 或 CLIENT。若未指定此参数，且若已安装了 DB2 服务器，则缺省值为 SERVER。否则，将它设置为 CLIENT。该实例的认证类型适用于实例拥有的所有数据库。

- –e

  允许更新存在的每个实例。可以使用 **db2ilist** 来显示存在的实例。

- –u Fenced ID

  命名受防护的用户定义的函数（UDF）和存储过程执行期间所归属的用户。若安装了 DB2 客户机或 “DB2 软件开发者工具箱”，则不需要这样做。对于其它 DB2 产品，它是必需参数。

  注: 受防护标识不能是 “root” 或 “bin”。

- –k

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

  – 具有本地和远程客户机的分区数据库服务器（DB2 扩充企业版缺省实例类型）
  – 具有本地和远程客户机的数据库服务器（DB2 通用数据库企业服务器版缺省实例类型）
  – 客户机（DB2 客户机缺省实例类型）

示例:

- 如果在创建实例后安装了 DB2 通用数据库工作组服务器版或 DB2 通用数据库企业服务器版，可输入以下命令来更新该实例:

  ```
  db2iupdt -u db2fenc1 db2inst1
  ```

- 若在创建实例后安装了 DB2 Connect 企业版，也可将该实例用作 Fenced ID:

  ```
  db2iupdt -u db2inst1 db2inst1
  ```

- 要更新客户机实例，可使用以下命令:

  ```
  db2iupdt db2inst1
  ```

相关任务:

- 第 134 页的『除去实例』

相关参考:

- 『db2ilist - List Instances Command』（*Command Reference*）
- 『db2iupdt - Update Instances Command』（*Command Reference*）

在 Windows 上更新实例配置

运行 **db2iupdt** 命令，并执行以下操作来更新指定的实例:

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

过程:

db2iupdt 命令可在 `\sqllib\bin` 目录中找到。

按如下所示使用该命令:

```
db2iupdt InstName
```

InstName 是实例所有者的登录名。

此命令还有其它可选的参数:

- /h: hostname

  在当前机器上存在一个或多个 TCP/IP 主机名的情况下重设缺省 TCP/IP 主机名。
- /p: instance profile path

  为已更新实例指定新的实例概要文件路径。
- /r: baseport,endport

  指定运行多分区时，分区数据库实例使用的 TCP/IP 端口范围。
- /u: username,password

  指定 DB2 服务的帐户名和密码。

相关任务:

- 第 21 页的『列出实例』
- 第 132 页的『在 UNIX 上更新实例配置』
- 第 134 页的『除去实例』

除去实例

过程:

使用“控制中心”来除去实例:

1. 展开对象树，直到您看到要除去的实例为止。
2. 右键单击该实例名，并从弹出菜单中选择**除去**。
3. 选择**确认**框，并单击**确定**。

要使用命令行除去实例，输入:

```
db2idrop <instance_name>
```

使用命令行除去实例的准备工作和详细信息包括:

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

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

```
db2idrop InstName
```

其中 InstName 是要删除的实例的名称。

此命令从实例列表中除去该实例条目并除去该实例目录。

8. （仅在 UNIX 操作系统上）可选择作为具有 root 用户权限的用户，除去该实例所有者的用户标识和组（若只用于该实例）。若计划重新创建该实例，则不要除去它们。

   此步骤是可选的，因为实例所有者和实例所有者组可用于其它用途。

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

注: 在 UNIX 操作系统上，试图使用 db2idrop 命令删除实例时，会生成一条消息，说明不能除去 sqllib 子目录，并且正在 adm 子目录中生成具有 .nfs 扩展名的几个文件。adm 子目录是安装了 NFS 的系统，而这些文件在服务器上是受控的。必须从安装目录的文件服务器中删除 *.nfs 文件。然后可除去 sqllib 子目录。

相关参考:

- 『db2stop - Stop DB2 Command』（*Command Reference*）
- 『TERMINATE Command』（*Command Reference*）
- 『STOP DATABASE MANAGER Command』（*Command Reference*）
- 『db2idrop - Remove Instance Command』（*Command Reference*）
- 『db2ilist - List Instances Command』（*Command Reference*）

更改节点和数据库配置文件

要更新数据库配置文件，使用“控制中心”中的“配置顾问程序”或者在带有适当选项的情况下运行 *db2 autoconfigure*。“配置顾问程序”通过建议修改某些配置参数并为它们提供建议值来帮助调整性能和平衡每个实例中单个数据库的内存需求。

注: 若修改任何参数，则在发生下列各项之前，不更新值:

- 对于数据库参数，在与所有应用程序断开连接之后，与数据库建立第一个新连接时
- 对于数据库管理器参数，下一次停止和启动该实例时

在大多数情况下，“配置顾问程序”所建议的值将比缺省值提供更好的性能，因为它们是根据有关工作负载和您自己的特定服务器的信息确定的。但是，这些值是为提高数据库系统的性能而设计的，并不一定能优化该系统。应该将这些值当作一个起始点，然后进一步调整以获得优化的性能。

先决条件:

如果打算更改任何数据库分区组（添加或删除分区或者除去现有分区），则必须更新节点配置文件。

如果打算更改数据库，则应查看配置参数的值。可以定期调整某些值，作为根据数据库的使用方式更改数据库的一部分。

过程:

要使用“控制中心”来更新数据库配置:

1. 展开对象树，直到您看到**数据库**文件夹为止。
2. 右键单击想要更改的实例或数据库，并单击**配置顾问程序**。
3. 单击每一页，并根据需要更改信息。
4. 单击**结果**页以查看对配置参数所作的任何建议更改。
5. 当您可以开始应用或保存更新时，单击**完成**。

要从命令行使用“配置顾问程序”，使用 AUTOCONFIGURE 命令。

要使用命令行来更新数据库管理器配置中的各个参数，输入:

```
UPDATE DBM CFG FOR <database_alias>
   USING <config_keyword>=<value>
```

可以在单个命令中更新一个或多个 <config_keyword>=<value> 组合。对数据库管理器配置文件的大多数更改只有在将它们装入内存之后才会生效。对于服务器配置参数，这在运行 START DATABASE MANAGER 命令期间发生。对于客户机配置参数，这在重新启动应用程序时发生。

要查看或打印当前数据库管理器配置参数，使用 GET DATABASE MANAGER CONFIGURATION 命令。

相关概念:

- 『基准程序测试』（*《管理指南: 性能》*）

相关任务:

- 第 136 页的『更改多分区中的数据库配置』
- 『用配置参数配置 DB2』（*《管理指南: 性能》*）

相关参考:

- 『GET DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）
- 『UPDATE DATABASE MANAGER CONFIGURATION Command』（*Command Reference*）

更改多分区中的数据库配置

过程:

当数据库分布在多分区上时，数据库配置文件在所有数据库分区上应是相同的。一致性是必需的，因为 SQL 编译器根据节点配置文件中的信息来编译分布式 SQL 语句，并创建一个存取方案以满足 SQL 语句的需要。根据预编译该语句的数据库分区，维护数据库分区上的不同配置文件可能产生不同的存取方案。使用 **db2_all** 来跨所有数据库分区维护配置文件。

相关概念:

- 第 317 页的『在分区数据库环境中发出命令』

相关任务:

- 第 135 页的『更改节点和数据库配置文件』

改变数据库

改变数据库的任务几乎与创建数据库的任务一样多。这些任务更新或删除先前创建的数据库的组件。这些任务包括:

- 『删除数据库』
- 第 138 页的『改变数据库分区组』
- 第 138 页的『改变表空间』
- 第 146 页的『删除模式』
- 第 149 页的第 6 章, 『改变表和其它相关表对象』
- 第 168 页的『改变用户定义的结构化类型』
- 第 168 页的『删除和更新类型表的行』
- 第 169 页的『重命名现有表或索引』
- 第 171 页的『删除表』
- 第 172 页的『删除用户定义的临时表』
- 第 172 页的『删除触发器』
- 第 173 页的『删除用户定义的函数（UDF）、函数映射或方法』
- 第 174 页的『删除用户定义的类型（UDT）或类型映射』
- 第 174 页的『改变或删除视图』
- 第 176 页的『恢复不可用视图』
- 第 176 页的『删除具体查询表或分级表』
- 第 177 页的『恢复不可用总结表』
- 第 178 页的『删除索引、索引扩展或索引规范』
- 第 179 页的『更改对象时的语句相关性』

删除数据库

过程:

虽然数据库中的某些对象可以改变，但是数据库本身不能改变：必须删除它，然后重新创建。因为删除数据库会删除它的所有对象、容器和相关的文件，所以此操作可能产生巨大的影响。删除的数据库从数据库目录中除去（取消编目）。

要使用“控制中心”删除数据库:

1. 展开对象树，直到您看到**数据库**文件夹为止
2. 右键单击要删除的数据库，并从弹出菜单中选择**删除**。
3. 单击**确认**框，并单击**确定**。

要使用命令行来删除数据库，输入:

```
DROP DATABASE <name>
```

以下命令删除数据库 SAMPLE:

```
DROP DATABASE SAMPLE
```

注: 若打算继续用 SAMPLE 数据库做实验，则不应将其删除。若已删除 SAMPLE 数据库而又发现需要它，可重新创建它。

相关参考:

- 『GET SNAPSHOT Command』（*Command Reference*）
- 『DROP DATABASE Command』（*Command Reference*）
- 『LIST ACTIVE DATABASES Command』（*Command Reference*）

改变数据库分区组

过程:

一旦添加或删除了分区，就必须将当前数据再分发至数据库分区组中的新分区集合。为此，使用 REDISTRIBUTE DATABASE PARTITION GROUP 命令。

相关概念:

- 『数据再分发』（*《管理指南: 性能》*）
- 『数据库服务器容量的管理』（*《管理指南: 性能》*）

相关任务:

- 『在分区之间再分发数据』（*《管理指南: 性能》*）

相关参考:

- 『REDISTRIBUTE DATABASE PARTITION GROUP Command』（*Command Reference*）

改变表空间

过程:

当创建一个数据库时，至少要创建三个表空间：一个目录表空间（SYSCATSPACE）；一个用户表空间（缺省名称是 USERSPACE1）以及一个系统临时表空间（缺省名称是 TEMPSPACE1）。必须至少使这三种表空间各有一个。您可以在您希望的情况下添加其他用户和临时表空间。

注: 不能删除目录表空间 SYSCATSPACE，也不能创建另外一个；且必须始终至少存在一个页大小为 4 KB 的系统临时表空间。可以创建其它系统临时表空间。在创建表空间之后，您也不能更改它的页大小或扩展数据块大小。

相关任务:

- 第 139 页的『将容器添加至 DMS 表空间』
- 第 140 页的『修改 DMS 表空间中的容器』
- 第 142 页的『将容器添加至分区上的 SMS 表空间』
- 第 143 页的『重命名表空间』
- 第 144 页的『删除用户表空间』
- 第 144 页的『删除系统临时表空间』
- 第 145 页的『删除用户临时表空间』

相关参考:

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

改变表空间的详细信息

此部分描述与改变表空间相关联的那些任务。

将容器添加至 DMS 表空间

过程:

可以通过将一个或多个容器添加至该表空间来增加 DMS 表空间（即，使用 MANAGED BY DATABASE 子句创建的）的大小。

将新容器添加至表空间或扩展现有容器时，会发生表空间重新平衡。重新平衡过程涉及将表空间从一个位置移至另一位置。在此过程中，将试图在表空间内分割数据。重新平衡不必在所有容器上进行，但这取决于许多因素，例如现有容器配置，新容器的大小和表空间满的程度。

将容器添加至现有表空间时，可能不会从组合分割区 0 开始添加它们。在映射中的什么位置开始添加它们是由数据库管理器确定的，并取决于正添加的容器的大小。如果要添加的容器不够大，则它的放置位置可能是结束于映射的最后一个组合分割区。如果要添加的容器足够大，则它的位置会从组合分割区 0 开始。

如果正在添加新的容器且创建新的分割集，则不会发生重新平衡。新的分割集是在 ALTER TABLESPACE 语句上使用 BEGIN NEW STRIPE SET 子句创建的。还可以在 ALTER TABLESPACE 语句上使用 ADD TO STRIPE SET 子句将容器添加至现有分割集。

在重新平衡期间，不限制对该表空间的存取。若需要添加多个容器，应该同时添加它们。

要使用“控制中心”来向 DMS 表空间添加容器:

1. 展开对象树，直到您看到**表空间**文件夹为止。
2. 右键单击要添加容器的表空间，并从弹出菜单中选择**改变**。
3. 单击**添加**，填写信息，并单击**确定**。

要使用命令行来向 DMS 表空间添加容器，输入:

```
ALTER TABLESPACE <name>
   ADD (DEVICE '<path>' <size>, FILE '<filename>' <size>)
```

以下示例举例说明如何将两个新设备容器（各含 10 000 页）添加至 UNIX 系统上的一个表空间中:

```
ALTER TABLESPACE RESOURCE
      ADD (DEVICE '/dev/rhd9'  10000,
           DEVICE  '/dev/rhd10' 10000)
```

注意，ALTER TABLESPACE 语句允许更改可以影响性能的表空间的其它属性。

相关概念:

- 『表空间对查询优化的影响』（*《管理指南: 性能》*）

- 『在 DMS 表空间中如何添加和扩展容器』（*《管理指南: 计划》*）

相关任务:

- 第 142 页的『将容器添加至分区上的 SMS 表空间』

相关参考:

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

修改 DMS 表空间中的容器

限制:

只能将每个裸设备用作一个容器。创建了裸设备之后，其大小是固定的。当您考虑使用调整大小或扩展选项来增大裸设备容器时，应先检查裸设备大小以确保您并未试图将设备容器大小增大为大于裸设备大小。

过程:

可调整 DMS 表空间（即使用 MANAGED BY DATABASE 子句创建的表空间）中容器的大小。

要使用“控制中心”来增加 DMS 表空间中一个或多个容器的大小:

1. 展开对象树，直到您看到**表空间**文件夹为止。
2. 右键单击要添加容器的表空间，并从弹出菜单中选择**改变**。
3. 单击**调整大小**，完成信息，并单击**确定**。

还可以从 DMS 表空间中删除现有容器，减少 DAS 表空间中现有容器的大小以及将新容器添加至 DMS 表空间而不需要在所在容器间重新平衡数据。

仅当正在删除或缩小其大小的数据块数目小于或等于表空间中上限标记之上的可用数据块数目时，才允许删除现有表空间容器以及缩小现有容器的大小。上限标记是表空间中分配的最高页的页数。此标记与表空间中已使用的页的数目不同，原因是上限标记之下的某些数据块可能可供重新使用。

表空间中上限标记之上的可用数据块数非常重要，原因是直至上限标记（包括上限标记）的所有数据块必须位于表空间内的同一逻辑位置。结果表空间必须有足够的空间才能容纳所有数据。如果没有足够的可用空间，则会产生一条错误消息（SQL20170N 或 SQLSTATE 57059）。

要删除容器，可在 ALTER TABLESPACE 语句上使用 DROP 选项。例如:

```
ALTER TABLESPACE TS1 DROP (FILE 'file1', DEVICE '/dev/rdisk1')
```

要缩小现有容器的大小，可使用 RESIZE 选项或 REDUCE 选项。使用 RESIZE 选项时，作为语句的一部分列示的所有容器都必须增大大小或减小大小。不能在同一语句中增大某些容器而缩小其它容器。如果知道容器大小的新下限，应考虑调整大小方法。如果不知道（或不关心）容器的当前大小，则应该考虑缩小方法。

要使用命令行来缩小 DMS 表空间中一个或多个容器的大小，输入:

```
ALTER TABLESPACE <name>
REDUCE (FILE '<filename>' <size>)
```

以下示例说明如何在基于 Windows 的系统上的表空间中缩小文件容器（含 1 000 页）：

```
ALTER TABLESPACE PAYROLL
  REDUCE (FILE 'd:\hldr\finance'  200)
```

在此操作之后，文件大小就从 1 000 页减少至 800 页。

要使用命令行来增大 DMS 表空间中一个或多个容器的大小，输入:

```
ALTER TABLESPACE <name>
   RESIZE (DEVICE '<path>' <size>)
```

以下示例说明如何在基于 UNIX 的系统上的表空间中增大两个设备容器（各含 1 000 页）：

```
ALTER TABLESPACE HISTORY
  RESIZE (DEVICE '/dev/rhd7'  2000,
          DEVICE  '/dev/rhd8' 2000)
```

在此操作之后，两个设备的大小都从 1 000 页增加至 2 000 页。在容器间重新平衡该表空间的内容。在重新平衡期间，不限制对该表空间的存取。

要使用命令行来扩展 DMS 表空间中一个或多个容器，输入:

```
ALTER TABLESPACE <name>
   EXTEND (FILE '<filename>' <size>)
```

以下示例说明如何在基于 Windows 的系统上的表空间中增大文件容器（各含 1 000 页）：

```
ALTER TABLESPACE PERSNEL
  EXTEND (FILE 'e:\wrkhist1'  200
          FILE  'f:\wrkhist2'  200)
```

在此操作之后，两个文件的大小都从 1 000 页增大至 1 200 页。在容器间重新平衡该表空间的内容。在重新平衡期间，不限制对该表空间的存取。

通过预取程序以并行方式执行 DMS 容器（文件容器和裸设备容器），这些容器是在表空间创建期间或之后添加的，或者是在表空间创建之后扩展的。要增加这些创建或调整容器大小操作的并行性，可以增加系统中运行的预取程序的数目。不以并行方式执行的唯一进程是记录这些操作以及在创建容器的情况下标记这些容器。

注: 要使 CREATE TABLESPACE 或 ALTER TABLESPACE 语句的并行性最大（对于将新的容器添加至现有的空间），确保预取程序数大于或等于要添加的容器数。预取程序数目由 *num_ioservers* 数据库配置参数控制。必须停止数据库以使新参数值生效。也就是说，必须断开所有应用程序和用户与数据库的连接以使更改生效。

注意，ALTER TABLESPACE 语句允许更改可以影响性能的表空间的其它属性。

相关参考:

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

添加或删除容器后自动调整预取大小

如果在添加或删除容器后可能会忘记更新表空间的预取大小，则应考虑允许数据库管理器自动确定预取大小。如果忘记更新预取大小，则数据库性能可能会明显降低。

设置 DB2® 通用数据库（DB2 UDB），以便自动预取大小成为使用版本 8.2（和更新版本）创建的表空间的缺省值。数据库管理器使用以下公式计算表空间的预取大小:

```
预取大小 =（容器数）X（每个容器的物理主轴数）
                    X 扩展数据块大小
```

要使表空间的预取大小不设置为 AUTOMATIC 有三种方法:

- 使用特定的预取大小创建表空间。手工选择预取大小值表示: 一旦与表空间关联的容器数出现调整，如有必要，记得调整预取大小。
- 创建表空间时，请勿使用预取大小，并将 *dft_prefetch_sz* 数据库配置参数设置为非 AUTOMATIC 值。在创建表空间时如果未显式提及预取大小，则 DB2 UDB 将检查此参数。如果发现除 AUTOMATIC 之外的值，则该值是用作缺省预取大小的值。一旦与表空间关联的容器数出现调整，则需要记得调整预取大小（如有必要）。
- 使用 ALTER TABLESPACE 语句手工改变预取大小。

使用 DB2_PARALLEL_IO

根据表空间的并行性，将预取请求分解为多个较小的预取请求，然后将请求提交至预取队列。使用 DB2_PARALLEL_IO 注册表变量来定义每个容器的物理主轴数以及对表空间上的并行 I/O 的影响。如果已禁用并行 I/O，则表空间的并行性与容器数相等。如果已启用并行 I/O，则表空间的并行性等于容器数乘以 DB2_PARALLEL_IO 注册表变量中给定的值。（换而言之，表空间的并行性等于预取大小除以表空间扩展数据块大小后的值。）

以下是 DB2_PARALLEL_IO 注册表变量如何影响预取大小的若干示例。（假设已使用 AUTOMATIC 预取大小定义以下所有表空间。）

- DB2_PARALLEL_IO=*
  - 所有表空间将使用每个容器主轴数等于 6 的缺省值。预取大小比启用并行 I/O 时大 6 倍。
  - 所有表空间均会启用并行 I/O。预取请求分解成多个较小请求，每个请求等于预取大小除以扩展数据块大小后的值（或等于容器数乘以主轴数）。
- DB2_PARALLEL_IO=*:3
  - 所有表空间将 3 作为每个容器的主轴数。
  - 所有表空间均会启用并行 I/O。
- DB2_PARALLEL_IO=*:3,1:1
  - 所有表空间将 3 作为每个容器的主轴数，表空间 1 除外，此表空间将使用 1。
  - 所有表空间均会启用并行 I/O。

相关任务:

- 第 138 页的『改变表空间』
- 第 139 页的『将容器添加至 DMS 表空间』
- 第 140 页的『修改 DMS 表空间中的容器』

将容器添加至分区上的 SMS 表空间

限制:

您只能将容器添加至当前没有任何容器的分区上的 SMS 表空间。

过程:

要使用命令行来向 SMS 表空间添加容器，输入以下内容:

```
ALTER TABLESPACE <name>
      ADD ('<path>')
      ON DBPARTITIONNUM (<partition_number>)
```

按编号指定的分区和分区范围内的每个分区（或节点）必须在定义表空间的数据库分区组中存在。partition_number 可能仅在语句的一个 *db-partitions 子句*中显式出现或在某个范围内出现。

以下示例显示如何在基于 UNIX 的操作系统上将新容器添加至由表空间 “plans” 使用的数据库分区组的第 3 号分区:

```
ALTER TABLESPACE plans
   ADD ('/dev/rhdisk0')
   ON DBPARTITIONNUM (3)
```

相关任务:

- 第 139 页的『将容器添加至 DMS 表空间』
- 第 140 页的『修改 DMS 表空间中的容器』

相关参考:

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

重命名表空间

限制:

不能重命名 SYSCATSPACE 表空间。

不能重命名处于 “前滚暂挂” 或 “正在前滚” 状态的表空间。

当复原在备份后已被重命名的表空间时，必须在 RESTORE DATABASE 命令中使用新的表空间名。若使用先前表空间名，则将找不到它。同样，若使用 ROLLFORWARD DATABASE 命令前滚该表空间，则确保使用新名称。若使用先前表空间名，则将找不到它。

过程:

可以给予现有表空间新名称，而无需关心该表空间中的各个对象。重命名表空间时，将更改所有引用该表空间的目录记录。

相关参考:

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

切换表空间的状态

过程:

若与表空间相关的容器已变得可存取，可以使用 ALTER TABLESPACE 语句的 SWITCH ONLINE 子句从表空间中除去 OFFLINE 状态。表空间已除去 OFFLINE 状态，而数据库的其余部分仍在运行并在使用中。

使用此子句的另一种方法是将所有应用程序与数据库断开连接，然后再次将这些应用程序连接至数据库。这样就可以从表空间中除去 OFFLINE 状态。

要使用命令行从表空间中除去 OFFLINE 状态，输入:

```
db2 ALTER TABLESPACE <name>
   SWITCH ONLINE
```

相关参考:

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

删除用户表空间

过程:

当删除用户表空间时，也会删除该表空间中的所有数据，释放容器，除去目录条目，并导致该表空间中定义的所有对象都被删除或标记为无效。

可以通过删除表空间来重新使用该空的表空间中的容器，但是，在试图重新使用这些容器之前，必须落实该 DROP TABLESPACE 命令。

可删除一个包含所有表数据的用户表空间，包括在该单个用户表空间中的索引和 LOB 数据。也可删除所包含的表跨几个表空间的一个用户表空间。即，在一个表空间中可能有数据，在另一个表空间中有索引，而在第三个表空间中有任何 LOB。必须在一条语句中同时删除所有三个表空间。包含跨越的表的所有表空间必须全部纳入此单条语句中，否则该删除请求将失败。

要使用“控制中心”来删除用户表空间:

1. 展开对象树，直到您看到**表空间**文件夹为止。
2. 右键单击要删除的表空间，并从弹出菜单中选择**删除**。
3. 选择**确认**框，并单击**确定**。

要使用命令行来删除用户表空间，输入:

```
DROP TABLESPACE <name>
```

以下 SQL 语句将删除表空间 ACCOUNTING:

```
DROP TABLESPACE ACCOUNTING
```

相关任务:

- 第 144 页的『删除系统临时表空间』
- 第 145 页的『删除用户临时表空间』

相关参考:

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

删除系统临时表空间

限制:

如果不首先创建另一系统临时表空间，则不能删除页大小为 4 KB 的系统临时表空间。新的系统临时表空间必须具有 4 KB 页大小，原因是数据库必须始终存在至少一个具有 4 KB 页大小的系统临时表空间。例如，如果希望单个系统表空间具有 4 KB 页大小，并且想要将一个容器添加至该表空间且它是一个 SMS 表空间，则必须先添加具有适当数目的容器的新 4 KB 页大小的系统临时表空间，再删除旧的系统临时表空间。（如果正在使用 DMS，则可以添加容器而不必删除并重新创建表空间。）

过程:

缺省表空间页大小为 4 KB。

要使用“控制中心”来删除系统表空间:

1. 展开对象树，直到您看到**表空间**文件夹为止。
2. 如果只有一个其它系统临时表空间，则右键单击**表空间**文件夹，并从弹出菜单中选择**创建 -> 使用向导创建表空间**。否则，跳过步骤 4。
3. 遵循向导中的步骤，以创建新的系统临时表空间（若需要的话）。
4. 再次单击**表空间**文件夹，以在窗口右边（“内容”窗格）显示表空间列表。
5. 右键单击要删除的系统临时表空间，并从弹出菜单中单击**删除**。
6. 选择**确认**框，并单击**确定**。

这是用来创建系统临时表空间的语句:

```
CREATE SYSTEM TEMPORARY TABLESPACE <name>
   MANAGED BY SYSTEM USING ('<directories>')
```

之后，要使用命令行来删除系统表空间，输入:

```
DROP TABLESPACE <name>
```

以下 SQL 语句创建一个称为 TEMPSPACE2 的新的系统临时表空间:

```
CREATE SYSTEM TEMPORARY TABLESPACE TEMPSPACE2
      MANAGED BY SYSTEM USING ('d:\systemp2')
```

一旦创建了 TEMPSPACE2，则可使用以下命令删除原来的系统临时表空间 TEMPSPACE1:

```
DROP TABLESPACE TEMPSPACE1
```

可以通过删除表空间来重新使用该空的表空间中的容器，但是，在试图重新使用这些容器之前，必须落实该 DROP TABLESPACE 命令。

相关任务:

- 第 144 页的『删除用户表空间』
- 第 145 页的『删除用户临时表空间』

相关参考:

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

删除用户临时表空间

过程:

仅当用户临时表空间中当前未定义已声明临时表时，才能删除该表空间。当删除表空间时，不会尝试删除该表空间中的所有已声明临时表。

注: 已声明临时表是在说明它的应用程序与数据库断开连接时隐式删除的。

相关任务:

- 第 144 页的『删除用户表空间』
- 第 144 页的『删除系统临时表空间』

相关参考:

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

删除模式

过程:

在删除模式之前，必须删除该模式中的所有对象或将它们移至另一个模式。当尝试 DROP 语句时，该模式名必须在目录中；否则会返回错误。

要使用“控制中心”删除模式:

1. 展开对象树，直到您看到**模式**文件夹为止。
2. 右键单击要删除的模式，并从弹出菜单中选择**删除**。
3. 选择**确认**框，并单击**确定**。

要使用命令行来删除模式，输入:

```
DROP SCHEMA <name>
```

在以下示例中，删除了模式“joeschma”:

```
DROP SCHEMA joeschma RESTRICT
```

RESTRICT 关键字强制执行一个规则，即不能在指定的模式中为要从数据库中删除的模式定义对象。

相关参考:

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

改变缓冲池

当使用现有缓冲池时，可能需要完成下列其中一个任务:

- 修改所有分区或单一分区上的缓冲池大小。
- 启用或禁用扩充存储器的使用。
- 将此缓冲池定义添加到新的数据库分区组中。
- 修改基于块的 I/O 的缓冲池的块区域。

先决条件:

语句的授权标识必须具有 SYSCTRL 或 SYSADM 权限。

过程:

1. 使用 SELECT BPNAME FROM SYSCAT.BUFFERPOOLS 来获取数据库中已存在的缓冲池名称的列表。
2. 从结果列表中选择缓冲池名称。
3. 确定需要进行哪些更改。
4. 确保您具有正确的授权标识来运行 ALTER BUFFERPOOL 语句。

   注: 两个关键参数是 IMMEDIATE 和 DEFERRED。当使用 IMMEDIATE 参数时，将立即更改缓冲池大小。如果数据库共享内存中保留的空间不足以分配新的空间，则会延迟运行该语句。

   当使用 DEFERRED 参数时，将在与那些应用程序断开连接之后再重新激活数据库时才会对缓冲池进行高速缓存。不需要保留内存空间；DB2 UDB 在激活时从系统中分配必需的内存。
5. 使用 ALTER BUFFERPOOL 语句来改变缓冲池对象的单个特性。

相关任务:

• 第 60 页的『创建缓冲池』

相关参考:

• 『ALTER BUFFERPOOL statement』（*SQL Reference, Volume 2*）

第 6 章 改变表和其它相关表对象

修改表和相关表对象的结构与内容所需的任务包括下列各项:

- 『对现有表的空间压缩』
- 第 152 页的『将列添加至现有表』
- 第 152 页的『修改列定义』
- 第 153 页的『从表或视图除去行』
- 第 170 页的『使用 MERGE 语句更新表和视图内容』
- 第 155 页的『修改标识列定义』
- 第 155 页的『改变约束』
- 第 161 页的『对现有表定义生成列』
- 第 163 页的『将表声明为易失的』
- 第 164 页的『更改分区键』
- 第 164 页的『更改表属性』
- 第 167 页的『改变具体查询表属性』
- 第 167 页的『刷新具体查询表中的数据』

注意，不能改变表的触发器；必须删除任何不再适合的触发器（请参阅第 172 页的『删除触发器』），然后添加它的替换项目（请参阅第 100 页的『创建触发器』）。

修改现有表及其相关表对象

对现有表的空间压缩

可将现有表更改为允许空间压缩的记录格式。只要字节计数不超出表空间中的表行的可允许长度，允许空间压缩的记录格式中列的字节计数之和可能会超出原始记录格式（不允许空间压缩）中列的字节计数之和。例如，在具有 4 KB 页大小的表空间中可允许的行长度是 4005 字节。如果超出了可允许行长度，则会返回错误消息 SQL0670N。字节计数公式在 CREATE TABLE 语句中作了说明。

类似地，可将现有表从允许空间压缩的记录格式更改为不允许空间压缩的记录格式。此情况对于列的字节计数之和同样适用；必要时返回错误消息 SQL0670N。

要确定是否应考虑对表进行空间压缩，应了解大多数值等于系统缺省值或 NULL 值的表将受益于新的行格式。例如，假设有一个 INTEGER 列且列的 90% 的列值为 0（INTEGER 数据类型的缺省值）或 NULL，压缩此表和此列将受益于新的行格式并节省大量的磁盘空间。

改变表时，可使用 VALUE COMPRESSION 子句来指定表在表级别也可能在列级别使用空间行格式。应使用 ACTIVATE VALUE COMPRESSION 来指定表将对表中的数据使用节省空间技术，或使用 DEACTIVATE VALUE COMPRESSION 来指定表将不再对表中数据使用节省空间技术。

如果使用 DEACTIVATE VALUE COMPRESSION，这将显式禁用与该表中的列相关联的所有 COMPRESS SYSTEM DEFAULT 选项。

在将表修改为新的行格式后，插入、装入或更新的所有后续行将使用新的行格式。要将每一行都修改为新的行格式，应在更改行格式之前运行对表的重组或对现有行执行更新操作。

相关概念:

- 第 81 页的『新表的空间压缩』

相关参考:

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

使用存储过程改变表

表是存储所有业务数据的位置。创建数据库之前，必须考虑要保留在数据库中的数据的类型和组织。需要进行许多规划，以确保考虑使用和处理您和您的业务可能需要的所有相关数据。但是，情况会有所变化。尽管进行出色的规划，但仍可能出现新需求或业务变更的情况，因此需要对数据库中的表作出更改。

您可能发现需要在表内以下列一种或多种方式进行更改:

- 重命名列
- 除去列
- 使用 SQL 标量函数改变列类型并变换现有数据
- 增加或减少列大小
- 更改列的缺省值
- 将列从 NOT NULL 更改为 NULLABLE
- 更改十进制的精度和小数位

进行这些类型的更改时，需要最大程度地减少原始表数据丢失的风险。DB2® 通用数据库（DB2 UDB）提供用户界面和存储过程，它们将允许您改变表。在明确指示全部所需改变表工作已完成之前，不会删除原始表及其关联的数据。

从用户界面调用的每个存储过程调用均执行一系列操作，例如删除、重新创建、装入数据以完成上面列示的操作。

对表中内容的改变存在限制。这些限制包括:

- 不支持改变具体查询表（MQT）。

  但是，支持改变拥有 MQT 的表。此外，在 ALTER TABLE 过程中，不会刷新（填充）已改变的基本表中定义的 MQT。在 MQT 中，ALTOBJ() 存储过程改变其基本表的同时，由于 MQT 内容完全从新的基本表重新构建，因此所有并非基本表的选择结果一部分的列丢失。
- 不支持改变类型表或任何现有引用列类型表范围内的表。
- 不支持使用昵称改变远程表。
- 不能对表内的列顺序重新排序。
- 添加和重命名与删除列操作互斥。

即这些列操作不能同时存在于单个改变表调用中。

- 不支持 DATALINK 数据类型。
- 由于无持久的对象锁定，所以对象的定义可能在 ALTOBJ() 调用之间更改。
- 经过 ALTER TABLE 过程后，与表压缩描述符关联的表概要文件（如 runstats 概要文件）已丢失。
- 在任何给定时候，每个表仅支持一个 ALTER TABLE 存储过程调用顺序。也就是说，一旦调用 ALTOBJ() 存储过程，则应在完成或回滚此过程之后，才在相同表上启动另一 ALTER TABLE。只要表相关性不发生冲突，则使用 ALTOBJ() 存储过程的同时改变多个表是受到支持的。

使用执行 ALTER TABLE 操作的存储过程时，可用选项由多个组件段组成。这些段包括:

- ALTER_OBJ('GENERATE','<sql statement>, 0, ?)

  此过程生成所有 SQL 语句，并将其放入元数据表中。

  注: 在生成方式中，SQL 语句参数不能为空；如果提供改变标识，则忽略它。
- ALTER_OBJ('VALIDATE',NULL,123,?)

  此过程验证生成的 SQL，但不包括数据的移动。以给定的用户标识 "123" 运行脚本以测试有效性。验证的结果置于元表中（元表也包含要改变的表中的其它信息）。
- ALTER_OBJ('APPLY_CONTINUE_ON_ERROR',NULL,123,?)

  此过程以给定的标识运行所有 SQL 语句，并将结果写入元表中。SQL 语句将包括如何构建新表、构建任何从属对象和填充新表。

  可以使用 UNDO 方式取回旧定义（请参阅下面的内容）。

  为 SQLCA 中的存储过程设置警告 SQLCODE；完成存储过程中的事务。
- ALTER_OBJ('APPLY_STOP_ON_ERROR',NULL,123,?)

  此过程以给定的标识依次运行每个 SQL 语句，遇到任何错误时则停止。

  对 SQLCA 中的存储过程设置错误 SQLCODE；自动回滚存储过程中的事务。
- ALTER_OBJ('UNDO',NULL,123,?)

  以给定的用户标识运行包含改变表操作所做的全部更改的脚本。所有这些更改均已撤销。

  注: 使用 ALTOBJ_UNDO 时，标识参数不能为空。
- ALTER_OBJ('FINISH',NULL,123,?)

  在给定的用户标识下，此过程删除原始表并清除元表中发现的所有条目。

  注: 此方式只能分别从其它所有方式调用。

相关参考:

- 『Supported functions and SQL administrative routines』（*SQL Reference, Volume 1*）
- 『ALTOBJ procedure』（《*SQL 管理例程*》）

将列添加至现有表

过程:

列定义包括列名、数据类型和任何需要的约束。

当将列添加至表时，必须在逻辑上将列放置在最右边的现有列定义的右边。当将新列添加至现有的表时，只修改系统目录中的表描述，所以表的存取时间不会立即受到影响。在使用 UPDATE 语句修改现有记录之前，不会实际改变它们。当从表中检索现有的行时，根据新列的定义，会为新列提供空值或缺省值。在创建表之后添加的列不能定义为 NOT NULL: 必须将它们定义为 NOT NULL WITH DEFAULT 或可空的。

要使用“控制中心”对现有表添加列:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 右键单击要向其添加列的表，并从弹出菜单中选择**改变**。
3. 检查**列**页，填写列的信息，并单击**确定**。

要使用命令行对现有表添加列，输入:

```
ALTER TABLE <table_name>
   ADD <column_name> <data_type> <null_attribute>
```

可以使用 SQL 语句来添加列。以下语句使用 ALTER TABLE 语句将三列添加至 EMPLOYEE 表:

```
ALTER TABLE EMPLOYEE
  ADD MIDINIT CHAR(1)   NOT NULL WITH DEFAULT
  ADD HIREDATE DATE
  ADD WORKDEPT CHAR(3)
```

相关任务:

- 第 152 页的『修改列定义』

相关参考:

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

修改列定义

过程:

可以通过增加现有 VARCHAR 或 VARGRAPHIC 列的长度来修改一列的特征。字符数可增加到与所用的页大小相关的一个值。

可以修改与列关联的缺省值。一旦定义了新的缺省值，就将对任何后续 SQL 操作中指示使用此缺省值的列使用新值。新值必须遵守赋值规则，且受到限制与 CREATE TABLE 语句下记录的限制相同。

注: 生成列无法通过该语句来改变其缺省值。

要使用“控制中心”来修改现有表列的长度:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 在右窗格中的表列表中，右键单击要修改其一个列的表，并从弹出菜单中选择**改变**。
3. 检查**列**页，选择该列，并单击**更改**。
4. 在**长度**中输入该列的新字节计数，并单击**确定**。

要使用命令行来修改现有表列的长度和类型，请输入:

```
ALTER TABLE <table_name>
   ALTER COLUMN <column_name>
   <modification_type>
```

例如，要将一列增加到 4000 个字符，使用类似于以下的语句:

```
ALTER TABLE t1
   ALTER COLUMN colnam1
   SET DATA TYPE VARCHAR(4000)
```

在另一个示例中，允许一列具有新的 VARGRAPHIC 值，使用类似以下内容的 SQL 语句:

```
ALTER TABLE t1
   ALTER COLUMN colnam2
   SET DATA TYPE VARGRAPHIC(2000)
```

不能改变类型表的列。但是，可将一个作用域添加到尚未定义作用域的现有的引用类型列中。例如:

```
ALTER TABLE t1
   ALTER COLUMN colnamt1
   ADD SCOPE typtab1
```

要使用命令行来修改现有表列的缺省值，请输入:

```
ALTER TABLE <table_name>
   ALTER COLUMN <column_name>
   SET DEFAULT 'new_default_value'
```

例如，要更改列的缺省值，请使用以下类似的语句:

```
ALTER TABLE t1
   ALTER COLUMN colnam1
   SET DEFAULT '123'
```

相关任务:

- 第 155 页的『修改标识列定义』

相关参考:

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

从表或视图除去行

过程:

可以通过删除行来更改表或视图的内容。从视图中删除行也会从基于该视图的表中删除行。使用 DELETE 语句来:

- 删除已由搜索条件任选确定的一行或多行。这称为*搜索 DELETE*。

- 只删除已由游标的当前位置确定的一行。这称为*定位 DELETE*。

可以将 DELETE 语句嵌入应用程序中或作为动态 SQL 语句发出。

若正在修改的表通过引用约束涉及其它表，则对行执行删除有一些注意事项。若标识的表或标识的视图的基本表是父代，则为删除选择的行不能有任何从属与删除规则 RESTRICT 有关。此外，DELETE 不得级联至具有与删除规则 RESTRICT 有关的从属派生行。

若 RESTRICT 删除规则未阻止删除操作，则删除所选行。

例如，要从表（DEPARTMENT）中删除部门（DEPTNO）“D11”，使用:

```
DELETE FROM department WHERE deptno='D11'
```

若在运行多行 DELETE 期间出错，则不会对表进行更改。若发生错误而阻止删除与搜索条件匹配的所有行且阻止现有的引用约束所需要的所有操作，则不会对表进行更改。

除非适当的锁定已经存在，否则在运行成功的 DELETE 语句期间需要互斥锁定。在 COMMIT 或 ROLLBACK 语句之后释放锁定。锁定可以防止其它应用程序对表执行操作。

相关概念:

- 『锁定与并行性控制』（*《管理指南: 性能》*）
- 『锁定和性能』（*《管理指南: 性能》*）
- 『影响锁定的因素』（*《管理指南: 性能》*）
- 『锁定的准则』（*《管理指南: 性能》*）

相关参考:

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

修改列的生成或标识属性

可以使用 ALTER TABLE 语句中的 ALTER COLUMN 子句，在表中添加和删除列的生成或标识属性。

可执行下列其中一项操作:

- 使用现有非生成列时，可以添加生成的表达式属性。修改的列则成为生成的列。
- 使用现有生成列时，可以删除生成的表达式属性。修改的列则成为正常的非生成列。
- 使用现有无标识列时，可以添加标识属性。修改的列则成为标识列。
- 使用现有标识列时，可以删除标识属性。修改的列则成为正常的非生成的无标识列。
- 使用现有生成列时，可以将生成列从 GENERATED ALWAYS 改变为 GENERATED BY DEFAULT。反之亦然，即可将生成列从 GENERATED BY DEFAULT 改变为 GENERATED ALWAYS。只有使用生成列时，此操作才有可能。
- 可以从用户定义的缺省列中删除缺省属性。执行此操作时，新缺省值为空。

- 可以删除缺省值、标识或生成属性，然后在相同的 ALTER COLUMN 语句中设置新的缺省值、标识或生成属性。
- 对于 CREATE TABLE 和 ALTER TABLE 语句，“ALWAYS” 是 GENERATED 子句中的可选字。这意味着在 ALTER TABLE 语句中使用时，GENERATED ALWAYS 与 GENERATED 等价。

相关任务:

- 第 88 页的『对新表定义生成列』
- 第 90 页的『对新表定义标识列』

修改标识列定义

过程:

若正在重新创建表然后执行导入或装入操作，并且在表中具有 IDENTITY 列，则在重新创建表的内容之后将复位该表以从 1 开始生成 IDENTITY 值。当将新行插入此重新创建的表时，您不想 IDENTITY 列再次从 1 开始。您不希望在 IDENTITY 列中有重复值。要防止此情况发生，您应该:

1. 重新创建表。
2. 使用 MODIFIED BY IDENTITYOVERRIDE 子句将数据装入表中。将数据装入表中但不会对行生成标识值。
3. 运行查询以获取 IDENTITY 列的最后一个计数器值:

   ```
   SELECT MAX(<IDENTITY column>)
   ```

   此查询将返回表的 IDENTITY 列值的等价值。
4. 使用 ALTER TABLE 语句的 RESTART 子句:

   ```
   ALTER TABLE <table name> ALTER COLUMN <IDENTITY column>
      RESTART WITH <last counter value>
   ```
5. 将新行插入表中。将基于在 RESTART WITH 子句中指定的值生成 IDENTITY 列值。

相关参考:

- 『MAX aggregate function』（*SQL Reference, Volume 1*）
- 『ALTER TABLE statement』（*SQL Reference, Volume 2*）
- 『LOAD Command』（*Command Reference*）

改变约束

只能通过删除旧约束，然后添加新约束取代它，从而改变约束。有关更多信息，请参阅:

- 『添加约束』
- 第 158 页的『删除唯一约束』

添加约束

使用 ALTER TABLE 语句添加约束。有关此语句（包括它的语法）的更多信息，请参阅 *SQL Reference* 手册。

添加唯一约束

过程:

可以将唯一约束添加至现有表。约束名不能与在 ALTER TABLE 语句内指定的任何其它约束相同，且必须在该表内是唯一的（这包括定义的任何引用完整性约束的名称）。在成功执行该语句之前，会对照新条件检查现有数据。

以下 SQL 语句将一个唯一约束添加至 EMPLOYEE 表，它表示唯一标识该表中的职员的一个新方法:

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT NEWID UNIQUE(EMPNO,HIREDATE)
```

相关任务:

- 第 83 页的『定义唯一约束』
- 第 158 页的『删除唯一约束』

相关参考:

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

添加主键

过程:

要将约束添加至大表，更有效的方法是，使该表处于检查暂挂状态，再添加约束，然后检查该表中违规行的合并列表。使用 SET INTEGRITY 语句来显式设置检查暂挂状态: 若该表是父表，则为所有从属表和派生表隐式设置检查暂挂。

要使用“控制中心”来添加主键:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 右键单击要修改的表，并从弹出菜单中选择**改变**。
3. 在**主键**页上，选择一个或多个列作为主键，并单击箭头来移动它们。
4. 可选: 输入主键的约束名。
5. 单击**确定**。

要使用命令行添加主键，输入:

```
ALTER TABLE <name>
   ADD CONSTRAINT <column_name>
   PRIMARY KEY <column_name>
```

相关任务:

- 第 157 页的『添加外键』

相关参考:

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

添加外键

过程:

当将一个外键添加至表中时，包含下列语句的程序包和高速缓存动态 SQL 可能被标记为无效:

- 插入或更新包含外键的表的语句
- 更新或删除父表的语句。

要使用“控制中心”来添加外键:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 右键单击要修改的表，并从弹出菜单中选择**改变**。
3. 在**外键**页上，单击**添加**。
4. 在**添加外键**窗口上，指定父表信息。
5. 选择一个或多个要作为外键的列，并单击箭头来移动它们。
6. 指定当删除或更新父表的行时要对从属表执行的操作。还可以为该外键添加约束名。
7. 单击**确定**。

要使用命令行添加外键，输入:

```
ALTER TABLE <name>
   ADD CONSTRAINT <column_name>
    FOREIGN KEY <column_name>
    ON DELETE <action_type>
    ON UPDATE <action_type>
```

以下示例显示 ALTER TABLE 语句如何将主键和外键添加至一个表:

```
ALTER TABLE PROJECT
     ADD CONSTRAINT PROJECT_KEY
         PRIMARY KEY (PROJNO)
   ALTER TABLE EMP_ACT
     ADD CONSTRAINT ACTIVITY_KEY
         PRIMARY KEY (EMPNO, PROJNO, ACTNO)
     ADD CONSTRAINT ACT_EMP_REF
         FOREIGN KEY (EMPNO)
         REFERENCES EMPLOYEE
         ON DELETE  RESTRICT
     ADD CONSTRAINT ACT_PROJ_REF
         FOREIGN KEY (PROJNO)
         REFERENCES PROJECT
         ON DELETE CASCADE
```

相关概念:

- 第 179 页的『更改对象时的语句相关性』

相关任务:

- 第 156 页的『添加主键』

相关参考:

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

添加表检查约束

过程:

可以使用 ALTER TABLE 语句将检查约束添加至现有的表。约束名不能与在 ALTER TABLE 语句内指定的任何其它约束相同，且必须在该表内是唯一的（这包括定义的任何引用完整性约束的名称）。在成功执行该语句之前，会对照新条件检查现有数据。

要将约束添加至大表，更有效的方法是，使该表处于检查暂挂状态，添加约束，然后检查该表中违规行的合并列表。使用 SET INTEGRITY 语句来显式设置检查暂挂状态: 若该表是父表，则为所有从属表和派生表隐式设置检查暂挂。

当添加表检查约束时，插入或更新该表的程序包和高速缓存的动态 SQL 可能被标记为无效。

要使用“控制中心”来添加表检查约束:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 右键单击要修改的表，并从弹出菜单中选择**改变**。
3. 在**检查约束**页上，单击**添加**。
4. 在**添加检查约束**窗口上，填写信息，并单击**确定**。
5. 在**检查约束**页上，单击**确定**。

要使用命令行来添加表检查约束，输入:

```
ALTER TABLE <name>
   ADD CONSTRAINT <name> (<constraint>)
```

以下 SQL 语句将一个约束添加至 EMPLOYEE 表，即每个职员的工资加佣金必须超过 $25,000:

```
ALTER TABLE EMPLOYEE
   ADD CONSTRAINT REVENUE CHECK (SALARY + COMM > 25000)
```

相关概念:

- 第 179 页的『更改对象时的语句相关性』

相关参考:

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

删除唯一约束

使用 ALTER TABLE 语句删除约束。有关此语句（包括它的语法）的更多信息，请参阅 *SQL Reference* 手册。

删除唯一约束

过程:

可以使用 ALTER TABLE 语句来显式删除唯一约束。一个表上的所有唯一约束的名称可以在 SYSCAT.INDEXES 系统目录视图中找到。

以下 SQL 语句从 EMPLOYEE 表中删除唯一约束 NEWID:

```
ALTER TABLE EMPLOYEE
     DROP UNIQUE NEWID
```

删除此唯一约束将导致使用该约束的任何程序包或高速缓存的动态 SQL 无效。

相关参考:

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

删除主键

过程:

要使用“控制中心”来删除主键:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 右键单击要修改的表，并从弹出菜单中选择**改变**。
3. 在**主键**页上，在右边选择要删除的主键，并单击箭头来将其移至左边的**可用列**框。
4. 单击**确定**。

要使用命令行来删除主键，输入:

```
ALTER TABLE <name>
  DROP PRIMARY KEY
```

当删除外键约束时，包含下列语句的程序包或高速缓存的动态 SQL 语句可能被标记为无效:

- 插入或更新包含外键的表的语句
- 更新或删除父表的语句

相关概念:

- 第 179 页的『更改对象时的语句相关性』

相关任务:

- 第 159 页的『删除外键』

相关参考:

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

删除外键

过程:

要使用“控制中心”来删除外键:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 右键单击要修改的表，并从弹出菜单中选择**改变**。
3. 在**外键**页上，单击**添加**。
4. 在右边选择要删除的外键，并单击箭头来将其移至左边的**可用列**框。
5. 在**外键**页上，单击**确定**。

要使用命令行来删除外键，输入:

```
ALTER TABLE <name>
   DROP FOREIGN KEY <foreign_key_name>
```

下列示例在 ALTER TABLE 语句中使用 DROP PRIMARY KEY 和 DROP FOREIGN KEY 子句来删除表上的主键和外键:

```
ALTER TABLE EMP_ACT
     DROP PRIMARY KEY     DROP FOREIGN KEY ACT_EMP_REF
     DROP FOREIGN KEY ACT_PROJ_REF
   ALTER TABLE PROJECT
     DROP PRIMARY KEY
```

相关概念:

- 第 179 页的『更改对象时的语句相关性』

相关任务:

- 第 159 页的『删除主键』

相关参考:

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

删除表检查约束

过程:

可以使用 ALTER TABLE 语句显式删除或更改表检查约束，或作为 DROP TABLE 语句的结果隐式删除它。

当删除一个表检查约束时，与该表有 INSERT 或 UPDATE 关系的所有程序包和高速缓存的动态 SQL 语句无效。一个表上的所有检查约束的名称可以在 SYSCAT.INDEXES 目录视图中找到。在尝试删除带有系统生成的名称的表检查约束之前，在 SYSCAT.CHECKS 目录视图中查找该名称。

要使用“控制中心”来删除表检查约束:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 右键单击要修改的表，并从弹出菜单中选择**改变**。
3. 在**检查约束**页上，选择要删除的检查约束，单击**除去**，并单击**确定**。

要使用命令行来删除表检查约束:

```
ALTER TABLE <table_name>
   DROP CHECK <check_constraint_name>
```

以下 SQL 语句从 EMPLOYEE 表中删除表检查约束 REVENUE:

```
ALTER TABLE EMPLOYEE
     DROP CHECK REVENUE
```

相关概念:

- 第 179 页的『更改对象时的语句相关性』

相关任务:

- 第 158 页的『添加表检查约束』

相关参考:

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

对现有表定义生成列

生成列在基本表上定义，在这些列中，存储的值是使用表达式计算得出的，而不是通过插入或更新操作指定。可以在创建表时创建生成列，也可以作为对现有表的修改来创建它。

先决条件:

只能在已对其定义了相等比较的数据类型上定义生成列。生成列的排除数据类型包括：结构化类型、LOB、CLOB、DBCLOB、LONG VARCHAR、LONG VARGRAPHIC 以及使用相同的排除数据类型定义的用户定义的类型。

不能在约束、唯一索引、引用约束、主键和全局临时表中使用生成列。使用 LIKE 和具体化视图创建的表不继承生成列属性。

限制:

不指定关键字 DEFAULT 时，不能插入或更新生成列。进行插入时，使用 DEFAULT 使得无需枚举列列表中的列。而是，可以在值列表中将生成列设置为 DEFAULT。 进行更新时，DEFAULT 允许重新计算特定生成列，SET INTEGRITY 已使这些生成列联机，但未进行检查。

处理触发器的顺序要求 BEFORE 触发器不能在它们的头部（在更新之前）或在它们的主体中引用生成列。在处理顺序中，生成列是在 BEFORE 触发器之后处理的。

db2look 实用程序将看不到生成列所生成的检查约束。

当使用复制时，目标表一定不能在其映射中使用生成列。使用复制时，有两个选项:

- 目标表必须将生成列定义为正常列，即非生成列
- 目标表必须在映射中省略掉生成列

当使用生成列时，有几个限制:

- 生成列一定不能相互依赖。
- 用来创建生成列的表达式一定不能包含子查询。这包括带有读取 SQL 数据（READS SQL DATA）的函数的表达式。
- 生成列上不允许检查约束。

过程:

执行下列步骤来定义生成列:

1. 将表置于检查暂挂状态。

```
SET INTEGRITY FOR t1 OFF
```

2. 改变表，以添加一个或多个生成列。

```
ALTER TABLE t1 ADD COLUMN c3 DOUBLE GENERATED ALWAYS AS (c1 + c2),
               ADD COLUMN c4 GENERATED ALWAYS AS
                  (CASE WHEN c1 > c3 THEN 1 ELSE NULL END))
```

3. 将正确的值指定给生成的列。这可以使用下列方法完成:
   - 使用以下命令重新计算并对生成列重新赋值:

     ```
     SET INTEGRITY FOR t1 IMMEDIATE CHECKED FORCE GENERATED
     ```

     如果此 SET INTEGRITY 语句因日志空间不足而失败，则增大可用的活动日志空间并重新发出 SET INTEGRITY 语句。

     注: 此时，可使用异常表。
   - 如果不能增大可用的活动日志空间，则使用 searched update 语句来对生成列赋予缺省值。
     a. 获取对表的互斥锁定。这将防止除未落实的读取事务之外的所有事务存取该表。注意，将在第一次间歇落实时释放表锁定，而其它事务将能够看到带有未赋予缺省值的生成列的行。

        ```
        LOCK TABLE t1
        ```

     b. 绕过生成列的检查

        ```
        SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
        ```

     c. 检查表中的其它完整性违规（如果适用的话）并使其脱离检查暂挂状态。

        ```
        SET INTEGRITY FOR t1 IMMEDIATE CHECKED
        ```

     d. 使用间歇落实和谓词更新生成列，以避免填充日志。

        ```
        UPDATE t1 SET (c3, c4) = (DEFAULT, DEFAULT) WHERE <predicate>
        ```

     e. 通过使用落实语句完成事务来解锁该表。

        ```
        COMMIT
        ```

   - 如果不能增大可用的活动日志空间，则还可以使用基于游标的方法:
     a. 声明表的 FOR UPDATE 游标。如果间歇落实之后要保留锁定，则应使用 WITH HOLD 选项。

        ```
        DECLARE C1 CURSOR WITH HOLD FOR S1
        ```

        其中 S1 定义为:

        ```
        SELECT '0' FROM t1 FOR UPDATE OF C3, C4
        ```

     b. 打开游标。

        ```
        OPEN C1
        ```

     c. 绕过生成列的检查

        ```
        SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
        ```

     d. 检查表中的其它完整性违规（如果适用的话）并使其脱离检查暂挂状态。

        ```
        SET INTEGRITY FOR t1 IMMEDIATE CHECKED
        ```

e. 让循环访存表中的所有行，对于访存的每一行，执行下列操作以便将生成列指定至其缺省表。一定要确保第一次访存在表脱离检查暂挂后立即执行，以确保在游标的持续时间内表处于锁定状态。

```
UPDATE t1 SET (C3, C4) = (DEFAULT, DEFAULT) WHERE CURRENT OF C1
```

进行间歇落实以避免日志填满。

f. 关闭游标并落实以对表解锁。

```
CLOSE C1
COMMIT
```

- 您知道该表是使用起始未记录的选项创建的。这样，该表的记录是关闭的，当使用生成列值时，存放普通的隐含项和风险。

a. 激活起始未记录的选项。

```
ALTER TABLE t1 ACTIVATE NOT LOGGED INITIALLY
```

b. 生成值。

```
SET INTEGRITY FOR t1 IMMEDIATE CHECKED FORCE GENERATION
```

c. 通过落实事务再次关闭起始未记录的选项。

```
COMMIT
```

也可以简单地通过应用表达式（就象该表达式是相等检查约束一样）检查生成列的值:

```
SET INTEGRITY FOR t1 IMMEDIATE CHECKED
```

若已在生成列中放置了值，例如使用 LOAD，且您知道这些值与生成表达式相匹配，则可以使该表脱离检查暂挂状态，而不必检查或指定值:

```
SET INTEGRITY FOR t1 GENERATED COLUMN IMMEDIATE UNCHECKED
```

相关任务:

- 第 88 页的『对新表定义生成列』

相关参考:

- 『ALTER TABLE statement』（*SQL Reference, Volume 2*）
- 『COMMIT statement』（*SQL Reference, Volume 2*）
- 『LOCK TABLE statement』（*SQL Reference, Volume 2*）
- 『SET INTEGRITY statement』（*SQL Reference, Volume 2*）
- 『UPDATE statement』（*SQL Reference, Volume 2*）
- 『db2look - DB2 Statistics and DDL Extraction Tool Command』（*Command Reference*）

将表声明为易失的

过程:

*易失的*表是在运行时它的内容可能从空变成非常巨大的一种表。此类型表的易失性或极端的可变性使 RUNSTATS 收集的统计信息变得不准确。统计信息是在一个时间点收集的，它也只反映该时间点的情况。若生成一个使用易失表的存取方案，可导致不正确或性能不好的计划。例如，若易失表为空时收集统计信息，优化器可能会使用表扫描而不是索引扫描来存取易失表。

为了防止这种情况，应考虑使用 ALTER TABLE 语句将该表声明为易失的。通过将该表声明为易失的，优化器将考虑使用索引扫描而不是表扫描。使用已声明易失表的存取方案将不依赖于该表的现有统计信息。

要使用“控制中心”来将表声明为易失的:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 右键单击要修改的表，并从弹出菜单中选择**改变**。
3. 在**表**页中，选择**基数在运行时显著变化**复选框，并单击**确定**。

要使用命令行来将表说明为“易失的”，输入:

```
ALTER TABLE <table_name>
    VOLATILE CARDINALITY
```

相关参考:

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

更改分区键

过程:

只能在单一分区数据库分区组中的表上更改分区键。首先删除现有分区键，然后创建另一个分区键。

以下 SQL 语句从 MIXREC 表中删除分区键 MIX_INT:

```
ALTER TABLE MIXREC
      DROP PARTITIONING KEY
```

不能更改多分区数据库分区组中的表的分区键。若尝试删除它，则会返回错误。

要更改多分区数据库分区组的分区键，执行以下两种操作均可:

- 将所有数据导出至单一分区数据库分区组，然后遵循以上指示信息。
- 导出所有数据，删除表，重新创建表并重新定义分区键，然后导入所有数据。

这两种方法都不适用于大数据库；因此，在实现大数据库的设计之前，要定义适当的分区键。

相关概念:

- 『分区键』（*《管理指南: 计划》*）

相关参考:

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

更改表属性

过程:

您可能需要更改表属性，如数据捕获选项、每页上可用空间的百分比（PCTFREE）、锁定大小或追加方式。

在一个表的每一页上要留下的可用空间量是通过 PCTFREE 指定的，它并且是有效使用群集索引的一条重要注意事项。要指定的数量取决于现有数据和预计的将来数据的性质。LOAD 和 REORG 会考虑 PCTFREE，但插入、更新和导入活动将忽略它。

将 PCTFREE 设置为更大的值将在更长的时期内维持群集，但是，这也将需要更多的磁盘空间。

可指定使用 LOCKSYZE 参数存取表时使用的锁定的大小（粒度）。在缺省情况下，在创建表时，定义行级别锁定。使用表级别锁定可提高查询的性能，方法是限制需要获取和释放的锁定的数目。

通过指定 APPEND ON，可提高表的整体性能。它允许更快的插入，而无需维护关于可用空间的信息。

不能将带群集索引的表改变为打开追加方式。类似地，不能在具有追加方式的表上创建群集索引。

相关概念:

- 『锁定与并行性控制』（*《管理指南: 性能》*）
- 『锁定属性』（*《管理指南: 性能》*）
- 『锁定和性能』（*《管理指南: 性能》*）
- 『影响锁定的因素』（*《管理指南: 性能》*）
- 『锁定的准则』（*《管理指南: 性能》*）

相关参考:

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

改变标识列

过程:

使用 ALTER TABLE 语句修改现有标识列的属性。

有多种方式来修改标识列，以便它具有序列的某些特征。

有某些任务专用于 ALTER TABLE 语句和标识列:

- RESTART 将与标识列关联的序列复位为隐式或显式指定的值，该值是在最初创建标识列时作为起始值指定的。
- RESTART WITH <numeric-constant> 将与标识列相关联的序列复位为准确的数字常数值。数字常数可以是任何正的或负的值，而且任何小数点右边不带有非零数字，可以将该值赋给标识列。

相关参考:

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

改变序列

过程:

使用 ALTER SEQUENCE 语句修改现有序列的属性。

可以修改的序列属性包括:

- 更改将来值的递增
- 建立新的最小值或最大值
- 更改高速缓存序列号的数目
- 更改序列是否将循环
- 更改是否必须按请求顺序生成序列号
- 重新启动序列

有两种任务不是序列创建的一部分。它们是:

- RESTART。将序列复位为隐式或显式指定的值，该值是在创建序列时作为起始值指定的。
- RESTART WITH <numeric-constant>。将序列复位为精确的数字常数值。数字常数可以是任何正的或负的值，而且任何小数点右边不带有非零数字。

在重新启动序列或更改为 CYCLE 之后，可能会生成重复序列号。ALTER SEQUENCE 语句仅影响将来的序列号。

不能更改序列的数据类型。而是必须删除当前序列，然后创建新序列，指定新的数据类型。

当改变序列时，会丢失 DB2 未使用的所有高速缓存序列值。

相关任务:

- 第 166 页的『删除序列』

相关参考:

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

删除序列

过程:

要删除序列，使用 DROP 语句。

可以通过使用下列命令来删除特定序列:

```
DROP SEQUENCE sequence_name
```

其中 sequence_name 是要删除的序列名，它包括用来正确标识现有序列的隐式或显式模式名。

不能使用 DROP SEQUENCE 语句删除系统为 IDENTITY 列创建的序列。

一旦删除序列，则也会删除对该序列的所有特权。

相关任务:

- 第 165 页的『改变序列』

相关参考:

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

改变具体查询表属性

过程:

在遵守某些限制的前提下，可将具体查询表更改为常规表或将常规表更改为具体查询表。不能更改其它表类型；只能更改常规表和具体查询表。例如，不能将复制具体查询表更改为常规表，反之亦然。

一旦将常规表改变为具体查询表，该表便处于检查暂挂状态。当使用此方法进行改变时，具体查询表定义中的全选择必须与原始表定义相匹配，即:

- 列数必须相同。
- 列名和位置必须匹配。
- 数据类型必须完全相同。

如果具体查询表是对原始表定义的，则不能将原始表本身改变为具体查询表。如果原始表有触发器、检查约束、引用约束或定义的唯一索引，则不能将其改变为具体查询表。如果改变表属性来定义具体查询表，则不允许在同一 ALTER TABLE 语句中以任何其它方法改变该表。

在将常规表改变为具体查询表时，具体查询表定义的全选择不能直接引用原始表或通过视图、别名或具体查询表间接引用原始表。

要将具体查询表更改为常规表，使用以下命令:

```
ALTER TABLE sumtable
   SET SUMMARY AS DEFINITION ONLY
```

要将常规表更改为具体查询表，使用以下命令:

```
ALTER TABLE regtable
   SET SUMMARY AS <fullselect>
```

将常规表改变为具体查询表时，有关全选择的限制与使用 CREATE SUMMARY TABLE 语句创建具体查询表时的限制非常相似。

相关任务:

- 第 112 页的『创建具体查询表』
- 第 167 页的『刷新具体查询表中的数据』
- 第 176 页的『删除具体查询表或分级表』

相关参考:

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

刷新具体查询表中的数据

过程:

可通过使用 REFRESH TABLE 语句来刷新一个或多个具体查询表中的数据。此语句可嵌入应用程序中，或可动态执行。要使用此语句，必须具有 SYSADM 或 DBADM 权限，或对要刷新的表具有 CONTROL 特权。

以下示例显示如何刷新具体查询表中的数据:

```
REFRESH TABLE SUMTAB1
```

相关任务:

- 第 112 页的『创建具体查询表』
- 第 167 页的『改变具体查询表属性』

相关参考:

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

改变用户定义的结构化类型

过程:

创建结构化类型之后，可能会发现需要添加或删除与该结构化类型相关的属性。这可使用 ALTER TYPE （结构化）语句来完成。

相关概念:

- 『User-defined structured types』（*Application Development Guide: Programming Server Applications*）
- 『Structured type hierarchies』（*Application Development Guide: Programming Server Applications*）

相关任务:

- 『Creating structured types』（*Application Development Guide: Programming Server Applications*）

相关参考:

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

删除和更新类型表的行

可使用搜索式或定位式 DELETE 语句从类型表中删除行。可使用搜索式或定位式 UPDATE 语句更新类型表中的行。

相关概念:

- 『Typed tables』（*Application Development Guide: Programming Server Applications*）

相关参考:

- 『DELETE statement』（*SQL Reference, Volume 2*）
- 『UPDATE statement』（*SQL Reference, Volume 2*）

重命名现有表或索引

可在模式内给予现有表或索引新名称，并维护对原表创建的授权和索引。

先决条件:

要重命名的现有表或索引可以是标识表或索引的别名。

限制:

要重命名的现有表或索引不能是目录表或索引、总结表或索引、类型表、已声明全局临时表以及昵称的名称，也不能是除表、视图或别名以外的对象。

不能在下列任何一个对象中引用现有表或索引:

- 视图
- 触发器
- 引用约束
- 总结表
- 现有引用列的作用域

并且，该表内不能有检查约束，也不能有除标识列之外的任何生成列。依赖于原表的任何程序包或高速缓存的动态 SQL 语句无效。最后，引用该原表的任何别名不会被修改。

应考虑检查相应系统目录表以确保重命名的表或索引不受以上任何一个限制的影响。

过程:

要使用“控制中心”来重命名现有表或索引:

1. 展开对象树，直到看到**表**或**视图**文件夹为止。
2. 右键单击想要重命名的表或视图，并从弹出菜单中选择**重命名**。
3. 输入新表或视图名，并单击**确定**。

要使用命令行来重命名现有的表，输入:

```
RENAME TABLE <schema_name>.<table_name> TO <new_name>
```

以下的 SQL 语句在 COMPANY 模式内将 EMPLOYEE 表重命名为 EMPL:

```
RENAME TABLE COMPANY.EMPLOYEE TO EMPL
```

要使用命令行重命名现有索引，输入:

```
RENAME INDEX <schema_name>.<index_name> TO <new_name>
```

以下 SQL 语句将 COMPANY 模式内的 EMPIND 索引重命名为 MSTRIND:

```
RENAME INDEX COMPANY.EMPIND TO MSTRIND
```

如果程序包引用刚刚重命名的表或索引，则程序包会无效且必须重新绑定。隐式重新绑定程序包，而不考虑是否存在同名索引。除非有更好的选择，否则程序包将使用它以前具有的索引，但使用新的名称。

相关参考:

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

使用 MERGE 语句更新表和视图内容

DB2 通用数据库能够使用来自另一个源的数据（通常是表引用的结果）更新表或视图。这种更新是使用 MERGE 语句来执行的。

可以根据 MERGE 语句中的指定指示信息删除或更新目标表中与源相匹配的行。可以插入目标表中不存在的行。

在视图中更新、删除或插入行会导致在视图所基于的表中更新、删除或插入相应行。

限制:

与 MERGE 语句相关联的授权标识必须具有适当的特权才能执行以下三种可能操作中的任何一种：对视图的目标表或基础表的更新、删除或插入操作。授权标识还应该对子查询中视图的表或基础表具有适当特权。

如果 MERGE 语句中发生了错误，则会回滚与 MERGE 相关联的整个操作集。

不能更新视图的目标表或基础表中在运行 MERGE 语句之前不存在的行。即，不允许更新作为 MERGE 语句的一部分插入的行。

如果将视图指定为 MERGE 语句的目标，则不应为该视图定义任何 INSTEAD OF 触发器；或者应为更新、删除和插入操作中的每一个定义 INSTEAD OF 触发器。

过程:

要对目标表执行更新、删除、插入或这三种操作的任意组合，在命令提示符处输入下列内容:

```
MERGE INTO <table or view name>
   USING <table reference> ON <search condition>
   WHEN <match condition> THEN <modification operation or signal statement>
```

每个 MERGE 语句可多次指定修改操作和信号语句。在单个 MERGE 语句中只能对目标表或视图中的每行执行一次操作。这意味着目标表或视图中的某行仅可标识为与表引用的结果表中的一行“匹配”（MATCHED）。

考虑存在 shipment 和 inventory 两个表的情况。通过使用 shipment 表，将行合并到 inventory 表中。对于匹配的行，按照 shipment 表中的数量来增加 inventory 表中的数量。否则，将新部件号插入到 inventory 表中。

```
MERGE INTO inventory AS in
   USING (SELECT partno, description, count FROM shipment
   WHERE shipment. partno IS NOT NULL) AS sh
   ON (in.partno = sh.partno)
   WHEN MATCHED THEN
      UPDATE SET
         description = sh.description
         quantity = in.quantity + sh.count
   WHEN NOT MATCHED THEN
INSERT
         (partno, description, quantity)
         VALUES (sh.partno, sh.description, sh.count)
```

此示例中没有任何 DELETE 选项。较复杂的匹配条件可考虑添加 DELETE 选项。有几个其它选项（例如，使用信号语句和 ELSE 子句）在此处未作说明，但可在 SQL Reference 中找到它们。

相关参考:

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

删除表

过程:

可以使用 DROP TABLE SQL 语句删除表。

当删除一个表时，也会删除 SYSCAT.TABLES 目录中包含有关该表的信息的那一行，并会影响从属于该表的任何其它对象。例如:

- 会删除所有的列名。
- 会删除基于该表的任何列创建的索引。
- 将基于该表的所有视图标记为不可用。
- 对删除的表和从属视图的所有特权被隐式撤销。
- 会删除在其中该表为父表或从属表的所有引用约束。
- 从属于删除的表的所有程序包和高速缓存的动态 SQL 语句被标记为无效，且该状态会保持至重新创建了从属对象为止。这包括这样一些程序包，它们从属于将被删除的层次结构中子表上的任何超表。
- 其引用的作用域为删除的表的任何引用列变为“无作用域”。
- 因为可以取消定义别名，所以该表上的别名定义不受影响
- 将从属于该删除的表的所有触发器标记为不可用。
- 通过任何 DATALINK 列链接的所有文件都取消链接。取消链接操作是异步执行的，表明这些文件可能不能立即用于其它操作。

要使用“控制中心”来删除表:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 右键单击要删除的表，并从弹出菜单中选择**删除**。
3. 选择**确认**框，并单击**确定**。

要使用命令行来删除表，输入:

```
DROP TABLE <table_name>
```

以下语句删除 DEPARTMENT 表:

```
DROP TABLE DEPARTMENT
```

若一个表有子表，则不能删除它。但是，可用单个 DROP TABLE HIERARCHY 语句删除一个表 e 中的所有表，如以下示例所示:

```
DROP TABLE HIERARCHY person
```

DROP TABLE HIERARCHY 语句必须命名要删除的层次结构的根表。

删除表层次结构与删除特定的表之间有一些差别:

- DROP TABLE HIERARCHY 不会激活将由各个 DROP 表语句激活的删除触发器。例如，删除个别子表将激活其超表上的删除触发器。
- DROP TABLE HIERARCHY 不为删除的表的各个行建立日志条目。而是将该层次结构的删除作为单个事件记录。

相关概念:

- 第 179 页的『更改对象时的语句相关性』

相关任务:

- 第 172 页的『删除用户定义的临时表』
- 第 176 页的『恢复不可用视图』

相关参考:

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

删除用户定义的临时表

用户定义的临时表是使用 DECLARE GLOBAL TEMPORARY TABLE 语句创建的。

先决条件:

删除这样的表时，表名必须由模式名 SESSION 限定，且该表名必须存在于创建该表的应用程序中。

限制:

程序包不能依赖于这种类型的表，因此，删除这样的表时，不使它们无效。

过程:

删除用户定义的临时表时，若其创建时间早于活动工作单元或保存点，则在功能上删除该表，应用程序无法存取该表。但是，该表仍在其表空间中保留了一些空间，这使得在落实工作单元或结束保存点之前，不能删除该用户临时表空间。

相关任务:

- 第 89 页的『创建用户定义的临时表』

相关参考:

- 『DROP statement』（*SQL Reference, Volume 2*）
- 『SET SCHEMA statement』（*SQL Reference, Volume 2*）

删除触发器

过程:

可以使用 DROP 语句删除触发器对象，但是此过程将导致从属程序包被标记为无效，如下所示:

- 若删除不带显式列列表的更新触发器，则对目标表起更新作用的程序包就会无效。

- 若删除带一个列列表的更新触发器，则仅当该程序包也可更新 CREATE TRIGGER 语句的列名列表中至少一列时，用于更新目标表的该程序包才无效。
- 若删除插入触发器，则用于插入目标表的程序包无效。
- 若删除删除触发器，则用于删除目标表的程序包无效。

程序包将保持无效，直到显式绑定或重新绑定该应用程序，或运行它且数据库管理器自动重新绑定它为止。

相关任务:

- 第 100 页的『创建触发器』

相关参考:

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

删除用户定义的函数（UDF）、函数映射或方法

可以使用 DROP 语句来删除用户定义的函数（UDF）、函数模板或函数映射。

先决条件:

其它对象可以从属于一个函数或函数模板。必须除去所有这样的从属，包括函数映射，才能删除该函数，但标记为不可用的程序包除外。

限制:

若视图、触发器、表检查约束或另一个 UDF 从属于某个 UDF，则不能删除该 UDF。不能删除由 CREATE DISTINCT TYPE 语句隐式生成的函数。将 SYSIBM 模式或 SYSFUN 模式中的函数删除是不可能的。

过程:

可以使用映射选项 DISABLE 禁用函数映射。

不会隐式重新绑定标记为不可用的程序包。必须使用 BIND 或 REBIND 命令来重新绑定程序包，或必须使用 PREP 命令对它进行预编译。删除 UDF 将导致使用它的任何程序包或高速缓存的动态 SQL 语句无效。

删除一个函数映射会将一个程序包标记为无效。重新绑定将自动进行并且优化器将尝试使用本地函数。对于本地函数是模板的情况，隐式重新绑定将失败。

相关参考:

- 『DROP statement』（*SQL Reference, Volume 2*）
- 『BIND Command』（*Command Reference*）
- 『PRECOMPILE Command』（*Command Reference*）
- 『REBIND Command』（*Command Reference*）

删除用户定义的类型（UDT）或类型映射

可使用 DROP 语句来删除用户定义的类型（UDT）或类型映射。

限制:

若 UDT 用于以下情况，则不能将其删除:

- 在现有的表或视图（单值类型）的列定义中
- 作为现有的类型表或带类型视图（结构化类型）的类型
- 作为另一个结构化类型的超类型

不能删除缺省类型映射；只能创建另一个类型映射来覆盖它。

数据库管理器尝试删除从属于此单值类型的所有函数。若不能删除该 UDF，则也不能删除该 UDT。若视图、触发器、表检查约束或另一个 UDF 从属于某个 UDF，则不能删除该 UDF。删除 UDT 将导致使用它的任何程序包或高速缓存的动态 SQL 语句无效。

注意，只能删除您或其他应用程序开发人员定义的变换；不能删除内置变换及其相关的组定义。

过程:

使用 DROP 语句来删除用户定义的类型。

若为一个 UDT 创建了一个变换，且计划删除该 UDT，应考虑它是否需要删除该变换。这可通过 DROP TRANSFORM 语句来完成。

相关概念:

- 第 106 页的『用户定义的类型（UDT）』

相关任务:

- 第 107 页的『创建用户定义的单值类型』
- 第 108 页的『创建类型映射』

相关参考:

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

改变或删除视图

ALTER VIEW 语句通过改变引用类型列来添加作用域，以修改现有视图定义。DROP 语句删除视图。

先决条件:

当改变视图时，必须将作用域添加到尚未定义作用域的现有的引用类型列中。此外，不能从超视图继承该列。

限制:

对视图的基本内容所作的更改需要使用触发器。对视图的其它更改需要删除视图然后重新创建视图。

过程:

在 ALTER VIEW 语句中列名的数据类型必须是 REF（类型表名或带类型视图名的类型）。还可以通过 INSTEAD OF 触发器修改视图的内容。

虽然程序包和高速缓存的动态语句都被标记为无效，但是不会影响其它数据库对象，如表和索引。

要使用“控制中心”来改变视图的定义:

1. 展开对象树，直到您看到**视图**文件夹为止。
2. 右键单击要修改的视图，并从弹出菜单中选择**改变**。
3. 在**改变视图**窗口中，输入或修改注释，并单击**确定**。

要使用命令行来改变视图，输入:

```
ALTER VIEW <view_name> ALTER <column name>
   ADD SCOPE <typed table or view name>
```

要使用“控制中心”来删除视图:

1. 展开对象树，直到您看到**视图**文件夹为止。
2. 右键单击要删除的视图，并从弹出菜单中选择**删除**。
3. 选择**确认**框，并单击**确定**。

要使用命令行来删除视图，输入:

```
DROP VIEW <view_name>
```

以下示例显示如何删除 EMP_VIEW:

```
DROP VIEW EMP_VIEW
```

从属于要删除的视图的任何其它视图将变得不可用。

对于表层次结构，可以在一条语句中删除整个视图层次结构，方法是命名该层次结构的根视图，如以下示例所示:

```
DROP VIEW HIERARCHY VPerson
```

相关概念:

- 第 179 页的『更改对象时的语句相关性』

相关任务:

- 第 100 页的『创建触发器』
- 第 109 页的『创建视图』
- 第 176 页的『恢复不可用视图』

相关参考:

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

恢复不可用视图

过程:

视图可能变得*不可用*:

- 由于撤销了对基础表的特权。
- 在删除了表、别名或函数的情况下。
- 在超视图变得不可用的情况下。(超视图是另一个带类型视图或子视图所基于的带类型视图。)
- 当删除它们所从属的视图时。

下列步骤可以帮助您恢复不可用视图:

1. 确定最初用于创建该视图的 SQL 语句。可以从 SYSCAT.VIEW 目录视图的 TEXT 列获取此信息。
2. 使用 CREATE VIEW 语句并使用相同的视图名和相同的定义来重新创建该视图。
3. 使用 GRANT 语句重新授予先前在该视图上授予的所有特权。(注意,在不可用视图上授予的所有特权都被撤销。)

若不希望恢复不可用视图,可以使用 DROP VIEW 语句显式删除它,或者可以使用相同的名称和不同的定义来创建一个新视图。

不可用视图只在 SYSCAT.TABLES 和 SYSCAT.VIEWS 目录视图中具有条目;SYSCAT.VIEWDEP、SYSCAT.TABAUTH、SYSCAT.COLUMNS 和 SYSCAT.COLAUTH 目录视图中的所有条目已被除去。

相关任务:

- 第 174 页的『改变或删除视图』

相关参考:

- 『CREATE VIEW statement』(*SQL Reference, Volume 2*)
- 『DROP statement』(*SQL Reference, Volume 2*)
- 『GRANT (Table, View, or Nickname Privileges) statement』(*SQL Reference, Volume 2*)
- 『SYSCAT.VIEWS catalog view』(*SQL Reference, Volume 1*)

删除具体查询表或分级表

过程:

不能改变具体查询表或分级表,但可删除它。

所有引用该表的索引、主键、外键和检查约束均被删除。所有引用该表的视图和触发器均变得不可用。从属于任何删除的对象的程序包或被标记为不可用的程序包均将无效。

要使用“控制中心”删除具体查询表:

1. 展开对象树，直到您看到**表**文件夹为止。
2. 右键单击要删除的具体查询表或分级表，并从弹出菜单中选择**删除**。
3. 选择**确认**框，并单击**确定**。

要使用命令行删除具体查询表或分级表，请输入:

```
DROP TABLE <table_name>
```

以下 SQL 语句删除具体查询表 XT:

```
DROP TABLE XT
```

可使用 DROP TABLE 语句显式删除具体查询表，如果删除了其中任何一个基础表，则可能会隐式删除该具体查询表。

可使用 DROP TABLE 语句显式删除分级表，如果删除了其关联的具体查询表，可能会隐式删除该分级表。

相关概念:

- 第 179 页的『更改对象时的语句相关性』

相关任务:

- 第 112 页的『创建具体查询表』
- 第 116 页的『创建分级表』

相关参考:

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

恢复不可用总结表

过程:

撤销基础表的 SELECT 特权将会导致总结表变得*不可用*。

下列步骤可帮助您恢复不可用总结表:

- 确定最初用于创建该总结表的 SQL 语句。可以从 SYSCAT.VIEW 目录视图的 TEXT 列获取此信息。
- 使用 CREATE SUMMARY 语句并使用相同的总结表名和相同的定义，来重新创建该总结表。
- 使用 GRANT 语句重新授予先前在该总结表上授予的所有特权。（注意，在不可用总结表上授予的所有特权都被撤销。）

若不希望恢复不可用总结表，可以使用 DROP TABLE 语句显式地删除它，或者可以使用相同的名称但是不同的定义来创建新的总结表。

不可用总结表只在 SYSCAT.TABLES 和 SYSCAT.VIEWS 目录视图中具有条目；在 SYSCAT.VIEWDEP、SYSCAT.TABAUTH、SYSCAT.COLUMNS 和 SYSCAT.COLAUTH 目录视图中的所有条目已被除去。

相关参考:

- 『CREATE TABLE statement』（*SQL Reference, Volume 2*）
- 『DROP statement』（*SQL Reference, Volume 2*）
- 『GRANT (Table, View, or Nickname Privileges) statement』（*SQL Reference, Volume 2*）
- 『SYSCAT.VIEWS catalog view』（*SQL Reference, Volume 1*）

删除索引、索引扩展或索引规范

限制:

不能更改索引定义、索引扩展或索引规范的任何子句；必须删除该索引或索引扩展，并再次创建它。（删除索引或索引规范不会导致删除任何其它对象，但可能导致一些程序包无效。）

索引扩展的名称必须标识目录中描述的索引扩展。RESTRICT 子句强制执行一个规则，即不能定义依赖于索引扩展定义的索引。若一个底层索引依赖于此索引扩展，则删除失败。

不能显式删除主键或唯一键索引（除非它是索引规范）。必须使用下列其中一种方法删除它:

- 若主索引或唯一约束是为主键或唯一键自动创建的，则删除主键或唯一键将会使该索引也被删除。使用 ALTER TABLE 语句来执行删除。
- 若主索引或唯一约束是用户定义的，则必须使用 ALTER TABLE 语句先将主键或唯一键删除。在删除主键或唯一键之后，该索引就不再被认为是主索引或唯一索引了，这时可以显式删除它。

过程:

要使用“控制中心”来删除索引、索引扩展或索引规范:

1. 展开对象树，直到您看到**索引**文件夹为止。
2. 右键单击要删除的索引，并从弹出菜单中选择**删除**。
3. 选择**确认**框，并单击**确定**。

要使用命令行来删除索引、索引扩展或索引规范，输入:

```
DROP INDEX <index_name>
```

以下 SQL 语句删除称为 PH 的索引:

```
DROP INDEX PH
```

以下 SQL 语句删除称为 IX_MAP 的索引扩展:

```
DROP INDEX EXTENSION ix_map RESTRICT
```

任何从属于删除的索引的程序包和高速缓存的动态 SQL 语句都被标记为无效。应用程序不受添加或删除索引所导致的更改的影响。

相关概念:

- 第 179 页的『更改对象时的语句相关性』

相关参考:

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

更改对象时的语句相关性

语句相关性包括程序包和高速缓存的动态 SQL 语句。*程序包*是一个数据库对象，它包含数据库管理器为了以最有效的方式存取特定应用程序的数据所需的信息。*绑定*是创建程序包的过程，当执行应用程序时数据库管理器需要该程序包来存取该数据库。

程序包和高速缓存的动态 SQL 语句可以从属于很多类型的对象。

可以显式引用这些对象，例如，在一个 SQL SELECT 语句中涉及的一个表或用户定义的函数。也可以隐式引用这些对象，例如，当删除父表中的一行时，为确保不违反引用约束需要检查的从属表。程序包还与已授予程序包创建者的特权有关。

如果程序包或高速缓存的动态 SQL 语句依赖于某个对象而该对象被删除，则该程序包或高速缓存的动态 SQL 语句将被置于“无效”状态。若程序包依赖于用户定义的函数而该函数被删除，则该程序包被置于“不可用”状态。

处于无效状态的高速缓存动态 SQL 语句在下次使用时将被自动重新优化。若该语句需要的一个对象已被删除，则执行该动态 SQL 语句可能失败，并伴有错误消息。

处于无效状态的程序包在下次使用时将被隐式重新绑定。也可以显式重新绑定这种程序包。若由于删除一个触发器而将一个程序包标记为无效，则重新绑定后的程序包不再调用该触发器。

必须显式重新绑定处于不可用状态的程序包，然后才能使用它。

联合数据库对象具有类似的相关性。例如，删除服务器将使引用与该服务器相关的别名的任何程序包或高速缓存动态 SQL 无效。

在某些情况下，重新绑定程序包是不可能的。例如，若一个表已删除但尚未重新创建，则不能重新绑定该程序包。在这种情况中，需要重新创建该对象或更改该应用程序，以使它不使用删除的对象。

在许多其它情况中，例如，若删除了一个约束，则重新绑定该程序包是可能的。

下列系统目录视图可帮助您确定程序包的状态和程序包的相关性:

- SYSCAT.PACKAGEAUTH
- SYSCAT.PACKAGEDEP
- SYSCAT.PACKAGES

相关概念:

- 『Package Creation Using the BIND Command』（*Application Development Guide: Programming Client Applications*）
- 『Application, Bind File, and Package Relationships』（*Application Development Guide: Programming Client Applications*）

- 『Package Rebinding』（*Application Development Guide: Programming Client Applications*）

相关参考:

- 『DROP statement』（*SQL Reference, Volume 2*）
- 『SYSCAT.PACKAGEAUTH catalog view』（*SQL Reference, Volume 1*）
- 『SYSCAT.PACKAGEDEP catalog view』（*SQL Reference, Volume 1*）
- 『SYSCAT.PACKAGES catalog view』（*SQL Reference, Volume 1*）
- 『BIND Command』（*Command Reference*）
- 『REBIND Command』（*Command Reference*）

第 2 部分 数据库安全性

第 7 章 控制数据库存取

数据库管理员和系统管理员的其中一个最重要的责任是保证数据库的安全性。保护数据库涉及到以下几个任务:

- 防止由于设备或系统故障而意外丢失数据或损害数据完整性。
- 防止对有价值的数据的未经授权的存取。您必须确保敏感信息未被不“需要知道”这些信息的人存取。
- 防止未经授权的人员通过恶意删除或篡改数据来进行破坏。
- 监视用户对数据的存取，这在第 229 页的第 8 章，『审计 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）活动』中做了讨论。

讨论下列主题:

- 『安装 DB2 通用数据库时的安全问题』
- 第 188 页的『服务器的认证方法』
- 第 192 页的『远程客户机的认证注意事项』
- 第 193 页的『分区数据库认证注意事项』
- 第 225 页的『防火墙支持简介』
- 第 196 页的『特权、权限级别和数据库权限』
- 第 211 页的『控制对数据库对象的存取』
- 第 220 页的『任务和必需的授权』
- 第 221 页的『将系统目录用于安全性说明』.

制定安全性计划: 首先定义数据库存取控制方案的目标，并指定哪些人在什么条件下可存取哪些内容。您的方案还应当描述如何通过使用数据库函数、其它程序的函数和管理过程来实现这些目标。

安装 DB2 通用数据库时的安全问题

从安装产品之时起，安全问题对于 DB2® 管理员而言是十分重要的。

要完成 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）的安装，需要用户标识、组名和密码。基于 GUI 的 DB2 UDB 安装程序为不同用户标识和组创建缺省值。根据是在 UNIX® 平台上还是在 Windows® 平台上进行安装来创建不同的缺省值:

- 在 UNIX 平台上，DB2 UDB 安装程序为 DAS（dasusr）、实例所有者（db2inst）和受防护的用户（db2fenc）创建不同的缺省用户。

  直到可以创建尚未存在的用户标识后，DB2 UDB 安装程序才能将 1 至 99 的数字追加到缺省用户名的后面。例如，如果用户 db2inst1 和 db2inst2 已存在，则 DB2 UDB 安装程序会创建用户 db2inst3。如果使用大于 10 的数字，则该名称的字符部分在缺省用户标识中将被截断。例如，如果用户标识 db2fenc9 已存在，DB2 UDB 安装程序截断用户标识中的 c，然后追加 10（即 db2fen10）。在将数字值追加到缺省 DAS 用户（例如 dasusr24）时，不发生截断。

- 在 Windows 平台上，DB2 UDB 安装程序为 DAS 用户、实例所有者和受防护的用户创建缺省用户 db2admin。与 UNIX 平台不同，不会将任何数字值追加到用户标识后面。

除管理员以外的用户可能会知道缺省值，并且在数据库和实例中以不适当的方式来使用这些缺省值，要将这种风险降到最低，请您在安装期间将缺省值更改为您选择的新用户标识或现有用户标识。

注: 响应文件安装不对用户标识或组名使用缺省值。这些值必须在响应文件中指定。

认证用户时，密码非常重要。如果在操作系统级别上未设置认证需求，且数据库正在使用该操作系统来认证用户，则将允许用户连接。例如，在 UNIX 操作系统上，将未定义的密码视为 NULL。在此情况下，任何不具备已定义密码的用户将被视为具有 NULL 密码。从操作系统的角度来看，这是一种匹配，用户得到验证，并且能够连接到数据库。如果要使操作系统为您的数据库执行用户认证，请使用操作系统级别的密码。

注: 如果要使数据库环境符合 Common Criteria 要求，则不能使用未定义的密码。

在安装 DB2 通用数据库之后，还可查看和更改（如果需要的话）已经授予用户的缺省特权。缺省情况下，安装过程在每种操作系统上均为以下用户授予系统管理（SYSADM）特权:

Windows 9x	任何 Windows 98 或 Windows ME 用户。
其它 Windows 环境	在 Windows NT®、Windows 2000、Windows XP 或 Windows Server 2003 上，属于管理员组的有效 DB2 UDB 用户名。
UNIX 平台	属于实例所有者的主组的有效 DB2 UDB 用户名。

SYSADM 特权是 DB2 通用数据库内提供的功能最强大的特权集合。因此，您可能不想让所有这些用户在缺省情况下都拥有 SYSADM 特权。DB2 UDB 使管理员能够授予和撤销组以及各个用户标识的特权。

通过更新数据库管理器配置参数 *sysadm_group*，管理员可以控制由哪个用户组拥有 SYSADM 特权。您必须遵循以下准则来完成 DB2 UDB 安装和后续实例及数据库创建的安全性需求。

任何定义为系统管理组的组（通过更新 *sysadm_group*）都必须存在。此组的名称应该能够让人轻松地识别出是为实例所有者创建的组。属于此组的用户标识和组对各自的实例均具有系统管理员权限。

管理员应该考虑创建可轻松识别为与特定实例相关联的实例所有者用户标识。作为其中一个组成员，此用户标识应该具有以上创建的 SYSADM 组的名称。另一项建议是只将此实例所有者用户标识用作实例所有者组的成员，而且不在任何其它组中使用该标识。这应该控制可以修改该实例或实例中的任何对象的用户标识和组的增多。

创建的用户标识必须与密码相关联，以便在被允许输入实例内的数据和数据库之前提供认证。创建密码时，建议遵循您所在组织的密码命名准则。

相关概念:

- 第 270 页的『NLS 环境中的命名规则』
- 第 271 页的『Unicode 环境中的命名规则』

- 第 187 页的『用户的 Windows NT 平台安全性注意事项』
- 第 187 页的『用户的 UNIX 平台安全性注意事项』
- 『认证』（*《管理指南: 计划》*）
- 『授权』（*《管理指南: 计划》*）
- 第 188 页的『实例目录的位置』
- 第 265 页的『通用命名规则』
- 第 268 页的『用户、用户标识和组命名规则』

使用访问令牌获取 Windows 用户的组信息

访问令牌是描述进程或线程的安全性上下文的对象。访问令牌中的信息包括与进程或线程关联的用户帐户的标识和特权。

登录时，系统通过比较密码与安全数据库中存储的信息来验证密码。如果密码得到认证，则系统产生访问令牌。您运行的每个进程均使用此访问令牌的副本。

也可根据高速缓存凭证获取访问令牌。通过系统认证之后，操作系统会高速缓存您的凭证。当不能联系域控制器时，可以在高速缓存中引用上一次登录的访问令牌。

访问令牌包括所属全部组的有关信息：本地组和各种域组（全局组、域本地组和通用组）。

注：使用远程连接时，即使启用了访问令牌支持，使用客户机认证的组查找也不受支持。

要启用访问令牌支持，必须使用 **db2set** 命令更新 DB2®_GRP_LOOKUP 注册表变量。更新此注册表变量时的选项包括：

- TOKEN

  此选项启用访问令牌支持以在定义用户帐户的位置查找用户所属的全部组。此位置通常位于 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）服务器的域或本地。

- TOKENLOCAL

  此选项启用访问令牌支持以在 DB2 UDB 服务器上查找用户所属的全部本地组。

- TOKENDOMAIN

  此选项启用访问令牌支持以在域上查找用户所属的全部域组。

启用访问令牌支持时，有几种限制会影响帐户管理基础结构。启用此支持后，DB2 UDB 收集有关连接至数据库的用户的组信息。CONNECT 或 ATTACH 请求成功之后，与其它授权标识相关的后续操作仍需要使用常规组枚举。嵌套全局组、域本地组和高速缓存凭证的访问令牌优点将不可用。例如，如果在连接后使用 SET SESSION_USER 在另一授权标识下运行，则仅使用常规组枚举来检查给予会话的新授权标识的权限。您仍需要对 DB2 UDB 知悉的各个授权标识授予和撤销显式特权，与对授权标识所属的组授予和撤销特权相反。

如果要向 SYSADM、SYSMAINT 或 SYSCTRL 分配组，则需要确保已分配的组既非嵌套全局组，也非域本地组，并且不需要高速缓存凭证能力。

应考虑使用 DB2_GRP_LOOKUP 注册表变量并指定组查找位置，以指示 DB2 UDB 应使用常规组枚举方法查找组的位置。例如，

```
db2set DB2_GRP_LOOKUP=LOCAL,TOKENLOCAL
```

这启用了访问令牌支持以枚举本地组。授权标识的组查找在 DB2 UDB 服务器上执行，与连接用户有所不同。

```
db2set DB2_GRP_LOOKUP=,TOKEN
```

这启用了访问令牌支持以在定义用户标识的位置枚举组。授权标识的组查找在定义用户标识的位置执行，与连接用户有所不同。

```
db2set DB2_GRP_LOOKUP=DOMAIN,TOKENDOMAIN
```

这启用了访问令牌支持以枚举域组。授权标识的组查找在定义用户标识的位置执行，与连接用户有所不同。

在使用 DYNAMICRULES RUN（此为缺省值）绑定的程序包中使用动态 SQL 的应用程序，是以运行应用程序的人士的特权运行的。在此情况下，已经提及的限制不适用。这将包括为使用 JDBC 和 DB2 CLI 而编写的应用程序。

可以对所有认证类型（CLIENT 认证除外）启用访问令牌支持。

注: 对于 Windows® NT 4.0 用户，如果 DB2 UDB 应用程序使用本地隐式连接，则访问令牌支持仅支持进程级的安全性上下文。即对待应用程序中的所有线程犹如其在执行应用程序的用户的安全性上下文中运行一般。如果不同的线程需要不同的用户安全性上下文，则应考虑转用 Windows 2000 或更新版本；或考虑将 DB2 UDB 应用程序更改为使用显式连接。

相关概念:

- 第 183 页的『安装 DB2 通用数据库时的安全问题』

有关基于操作系统的安全性的详细信息

每个操作系统均提供了多种方法来管理安全性。本节讨论了一些与操作系统相关联的安全性问题。

注: 当使用 SERVER_ENCRYPT 认证时，DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）用于执行用户标识和密码加密的加密例程以及当使用 DATA_ENCRYPT 认证时用于执行用户标识、密码和用户数据加密的加密例程符合 FIPS 140-2 标准。

加密例程由 IBM Crypto for C（ICC）V1.2.1 提供。在 NIST Web 站点上可以找到 ICC 的 FIPS 140-2 确认证书编号 384，网址为：http://csrc.nist.gov/cryptval/140-1/140crt/140crt384.pdf

在 NIST Web 站点上还可以找到“安全策略”，网址为：http://csrc.nist.gov/cryptval/140-1/140sp/140sp384.pdf

为了以符合 FIPS 140-2 标准的方式安装 DB2 UDB，必须遵循“安全策略”的指引，并参考 5.3.2 节以获取更多信息。

仅在下列系统上才提供 FIPS 140-2 一致性: AIX、Microsoft Windows、Solaris operating environment、Linux 和 HP-UX。请参考“确认证书”和“安全策略”以获取有关受支持的系统的全部详细信息。

用户的 Windows NT 平台安全性注意事项

在定义帐户的机器上，属于本地“管理员”组的任何有效 DB2® 通用数据库（DB2 UDB）用户帐户都被授予“系统管理”（SYSADM）权限。

在 Windows® 域环境中，缺省情况下，只有属于“域控制器”上的“管理员”组的域用户才对实例具有 SYSADM 权限。因为 DB2 UDB 总是在定义帐户的机器上执行授权，所以向服务器上的本地“管理员”组添加域用户并不将域用户 SYSADM 权限授予该组。

注: 在域环境中（例如在 Windows NT® 中），DB2 UDB 只认证用户标识所属的符合要求和限制的前 64 个组。您具有的组可以多于 64 个。

为了避免向 PDC 上的“管理员”组添加域用户，应创建一个全局组并添加要授予其 SYSADM 权限的用户（域用户和本地用户）。为此，输入下列命令:

```
DB2STOP
DB2 UPDATE DBM CFG USING SYSADM_GROUP global_group
DB2START
```

相关概念:

- 第 187 页的『用户的 UNIX 平台安全性注意事项』

Windows 本地系统帐户支持

在 Windows® 平台上，DB2® 通用数据库（DB2 UDB）支持应用程序在本地隐式连接的本地系统帐户（LSA）环境下运行。编写在此帐户下运行的应用程序的开发者需要意识到 DB2 UDB 对模式名称以“SYS”开头的对象存在限制。因此，如果应用程序包含创建 DB2 UBD 对象的 DDL，则应如下编写:

- 对于静态 SQL，则应与 QUALIFIER 选项的值（而不是缺省值）绑定。
- 对于动态 SQL，要创建的对象应以 DB2 UDB 支持的模式名称显式限定，或者 CURRENT SCHEMA 寄存器必须设置为 DB2 UDB 支持的模式名称。

DB2 UDB 实例启动后，首次组查找请求收集 LSA 的组信息，并且实例重新启动前将不会刷新此信息。

相关概念:

- 第 183 页的『安装 DB2 通用数据库时的安全问题』

用户的 UNIX 平台安全性注意事项

DB2® 通用数据库（DB2 UDB）不支持 root 用户直接充当数据库管理员。应使用 **su - <instance owner>** 作为数据库管理员。

鉴于安全原因，我们建议不要使用实例名作为 Fenced ID。但若计划不使用受防护的 UDF 或存储过程，可以将 Fenced ID 设置为实例名，而不用创建另一个用户标识。

建议创建一个将被识别为与此组相关的用户标识。将受防护的 UDF 和存储过程的用户指定为实例创建脚本（**db2icrt ... -u <FencedID>**）的参数。若安装了 “DB2 客户机” 或 “DB2 软件开发者工具箱”，则不需要这样做。

相关概念:

- 第 187 页的『用户的 Windows NT 平台安全性注意事项』

实例目录的位置

在 UNIX® 上，**db2icrt** 命令在实例所有者主目录下创建 SQL 主库（sqllib）目录。

在 Windows® 操作系统上，该实例目录位于安装 DB2® 的目录的 /sqllib 子目录中。

相关概念:

- 第 14 页的『实例创建』

相关任务:

- 第 18 页的『创建附加实例』

安全插件

通过*安全插件*来执行 DB2® 通用数据库（DB2 UDB）中的认证。有关更多信息，请参阅 “安全插件”（来自 *Application Development Guide: Programming Client Applications*）。

服务器的认证方法

存取实例或数据库首先要求*认证*用户。每个实例的*认证类型*确定如何以及在何处验证用户。认证类型存储在服务器上的数据库管理器配置文件中。它是在创建实例时进行的初始设置。每个实例都有一个认证类型，该类型控制对数据库服务器和该服务器控制下的所有数据库的存取权。

若打算从联合数据库存取数据源，必须考虑数据源认证处理和联合认证类型的定义。

提供了下列认证类型:

SERVER

指定使用本地操作系统安全性在服务器上进行认证。若在连接尝试期间指定了用户标识和密码，则在服务器上将它们与有效的用户标识和密码比较，以确定是否允许该用户存取这个实例。这是缺省的安全性机制。

注:

1. 服务器代码检测一个连接是本地的还是远程的。对于本地连接，当认证是 SERVER 时，要使认证成功，不需要用户标识和密码。
2. 如果安装 DB2® 通用数据库（DB2 UDB）来设置 Common Criteria 认证配置，则必须指定 SERVER。

SERVER_ENCRYPT

指定服务器接受加密的 SERVER 认证方案。若未指定客户机认证，使用在服务器中选择的方法认证客户机。

如果客户机认证为 SERVER，则可通过将用户标识和密码传送至该服务器来认证客户机。如果客户机认证为 SERVER_ENCRYPT，可通过传送加密的用户标识和加密的密码来认证客户机。

若在客户机中指定了 SERVER_ENCRYPT，并且在服务器中指定了 SERVER，则会由于认证级别不匹配而返回错误。

CLIENT

指定使用操作系统安全性在调用应用程序所在的数据库分区上执行认证。在客户机节点上，将在一个连接尝试期间指定的用户标识和密码与有效的用户标识和密码的组合比较，以确定是否允许此用户标识 存取这个实例。不在数据库服务器上执行其它认证。这有时称为单一注册。

若用户执行本地登录或客户机登录，则只有该本地客户机工作站识别该用户。

若远程实例有 CLIENT 认证，则另两个参数确定最终的认证类型：*trust_allclnts* 和 *trust_clntauth*。

只用于 TRUSTED 客户机的 CLIENT 级安全性:

可信的客户机是具有可靠的、本地安全性系统的客户机。具体地说，除 Windows® 9x 操作系统外，所有客户机都是可信客户机。

当已选择认证类型 CLIENT 时，可选择一个附加选项来保护其操作环境没有固有安全性的客户机。

要阻止不安全的客户机存取系统，管理员可将 *trust_allclnts* 参数设置为 NO 来选择 “可信客户机认证”。这意味着所有可信平台都可以代表服务器认证用户。在服务器上认证不可信的客户机，必须提供一个用户标识和密码。使用 *trust_allclnts* 配置参数来指示您是否信赖客户机。此参数的缺省值是 YES。

注: 可以信赖所有客户机（*trust_allclnts* 为 YES），而让一些客户机作为没有本机保密安全性系统来认证的那些客户机。

甚至对于可信的客户机，您也可能希望在服务器上完成认证。要指示在哪里验证可信的客户机，使用 *trust_clntauth* 配置参数。此参数的缺省值是 CLIENT。

注: 仅对于可信的客户机，若在试图 CONNECT 或 ATTACH 时没有显式提供用户标识或密码，则对用户的验证在客户机上进行。*trust_clntauth* 参数仅用于确定在什么位置验证 USER 或 USING 子句上提供的信息。

要阻止所有客户机（来自 DB2 OS/390® 和 z/OS™ 版、DB2 VM 和 VSE 版以及 DB2 iSeries™ 版的 DRDA® 客户机除外）存取系统，将 *trust_allclnts* 参数设置为 DRDAONLY。只有这些客户机受信赖，因此才可执行客户机端的认证。所有其它客户机必须提供用户标识和密码，以供服务器认证。

trust_clntauth 参数用于确定在何处认证以上客户机：若 *trust_clntauth* 是 “client”，则在客户机处进行认证。若 *trust_clntauth* 是 “server”，没有提供密码时在客户机处认证，而提供密码时则在服务器处认证。

表 5. 使用 TRUST_ALLCLNTS 和 TRUST_CLNTAUTH 参数组合的认证方式。

TRUST_ ALLCLNTS	TRUST_ CLNTAUTH	不提供密码的不可信的非 DRDA 客户机认证	提供密码的不可信的非 DRDA 客户机认证	不提供密码的可信的非 DRDA 客户机认证	提供密码的可信的非 DRDA 客户机认证	不提供密码的 DRDA 客户机认证	提供密码的 DRDA 客户机认证
YES	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT	CLIENT

表 5. 使用 TRUST_ALLCLNTS 和 TRUST_CLNTAUTH 参数组合的认证方式。 (续)

TRUST_ ALLCLNTS	TRUST_ CLNTAUTH	不提供密码的不可信的非 DRDA 客户机认证	提供密码的不可信的非 DRDA 客户机认证	不提供密码的可信的非 DRDA 客户机认证	提供密码的可信的非 DRDA 客户机认证	不提供密码的 DRDA 客户机认证	提供密码的 DRDA 客户机认证
YES	SERVER	CLIENT	SERVER	CLIENT	SERVER	CLIENT	SERVER
NO	CLIENT	SERVER	SERVER	CLIENT	CLIENT	CLIENT	CLIENT
NO	SERVER	SERVER	SERVER	CLIENT	SERVER	CLIENT	SERVER
DRDAONLY	CLIENT	SERVER	SERVER	SERVER	SERVER	CLIENT	CLIENT
DRDAONLY	SERVER	SERVER	SERVER	SERVER	SERVER	CLIENT	SERVER

KERBEROS

当 DB2 UDB 客户机和服务器均位于支持 Kerberos 安全协议的操作系统上时，使用此项。通过使用传统密码术来创建共享密钥，Kerberos 安全协议作为第三方认证服务执行认证。此密钥成为用户的凭证，在所有请求本地或网络服务的场合中，都使用它来验证用户的标识。此密钥消除了将用户名和密码作为纯文本通过网络传送这一需要。使用 Kerberos 安全协议，您能够对远程 DB2 UDB 服务器使用单一注册。在运行 Windows 2000、AIX® 和 Solaris Operating Environment 的客户机和服务器上支持 KERBEROS 认证类型。

Kerberos 认证工作原理如下所示:

1. 用户对域控制器上的 Kerberos 密钥分发中心（KDC）使用域帐户认证登录客户机。密钥分发中心将授予凭单的凭单（TGT）发送至客户机。
2. 在连接的第一阶段，服务器将目标主体名称发送至客户机，该主体名称是 DB2 UDB 服务器服务的服务帐户名。通过使用服务器的目标主体名称和授予目标的凭证，客户机向授予凭证的服务（TGS）请求服务凭单，该凭证也驻留在域控制器中。如果客户机的授予凭单的凭单和服务器的目标主体名称都有效，则 TGS 向客户机发出服务凭单。记录在数据库目录中的主体名称现在可能被指定为 name/instance@REALM。（此外，这是使用 DB2 UDB 版本 7.1 及其后续版本的 Windows 2000 接受的当前 DOMAIN\userID 和 userID@xxx.xxx.xxx.com 格式。）
3. 客户机通过通信信道（例如，它可能是 TCP/IP）将此服务凭单发送至服务器。
4. 服务器验证客户机的服务凭单。如果客户机的服务凭单有效，则认证完成。

可能会对客户机上的数据库进行编目，并对服务器的目标主体名称显式指定 Kerberos 认证类型。使用此方法，可忽略连接的第一个阶段。

如果指定了用户标识和密码，则客户机将请求该用户帐户的授予凭单的凭单并将其用于认证。

KRB_SERVER_ENCRYPT

指定服务器接受 KERBEROS 认证或加密的 SERVER 认证方案。若客户机认证是 KERBEROS，则使用 Kerberos 安全性系统认证客户机。如果客户机认证是 SERVER_ENCRYPT，则使用用户标识和加密密码认证客户机。如果未指定客户机认证，如有可能，客户机将使用 Kerberos，否则它将使用密码加密。对于其它客户机认证类型，返回一个认证错误。不能将客户机的认证类型指定为 KRB_SERVER_ENCRYPT

注: Kerberos 认证类型只有在运行 Windows 2000、Windows XP、Windows Windows Server 2003 和 AIX 操作系统以及 Solaris Operating Environment 的客户机和服务器上才受支持。另外，客户机和服务器都必须属于同一个 Windows 域或属于可信域。在服务器支持 Kerberos 且某些（但并非全部）客户机支持 Kerberos 认证的情况下，应使用此认证类型。

DATA_ENCRYPT

服务器接受加密的 SERVER 认证方案和用户数据的加密。认证显式地以与使用 SERVER_ENCRYPT 显示的相同方式工作。有关更多信息，请参阅认证类型。

使用此认证类型时，加密以下用户数据:

- SQL 语句。
- SQL 程序变量数据。
- 从处理 SQL 语句和包括数据描述的服务器中输出的数据。
- 查询输出的某些或所有答案集合数据。
- 大对象（LOB）数据流动。
- SQLDA 描述符。

DATA_ENCRYPT_CMP

服务器接受加密的 SERVER 认证方案和用户数据的加密。另外，此认证类型允许与不支持 DATA_ENCRYPT 认证类型的下层产品兼容。允许这些产品与 SERVER_ENCRYPT 认证类型连接，且无需加密用户数据。支持新认证类型的产品必须使用它。此认证类型仅在服务器的数据库管理器配置文件中才有效，用于 CATALOG DATABASE 命令时无效。

GSSPLUGIN

指定服务器使用 GSS-API 插件来执行认证。如果未指定客户机认证，服务器将服务器支持的插件列表（包括在 *srvcon_gssplugin_list* 数据库管理器配置参数中列示的任何 Kerberos 插件）返回至客户机。客户机从列表中选择在客户机插件目录中找到的第一个插件。如果客户机不支持列表中的任何插件，则使用 Kerberos 认证方案（如果返回的话）认证客户机。如果客户机认证是 GSSPLUGIN 认证方案，则使用列表中第一个受支持的插件来认证客户机。

GSS_SERVER_ENCRYPT

指定服务器接受插件认证或加密的服务器认证方案。如果通过插件执行客户机认证，则使用服务器支持的插件列表中第一个客户机支持的插件来认证客户机。

如果未指定客户机认证且在执行隐式连接（即，建立连接时，客户机不提供用户标识和密码），则服务器返回服务器支持的插件列表、Kerberos 认证方案（如果列表中的某个插件是基于 Kerberos 的）和加密的服务器认证方案。使用客户机插件目录中找到的第一个受支持的插件来认证客户机。如果客户机不支持列表中的任何插件，则使用 Kerberos 认证方案认证客户机。如果客户机不支持 Kerberos 认证方案，则使用加密的服务器认证方案来认证客户机，且连接将因为缺少密码而失败。如果 DB2 UDB 提供的 Kerberos 插件存在于操作系统上或者为 *srvcon_gssplugin_list* 数据库管理器配置参数指定基于 Kerberos 的插件，则客户机支持 Kerberos 认证方案。

如果未指定客户机认证且在执行显式连接（即，同时提供用户标识和密码），则认证类型等价于 SERVER_ENCRYPT。

注:

1. 因为对配置文件本身的存取受到配置文件中信息的保护，所以在更改认证信息时，不要无意中将自己锁定在实例之外。下列数据库管理器配置文件参数控制对实例的存取:
   - AUTHENTICATION *
   - SYSADM_GROUP *
   - TRUST_ALLCLNTS
   - TRUST_CLNTAUTH
   - SYSCTRL_GROUP
   - SYSMAINT_GROUP

   * 指示两个最重要的参数以及最可能引起问题的那些参数。

   可以采取一些措施来确保这种情况不会发生: 如果意外将自己锁在 DB2 UDB 系统之外，所有平台上都提供了一个故障保险选项，它将允许您使用具有较高特权的本地操作系统安全性用户重设通常的 DB2 UDB 安全性检查来更新数据库管理器配置文件。此用户始终具有更新数据库管理器配置文件并校正该问题的特权。但是，这种绕过安全性检查的做法只限于对数据库管理器配置文件进行本地更新。不能以远程方式或对任何其它 DB2 UDB 命令使用故障保险用户。此特殊用户被标识为:
   - UNIX® 平台: 实例所有者
   - NT 平台: 本地“管理员”组的成员
   - 其它平台: 由于在其它平台上没有本地安全性，因此所有用户无论如何都要通过本地安全性检查

相关概念:

- 第 192 页的『远程客户机的认证注意事项』
- 第 193 页的『分区数据库认证注意事项』
- 第 333 页的『DB2 Windows NT 版和 Windows NT 安全性简介』

相关参考:

- 『authentication - 认证类型配置参数』（*《管理指南: 性能》*）
- 『trust_allclnts - 信赖所有客户机配置参数』（*《管理指南: 性能》*）
- 『trust_clntauth - 可信的客户机认证配置参数』（*《管理指南: 性能》*）

远程客户机的认证注意事项

当对数据库进行编目以便远程存取时，可在数据库目录条目中指定认证类型。

对于使用 DB2® Connect 来存取的数据库: 如果未指定值，则采用 SERVER 认证。

该认证类型不是必需的。如果未指定它，则缺省下客户机将使用 SERVER_ENCRYPT。但是，如果服务器不支持 SERVER_ENCRYPT，则客户机将尝试使用服务器支持的值重试。如果服务器支持多个认证类型，则客户机不会从中进行选择，而是返回错误。返回此错误以确保使用正确的认证类型。在这种情况下，客户机必须使用受支持的认证类型对数据库进行编目。如果指定了认证类型，且指定的值与服务器上的值相匹配，认证会立即开始。如果检测出不匹配，则 DB2 Universal Database™（DB2 UDB）（DB2

通用数据库）会尝试恢复。恢复可能导致更多的流来协调差异或导致错误（如果 DB2 UDB 不能恢复的话）。在不匹配的情况下，认为服务器上的值是正确的。

相关概念:

- 第 188 页的『服务器的认证方法』

分区数据库认证注意事项

在一个分区数据库中，必须为数据库的每个分区定义同一组用户和组。若这些定义不相同，则用户也许是被授权在不同的分区上做不同的事情。建议所有分区保持一致。

相关概念:

- 第 188 页的『服务器的认证方法』

Kerberos 认证详细信息

DB2® 通用数据库（DB2 UDB）为 AIX® V5.2、Solaris Operating Environment V8、Red Hat Enterprise Linux Server Advanced Server 2.1（32 位 Intel）和 Windows® 2000 及更新的操作系统上的 Kerberos 认证协议提供支持。

提供的 Kerberos 支持是名为 “IBMkrb5” 的 GSS-API 安全插件，此插件既可用作服务器认证插件，也可用作客户机认证插件。库分别位于 UNIX® 和 Linux 的 `sqllib/security{32|64}/plugin/IBM/{client|server}`目录中以及 Windows 的 `sqllib/security/plugin/IBM{client|server}` 目录中。

注: 对于 64 位 Windows，插件库名为 `IBMkrb564.dll`。此外，UNIX 和 Linux 插件的实际插件源代码 `IBMkrb5.C` 可从 `sqllib/samples/security/plugins` 目录中获得。

尝试将 Kerberos 认证与 DB2 UDB 配合使用之前，强烈建议您先深入地了解 Kerberos 的使用和配置。

Kerberos 的描述和简介

Kerberos 是第三方网络认证协议，它使用共享密钥系统，在不安全的网络环境中安全地认证用户。此协议最初于二十世纪八十年代末期在 MIT 开发而成，最新版本的 Kerberos 5 由因特网工程任务组织（IETF）RFC 1510 予以描述。（IETF 的 URL 是 http://www.ieft.org。RFC 1510 的标题为 “The Kerberos Network Authentication Service (V5)”。）它使用三层系统，其中的加密凭单（由名为 “Kerberos 密钥分发中心” （简称 KDC）的单独服务器提供）在应用程序服务器和客户机之间交换，而不在文本用户标识和密码对之间交换。这些加密服务凭单（称为*凭证*）具有有限的生存期，仅为客户机和服务器所知。这样可减少安全风险，即使凭单在网络上被拦截。每个用户（在 Kerberos 术语中称为*主体*）拥有与 KDC 共享的专用加密密钥。向 KDC 注册的主体和电脑集合称为*域*。

Kerberos 的重要特征在于允许单一登录环境，用户只需在 Kerberos 域内向资源验证其标识一次即可。使用 DB2 UDB 时，这意味着用户可与 DB2 UDB 服务器连接，而无需提供用户标识或密码。另一优势在于，由于使用主体的中央资源库，从而简化了用户标识管理。最后，Kerberos 支持相互认证，允许客户机验证服务器的标识。

Kerberos 安装

DB2 UDB 和其 Kerberos 支持依赖于在所有涉及的机器上正确安装和配置的 Kerberos 层，然后才涉及 DB2 UDB。这包括但不限于下列需求:

1. 客户机、服务器和主体必须属于相同的域，或其它可信的域（在 Windows 术语中称为“可信的域”）
2. 创建合适的主体
3. 如果合适的话，则创建服务器密钥表文件
4. 所有涉及的机器必须同步系统时钟（Kerberos 通常允许 5 分钟的时间偏差，否则获取凭证时可能出现预认证错误。）

有关安装和配置 Kerberos 的详细信息，请参阅随已安装的 Kerberos 产品一起提供的文档。

DB2 UDB 唯一关心的是能否根据连接应用程序（即认证）提供的凭证成功创建 Kerberos 安全性上下文。其它 Kerberos 功能（如消息签名或加密）将不可用。此外，如果可用的话，则支持相互认证。

Kerberos 必备软件如下所示:

- 具有 IBM® Network Authentication Service (NAS) Toolkit 1.3 的 AIX V5.2
- 具有 SEAM（Sun Enterprise Authentication Mechanism）和 IBM NAS Toolkit 1.3 的 Solaris Operating Environment V8
- 具有 `krb5-libs` 和 `krb5-workstation`文件集的 Red Hat Enterprise Linux Advanced Server 2.1
- Windows 2000（和更新的操作系统）Server

Kerberos 和客户机主体

主体可以用两部分或多部分格式（即 *name@REALM* 或 *name/instance@REALM*）找到。由于“name”部分将用于授权标识（AUTHID）映射中，该名称必须遵循 DB2 UDB 命名规则。这意味着名称长度最长可达 30 个字符并且它必须遵循选择所使用字符的现有限制。（AUTHID 映射将在后面的主题中进行讨论。）

注: Windows 将 Kerberos 主体与域用户直接关联。这意味着 Kerberos 认证不可用于未与域相关联 Windows 机器。此外，Windows 只支持两部分名称（即 *name@domain*）。

主体本身必须能够获得出站凭证，拥有此凭证即可请求和接收目标数据库的服务凭单。这通常是通过 UNIX 或 Linux 上的 **kinit** 命令实现的，并在登录 Windows 时隐式执行。

Kerberos 和授权标识映射

与存在范围通常仅限于单台机器的操作系统用户标识不同，Kerberos 主体能够在除自己以外的域中获得认证。通过使用域名来完全限定主体以避免出现重复的主体名称。在 Kerberos 中，全限定主体采用 `name/instance@REALM` 形式，其中实例字段实际上可能是由“/”分隔开来的多个实例（即 `name/instance1/instance2@REALM`），或许可能被同时忽略。明显的限制是域名在网络内定义的所有域中必须是唯一的。对于 DB2 UDB 而言，问题在于为了提供主体至 AUTHID 的简单映射，即主体名称之间一对一的映射，

也就是需要全限定主体中的“名称”和 AUTHID。由于 AUTHID 在 DB2 UDB 中用作缺省模式并应该以简便方式和逻辑方式派生，所以需要简单映射。因此，数据库管理员需要小心下列潜在问题:

- 来自不同域但具有相同名称的主体将被映射至相同的 AUTHID。
- 名称相同但实例不同的主体将被映射至相同的 AUTHID。

考虑到上述问题，提供以下建议:

- 在所有将访问 DB2 UDB 服务器的可信域中为名称保持唯一的名称空间
- 不论实例如何，所有名称相同的主体应属于同一用户。

Kerberos 和服务器主体

在 UNIX 或 Linux 上，假设 DB2 UDB 实例的服务器主体名称为 `<instance name>/<fully qualified hostname>@REALM`。由于初始化时服务器名由插件报告给 DB2 UDB，所以此主体必须能够接受 Kerberos 安全性上下文，并且必须在启动 DB2 UDB 实例之前已存在。

在 Windows 上，服务器主体被视为用来启动 DB2 UDB 服务器的域帐户。例外情况是实例可能由本地 SYSTEM 帐户启动，在此情况下，服务器主体名称作为 `host/<hostname>` 报告；只有客户机和服务器均属于 Windows 域时，此主体名称才有效。

Windows 不支持超过两部分的名称。当 Windows 客户机尝试与 UNIX 服务器连接时，将引起问题。因此，如果需要与 UNIX Kerberos 的互操作性，则可能需要在 Windows 域中设置 Kerberos 主体至 Windows 帐户的映射。（有关相关指示信息，请参阅相应的 Microsoft® 文档）

Kerberos 密钥表文件

UNIX 或 Linux 上希望接受安全性上下文请求的每个 Kerberos 服务必须将其凭证放置在*密钥表*文件中。这适用于 DB2 UDB 作为服务器主体使用的主体。只对缺省密钥表文件搜索服务器的密钥。有关将密钥添加到密钥表文件的指示信息，请参阅随 Kerberos 产品提供的文档。

Windows 上并无密钥表文件的概念，系统自动处理存储并获取主体的凭证句柄。

Kerberos 和组

Kerberos 是不会处理组概念的认证协议。因此，DB2 UDB 依赖本地操作系统来获取 Kerberos 主体的组列表。对于 UNIX 或 Linux，要求每个主体均需存在一个等价的系统帐户。例如，对于主体 `name@REALM`，DB2 UDB 通过查询本地操作系统以获取操作系统用户 *name* 所属的全部组名来收集组信息。如果操作系统用户不存在，则 AUTHID 将只属于 PUBLIC 组。另一方面，Windows 自动将域帐户与 Kerberos 主体相关联，无需其它步骤来创建单独的操作系统帐户。

在客户机上启用 Kerberos 认证

clnt_krb_plugin 数据库管理器配置参数应更新为正在使用的 Kerberos 插件的名称。在支持的平台上，此参数应设置为 `IBMkrb5`。如果 AUTHENTICATION 参数设置为 KERBEROS 或 KRB_SERVER_ENCRYPT，则此参数将通知 DB2 UDB，它可以将 Kerberos 用于连接和本地实例级别操作。否则，不会提供客户机端 Kerberos 支持。

注: 不会执行任何检查以验证 Kerberos 支持是否可用。

（可选）在客户机上对数据库进行编目时，可能会指定认证类型:

```
db2 catalog db testdb at node testnode authentication kerberos target
   principal service/host@REALM
```

但是，如果未提供认证信息，则服务器向客户机发送服务器主体的名称。

在服务器上启用 Kerberos 认证

srvcon_gssplugin_list 数据库管理器配置参数应该用服务器 Kerberos 插件名进行更新。虽然此参数可能包含受支持的插件的列表，但可能只会指定一个 Kerberos 插件。但是，如果此字段为空，并且 AUTHENTICATION 设置为 KERBEROS 或 KRB_SERVER_ENCRYPT，则提供并使用缺省的 Kerberos 插件（IBMkrb5）。如果根据 Kerberos 认证是用于所有情况还是只用于入局连接来决定其使用情况，则 AUTHENTICATION 或 SVRCON_AUTH 参数应设置为 KERBEROS 或 KRB_SERVER_ENCRYPT。

创建 Kerberos 插件

创建 Kerberos 插件时，应考虑以下若干注意事项:

- 将 Kerberos 插件编写为 GSS-API 插件，此插件具有明显的异常，函数指针数组中的 *plugintype* 必须设置为 DB2SEC_PLUGIN_TYPE_KERBEROS，而此函数指针数组已返回至初始化函数中的 DB2 UDB。
- 在某些情况下，服务器可能将服务器主体名称报告给客户机。同样，由于 DRDA® 规定主体名称为 GSS_C_NT_USER_NAME 格式（server/host@REALM），因此不应以 GSS_C_NT_HOSTBASED_SERVICE 格式（service@host）指定主体名称。
- 通常情况下，可以由 KRB5_KTNAME 环境变量指定缺省密钥表文件。但是，由于服务器插件将在 DB2 UDB 引擎进程中运行，因此可能无法存取此环境变量。

相关概念:

- 第 188 页的『服务器的认证方法』

特权、权限级别和数据库权限

*特权*使用户能够创建或存取数据库资源。*权限级别*提供一个方法，以将特权及较高级别数据库管理器维护和实用程序操作分组。*数据库权限*使用户可以在数据库级别上执行活动。可以同时使用特权、权限级别和数据库权限来控制对数据库管理器及其数据库对象的存取。用户只可访问他们对其拥有所需特权、权限级别或数据库权限的那些对象，DB2® 通用数据库（DB2 UDB）确定何时对已认证的用户执行权限检查。

数据库管理器要求对每个用户特别授权（显式或隐式均可）以使用执行特定任务所需的每项数据库函数。授予用户显式权限或特权（数据库目录中的 GRANTEETYPE 为 U）。授予用户所属组的隐式权限或特权（数据库目录中的 GRANTEETYPE 为 G）。因此，要创建表，必须授权用户创建表；要改变表，必须授权用户改变表，等等。

第 197 页的图 3 举例说明权限及其控制范围（数据库和数据库管理器）之间的关系。

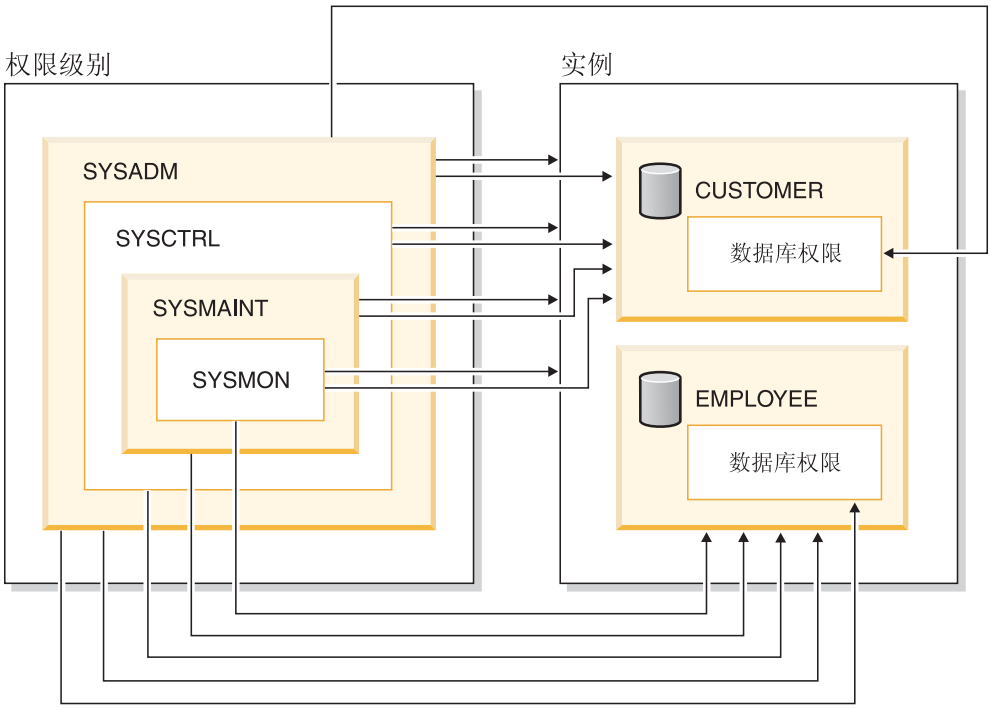


图 3. 权限的层次结构

用户或组可以具有下列一个或多个权限或特权:

- 管理权限:
  - SYSADM（系统管理员）

    SYSADM 权限级别提供对数据库管理器所创建和维护的全部资源的控制。系统管理员拥有所有 DBADM、SYSCTRL、SYSMAINT 和 SYSMON 权限并有权授予和撤销 DBADM 权限。

    拥有 SYSADM 权限的用户负责控制数据库管理器并确保数据的安全和完整性。SYSADM 权限提供对数据库中所有对象的隐式特权、控制哪些用户可以访问数据库管理器，并控制此访问的范围。有关 SYSADM 权限的更多信息，请参阅“系统管理权限（SYSADM）”。
  - DBADM（数据库管理员）

    DBADM 数据库权限提供对单个数据库的管理权限。此数据库管理员拥有创建对象、发出数据库命令和存取表数据所需的特权。数据库管理员还可以授权和撤销 CONTROL 和各个特权。有关 DBADM 权限的更多信息，请参阅“数据库管理权限（DBADM）”。
- 系统控制权限:
  - SYSCTRL（系统控制）

    SYSCTRL 权限级别提供对影响系统资源的操作的控制。例如，具有 SYSCTRL 权限的用户可以创建、更新、停止或删除数据库。此用户还可以停止实例，但不能存取表数据。具有 SYSCTRL 权限的用户还具有 SYSMON 权限。有关 SYSCTRL 权限的更多信息，请参阅“系统控制权限（SYSCTRL）”。
  - SYSMAINT（系统维护）

SYSMAINT 权限级别提供在所有与实例关联的数据库上执行维护操作所需的权限。具有 SYSMAINT 权限的用户可以更新数据库配置、备份数据库或表空间、复原现有数据库并监视数据库。类似于 SYSCTRL，SYSMAINT 不提供对表数据的存取。具有 SYSMAINT 权限的用户也具有 SYSMON 权限。有关 SYSMAINT 权限的更多信息，请参阅“系统维护权限（SYSMAINT）”。

- SYSMON（系统监视器权限）

  SYSMON 权限级别提供使用数据库系统监视器所需的权限。有关 SYSMON 权限的更多信息，请参阅“系统监视器权限（SYSMON）”

- 数据库权限

  要执行诸如创建表或例程或者用于将数据装入表等活动，需要特定数据库权限。有关更多信息，请参阅“数据库权限”。

- 特权:

  对数据库对象执行活动（例如，创建和删除索引）需要使用特权。特权严格定义用户可以执行的任务。例如，用户可能具有在表上创建索引的特权，但不具有在同一表上创建触发器的特权。

  – CONTROL 特权

    拥有对象的 CONTROL 特权允许用户存取该数据库对象，并授予和撤销其他用户对该对象的特权。

    注: CONTROL 特权仅适用于表、视图、昵称、索引和程序包。

    如果其他用户需要该对象的 CONTROL 特权，则具有 SYSADM 或 DBADM 权限的用户必须将 CONTROL 特权授予该对象。无法从对象所有者处撤销 CONTROL 特权。

    在某些情况下，对象的创建者自动获得对象的 CONTROL 特权。有关更多信息，请参阅“对象创建、所有权和特权”。

  – 可以授予各个特权以允许用户对特定对象执行特定任务。

    具有管理权限（SYSADM 或 DBADM）或 CONTROL 特权的用户可以授予和撤销用户的特权。

    各个特权和数据库权限允许特定功能，但不包括授予其他用户相同特权或权限的权限。将表、视图、模式、程序包、例程和序列特权授予其他用户的权限可以通过 GRANT 语句上的 WITH GRANT OPTION 扩展至其他用户。但是，WITH GRANT OPTION 不允许授予该特权的人在授权后撤销该特权。必须具有 SYSADM 权限、DBADM 权限或 CONTROL 特权才能撤销该特权。

    还可以将特权授予 PUBLIC。无论各个用户先前是否被授予特权，PUBLIC 特权适用于所有用户（权限名称），包括任何将来用户。

  – 可以将隐式特权授予具有执行程序包的特权的用户。当用户可运行应用程序时，他们不一定需要对程序包内使用的数据对象具有显式特权。

可以将各个特权或权限的任何组合授予一个用户或组。当特权与对象关联时，对象必须已存在。例如，除非先前已创建一个表，否则不能授予用户对该表的 SELECT 特权。

注: 当授予一个授权名权限和特权且没有使用该授权名创建用户时，必须小心。稍后，可以使用该授权名创建一个用户，并且该用户自动接收与该授权名相关的所有权限和特权。

REVOKE 语句用于撤销先前授予的特权。在 DB2 UDB 中，撤销权限名称的特权会撤销所有权限名称授予的特权。

撤销某个权限名称的特权不会撤销由该权限名称授予任何其它权限名称的相同特权。例如，假定 CLAIRE 将 SELECT WITH GRANT OPTION 授予 RICK，然后 RICK 将 SELECT 授予 BOBBY 和 CHRIS。如果 CLAIRE 撤销 RICK 的 SELECT 特权，则 BOBBY 和 CHRIS 仍保留该选择特权。

相关概念:

- 第 200 页的『系统管理权限（SYSADM）』
- 第 201 页的『系统控制权限（SYSCTRL）』
- 第 202 页的『系统维护权限（SYSMAINT）』
- 第 202 页的『数据库管理权限（DBADM）』
- 第 204 页的『LOAD 权限』
- 第 204 页的『数据库权限』
- 第 206 页的『模式特权』
- 第 208 页的『表空间特权』
- 第 208 页的『表和视图特权』
- 第 210 页的『程序包特权』
- 第 210 页的『索引特权』
- 第 211 页的『序列特权』
- 第 211 页的『控制对数据库对象的存取』
- 第 215 页的『对程序包的间接特权』
- 第 211 页的『例程特权』
- 第 199 页的『对象创建、所有权和特权』
- 第 203 页的『系统监视器权限（SYSMON）』

对象创建、所有权和特权

创建对象时，可将对象的*所有权*分配给一个权限名称。所有权意味着用户有权在任何 SQL 语句中引用此对象。

在模式内创建对象时，此语句的授权标识必须具有在隐式或显式指定模式中创建对象所需的特权。即权限名称必须是模式的所有者或对模式拥有 CREATEIN 特权的名称。

注: 创建表空间、缓冲池或数据库分区组时，此需求不适用。这些对象并非在模式中创建。

创建对象时，语句的授权标识是此对象的所有者。

注: 存在一个例外情况。如果对 CREATE SCHEMA 语句指定 AUTHORIZATION 选项，则作为 CREATE SCHEMA 操作部分创建的其它任何对象由 AUTHORIZATION

选项指定的授权标识所有。但是，初始 CREATE SCHEMA 操作后在模式中创建的任何对象由与特定 CREATE 语句关联的授权标识所有。

例如，语句 `CREATE SCHEMA SCOTTSTUFF AUTHORIZATION SCOTT CREATE TABLE T1 (C! INT)` 创建模式 SCOTTSTUFF 和表 SCOTTSTUFF.T1，这两者均属 SCOTT 所有。假设用户 BOBBY 对 SCOTTSTUFF 模式拥有 CREATEIN 特权，并在 SCOTTSTUFF.T1 表上创建索引。因为索引在模式之后创建，因此 BOBBY 在 SCOTTSTUFF.T1 上拥有索引。

根据要创建的对象类型向对象所有者分配特权:

- 对新创建的表、索引和程序包隐式授予 CONTROL 特权。此特权允许对象创建程序存取数据库对象，并授予和撤销其他用户对此对象的特权。如果其他用户需要该对象的 CONTROL 特权，则具有 SYSADM 或 DBADM 权限的用户必须将 CONTROL 特权授予该对象。对象所有者无法撤销 CONTROL 特权。
- 如果对象所有者对视图定义引用的所有表、视图和昵称具有 CONTROL 特权，则对新创建的视图隐式授予 CONTROL 特权。
- 其它对象（如触发器、例程、序列、表空间和缓冲池）没有与其关联的 CONTROL 特权。但是，对象所有者自动获得与对象关联的各项特权（并可使用 GRANT 语句的 WITH GRANT 选项向其他用户提供这些特权（如果支持））。此外，对象所有者可以改变或删除对象，或为对象添加注释。这些权限对于对象所有者是隐式的且不能撤销。

相关概念:

- 第 196 页的『特权、权限级别和数据库权限』
- 第 206 页的『模式特权』
- 第 208 页的『表空间特权』
- 第 208 页的『表和视图特权』
- 第 210 页的『程序包特权』
- 第 210 页的『索引特权』
- 第 211 页的『序列特权』
- 第 211 页的『例程特权』

有关特权、权限和授权的详细信息

本节中将会讨论每种权限，然后接着讨论不同的特权。

系统管理权限（SYSADM）

SYSADM 权限级别是最高级别的管理权限。具有 SYSADM 权限的用户可以运行实用程序，发出数据库及数据库管理器命令，存取在此数据库管理器实例内任何数据库中任何表内的数据。它提供控制实例中所有数据库对象的能力，这些对象包括数据库、表、视图、索引、程序包、模式、服务器、别名、数据类型、函数、过程、触发器、表空间、数据库分区组、缓冲池和事件监视器。

对 *sysadm_group* 配置参数指定的组指定 SYSADM 权限。通过您的平台上使用的安全性设施，在数据库管理器外控制该组的成员资格。

只有具有 SYSADM 权限的用户才可以执行下列功能:

- 迁移数据库
- 更改数据库管理器配置文件（包括指定具有 SYSCTRL、SYSMAINT 或 SYSMON 权限的组）
- 授予 DBADM 权限

注: 当具有 SYSADM 权限的用户创建数据库时，自动授予该用户在数据库上的显式 DBADM 权限。如果从 SYSADM 组中除去此数据库创建者，而且想防止用户作为 DBADM 存取该数据库，则必须显式撤销用户的 DBADM 权限。

相关概念:

- 第 201 页的『系统控制权限（SYSCTRL）』
- 第 202 页的『系统维护权限（SYSMAINT）』
- 第 219 页的『数据加密』
- 第 203 页的『系统监视器权限（SYSMON）』

系统控制权限（SYSCTRL）

SYSCTRL 权限是最高级别的系统控制权限。此权限提供对数据库管理器实例和其数据库执行维护和实用程序操作的能力。这些操作可以影响系统资源，但是它们并不允许直接存取此数据库中的数据。系统控制权限是为管理包含敏感数据的数据库管理器实例的用户而设计的。

对 *sysctrl_group* 配置参数指定的组指定 SYSCTRL 权限。如果指定了一个组，则通过平台上使用的安全设施从数据库管理器外部控制该组的成员资格。

只有具有 SYSCTRL 权限或更高权限的用户才可以执行下列操作:

- 更新数据库、节点或分布式连接服务（DCS）目录
- 强制用户从系统注销
- 创建或删除一个数据库
- 删除、创建或改变一个表空间
- 复原为新数据库

另外，具有 SYSCTRL 权限的用户可以执行具有“系统维护权限”（SYSMAINT）和“系统监视器权限”（SYSMON）的用户的函数。

具有 SYSCTRL 权限的用户也有与一个数据库连接的隐式特权。

注: 当具有 SYSCTRL 权限的用户创建数据库时，将自动授予他们对此数据库的显式 DBADM 权限。若从 SYSCTRL 组中除去了此数据库创建者，而且您也想阻止他们作为 DBADM 存取该数据库，则必须显式撤销此 DBADM 权限。

相关概念:

- 第 202 页的『系统维护权限（SYSMAINT）』
- 第 202 页的『数据库管理权限（DBADM）』
- 第 203 页的『系统监视器权限（SYSMON）』

系统维护权限（SYSMAINT）

SYSMAINT 权限是第二级别的系统控制权限。此权限提供对数据库管理器实例及其数据库执行维护和实用程序操作的能力。这些操作可以影响系统资源，但是它们并不允许直接存取此数据库中的数据。系统维护权限是为某些用户设计的，这些用户维护包含敏感数据的数据库管理器实例中的数据库。

对 *sysmaint_group* 配置参数指定的组指定 SYSMAINT 权限。如果指定了一个组，则通过平台上使用的安全设施从数据库管理器外部控制该组的成员资格。

只有具有 SYSMAINT 或更高系统权限的用户才可以执行下列操作:

- 更新数据库配置文件
- 备份数据库或表空间
- 复原为现有的数据库
- 执行前滚恢复
- 启动或停止实例
- 复原表空间
- 运行跟踪
- 拍摄数据库管理器实例或其数据库的数据库系统监视器快照

具有 SYSMAINT、DBADM 或更高权限的用户可以执行下列操作:

- 查询表空间的状态
- 更新日志历史文件
- 停顿表空间
- 重组表
- 使用 **RUNSTATS** 实用程序收集目录统计信息。

具有 SYSMAINT 权限的用户也有与数据库连接的隐式特权，并且可以执行具有“系统监视器权限”（SYSMON）的用户的函数。

相关概念:

- 第 202 页的『数据库管理权限（DBADM）』
- 第 203 页的『系统监视器权限（SYSMON）』

数据库管理权限（DBADM）

DBADM 权限是第二个最高级别的管理权限。它只适用于特定的数据库，并允许用户运行特定的实用程序，发出数据库命令，存取该数据库中的任何表中的数据。当授予 DBADM 权限时，也同时授予 BINDADD、CONNECT、CREATETAB、CREATE_EXTERNAL_ROUTINE、CREATE_NOT_FENCED_ROUTINE、IMPLICIT_SCHEMA、QUIESCE_CONNECT 和 LOAD 数据库权限。只有具有 SYSADM 权限的用户才可以授予或撤销 DBADM 权限。具有 DBADM 权限的用户可以授予其他用户对该数据库的特权，并可以从任何用户撤销任何特权，而不管是谁授予的。

只有具有 DBADM 或更高权限的用户才可以执行下列操作:

- 读取日志文件

- 创建、激活和删除事件监视器。

具有 DBADM、SYSMAINT 或更高权限的用户可以执行下列操作:

- 查询表空间的状态
- 更新日志历史文件
- 停顿表空间。
- 重组表
- 使用 **RUNSTATS** 实用程序收集目录统计信息。

注: DBADM 只能对您持有 DBADM 权限的数据库执行上述功能。

相关概念:

- 第 200 页的『系统管理权限（SYSADM）』
- 第 201 页的『系统控制权限（SYSCTRL）』
- 第 202 页的『系统维护权限（SYSMAINT）』
- 第 204 页的『LOAD 权限』
- 第 204 页的『数据库权限』
- 第 206 页的『隐式模式权限（IMPLICIT_SCHEMA）注意事项』

系统监视器权限（SYSMON）

SYSMON 权限提供拍摄数据库管理器实例或其数据库的数据库系统监视器快照的能力。SYSMON 权限已分配给 *sysmon_group* 配置参数指定的组。如果指定组，则通过平台上使用的安全性设施在数据库管理器外控制此组中的成员资格。

SYSMON 权限使用户可以运行下列命令:

- GET DATABASE MANAGER MONITOR SWITCHES
- GET MONITOR SWITCHES
- GET SNAPSHOT
- LIST ACTIVE DATABASES
- LIST APPLICATIONS
- LIST DCS APPLICATIONS
- RESET MONITOR
- UPDATE MONITOR SWITCHES

SYSMON 权限使用户可以使用下列 API:

- db2GetSnapshot - 获取快照
- db2GetSnapshotSize - 估计 db2GetSnapshot() 输出缓冲区所需的大小
- db2MonitorSwitches - 获取 / 更新监视器开关
- db2ResetMonitor - 复位监视器

SYSMON 权限使用户可以使用下列 SQL 表函数:

- 无需预先运行 SYSPROC.SNAPSHOT_FILEW 的所有快照表函数。

  SYSPROC.SNAPSHOT_FILEW 拍摄快照并将其内容保存到文件中。如果通过空输入参数调用任何快照表函数，则返回文件内容，而不是实时系统快照。

具有 SYSADM、SYSCTRL 或 SYSMAINT 权限级别的用户也拥有 SYSMON 权限。

相关参考:

- 『sysmon_group – 系统监视器权限组名配置参数』（*《管理指南: 性能》*）

LOAD 权限

在数据库级具有 LOAD 权限以及对表具有 INSERT 特权的用户可以使用 **LOAD** 命令来将数据装入到表中。

如果先前的装入操作是用来插入数据的装入操作，在数据库级具有 LOAD 权限且对表具有 INSERT 特权的用户可以执行 **LOAD RESTART** 或 **LOAD TERMINATE** 操作。

在数据库级具有 LOAD 权限同时对表具有 INSERT 和 DELETE 特权的用户可以使用 **LOAD REPLACE** 命令。

如果先前的装入操作是装入替换，则还必须对该用户授予 DELETE 特权，该用户才能执行 **LOAD RESTART** 或 **LOAD TERMINATE** 操作。

如果将异常表用作装入操作的一部分，则用户对异常表必须具有 INSERT 特权。

具有此权限的用户可以执行 **QUIESCE TABLESPACES FOR TABLE**、**RUNSTATS** 和 **LIST TABLESPACES** 命令。

相关概念:

- 『Privileges, authorities, and authorizations required to use Load』（*Data Movement Utilities Guide and Reference*）
- 第 208 页的『表和视图特权』

相关参考:

- 『RUNSTATS Command』（*Command Reference*）
- 『QUIESCE TABLESPACES FOR TABLE Command』（*Command Reference*）
- 『LIST TABLESPACES Command』（*Command Reference*）
- 『LOAD Command』（*Command Reference*）

数据库权限

第 205 页的图 4 显示数据库权限。

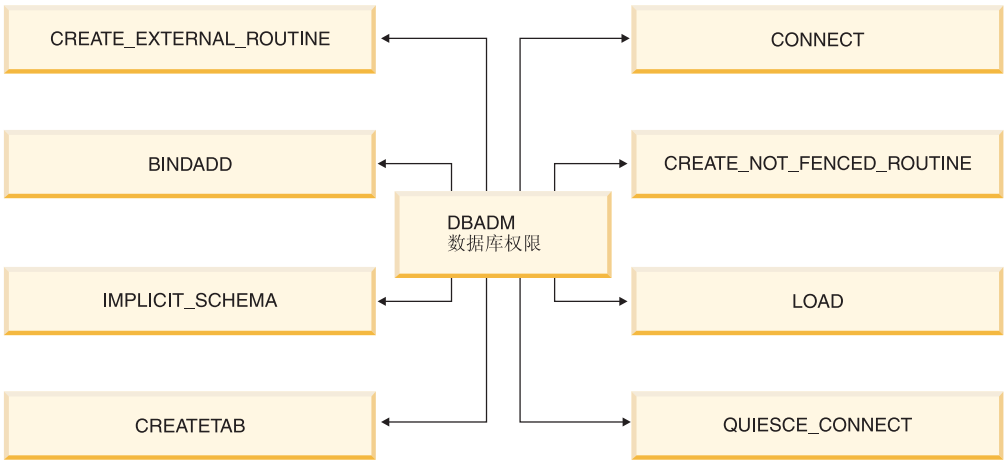


图 4. 数据库权限

数据库权限涉及对作为整体的数据库所执行的操作。任何具有 DBADM 权限的用户拥有如下一组完整的数据库权限:

- CONNECT 允许用户存取数据库。
- BINDADD 允许用户在数据库中创建新的程序包。
- CREATETAB 允许用户在数据库中创建新的表。
- CREATE_EXTERNAL_ROUTINE 允许用户创建过程以供应用程序和数据库的其他用户使用。
- CREATE_NOT_FENCED_ROUTINE 允许用户创建“未防护的”用户定义的函数（UDF）或过程。因为数据库管理器不会阻止这些 UDF 或过程保护它的存储器或控制块，所以“未防护的” UDF 或过程必须经过严格的测试。（因此，允许按“未防护的”形式运行的、一个编写和测试都不严密的 UDF 或过程，可能在系统中引起严重问题。）

  注: 将 CREATE_EXTERNAL_ROUTINE 自动授权给任何已授予 CREATE_NOT_FENCED_ROUTINE 权限的用户。
- 使用 CREATE 语句和尚不存在的模式名来创建对象时，IMPLICIT_SCHEMA 允许任何用户隐式创建一个模式。SYSIBM 成为隐式创建的模式的所有者，并且授予 PUBLIC 在此模式中创建对象的特权。
- LOAD 允许用户将数据装入表中。
- QUIESCE_CONNECT 允许用户在数据库处于停顿状态时存取数据库。

只有具有 SYSADM 或 DBADM 权限的用户才可以授予和撤销其他用户的数据库特权。

注: 创建数据库时，会自动将下列数据库权限授予 PUBLIC:

- CREATETAB 数据库权限
- BINDADD 数据库权限
- CONNECT 数据库权限
- IMPLICIT_SCHEMA 数据库权限
- 对 USERSPACE1 表空间的 USE 特权
- 对系统目录视图的 SELECT 特权

要除去任何数据库权限，DBADM 或 SYSADM 必须从 PUBLIC 显式撤销数据库权限。

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

隐式模式权限（IMPLICIT_SCHEMA）注意事项

当创建新数据库时，PUBLIC 会被授予 IMPLICIT_SCHEMA 数据库权限。使用此权限，任何用户都可以创建一个对象并指定尚未存在的模式名，来创建一个模式。SYSIBM 成为隐式创建的模式的所有者，并且授予 PUBLIC 在此模式中创建对象的特权。

若数据库需要控制可隐式创建模式对象的人，则应当从 PUBLIC 撤销 IMPLICIT_SCHEMA 数据库权限。一旦撤销了该权限，就只有三种创建模式对象的方法:

- 任何用户都可以在一个 CREATE SCHEMA 语句上使用他们自己的授权名来创建一个模式。
- 任何具有 DBADM 权限的用户都可以显式地创建任何尚不存在的模式，并且可以选择是否指定另一个用户作为此模式的所有者。
- 任何具有 DBADM 权限的用户都具有 IMPLICIT_SCHEMA 数据库权限（与 PUBLIC 无关），这样，他们在创建其它数据库对象时可以使用任何名称隐式地创建一个模式。SYSIBM 成为隐式创建的模式的所有者，而且 PUBLIC 具有在此模式中创建对象的特权。

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

模式特权

模式特权属于对象特权类别。对象特权显示在第 207 页的图 5 中。

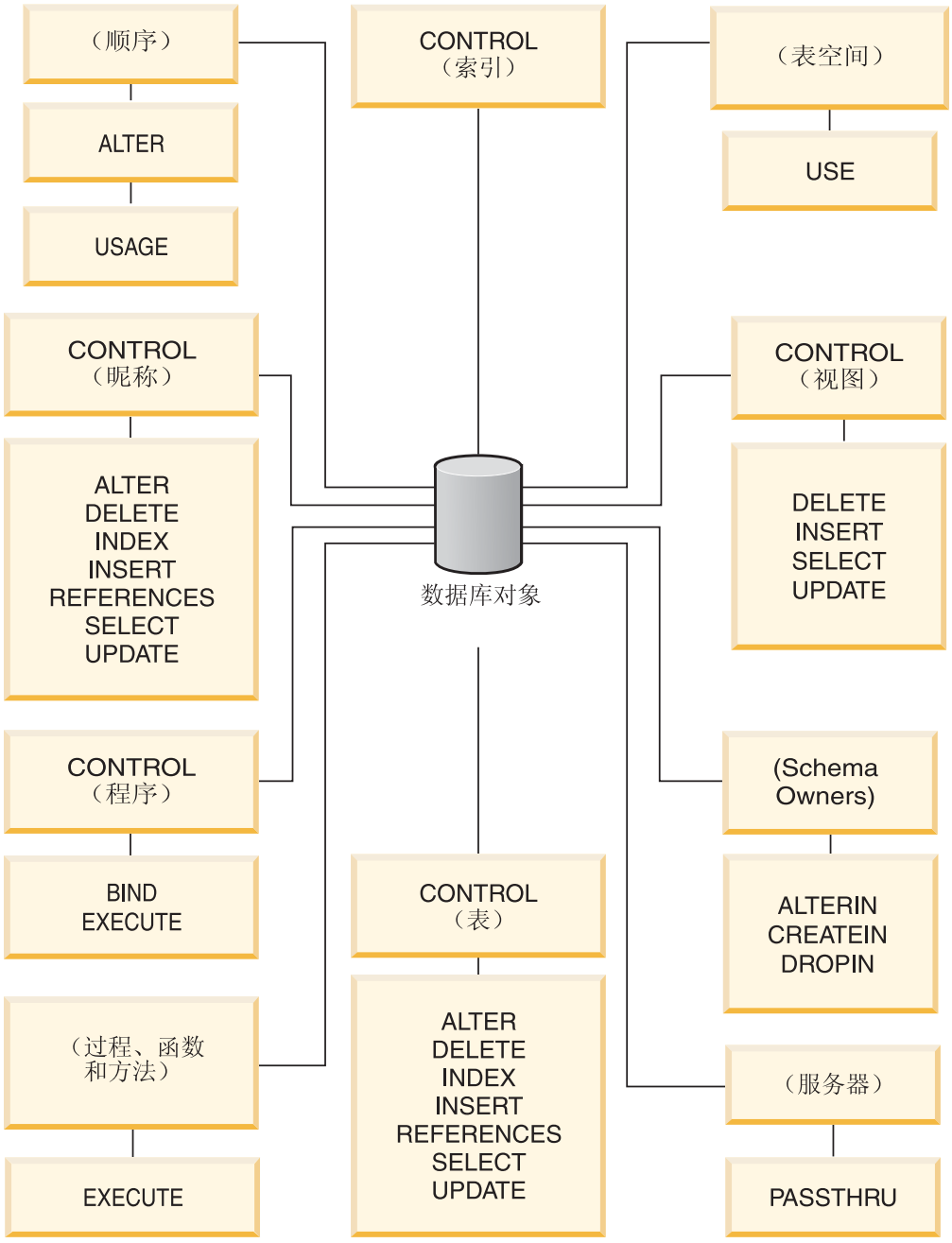


图 5. 对象特权

模式特权涉及到对一个数据库中的模式所执行的操作。可以授予用户下列任何一个特权:

- CREATEIN 允许用户在模式中创建对象。
- ALTERIN 允许用户在模式中改变对象。
- DROPIN 允许用户在模式中删除对象。

模式所有者具有所有这些特权，并且有将这些特权授予其他用户的能力。在模式对象中操纵的对象包括: 表、视图、索引、程序包、数据类型、函数、触发器、过程和别名。

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

相关参考:

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

表空间特权

表空间特权涉及对数据库中表空间的操作。可以将对表空间的 USE 特权授予用户，这允许它们在该表空间中创建表。

表空间的所有者（通常是具有 SYSADM 或 SYSCTRL 权限的创建者）具有 USE 特权，有能力将此特权授予别人。在缺省情况下，当创建数据库时，将表空间 USERSPACE1 的 USE 特权授予 PUBLIC（虽然可以撤销此特权）。

USE 特权不能与 SYSCATSPACE 或任何系统临时表空间配合使用。

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

相关参考:

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

表和视图特权

表和视图特权涉及到对一个数据库中的表或视图所执行的操作。用户必须对数据库具有 CONNECT 权限，才可使用下列任何一个特权:

- CONTROL 给用户提供对表或视图的所有特权，包括删除它以及授予和撤销各个表特权的能力。必须具有 SYSADM 或 DBADM 权限，才可授予 CONTROL。一个表的创建者自动接收表的 CONTROL 特权。仅当视图的创建者对视图定义中引用的所有表、视图和昵称具有 CONTROL 特权时，或他们具有 SYSADM 或 DBADM 权限时，才自动接收 CONTROL 特权。
- ALTER 允许用户修改表，例如，为表添加列或唯一约束。具有 ALTER 特权用户还可以 COMMENT ON 一个表，或者表的一列。有关可能对表执行的修改的信息，请参阅 ALTER TABLE 和 COMMENT 语句。
- DELETE 允许用户从表或视图中删除行。
- INDEX 允许用户对表创建一个索引。索引创建者自动具有索引的 CONTROL 特权。
- INSERT 允许用户将行插入表或视图，并运行 IMPORT 实用程序。
- REFERENCES 允许用户创建和删除一个外键，并指定该表为关系中的父表。用户可能只对特定的列拥有此特权。
- SELECT 允许用户检索表或视图中的行，对表创建视图，并运行“导出”实用程序。
- UPDATE 允许用户更改表或视图中的条目，或表或视图中的一个或多个特定列的条目。用户只能对特定的列拥有此特权。

将这些特权授予其他用户的特权也可在 GRANT 语句上使用 WITH GRANT OPTION 来授予。

注: 当授予一个用户或组对某个表的 CONTROL 特权时，将使用 WITH GRANT OPTION 自动授予对该表的所有其它特权。若接着从某个用户撤销了对该表的 CONTROL 特权，该用户将仍然保留自动授予的其它特权。要撤销使用 CONTROL 特权授予的所有特权，必须显式撤销每个单独的特权，或者在 REVOKE 语句上指定 ALL 关键字，例如:

```
REVOKE ALL
  ON EMPLOYEE FROM USER HERON
```

当使用类型表时，表和视图的特权有特别的意义。

注: 可在一个表层次结构的每一层单独授予特权。因此，对于类型表的层次结构中的一个超表，被授予对该超表的特权的用户也可间接影响任何子表。但是，若对一个子表持有需要的特权，则用户只能直接对该子表操作。

在一个表层次结构中，表之间的超表 / 子表关系表示象 SELECT、UPDATE 和 DELETE 这样的操作将影响该操作的目标表和其所有子表（若有）中的行。这种行为称为*可替换能力*。例如，假设创建了一个类型为 Employee_t 的 Employee 表，它具有类型为 Manager_t 的子表 Manager。经理是一种特殊的职员，这由结构化类型 Employee_t 与 Manager_t 之间的类型 / 子类型关系，和对应的在表 Employee 与 Manager 之间的表 / 子表关系来指示。由于这种关系，SQL 查询:

```
SELECT * FROM Employee
```

将返回职员和经理的对象标识和 Employee_t 属性。类似地，更新操作:

```
UPDATE Employee SET Salary = Salary + 1000
```

将给经理和正式职员加薪一千元。

对 Employee 具有 SELECT 特权的用户可以执行这个 SELECT 操作，即使他们对 Manager 没有显式 SELECT 特权。但是，将不允许这类用户直接对 Manager 子表执行 SELECT 操作，因此，这类用户将不能存取 Manager 表的任何非继承列。

类似地，对 Employee 具有 UPDATE 特权的用户将能够对 Manager 执行 UPDATE 操作，从而影响正规的职员和经理，即使该用户对 Manager 表不具有显式的 UPDATE 特权。但是，将不允许这类用户直接对 Manager 子表执行 UPDATE 操作，因而，这类用户将不能更新 Manager 表的任何非继承列。

相关概念:

- 第 210 页的『索引特权』

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

相关参考:

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

程序包特权

程序包是一个数据库对象，它包含数据库管理器以适合于特定应用程序的最有效方式存取数据所需的信息。程序包特权使用户能够创建和操纵程序包。用户必须对数据库具有 CONNECT 权限，才可使用下列任何特权:

- CONTROL 给用户提供重新绑定、删除或执行程序包的能力，以及将那些特权授予其他用户的能力。程序包的创建者自动接收此特权。具有 CONTROL 特权的用户被授予 BIND 和 EXECUTE 特权，还可以使用 GRANT 语句将这些特权授予其他用户。（如果使用 WITH GRANT OPTION 授予特权，则接收 BIND 或 EXECUTE 特权的用户可以依次将此特权授予其他用户。）要授予 CONTROL 特权，用户必须具有 SYSADM 或 DBADM 权限。
- 对程序包的 BIND 特权允许用户重新绑定或绑定该程序包以及添加具有相同程序包名和创建者的新程序包版本。
- EXECUTE 允许用户执行或运行程序包。

注: 所有程序包特权适用于共享相同程序包名和创建者的所有 VERSION。

除这些程序包特权外，BINDADD 数据库特权还允许用户创建新程序包或重新绑定数据库中的现有程序包。

按昵称引用的对象需要对包含该对象的数据源遍历认证检查。另外，程序包用户必须对数据源中的数据源对象拥有适当特权或权限级别。

包含昵称的程序包可能需要其它授权步骤，因为与 DB2 系列数据源通信时，DB2® 通用数据库（DB2 UDB）使用动态 SQL。在数据源运行程序包的授权标识必须有恰当的权限，才可在该数据源动态执行此程序包。

相关概念:

- 第 204 页的『数据库权限』

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

索引特权

索引或索引规范的创建者自动接收该索引的 CONTROL 特权。索引的 CONTROL 特权实际是删除此索引的能力。要授予对一个索引的 CONTROL 特权，用户必须具有 SYSADM 或 DBADM 权限。

表级别 INDEX 特权允许用户对该表创建索引。

昵称级别 INDEX 特权允许用户对该昵称创建索引规范。

相关概念:

- 第 208 页的『表和视图特权』

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

序列特权

序列的创建者自动接收对序列的 USAGE 和 ALTER 特权。要使用序列的 NEXT VALUE 和 PREVIOUS VALUE 表达式，需要具有 USAGE 特权。要允许其他用户使用 NEXT VALUE 和 PREVIOUS VALUE 表达式，必须将序列特权授予 PUBLIC。这就允许所有用户使用具有指定序列的表达式。

序列上的 ALTER 特权允许用户执行诸如重新启动序列或更改将来序列值增量之类的任务。序列的创建者可以授予其他用户 ALTER 特权，并且如果使用 WITH GRANT OPTION，则这些用户可以依次将这些特权授予其他用户。

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

相关参考:

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

例程特权

执行特权涉及对所有类型的例程（如数据库中的函数、过程和方法）执行的操作。一旦具有 EXECUTE 特权，用户就可以调用例程、创建源于该例程（仅应用于函数）的函数以及在任何 DDL 语句（如 CREATE VIEW 和 CREATE TRIGGER）中引用例程。

定义外部存储过程、函数或方法的用户接收 EXECUTE WITH GRANT 特权。如果通过 WITH GRANT OPTION 将 EXECUTE 特权授予其他用户，则该用户可以依次将 EXECUTE 特权授予另一用户。

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

控制对数据库对象的存取

控制数据存取需要了解直接和间接特权、管理权限和程序包。本节说明这些主题并提供一些示例。

直接授予的特权存储在系统目录中。

控制授权有三种方法:

- 通过使用 GRANT 和 REVOKE 语句控制的特权来控制显式授权
- 通过创建和删除对象来控制隐式授权
- 间接特权与程序包相关

注: 在 GRANT 或 REVOKE 语句或 “控制中心” 中使用时，数据库组名一定不能超过 8 个字符。虽然接受长度超过 8 个字符的数据库组名，但是当属于该组的用户存取数据库对象时，较长的名称会产生错误消息。

相关概念:

- 第 221 页的『将系统目录用于安全性说明』

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

有关控制对数据库对象的存取的详细信息

控制对数据库对象的存取是通过使用 GRANT 和 REVOKE 语句实现的。此部分同时也会讨论隐式存取授权和间接特权。

授权特权

限制:

要授予对大多数数据库对象的特权，用户必须对该对象具有 SYSADM 权限、DBADM 权限或 CONTROL 特权；或者，用户必须持有使用 WITH GRANT OPTION 授予的特权。只能授予对现有对象的特权。要将 CONTROL 特权授予其他用户，此用户必须具有 SYSADM 或 DBADM 权限。要授予 DBADM 特权，用户必须具有 SYSADM 权限。

过程:

GRANT 语句允许授权用户授予特权。可以在一条语句中将一个特权授予一个或多个授权名；或授予 PUBLIC，这使该特权可供所有用户使用。注意授权名可以是单独的用户，也可以是组。

在存在具有相同名称的用户和组的操作系统上，应当指定是将该特权授予用户还是授予组。GRANT 和 REVOKE 语句都支持关键字 USER和 GROUP。如果未使用这些可选的关键字，则数据库管理器检查操作系统安全性设施，以确定该授权名是标识用户还是组。若该授权名可能既是用户又是组，则返回一个错误。

以下示例将 EMPLOYEE 表的 SELECT 特权授予用户 HERON:

```
GRANT SELECT
   ON EMPLOYEE TO USER HERON
```

以下示例将 EMPLOYEE 表的 SELECT 特权授予组 HERON:

```
GRANT SELECT
     ON EMPLOYEE TO GROUP HERON
```

相关概念:

- 第 211 页的『控制对数据库对象的存取』

相关任务:

- 第 213 页的『撤销特权』

相关参考:

- 『GRANT (Database Authorities) statement』（*SQL Reference, Volume 2*）
- 『GRANT (Index Privileges) statement』（*SQL Reference, Volume 2*）
- 『GRANT (Package Privileges) statement』（*SQL Reference, Volume 2*）
- 『GRANT (Schema Privileges) statement』（*SQL Reference, Volume 2*）

- 『GRANT (Table, View, or Nickname Privileges) statement』（*SQL Reference, Volume 2*）
- 『GRANT (Server Privileges) statement』（*SQL Reference, Volume 2*）
- 『GRANT (Table Space Privileges) statement』（*SQL Reference, Volume 2*）
- 『GRANT (Sequence Privileges) statement』（*SQL Reference, Volume 2*）
- 『GRANT (Routine Privileges) statement』（*SQL Reference, Volume 2*）

撤销特权

REVOKE 语句允许授权用户撤销先前已授予其他用户的特权。

限制:

要撤销对数据库对象的特权，必须对该对象具有 DBADM 权限、SYSADM 权限或 CONTROL 特权。注意，持有使用 WITH GRANT OPTION 授予的特权并不足以撤销该特权。要从另一个用户撤销 CONTROL 特权，必须具有 SYSADM 或 DBADM 权限。要撤销 DBADM 权限，必须具有 SYSADM 权限。只能撤销对现有对象的特权。

注: 不具有 DBADM 权限或 CONTROL 特权的用户，不能撤销他们使用 WITH GRANT OPTION 授予的特权。另外，由被撤销特权的人授予特权的那些人不会被撤销特权。

若从具有 DBADM 权限的用户撤销一个显式授予的表（或视图）特权，**将不会**从在该表上定义的其它视图撤销特权。这是因为视图特权可通过 DBADM 权限得到，并不依赖于基础表上的显式特权。

过程:

若已经把一个特权授予有相同名称的一个用户和一个组，当撤销此特权时必须指定 GROUP 或 USER 关键字。以下示例从用户 HERON 撤销对 EMPLOYEE 表的 SELECT 特权:

```
REVOKE SELECT
      ON EMPLOYEE FROM USER HERON
```

以下示例从组 HERON 撤销对 EMPLOYEE 表的 SELECT 特权:

```
REVOKE SELECT
      ON EMPLOYEE FROM GROUP HERON
```

注意从一个组中撤销特权并不能从该组的所有成员中撤销该特权。若单独的名称已被直接授予一个特权，则此名称将保留它，直到被直接撤销该特权为止。

若从一个用户撤销了表特权，根据已撤销的表特权，则也撤销对该用户创建的任何视图的特权。但是，只撤销系统隐式授予的那些特权。若该视图的一个特权是另一个用户直接授予的，则此特权仍然会被保留。

您可能会遇到这样的情况，您想要将一种特权授予一个组，然后仅从组中的其中一个成员撤销该特权。仅有两种方式执行该操作而不会接收到错误消息 SQL0556N:

- 您可以从组中除去该成员；或者创建具有较少成员的新组，并将特权授予新组。
- 可以从组中撤销特权，然后将特权授予各个用户（授权标识）。

注: 当从一个用户撤销对一个表或视图的 CONTROL 特权时，此用户仍继续有将特权授予其他用户的能力。当授予 CONTROL 特权时，用户也可接收使用 WITH GRANT OPTION 提供的所有其它特权。一旦撤销了 CONTROL，则使用 WITH GRANT OPTION 提供的所有其它特权会保留，一直到显式撤销它们为止。

根据被撤销的特权的所有程序包都标记为无效，但是若一个具有合适权限的用户重新绑定它们，则可以变得有效。若随后又将特权授予应用程序的绑定者，也可以重新构建程序包；运行此应用程序将触发一个成功的隐式重新绑定。若从 PUBLIC 撤销了特权，则所有由只能根据 PUBLIC 特权绑定的用户绑定的程序包都无效。若从用户撤销了 DBADM 权限，则该用户绑定的所有程序包全都无效，包括与数据库实用程序相关的那些程序包。试图使用标记为无效的程序包将引起系统试图重新绑定此程序包。若此重新绑定尝试失败，则发生错误（SQLCODE -727）。在这种情况下，必须由具有如下权限的用户显式地重新绑定程序包:

- 重新绑定程序包的权限
- 对程序包中使用的对象的适当权限

应当在撤销特权时重新绑定这些程序包。

如果根据一个或多个特权定义触发器或 SQL 函数，并且失去了这些特权中的一个或多个特权，则不能使用该触发器或 SQL 函数。

相关任务:

- 第 212 页的『授权特权』

相关参考:

- 『REVOKE (Database Authorities) statement』（*SQL Reference, Volume 2*）
- 『REVOKE (Index Privileges) statement』（*SQL Reference, Volume 2*）
- 『REVOKE (Package Privileges) statement』（*SQL Reference, Volume 2*）
- 『REVOKE (Schema Privileges) statement』（*SQL Reference, Volume 2*）
- 『REVOKE (Table, View, or Nickname Privileges) statement』（*SQL Reference, Volume 2*）
- 『REVOKE (Server Privileges) statement』（*SQL Reference, Volume 2*）
- 『REVOKE (Table Space Privileges) statement』（*SQL Reference, Volume 2*）
- 『REVOKE (Routine Privileges) statement』（*SQL Reference, Volume 2*）

通过创建和删除对象来管理隐式授权

过程:

数据库管理器隐式授予用户某种特权以创建数据库对象，如表或程序包。当具有 SYSADM 或 DBADM 权限的用户创建对象时，就授予了特权。类似地，当删除一个对象时，就除去了特权。

当创建的对象是表、昵称、索引或程序包时，用户会接收到该对象的 CONTROL 特权。当对象是视图时，只有在用户对该视图定义中引用的所有表、视图和昵称具有 CONTROL 特权时，才隐式授予此视图 CONTROL 特权。

当显式创建的对象是一个模式时，将使用 WITH GRANT OPTION 授予此模式所有者 ALTERIN、CREATEIN 和 DROPIN 特权。隐式创建的模式具有授予 PUBLIC 的 CREATEIN。

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

建立程序包的所有权

过程:

BIND 和 PRECOMPILE 命令创建或更改应用程序包。在任何一个命令上，使用 OWNER 选项来命名生成的程序包的所有者。命名程序包的所有者有简单规则:

- 任何用户可将自己指定为所有者。若未指定 OWNER 选项，则这是缺省值。
- 具有 SYSADM 或 DBADM 权限的标识可使用 OWNER 选项将任何授权标识指定为所有者。

并非所有可使用 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）数据库产品绑定程序包的操作系统都支持 OWNER 选项。

相关参考:

- 『BIND Command』（*Command Reference*）
- 『PRECOMPILE Command』（*Command Reference*）

对程序包的间接特权

对一个数据库中数据的存取可以由应用程序请求，也可由参与交互式工作站会话的人请求。程序包包含允许用户对许多数据库对象执行不同操作的语句。其中每个操作需要一个或多个特权。

授予绑定程序包的个人和 PUBLIC 的特权用于在绑定静态 SQL 时的权限检查。通过组授予的特权不用于在绑定静态 SQL 时的权限检查。除非在绑定程序包时指定 VALIDATE RUN，否则绑定程序包、具有有效*授权标识*的用户必须被显式授予执行该程序包中静态 SQL 语句所需的全部特权，或者通过 PUBLIC 被隐式授予需要的特权。若执行 BIND 时指定了 VALIDATE RUN，则此程序包中任何静态 SQL 语句的所有授权失败都不会导致 BIND 失败，那些 SQL 语句在运行时重新验证。当检查以确保用户具有合适的授权（BIND 或 BINDADD 特权）来绑定程序包时，PUBLIC、组和用户特权全*部*都会用到。

程序包可包括静态和动态的 SQL。要处理包含静态 SQL 的程序包，用户只需要对该程序包具有 EXECUTE 特权。然后对于该程序包中的任何静态 SQL，此用户可以间接获得该程序包绑定者的特权，但是只在此程序包所施加的限制内。

如果程序包包括动态 SQL，则在预编译或绑定程序包时，所需特权取决于为 DYNAMICRULES 指定的值。有关更多信息，请参阅描述动态 SQL 上 DYNAMICRULES 作用的主题。

相关概念:

- 第 216 页的『对包含别名的程序包的间接特权』
- 『Effect of DYNAMICRULES bind option on dynamic SQL』（*Application Development Guide: Programming Client Applications*）

相关参考:

- 『BIND Command』（*Command Reference*）

对包含别名的程序包的间接特权

当程序包包含对别名的引用时，对程序包创建者和程序包用户的授权处理比较复杂。当程序包创建者成功绑定包含别名的程序包时，程序包创建者不必通过在数据源处对别名引用的表和视图所做的认证检查或特权检查。但是，此程序包的执行者必须通过数据源的认证和权限检查。

例如，假设程序包创建者的 .SQC 文件包含几条 SQL 语句。一条静态语句引用了本地表。另一条动态语句引用了别名。当绑定此程序包时，使用程序包创建者的授权标识来验证对本地表和昵称的特权，但不对昵称标识的数据源对象执行检查。当另一个用户执行此程序包时，假设他对该程序包具有 EXECUTE 特权，则该用户不必通过对引用此表的语句所做的任何附加的特权检查。但是，对于引用别名的语句，执行此程序包的用户必须通过数据源的认证检查和特权检查。

当 .SQC 文件仅包含动态 SQL 语句以及表与昵称的混合引用时，对本地对象和昵称的 DB2® 通用数据库（DB2 UDB）权限检查是类似的。程序包用户必须通过在语句内设置的任何本地对象（表和视图）的特权检查，还要通过别名对象的特权检查（程序包用户必须通过在包含别名标识的对象的数据源处的认证检查和特权检查）。在这两种情况下，程序包的用户都必须具有 EXECUTE 特权。

程序包执行者的标识和密码用于所有数据源认证和特权处理。可通过创建用户映射更改此信息。

注: 不能在静态 SQL 中指定别名。不要对包含别名的程序包使用 DYNAMICRULES 选项（设置为 BIND）。

包含昵称的程序包可能需要其它授权步骤，原因是与 DB2 系列数据源通信时，DB2 UDB 使用动态 SQL。在数据源运行程序包的授权标识必须有恰当的权限，才可在该数据源动态执行此程序包。

相关概念:

- 第 215 页的『对程序包的间接特权』

使用视图控制对数据的存取

视图通过允许下列操作，提供了一种方法来控制对表的存取或扩展对表的特权:

- 只存取表的指定列。

  对于要求只存取一个表的特定列的用户和应用程序，授权用户可以创建一个视图来限制这些列只被需要它们的那些人存取。
- 只存取表的所有行的一个子集。

通过在一个视图定义的子查询中指定 WHERE 子句，授权用户可以限制通过一个视图存取的行。

- 只存取数据源表或视图中的行或列的一个子集。如果通过昵称存取数据源，可创建引用昵称的本地 DB2® 通用数据库（DB2 UDB）视图。这些视图可从一个或多个数据源引用别名。

  注: 因为可以创建一个视图来包含对多个数据源的别名引用，因此用户可从一个视图存取多个数据源中的数据。这些视图称为*多位置视图*。当将一个分布式环境中各敏感表的列信息连接在一起时，或当各个用户缺少在数据源需要的特定对象的特权时，这类视图可以发挥作用。

要创建视图，用户必须对此视图定义中引用的每个表、视图或昵称具有 SYSADM 权限、DBADM 权限、CONTROL 或 SELECT 特权。用户还必须能够在为此视图指定的模式中创建对象。即，对现有模式具有 CREATEIN 特权，或者，若此模式还不存在，则对数据库具有 IMPLICIT_SCHEMA 权限。

如果要创建引用昵称的视图，不需要对视图中昵称引用的数据源对象（表和视图）有其它权限；但是，当视图的用户存取该视图时，必须对基础数据源对象具有 SELECT 权限或等效的权限级别。

若用户在数据源处对基础对象（表和视图）没有合适的权限，可执行下列操作:

1. 以用户可存取的数据源表中的那些列为基础，创建一个数据源视图
2. 授予用户对此视图的 SELECT 特权
3. 创建一个引用此视图的别名

然后用户可发出引用新别名的 SELECT 语句来存取那些列。

以下方案提供了如何使用视图来限制存取信息的一个更详细的示例。

因种种原因，许多人可能需要存取 STAFF 表中的信息。例如:

- 人事部门需要能更新和查看整个表。

  通过授予组 PERSONNL 对 STAFF 表的 SELECT 和 UPDATE 特权，可以很容易地满足此要求:

  ```
  GRANT SELECT,UPDATE ON TABLE STAFF TO GROUP PERSONNL
  ```

- 各个部门的经理需要查看他们职员的工资信息。

  可以通过为每个部门经理创建一个视图来满足此要求。例如，可以为部门号为 51 的经理创建如下视图:

  ```
  CREATE VIEW EMP051 AS
     SELECT NAME,SALARY,JOB FROM STAFF
     WHERE DEPT=51
  GRANT SELECT ON TABLE EMP051 TO JANE
  ```

  具有授权名 JANE 的经理将象查询 STAFF 表一样查询 EMP051 视图。当存取 STAFF 表的 EMP051 视图时，此经理会看到如下信息:

NAME	SALARY	JOB
Fraye	45150.0	Mgr
Williams	37156.5	Sales

NAME	SALARY	JOB
Smith	35654.5	Sales
Lundquist	26369.8	Clerk
Wheeler	22460.0	Clerk

- 所有用户都需要能够找到其他职员。可以根据 STAFF 表的 NAME 列和 ORG 表的 LOCATION 列创建一个视图，并通过两个表各自的 DEPT 和 DEPTNUMB 列将这两个表连接，来满足此要求:

```
CREATE VIEW EMPLOCS AS
      SELECT NAME, LOCATION FROM STAFF, ORG
      WHERE STAFF.DEPT=ORG.DEPTNUMB
   GRANT SELECT ON TABLE EMPLOCS TO PUBLIC
```

存取职员位置视图的用户将看到如下信息:

NAME	LOCATION
Molinare	New York
Lu	New York
Daniels	New York
Jones	New York
Hanes	Boston
Rothman	Boston
Ngan	Boston
Kermisch	Boston
Sanders	Washington
Pernal	Washington
James	Washington
Sneider	Washington
Marenghi	Atlanta
O'Brien	Atlanta
Quigley	Atlanta
Naughton	Atlanta
Abrahams	Atlanta
Koonitz	Chicago
Plotz	Chicago
Yamaguchi	Chicago
Scoutten	Chicago
Fraye	Dallas
Williams	Dallas
Smith	Dallas
Lundquist	Dallas
Wheeler	Dallas
Lea	San Francisco
Wilson	San Francisco

NAME	LOCATION
Graham	San Francisco
Gonzales	San Francisco
Burke	San Francisco
Quill	Denver
Davis	Denver
Edwards	Denver
Gafney	Denver

相关任务:

- 第 109 页的『创建视图』
- 第 212 页的『授权特权』

使用审计设施监视对数据的存取

DB2® 通用数据库（DB2 UDB）审计设施为一系列预定义数据库事件生成审计跟踪，并允许您维护它。当不是阻止存取数据的设施时，审计设施可以监视对数据对象进行存取或修改的尝试，并可对这些尝试进行记录。

要使用审计设施管理员工具 **db2audit**，需要具有 SYSADM 权限。

相关概念:

- 第 229 页的『DB2 通用数据库（DB2 UDB）审计设施简介』

数据加密

安全性计划的一个部分可能包括加密您的数据。为此，可以使用加密和解密内置函数: ENCRYPT、DECRYPT_BIN、DECRYPT_CHAR 和 GETHINT。

ENCRYPT 函数使用基于密码的加密方法对数据进行加密。这些函数还允许您封装密码提示。密码提示嵌入在加密数据中。一旦加密，对数据进行解密的唯一方式是通过使用正确的密码来解密。选择使用这些函数的开发者应该对忘记的密码和不能用的数据如何管理进行计划。

ENCRYPT 函数的结果是 VARCHAR FOR BIT DATA（具有 32 631 的限制）。

只能加密 CHAR、VARCHAR 和 FOR BIT DATA。

DECRYPT_BIN 和 DECRYPT_CHAR 函数使用基于密码的解密对数据进行解密。

DECRYPT_BIN 始终返回 VARCHAR FOR BIT DATA，而 DECRYPT_CHAR 始终返回 VARCHAR。因为第一个自变量可能是 CHAR FOR BIT DATA 或 VARCHAR FOR BIT DATA，所以存在结果与第一个自变量不同的情况。

结果的长度取决于到下一个 8 字节边界的字节数。当指定可选提示参数时，结果的长度可能是数据自变量的长度加上 40 再加上到下一个 8 字节边界的字节数。或者，如果未指定可选提示参数，结果的长度可能是数据自变量的长度加上 8 再加上到下一个 8 字节边界的字节数。

GETHINT 函数返回封装的密码提示。密码提示是将帮助数据所有者回忆起密码的短语。例如，单词“大海”可以用作回忆密码“太平洋”的提示。

以下列两种方式中的一种方法确定用于对数据进行加密的密码:

- 密码自变量。密码是当调用 ENCRYPT 函数时显式传送的字符串。使用给出的密码对数据进行加密和解密。
- 加密密码专用寄存器。SET ENCRYPTION PASSWORD 语句对密码值进行加密，并将加密后的密码发送至数据库管理器以存储在专用寄存器中。未使用密码参数调用的 ENCRYPT、DECRYPT_BIN 和 DECRYPT_CHAR 函数使用 ENCRYPTION PASSWOED 专用寄存器中的值。ENCRYPTION PASSWORD 专用寄存器只以加密格式存储。

  专用寄存器的初始或缺省值是一个空字符串。

密码的有效长度在 6 和 127 之间，包括 6 和 127。提示的有效长度在 0 和 32 之间，包括 0 和 32。

相关参考:

- 『SET ENCRYPTION PASSWORD statement』（*SQL Reference, Volume 2*）
- 『DECRYPT_BIN and DECRYPT_CHAR scalar functions』（*SQL Reference, Volume 1*）
- 『ENCRYPT scalar function』（*SQL Reference, Volume 1*）
- 『GETHINT scalar function』（*SQL Reference, Volume 1*）

任务和必需的授权

并非所有机构都以相同方式来划分工作职责。表 6 列出了其它一些常见的职务、与这些职务通常对应的任务以及完成这些任务需要的权限或特权。

表 6. 常见职务、任务和必需的授权

职务	任务	必需的授权
部门管理员	监督部门系统；创建数据库	SYSCTRL 权限。若部门有其自己的实例，则为 SYSADM 权限。
安全性管理员	授予其他用户一些或所有授权和特权	SYSADM 或 DBADM 权限。
数据库管理员	设计、开发、操作、保护和维护一个或多个数据库	对一个或多个数据库的 DBADM 和 SYSMAINT 权限。某些情况下，为 SYSCTRL 权限。
系统操作员	监视数据库并执行备份功能	SYSMAINT 权限。
应用程序员	开发和测试数据库管理器应用程序；也可以创建测试数据表	对现有程序包的 BINDADD、BIND，对一个或多个数据库的 CONNECT 和 CREATETAB，某些特定模式特权，以及对某些表的特权的列表。 可能还需要 CREATE_EXTERNAL_ROUTINE。
用户分析员	通过检查系统目录视图来定义一个应用程序的数据需求	对目录视图的 SELECT；对一个或多个数据库的 CONNECT。

表 6. 常见职务、任务和必需的授权 (续)

职务	任务	必需的授权
程序最终用户	执行应用程序	对程序包的 EXECUTE; 对一个或多个数据库的 CONNECT。请参阅此表后的注释。
信息中心顾问	定义查询用户的数据需求; 通过创建表和视图, 并授予对数据库对象的存取权来提供数据	对一个或多个数据库的 DBADM 权限。
查询用户	发出 SQL 语句来检索、添加、删除或更改数据; 可以将结果作为表来保存	对一个或多个数据库的 CONNECT; 对要创建的表和视图的模式的 CREATEIN; 以及对某些表和视图的 SELECT、 INSERT、UPDATE、DELETE。

注: 如果应用程序包含动态 SQL 语句, 则“程序最终用户”除 EXECUTE 和 CONNECT 外可能还需要其它特权（如 SELECT、INSERT、DELETE 和 UPDATE）。

相关概念:

- 第 200 页的『系统管理权限（SYSADM）』
- 第 201 页的『系统控制权限（SYSCTRL）』
- 第 202 页的『系统维护权限（SYSMAINT）』
- 第 202 页的『数据库管理权限（DBADM）』
- 第 204 页的『LOAD 权限』
- 第 204 页的『数据库权限』

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

将系统目录用于安全性说明

有关每个数据库的信息自动在一个称为系统目录的视图集合中维护, 系统目录是在生成数据库时创建的。此系统目录描述表、列、索引、程序、特权和其它对象。

这些视图列示用户持有的特权以及授予每个特权的用户的标识:

SYSCAT.DBAUTH	列示数据库特权
SYSCAT.TABAUTH	列示表和视图的特权
SYSCAT.COLAUTH	列示列的特权
SYSCAT.PACKAGEAUTH	列示程序包特权
SYSCAT.INDEXAUTH	列示索引特权
SYSCAT.SCHEMAAUTH	列示模式特权
SYSCAT.PASSTHRUAUTH	列示服务器特权
SYSCAT.ROUTINEAUTH	列示例程（函数、方法和存储过程）特权

系统授予用户的特权将让 SYSIBM 作为授权者。SYSADM、SYSMAINT 和 SYSCTRL 未在系统目录中列出。

CREATE 和 GRANT 语句在系统目录中设置特权。具有 SYSADM 和 DBADM 权限的用户可以授予和撤销对系统目录视图的 SELECT 特权。

相关任务:

- 第 222 页的『检索已授予特权的授权名』
- 第 223 页的『检索具有 DBADM 权限的全部名称』
- 第 223 页的『检索授权存取表的名称』
- 第 224 页的『检索授予用户的所有特权』
- 第 224 页的『保护系统目录视图』

相关参考:

- 『SYSCAT.COLAUTH catalog view』（*SQL Reference, Volume 1*）
- 『SYSCAT.DBAUTH catalog view』（*SQL Reference, Volume 1*）
- 『SYSCAT.INDEXAUTH catalog view』（*SQL Reference, Volume 1*）
- 『SYSCAT.PACKAGEAUTH catalog view』（*SQL Reference, Volume 1*）
- 『SYSCAT.SCHEMAAUTH catalog view』（*SQL Reference, Volume 1*）
- 『SYSCAT.TABAUTH catalog view』（*SQL Reference, Volume 1*）
- 『SYSCAT.PASSTHRUAUTH catalog view』（*SQL Reference, Volume 1*）
- 『SYSCAT.ROUTINEAUTH catalog view』（*SQL Reference, Volume 1*）

有关使用系统目录来处理安全性问题的详细信息

本节综述了一些用于确定谁在数据库中具有什么特权的方法。

检索已授予特权的授权名

过程:

没有单个系统目录视图包含全部特权的信息。如下语句检索具有特权的所有授权名:

```
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'DATABASE' FROM SYSCAT.DBAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'TABLE   ' FROM SYSCAT.TABAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'PACKAGE ' FROM SYSCAT.PACKAGEAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'INDEX   ' FROM SYSCAT.INDEXAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'COLUMN  ' FROM SYSCAT.COLAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SCHEMA  ' FROM SYSCAT.SCHEMAAUTH
UNION
SELECT DISTINCT GRANTEE, GRANTEETYPE, 'SERVER  ' FROM SYSCAT.PASSTHRUAUTH
ORDER BY GRANTEE, GRANTEETYPE, 3
```

应定期将此语句检索到的列表与系统安全性设施中定义的用户名和组名的列表比较。然后，可以标识不再有效的那些授权名。

注: 若您支持远程数据库客户机，有可能只在远程客户机定义了此授权名，而没有在数据库服务器上定义。

相关概念:

- 第 221 页的『将系统目录用于安全性说明』

检索具有 DBADM 权限的全部名称

过程:

如下语句检索被直接授予 DBADM 权限的所有授权名:

```
SELECT DISTINCT GRANTEE FROM SYSCAT.DBAUTH
      WHERE DBADMAUTH = 'Y'
```

相关概念:

- 第 202 页的『数据库管理权限（DBADM）』
- 第 221 页的『将系统目录用于安全性说明』

检索授权存取表的名称

过程:

如下语句检索被直接授权存取具有限定符 JAMES 的表 EMPLOYEE 的所有授权名:

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
      WHERE TABNAME = 'EMPLOYEE'
        AND TABSCHEMA = 'JAMES'
   UNION
   SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
      WHERE TABNAME = 'EMPLOYEE'
        AND TABSCHEMA = 'JAMES'
```

要了解谁可以更新具有限定符 JAME 的表 EMPLOYEE，发出如下语句:

```
SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.TABAUTH
      WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
         (CONTROLAUTH  = 'Y' OR
          UPDATEAUTH   = 'Y' OR  UPDATEAUTH   = 'G')
   UNION
   SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.DBAUTH
      WHERE DBADMAUTH = 'Y'
   UNION
   SELECT DISTINCT GRANTEETYPE, GRANTEE FROM SYSCAT.COLAUTH
      WHERE TABNAME = 'EMPLOYEE' AND TABSCHEMA = 'JAMES' AND
      PRIVTYPE = 'U'
```

以上语句检索具有 DBADM 权限的任何授权名，以及被直接授予 CONTROL 或 UPDATE 特权的那些名称。但是，它不会返回只持有 SYSADM 权限的用户的授权名。

记住某些授权名可以是组，而不只是各个用户。

相关概念:

- 第 208 页的『表和视图特权』
- 第 221 页的『将系统目录用于安全性说明』

检索授予用户的所有特权

过程:

通过在系统目录视图上进行查询，用户可以检索他们持有的特权的列表和他们授予其他用户的特权的列表。例如，如下语句检索直接授予单独的授权名的数据库特权的列表:

```
SELECT * FROM SYSCAT.DBAUTH
   WHERE GRANTEE = USER AND GRANTEETYPE = 'U'
```

如下语句检索由一个特定用户直接授予的表特权的列表:

```
SELECT * FROM SYSCAT.TABAUTH
   WHERE GRANTOR  = USER
```

如下语句检索由一个特定用户直接授予的各个列特权的列表:

```
SELECT * FROM SYSCAT.COLAUTH
   WHERE GRANTOR  = USER
```

这些语句中的关键字 USER 始终等于一个用户的授权名的值。USER 是一个只读专用寄存器。

相关概念:

- 第 196 页的『特权、权限级别和数据库权限』
- 第 204 页的『数据库权限』
- 第 221 页的『将系统目录用于安全性说明』

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

保护系统目录视图

过程:

在创建数据库期间，会将系统目录视图的 SELECT 特权授予 PUBLIC。大多数情况下，这样做不会引起任何安全性问题。但是，对于特别敏感的数据，这可能不恰当，因为这些表描述数据库中的每个对象。若是这种情况，要考虑从 PUBLIC 撤销 SELECT 特权；然后按需要将 SELECT 特权授予特定用户。授予和撤销对系统目录视图的 SELECT 与对任何其它视图授予和撤销权限的方式相同，但是必须具有 SYSADM 或 DBADM 权限，才可执行此操作。

至少，应当考虑限制存取下列目录视图:

- SYSCAT.DBAUTH
- SYSCAT.TABAUTH
- SYSCAT.PACKAGEAUTH
- SYSCAT.INDEXAUTH
- SYSCAT.COLAUTH
- SYSCAT.PASSTHRUAUTH
- SYSCAT.SCHEMAAUTH

这将防止有关用户特权的信息对可存取该数据库的任何人可用。若具有此信息，不道德的用户可以不经授权而存取该数据库。

还应检查对其收集统计信息的列。记录在系统目录中的某些统计信息包含可能是您环境中的敏感信息的数据值。若这些统计信息包含敏感数据，可能希望从 PUBLIC 撤销对 SYSCAT.COLUMNS 和 SYSCAT.COLDIST 目录视图的 SELECT 特权。

若希望限制对系统目录视图的存取，您可以定义视图，来让每个授权名检索有关它自己特权的信息。

例如，如下视图 MYSELECTS 包括每个特定的表的所有者和名称，已将该表的 SELECT 特权直接授予了一个用户的授权名:

```
CREATE VIEW MYSELECTS AS
      SELECT TABSCHEMA, TABNAME FROM SYSCAT.TABAUTH
      WHERE GRANTEETYPE = 'U'
        AND GRANTEE = USER
        AND SELECTAUTH = 'Y'
```

在此语句中的关键字 USER 始终等于该授权名的值。

如下语句使此视图可供每个授权名使用:

```
GRANT SELECT ON TABLE MYSELECTS TO PUBLIC
```

最后，应记住要撤销对基本表的 SELECT 特权:

```
REVOKE SELECT ON TABLE SYSCAT.TABAUTH FROM PUBLIC
```

相关概念:

- 『目录统计信息』（*《管理指南: 性能》*）
- 第 204 页的『数据库权限』
- 第 221 页的『将系统目录用于安全性说明』

相关任务:

- 第 212 页的『授权特权』
- 第 213 页的『撤销特权』

防火墙支持简介

*防火墙*是一组相关的程序，位于网络网关服务器上，用来防止对系统或网关进行未授权的存取。

有四种类型的防火墙:

1. 网络级别、包过滤器或屏蔽路由器防火墙
2. 典型应用程序级别代理防火墙
3. 电路级别或透明代理防火墙
4. 有状态多层检查（SMLI）防火墙

存在合并以上列示的其中一种类型的防火墙的现有防火墙产品。存在许多合并以上列示类型组合的其它防火墙产品。

相关概念:

- 第 226 页的『屏蔽路由器防火墙』
- 第 226 页的『应用程序代理防火墙』
- 第 226 页的『电路级别防火墙』
- 第 227 页的『有状态的多层检查（SMLI）防火墙』

屏蔽路由器防火墙

此类型的防火墙也称为网络级别或信息包过滤器防火墙。这种防火墙的工作方式是根据协议属性屏蔽入局信息包。屏蔽的协议属性可能包括源或目标地址、协议类型、源或目标端口或某些其它特定于协议的属性。

对于所有防火墙解决方案（SOCKS 除外），您需要确保 DB2® 通用数据库（DB2 UDB）使用的所有端口对入局和出局信息包开放。DB2 UDB 将端口 523 用于 DB2 管理服务器（DAS），此 DAS 由 DB2 UDB 工具使用。通过使用 services 文件来将服务器数据库管理器配置文件中的服务名称映射至其端口号来确定所有服务器实例使用的端口。

相关概念:

- 第 225 页的『防火墙支持简介』

应用程序代理防火墙

代理或代理服务器是一种技术，它充当 Web 客户机与 Web 服务器之间的媒介。代理防火墙充当网关，用于接收来自客户机的请求。防火墙接收到客户机请求时，由代理软件确定最终服务器目标地址。应用程序代理代表客户机转换地址、执行附加存取控制检查、登录（如果需要的话）并连接至服务器。

防火墙机器上的 DB2® Connect 产品可充当目标服务器的代理。另外，在防火墙上充当最终目标服务器的中继段服务器的 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）服务器充当应用程序代理的角色。

相关概念:

- 第 225 页的『防火墙支持简介』

电路级别防火墙

此类型的防火墙又称为透明代理防火墙。透明代理防火墙不会修改代理认证和标识所需的请求或响应之外的请求或响应。透明代理防火墙的一个示例是 SOCKS。

DB2® 通用数据库（DB2 UDB）支持 SOCKS V4。

相关概念:

- 第 225 页的『防火墙支持简介』

有状态的多层检查（SMLI）防火墙

此类型的防火墙是检查所有七层“开放式系统互连”（OSI）模型的信息包过滤的完善形式。检查每一个信息包并与友好的信息包的已知状态进行比较。屏蔽路由器防火墙只检查信息包头，而 SMLI 防火墙会检查整个信息包，包括数据。

相关概念:

- 第 225 页的『防火墙支持简介』

第 8 章 审计 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）活动

DB2 通用数据库（DB2 UDB）审计设施简介

认证、权限和特权可以用于控制对数据的已知存取或预期存取，但是这些方法可能不足以防止对数据的未知存取或未预期存取。为帮助检测后一种类型的数据存取，DB2® 通用数据库（DB2 UDB）提供了一个审计设施。成功监视不需要的数据存取和后续分析，可改善对数据存取的控制，并最终防止对数据的恶意存取或粗心的未经授权的存取。监视应用程序和单独的用户存取（包括系统管理操作）可提供有关数据库系统活动的历史记录。

DB2 UDB 审计设施为一系列预定义数据库事件生成审计跟踪，并允许您维护它。将此设施生成的记录保存在审计日志文件中。对这些记录的分析可揭示标识系统误用的使用模式。一旦标识出该模式，则可执行操作来减少或消除这类系统误用。

审计设施在实例级别运行，记录所有实例级别活动和数据库级活动。

当在分区数据库环境中工作时，许多可审计的事件将在与用户连接的分区（协调程序节点）或目录节点（若它们不是相同的分区）中发生。这意味着审计记录可由多分区生成。每个审计记录的一部分包含有关协调程序节点和始发节点标识的信息。

审计日志（db2audit.log）和审计配置文件（db2audit.cfg）位于该实例的 `security` 子目录中。在您创建一个实例时，操作系统会尽可能设置对这些文件的读 / 写许可权。缺省情况下，许可权仅是该实例所有者的读 / 写权限。建议您不要更改这些许可权。

审计设施管理员工具 db2audit 的用户必须具有 SYSADM 权限。

必须明确停止和启动审计设施。当启动时，审计设施使用现有的审计配置信息。因为审计设施与 DB2 UDB 服务器无关，所以即使停止该实例，审计设施将仍然是活动的。事实上，当停止该实例时，可在审计日志中生成审计记录。

审计设施的授权用户可控制审计设施内的下列操作:

- 开始记录 DB2 UDB 实例内可审计的事件。
- 停止记录 DB2 UDB 实例内可审计的事件。
- 配置审计设施的行为，包括选择要记录的可审计事件的类别。
- 请求当前审计配置的描述。
- 从该实例中清除任何暂挂的审计记录，并将它们写入审计日志中。
- 格式化审计日志中的审计记录，并将它们复制到一个平面文件或 ASCII 定界的文件，来抽取这些审计记录。因为下列两个原因中的一个原因来执行抽取: 准备分析日志记录，或者准备修剪日志记录。
- 修剪当前审计日志中的审计记录。

注: 使用审计实用程序之前，通过发出 db2audit start 命令确保打开了审计设施。

可生成不同类别的审计记录。在可用于审计的事件类别的描述中（下面），您应注意每个类别的名称后面是一个单词的关键字，它用来标识该类别的类型。可用于审计的事件类别是:

- 审计（AUDIT）。当更改审计设置或存取审计日志时会生成记录。
- 权限检查（CHECKING）。对存取或操纵 DB2 UDB 对象或函数的尝试进行权限检查期间会生成记录。
- 对象维护（OBJMAINT）。当创建或删除数据对象时会生成记录。
- 安全维护（SECMAINT）。当授予或撤销对象或数据库特权或 DBADM 权限时会生成记录。当修改数据库管理器安全性配置参数 SYSADM_GROUP、SYSCTRL_GROUP 或 SYSMAINT_GROUP 时也会生成记录。
- 系统管理（SYSADMIN）。当执行需要 SYSADM、SYSMAINT 或 SYSCTRL 权限的操作时会生成记录。
- 用户验证（VALIDATE）。当认证用户或检索系统安全性信息时会生成记录。
- 操作上下文（CONTEXT）。当执行数据库操作时，生成记录以显示该操作上下文。此类别允许对审计日志文件进行更好的解释。当与该日志的事件相关因子字段一起使用时，可将一组事件重新与单个数据库操作关联。例如，动态 SQL 的 SQL 语句、静态 SQL的程序包标识或可执行的操作类型的指示符（如 CONNECT）均可提供分析审计结果时所需的上下文。

  注: 提供该操作上下文的 SQL 语句可能很长，并可在 CONTEXT 记录内完全显示。这可能使 CONTEXT 记录变得很大。
- 您可以审计失败的操作和／或成功的操作。

对该数据库执行的任何操作可能生成几个记录。生成的和移至审计日志的实际记录数目取决于审计设施配置所指定的要记录的事件类别数。它还取决于是审计成功的操作，还是失败的操作，或二者兼有。由于此原因，对要审计的事件的选择十分重要。

相关概念:

- 第 230 页的『审计设施行为』
- 第 242 页的『审计设施记录布局（简介）』
- 第 258 页的『审计设施技巧和方法』

相关任务:

- 第 259 页的『控制 DB2 UDB 审计设施活动』

相关参考:

- 第 232 页的『审计设施的使用』
- 第 242 页的『审计设施消息』

审计设施行为

审计设施记录可审计的事件，包括那些影响的数据库实例。因此，审计设施是 DB2® 通用数据库（DB2 UDB）中的一个独立部件，即使停止 DB2 UDB 实例，它也可以运行。若审计设施是活动的，当启动一个已停止的实例时，对该实例中的数据库事件的审计将继续。

将审计记录写入审计日志的时间安排可对该实例中的数据库的性能有显著影响。写入审计记录可与导致生成那些记录的事件同步或异步发生。*audit_buf_sz* 数据库管理器配置参数的值确定何时写入审计记录。

若此参数值为零（0），则同步写入。生成审计记录的事件将等到将该记录写入磁盘为止。与每个记录相关的等待导致 DB2 UDB 性能降低。

若 *audit_buf_sz* 的值大于零，则异步写入该记录。*audit_buf_sz* 的值大于零时，该值是 4 KB 页的一个倍数，用来创建内部缓冲区。该内部缓冲区用来在将一组审计记录写入磁盘之前保存大量审计记录。作为审计事件的结果生成审计记录的语句将不会等到将该记录写入磁盘，它可继续其操作。

在异步情况下，审计记录可能可以在未填写的缓冲区中保留一段时间。要防止这种情况的持续时间过长，数据库管理器将强制定期写入审计记录。审计设施的授权用户也可用显式请求清除审计缓冲区。

记录是同步写入还是异步写入，使发生错误时情况有些不同。在异步方式中，可能会有某些记录丢失，因为审计记录是在写入磁盘之前缓冲存储的。在同步方式中，可能有一个记录丢失，因为错误只能阻止最多一个审计记录写入。

ERRORTYPE 审计设施参数的设置控制如何在 DB2 UDB 和审计设施之间管理错误。当审计设施是活动的，并且 ERRORTYPE 审计设施参数的设置是 AUDIT 时，将审计设施作为 DB2 UDB 的任何其它部件一样对待。对于与被视为成功的语句相关的一个审计事件，必须写入审计记录（在同步方式下，写至磁盘；在异步方式下，写至审计缓冲区）。当在此方式下运行时，不管何时遇到错误，都将一个负的 SQLCODE 返回至生成审计记录的语句的应用程序。若将错误类型设置为 NORMAL，则忽略来自 db2audit 的任何错误，并返回该操作的 SQLCODE。

根据该 API 或 SQL 语句以及 DB2 UDB 实例的审计设置，可为特定事件生成一个或几个审计记录，或不生成审计记录。例如，具有一个 SELECT 子查询的 SQL UPDATE 语句可生成两个审计记录，一个记录包含对一个表的 UPDATE 特权的权限检查结果，另一个记录包含对一个表的 SELECT 特权的权限检查结果。

对于动态数据处理语言（DML）语句，在预编译该语句时会对所有权限检查生成审计记录。不会再次审计同一用户对那些语句的重新使用，因为那时不进行权限检查。但是，若已更改其中一个包含特权信息的目录表，则在下一个工作单元中，再次检查高速缓存的动态 SQL 语句的语句特权，并创建一个或多个新的审计记录。

对于仅包含静态 DML 语句的程序包，可生成审计记录的唯一可审计的事件是权限检查，它查看用户是否具有执行该程序包的特权。在预编译或绑定该程序包时，执行该程序包中静态 SQL 语句所需的权限检查和可能的审计记录创建。程序包内的静态 SQL 语句的执行是不可审计的。当用户显式地再次绑定程序包时，或系统隐式再次绑定程序包时，则为该静态 SQL 语句所需的权限检查生成审计记录。

对于在执行语句时执行权限检查的语句（例如，数据定义语言（DDL）、GRANT 和 REVOKE 语句），不管何时使用这些语句，都将生成审计记录。

注: 当执行 DDL 时，无论该语句的实际节号可能是什么，在审计记录中为所有事件（除上下文事件外）记录的节号都将为零（0）。

相关概念:

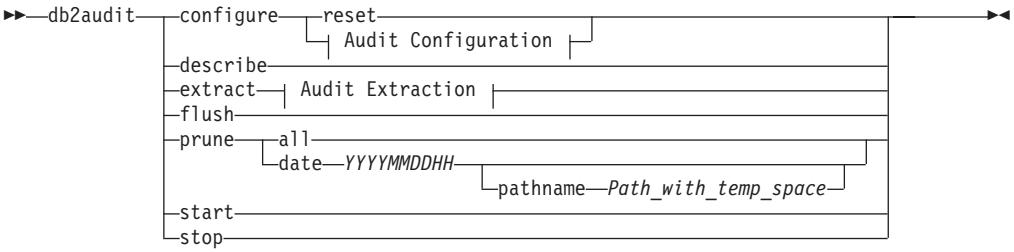
- 第 229 页的『DB2 通用数据库（DB2 UDB）审计设施简介』

相关参考:

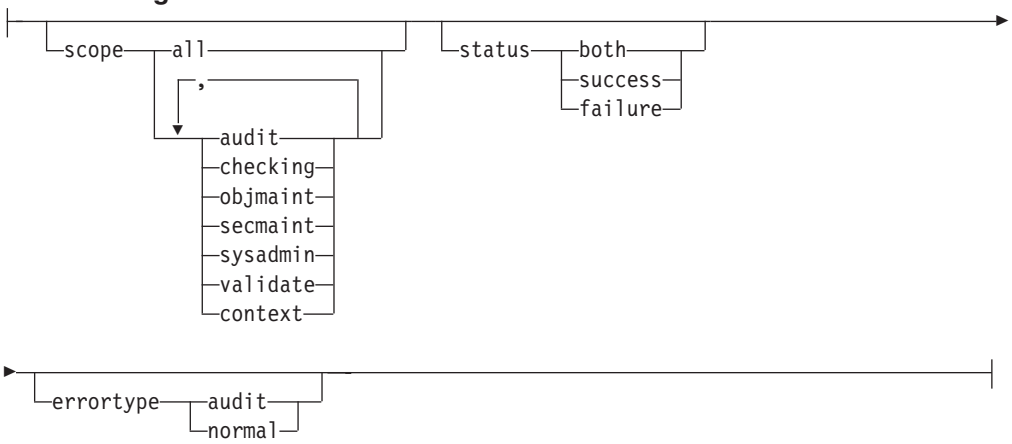
- 『audit_buf_sz - 审计缓冲区大小配置参数』（《*管理指南: 性能*》）
- 第 232 页的『审计设施的使用』

审计设施的使用

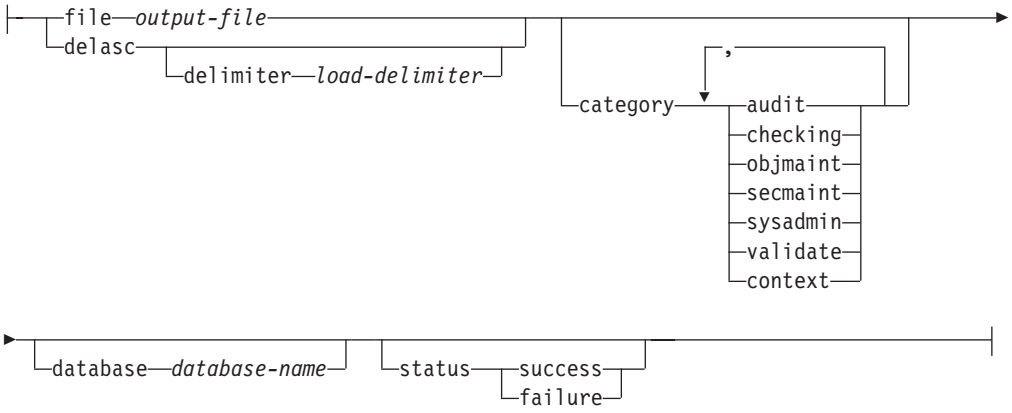
查看下列语法图中的每一部分将会帮助您理解可以如何使用审计设施。



Audit Configuration:



Audit Extraction:



以下是每个参数的描述和隐含用法:

configure

此参数允许修改该实例的 security 子目录中的 db2audit.cfg 配置文件。即使

在关闭该实例时也可更新此文件。当实例在活动时发生的更新会动态影响所有分区上 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）正在执行的审计。若已启动审计设施，并且正在审计可审计事件的*审计*类别，则配置文件上发生的配置操作会导致创建一条审计记录。

以下是可能对配置文件执行的操作:

- RESET。此操作导致配置文件还原为初始配置（其中 SCOPE 是除 CONTEXT 外的所有类别，STATUS 为 FAILURE，ERRORTYPE 为 NORMAL，而审计设施为 OFF）。若原始文件已丢失或损坏，此操作将创建一个新的审计配置文件。
- SCOPE。此操作指定要审计哪一类或哪几类的事件。此操作也允许集中审计并减少日志的增长。建议尽可能地限制所记录的事件的数量和类型，否则审计日志将迅速增长。

  注: 请注意，缺省 SCOPE 是除 CONTEXT 外的所有类别，并且它可能导致迅速生成记录。与同步或异步方式一起使用，类别的选择可能导致性能明显降低以及磁盘需求的明显增加。
- STATUS。此操作指定是记录成功的事件，还是记录失败的事件，还是既记录成功事件又记录失败事件。

  注: 上下文事件发生在操作的状态已知之前。因此，无论与此参数相关的值如何，都记录这类事件。
- ERRORTYPE。此操作指定是将审计错误返回给用户还是忽略它。此参数的值可以是:
  - AUDIT。所有错误，包括在审计设施中发生的错误，均由 DB2 UDB 管理，并将所有负面 SQLCODE 报告回调用程序。
  - NORMAL。忽略 db2audit 生成的任何错误，并只将与执行的操作相关的错误的 SQLCODE 返回至该应用程序。

describe

此参数将当前审计配置信息和状态显示至标准输出。

extract

此参数允许将审计记录从审计日志移动至指示的目的地。如果未指定可选子句，则抽取所有审计记录并将其放在平面报告文件中。若 *output_file* 已存在，则返回一条错误消息。

以下是在抽取时可使用的可能选项:

- FILE。将抽取的审计记录放在文件（*output_file*）中。如果未指定文件名，将记录写入 `sqllib` 的 `security` 子目录中的 `db2audit.out` 文件。如果未指定目录，*output_file* 将被写入当前工作目录。
- DELASC。使用定界的 ASCII 格式来放置抽取的审计记录，以便适合装入 DB2 UDB 关系表中。将输出放在单独的文件中: 每种类别一个文件。这些文件名是:
  - `audit.del`
  - `checking.del`
  - `objmaint.del`
  - `secmaint.del`
  - `sysadmin.del`

– validate.del
– context.del

总是将这些文件写入 sqllib 的 security 子目录。

当从审计日志中抽取时，DELASC 选项也允许您覆盖缺省审计字符串定界符（″0xff″）。将使用 DELASC DELIMITER，后接您希望使用的新定界符，以便准备装入将容纳审计记录的表中。新装入定界符可以是单个字符（如 !）或代表十六进制数的四字节字符串（如 0xff）。

- CATEGORY。将抽取指定类别的审计事件的审计记录。若未指定，则所有类别都适合抽取。
- DATABASE。将抽取指定数据库的审计记录。若未指定，则所有数据库都适合抽取。
- STATUS。将抽取指定状态的审计记录。若未指定，则所有记录都适合抽取。

flush 此参数强制将任何暂挂的审计记录写入审计日志。若审计设施处于错误状态，则也将引擎中的审计状态从“不能记录”复位为“准备记录”。

prune 此参数允许从审计日志中删除审计记录。若审计设施是活动的并已为审计指定了事件的“审计”类别，则在修剪审计日志之后将记录一条审计记录。

以下是在修剪时可使用的可能选项:

- ALL。将删除审计日志中的所有审计记录。
- DATE yyyymmddhh。用户可指定从审计日志中删除在指定的日期／时间当时或之前发生的所有审计记录。作为选项，用户可提供一个

  pathname

  审计设施在修剪审计日志时将该路径名作为一个临时空间。当日志文件驻留的磁盘已满，并且没有足够的磁盘空间允许修剪操作时，此临时空间允许修剪审计日志。

start 此参数导致审计设施根据 db2audit.cfg 文件的内容开始审计事件。在分区的 DB2 UDB 实例中，当指定此子句时，将在所有分区上开始审计。若已为审计指定了事件的“审计”类别，则当启动审计设施时，将记录一条审计记录。

stop 此参数导致审计设施停止审计事件。在分区的 DB2 UDB 实例中，当指定此子句时，将在所有分区上停止审计。若已为审计指定了事件的“审计”类别，则当停止审计设施时，将记录一条审计记录。

相关概念:

- 第 229 页的『DB2 通用数据库（DB2 UDB）审计设施简介』
- 第 258 页的『审计设施技巧和方法』

相关参考:

- 『db2audit - Audit Facility Administrator Tool Command』（*Command Reference*）

使用 DB2 表中的 DB2 审计数据

以下主题描述如何创建 DB2 审计数据、如何创建表来容纳此数据、如何以 DB2 审计数据填充表和如何从表中选择 DB2 审计数据。

使用 DB2 表中的 DB2 审计数据

缺省情况下，使用 DB2 审计设施维护数据库活动的审计跟踪时，审计设施将审计记录放入日志文件中。如果愿意，可以将审计记录从日志文件写入文本文件，也可以将审计记录从日志文件写入定界 ASCII 文件，然后将 ASCII 文件的内容装入 DB2 表。审计数据位于 DB2 表中时，可以从表中选择数据来回答有关 DB2 实例活动的问题。

过程:

要使用 DB2 表中的审计数据:

1. 创建表来容纳 DB2 审计数据。
2. 创建 DB2 审计数据文件。
3. 使用装入实用程序将数据填充到表中。
4. 选择表数据。

相关概念:

- 第 230 页的『审计设施行为』
- 第 258 页的『审计设施技巧和方法』

相关任务:

- 第 238 页的『创建 DB2 审计数据文件』
- 第 235 页的『创建表来容纳 DB2 审计数据』
- 第 239 页的『将 DB2 审计数据装入表中』
- 第 241 页的『从表中选择 DB2 审计数据』

相关参考:

- 第 232 页的『审计设施的使用』

创建表来容纳 DB2 审计数据

使用表中的审计数据之前，需要创建表来容纳数据。应考虑在单独的模式中创建这些表，以将表中的数据与未授权用户相隔离。

先决条件:

- 有关创建模式所需的权限和特权，请参阅 CREATE SCHEMA 语句。
- 有关创建表所需的权限和特权，请参阅 CREATE TABLE 语句。
- 确定想要用来容纳表的表空间。（本主题未描述如何创建表空间。）

过程:

以下实例说明如何创建表来容纳所有 ASCII 文件的全部记录。如果愿意，可以创建单独模式来包含这些表。

如果不想使用文件中包含的所有数据，则可以忽略表定义中的列或绕过创建表（如果需要）。如果忽略表定义的列，则必须修改将数据装入这些表所用的命令。

1. 发出 **db2** 命令以打开 DB2 命令窗口。
2. 可选。创建模式来容纳表。发出下列命令。对于此示例，模式名为 AUDIT

   ```
   CREATE SCHEMA AUDIT
   ```

3. 可选。如果创建 AUDIT 模式，则在创建任何表之前切换至该模式。发出下列命令:

```
SET CURRENT SCHEMA = 'AUDIT'
```

4. 要创建包含 audit.del 文件的记录的表，请发出下列 SQL 语句:

```
CREATE TABLE AUDIT (TIMESTAMP CHAR(26),
                    CATEGORY CHAR(8),
                    EVENT VARCHAR(32),
                    CORRELATOR INTEGER,
                    STATUS INTEGER,
                    USERID VARCHAR(1024),
                    AUTHID VARCHAR(128))
```

5. 要创建将包含 checking.del 文件的记录的表，请发出下列 SQL 语句:

```
CREATE TABLE CHECKING (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       ACCESSAPP CHAR(18),
                       ACCESSATT CHAR(18),
                       PKGVER VARCHAR(64))
```

6. 要创建将包含 objmaint.del 文件的记录的表，请发出下列 SQL 语句:

```
CREATE TABLE OBJMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       OBJSCHEMA VARCHAR(128),
                       OBJNAME VARCHAR(128),
                       OBJTYPE VARCHAR(32),
                       PACKVER VARCHAR(64))
```

7. 要创建将包含 secmaint.del 文件的记录的表，请发出下列 SQL 语句:

```
CREATE TABLE SECMAINT (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
```

```
                                    APPID VARCHAR(255),
                                    APPNAME VARCHAR(1024),
                                    PKGSCHEMA VARCHAR(128),
                                    PKGNAME VARCHAR(128),
                                    PKGSECNUM SMALLINT,
                                    OBJSCHEMA VARCHAR(128),
                                    OBJNAME VARCHAR(128),
                                    OBJTYPE VARCHAR(32),
                                    GRANTOR VARCHAR(128),
                                    GRANTEE VARCHAR(128),
                                    GRANTEETYPE VARCHAR(32),
                                    PRIVAUTH CHAR(18),
                                    PKGVER VARCHAR(64))
```

8. 要创建将包含 sysadmin.del 文件的记录的表，请发出下列 SQL 语句:

```
CREATE TABLE SYSADMIN (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64))
```

9. 要创建将包含 validate.del 文件的记录的表，请发出下列 SQL语句:

```
CREATE TABLE VALIDATE (TIMESTAMP CHAR(26),
                       CATEGORY CHAR(8),
                       EVENT VARCHAR(32),
                       CORRELATOR INTEGER,
                       STATUS INTEGER,
                       DATABASE CHAR(8),
                       USERID VARCHAR(1024),
                       AUTHID VARCHAR(128),
                       EXECID VARCHAR(1024),
                       NODENUM SMALLINT,
                       COORDNUM SMALLINT,
                       APPID VARCHAR(255),
                       APPNAME VARCHAR(1024),
                       AUTHTYPE VARCHAR(32),
                       PKGSCHEMA VARCHAR(128),
                       PKGNAME VARCHAR(128),
                       PKGSECNUM SMALLINT,
                       PKGVER VARCHAR(64)
                       PLUGINNAME VARCHAR(32))
```

10. 要创建将包含 context.del 文件的记录的表，请发出下列 SQL 语句:

```
CREATE TABLE CONTEXT (TIMESTAMP CHAR(26),
                      CATEGORY CHAR(8),
                      EVENT VARCHAR(32),
                      CORRELATOR INTEGER,
                      DATABASE CHAR(8),
                      USERID VARCHAR(1024),
                      AUTHID VARCHAR(128),
                      NODENUM SMALLINT,
                      COORDNUM SMALLINT,
                      APPID VARCHAR(255),
                      APPNAME VARCHAR(1024),
                      PKGSCHEMA VARCHAR(128),
```

```
PKGNAME VARCHAR(128),
PKGSECNUM SMALLING,
STMTTEXT CLOB(2M),
PKGVER VARCHAR(64))
```

11. 创建表之后，发出 COMMIT 语句以确保将该表定义写入磁盘。
12. 创建表之后，准备将审计记录从 db2audit.log 文件抽取到定界 ASCII 文件。

相关任务:

- 第 76 页的『设置模式』

相关参考:

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

创建 DB2 审计数据文件

缺省情况下，DB2 审计设施将审计数据写入 db2audit.log 文件中。此文件中的记录不能装入表中。必须将可用来填充表的审计记录抽取到定界 ASCII 文件。

先决条件:

要使用 **db2audit** 命令，必须具有 SYSADM 权限。

过程:

要将审计设施记录写入定界 ASCII 文件:

1. 查看有关『审计设施用途』的主题来确定想要审计的 DB2 活动的类型: 当对审计设施设置的配置感到满意时，发出以下命令开始审计:

   ```
   db2audit start
   ```

2. 发出以下命令以确保所有审计记录均从内存刷新至 db2audit.log 文件:

   ```
   db2audit flush
   ```

3. 发出以下命令将审计记录从 db2audit.log 移至定界 ASCII 文件:

   ```
   db2audit extract delasc
   ```

   以下文件在 sqllib 的 security 子目录中创建。如果不是审计特定类型的事件，则创建此事件的文件，但文件为空。
   - audit.del
   - checking.del
   - objmaint.del
   - secmaint.del
   - sysadmin.del
   - validate.del
   - context.del

4. 发出以下命令来从刚刚抽取的 db2audit.log 文件中删除审计记录:

   ```
   db2audit prune date YYYYMMDDHH
   ```

   其中 *YYYYMMDDHH* 为当前年、月、日和小时。由于下一步以审计记录填充表时需要此信息，因此记下所使用的值。

审计设施将继续向 db2audit.log 文件写入新的审计记录，这些记录将具有迟于 *YYYYMMDDHH* 的时间戳记。从 db2audit.log 文件中删除已抽取的记录，避免再次抽取相同的记录。所有 *YYYYMMDDHH* 以后写入的审计记录将在下次抽取审计数据时写入 .del 文件中。

5. 创建审计数据文件后，下一步是使用装入实用程序将审计数据填充到表中。

相关参考:

- 『db2audit - Audit Facility Administrator Tool Command』（*Command Reference*）
- 第 232 页的『审计设施的使用』
- 第 243 页的『AUDIT 事件的审计记录布局』
- 第 244 页的『CHECKING 事件的审计记录布局』
- 第 249 页的『OBJMAINT 事件的审计记录布局』
- 第 250 页的『SECMAINT 事件的审计记录布局』
- 第 254 页的『SYSADMIN 事件的审计记录布局』
- 第 256 页的『VALIDATE 事件的审计记录布局』
- 第 257 页的『CONTEXT 事件的审计记录布局』

将 DB2 审计数据装入表中

在创建表来容纳审计数据后，将 ASCII 文件中的数据装入表中。

先决条件:

有关更多信息，请参阅使用装入实用程序所需的特权、权限和授权上的主题。

过程:

使用装入实用程序将数据装入表中。对每个表发出单独的装入命令。如果忽略表定义中的一个或多个列，则必须修改使用的 LOAD 命令版本才能成功装入数据。此外，如果抽取审计数据时指定缺省值（0xff）之外的定界符字符，则还必须修改所用的 LOAD 命令的版本（有关更多信息，请参阅用于装入的文件类型修改器主题）。

1. 发出 **db2** 命令以打开 DB2 命令窗口。
2. 要装入 AUDIT 表，请发出下列命令:

```
LOAD FROM audit.del OF del MODIFIED BY CHARDEL0xff INSERT INTO schema.AUDIT
```

注: 指定文件名时，请使用全限定路径名。例如，如果将 DB2 UDB 安装在基于 Windows 计算机的 C: 驱动器上，则应指定 C:\Program Files\IBM\SQLLIB*instance*\security\audit.del 作为 audit.del 文件的全限定文件名。

装入 AUDIT 表之后，发出下列 DELETE 语句以确保下次装入时不会将重复的行装入表中。从 db2audit.log 文件中抽取审计记录时，文件中的所有记录均写入 .del 文件中。.del 文件可能包含后来修剪审计日志之后写入的记录（原因是 **db2audit prune** 命令仅修剪指定时间之前的记录）。下次抽取审计记录时，新的 .del 文件将包含以前抽取的记录，但 **db2auditprune** 命令不会将其删除（原因是其在指定修剪操作时间后写入）。从表中删除 db2audit.log 文件修剪之前的行，以确保表中不包含重复的行，并且未丢失任何审计记录。

```
DELETE FROM schema.AUDIT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

其中 *YYYYMMDDHH* 为修剪 db2audit.log 文件时指定的值。由于 DB2 审计设施在修剪后继续向 db2audit.log 文件写入审计记录，所以必须对分钟和秒钟指定 0000 以确保 db2audit.log 文件修剪后写入的审计记录不会从表中删除。

3. 要装入 CHECKING 表，请发出下列命令:

```
LOAD FROM checking.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
   schema.CHECKING
```

装入 CHECKING 表之后，发出下列 SQL 语句以确保下次装入时不会将重复的行装入表中:

```
DELETE FROM schema.CHECKING WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

其中 *YYYYMMDDHH* 为修剪日志文件时指定的值。

4. 要装入 OBJMAINT 表，请发出下列命令:

```
LOAD FROM objmaint.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
   schema.OBJMAINT
```

装入 OBJMAINT 表之后，发出下列 SQL 语句以确保下次装入时不会将重复的行装入表中:

```
DELETE FROM schema.OBJMAINT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

其中 *YYYYMMDDHH* 为修剪日志文件时指定的值。

5. 要装入 SECMAINT 表，请发出下列命令:

```
LOAD FROM secmaint.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
   schema.SECMAINT
```

装入 SECMAINT 表之后，发出下列 SQL 语句以确保下次装入时不会将重复的行装入表中:

```
DELETE FROM schema.SECMAINT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

其中 *YYYYMMDDHH* 为修剪日志文件时指定的值。

6. 要装入 SYSADMIN 表，请发出下列命令:

```
LOAD FROM sysadmin.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
   schema.SYSADMIN
```

装入 SYSADMIN 表之后，发出下列 SQL 语句以确保下次装入时不会将重复的行装入表中:

```
DELETE FROM schema.SYSADMIN WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

其中 *YYYYMMDDHH* 为修剪日志文件时指定的值。

7. 要装入 VALIDATE 表，请发出下列命令:

```
LOAD FROM validate.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
   schema.VALIDATE
```

装入 VALIDATE 表之后，发出下列 SQL 语句以确保下次装入时不会将重复的行装入表中:

```
DELETE FROM schema.VALIDATE WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

其中 *YYYYMMDDHH* 为修剪日志文件时指定的值。

8. 要装入 CONTEXT 表，请发出下列命令:

```
LOAD FROM context.del OF del MODIFIED BY CHARDEL0xff INSERT INTO
   schema.CONTEXT
```

装入 CONTEXT 表之后，发出下列 SQL 语句以确保下次装入时不会将重复的行装入表中:

```
DELETE FROM schema.CONTEXT WHERE TIMESTAMP > TIMESTAMP('YYYYMMDDHH0000')
```

其中 *YYYYMMDDHH* 为修剪日志文件时指定的值。

9. 在完成将数据装入表之后，从 sqllib 目录的 security 子目录删除 .del 文件。
10. 将审计数据装入表之后，您已准备就绪可以从这些表中选择数据。

如果初次填充表之后要再次执行此操作，则使用 INSERT 选项将新的表数据添加至现有表数据。如果要从表中除去以前 **db2audit extract** 操作的记录，则使用 REPLACE 选项再次装入表。在任一情况下，记得将记录抽取到 del 文件之前将审计记录刷新到 db2audit.log 文件，同时在抽取记录后修剪 db2audit.log 文件，这样就不会将相同的记录多次装入表中。

相关任务:

• 第 235 页的『使用 DB2 表中的 DB2 审计数据』

从表中选择 DB2 审计数据

将审计数据成功装入表中后，可从这些表中选择数据以进一步分析。

先决条件:

有关从表中选择数据所需的权限和特权的信息，请参阅有关 SELECT 语句的主题。

过程:

要选择表中的所有行:

1. 发出 **db2** 命令以打开 DB2 命令窗口。
2. 对要从中选择审计数据的每个表发出下列格式的 SQL 语句:

```
SELECT * FROM schema.table
```

例如，要从 AUDIT 模式的 CHECKING 表中选择全部数据，请使用下列语句:

```
SELECT * FROM AUDIT.CHECKING
```

您所执行的选择应反映要对数据执行的分析的类型。例如，您可根据授权标识（authid）选择记录来确定对此授权标识执行的活动的类型:

```
SELECT * FROM AUDIT.CHECKING WHERE AUTHID = authorization ID
```

其中 *authorization ID* 是要为其分析数据的用户标识。

有关审计数据中可以包括的值的描述，请参阅表的相应审计记录布局主题和表可能返回的值列表。

相关参考:

- 『Subselect』（*SQL Reference, Volume 1*）
- 『SELECT statement』（*SQL Reference, Volume 2*）

- 第 243 页的『AUDIT 事件的审计记录布局』
- 第 244 页的『CHECKING 事件的审计记录布局』
- 第 246 页的『可能的 CHECKING 存取批准原因的列表』
- 第 247 页的『可能的 CHECKING 存取尝试类型的列表』
- 第 249 页的『OBJMAINT 事件的审计记录布局』
- 第 250 页的『SECMAINT 事件的审计记录布局』
- 第 251 页的『可能的 SECMAINT 特权或权限的列表』
- 第 254 页的『SYSADMIN 事件的审计记录布局』
- 第 254 页的『可能的 SYSADMIN 审计事件的列表』
- 第 256 页的『VALIDATE 事件的审计记录布局』
- 第 257 页的『CONTEXT 事件的审计记录布局』
- 第 257 页的『可能的 CONTEXT 审计事件的列表』

审计设施消息

SQL1322N 当写入审计日志文件时出错。

说明: 当调用 DB2 Universal Database™（DB2 通用数据库）审计设施以将审计事件记录到审计日志文件中时，该设施遇到错误。审计日志驻留的文件系统上没有空间。

用户响应: 系统管理员应在此文件系统上释放空间，或修剪审计日志以减小其大小。

当有更多空间可用时，使用 db2audit 来删除内存中的任何数据，并将该审计程序复位为就绪状态。确保在修剪该日志前，进行适当的抽取，或建立该日志的副本，因为删除的记录是不可恢复的。

sqlcode: -1322

sqlstate: 50830

SQL1323N 存取审计配置文件时出错。

说明: 审计配置文件（db2audit.cfg）不能打开或者无效。此错误的可能原因是 db2audit.cfg 文件不存在或已损坏。

用户响应: 执行下列其中一项操作:

- 从该文件的保存版本复原。
- 通过发出以下命令复位审计设施配置文件:

```
db2audit reset
```

sqlcode: -1323

sqlstate: 57019

相关概念:

- 第 229 页的『DB2 通用数据库（DB2 UDB）审计设施简介』

审计设施记录布局（简介）

当使用 DELASC 抽取选项从审计日志中抽取审计记录时，每个记录将具有下列表中显示的其中一个格式。每个表将从显示样本记录的内容开始。该记录中每一项的描述显示在相关的表中，一次显示一行。若该项很重要，将突出显示（**黑体**）该项的名称。这些项包含您最感兴趣的信息。

注:

1. 并非样本记录中的所有字段都有值。
2. 某些字段（如“尝试的存取”）以定界的 ASCII 格式存储为位图。但是，在此平面报告文件中，这些字段将显示为一组字符串，表示位图值。

3. 已将称为“程序包版本”的新字段添加至 CHECKING、OBJMAINT、SECMAINT、SYSADMIN、VALIDATE 和 CONTEXT 事件的记录布局。

相关参考:

- 第 243 页的『AUDIT 事件的审计记录布局』
- 第 244 页的『CHECKING 事件的审计记录布局』
- 第 249 页的『OBJMAINT 事件的审计记录布局』
- 第 250 页的『SECMAINT 事件的审计记录布局』
- 第 254 页的『SYSADMIN 事件的审计记录布局』
- 第 256 页的『VALIDATE 事件的审计记录布局』
- 第 257 页的『CONTEXT 事件的审计记录布局』

有关审计设施记录布局的详细信息

本节说明了各种审计设施记录布局。

AUDIT 事件的审计记录布局

表 7. AUDIT 事件的审计记录布局

timestamp=1998-06-24-11.54.05.151232;**category=AUDIT;audit event=START;**
event correlator=0;event status=0;
userid=boss;authid=BOSS;		

名称	**格式**	**描述**
Timestamp	CHAR(26)	审计事件的日期和时间。
Category	CHAR(8)	审计事件的类别。可能的值是: AUDIT
Audit Event	VARCHAR(32)	特定的审计事件。 可能的值包括: CONFIGURE、DB2AUD、EXTRACT、FLUSH、PRUNE、START、STOP 和 UPDATE_ADMIN_CFG
Event Correlator	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件相关。
Event Status	INTEGER	审计事件的状态，由 SQLCODE 表示，其中 成功的事件 > = 0 失败的事件 < 0
User ID	VARCHAR(1024)	审计事件发生时的用户标识。
Authorization ID	VARCHAR(128)	审计事件发生时的授权标识。

相关概念:

- 第 242 页的『审计设施记录布局（简介）』

CHECKING 事件的审计记录布局

表 8. CHECKING 事件的审计记录布局

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;audit event=CHECKING_OBJECT;
  event correlator=2;event status=0;
  database=FOO;userid=boss;authid=BOSS;
  application id=*LOCAL.newton.980624124210;application name=testapp;
  package schema=NULLID;package name=SYSSH200;
  package section=0;object schema=GSTAGER;object name=NONE;object type=REOPT_VALUES;
  access approval reason=DBADM;access attempted=STORE;
```

名称	格式	描述
Timestamp	CHAR(26)	审计事件的日期和时间。
Category	CHAR(8)	审计事件的类别。可能的值是: CHECKING
Audit Event	VARCHAR(32)	特定的审计事件。 可能的值包括: CHECKING_OBJECT 和 CHECKING_FUNCTION
Event Correlator	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件相关。
Event Status	INTEGER	审计事件的状态, 由 SQLCODE 表示, 其中 成功的事件 > = 0 失败的事件 < 0
Database Name	CHAR(8)	为之生成该事件的数据库的名称。若它是实例级别审计事件, 则为空白。
User ID	VARCHAR(1024)	审计事件发生时的用户标识。
Authorization ID	VARCHAR(128)	审计事件发生时的授权标识。
Origin Node Number	SMALLINT	审计事件发生时所在的节点号。
Coordinator Node Number	SMALLINT	协调程序节点的节点号。
Application ID	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
Application Name	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
Package Schema	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
Package Name	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。
Package Section Number	SMALLINT	审计事件发生时正在使用的程序包中的节号。
Object Schema	VARCHAR(128)	为之生成审计事件的对象的模式。
Object Name	VARCHAR(128)	为之生成审计事件的对象的名称。
Object Type	VARCHAR(32)	为之生成审计事件的对象的类型。可能的值包括: 显示在标题为『审计记录对象类型』的主题中的那些值。
Access Approval Reason	CHAR(18)	指示为此审计事件批准存取的原因。可能的值包括: 显示在标题为『可能的 CHECKING 存取批准原因的列表』的主题中的那些值。
Access Attempted	CHAR(18)	指定尝试的存取类型。可能的值包括: 显示在标题为『可能的 CHECKING 存取尝试类型的列表』的主题中的那些值。
Package Version	VARCHAR (64)	审计事件发生时正在使用的程序包的版本。

相关概念:

- 第 242 页的『审计设施记录布局（简介）』

相关参考:

- 第 246 页的『可能的 CHECKING 存取批准原因的列表』
- 第 247 页的『可能的 CHECKING 存取尝试类型的列表』
- 第 245 页的『审计记录对象类型』

审计记录对象类型

表 9. 基于审计事件的审计记录对象类型

对象类型	CHECKING 事件	OBJMAINT 事件	SECMAINT 事件
NONE	X	X	X
TABLE	X	X	X
VIEW	X	X	X
ALIAS	X	X	
FUNCTION	X	X	X
INDEX	X	X	X
INDEX EXTENSION		X	
PACKAGE	X	X	X
PACKAGE CACHE	X		
DATA_TYPE		X	
NODEGROUP	X	X	
SCHEMA	X	X	X
STORED_PROCEDURE	X	X	X
METHOD_BODY	X	X	X
BUFFERPOOL	X	X	
SEQUENCE	X	X	
TABLESPACE	X	X	X
EVENT_MONITOR	X	X	
TRIGGER		X	
DATABASE	X		X
INSTANCE	X		
FOREIGN_KEY		X	
PRIMARY_KEY		X	
UNIQUE_CONSTRAINT		X	
CHECK_CONSTRAINT		X	
WRAPPER	X	X	
SERVER	X	X	X
NICKNAME	X	X	X
USER MAPPING	X	X	
SERVER OPTION	X	X	
TYPE&TRANSFORM	X	X	
TYPE MAPPING	X	X	

表 9. 基于审计事件的审计记录对象类型 (续)

对象类型	CHECKING 事件	OBJMAINT 事件	SECMAINT 事件
FUNCTION MAPPING	X	X	
SUMMARY TABLES	X	X	X
JAR_FILE		X	
ALL	X		
REOPT_VALUES	X		

相关参考:

- 第 244 页的『CHECKING 事件的审计记录布局』
- 第 249 页的『OBJMAINT 事件的审计记录布局』
- 第 250 页的『SECMAINT 事件的审计记录布局』

可能的 CHECKING 存取批准原因的列表

以下是可能的 CHECKING 存取批准原因的列表:

0x0000000000000001 ACCESS DENIED
未批准存取; 确切地说, 拒绝了存取。

0x0000000000000002 SYSADM
批准存取; 该应用程序 / 用户具有 SYSADM 权限。

0x0000000000000004 SYSCTRL
批准存取; 该应用程序 / 用户具有 SYSCTRL 权限。

0x0000000000000008 SYSMAINT
批准存取; 该应用程序 / 用户具有 SYSMAINT 权限。

0x0000000000000010 DBADM
批准存取; 该应用程序 / 用户具有 DBADM 权限。

0x0000000000000020 DATABASE PRIVILEGE
批准存取; 该应用程序 / 用户具有使用该数据库的显式特权。

0x0000000000000040 OBJECT PRIVILEGE
批准存取; 该应用程序 / 用户具有使用该对象或功能的显式特权。

0x0000000000000080 DEFINER
批准存取; 该应用程序 / 用户是该对象或功能的定义者。

0x0000000000000100 OWNER
批准存取; 该应用程序 / 用户是该对象或功能的所有者。

0x0000000000000200 CONTROL
批准存取; 该应用程序 / 用户对该对象或功能具有 CONTROL 特权。

0x0000000000000400 BIND
批准存取; 该应用程序 / 用户对该程序包具有绑定特权。

0x0000000000000800 SYSQUIESCE
批准存取; 如果实例或数据库处于停顿方式, 应用程序 / 用户可连接。

0x0000000000001000 SYSMON
批准存取；该应用程序／用户具有 SYSMON 权限。

相关参考:

- 第 244 页的『CHECKING 事件的审计记录布局』
- 第 247 页的『可能的 CHECKING 存取尝试类型的列表』

可能的 CHECKING 存取尝试类型的列表

以下是可能的 CHECKING 存取尝试类型的列表:

0x0000000000000002 ALTER
试图改变一个对象。

0x0000000000000004 DELETE
试图删除一个对象。

0x0000000000000008 INDEX
试图使用一个索引。

0x0000000000000010 INSERT
试图插入到对象中。

0x0000000000000020 SELECT
试图查询一个表或视图。

0x0000000000000040 UPDATE
试图更新对象中的数据。

0x0000000000000080 REFERENCE
试图在对象间建立引用约束。

0x0000000000000100 CREATE
试图创建一个对象。

0x0000000000000200 DROP
试图删除一个对象。

0x0000000000000400 CREATEIN
试图在另一个模式内创建一个对象。

0x0000000000000800 DROPIN
试图删除在另一个模式内找到的对象。

0x0000000000001000 ALTERIN
试图改变或修改在另一个模式内找到的对象。

0x0000000000002000 EXECUTE
试图执行或运行应用程序或调用例程、创建源于例程的函数（仅适用于函数）或在任何 DDL 语句中引用例程。

0x0000000000004000 BIND
试图绑定或预编译一个应用程序。

0x0000000000008000 SET EVENT MONITOR
试图设置事件监视器开关。

0x0000000000010000 SET CONSTRAINTS
试图设置对一个对象的约束。

0x0000000000020000 COMMENT ON
试图创建有关一个对象的注释。

0x0000000000040000 GRANT
试图将使用一个对象的特权授予另一个用户标识。

0x0000000000080000 REVOKE
试图从一个用户标识撤销使用一个对象的特权。

0x0000000000100000 LOCK
试图锁定一个对象。

0x0000000000200000 RENAME
试图重命名一个对象。

0x0000000000400000 CONNECT
试图与一个对象连接。

0x0000000000800000 SYS 组的成员
试图存取或使用 SYS 组的成员。

0x0000000001000000 Access All
试图执行语句，对对象的所有必需特权被挂起（仅用于 DBADM/SYSADM）。

0x0000000002000000 Drop All
尝试删除多个对象。

0x0000000004000000 LOAD
尝试在表空间中装入表。

0x0000000008000000 USE
试图在表空间中创建表。

0x0000000010000000 SET SESSION_USER
尝试执行 SET SESSION_USER 语句。

0x0000000020000000 FLUSH
尝试执行 FLUSH 语句。

0x0000000040000000 STORE
尝试查看 EXPLAIN_PREDICATE 表中重新优化语句的值。

相关参考:

- 第 244 页的『CHECKING 事件的审计记录布局』
- 第 246 页的『可能的 CHECKING 存取批准原因的列表』

OBJMAINT 事件的审计记录布局

表 10. OBJMAINT 事件的审计记录布局

timestamp=1998-06-24-08.42.41.957524;**category=OBJMAINT;audit event=CREATE_OBJECT;**
event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;
package section=0;object schema=BOSS;object name=AUDIT;object type=TABLE;		

名称	**格式**	**描述**
Timestamp	CHAR(26)	审计事件的日期和时间。
Category	CHAR(8)	审计事件的类别。可能的值是: OBJMAINT
Audit Event	VARCHAR(32)	特定的审计事件。 可能的值包括: CREATE_OBJECT、RENAME_OBJECT 和 DROP_OBJECT
Event Correlator	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件相关。
Event Status	INTEGER	审计事件的状态，由 SQLCODE 表示，其中 成功的事件 > = 0 失败的事件 < 0
Database Name	CHAR(8)	为之生成该事件的数据库的名称。若它是实例级别审计事件，则为空白。
User ID	VARCHAR(1024)	审计事件发生时的用户标识。
Authorization ID	VARCHAR(128)	审计事件发生时的授权标识。
Origin Node Number	SMALLINT	审计事件发生时所在的节点号。
Coordinator Node Number	SMALLINT	协调程序节点的节点号。
Application ID	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
Application Name	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
Package Schema	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
Package Name	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。
Package Section Number	SMALLINT	审计事件发生时正在使用的程序包中的节号。
Object Schema	VARCHAR(128)	为之生成审计事件的对象的模式。
Object Name	VARCHAR(128)	为之生成审计事件的对象的名称。
Object Type	VARCHAR(32)	为之生成审计事件的对象的类型。可能的值包括: 显示在标题为『审计记录对象类型』的主题中的那些值。
Package Version	VARCHAR (64)	审计事件发生时正在使用的程序包的版本。

相关概念:

- 第 229 页的『DB2 通用数据库（DB2 UDB）审计设施简介』

相关参考:

- 第 245 页的『审计记录对象类型』

SECMAINT 事件的审计记录布局

表 11. SECMAINT 事件的审计记录布局

```
timestamp=1998-06-24-11.57.45.188101;category=SECMAINT;audit event=GRANT;
 event correlator=4;event status=0;
 database=FOO;userid=boss;authid=BOSS;
 application id=*LOCAL.boss.980624155728;application name=db2bp;
 package schema=NULLID;package name=SQLC28A1;
 package section=0;object schema=BOSS;object name=T1;object type=TABLE;
 grantor=BOSS;grantee=WORKER;grantee type=USER;privilege=SELECT;
```

名称	格式	描述
Timestamp	CHAR(26)	审计事件的日期和时间。
Category	CHAR(8)	审计事件的类别。可能的值是: SECMAINT
Audit Event	VARCHAR(32)	特定的审计事件。 可能的值包括: GRANT、REVOKE、IMPLICIT_GRANT、IMPLICIT_REVOKE、SET_SESSION_USER 和 UPDATE_DBM_CFG。
Event Correlator	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件相关。
Event Status	INTEGER	审计事件的状态，由 SQLCODE 表示，其中 成功的事件 > = 0 失败的事件 < 0
Database Name	CHAR(8)	为之生成该事件的数据库的名称。若它是实例级别审计事件，则为空白。
User ID	VARCHAR(1024)	审计事件发生时的用户标识。
Authorization ID	VARCHAR(128)	审计事件发生时的授权标识。
Origin Node Number	SMALLINT	审计事件发生时所在的节点号。
Coordinator Node Number	SMALLINT	协调程序节点的节点号。
Application ID	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
Application Name	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
Package Schema	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
Package Name	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。
Package Section Number	SMALLINT	审计事件发生时正在使用的程序包中的节号。
Object Schema	VARCHAR(128)	为之生成审计事件的对象的模式。
Object Name	VARCHAR(128)	为之生成审计事件的对象的名称。
Object Type	VARCHAR(32)	为之生成审计事件的对象的类型。可能的值包括: 显示在标题为『审计记录对象类型』的主题中的那些值。
Grantor	VARCHAR(128)	授权者标识。
Grantee	VARCHAR(128)	被授予或撤消特权或权限的被授权者标识。
Grantee Type	VARCHAR(32)	被授予或撤消权限的被授权者的类型。可能的值包括: USER、GROUP 或 BOTH。

表 11. SECMAINT 事件的审计记录布局 (续)

```
timestamp=1998-06-24-11.57.45.188101;category=SECMAINT;audit event=GRANT;
 event correlator=4;event status=0;
 database=FOO;userid=boss;authid=BOSS;
 application id=*LOCAL.boss.980624155728;application name=db2bp;
 package schema=NULLID;package name=SQLC28A1;
 package section=0;object schema=BOSS;object name=T1;object type=TABLE;
 grantor=BOSS;grantee=WORKER;grantee type=USER;privilege=SELECT;
```

名称	格式	描述
Privilege or Authority	CHAR(18)	指示授予或撤消的特权或权限的类型。可能的值包括: 显示在标题为『可能的 SECMAINT 特权或权限的列表』的主题中的那些值。
Package Version	VARCHAR (64)	审计事件发生时正在使用的程序包的版本。

相关概念:

- 第 242 页的『审计设施记录布局（简介）』

相关参考:

- 第 251 页的『可能的 SECMAINT 特权或权限的列表』
- 第 245 页的『审计记录对象类型』

可能的 SECMAINT 特权或权限的列表

以下是可能的 SECMAINT 特权或权限的列表:

0x0000000000000001 Control Table
授予的或撤销的对表或视图的控制特权。

0x0000000000000002 ALTER TABLE
授予的或撤销的改变一个表的特权。

0x0000000000000004 ALTER TABLE with GRANT
对于允许授予特权的一个表，授予的或撤销的改变该表的特权。

0x0000000000000008 DELETE TABLE
授予的或撤销的删除表或视图的特权。

0x0000000000000010 DELETE TABLE with GRANT
对于允许授予特权的一个表，授予的或撤销的删除该索引的特权。

0x0000000000000020 Table Index
授予的或撤销的对索引的特权。

0x0000000000000040 Table Index with GRANT
对于允许授予特权的一个索引，授予的或撤销的对该索引的特权。

0x0000000000000080 Table INSERT
授予的或撤销的对表或视图进行插入的特权。

0x0000000000000100 Table INSERT with GRANT
对于允许授予特权的一个表，对表进行插入授予或撤销的特权。

0x0000000000000200 Table SELECT
授予的或撤销的对表进行选择的特权。

0x0000000000000400 Table SELECT with GRANT
对于允许授予特权的一个表，授予的或撤销的对表进行选择的特权。

0x0000000000000800 Table UPDATE
授予的或撤销的对表或视图进行更新的特权。

0x0000000000001000 Table UPDATE with GRANT
对于允许授予特权的表或视图，授予或撤销的对表或视图进行更新的特权。

0x0000000000002000 Table REFERENCE
授予的或撤销的对表进行引用的特权。

0x0000000000004000 Table REFERENCE with GRANT
对于允许授予特权的一个表，授予的或撤销的对表进行引用的特权。

0x0000000000020000 CREATEIN Schema
授予的或撤销的对一个模式的 CREATEIN 特权。

0x0000000000040000 CREATEIN Schema with GRANT
对于允许授予特权的一个模式，授予的或撤销的对该模式的 CREATEIN 特权。

0x0000000000080000 DROPIN Schema
授予的或撤销的对一个模式的 DROPIN 特权。

0x0000000000100000 DROPIN Schema with GRANT
对于允许授予特权的一个模式，授予的或撤销的对该模式的 DROPIN 特权。

0x0000000000200000 ALTERIN Schema
授予的或撤销的对一个模式的 ALTERIN 特权。

0x0000000000400000 ALTERIN Schema with GRANT
对于允许授予特权的一个模式，授予的或撤销的对该模式的 ALTERIN 特权。

0x0000000000800000 DBADM Authority
授予的或撤销的 DBADM 权限。

0x0000000001000000 CREATETAB Authority
授予的或撤销的 Createtab 权限。

0x0000000002000000 BINDADD Authority
授予的或撤销的 Bindadd 权限。

0x0000000004000000 CONNECT Authority
授予的或撤销的 CONNECT 权限。

0x0000000008000000 Create not fenced Authority
授予的或撤销的 Create not fenced 权限。

0x0000000010000000 Implicit Schema Authority
授予的或撤销的 Implicit Schema 权限。

0x0000000020000000 Server PASSTHRU
授予的或撤销的对此服务器（联合数据库数据源）使用传递（Pass-Through）设施的特权。

0x0000000100000000 Table Space USE
授予的或撤销的在表空间中创建表的特权。

0x0000000200000000 Table Space USE with GRANT
授予的或撤销的在表空间中创建表并允许进行特权授权的特权。

0x0000000400000000 Column UPDATE
授予的或撤销的对一个表的一个或多个特定列进行更新的特权。

0x0000000800000000 Column UPDATE with GRANT
对于允许授予特权的一个表，授予的或撤销的对该表的一个或多个特定列进行更新的特权。

0x0000001000000000 Column REFERENCE
授予的或撤销的对一个表的一个或多个特定列进行引用的特权。

0x0000002000000000 Column REFERENCE with GRANT
对于允许授予特权的一个表，授予的或撤销的对该表的一个或多个特定列进行引用的特权。

0x0000004000000000 LOAD Authority
授予的或撤销的 LOAD 权限。

0x0000008000000000 Package BIND
授予的或撤销的对一个程序包的 BIND 特权。

0x0000010000000000 Package BIND with GRANT
对于允许授予特权的一个程序包，被授予或撤销的对该程序包的 BIND 特权。

0x0000020000000000 EXECUTE
授予的或撤销的对程序包或例程的 EXECUTE 特权。

0x0000040000000000 EXECUTE with GRANT
对于允许授予特权的程序包或例程，授予的或撤销的对该程序包或例程的 EXECUTE 特权。

0x0000080000000000 EXECUTE IN SCHEMA
对模式中所有例程被授予的或被撤销的 EXECUTE 特权。

0x0000100000000000 EXECUTE IN SCHEMA with GRANT
对于允许授予特权的模式中的所有例程，授予的或撤销的对这些例程的 EXECUTE 特权。

0x0000200000000000 EXECUTE IN TYPE
授予的或被撤销的对某个类型中的所有例程的 EXECUTE 特权。

0x0000400000000000 EXECUTE IN TYPE with GRANT
对于允许授予特权的类型中的所有例程，被授予的或被撤销的对这些例程的 EXECUTE 特权。

0x0000800000000000 CREATE EXTERNAL ROUTINE
被授予或被撤销的 CREATE EXTERNAL ROUTINE 特权。

0x0001000000000000 QUIESCE_CONNECT
被授予的或被撤销的 QUIESCE_CONNECT 特权。

相关参考:

- 第 250 页的『SECMAINT 事件的审计记录布局』

SYSADMIN 事件的审计记录布局

表 12. SYSADMIN 事件的审计记录布局

timestamp=1998-06-24-11.54.04.129923;**category=SYSADMIN;audit event=DB2AUDIT;**
event correlator=1;event status=0;
userid=boss;authid=BOSS;
application id=*LOCAL.boss.980624155404;application name=db2audit;		

名称	**格式**	**描述**
Timestamp	CHAR(26)	审计事件的日期和时间。
Category	CHAR(8)	审计事件的类别。可能的值是: SYSADMIN
Audit Event	VARCHAR(32)	特定的审计事件。 可能的值包括: 显示在此表后的列表中的那些值。
Event Correlator	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件相关。
Event Status	INTEGER	审计事件的状态，由 SQLCODE 表示，其中 成功的事件 > = 0 失败的事件 < 0
Database Name	CHAR(8)	为之生成该事件的数据库的名称。若它是实例级别审计事件，则为空白。
User ID	VARCHAR(1024)	审计事件发生时的用户标识。
Authorization ID	VARCHAR(128)	审计事件发生时的授权标识。
Origin Node Number	SMALLINT	审计事件发生时所在的节点号。
Coordinator Node Number	SMALLINT	协调程序节点的节点号。
Application ID	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
Application Name	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
Package Schema	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
Package Name	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。
Package Section Number	SMALLINT	审计事件发生时正在使用的程序包中的节号。
Package Version	VARCHAR (64)	审计事件发生时正在使用的程序包的版本。

相关概念:

- 第 242 页的『审计设施记录布局（简介）』

相关参考:

- 第 254 页的『可能的 SYSADMIN 审计事件的列表』

可能的 SYSADMIN 审计事件的列表

以下是可能的 SYSADMIN 审计事件的列表:

表 13. SYSADMIN 审计事件

START_DB2
STOP_DB2
CREATE_DATABASE
DROP_DATABASE
UPDATE_DBM_CFG
UPDATE_DB_CFG
CREATE_TABLESPACE
DROP_TABLESPACE
ALTER_TABLESPACE
RENAME_TABLESPACE
CREATE_NODEGROUP
DROP_NODEGROUP
ALTER_NODEGROUP
CREATE_BUFFERPOOL
DROP_BUFFERPOOL
ALTER_BUFFERPOOL
CREATE_EVENT_MONITOR
DROP_EVENT_MONITOR
ENABLE_MULTIPAGE
MIGRATE_DB_DIR
DB2TRC
DB2SET
ACTIVATE_DB
ADD_NODE
BACKUP_DB
CATALOG_NODE
CATALOG_DB
CATALOG_DCS_DB
CHANGE_DB_COMMENT
DEACTIVATE_DB
DROP_NODE_VERIFY
FORCE_APPLICATION
GET_SNAPSHOT
LIST_DRDA_INDOUBT_TRANSACTIONS
MIGRATE_DB
RESET_ADMIN_CFG
RESET_DB_CFG
RESET_DBM_CFG
RESET_MONITOR
RESTORE_DB
ROLLFORWARD_DB
SET_RUNTIME_DEGREE
SET_TABLESPACE_CONTAINERS
UNCATALOG_DB
UNCATALOG_DCS_DB
UNCATALOG_NODE
UPDATE_ADMIN_CFG
UPDATE_MON_SWITCHES
LOAD_TABLE
DB2AUDIT
SET_APPL_PRIORITY
CREATE_DB_AT_NODE
KILLDBM
MIGRATE_SYSTEM_DIRECTORY
DB2REMOT
DB2AUD
MERGE_DBM_CONFIG_FILE
UPDATE_CLI_CONFIGURATION
OPEN_TABLESPACE_QUERY
SINGLE_TABLESPACE_QUERY
CLOSE_TABLESPACE_QUERY
FETCH_TABLESPACE
OPEN_CONTAINER_QUERY
FETCH_CONTAINER_QUERY
CLOSE_CONTAINER_QUERY
GET_TABLESPACE_STATISTICS
DESCRIBE_DATABASE
ESTIMATE_SNAPSHOT_SIZE
READ_ASYNC_LOG_RECORD
PRUNE_RECOVERY_HISTORY
UPDATE_RECOVERY_HISTORY
QUIESCE_TABLESPACE
UNLOAD_TABLE
UPDATE_DATABASE_VERSION
CREATE_INSTANCE
DELETE_INSTANCE
SET_EVENT_MONITOR
GRANT_DBADM
REVOKE_DBADM
GRANT_DB_AUTHORITIES
REVOKE_DB_AUTHORITIES
REDIST_NODEGROUP

相关参考:

• 第 254 页的『SYSADMIN 事件的审计记录布局』

VALIDATE 事件的审计记录布局

表 14. VALIDATE 事件的审计记录布局

timestamp=1998-06-24-08.42.11.527490;**category=VALIDATE;audit event=CHECK_GROUP_MEMBERSHIP;**
event correlator=2;event status=-1092;
database=FOO;userid=boss;authid=BOSS;**execution id=newton;**
application id=*LOCAL.newton.980624124210;application name=testapp;
auth type=SERVER;		

名称	**格式**	**描述**
Timestamp	CHAR(26)	审计事件的日期和时间。
Category	CHAR(8)	审计事件的类别。可能的值是: VALIDATE
Audit Event	VARCHAR(32)	特定的审计事件。 可能的值包括: GET_GROUPS, GET_USERID, AUTHENTICATE_PASSWORD 和 VALIDATE_USER。
Event Correlator	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件相关。
Event Status	INTEGER	审计事件的状态, 由 SQLCODE 表示, 其中 成功的事件 > = 0 失败的事件 < 0
Database Name	CHAR(8)	为之生成该事件的数据库的名称。若它是实例级别审计事件, 则为空白。
User ID	VARCHAR(1024)	审计事件发生时的用户标识。
Authorization ID	VARCHAR(128)	审计事件发生时的授权标识。
Execution ID	VARCHAR(1024)	审计事件发生时正在使用的执行标识。
Origin Node Number	SMALLINT	审计事件发生时所在的节点号。
Coordinator Node Number	SMALLINT	协调程序节点的节点号。
Application ID	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
Application Name	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
Authentication Type	VARCHAR(32)	审计事件发生时的认证类型。
Package Schema	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
Package Name	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。
Package Section Number	SMALLINT	审计事件发生时正在使用的程序包中的节号。
Package Version	VARCHAR (64)	审计事件发生时正在使用的程序包的版本。
Plug-in Name	VARCHAR(32)	审计事件发生时正在使用的插件的名称。

相关概念:

- 第 242 页的『审计设施记录布局（简介）』

CONTEXT 事件的审计记录布局

表 15. CONTEXT 事件的审计记录布局

```
timestamp=1998-06-24-08.42.41.476840;category=CONTEXT;audit event=EXECUTE_IMMEDIATE;
 event correlator=3;
 database=FOO;userid=boss;authid=BOSS;
 application id=*LOCAL.newton.980624124210;application name=testapp;
 package schema=NULLID;package name=SQLC28A1;
 package section=203;text=create table audit(c1 char(10), c2 integer);
```

名称	格式	描述
Timestamp	CHAR(26)	审计事件的日期和时间。
Category	CHAR(8)	审计事件的类别。可能的值是: CONTEXT
Audit Event	VARCHAR(32)	特定的审计事件。 可能的值包括: 显示在此表后的列表中的那些值。
Event Correlator	INTEGER	正在审计的操作的相关标识。可用来标识哪些审计记录与单个事件相关。
Database Name	CHAR(8)	为之生成该事件的数据库的名称。若它是实例级别审计事件，则为空白。
User ID	VARCHAR(1024)	审计事件发生时的用户标识。
Authorization ID	VARCHAR(128)	审计事件发生时的授权标识。
Origin Node Number	SMALLINT	审计事件发生时所在的节点号。
Coordinator Node Number	SMALLINT	协调程序节点的节点号。
Application ID	VARCHAR(255)	审计事件发生时正在使用的应用程序标识。
Application Name	VARCHAR(1024)	审计事件发生时正在使用的应用程序名。
Package Schema	VARCHAR(128)	审计事件发生时正在使用的程序包的模式。
Package Name	VARCHAR(128)	审计事件发生时正在使用的程序包的名称。
Package Section Number	SMALLINT	审计事件发生时正在使用的程序包中的节号。
Statement Text (statement)	CLOB(2M)	SQL 语句的文本（若适用）。若 SQL 语句文本不可用，则为 null。
Package Version	VARCHAR (64)	审计事件发生时正在使用的程序包的版本。

相关概念:

- 第 242 页的『审计设施记录布局（简介）』

相关参考:

- 第 257 页的『可能的 CONTEXT 审计事件的列表』

可能的 CONTEXT 审计事件的列表

以下是可能的 CONTEXT 审计事件的列表:

表 16. CONTEXT 审计事件

CONNECT	SET_APPL_PRIORITY
CONNECT_RESET	RESET_DB_CFG
ATTACH	GET_DB_CFG
DETACH	GET_DFLT_CFG
DARI_START	UPDATE_DBM_CFG
DARI_STOP	SET_MONITOR
BACKUP_DB	GET_SNAPSHOT
RESTORE_DB	ESTIMATE_SNAPSHOT_SIZE
ROLLFORWARD_DB	RESET_MONITOR
OPEN_TABLESPACE_QUERY	OPEN_HISTORY_FILE
FETCH_TABLESPACE	CLOSE_HISTORY_FILE
CLOSE_TABLESPACE_QUERY	FETCH_HISTORY_FILE
OPEN_CONTAINER_QUERY	SET_RUNTIME_DEGREE
CLOSE_CONTAINER_QUERY	UPDATE_AUDIT
FETCH_CONTAINER_QUERY	DBM_CFG_OPERATION
SET_TABLESPACE_CONTAINERS	DISCOVER
GET_TABLESPACE_STATISTIC	OPEN_CURSOR
READ_ASYNC_LOG_RECORD	CLOSE_CURSOR
QUIESCE_TABLESPACE	FETCH_CURSOR
LOAD_TABLE	EXECUTE
UNLOAD_TABLE	EXECUTE_IMMEDIATE
UPDATE_RECOVERY_HISTORY	PREPARE
PRUNE_RECOVERY_HISTORY	DESCRIBE
SINGLE_TABLESPACE_QUERY	BIND
LOAD_MSG_FILE	REBIND
UNQUIESCE_TABLESPACE	RUNSTATS
ENABLE_MULTIPAGE	REORG
DESCRIBE_DATABASE	REDISTRIBUTE
DROP_DATABASE	COMMIT
CREATE_DATABASE	ROLLBACK
ADD_NODE	REQUEST_ROLLBACK
FORCE_APPLICATION	IMPLICIT_REBIND

相关参考:

- 第 257 页的『CONTEXT 事件的审计记录布局』

审计设施技巧和方法

在大多数情况下，当使用 CHECKING 事件时，审计记录中的对象类型字段是要检查的对象，以了解试图存取该对象的用户标识是否拥有必需的特权或权限。例如，若一个用户试图通过添加列来改变一个表，则 CHECKING 事件审计记录将指示尝试的存取是“ALTER”，且要检查的对象类型是“TABLE”（注意: 不是列，因为它是必须检查的表特权）。

但是，当该检查要验证是否存在一个数据库权限来允许用户标识创建或绑定对象或删除对象时，虽然存在对该数据库的检查，对象类型字段仍将指定要创建、绑定或删除的对象（而不是数据库本身）。

当在表上创建一个索引时，需要具有创建索引的特权，因此 CHECKING 事件审计记录将具有存取尝试类型“索引”而不是“创建”。

当绑定一个已存在的程序包时，会为该程序包的 DROP 创建 OBJMAINT 事件审计记录，然后为该程序包新副本的 CREATE 创建另一个 OBJMAINT 事件审计记录。

SQL“数据定义语言”（DDL）可生成记录为成功的 OBJMAINT 或 SECMAINT 事件。但是，在记录该事件后，一个后续的错误可能会导致 ROLLBACK 发生。这样就不能创建该对象；或者 GRANT 或 REVOKE 操作不能完成。在这种情况下，使用 CONTEXT 事件变得很重要。这类 CONTEXT 事件审计记录，特别是结束该事件的语句，将指示尝试的操作的完成性质。

当抽取定界 ASCII 格式（适合装入 DB2® 通用数据库（DB2 UDB）关系表）的审计记录时，应清楚该语句文本字段内使用的定界符的有关情况。这可在抽取该定界的 ASCII 文件时执行，并使用下列语句来执行:

```
db2audit extract delasc delimiter <load delimiter>
```

*装入定界符*可以是单个字符（如 "），或是表示十六进制值的四字节字符串（如“0xff”）。有效命令的示例是:

```
db2audit extract delasc
   db2audit extract delasc delimiter !
   db2audit extract delasc delimiter 0xff
```

若您在抽取时使用的定界符不是缺省装入定界符（“”），则应在 LOAD 命令上使用 MODIFIED BY 选项。下面是将“0xff”用作定界符的 LOAD 命令的示例一部分:

```
db2 load from context.del of del modified by chardel0xff replace into ...
```

它将覆盖缺省装入字符串定界符“0xff”。

相关概念:

- 第 242 页的『审计设施记录布局（简介）』

相关参考:

- 第 232 页的『审计设施的使用』

控制 DB2 UDB 审计设施活动

过程:

当讨论控制审计设施活动时，我们将使用一个简单的方案：用户 *newton* 运行一个名为 *testapp* 的应用程序，以连接和创建一个表。在下面讨论的每个示例中都使用这个相同的应用程序。

我们从一个极端情况的示例开始：您已确定审计所有成功和不成功的审计事件，因此将以下列方式配置审计设施:

```
db2audit configure scope all status both
```

注: 它为每个可能的可审计事件创建审计记录。因此，会将许多记录写入审计日志，这降低了数据库管理器的性能。这里显示这个极端情况，仅用于演示；不建议您使用上面显示的命令配置审计设施。

在对此配置开始使用审计设施（使用 “db2audit start” ），并运行 *testapp* 应用程序之后，生成了下列记录，将它们放在审计日志中。通过从该日志中抽取审计记录，您将看到为该应用程序执行的两个操作生成的下列记录:

操作　创建的记录的类型

CONNECT

```
timestamp=1998-06-24-08.42.10.555345;category=CONTEXT;
audit event=CONNECT;event correlator=2;database=FOO;
application id=*LOCAL.newton.980624124210;
application name=testapp;

timestamp=1998-06-24-08.42.10.944374;category=VALIDATE;
audit event=AUTHENTICATION;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;execution id=newton;
application id=*LOCAL.newton.980624124210;application name=testapp;
auth type=SERVER;

timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.561187;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.594620;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.622984;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
object name=FOO;object type=DATABASE;access approval reason=DATABASE;
access attempted=CONNECT;

timestamp=1998-06-24-08.42.11.801554;category=CONTEXT;
audit event=COMMIT;event correlator=2;database=FOO;userid=boss;
authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;

timestamp=1998-06-24-08.42.41.450975;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;object schema=NULLID;
object name=SQLC28A1;object type=PACKAGE;
access approval reason=OBJECT;access attempted=EXECUTE;
```

CREATE TABLE

```
timestamp=1998-06-24-08.42.41.476840;category=CONTEXT;
audit event=EXECUTE_IMMEDIATE;event correlator=3;database=FOO;
userid=boss;authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;package schema=NULLID;package name=SQLC28A1;
package section=203;text=create table audit(c1 char(10), c2 integer);

timestamp=1998-06-24-08.42.41.539692;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
```

```
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;
access approval reason=DATABASE;access attempted=CREATE;

timestamp=1998-06-24-08.42.41.570876;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object name=BOSS;object type=SCHEMA;access approval reason=DATABASE;
access attempted=CREATE;

timestamp=1998-06-24-08.42.41.957524;category=OBJMAINT;
audit event=CREATE_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;

timestamp=1998-06-24-08.42.42.018900;category=CONTEXT;
audit event=COMMIT;event correlator=3;database=FOO;userid=boss;
authid=BOSS;application id=*LOCAL.newton.980624124210;
application name=testapp;package schema=NULLID;
package name=SQLC28A1;
```

正如您所见到的那样，由于审计配置请求所有可能的审计事件和类型的审计，因此从该审计配置生成了相当多的审计记录。

在大多数情况下，您将配置审计设施，以获得您希望审计的事件的更具体或更集中的视图。例如，您可能只想审计那些失败的事件。在这种情况下，可按如下所示配置审计设施:

```
db2audit configure scope audit,checking,objmaint,secmaint,sysadmin,
      validate status failure
```

注: 此配置是初始审计配置，或是在复位该审计配置时出现的配置。

在对此配置开始使用审计设施并运行 *testapp* 应用程序之后，生成了下列记录，将它们放在审计日志中。（而且我们假定 *testapp* 以前未运行过。）通过从该日志中抽取审计记录，您将看到为该应用程序执行的两个操作生成的下列记录:

操作　创建的记录的类型

CONNECT

```
timestamp=1998-06-24-08.42.11.527490;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.561187;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;

timestamp=1998-06-24-08.42.11.594620;category=VALIDATE;
audit event=CHECK_GROUP_MEMBERSHIP;event correlator=2;
event status=-1092;database=FOO;userid=boss;authid=BOSS;
execution id=newton;application id=*LOCAL.newton.980624124210;
application name=testapp;auth type=SERVER;
```

CREATE TABLE

（无）

由于此审计配置仅在该事件尝试失败时才请求所有可能的审计事件（除 CONTEXT 外）的审计，因此，此审计配置生成的审计记录相当少。通过更改该审计配置，您可控制生成的审计记录的类型和性质。

当您想审计的那些事件被成功授予对一个对象的特权时，审计设施可允许您创建审计记录。在这种情况下，您可按如下所示配置审计设施:

```
db2audit configure scope checking status success
```

在对此配置开始使用审计设施并运行 *testapp* 应用程序之后，生成了下列记录，将它们放在审计日志中。（而且我们假定 *testapp* 以前未运行过。）通过从该日志中抽取审计记录，您将看到为该应用程序执行的两个操作生成的下列记录:

操作　创建的记录的类型

CONNECT

```
timestamp=1998-06-24-08.42.11.622984;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;

timestamp=1998-06-24-08.42.41.450975;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=2;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;object schema=NULLID;
object name=SQLC28A1;object type=PACKAGE;
access approval reason=OBJECT;access attempted=EXECUTE;

timestamp=1998-06-24-08.42.41.539692;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object schema=BOSS;object name=AUDIT;object type=TABLE;
access approval reason=DATABASE;access attempted=CREATE;

timestamp=1998-06-24-08.42.41.570876;category=CHECKING;
audit event=CHECKING_OBJECT;event correlator=3;event status=0;
database=FOO;userid=boss;authid=BOSS;
application id=*LOCAL.newton.980624124210;application name=testapp;
package schema=NULLID;package name=SQLC28A1;package section=0;
object name=BOSS;object type=SCHEMA;access approval reason=DATABASE;
access attempted=CREATE;
```

CREATE TABLE

（无）

相关概念:

- 第 242 页的『审计设施记录布局（简介）』

相关参考:

- 第 232 页的『审计设施的使用』

第 3 部分 附录

附录 A. 符合命名规则

通用命名规则

所有对象和用户的命名都有规则。某些规则与所用的平台有关。例如，有的规则是关于名称中的大小写使用。

- 在 UNIX® 平台上，名称必须小写。
- 在 Windows® 平台上，名称可以是大写、小写或大小写混合。

除非另有指定，否则所有名称均可以包括下列字符:

- A 到 Z。当在大多数名称中使用时，字符 A 至 Z 将从小写形式转换为大写形式。
- 0 至 9。
- ! % ( ) { } . – ^ ~ _（下划线）@、#、$ 和空格。
- \（反斜杠）。

名称不能以数字或下划线字符开始。

不要使用 SQL 保留字来命名表、视图、列、索引或授权标识。

有一些其它特殊字符，它们所起的作用取决于操作系统以及使用 DB2® 通用数据库（DB2 UDB）的位置。虽然它们可能生效，但并不保证它们会生效。建议在数据库中命名对象时不要使用这些其它特殊字符。

还需要考虑对象命名规则、工作站命名规则、NLS 环境中的命名规则以及 Unicode 环境中的命名规则。

相关概念:

- 第 265 页的『DB2 UDB 对象命名规则』
- 第 269 页的『工作站命名规则』
- 第 268 页的『用户、用户标识和组命名规则』
- 第 268 页的『联合数据库对象命名规则』

DB2 UDB 对象命名规则

所有对象都遵从 “一般命名规则”。另外，某些对象具有下表所示的其它限制。

表 17. 数据库、数据库别名和实例命名规则

对象	准则
• 数据库 • 数据库别名 • 实例	• 在对数据库名称进行编目的位置中，这些数据库名称必须是唯一的。在基于 UNIX® 的 DB2® 通用数据库（DB2 UDB）的实现上，此位置是目录路径，而在 Windows® 实现上，它是逻辑磁盘。 • 在系统数据库目录中，数据库别名必须是唯一的。创建新数据库时，别名缺省为数据库名称。因此，就不能使用作为数据库别名存在的名称来创建数据库，即使不存在使用该名称的数据库。 • 数据库、数据库别名和实例名称最多可有 8 个字节。 • 在 Windows NT®、Windows 2000、Windows XP 和 Windows Server 2003 系统上，任何实例都不能与服务名称同名。 **注**: 为避免潜在的问题，在想要在通信环境中使用数据库的情况下，不要在数据库名称中使用特殊字符 @、# 和 $。而且，因为并非所有键盘都使用这些字符，所以，如果打算使用另一种语言版本的数据库，不要使用这些特殊字符。

表 18. 数据库对象命名规则

对象	准则
• 别名 • 缓冲池 • 列 • 事件监视器 • 索引 • 方法 • 节点组 • 程序包 • 程序包版本 • 模式 • 存储过程 • 表 • 表空间 • 触发器 • UDF • UDT • 视图	最多可以包含 18 个字节，下列名称除外: • 表名（包括视图名、总结表名、别名和相关名），最多可以包含 128 个字节 • 列名最多可以包含 30 个字节 • 程序包名，最多可以包含 8 个字节 • 模式名，最多可以包含 30 个字节 • 程序包版本，最多可以包含 64 个字节 • 对象名还可以包括: – 有效的强调字符（例如，ö） – 除了多字节空格之外的多字节字符（对于多字节环境） • 程序包名和程序包版本还可以包括句号（.）、连字符（-）和冒号（:）。

表 19. 联合数据库对象命名规则

对象	准则
• 函数映射 • 索引规范 • 别名 • 服务器 • 类型映射 • 用户映射 • 包装器	• 别名、映射、索引规范、服务器和包装器名不能超过 128 个字节。 • 服务器和别名选项及选项设置限制为 255 个字节。 • 联合数据库对象的名称还可以包括: – 有效的强调字母（例如 ö） – 除了多字节空格之外的多字节字符（对于多字节环境）

定界标识和对象名:

可以使用关键字。如果在某个上下文中使用了一个关键字，而该关键字还可以解释为 SQL 关键字，则必须将其指定为定界标识。

使用定界标识时，可能会创建违反这些命名规则的对象；但是，如果随后使用该对象，则可能导致错误。例如，如果您创建一列，其名称中包括 + 号或 - 号，然后又在索引中使用该列，则当您试图重组该表时将遇到问题。

其它模式名信息:

- 用户定义的类型（UDT）不能具有长度超过 8 个字节的模式名。
- 下列模式名是保留字，一定不能使用: SYSCAT、SYSFUN、SYSIBM 和 SYSSTAT。
- 要避免将来的潜在迁移问题，切勿使用以 SYS 开始的模式名。数据库管理器不允许您使用以 SYS 开始的模式名来创建触发器、用户定义的类型或用户定义的函数。
- 建议不要将 SESSION 用作模式名。已声明临时表必须用 SESSION 来限定。因此，可以让应用程序使用与持久表完全相同的名称来声明临时表，在这种情况下，应用程序逻辑可能会变得过分复杂。除非在处理已声明临时表，否则应避免使用模式 SESSION。

相关概念:

- 第 265 页的『通用命名规则』

定界标识和对象名

可以使用关键字。如果在某个上下文中使用了一个关键字，而该关键字还可以解释为 SQL 关键字，则必须将其指定为定界标识。

使用定界标识时，可能会创建违反这些命名规则的对象；但是，如果随后使用该对象，则可能导致错误。例如，如果您创建一列，其名称中包括 + 号或 - 号，然后又在索引中使用该列，则当您试图重组该表时将遇到问题。

相关概念:

- 第 265 页的『通用命名规则』

用户、用户标识和组命名规则

表 20. 用户、用户标识和组命名规则

对象	准则
• 组名 • 用户名 • 用户标识	• 组名最多可以包含 30 个字节。 • 在基于 UNIX® 的系统上，用户标识最多可以包含 8 个字符。 • 在 Windows® 上，用户名最多可以包含 30 个字符。Windows NT®、Windows 2000、Windows XP 和 Windows Server 2003 当前具有 20 个字符的实际限制。 • 在不使用“客户机”认证时，如果显式指定用户名和密码，支持使用长度超过 8 个字符的用户名将非 Windows 32 位客户机连接至 Windows NT、Windows 2000、Windows XP 和 Windows Server 2003。 • 名称和标识不能: – 是 USERS、ADMINS、GUESTS、PUBLIC、LOCAL 或任何 SQL 保留字 – 以 IBM®、SQL 或 SYS 开头。 – 包括强调字符。

注:

1. 一些操作系统允许区分大小写的用户标识和密码。应该检查操作系统文档以了解是否是这样。
2. 从成功的 CONNECT 或 ATTACH 返回的授权标识被截断为 8 个字符。将省略号（...）追加至授权标识并且 SQLWARN 字段包含警告以指示截断。
3. 从用户标识和密码中除去尾部空格。

相关概念:

- 第 265 页的『通用命名规则』
- 第 268 页的『联合数据库对象命名规则』

联合数据库对象命名规则

表 21. 联合数据库对象命名规则

对象	准则
• 函数映射 • 索引规范 • 别名 • 服务器 • 类型映射 • 用户映射 • 包装器	• 别名、映射、索引规范、服务器和包装器名不能超过 128 个字节。 • 服务器和别名选项及选项设置限制为 255 个字节。 • 联合数据库对象的名称还可以包括: – 有效的强调字母（例如 ö） – 除了多字节空格之外的多字节字符（对于多字节环境）

相关概念:

- 第 265 页的『通用命名规则』

与模式名使用有关的其它限制和建议

- 用户定义的类型（UDT）不能具有长度超过 8 个字节的模式名。
- 下列模式名是保留字，一定不能使用: SYSCAT、SYSFUN、SYSIBM 和 SYSSTAT。
- 要避免将来的潜在迁移问题，切勿使用以 SYS 开始的模式名。数据库管理器不允许您使用以 SYS 开始的模式名来创建触发器、用户定义的类型或用户定义的函数。
- 建议不要将 SESSION 用作模式名。已声明临时表必须用 SESSION 来限定。因此，可以让应用程序使用与持久表完全相同的名称来声明临时表，在这种情况下，应用程序逻辑可能会变得过分复杂。除非在处理已声明临时表，否则应避免使用模式 SESSION。

相关概念:

- 第 265 页的『通用命名规则』

在服务器上维护密码

可能需要执行密码维护任务。因为在服务器上需要这样的任务，并且许多用户无法使用或不能很好地使用服务器环境，所以执行这些任务可能会比较困难。DB2® 通用数据库（DB2 UDB）提供了一种方法来更新和验证密码，而不必在服务器上进行。例如，“DB2 OS/390® 版版本 5”支持使用此方法来更改用户的密码。如果接收到错误消息 SQL1404N“密码到期”，则使用 CONNECT 语句按如下所示更改密码:

```
CONNECT TO <database> USER <userid> USING <password>
   NEW <new_password> CONFIRM <new_password>
```

也可以使用“DB2 UDB 配置助手”（CA）的“密码更改”对话框更改密码。

相关概念:

- 第 265 页的『通用命名规则』
- 第 265 页的『DB2 UDB 对象命名规则』
- 第 269 页的『工作站命名规则』
- 第 268 页的『用户、用户标识和组命名规则』
- 第 268 页的『联合数据库对象命名规则』
- 第 267 页的『定界标识和对象名』
- 第 269 页的『与模式名使用有关的其它限制和建议』

工作站命名规则

*工作站名称*指定驻留在本地工作站上的数据库服务器、数据库客户机或 DB2® 通用数据库（DB2 UDB）个人版的 NetBIOS 名称。此名称存储在数据库管理器配置文件中。该工作站名称为*工作站 nname*。

另外，指定的名称:

- 可包含 1 至 8 个字符

- 不能包括 &、# 或 @
- 在网络内必须是唯一的

在分区数据库系统中，仍然只有一个工作站 *nname*，它表示整个分区数据库系统，但是每个节点都有它自己派生的唯一的 NetBIOS *nname*。

表示分区数据库系统的工作站 *nname* 存储在实例拥有的数据库分区服务器的数据库管理器配置文件中。

每个节点的唯一 *nname* 是工作站 *nname* 和节点号的派生项的组合。

如果一个节点不实例拥有，则它的 NetBIOS *nname* 是按如下所示派生的:

1. 实例拥有的机器的工作站 *nname* 的第一个字符用作该节点的 NetBIOS *nname* 的第一个字符。
2. 后面的 1 至 3 个字符表示节点号。范围是 1 到 999。
3. 其余字符来自于实例拥有的机器的工作站 *nname*。余下字符的数目取决于实例拥有的机器的工作站 *nname* 的长度。此数目可以是 0 至 4。

例如:

实例拥有的机器的*工作站 nname*	**节点号**	**派生的节点 NetBIOS** *nname*
GEORGE	3	G3ORGE
A	7	A7
B2	94	B942
N0076543	21	N216543
GEORGE5	1	G1RGE5

如果您在安装期间更改了缺省工作站 *nname*，则该工作站 *nname* 的最后 4 个字符应该在整个 NetBIOS 网络中是唯一的，以便将派生出冲突的 NetBIOS *nname* 的可能性减至最小。

相关概念:

- 第 265 页的『通用命名规则』

NLS 环境中的命名规则

可以在数据库名称中使用的基本字符集包含单字节大写和小写拉丁字母（A...Z 和 a...z）、阿拉伯数字（0...9）和下划线字符（_）。此列表增加了三个特殊字符（#、@ 和 $）以提供与主机数据库产品的兼容性。在 NLS 环境中使用特殊字符 #、@ 和 $ 应小心，因为它们未包括在 NLS 主机（EBCDIC）不变量字符集中。视正在使用的代码页而定，还可以使用来自扩展字符集的字符。如果要在一个多代码页环境中使用数据库，则必须确保所有代码页都支持您计划使用的扩展字符集中的任何元素。

当命名数据库对象（如表和视图）、程序标号、主变量、游标时，也可使用扩展字符集（例如，带有区分标记的字母）中的元素。哪些字符正好可用取决于正在使用的代码页。

DBCS 标识的扩展字符集定义:

在 DBCS 环境中，扩展字符集包含基本字符集中的所有字符以及下列各项:

- 除双字节空间外，每个 DBCS 代码页中的所有双字节字符都是有效的字母。
- 双字节空间是特殊字符。
- 在每个混合的代码页中可用的单字节字符被分配给各种类别，如下所示:

类别	在每个混合代码页内有效的代码点
数字	x30-39
字母	x23-24、x40-5A、x61-7A、xA6-DF（A6-DF 仅用于代码页 932 和 942）
特殊字符	所有其它有效的单字节字符代码点

相关概念:

- 第 265 页的『通用命名规则』
- 第 265 页的『DB2 UDB 对象命名规则』
- 第 269 页的『工作站命名规则』

Unicode 环境中的命名规则

在 UCS-2 数据库中，所有标识都使用多字节 UTF-8。因此，对于 DB2® 通用数据库（DB2 UDB）允许在其中使用扩展字符集中的字符（例如，强调字符或多字节字符）的标识，可以在该标识中使用任何 UCS-2 字符。

客户机可输入其环境支持的任何字符，并且标识中的所有字符都由数据库管理器转换为 UTF-8。在为 UCS-2 数据库指定标识中的本地语言字符时，必须考虑以下两点:

- 每个非 ASCII 字符均需要两个到四个字节。因此，根据 ASCII 与非 ASCII 字符的比率，一个 *n* 字节标识只能容纳 *n*/4 至 *n* 个字符。如果仅有一个或两个非 ASCII 字符（例如，强调字符），则该限制接近 *n* 个字符，而对于全部由非 ASCII 字符组成的标识（例如，日语），只能使用 *n*/4 至 *n*/3 个字符。
- 如果要在不同的客户机环境中输入标识，则应该使用这些客户机可用的公共字符子集来定义标识。例如，如果要从拉丁语系 1、阿拉伯语和日语环境中存取 UCS-2 数据库，则所有标识应该严格限制为 ASCII。

相关概念:

- 第 265 页的『通用命名规则』
- 第 265 页的『DB2 UDB 对象命名规则』
- 第 269 页的『工作站命名规则』

附录 B. 使用自动客户机重新路由

本附录描述自动客户机如何重新路由工作以及如何确保该客户机在环境中正确工作。

自动客户机重新路由的描述和设置

当服务器崩溃时，与服务器相连的每台客户机均会接收到通信错误，这会使导致应用程序错误的连接终止。如果可用性极其重要，则应实现冗余设置或具备将服务器故障转移到备用节点的能力。在上述任一情况下，DB2® 通用数据库（DB2 UDB）客户机代码尝试重新建立与原始服务器的连接，此服务器可能正在故障转移节点上运行（IP 地址也进行故障转移），或者建立与新服务器的连接。

重新建立连接之后，应用程序接收到一则错误消息，通知其发生了事务故障，但应用程序可以继续下一事务。

自动客户机重新路由功能的主要目标是使 DB2 UDB 客户机应用程序能够恢复通信，以便应用程序可以继续其工作，并将中断时间减至最少。如名称所示，重新路由的主要功能在于其支持连续操作。但是只有存在对客户机连接标识的备用位置时，才能进行重新路由。

可在以下可配置环境中使用自动客户机重新路由功能:

1. 具有数据分区功能（DPF）的企业服务器版（ESE）
2. Data Propagator（DPROPR）样式复制
3. 高可用性群集多处理器（HACMP）
4. 高可用性灾难恢复（HADR）。

   自动客户机重新路由与 HADR 配合工作，允许客户机应用程序在将正在存取的数据库故障转移之后继续其工作，并将中断时间减至最少。

要使 DB2 UDB 能够恢复通信，必须在通信丢失之前指定备用服务器位置。可以使用 UPDATE ALTERNATE SERVER FOR DATABASE 命令指定备用服务器。为了确保指定的备用服务器位置适用于所有客户机，必须在服务器端指定备用服务器位置。如果在客户机实例中设置它，则会忽略备用服务器。

例如，假定数据库位于名为 “N1” 的节点处（主机名为 XXX，端口号为 YYY）。数据库管理员希望将备用服务器位置设置为主机名 = AAA，端口号为 123。以下是数据库管理员要在节点 N1 处（位于服务器实例上）运行的命令:

```
db2 update alternate server for database db2 using hostname AAA port 123
```

数据库管理员在服务器实例的特定数据库上指定备用服务器位置之后，该备用服务器位置在连接时会被返回到客户机。如果通信因某种原因而丢失，DB2 UDB 客户机代码将可以使用从服务器返回的备用服务器信息重新建立连接。如果未在服务器上指定备用服务器，则不会处理自动客户机重新路由。

通常情况下，如果指定备用服务器，当检测到通信错误（sqlcode -30081）或 sqlcode -1224 时会启用自动客户机重新路由。但是，在高可用性灾难恢复（HADR）环境中，如果从 HADR 备用服务器返回 sqlcode -1776，则也会启用自动客户机重新路由。

相关概念:

- 第 274 页的『自动客户机重新路由限制』

相关参考:

- 第 275 页的『自动客户机重新路由示例』

自动客户机重新路由限制

对使用自动客户机重新路由功能存在以下限制:

- 只有当用于连接 DB2® 通用数据库（DB2 UDB）服务器或 DB2 Connect™ 服务器的通信协议为 TCP/IP 时才支持自动客户机重新路由。这意味着，如果连接使用 TCP/IP 之外的不同协议，将不会启用自动客户机重新路由功能。即使设置 DB2 UDB 以便进行回送，仍然必须使用 TCP/IP 通信协议以适应自动客户机重新路由功能。
- 如果在备用服务器位置重新建立连接，相同数据库别名的任何新连接将连接至备用服务器位置。如果要在原始位置上的问题解决之后建立与原始位置的任何新连接，下列几个选项可供选择:
  - 需要将备用服务器脱机并允许连接转移回原始服务器。（这假定已使用 UPDATE ALTERNATE SERVER 命令对原始服务器进行了编目，以便将其设置为备用服务器的备用位置。）
  - 可以对新连接要使用的新数据库别名进行编目。
  - 可以对数据库条目取消进行编目，然后再次对其进行编目。
- DB2 UDB Linux 版、UNIX® 版和 Windows® 操作系统版将支持客户机端和服务器端中的自动客户机重新路由。其它 DB2 UDB 系列当前不支持此功能。
- DB2 UDB z/OS™ 版上的数据共享 SYSPLEX 设置可能返回可用于连接的服务器列表。但是，当通信故障发生时，列表仅保留在内存中。此时，DB2 UDB 客户机将使用此列表确定用于连接的相应备用服务器的位置。
- 当与原始主机服务器上安装的 DB2 UDB 比较时，备用主机服务器中安装的 DB2 UDB 服务器的版本必须相同（但是可以具有更高版本的修订包）。
- 无论您是否具有在客户机上更新数据库目录的权限，备用服务器信息始终保留在内存中。换而言之，如果没有更新数据库目录的权限（或者因为它是只读数据库目录），其它应用程序将不能确定和使用备用服务器，原因是内存未在应用程序之间共享。
- 相同的认证应用于所有备用位置。这意味着，如果备用位置的认证类型与原始位置的认证类型不同，客户机将无法重新建立数据库连接。
- 当发生通信故障时，所有会话资源（如用于联合处理和专用寄存器的全局临时表、标识、顺序、游标、服务器选项（SET SERVER OPTION））均会丢失。应用程序负责重新建立会话资源以便继续处理工作。因为 DB2 UDB 重新播放在通信错误之前发出的专用寄存器语句，因此重新建立连接之后，无需运行任何专用寄存器语句。但是，某些专用寄存器不能重新播放。这些语句是:
  - SET ENCRYPTPW
  - SET EVENT MONITOR STATE

- SET SESSION AUTHORIZATION
- SET TRANSFORM GROUP

注: 如果客户机正在使用 CLI、JCC 类型 2 或类型 4 驱动程序，对于那些已对原始服务器预编译的 SQL 语句而言，重新建立连接之后，通过新服务器重新隐式预编译这些语句。但是，对于嵌入式 SQL 例程（例如，SQC 或 SQX 应用程序），不会重新预编译这些语句。

执行自动客户机重新路由的另一种方法是使用 DNS 条目为 DNS 条目指定备用 IP 地址。其思想是在 DNS 条目中指定第二个 IP 地址（备用服务器位置），客户机虽然不了解备用服务器，但在连接时 DB2 UDB 将在此 DNS 条目的 IP 地址之间进行切换。

相关概念:

- 第 65 页的『自动客户机重新路由实现』
- 第 273 页的『自动客户机重新路由的描述和设置』

相关任务:

- 第 62 页的『为数据库标识备用服务器』

相关参考:

- 第 275 页的『自动客户机重新路由示例』

自动客户机重新路由示例

以下是客户机应用程序的自动客户机重新路由示例（显示仅使用伪码）：

```
int checkpoint = 0;

check_sqlca(unsigned char *str, struct sqlca *sqlca)
{
   if (sqlca->sqlcode == -30081)
   {
      // as communication is lost, terminate the application right away
      exit(1);
   }
   else
   {
      // print out the error
      printf(...);

 if (sqlca->sqlcode == -30108)
 {
    // connection is re-established, re-execute the failed transaction
         if (checkpoint == 0)
         {
             goto checkpt0;
         }
    else if (checkpoint == 1)
         {
            goto checkpt1;
         }
         else if (checkpoint == 2)
         {
             goto checkpt2;
         }
         ....
         exit;
 }
   }
```

```
        }

main( )
        {
           connect to mydb;
           check_sqlca("connect failed", &sqlca);

        checkpt0:
           EXEC SQL set current schema XXX;
           check_sqlca("set current schema XXX failed", &sqlca);

           EXEC SQL create table t1...;
           check_sqlca("create table t1 failed", &sqlca);

           EXEC SQL commit;
           check_sqlca("commit failed", &sqlca);

           if (sqlca.sqlcode == 0)
           {
              checkpoint = 1;
           }

        checkpt1:
           EXEC SQL set current schema YYY;
           check_sqlca("set current schema YYY failed", &sqlca);

           EXEC SQL create table t2...;
           check_sqlca("create table t2 failed", &sqlca);

           EXEC SQL commit;
           check_sqlca("commit failed", &sqlca);

           if (sqlca.sqlcode == 0)
           {
              checkpoint = 2;
           }
        ...
        }
```

在客户机处对名为“mydb”的数据库进行编目，此数据库引用节点“hornet”，其中“hornet”也在节点目录（主机名为“hornet”，端口号为 456）中进行编目。

示例 1（涉及非 HADR 数据库）

在服务器“hornet”（主机名为 hornet，并且带有端口号）处，创建数据库“mydb”。而且，还在备用服务器（主机名为“montero”，端口号为 456）处创建数据库“mydb”。您还需要在服务器“hornet”处为数据库“mydb”更新备用服务器，如下所示:

```
db2 update alternate server for database mydb using hostname montero port 456
```

在上述样本应用程序中，未设置自动客户机重新路由功能，如果 `create table t1` 语句中存在通信错误，应用程序将被终止。设置自动客户机重新路由功能之后，DB2 UDB 将尝试再次建立与主机“hornet”（端口为 456）的连接。如果仍然无法工作，DB2 UDB 将尝试备用服务器位置（主机名为“montero”，端口为 456）。假定连接至备用服务器位置时没有出现通信错误，则应用程序可以继续运行后面的语句（并重新运行发生故障的事务）。

示例 2（涉及 HADR 数据库）

在服务器“hornet”（主机名为 hornet，并且带有端口号）处，创建主数据库“mydb”。另外还在端口为 456 的主机“montero”处创建辅助数据库。有关如何为主数据库和辅

助数据库设置 HADR 的信息可于《*数据恢复及高可用性指南与参考*》中找到。您还需要为数据库 “mydb” 更新备用服务器，如下所示:

```
db2 update alternate server for database mydb using hostname montero port 456
```

在上述样本应用程序中，未设置自动客户机重新路由功能，如果 `create table t1` 语句中存在通信错误，应用程序将被终止。设置自动客户机重新路由功能之后，DB2 UDB 将尝试再次建立与主机 “hornet”（端口为 456）的连接。如果仍然无法工作，DB2 UDB 将尝试备用服务器位置（主机名为 “montero”，端口为 456）。假定连接至备用服务器位置时没有出现通信错误，则应用程序可以继续运行后面的语句（并重新运行发生故障的事务）。

相关概念:

- 第 273 页的『自动客户机重新路由的描述和设置』

相关任务:

- 第 62 页的『为数据库标识备用服务器』

附录 C. 使用“轻量级目录访问协议”（LDAP）目录服务

“轻量级目录访问协议”（LDAP）简介

“轻量级目录访问协议”（LDAP）是一个业界标准的访问目录服务的方法。目录服务是一个关于分布式环境中的多个系统和服务的资源信息的资源库；它提供对这些资源的客户机和服务器访问。每个数据库服务器实例都会将它的存在情况发布给 LDAP 服务器，并在创建数据库时向 LDAP 目录提供数据库信息。当客户机与数据库连接后，可从 LDAP 目录检索服务器的目录信息。不再要求每个客户机将目录信息以本地方式存储在每台机器上。客户机应用程序搜索 LDAP 目录，查找连接数据库所需的信息。

由于存在高速缓存机制，因此客户机仅需要搜索 LDAP 目录服务器一次。一旦从 LDAP 目录服务器中检索到信息，就根据 *dir_cache* 数据库管理器配置参数和 DB2LDAPCACHE 注册表变量的值，在本地机器上存储或高速缓存此信息。*dir_cache* 数据库管理器配置参数用于在内存高速缓存中存储数据库、节点和 DCS 目录文件。应用程序使用目录高速缓存直至应用程序关闭。DB2LDAPCACHE 注册表变量用于在本地磁盘高速缓存中存储数据库、节点和 DCS 目录文件。

- 若 DB2LDAPCACHE=NO 且 *dir_cache*=NO，则总是从 LDAP 读取信息。
- 如果 DB2LDAPCACHE=NO 且 *dir_cache*=YES，则从 LDAP 读取一次信息，并将该信息插入到 DB2® 高速缓存中。
- 如果设置或未设置 DB2LDAPCACHE=YES，则从 LDAP 中读取信息一次并将其高速缓存至本地数据库、节点和 DCS 目录中。

注：DB2LDAPCACHE 注册表变量仅适用于数据库和节点目录。

相关概念:

- 第 63 页的『“轻量级目录访问协议”（LDAP）目录服务』
- 第 281 页的『对 Active Directory 的支持』
- 第 294 页的『LDAP 支持和 DB2 Connect』
- 第 294 页的『LDAP 环境中的安全性注意事项』
- 第 295 页的『Active Directory 的安全性注意事项』
- 『DB2 注册表和环境变量』（*《管理指南: 性能》*）
- 第 296 页的『扩展具有 DB2 对象类和属性的 LDAP 目录模式』

相关任务:

- 第 282 页的『配置 DB2 以使用 Active Directory』
- 第 282 页的『在 IBM LDAP 环境中配置 DB2』
- 第 283 页的『创建 LDAP 用户』
- 第 284 页的『为 DB2 应用程序配置 LDAP 用户』
- 第 284 页的『安装后注册 DB2 服务器』
- 第 286 页的『更新 DB2 服务器的协议信息』
- 第 287 页的『对节点别名进行编目以进行连接（ATTACH）』

- 第 287 页的『注销 DB2 服务器』
- 第 288 页的『在 LDAP 目录中注册数据库』
- 第 288 页的『在 LDAP 环境中连接至远程服务器』
- 第 289 页的『从 LDAP 目录中注销数据库』
- 第 289 页的『在本地数据库和节点目录中刷新 LDAP 条目』
- 第 290 页的『搜索 LDAP 目录分区或域』
- 第 291 页的『在 LDAP 中注册主机数据库』
- 第 292 页的『在 LDAP 环境中设置用户级别的 DB2 注册表变量』
- 第 293 页的『在安装完成后启用 LDAP 支持』
- 第 294 页的『禁用 LDAP 支持』
- 第 296 页的『扩展 Active Directory 的目录模式』

相关参考:

- 第 280 页的『受支持的 LDAP 客户机和服务器配置』
- 第 298 页的『Active Directory 中的 DB2 对象』
- 第 305 页的『DB2 使用的 LDAP 对象类和属性』
- 第 298 页的『Netscape LDAP 目录支持和属性定义』

受支持的 LDAP 客户机和服务器配置

下表概述受支持的 LDAP 客户机和服务器配置:

表 22. 受支持的 LDAP 客户机和服务器配置

	IBM SecureWay Directory	Microsoft Active Directory	Netscape LDAP 服务器
IBM LDAP 客户机	受支持	受支持	受支持
Microsoft LDAP/ADSI 客户机	受支持	受支持	受支持

IBM SecureWay Directory V3.1 是可用于 Windows NT、AIX、Solaris Operating Environment 和 HP-UX 的 LDAP V3 服务器。在 AIX 和 iSeries（AS/400）上，SecureWay Directory 与 “OS/390 安全性服务器” 一起，作为基本操作系统的一部分交付。

32 位版本的 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）支持 AIX、Solaris Operating Environment、HP-UX 11.11、Windows、Linux IA32 和 Linux/390 上的 IBM LDAP 客户机。

Microsoft Active Directory 是一个 LDAP 版本 3 服务器，可作为 “Windows 2000 服务器” 操作系统的一部分获得。

Microsoft LDAP Client 是 Windows 操作系统附带包括的。

当在 Windows 操作系统上运行时，DB2 支持使用 IBM LDAP 客户机或 Microsoft LDAP Client 来访问 “IBM SecureWay Directory 服务器”。要显式地选择 IBM LDAP 客户机，使用 **db2set** 命令来将 DB2LDAP_CLIENT_PROVIDER 注册表变量设置为 “IBM”。

相关概念:

- 第 279 页的『“轻量级目录访问协议”（LDAP）简介』
- 第 281 页的『对 Active Directory 的支持』

对 Active Directory 的支持

DB2® 通用数据库（DB2 UDB）按如下所示使用 Active Directory:

1. DB2 数据库服务器作为 ibm_db2Node 对象发布在 Active Directory 中。ibm_db2Node 对象类是 ServiceConnectionPoint(SCP) 对象类的子类。每个 ibm_db2Node 对象都包含协议配置信息以允许客户机应用程序连接至 DB2 UDB 数据库服务器。创建新数据库时，在 Active Directory 中，将该数据库作为 ibm_db2Database 对象发布在 ibm_db2Node 对象下面。
2. 当连接至远程数据库时，DB2 客户机通过 LDAP 接口查询 Active Directory，以查找 ibm_db2Database 对象。用于连接数据库服务器的协议通信（绑定信息）是从 ibm_db2Node 对象获取的，ibm_db2Database 对象在该对象下面创建。

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

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

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

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

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

相关概念:

- 第 295 页的『Active Directory 的安全性注意事项』

相关任务:

- 第 282 页的『配置 DB2 以使用 Active Directory』
- 第 296 页的『扩展 Active Directory 的目录模式』

相关参考:

- 第 298 页的『Active Directory 中的 DB2 对象』

配置 DB2 以使用 Active Directory

过程:

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

1. 运行 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）的机器必须属于 Windows 2000 或 Windows Server 2003 域。
2. 已安装 Microsoft LDAP 客户机。Microsoft® LDAP 客户机是 Windows 2000、Windows XP 和 Windows Server 2003 操作系统的一部分。对于 Windows 98、Windows NT 或 Windows Me，需要验证 Active Directory 客户机扩展（`wldap32.dll`）是否存在于系统目录之下。
3. 启用 LDAP 支持。对于 Windows 2000、Windows XP 或 Windows Server 2003，通过安装程序可以启用 LDAP 支持。对于 Windows 98、Windows NT 或 Windows Me，必须使用 **db2set** 命令将 DB2_ENABLE_LDAP 注册表变量设置为 “YES” 来显式启用 LDAP。
4. 当运行 DB2 UDB 时登录域用户帐户以便从 Active Directory 读取信息。

相关概念:

- 第 281 页的『对 Active Directory 的支持』
- 『DB2 注册表和环境变量』（*《管理指南: 性能》*）

相关任务:

- 第 284 页的『为 DB2 应用程序配置 LDAP 用户』

在 IBM LDAP 环境中配置 DB2

过程:

在可以在 IBM LDAP 环境中使用 DB2 之前，必须在每台机器上配置下列各项:

- 启用 LDAP 支持。对于 Windows 2000，LDAP 支持是通过安装程序启用的。对于 Windows 98 或 Windows NT，必须使用 **db2set** 命令将 DB2_ENABLE_LDAP 注册表变量设置为 “YES” 来显式启用 LDAP。在所有 Windows 操作系统上使用的缺省 LDAP 客户机是 Microsoft 的客户机。如果要使用 IBM LDAP 客户机，必须使用 **db2set** 命令将 DB2LDAP_CLIENT_PROVIDER 注册表变量设置为 “IBM”。
- LDAP 服务器的 TCP/IP 主机名和端口号。这些值可以在无人照管安装期间使用 DB2LDAPHOST 响应关键字输入，也可以在以后使用 DB2SET 命令手工设置它们:

  ```
  db2set DB2LDAPHOST=<hostname[:port]>
  ```

  其中 `hostname` 是 LDAP 服务器的 TCP/IP 主机名，而 `[:port]` 是端口号。若未指定端口号，则 DB2 将使用缺省 LDAP 端口（389）。

  DB2 对象位于 LDAP 基本专有名称（baseDN）中。若是在使用 IBM SecureWay LDAP 目录服务器版本 3.1，则不必配置基本专有名称，因为 DB2 可以动态地从服务器获取此信息。但是，若是在使用 “IBM eNetwork 目录服务器” 版本 2.1，则必须使用 DB2SET 命令在每台机器上配置 LDAP 基本专有名称:

  ```
  db2set DB2LDAP_BASEDN=<baseDN>
  ```

其中 baseDN 是 LDAP 服务器上定义的 LDAP 后缀的名称。此 LDAP 后缀用来包含 DB2 对象。

- LDAP 用户的专有名称（DN）和密码。仅当您计划使用 LDAP 来存储 DB2 用户特定信息时，这些信息才是必需的。

相关概念:

- 『DB2 注册表和环境变量』（*《管理指南: 性能》*）

相关任务:

- 第 282 页的『配置 DB2 以使用 Active Directory』
- 第 283 页的『创建 LDAP 用户』

相关参考:

- 『db2set - DB2 Profile Registry Command』（*Command Reference*）

创建 LDAP 用户

过程:

DB2 支持在用户级别设置 DB2 注册表变量和 CLI 配置。（在 AIX 和 Solaris 平台上，此功能不可用。）用户级别支持在多用户环境中提供了用户特定设置。 “Windows NT 终端服务器”便是一个示例，每个登录用户都可以定制他或她自己的环境，而不会干扰系统环境或另一用户的环境。

当使用 IBM LDAP 目录时，必须定义 LDAP 用户，然后才能在 LDAP 中存储用户级别信息。可以使用下列其中一种方法创建 LDAP 用户:

- 创建一个 LDIF 文件来包含用户对象的所有属性，然后运行 LDIF 导入实用程序来将该对象导入到 LDAP 目录中。用于 IBM LDAP 服务器的 LDIF 实用程序是“LDIF2DB”。
- 使用“目录管理工具”（DMT）（仅可用于“IBM SecureWay LDAP 目录服务器”版本 3.1）来创建用户对象。

包含人员对象属性的 LDIF 文件外观类似于:

文件名: newuser.ldif

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

以下是使用 IBM LDIF 导入实用程序导入 LDIF 文件的 LDIF 命令的一个示例:

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

注:

1. 必须从 LDAP 服务器运行 LDIF2DB 命令。

2. 必须将必需的存取权（ACL）授予 LDAP 用户对象，以使 LDAP 用户可以添加、删除、读取和写入他自己的对象。要授予用户对象的 ACL，使用 “LDAP 目录服务器 Web 管理” 工具。

相关任务:

- 第 282 页的『在 IBM LDAP 环境中配置 DB2』
- 第 284 页的『为 DB2 应用程序配置 LDAP 用户』

为 DB2 应用程序配置 LDAP 用户

过程:

使用 Microsoft LDAP Client 时，LDAP 用户与操作系统用户帐户相同。但是，使用 IBM LDAP 客户机时，在使用 DB2 之前，必须配置当前登录用户的 LDAP 用户专有名称（DN）和密码。这可以使用 db2ldcfg 实用程序完成:

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

例如:

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

相关任务:

- 第 283 页的『创建 LDAP 用户』

相关参考:

- 『db2ldcfg - Configure LDAP Environment Command』（*Command Reference*）

安装后注册 DB2 服务器

过程:

必须在 LDAP 中注册每个 DB2 服务器实例，以便发布客户机应用程序用来连接 DB2 服务器实例的协议配置信息。当注册一个数据库服务器实例时，需要指定一个节点名。该节点名由客户机应用程序在连接服务器时使用。可使用 **CATALOG LDAP NODE** 命令对 LDAP 节点的另一别名进行编目。

注: 若是在 Windows 2000 域环境中工作，则安装期间，将在 Active Directory 中用下列信息自动注册 DB2 服务器实例:

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

若 TCP/IP 主机名长于 8 个字符，则它将被截断为 8 个字符。

REGISTER 命令如下所示:

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

`protocol` 子句指定与此数据库服务器连接时要使用的通信协议。

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

注: 每台机器不能使用相同的 ldap_node_name，因为在 LDAP 中该名称必须是唯一的。以每台机器的主机名替换 **REGISTER** 命令中的 ldap_node_name。例如:

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

"<>" 被替换为在其上运行 **rah** 命令的每台机器的主机名。如果出现有多个 DB2 通用数据库企业服务器版实例这种极少见的情况，可将实例与主机索引的组合用作 **rah** 命令中的节点名。

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

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

以下是计算机名称的约定:

- 若配置了 TCP/IP，则计算机名称必须与 TCP/IP 主机名相同。
- 若配置了 APPN，则使用伙伴 LU 名作为计算机名称。

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

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

其中 n.nn.nn.nn 是群集 IP 地址。

相关概念:

- 第 317 页的『rah 和 db2_all 命令概述』

相关任务:

- 第 286 页的『更新 DB2 服务器的协议信息』
- 第 287 页的『对节点别名进行编目以进行连接（ATTACH）』
- 第 287 页的『注销 DB2 服务器』
- 第 288 页的『在 LDAP 环境中连接至远程服务器』

相关参考:

- 『REGISTER Command』（*Command Reference*）
- 『CATALOG LDAP NODE Command』（*Command Reference*）

更新 DB2 服务器的协议信息

过程:

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

要更新本地机器上的 LDAP 中的 DB2 服务器，使用以下命令:

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

可更新的协议配置参数的示例包括:

- TCP/IP 主机名和服务名称或端口号参数。
- APPC 协议信息，如 TP 名、伙伴 LU 或方式。
- NetBIOS 工作站名。

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

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

相关任务:

- 第 284 页的『安装后注册 DB2 服务器』
- 第 287 页的『对节点别名进行编目以进行连接（ATTACH）』
- 第 288 页的『在 LDAP 环境中连接至远程服务器』

相关参考:

- 『UPDATE LDAP NODE Command』（*Command Reference*）

将 LDAP 客户机重新路由至另一服务器

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

先决条件:

DB2_ENABLE_LDAP 注册表变量设置为 “Yes”。

过程:

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

使用 UPDATE ALTERNATE SERVER FOR LDAP DATABASE 命令为 LDAP 中代表 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）的数据库定义备用服务器。

一旦建立，在连接时会将此备用服务器信息返回至客户机。

注: 在服务器实例输入 UPDATE ALTERNATE SERVER FOR DATABASE 命令（注意，不是 “FOR LDAP DATABASE” ）时，如果在服务器上启用 LDAP 支持（DB2_ENABLE_LDAP=Yes），且高速缓存 LDAP 用户标识和密码（先前已运行 **db2ldcfg**），则在 LDAP 服务器上自动或隐式更新数据库的备用服务器。这与显式输入 UPDATE ALTERNATE SERVER FOR LDAP DATABASE 的效果相同。

相关概念:

- 第 65 页的『自动客户机重新路由实现』
- 第 273 页的『自动客户机重新路由的描述和设置』

对节点别名进行编目以进行连接（ATTACH）

过程:

当在 LDAP 中注册服务器时，必须指定 DB2 服务器的节点名。应用程序使用该节点名连接数据库服务器。若需要另一个节点名，如在将节点名硬编码进应用程序中时，可使用 CATALOG LDAP NODE 命令进行更改。该命令类似于:

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

要取消对 LDAP 节点进行编目，使用 UNCATALOG LDAP NODE COMMAND 命令。该命令类似于:

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

相关任务:

- 第 284 页的『安装后注册 DB2 服务器』
- 第 288 页的『在 LDAP 环境中连接至远程服务器』

相关参考:

- 『CATALOG LDAP NODE Command』（*Command Reference*）
- 『UNCATALOG LDAP NODE Command』（*Command Reference*）

注销 DB2 服务器

过程:

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

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

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

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

相关任务:

- 第 284 页的『安装后注册 DB2 服务器』

相关参考:

- 『DEREGISTER Command』（*Command Reference*）

在 LDAP 目录中注册数据库

过程:

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

若在 LDAP 目录中已存在该名称，仍然会在本地机器上创建该数据库，但是会返回一个警告消息，说明在 LDAP 目录中发生名称冲突。因此，可在 LDAP 目录中手工对数据库进行编目。用户可使用 CATALOG LDAP DATABASE 命令在 LDAP 中注册远程服务器上的数据库。当注册远程数据库时，指定表示远程数据库服务器的 LDAP 节点的名称。在注册数据库**之前**，**必须**使用 REGISTER DB2 SERVER IN LDAP 命令在 LDAP 中注册远程数据库服务器。

要在 LDAP 中手工注册数据库，使用 CATALOG LDAP DATABASE 命令:

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

相关任务:

- 第 284 页的『安装后注册 DB2 服务器』
- 第 289 页的『从 LDAP 目录中注销数据库』

相关参考:

- 『CATALOG LDAP DATABASE Command』（*Command Reference*）

在 LDAP 环境中连接至远程服务器

过程:

在 LDAP 环境中，可在 ATTACH 命令中使用 LDAP 节点名连接远程数据库服务器:

```
db2 attach to <ldap_node_name>
```

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

详言之，由于存在高速缓存机制，因此客户机仅搜索 LDAP 服务器一次。一旦检索到信息，就根据 *dir_cache* 数据库管理器配置参数和 DB2LDAPCACHE 注册表变量的值，在本地机器上存储或高速缓存此信息。

- 若 DB2LDAPCACHE=NO 且 *dir_cache*=NO，则总是从 LDAP 读取信息。

- 若 DB2LDAPCACHE=NO 且 *dir_cache*=YES，则从 LDAP 读取一次信息，并将该信息插入 DB2 高速缓存。
- 如果设置或未设置 DB2LDAPCACHE=YES，则从 LDAP 服务器中读取信息一次并将其高速缓存至本地数据库、节点和 DCS 目录中。

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

相关概念:

- 『DB2 注册表和环境变量』（*《管理指南：性能》*）

相关任务:

- 第 284 页的『安装后注册 DB2 服务器』
- 第 286 页的『更新 DB2 服务器的协议信息』
- 第 287 页的『对节点别名进行编目以进行连接（ATTACH）』
- 第 288 页的『在 LDAP 目录中注册数据库』

相关参考:

- 『ATTACH Command』（*Command Reference*）

从 LDAP 目录中注销数据库

过程:

当发生下列情况时，就自动从 LDAP 注销数据库:

- 删除数据库。
- 从 LDAP 中注销自己的实例。

可使用以下命令从 LDAP 中手工注销数据库:

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

相关任务:

- 第 288 页的『在 LDAP 目录中注册数据库』

相关参考:

- 『UNCATALOG LDAP DATABASE Command』（*Command Reference*）

在本地数据库和节点目录中刷新 LDAP 条目

过程:

LDAP 信息可能会更改，因此有必要刷新本地目录和节点目录中的 LDAP 条目。本地数据库和节点目录用来高速缓存 LDAP 中的条目。

详言之，由于存在高速缓存机制，因此客户机仅搜索 LDAP 服务器一次。一旦检索到信息，就根据 *dir_cache* 数据库管理器配置参数和 DB2LDAPCACHE 注册表变量的值，在本地机器上存储或高速缓存此信息。

- 若 DB2LDAPCACHE=NO 且 *dir_cache*=NO，则总是从 LDAP 读取信息。

- 若 DB2LDAPCACHE=NO 且 *dir_cache*=YES，则从 LDAP 读取一次信息，并将该信息插入 DB2 高速缓存。
- 如果设置或未设置 DB2LDAPCACHE=YES，则从 LDAP 服务器中读取信息一次并将其高速缓存至本地数据库、节点和 DCS 目录中。

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

要刷新引用 LDAP 资源的数据库条目，使用以下命令：

```
db2 refresh ldap database directory
```

要刷新本地机器上引用 LDAP 资源的节点条目，使用以下命令：

```
db2 refresh ldap node directory
```

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

要确保用一种快速的方法完成刷新，可以这样做：

- 安排定期运行的刷新。
- 在系统启动期间运行 REFRESH 命令。
- 在所有客户机上使用一个可用的管理程序包来调用 REFRESH 命令。
- 设置 DB2LDAPCACHE="NO" 以避免将 LDAP 信息高速缓存在数据库、节点和 DCS 目录中。

相关概念：

- 『DB2 注册表和环境变量』（*《管理指南：性能》*）

相关参考：

- 『dir_cache - 目录高速缓存支持配置参数』（*《管理指南：性能》*）
- 『REFRESH LDAP Command』（*Command Reference*）

搜索 LDAP 目录分区或域

过程：

DB2 UDB 在 Windows 2000 环境中搜索当前 LDAP 目录分区或当前 Active Directory 域。在有多个 LDAP 目录分区或域的环境中，可设置搜索范围。例如，若在当前的分区或域中找不到信息，可请求自动搜索所有其它分区或域。另一方面，可限制搜索范围仅搜索本地机器。

搜索范围由 DB2 UDB 概要文件注册表变量 DB2LDAP_SEARCH_SCOPE 控制。要设置 LDAP 中的全局级别搜索范围值，在 *db2set* 命令中使用 "-gl" 选项，该选项表示 "LDAP 中的全局"：

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

可能的值包括： "local"、"domain" 或 "global"。缺省值是 "domain"，它将搜索范围限制在当前目录分区。可在 LDAP 中设置搜索范围，以作为整个企业的缺省搜索范围设置。例如，可能需要在创建新的数据库后将搜索范围初始化为 "global"。这就允许所有的客户机搜索所有其它分区或域，以查找在特定的分区或域中定义的数据库。

在每台客户机首次连接之后，一旦在每台机器上记录了该条目，搜索范围可更改为“local”。一旦更改为“local”，每台客户机将不扫描任何分区或域。

注：DB2 UDB 概要文件注册表变量 DB2LDAP_KEEP_CONNECTION 和 DB2LDAP_SEARCH_SCOPE 是唯一支持在 LDAP 中的全局级别设置该变量的注册表变量。

相关概念:

- 『DB2 注册表和环境变量』（*《管理指南: 性能》*）

相关任务:

- 第 26 页的『声明注册表和环境变量』

在 LDAP 中注册主机数据库

过程:

当在 LDAP 中注册主机数据库时，有两个可能的配置:

- 直接连接主机数据库；或者
- 通过网关连接主机数据库。

在第一种情况下，用户将在 LDAP 中注册主机服务器，然后在 LDAP 中对主机数据库进行编目，并指定主机服务器的节点名。在第二种情况下，用户应在 LDAP 中注册网关服务器，然后在 LDAP 中对主机数据库进行编目，并指定网关服务器的节点名。

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

作为显示上述两种情况的示例，请考虑: 假设有一个名为 NIAGARA_FALLS 的主机数据库。它可接受使用 APPN 和 TCP/IP 的入局连接。若客户机因没有 DB2 Connect 而不能直接连接主机，则它将使用名为“goto@niagara”的网关进行连接。

需要完成下列步骤:

1. 在 LDAP 中为 APPN 连接注册主机数据库服务器。REMOTE 和 INSTANCE 子句是任意的。将 NODETYPE 子句设置为“DCS”，指示这是主机数据库服务器。

   ```
   db2 register ldap as nfappn appn network CAIBMOML partnerlu NFLU
      mode IBMRDB remote mvssys instance msvinst nodetype dcs
   ```

2. 在 LDAP 中为 TCP/IP 连接注册主机数据库服务器。服务器的 TCP/IP 主机名是“myhost”，端口号是“446”。与步骤 1 类似，将 NODETYPE 子句设置为“DCS”，以指示这是主机数据库服务器。

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

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

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

4. 使用 TCP/IP 连接来在 LDAP 中对主机数据库进行编目。主机数据库名称是“NIAGARA_FALLS”，数据库别名是“nftcpip”。使用 GWNODE 子句来指定 DB2 Connect 网关服务器的节点名。

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

5. 使用 APPN 连接来在 LDAP 中对主机数据库进行编目。

```
db2 catalog ldap database NIAGARA_FALLS as nfappn at node nfappn
   gwnode whasf authentication dcs
```

在完成上面显示的注册和编目之后，若要使用 TCPIP 连接主机，则连接 “nftcpip”。若要使用 APPN 连接主机，则连接至 “nfappn”。若客户机工作站上没有 DB2 Connect，则该连接将使用 TCPIP 来通过网关，并且根据您使用的是 “nftcpip” 还是 “nfappn” 分别使用 TCP/IP 或 APPN 从那里来与主机连接。

之后，通常可在 LDAP 中手工配置主机数据库信息，以便每台客户机不必在自己的机器上以本地方式手工对该数据库和节点进行编目。该过程如下:

1. 在 LDAP 中注册主机数据库服务器。在 REGISTER 命令中，必须使用 REMOTE、INSTANCE 和 NODETYPE 子句分别指定远程计算机名称、实例名和主机数据库服务器的节点类型。REMOTE 子句可以设置为主机名或主机服务器的 LU 名。INSTANCE 子句可以设置为任何长度不超过 8 个字符的字符串。(例如，可以将实例名设置为 “DB2”。) 必须将 NODE TYPE 子句设置为 “DCS”，以指示这是主机数据库服务器。
2. 使用 CATALOG LDAP DATABASE 命令在 LDAP 中注册主机数据库。可使用 PARMS 参数指定任何附加的 DRDA 子句。应将数据库认证类型设置为 “DCS”。

相关参考:

- 『REGISTER Command』 (*Command Reference*)
- 『CATALOG LDAP DATABASE Command』 (*Command Reference*)

在 LDAP 环境中设置用户级别的 DB2 注册表变量

过程:

在 LDAP 环境中，可在用户级别设置 DB2 概要文件注册表变量，这样可允许用户定制自己的 DB2 环境。要在用户级别设置 DB2 概要文件注册表变量，使用 -ul 选项:

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

注: 这在 AIX 或 Solaris Operating Environment 上不受支持。

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

- 更新或重新设置用户级别 DB2 注册表变量。
- 在用户级别刷新 LDAP 概要文件变量的命令是:

```
db2set -ur
```

相关任务:

- 第 26 页的『声明注册表和环境变量』

相关参考:

- 『db2set - DB2 Profile Registry Command』 (*Command Reference*)

在安装完成后启用 LDAP 支持

过程:

要在安装过程完成后启用 LDAP 支持，在每台机器上使用以下过程:

- 安装 LDAP 支持二进制文件。运行安装程序并从定制安装中选择 “LDAP 目录开发” 支持。安装程序安装二进制文件并将 DB2 概要文件注册表变量 DB2_ENABLE_LDAP 设置为 “YES”。

  注: 对于 Windows 98、Windows NT 和 UNIX 平台，必须使用 **db2set** 命令将 DB2_ENABLE_LDAP 注册表变量设置为 “YES” 来显式启用 LDAP。

- （仅在 UNIX 平台上）使用以下命令说明 LDAP 服务器的 TCP/IP 主机名和（可选的）端口号:

  ```
  db2set DB2LDAPHOST=<base_domain_name>[:port_number]
  ```

  其中 base_domain_name 是 LDAP 服务器的 TCP/IP 主机名，而 [:port] 是端口号。若未指定端口号，则 DB2 将使用缺省 LDAP 端口（389）。

  DB2 对象位于 LDAP 基本专有名称（baseDN）中。若是在使用 IBM SecureWay LDAP 目录服务器版本 3.1，则不必配置基本专有名称，因为 DB2 可以动态地从服务器获取此信息。但是，若是在使用 “IBM eNetwork 目录服务器” 版本 2.1，则必须使用 DB2SET 命令在每台机器上配置 LDAP 基本专有名称:

  ```
  db2set DB2LDAP_BASEDN=<baseDN>
  ```

  其中 baseDN 是 LDAP 服务器上定义的 LDAP 后缀的名称。此 LDAP 后缀用来包含 DB2 对象。

- 使用 REGISTER LDAP AS 命令在 LDAP 中注册 DB2 服务器的当前实例。例如:

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

- 若有要在 LDAP 中注册的数据库，运行 CATALOG LDAP DATABASE 命令。例如:

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

- 输入 LDAP 用户的专有名称（DN）和密码。仅当您计划使用 LDAP 来存储 DB2 用户特定信息时，这些信息才是必需的。

相关概念:

- 『DB2 注册表和环境变量』（*《管理指南: 性能》*）

相关任务:

- 第 294 页的『禁用 LDAP 支持』

相关参考:

- 『REGISTER Command』（*Command Reference*）
- 『db2set - DB2 Profile Registry Command』（*Command Reference*）
- 『CATALOG LDAP DATABASE Command』（*Command Reference*）

禁用 LDAP 支持

过程:

要禁用 LDAP 支持，使用下列过程:

- 对每个 DB2 服务器实例，注销 LDAP 中的 DB2 服务器:

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

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

相关任务:

- 第 26 页的『声明注册表和环境变量』
- 第 293 页的『在安装完成后启用 LDAP 支持』

相关参考:

- 『DEREGISTER Command』（*Command Reference*）

LDAP 支持和 DB2 Connect

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

相关概念:

- 第 279 页的『“轻量级目录访问协议”（LDAP）简介』
- 第 294 页的『LDAP 环境中的安全性注意事项』

LDAP 环境中的安全性注意事项

在存取 LDAP 目录中的信息之前，LDAP 服务器要认证应用程序或用户。认证过程已*绑定*至 LDAP 服务器。

对存储在 LDAP 目录中的信息进行存取控制很重要，这样可以防止匿名用户添加、删除或修改信息。

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

缺省情况下，按如下所示定义存取控制:

- 对于 LDAP 中的数据库条目和节点条目，每个人（或任何匿名用户）都有读存取权。只有目录管理员以及对象的所有者或创建者具有读 / 写存取权。
- 对于用户概要文件，概要文件所有者和目录管理员具有读 / 写存取权。若一个用户没有“目录管理员”权限，则不能存取另一个用户的概要文件。

注: 权限检查总是由 LDAP 服务器执行，而不是由 DB2® 执行。LDAP 权限检查与 DB2 权限无关。具有 SYSADM 权限的帐户或授权标识也许不能存取 LDAP 目录。

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

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

相关概念:

• 第 295 页的『Active Directory 的安全性注意事项』

Active Directory 的安全性注意事项

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

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

1. 启动 *Active Directory 用户和计算机*管理工具

   （开始 -> 程序 -> 管理工具 -> Active Directory 用户和计算机）
2. 在*查看*下面，选择*高级功能*
3. 选择*计算机*容器
4. 右键单击表示安装有 DB2 的服务器的计算机对象，并选择*属性*
5. 选择*安全性*选项卡，然后向指定的用户或组添加必需的存取权

用户级别的 DB2 注册表变量和 CLI 设置是在用户对象下面的 DB2 属性对象中维护的。要在用户级别设置 DB2 注册表变量或 CLI 设置，用户需要具有足够的存取权才能在“用户”对象下面创建对象。

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

1. 启动 *Active Directory 用户和计算机*管理工具

   （开始 -> 程序 -> 管理工具 -> Active Directory 用户和计算机）
2. 在“用户”容器下面选择该用户对象
3. 右键单击该用户对象并选择*属性*
4. 选择*安全性*选项卡
5. 使用添加“按钮”将用户名添加至列表
6. 授予“写入”和“创建所有子对象”存取权
7. 使用“高级”设置，设置许可权以应用到“此对象和所有子对象”
8. 选择复选框“允许父代的可继承许可权传播至此对象”

相关概念:

• 第 294 页的『LDAP 环境中的安全性注意事项』

扩展具有 DB2 对象类和属性的 LDAP 目录模式

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

在 DB2® 可以将信息存储到 LDAP 中之前，LDAP 服务器的“目录模式”必须包括 DB2 使用的对象类和属性。向基本模式中添加新对象类和属性的过程称作扩展“目录模式”。

注: 如果使用 IBM® SecureWay® LDAP Directory V3.1，则 DB2 UDB 版本 8.1 和更早版本需要的所有对象类和属性均包括在基本模式中。在这种情况下，不必扩展具有 DB2 对象类和属性的基本模式。但是，DB2 UDB 版本 8.2 的两个新属性未包括在基本模式中。在这种情况下，必须扩展具有这两个新 DB2 UDB 属性的基本模式。

相关概念:

- 第 300 页的『扩展 IBM SecureWay Directory Server 的目录模式』

相关任务:

- 第 296 页的『扩展 Active Directory 的目录模式』

扩展 Active Directory 的目录模式

过程:

DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）可以在 Active Directory 中存储信息之前，需要扩展目录模式以包括新的 DB2 UDB 对象类和属性。向目录模式中添加新对象类和属性的过程称作“模式扩展”。

在任何作为 Windows 域一部分的机器上首次安装 DB2 UDB 之前，必须运行“DB2 UDB 模式安装”程序 **db2schex** 以扩展 Active Directory 的模式。

db2schex 程序可以在产品 CD-ROM 上找到。此程序在 CD-ROM 上的位置在 db2 目录的 windows 子目录和 utilities 子目录下面。例如:

```
x:\db2\windows\utilities\
```

其中 x: 是 CD-ROM 驱动器。

按如下所示使用该命令:

```
db2schex
```

此命令还有其它可选子句:

- –b UserDN

  指定用户“专有名称”。

- –w Password

  指定 BIND 密码。

- –u

卸载模式。

- –k

  强制卸载继续，忽略错误。

注:

1. 若未指定 UserDN 和密码，则 **db2schex** 作为当前已登录的用户绑定。
2. 可指定 userDN 子句作为 Windows NT 用户名。
3. 要更新模式，您必须是“模式管理员”组的成员，或已被授予更新模式的权限。

需要运行 **db2schex.exe** 命令，此命令随 DB2 UDB 版本 8.2 产品附带以扩展目录模式。

如果已运行先前版本的 DB2 Windows 版产品附带的 **db2schex.exe** 命令，则再次运行 DB2 UDB 版本 8.2 附带的同一命令时，它将下列两个可选的属性添加到 ibm-db2Database 类中:

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

如果尚未运行先前版本的 DB2 UDB Windows 版产品附带的 **db2schex.exe** 命令，则运行 DB2 UDB 版本 8.2 附带的同一命令时，它将添加所有类和属性以提供 DB2 UDB LDAP 支持。

示例:

- 要安装 DB2 UDB 模式:

  ```
  db2schex
  ```

- 要安装 DB2 UDB 模式并指定绑定 DN 和密码:

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

  或者,

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

- 要卸载 DB2 UDB 模式:

  ```
  db2schex -u
  ```

- 要卸载 DB2 UDB 模式并忽略错误:

  ```
  db2schex -u -k
  ```

Active Directory 的“DB2 UDB 模式安装”程序执行下列任务:

注:

1. 检测哪一个服务器是“模式主机”
2. 绑定至作为“模式主机”的“域控制器”
3. 确保该用户有充分的权限来将类和属性添加到该模式
4. 确保模式主机可写（即，除去了注册表中的安全互锁装置）
5. 创建所有新属性
6. 创建所有新对象类
7. 检测错误，如果发生错误，则该程序将回滚对模式的任何更改

相关概念:

- 第 296 页的『扩展具有 DB2 对象类和属性的 LDAP 目录模式』

Active Directory 中的 DB2 对象

DB2 在 Active Directory 中的两个位置中创建对象:

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

相关参考:

- 第 305 页的『DB2 使用的 LDAP 对象类和属性』

Netscape LDAP 目录支持和属性定义

受支持的 Netscape LDAP 服务器级别是 v4.12 或更新版本。

在 “Netscape LDAP 服务器 V4.12” 或更新版本中， “Netscape 目录服务器” 允许应用程序通过将属性和对象类定义添加到 `slapd.user_oc.conf` 和 `slapd.user_at.conf` 两个文件来扩展模式。这两个文件位于

```
<Netscape_install path>\slapd-<machine_name>\config
```

目录中。

注: 如果正在使用 “iPlan 目录服务器 5.0”，则必须查看随该产品附带的文档，以获取有关如何扩展模式的详细指示信息。

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

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

attribute binProperty                    1.3.18.0.2.4.305     bin
attribute binPropertyType                1.3.18.0.2.4.306     cis
attribute cesProperty                    1.3.18.0.2.4.307     ces
attribute cesPropertyType                1.3.18.0.2.4.308     cis
attribute cisProperty                    1.3.18.0.2.4.309     cis
attribute cisPropertyType                1.3.18.0.2.4.310     cis
attribute propertyType                   1.3.18.0.2.4.320     cis
attribute systemName                     1.3.18.0.2.4.329     cis
attribute db2nodeName                    1.3.18.0.2.4.419     cis
attribute db2nodeAlias                   1.3.18.0.2.4.420     cis
attribute db2instanceName                1.3.18.0.2.4.428     cis
attribute db2Type                        1.3.18.0.2.4.418     cis
attribute db2databaseName                1.3.18.0.2.4.421     cis
```

```
 attribute db2databaseAlias               1.3.18.0.2.4.422     cis
 attribute db2nodePtr                     1.3.18.0.2.4.423     dn
 attribute db2gwPtr                       1.3.18.0.2.4.424     dn
 attribute db2additionalParameters        1.3.18.0.2.4.426     cis
 attribute db2ARLibrary                   1.3.18.0.2.4.427     cis
 attribute db2authenticationLocation      1.3.18.0.2.4.425     cis
 attribute db2databaseRelease             1.3.18.0.2.4.429     cis
 attribute DCEPrincipalName               1.3.18.0.2.4.443     cis
```

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

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

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


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

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

在添加 DB2 模式定义之后，必须重新启动该 Directory Server 以使所有更改生效。

相关参考:

- 第 305 页的『DB2 使用的 LDAP 对象类和属性』

扩展 IBM SecureWay Directory Server 的目录模式

如果在使用 IBM® SecureWay® LDAP Directory Server V3.1 或更新版本，则版本 8.2 之前的 DB2® 通用数据库（DB2 UDB）所需的所有对象类和属性均包括在基本模式中。运行以下命令以使用版本 8.2 中引入的新 DB2 UDB 属性扩展基本模式:

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

以下是 `altgwnode.ldif` 文件的内容:

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


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


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

altgwnode.ldif 和 altgwnode.readme 文件可在此 URL 处找到: ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap

在添加 DB2 模式定义之后，必须重新启动该 Directory Server 以使所有更改生效。

相关概念:

- 第 296 页的『扩展具有 DB2 对象类和属性的 LDAP 目录模式』
- 第 302 页的『扩展 Sun One Directory Server 的目录模式』

相关任务:

- 第 296 页的『扩展 Active Directory 的目录模式』

扩展 Sun One Directory Server 的目录模式

Sun One Directory Server 也称为 Netscape 或 iPlanet 目录服务器。

要使 Sun One Directory Server 在您的环境中工作，将 60ibmdb2.ldif 文件添加到下列目录:

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

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

以下是文件的内容:

```
    ########################################################################
    # IBM&regtm; DB2&regtm; Universal Database
    ########################################################################
    dn: cn=schema
    ########################################################################
    # Attribute Definitions (Before V8.2)
    ########################################################################
    attributetypes: ( 1.3.18.0.2.4.305  NAME 'binProperty'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.5               X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.306  NAME 'binPropertyType'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.307  NAME 'cesProperty'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.308  NAME 'cesPropertyType'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.309  NAME 'cisProperty'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.310  NAME 'cisPropertyType'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.320  NAME 'propertyType'              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.329  NAME 'systemName'                SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.419  NAME 'db2nodeName'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.420  NAME 'db2nodeAlias'              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.428  NAME 'db2instanceName'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.418  NAME 'db2Type'                   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.421  NAME 'db2databaseName'           SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.422  NAME 'db2databaseAlias'          SYNTAX 1.3.6.1.4.1.1466.115.121.1.15              X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.426  NAME 'db2additionalParameters'   SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.427  NAME 'db2ARLibrary'              SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.425  NAME 'db2authenticationLocation' SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.429  NAME 'db2databaseRelease'        SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.443  NAME 'DCEPrincipalName'          SYNTAX 1.3.6.1.4.1.1466.115.121.1.15 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.423  NAME 'db2nodePtr'                SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.424  NAME 'db2gwPtr'                  SYNTAX 1.3.6.1.4.1.1466.115.121.1.12 SINGLE-VALUE X-ORIGIN 'IBM DB2' )
    ########################################################################
    # Attribute Definitions (V8.2)
    ########################################################################
    attributetypes: ( 1.3.18.0.2.4.3092 NAME 'db2altgwPtr'               SYNTAX 1.3.6.1.4.1.1466.115.121.1.12              X-ORIGIN 'IBM DB2' )
    attributetypes: ( 1.3.18.0.2.4.3093 NAME 'db2altnodePtr'             SYNTAX 1.3.6.1.4.1.1466.115.121.1.12              X-ORIGIN 'IBM DB2' )
    ########################################################################
# Object Class Definitions
    # DB2Database for V8.2 has the above two new optional attributes.
    ########################################################################
    objectClasses: ( 1.3.18.0.2.6.90  NAME 'eProperty'          SUP top STRUCTURAL
     MAY ( cn $ propertyType $ binProperty $ binPropertyType $ cesProperty $ cesPropertyType $ cisProperty $ cisPropertyType )
     X-ORIGIN 'IBM DB2' )
    objectClasses: ( 1.3.18.0.2.6.84  NAME 'eApplicationSystem' SUP top STRUCTURAL MUST systemName
     X-ORIGIN 'IBM DB2' )
    objectClasses: ( 1.3.18.0.2.6.116 NAME 'DB2Node'            SUP top STRUCTURAL MUST db2nodeName
     MAY ( db2instanceName $ db2nodeAlias $ db2Type $ description $ host $ protocolInformation )
     X-ORIGIN 'IBM DB2' )
    objectClasses: ( 1.3.18.0.2.6.117 NAME 'DB2Database'        SUP top STRUCTURAL MUST (db2databaseName $ db2nodePtr )
     MAY ( db2additionalParameters $ db2altgwPtr $ db2altnodePtr $ db2ARLibrary $ db2authenticationLocation $ db2databaseAlias $ db2databaseRelease $ db2gwPtr $ DCEPrincipalName $ description )
     X-ORIGIN 'IBM DB2' )
```

60ibmdb2.ldif 和 60ibmdb2.readme 文件可在此 URL 处找到: ftp://ftp.software.ibm.com/ps/products/db2/tools/ldap

在添加 DB2 模式定义之后，必须重新启动该 Directory Server 以使所有更改生效。

相关概念:

- 第 296 页的『扩展具有 DB2 对象类和属性的 LDAP 目录模式』
- 第 300 页的『扩展 IBM SecureWay Directory Server 的目录模式』

相关任务:

- 第 296 页的『扩展 Active Directory 的目录模式』

DB2 使用的 LDAP 对象类和属性

下面的表描述了 DB2 使用的对象类:

表 23. cimManagedElement

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

表 24. cimSetting

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

表 25. eProperty

类	eProperty
Active Directory LDAP 显示名	ibm-eProperty
Active Directory 公共名（cn）	ibm-eProperty

表 25. *eProperty (续)*

类	**eProperty**
描述	用来指定用户首选项属性的任何应用程序特定设置
SubClassOf	cimSetting
必需的属性	
可选的属性	propertyType cisPropertyType cisProperty cesPropertyType cesProperty binPropertyType binProperty
类型	结构
OID（对象标识）	1.3.18.0.2.6.90
GUID（全局唯一标识）	b3afd69c-5c5b-11d3-b818-002035559151

表 26. *DB2Node*

类	**DB2Node**
Active Directory LDAP 显示名	ibm-db2Node
Active Directory 公共名（cn）	ibm-db2Node
描述	表示 DB2 服务器
SubClassOf	eSap / ServiceConnectionPoint
必需的属性	db2nodeName
可选的属性	db2nodeAlias db2instanceName db2Type host / dNSHostName（请参阅注释 2） protocolInformation/ServiceBindingInformation
类型	结构
OID（对象标识）	1.3.18.0.2.6.116
GUID（全局唯一标识）	b3afd65a-5c5b-11d3-b818-002035559151

表 26. DB2Node (续)

类	DB2Node
特殊注意事项	1. *DB2Node* 类是从 IBM SecureWay Directory 下面的 *eSap* 对象类和 Microsoft Active Directory 下面的 *ServiceConnectionPoint* 对象类派生的。 2. *host* 用于 IBM SecureWay 环境。*dNSHostName* 属性在 Microsoft Active Directory 下使用。 3. *protocolInformation* 仅在 IBM SecureWay 环境中使用。对于 Microsoft Active Directory，从 ServiceConnectionPoint 类继承的 *ServiceBindingInformation* 属性用来包含协议信息。

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

TCP/IP 的标记为:

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

APPN 的标记为:

- “APPN”
- 网络标识
- 伙伴 LU
- 事务程序（TP）名（仅支持“应用程序 TP”，不支持“服务 TP” - 十六进制的 TP）
- 方式
- 安全性（“NONE”、“SAME”或“PROGRAM”）
- （可选）LAN 适配器地址
- （可选）更改密码 LU

注: 在 DB2 UDB Windows 版客户机上，如果未在本地 SNA 堆栈上配置 APPN 信息；并且，如果在 LDAP 中找到了 LAN 适配器地址和可选更改密码 LU，则 DB2 UDB 客户机尝试使用此信息来配置 SNA 堆栈（如果它知道如何配置堆栈的话）。

NetBIOS 的标记为:

- “NETBIOS”
- 服务器 NetBIOS 工作站名称

“命名管道”的标记为:

- “NPIPE”
- 服务器的计算机名称

- 服务器的实例名

表 27. DB2Database

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

表 28. db2additionalParameters

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

表 29. db2authenticationLocation

属性	db2authenticationLocation
Active Directory LDAP 显示名	ibm-db2AuthenticationLocation
Active Directory 公共名（cn）	ibm-db2AuthenticationLocation
描述	指定执行认证的位置

表 29. *db2authenticationLocation (续)*

属性	db2authenticationLocation
语法	忽略大小写的字符串
最大长度	64
多值	单值
OID（对象标识）	1.3.18.0.2.4.425
GUID（全局唯一标识）	b3afd317-5c5b-11d3-b818-002035559151
注释	有效值有： CLIENT、SERVER、DCS、DCE、KERBEROS、SVRENCRYPT 或 DCSENCRYPT

表 30. *db2ARLibrary*

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

表 31. *db2databaseAlias*

属性	db2databaseAlias
Active Directory LDAP 显示名	ibm-db2DatabaseAlias
Active Directory 公共名（cn）	ibm-db2DatabaseAlias
描述	数据库别名
语法	忽略大小写的字符串
最大长度	1024
多值	多值
OID（对象标识）	1.3.18.0.2.4.422
GUID（全局唯一标识）	b3afd318-5c5b-11d3-b818-002035559151

表 32. *db2databaseName*

属性	db2databaseName
Active Directory LDAP 显示名	ibm-db2DatabaseName
Active Directory 公共名（cn）	ibm-db2DatabaseName
描述	数据库名称
语法	忽略大小写的字符串
最大长度	1024
多值	单值
OID（对象标识）	1.3.18.0.2.4.421

表 32. *db2databaseName (续)*

属性	**db2databaseName**
GUID（全局唯一标识）	b3afd319-5c5b-11d3-b818-002035559151

表 33. *db2databaseRelease*

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

表 34. *db2nodeAlias*

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

表 35. *db2nodeName*

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

表 36. *db2nodePtr*

属性	**db2nodePtr**
Active Directory LDAP 显示名	ibm-db2NodePtr
Active Directory 公共名（cn）	ibm-db2NodePtr

表 36. db2nodePtr (续)

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

表 37. db2altnodePtr

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

表 38. db2gwPtr

属性	db2gwPtr
Active Directory LDAP 显示名	ibm-db2GwPtr
Active Directory 公共名（cn）	ibm-db2GwPtr
描述	指向表示网关服务器的“节点”对象的指针，可从该网关服务器存取数据库
语法	专有名称
最大长度	1000
多值	单值
OID（对象标识）	1.3.18.0.2.4.424
GUID（全局唯一标识）	b3afd31b-5c5b-11d3-b818-002035559151

表 39. db2altgwPtr

属性	db2altgwPtr
Active Directory LDAP 显示名	ibm-db2AltGwPtr
Active Directory 公共名（cn）	ibm-db2AltGwPtr
描述	指向表示备用网关服务器的“节点”对象的指针
语法	专有名称
最大长度	1000

表 39. db2altgwPtr (续)

属性	db2altgwPtr
多值	多值
OID（对象标识）	1.3.18.0.2.4.3092
GUID（全局唯一标识）	70ab425d-65cc-4d7f-91d8-084888b3a6db

表 40. db2instanceName

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

表 41. db2Type

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

表 42. DCEPrincipalName

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

表 43. *cesProperty*

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

表 44. *cesPropertyType*

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

表 45. *cisProperty*

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

表 46. *cisPropertyType*

属性	**cisPropertyType**
Active Directory LDAP 显示名	ibm-cisPropertyType
Active Directory 公共名（cn）	ibm-cisPropertyType

表 46. cisPropertyType (续)

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

表 47. binProperty

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

表 48. binPropertyType

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

表 49. PropertyType

属性	PropertyType
Active Directory LDAP 显示名	ibm-propertyType
Active Directory 公共名（cn）	ibm-propertyType
描述	此属性的值描述 eProperty 对象的语义特征

表 49. *PropertyType (续)*

属性	PropertyType
语法	忽略大小写的字符串
最大长度	128
多值	多值
OID（对象标识）	1.3.18.0.2.4.320
GUID（全局唯一标识）	b3afd4ed-5c5b-11d3-b818-002035559151

表 50. *settingID*

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

相关概念:

- 第 279 页的『 “轻量级目录访问协议”（LDAP）简介 』

附录 D. 向多个数据库分区发出命令

在分区数据库环境中发出命令

在分区数据库系统中，您可能想要发出将在实例中的机器或在多个数据库分区服务器（节点）上运行的命令。为此，您可以使用 **rah** 命令或 **db2_all** 命令。**rah** 命令允许您发出将在实例中的机器上运行的命令。若想要命令在实例中的多个数据库分区服务器上运行，则运行 **db2_all** 命令。本章节概述了这些命令。以下的信息仅适用于分区数据库系统。

注:

1. 在基于 UNIX® 的平台上，您的登录 shell 程序可以是 Korn shell 程序或任何其它的 shell；但是，不同 shell 处理包含特殊字符的命令所用的方式不同。
2. 在 Windows NT 上，要运行 **rah** 命令或 **db2_all** 命令，您必须使用“管理员”组成员的用户帐户来登录。

要确定命令的范围，请参阅 *Command Reference*。本书指示命令是在单个的还是所有的数据库分区服务器上运行。若命令在一个数据库分区服务器上运行，而您想该命令在其所有上面运行，则使用 **db2_all**。例外情况是 **db2trc** 命令，它在一台机器的所有逻辑节点（数据库分区服务器）上运行。若想要在所有机器的所有逻辑节点上运行 **db2trc**，则使用 **rah**。

相关概念:

- 第 317 页的『rah 和 db2_all 命令概述』
- 第 319 页的『指定 rah 和 db2_all 命令』

相关参考:

- 第 318 页的『rah 和 db2_all 命令描述』

rah 和 db2_all 命令概述

可以依次在各个数据库分区服务器上按顺序运行命令，或可以用并行的方式运行命令。在基于 UNIX® 的平台上，如果以并行方式运行这些命令，您可以选择将输出发送至缓冲区并收集该输出以便显示（缺省行为），或者可在发出该命令的机器上显示该输出。在 Windows NT 上，若以并行方式运行这些命令，则在发出该命令的机器上显示输出。

要使用 **rah** 命令，输入:

```
rah command
```

要使用 **db2_all** 命令，输入:

```
db2_all command
```

有关 **rah** 语法的帮助，输入

```
rah "?"
```

该命令几乎可以是在交互式提示符下输入的任何内容，例如，要按顺序运行的多个命令。在基于 UNIX 的平台上，使用分号（;）将多个命令分开。在 Windows NT 上，使用 & 符号将多个命令分开。不要在最后一个命令之后使用分隔符。

以下示例显示如何使用 **db2_all** 命令来更改在节点配置文件中指定的所有数据库分区上的数据库配置。因为 ; 字符在双引号之内，所以将同时运行请求:

```
db2_all ";UPDATE DB CFG FOR sample USING LOGFILSIZ 100"
```

相关概念:

- 第 317 页的『在分区数据库环境中发出命令』
- 第 319 页的『指定 rah 和 db2_all 命令』

相关参考:

- 第 318 页的『rah 和 db2_all 命令描述』

rah 和 db2_all 命令描述

可以使用下列命令:

命令	描述
rah	在所有机器上运行该命令。
db2_all	在指定的所有数据库分区服务器上运行该命令。
db2_kill	突然停止正在多个数据库服务器上运行的所有进程，并清除所有数据库分区服务器上的所有资源。此命令使数据库变得不一致。除非有 IBM 服务中心指导，否则不要发出此命令。
db2_call_stack	在基于 UNIX 的平台上，使在所有数据库分区服务器上运行的所有进程将调用回溯写入 syslog。 在 Windows NT 上，使在所有数据库分区服务器上运行的所有进程将调用回溯写入实例目录中的 P*xxxx.nnn* 文件，其中 P*xxxx* 是进程标识，而 *nnn* 是节点号。

在基于 UNIX 的平台上，这些命令执行带特定隐式设置的 **rah**，如:

- 以并行方式在所有机器上运行
- 将命令输出分别缓存到 /tmp/$USER/db2_kill 和/tmp/$USER/db2_call_stack 中。

在 Windows NT 上，这些命令执行 **rah**，以便在所有机器上以并行方式运行。

相关概念:

- 第 317 页的『rah 和 db2_all 命令概述』
- 第 319 页的『指定 rah 和 db2_all 命令』
- 第 320 页的『在基于 UNIX 的平台上以并行方式运行命令』

指定 rah 和 db2_all 命令

可以:

- 从命令行指定命令作为参数。
- 为响应提示而指定命令（若不指定任何参数）。

若命令包含下列特殊字符则应该使用提示方法:

```
| & ; < > ( ) { } [ ] unsubstituted $
```

若在命令行上指定命令作为参数，而且若它包含刚才列出的任何特殊字符，必须以双引号将命令括起来。

注: 在基于 UNIX® 的平台上，将该命令添加至命令历史中，就象您在提示符处输入它一样。

可以正常输入命令中的所有特殊字符（除\外，不必以双引号括起来）。若需要在命令中包括 \，则必须输入两个反斜杠（\\）。

注: 在基于 UNIX 的平台上，若您未使用 Korn shell 程序，则可以正常地输入该命令中的所有特殊字符（除 "、\、不可替换的 $ 和单引号（'）外，其它字符都不需要用引号引起来）。若需要在命令中包括其中一个字符，则必须在字符前加三个反斜杠（\\\）。例如，若需要在命令中包括一个 \，则必须输入四个反斜杠（\\\\）。

若需要在命令中包括双引号（"），则必须在双引号前加三个反斜杠，例如 \\\"。

注:

1. 在基于 UNIX 的平台上，除非命令 shell 提供了在被括上单引号的字符串中输入单引号的某种方法，否则您不能在命令中包括单引号（'）。
2. 在 Windows NT 上，除非命令窗口提供了在被括上单引号的字符串中输入单引号的某种方法，否则您不能在命令中包括单引号（'）。

当运行任何 Korn shell 程序的 shell 脚本，而该脚本包含从后台中的标准输入进行读取的逻辑时，您应该明确地将标准输入重定向到一个源，在此源上进程可以读取而不必在终端上停止（SIGTTIN 消息）。要重定向标准输入，可以用下列格式运行脚本:

```
shell_script </dev/null &
```

若没有提供输入。

同样，当在后台运行 db2_all 时，应该始终指定 </dev/null。例如:

```
db2_all ";run_this_command" </dev/null &
```

通过执行此操作，可以重定向标准输入，并避免在终端上停止。

当您不关心来自远程命令的输出时，此方法的一种替代方法是在 db2_all 前缀中使用“daemonize”选项:

```
db2_all ";daemonize_this_command" &
```

相关概念:

- 第 320 页的『在基于 UNIX 的平台上以并行方式运行命令』
- 第 321 页的『附加 rah 信息（仅适用于 Solaris 和 AIX）』

相关任务:

- 第 327 页的『在 Windows NT 上为 rah 设置缺省环境概要文件』

相关参考:

- 第 318 页的『rah 和 db2_all 命令描述』
- 第 322 页的『rah 命令前缀序列』
- 第 325 页的『控制 rah 命令』

在基于 UNIX 的平台上以并行方式运行命令

注: 本节中的信息只适用于基于 UNIX® 的平台。

缺省情况下，命令在每台机器上顺序运行，但通过在命令前加上某些前缀序列，可以指定使用后台 rshell 以并行方式运行命令。若 rshell 在后台运行，则每个命令将输出放置在其远程机器的缓冲区文件中。此进程分两个部分检索输出:

1. 在远程命令完成后。
2. 在 rshell 终止后，若某些进程仍在运行，则 rshell 可能会过一段时间才终止。

缺省情况下，缓冲区文件的名称为 `/tmp/$USER/rahout`，但可以通过环境变量 $RAHBUFDIR/$RAHBUFNAME 指定该名称。

当指定想要命令同时运行时，缺省情况下，此脚本将附加命令作为前缀加到发送至所有主机的命令上，以检查 $RAHBUFDIR 和 $RAHBUFNAME 是否可用于缓冲区文件。这会创建 $RAHBUFDIR。为避免此类情况，导出环境变量 `RAHCHECKBUF=no`。如果知道目录存在且可用，则可以执行此操作以节省时间。

在使用 **rah** 同时在多台机器上运行命令之前:

- 确保对于您的用户标识，目录 `/tmp/$USER` 在每台机器上存在。要在目录尚不存在的情况下创建目录，运行:

  ```
  rah ")mkdir /tmp/$USER"
  ```

- 将下一行添加至 `.kshrc` （对于 Korn shell 程序语法）或 `.profile`，并将它输入到当前会话中:

  ```
  export RAHCHECKBUF=no
  ```

- 确保您运行远程命令的每台机器的标识在其 `.rhosts` 文件中有一个条目对应于运行 **rah** 的标识；并且运行 **rah** 的标识在其 `.rhosts` 文件中有一个条目对应于运行远程命令的每台机器的标识。

相关概念:

- 第 321 页的『附加 rah 信息（仅适用于 Solaris 和 AIX）』

相关任务:

- 第 321 页的『在基于 UNIX 的平台上监视 rah 进程』

相关参考:

- 第 322 页的『rah 命令前缀序列』
- 第 327 页的『在基于 UNIX 的平台上确定 rah 的问题』

在基于 UNIX 的平台上监视 rah 进程

过程:

注: 本节中的信息只适用于基于 UNIX 的平台。
当任何远程命令仍在运行或仍在累加缓冲输出时，由 rah 启动的进程对活动进行监视以:

- 将消息写入终端，指示哪些命令尚未运行
- 检索缓冲输出

在环境变量 RAHWAITTIME 控制的时间间隔写出信息性消息。请参阅帮助信息以获取如何指定时间间隔的详细信息。通过导出 RAHWAITTIME=0，可以完全不显示所有信息性消息。

主监视过程是一个命令，其命令名（由 ps 命令显示）为 **rahwaitfor**。第一条信息性消息告诉您此进程的 pid（进程标识）。所有其它监视进程将表现为运行 **rah** 脚本（或符号链接的名称）的 **ksh** 命令。若愿意，可通过以下命令停止所有监视进程:

```
kill <pid>
```

其中 <pid> 是主监视进程的进程标识。不要指定信号编号。将它保留为缺省值 15。这根本不会影响远程命令，但会阻止自动显示缓冲输出。注意，在执行单个 **rah** 时，可能有两组或更多组不同监视进程在不同时间执行。但是，若任何时候停止当前组监视进程，则不会再启动其它监视进程。

若正规登录 shell 程序不是 Korn shell 程序（例如 /bin/ksh），则可以使用 **rah**，但如何输入包含下列特殊字符的命令的规则稍微有些差异:

```
"  unsubstituted $ '
```

有关更多信息，输入 rah "?". 而且，在基于 UNIX 的环境中，若执行远程命令的标识的登录 shell 程序不是 Korn shell 程序，则以该标识执行 **rah** 的登录 shell 程序也不得是 Korn shell 程序。（**rah** 根据本地标识来决定远程标识的 shell 是否为 Korn shell 程序）。该 shell 不得对单引号内的字符串执行任何替换或特殊处理。必须照原样保持它。

相关概念:

- 第 320 页的『在基于 UNIX 的平台上以并行方式运行命令』
- 第 321 页的『附加 rah 信息（仅适用于 Solaris 和 AIX）』

附加 rah 信息（仅适用于 Solaris 和 AIX）

为了增强性能，在大型系统上扩充了 rah 以使用 tree_logic。也就是说，rah 将检查该列表包含多少个节点，若该数目超过阈值，它会构造列表的一个子集，并将它自己的递归调用发送到那些节点。在那些节点上，递归调用的 rah 遵循相同的逻辑，直到该列表小得能够符合将该命令发送到列表中所有节点的标准逻辑（现在称为“树叶”逻辑）为止。该阈值可由环境变量 RAHTREETHRESH 指定，缺省值为 15。

对于每个物理节点存在多逻辑节点的系统，db2_all 比较愿意将递归调用发送到各个不同的物理节点，然后 rsh 到同一个物理节点上的其它逻辑节点，这样可减少物理节点间的通信量。（这种方法只适合 db2_all，不适合 rah，因为 rah 始终只发送到不同的物理节点。）

相关概念:

- 第 320 页的『在基于 UNIX 的平台上以并行方式运行命令』

相关任务:

- 第 321 页的『在基于 UNIX 的平台上监视 rah 进程』

rah 命令前缀序列

前缀序列是一个或多个特殊字符。在命令字符前输入一个或多个前缀序列而不插入任何空格。若想指定多个序列，可以任何顺序输入它们，但任何多字符序列中的输入字符必须按顺序输入。若您输入任何前缀序列，则您必须将整个命令（包括该前缀序列）置于双引号内，如下例所示:

- 在基于 UNIX 的平台上:

```
rah "};ps -F pid,ppid,etime,args -u $USER"
```

- 在 Windows NT 上:

```
rah "||db2 get db cfg for sample"
```

前缀序列有:

顺序	目的
|	在后台按顺序运行命令。
|&	在后台按顺序运行这些命令，并在所有远程命令完成之后，终止该命令，即使有一些进程仍在运行。例如，若子进程（在基于 UNIX 的平台上）或后台进程（在 Windows 上）仍在运行，可能造成延迟。在此情况下，该命令启动独立的后台进程来检索命令终止之后生成的任何远程输出，并将该输出写回至源机器。 **注**: 在基于 UNIX 的平台上，指定 & 会降低性能，因为需要运行更多的 **rsh** 命令。
||	在后台以并行方式运行命令。
||&	在后台以并行方式运行命令并在所有远程命令完成之后终止命令，如以上 |& 的情况所述。 **注**: 在基于 UNIX 的平台上，指定 & 会降低性能，因为需要运行更多的 **rsh** 命令。
;	与以上的 ||& 相同。这是一个较短的替代格式。 **注**: 在基于 UNIX 的平台上，指定 ; 会降低性能（相对于 || 而言），因为需要运行更多的 **rsh** 命令。
]	在执行命令之前预先暂挂用户概要文件的点执行。 **注**: 仅在基于 UNIX 的平台上可用。

} 在执行命令之前预先暂挂在 $RAHENV 中命名的文件的点执行（可能是 .kshrc）。

注: 仅在基于 UNIX 的平台上可用。

]} 在执行命令之前，预先暂挂用户概要文件的点执行，然后执行在 $RAHENV 中命名的文件（可能是 .kshrc）。

注: 仅在基于 UNIX 的平台上可用。

) 停止执行用户概要文件和 $RAHENV 中命名的文件。

注: 仅在基于 UNIX 的平台上可用。

' 将命令调用回传至机器。

< 发送至除此机器外的所有机器。

<<–nnn< 发送至除数据库分区服务器 *nnn* 外的所有数据库分区服务器（db2nodes.cfg 中除节点号为 *nnn* 之外的所有数据库分区服务器，请参阅本表中最后一个前缀序列后的第一段）。

<<+nnn< 仅发送至数据库分区服务器 *nnn*（db2nodes.cfg 中节点号为 *nnn* 的数据库分区服务器，参阅本表中最后一个前缀序列后的第一段）。

（空格字符） 在后台运行远程命令，stdin、stdout 和 stderr 全部关闭。此选项仅当在后台运行命令时才有效，即仅在还包括 \ 或 ; 的前缀序列中有效。它允许命令尽快完成（远程命令一启动就完成）。若在 **rah** 命令行上指定此前缀序列，则将该命令用单引号括起来，或用双引号括起该命令并在前缀字符之前加 \ 。例如，

```
rah '; mydaemon'
```

或

```
rah ";\ mydaemon"
```

当作为后台进程运行时，**rah** 命令从不会等待任何要返回的输出。

> 用机器名替换找到的 <>。

" 用机器索引替换找到的 ()，用节点号替换找到的 ##。

注:

1. 机器索引是与数据库系统中的机器关联的号码。若不在运行多逻辑节点，则机器的机器索引对应于节点配置文件中该机器的节点号。要在多逻辑节点环境中获取机器的机器索引，不要计算那些运行多逻辑节点的机器的重复条目。例如，若 MACH1 正在运行两个逻辑节点，MACH2 也正在运行两个逻辑节点，则节点配置文件中 MACH3 的节点号为 5。但是，MACH3 的机器索引应是 3。

   在 Windows NT 上，不要编辑节点配置文件。要获取机器索引，使用 **db2nlist** 命令。
2. 当指定 " 时，未从机器列表中删除重复。

当使用 <<-nnn< 和 <<+nnn<前缀序列时，*nnn* 是任何 1、2 或 3 位分区号，该分区号必须与 db2nodes.cfg 文件中的 *nodenum* 值匹配。

注: 前缀序列被认为是命令的一部分。若指定前缀序列作为命令的一部分，必须将整个命令，包括前缀序列，括在双引号内。

相关概念:

- 第 319 页的『指定 rah 和 db2_all 命令』
- 第 320 页的『在基于 UNIX 的平台上以并行方式运行命令』

相关参考:

- 第 318 页的『rah 和 db2_all 命令描述』

指定分区环境中的机器的列表

过程:

缺省情况下，机器列表取自节点配置文件 db2nodes.cfg。可以通过以下方法覆盖此设置:

- 通过导出（在基于 UNIX 的平台上）或设置（在 Windows NT 上）环境变量 RAHOSTFILE，指定包含机器列表的文件的路径名。
- 通过导出（在基于 UNIX 的平台上）或设置（在 Windows NT 上）环境变量 RAHOSTLIST，明确指定该列表为由空格分隔的一连串名称。

  注: 若这两个环境变量都被指定，则 RAHOSTLIST 具有优先顺序。

注: 在 Windows NT 上，要避免将不一致性引入该节点配置文件，不要以手工方式编辑它。要获取实例中机器的列表，使用 **db2nlist** 命令。

相关任务:

- 第 324 页的『除去分区环境中机器列表中的重复条目』

除去分区环境中机器列表中的重复条目

过程:

如果正在一台机器上运行具有多逻辑节点（数据库分区服务器）的 DB2 通用数据库企业服务器版，则 db2nodes.cfg 文件将包含此机器的多个条目。在此情况下，**rah** 命令需要知道您是希望该命令在每台机器上只执行一次，还是为 db2nodes.cfg 文件中列出的每个逻辑节点均执行一次。使用 **rah** 命令来指定机器。使用 **db2_all** 命令来指定逻辑节点。

注: 在基于 UNIX 的平台上，若您指定机器，则 **rah** 将会正常地从机器列表中删除重复的项，例外情况：若您指定逻辑节点，则 **db2_all** 会将下列赋值追加至您的命令之前:

```
export DB2NODE=nnn  （对于 Korn shell 程序语法）
```

其中 *nnn* 取自 db2nodes.cfg 文件中的相应行的节点号，以便将命令路由至所希望的数据库分区服务器。

当指定逻辑节点时，可以使用 <<-nnn< 和 <<+nnn< 前缀顺序来限制列表包括除某个逻辑节点之外的所有逻辑节点，或指定一个数据库分区服务器。若想要首先在目录节点

运行命令，并在该命令完成时，在所有其它数据库分区服务器上运行同一命令（可能以并行形式），则可能要执行此操作。当运行 **db2 restart database** 命令时，它常常是必需的。必须知道目录节点的节点号才能执行此操作。

若您使用 **rah** 命令来执行 **db2 restart database**，重复条目就会从机器列表中被删除。但是，若指定 " 前缀，则不删除重复项，因为认为使用 " 前缀会隐含发送至每个数据库分区服务器，而不是发送至每台机器。

相关任务:

- 第 324 页的『指定分区环境中的机器的列表』

相关参考:

- 『RESTART DATABASE Command』（*Command Reference*）
- 第 322 页的『rah 命令前缀序列』

控制 rah 命令

可使用下列环境变量来控制 **rah** 命令。

表 51.

名称	含义	缺省值
$RAHBUFDIR **注**: 仅在基于 UNIX 的平台上可用。	缓冲区目录	/tmp/$USER
$RAHBUFNAME **注**: 仅在基于 UNIX 的平台上可用。	缓冲区文件名	rahout
$RAHOSTFILE（在基于 UNIX 的平台上）；RAHOSTFILE（在 Windows NT 上）	包含主机列表的文件	db2nodes.cfg
$RAHOSTFILE（在基于 UNIX 的平台上）；RAHOSTLIST（在 Windows NT 上）	字符串形式的主机列表	取自 $RAHOSTFILE
$RAHCHECKBUF **注**: 仅在基于 UNIX 的平台上可用。	若设置为 "no"，则绕过检查	未设置
$RAHSLEEPTIME（在基于 UNIX 的平台上）； RAHSLEEPTIME（在 Windows NT 上）	以秒计的时间，此脚本将在此时间内等待来自以并行形式运行的命令的初始输出	对于 **db2_kill** 为 86400 秒，对于其它所有命令则为 200 秒

表 51. (续)

名称	含义	缺省值
$RAHWAITTIME（在基于 UNIX 的平台上）；RAHWAITTIME（在 Windows NT 上）	在 Windows NT 上，连续检查远程作业是否仍在运行的时间间隔（以秒计）。 在基于 UNIX 的平台上，连续检查远程作业是否仍在运行以及 rah: 正在等待 <pid> ... 消息的时间间隔（以秒计）。 对于所有平台，指定任何正整数。具有前导零的前缀值可抑制消息，例如，导出 RAHWAITTIME=045。 不必指定较小的值，因为 **rah** 不依靠这些检查来检测作业是否完成。	45 秒
$RAHENV **注:** 仅在基于 UNIX 的平台上可用。	若 $RAHDOTFILES=E 或 K 或 PE 或 B，则指定要执行的文件名	$ENV
$RAHUSER（在基于 UNIX 的平台上）；RAHUSER（在 Windows NT 上）	在基于 UNIX 的平台上，运行远程命令所使用的用户标识。 在 Windows NT 上，与 DB2 “远程命令服务” 相关的登录帐户	$USER

注: 在基于 UNIX 的平台上，将使用运行 **rah** 的 $RAHENV 的值，而不是远程 shell 设置的值（若有的话）。

相关参考:

- 第 326 页的『在基于 UNIX 的平台上使用 $RAHDOTFILES』

在基于 UNIX 的平台上使用 $RAHDOTFILES

注: 本节中的信息只适用于基于 UNIX 的平台。

以下是未指定前缀序列时运行的 . 文件:

P .profile

E 在 $RAHENV 中命名的文件（可能是 .kshrc）

K 与 E 相同

PE 后跟 $RAHENV 中命名的文件的 .profile（可能是 .kshrc）

B 与 PE 相同

N 无

注: 若登录 shell 程序不是 Korn shell 程序，则将在 Korn shell 程序进程中执行您指定要执行的任何点文件，因此，必须遵守 Korn shell 程序语法。例如，若登录 shell 程序是 C shell，要对 **rah** 执行的命令设置 .cshrc 环境，则应该创建等效于 .cshrc 的 Korn shell 程序 INSTHOME/.profile，并在 INSTHOME/.cshrc 中指定:

```
setenv RAHDOTFILES P
```

或应该创建等效于 .cshrc 的 Korn shell 程序 INSTHOME/.kshrc，并在 INSTHOME/.cshrc 中指定:

```
setenv RAHDOTFILES E
  setenv RAHENV INSTHOME/.kshrc
```

并且，若没有 tty（由 **rsh** 调用时），则 .cshrc 必须不写入标准输出。通过将写至标准输出的任何行用引号括起来，可确保这一点，例如,

```
if { tty -s } then echo "executed .cshrc";
  endif
```

相关参考:

• 第 325 页的『控制 rah 命令』

在 Windows NT 上为 rah 设置缺省环境概要文件

过程:

注: 本节中的信息只适用于 Windows NT。
要设置 **rah** 命令的缺省环境概要文件，使用文件 db2rah.env，应该在实例目录中创建该文件。该文件应该有以下格式:

```
; This is a comment line
   DB2INSTANCE=instancename
   DB2DBDFT=database
   ; End of file
```

可以指定为 **rah** 进行环境初始化所需要的所有环境变量。

相关概念:

• 第 319 页的『指定 rah 和 db2_all 命令』

在基于 UNIX 的平台上确定 rah 的问题

注: 本节中的信息只适用于基于 UNIX 的平台。
以下是一些建议，告诉您如何处理在运行 **rah** 时可能遇到的某些问题:

1. **rah** 挂起（或运行很长的时间）

   此问题可能是由下列原因引起的:

   - **rah** 已确定它需要缓冲区输出，而您未导出 RAHCHECKBUF=no。因此，在运行命令之前，**rah** 向所有机器发送一条命令以检查缓冲区目录是否存在，如果该目录不存在，则会创建该目录。
   - 您在其中发送命令的一个或多台机器不响应。**rsh** 命令最终将超时，但超时时间间隔相当长，通常为 60 秒左右。

2. 已接受到下列各类消息:
   - 登录不正确
   - 拒绝许可权

   其中一台机器没有在其 .hosts 文件中正确定义运行 **rah** 的标识，或运行 **rah** 的标识没有在其 .rhosts 文件中正确定义其中一台机器。

3. 当使用后台 rshell 以并行方式运行命令时，虽然这些命令在期望的时间内在机器上运行并完成，但是 **rah** 要耗费较长的时间来检测它并设置 shell 提示符。

   运行 **rah** 的标识没有在其 .rhosts 文件中正确定义其中某台机器。

4. 虽然 **rah** 从 shell 命令行运行时运行情况良好，但是若使用 rsh 远程运行 **rah**，例如，

   ```
   rsh somewher -l $USER db2_kill
   ```

   ，则 **rah** 将永不会完成。

   这是正常的。**rah** 会启动后台监视进程，这些进程在 rah 退出后继续运行。那些进程通常将在与您运行的命令关联的所有进程将它们终止之后才结束。在 **db2_kill** 的情况下，这意味着终止所有数据库管理器。可通过寻找其命令是 **rahwaitfor** 和 kill <process_id> 的进程来终止监视进程。不要指定信号编号。而是要使用缺省值（15）。

5. 当在同一 $RAHUSER 下发出多个 **rah** 命令时，**rah** 的输出未正确显示，或 **rah** 错误地报告 $RAHBUFNAME 不存在。

   这是因为，并行执行多个 **rah** 正在尝试使用同一缓冲区文件（例如 $RAHBUFDIR/$RAHBUFNAME）对输出进行缓冲。要避免此问题，将不同的 $RAHBUFNAME 用于每个并行 **rah** 命令，例如，在下列 ksh 中:

   ```
   export RAHBUFNAME=rahout
     rah ";$command_1" &
     export RAHBUFNAME=rah2out
     rah ";$command_2" &
   ```

   或使用使 shell 自动选择唯一名称的方法，如:

   ```
   RAHBUFNAME=rahout.$$ db2_all "....."
   ```

   若磁盘空间有限，则无论使用哪种方法，都必须确保在某个时间清除缓冲区文件。**rah** 不会在执行结束时擦除缓冲区文件，不过，下次您指定与现有文件相同的缓冲区文件时，它将擦除并重新使用该现有文件。

6. 输入

   ```
   rah '"print from ()'
   ```

   并接收到消息:

   ```
   ksh: syntax error at line 1 : (' unexpected
   ```

   替代 () 和 ## 的先决条件是:

   - 使用 **db2_all**，而不是使用 **rah**。
   - 确保通过导出 RAHOSTFILE 或缺省为您的 /sqllib/db2nodes.cfg 文件来使用 RAHOSTFILE。没有这些先决条件，**rah** 将照原样保持 () 和 ##。您接收到错误，因为命令 **print from ()** 无效。

   要获取以并行方式运行命令时的性能提示，除非确实需要由 & 提供的功能，否则，使用 | 代替 |&，并使用 || 代替 ||& 或 ;。指定 & 需要更多 **rsh** 命令，因此会降低性能。

相关参考:

- 第 325 页的『控制 rah 命令』

附录 E. 使用 Windows 管理规范（WMI）支持

“Windows 管理规范”（WMI）简介

有一个建立管理基础结构标准的业界开端，它提供结合各种硬件和软件管理系统的信息的一种方法。此开端称为“基于 Web 的企业管理”（WBEM）。WBEM 是基于“公共信息模型”（CIM）模式的，它是由 Desktop Management Task Force（DMTF）派生的业界标准。

“Microsoft® Windows® 管理规范”（WMI）实现了受支持的 Windows 平台的 WBEM 开端。WMI 在 Windows 企业网络中非常有用，使用它可以减少维护和管理企业网络组件的成本。WMI 提供了下列各项:

- Windows 操作、配置和状态的一致模型。
- 允许存取管理信息的 COM API。
- 允许使用其它 Windows 管理服务。
- 一个灵活且可扩展的体系结构，它允许供应商编写其它 WMI 提供程序以支持新设备、应用程序和其它增强功能。
- 用来创建信息的详细查询的“WMI 查询语言”（WQL）。
- 一个 API，管理应用程序开发者可使用它来编写 Visual Basic 或“Windows 脚本编制主机”（WSH）脚本。

WMI 体系结构分为两个部分:

1. 包括“CIM 对象管理器”（CIMOM）的管理基础结构以及用来管理数据的中央存储区（称为 CIMOM 对象库）。CIMOM 允许应用程序以统一的方式存取管理数据。
2. WMI 提供程序。WMI 提供程序是 CIMOM 与受管对象之间的媒介。通过使用 WMI API，WMI 提供程序向 CIMOM 提供来自受管对象的数据、代表管理应用程序处理请求并生成事件通知。

“Windows 管理规范”（WMI）提供程序是标准的 COM 或 DCOM 服务器，它们充当受管对象与“CIM 对象管理器”（CIMOM）之间的媒介。如果 CIMOM 接收到管理应用程序对 CIMOM 对象库中不存在的数据或对事件的请求，则 CIMOM 将请求转发至 WMI 提供程序。WMI 提供程序为特定于其特定域的受管对象提供数据和事件通知。

相关概念:

- 第 329 页的『DB2 通用数据库与 Windows 管理规范集成』

DB2 通用数据库与 Windows 管理规范集成

“Windows® 管理规范”（WMI）可通过 DB2® 性能计数器并使用内置 PerfMon 提供程序来访问快照监视器。

WMI 可通过使用内置“注册表”提供程序存取 DB2 概要文件注册表变量。

“WMI 软件开发工具箱”（WMI SDK）包括几个内置提供程序:

- PerfMon 提供程序
- 注册表事件提供程序
- 注册表提供程序
- Windows NT® 事件日志提供程序
- Win32 提供程序
- WDM 提供程序

WMI 可使用内置 “Windows NT 事件日志” 提供程序来存取 “事件日志” 中的 DB2 错误。

DB2 Universal Database™（UDB）（DB2 通用数据库）具有 “DB2 WMI 管理” 提供程序及样本 WMI 脚本文件，用于存取下列受管对象:

1. 数据库服务器的实例，包括分区的实例。可以执行下列操作:
   - 枚举实例
   - 配置数据库管理器参数
   - 启动 / 停止 / 查询 DB2 服务器服务的状态
   - 设置或建立通信
2. 数据库。可以执行下列操作:
   - 枚举数据库
   - 配置数据库参数
   - 创建 / 删除数据库
   - 备份 / 复原 / 前滚数据库

运行 WMI 应用程序之前需要向系统注册 DB2 WMI 提供程序。通过输入下列命令完成注册:

- mofcomp %DB2PATH%\bin\db2wmi.mof

  此命令将 DB2 WMI 模式的定义装入到系统中。
- regsvr %DB2PATH%\bin\db2wmi.dll

  此命令向 Windows 注册 DB2 WMI 提供程序 COM DLL。

在两个命令中，%DB2PATH% 是安装 DB2 的路径。另外，db2wmi.mof 是包含 the DB2 WMI 模式定义的 .MOF 文件。

与 WMI 基础结构集成有几个好处:

1. 您可以在基于 Windows 的环境中使用 WMI 提供的工具很容易地编写脚本来管理 DB2 服务器。提供了样本 Visual Basic（VBS）脚本，可用它来执行简单任务，例如，列出实例、创建和删除数据库以及更新配置参数。样本脚本包括在 DB2 应用程序开发 Windows 版产品中。
2. 可以创建功能强大的管理应用程序，它们使用 WMI 执行许多任务。这些任务可包括:
   - 显示系统信息
   - 监视 DB2 性能
   - 监视 DB2 系统资源消耗情况

通过此类型的管理应用程序监视系统事件和 DB2 事件，可以更好地管理数据库。

3. 可使用现有 COM 和 Visual Basic 编程知识和技巧。通过提供 COM 或 Visual Basic 接口，程序员可以在开发企业管理应用程序时节省时间。

相关概念:

• 第 329 页的『 “Windows 管理规范” （WMI）简介 』

附录 F. 使用 Windows NT 安全性

DB2 Windows NT 版和 Windows NT 安全性简介

Windows® NT 域是通过特定且唯一的名称引用的客户机和服务器的组合；且共享称为“安全性存取管理器”（SAM）的单个用户帐户数据库。域中的其中一台计算机是域控制器。域控制器管理用户域交互作用的各个方面。域控制器使用域用户帐户数据库中的信息来认证登录域帐户的用户。对于每个域，都有一个域控制器作为主域控制器（PDC）。在域中，还可以有备份域控制器（BDC），它在没有主域控制器或主域控制器不可用时认证用户帐户。备份域控制器拥有 SAM 数据库的副本，会针对 PDC 上的主副本定期同步。

只需要在主域控制器中义用户帐户、用户标识和密码就可以存取域资源。

对于安装 Windows NT® 服务器时的设置过程，可选择创建下列对象:

- 在新域中创建主域控制器
- 在已知域中创建备份域控制器
- 在已知域中创建独立服务器

在新域中选择“控制器”会使该服务器成为主域控制器。

用户可能要登录本地机器，或者当在“Windows NT 域”中安装机器时，用户可能要登录该域。DB2® Windows NT 版支持这三个选项。要认证用户，DB2 首先检查本地机器，然后检查当前域的“域控制器”，最后检查“域控制器”认识的任何一个“可信域”。

为举例说明这是如何进行的，假设 DB2 实例需要服务器认证。该配置如下所示:

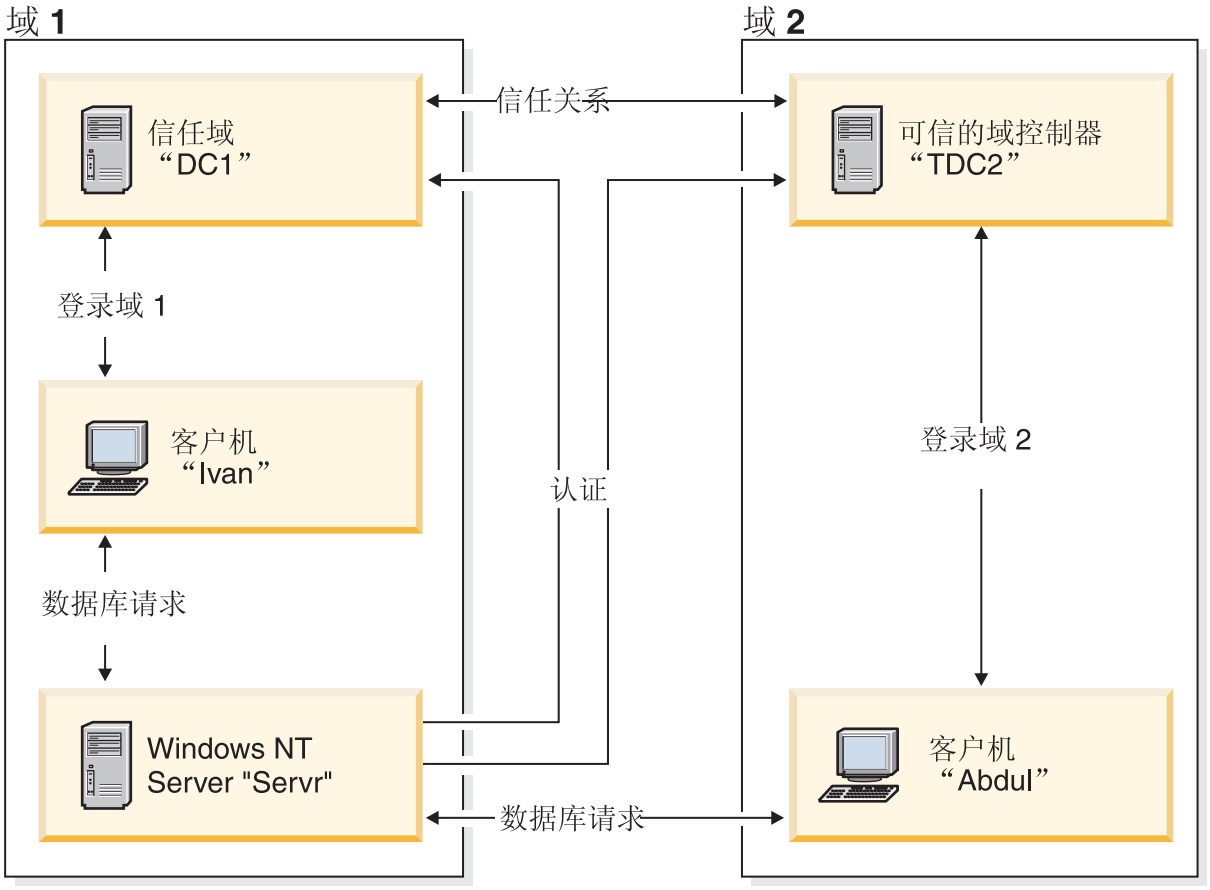


图 6. 使用 Windows NT 域的认证

除非客户机正在运行 Windows 9x，否则，每台机器应有一个安全性数据库，即“安全性存取管理”（SAM）。Windows 9x 机器没有 SAM 数据库。DC1 是域控制器，其中登记了客户机 Ivan 和 “DB2 Windows NT 版服务器” Servr。TDC2 是 DC1 的一个可信域，客户机 Abdul 是 TDC2 的域的一个成员。

相关概念:

- 第 337 页的『Windows NT 上的组和用户认证』

相关任务:

- 第 336 页的『将备份域控制器与 DB2 UDB 配合使用』
- 第 338 页的『在备份域控制器上安装 DB2』
- 第 339 页的『使用组和域安全性的 DB2 Windows NT 版认证』

具有服务器认证的 DB2 Windows NT 版方案

1. Abdul 登录 TDC2 域（即，它在 TDC2 SAM 数据库中是可识别的）。
2. Abdul 然后与一个 DB2 数据库连接，该数据库将被进行编目以驻留在 SRV3 上:

   ```
   db2 connect to remotedb user Abdul using fredpw
   ```

3. SRV3 确定可识别 Abdul 的位置。用于查找此信息的 API 首先搜索本地机器（SRV3），然后搜索域控制器（DC1），最后尝试搜索任何可信域。在 TDC2 上找到用户名 Abdul。此搜索顺序需要用户和组的单个名称空间。

4. 然后 SRV3:
   a. 在 TDC2 上验证用户名和密码。
   b. 通过询问 TDC2 来了解 Abdul 是否是管理员。
   c. 通过询问 TDC2 来枚举所有 Abdul 的组。

相关概念:

• 第 333 页的『DB2 Windows NT 版和 Windows NT 安全性简介』

具有客户机认证和 Windows NT 客户机的 DB2 Windows NT 版方案

1. 管理员 Dale 登录 SRV3，并将数据库实例的认证更改为“客户机”：

```
db2 update dbm cfg using authentication client
db2stop
 db2start
```

2. Ivan 在 Windows 客户机上登录 DC1 域（即，他在 DC1 SAM 数据库中是可识别的）。
3. Ivan 然后与一个 DB2 数据库连接，该数据库将被进行编目以驻留在 SRV3 上:

```
DB2 CONNECT to remotedb user Ivan using johnpw
```

4. Ivan 的机器验证用户名和密码。用于查找此信息的 API 首先搜索本地机器（Ivan），然后搜索域控制器（DC1），最后尝试搜索任何可信域。在 DC1 上找到用户名 Ivan。
5. 然后，Ivan 的机器用 DC1 来验证用户名和密码。
6. 然后 SRV3:
   a. 确定可识别 Ivan 的位置。
   b. 通过询问 DC1 来了解 Ivan 是否是管理员。
   c. 通过询问 DC1 来枚举所有 Ivan 的组。

注: 在尝试连接 DB2 数据库之前，确保已启动“DB2 安全服务”。“安全服务”是作为 Windows 安装的一部分安装的。然后安装 DB2 并“注册”为 Windows NT 服务，但是它不会自动启动。要启动“DB2 安全服务”，输入 `NET START DB2NTSECSERVER` 命令。

相关概念:

• 第 333 页的『DB2 Windows NT 版和 Windows NT 安全性简介』

具有客户机认证和 Windows 9x 客户机的 DB2 Windows NT 版方案

1. 管理员 Dale 登录 SRV3，并将数据库实例的认证更改为“客户机”：

```
db2 update dbm cfg using authentication client
db2stop
 db2start
```

2. Ivan 在 Windows 9x 客户机上登录 DC1 域（即，他在 DC1 SAM 数据库中是可识别的）。
3. Ivan 然后与一个 DB2 数据库连接，该数据库将被进行编目以驻留在 SRV3 上:

```
db2 connect to remotedb user Ivan using johnpw
```

4. Ivan 的 Windows 9x 机器不能验证用户名和密码。因此，假设该用户名和密码有效。
5. 然后 SRV3:
   a. 确定可识别 Ivan 的位置。

b. 通过询问 DC1 来了解 Ivan 是否是管理员。
c. 通过询问 DC1 来枚举所有 Ivan 的组。

注: 因为 Windows 9x 客户机不能验证给定用户名和密码，所以在 Windows 9x 中的客户机认证本质上是不安全的。但是，如果 Windows 9x 机器可存取 Windows NT 安全性提供程序，则可将 Windows 9x 系统配置为验证传递（Pass-Through）登录，以强制执行一些安全性措施。有关如何以此方式配置 Windows 9x 系统的详细信息，请参阅 Microsoft 的 Windows 9x 文档。

相关概念:

- 第 333 页的『DB2 Windows NT 版和 Windows NT 安全性简介』
- 第 336 页的『对全局组的支持（在 Windows 上）』

对全局组的支持（在 Windows 上）

DB2® 通用数据库（DB2 UDB）也支持全局组。为使用全局组，您必须将全局组包括在本地组中。当 DB2 UDB 枚举某个人所属的所有组时，它也列出用户间接所属的本地组（由于在一个全局组中，而该全局组本身是一个或多个本地组的成员）。

可在两种可能的方案中使用全局组:

- 包括在本地组中。必须对此本地组授予许可权。
- 包括在域控制器上。必须对此全局组授予许可权。

相关概念:

- 第 337 页的『Windows NT 上的组和用户认证』

将备份域控制器与 DB2 UDB 配合使用

过程:

如果用于 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）的服务器也充当备份域控制器，则当您配置 DB2 UDB 以使用备份域控制器时，可改进 DB2 UDB 性能并减少网络通信。

通过设置 DB2DMNBCKCTLR 注册表变量来将备份域控制器指定为 DB2 UDB。

如果您知道 DB2 UDB 服务器是其备份域控制器的域的名称，则使用:

```
db2dmnbckctlr=<domain_name>
```

其中 domain_name 必须是大写的。

要让 DB2 UDB 确定本地机器是其备份域控制器的域，则使用:

```
DB2DMNBCKCTLR=?
```

注: 缺省情况下，DB2 UDB 不使用现有备份域控制器，原因是备份域控制器可能与主域控制器不同步，从而导致安全漏洞。当更新了主域控制器的安全性数据库，但没有将该更改传播至备份域控制器时，则域控制器可能脱离同步。若存在网络等待时间，或者若计算机浏览器服务未运行，也可能发生这种情况。

相关任务:

- 第 338 页的『在备份域控制器上安装 DB2』

DB2 Windows NT 版的用户认证

对于 Windows NT 用户而言，用户认证可能会产生问题，原因在于操作系统认证的方法。本节描述了 DB2 Windows NT 版下的用户认证的一些注意事项:

- 『DB2 Windows NT 版用户名和组名限制』
- 第 338 页的『DB2 Windows NT 版安全服务』
- 第 338 页的『在备份域控制器上安装 DB2』
- 第 339 页的『使用组和域安全性的 DB2 Windows NT 版认证』

DB2 Windows NT 版用户名和组名限制

下面是此环境中的限制:

- 在 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）中，用户名和组名最长为 30 个字符。
- Windows NT 下的用户名不区分大小写；但是，密码区分大小写。
- 用户名和组名可以是大写字符与小写字符的组合。但是，当在 DB2 UDB 中使用时，它们通常被转换为大写字符。例如，若连接数据库并创建表 schema1.table1，则此表作为 SCHEMA1.TABLE1 存储在数据库中。（若您希望使用小写对象名，则从命令行处理器发出命令并将对象名括在引号中，或使用第三方 ODBC 前端工具。）
- 用户不能属于 64 个组以上。
- DB2 UDB 支持单一名称空间。即，在可信域环境中运行时，相同名称的用户帐户不应在多个域中存在或在服务器的本地 SAM 和另一域中存在。

相关概念:

- 第 337 页的『Windows NT 上的组和用户认证』
- 第 338 页的『Windows NT 上的域之间的信赖关系』

Windows NT 上的组和用户认证

在 Windows® NT 上，用户是通过使用称为“用户管理器”的 Windows NT® 管理工具创建用户帐户来定义的。

包含其它帐户（又称为成员）的帐户是一个组。组允许 Windows NT 管理员同时将权限和许可权授予组内的各个用户而不必分别维护每个用户。与用户帐户一样，组是在“安全性存取管理器”（SAM）数据库中定义并维护的。

有两种类型的组:

- 本地组。本地组可以包括在本地帐户数据库中创建的用户帐户。如果本地组在域中的某台机器上，则本地组还可以包含 Windows NT 域中的域帐户和组。如果本地组是在工作站上创建的，则它是特定于该工作站的。
- 全局组。全局组只存在于域控制器上，且包含域的 SAM 数据库中的用户帐户。即，全局组只包含在其上创建它的域中的用户帐户；它不能包含任何其它组作为成员。可在全局组自己的域中的服务器和工作站以及可信域中使用全局组。

相关概念:

- 第 338 页的『Windows NT 上的域之间的信赖关系』
- 第 336 页的『对全局组的支持（在 Windows 上）』

相关任务:

- 第 339 页的『使用组和域安全性的 DB2 Windows NT 版认证』

相关参考:

- 第 337 页的『DB2 Windows NT 版用户名和组名限制』

Windows NT 上的域之间的信赖关系

信赖关系是两个域之间的管理和通信链路。两个域之间的信赖关系允许在定义用户帐户的域之外的域中使用这些帐户和全局组。共享帐户信息以验证驻留在可信域中未经认证的用户帐户和全局组的权限和许可权。信赖关系通过将两个或多个域组合成单个管理单元来简化用户管理。

信赖关系中有两个域:

- 信赖域。此域信赖另一个域以认证它们的用户。
- 可信域。此域代表（信赖）另一个域认证用户。

信赖关系是不可传递的。这表示需要在两个域之间在每个方向上建立显式信赖关系。例如，信赖域可能不一定是可信域。

相关概念:

- 第 337 页的『Windows NT 上的组和用户认证』
- 第 336 页的『对全局组的支持（在 Windows 上）』

相关参考:

- 第 337 页的『DB2 Windows NT 版用户名和组名限制』

DB2 Windows NT 版安全服务

在 DB2® 通用数据库（DB2 UDB）中，已将用户名和密码的认证集成到 “DB2 系统控制器” 中。仅当将客户机与配置了 CLIENT 认证的服务器相连接时才要求 “安全服务”。

相关概念:

- 第 333 页的『DB2 Windows NT 版和 Windows NT 安全性简介』

在备份域控制器上安装 DB2

过程:

在 Windows NT 4.0 环境中，可以在主或备份控制器上认证用户。在每个站点都有一个中央主域控制器和一个或多个备份域控制器（BDC）的大型分布式 LAN 中，此特征非常重要。可以在用户的站点上的备份域控制器上认证用户，而无需呼叫主域控制器（PDC）来进行认证。

在这种情况下，具有备份域控制器的优点是: 认证用户的速度更快，LAN 不会象在没有 BDC 的情况下那么拥挤。

在下列条件下，会在 BDC 上发生认证:

- DB2 Windows NT 版服务器已安装在备份域控制器上。
- 已适当地设置了 DB2DMNBCKCTLR 概要文件注册表变量。

若 DB2DMNBCKCTLR 概要文件注册表变量未设置或设置为空白，则 DB2 Windows NT 版在主域控制器上执行认证。

DB2DMNBCKCTLR 的有效声明设置只有 “?” 或域名。

若 DB2DMNBCKCTLR 概要文件注册表变量设置为问号（DB2DMNBCKCTLR=?），则在下列条件下，DB2 Windows NT 版将在备份域控制器上执行其认证:

- cachedPrimaryDomain 是对此机器所属的域的名称设置的注册表值。（可以在 **HKEY_LOCAL_MACHINE —> Software —> Microsoft —> Windows NT —> Current Version —> WinLogon** 下面找到此设置。）
- “服务器管理器” 显示备份域控制器活动且可用。（即此机器的图标不是灰色的。）
- DB2 Windows NT 服务器的注册表指示该系统是指定的域上的备份域控制器。

DB2DMNBCKCTLR=? 设置在正常情况下可以起作用；但并非在所有环境下都能起作用。域上提供的关于服务器的信息是动态的，必须运行 “计算机浏览器” 才能保持此信息准确和最新。大型 LAN 可能未运行 “计算机浏览器”，因而 “服务器管理器” 的信息可能不是最新的。在这种情况下，还有一种方法来告知 DB2 Windows NT 版在备份域控制器上执行认证：设置 DB2DMNBCKCTLR=*xxx*，其中 *xxx* 是 DB2 服务器的 Windows NT 域名。借助此设置，在下列条件下，认证将发生在备份域控制器上:

- cachedPrimaryDomain 是对此机器所属的域的名称设置的注册表值。（可以在 **HKEY_LOCAL_MACHINE —> Software —> Microsoft —> Windows NT —> Current Version —> WinLogon** 下面找到此设置。）
- 该机器配置成指定域的备份域控制器。（若该机器设置为另一个域的备份域控制器，则此设置将导致错误。）

相关任务:

- 第 336 页的『将备份域控制器与 DB2 UDB 配合使用』

使用组和域安全性的 DB2 Windows NT 版认证

过程:

授予特权或定义权限级别时，DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）允许您指定本地组或全局组。如果用户的帐户是在本地或全局组中显式定义的，或者是作为定义为本地组成员的全局组成员隐式定义的，则确定用户是组的成员。

DB2 Windows NT 版支持下列类型的组:

- 本地组
- 全局组
- 作为本地组成员的全局组

DB2 Windows NT 版使用用户所在的安全性数据库来枚举本地组和全局组，该用户是这些组的成员。DB2 UDB 提供了一种覆盖，无论用户帐户的位置如何，强制组枚举出现在安装 DB2 UDB 的本地 Windows NT 服务器上。此覆盖可以使用下列命令实现:

– 对于全局设置:

```
db2set -g DB2_GRP_LOOKUP=local
```

– 对于实例设置:

```
db2set -i <instance name> DB2_GRP_LOOKUP=local
```

发出此命令之后，必须停止并启动 DB2 UDB 实例以使更改生效。然后，创建本地组并将域帐户或全局组包括在本地组中。

要查看设置的所有 DB2 概要文件注册表变量，输入

```
db2set -all
```

如果 DB2_GRP_LOOKUP 概要文件注册表变量设置为 local，则 DB2 UDB 只尝试在本地机器上查找用户。若在本地机器上找不到该用户，或未将该用户定义为本地或全局组的成员，则认证失败。DB2 **并不**尝试在该域中的另一机器上或在域控制器上查找该用户。

若 DB2_GRP_LOOKUP 概要文件注册表变量未设置，则:

1. DB2 UDB 首先尝试在同一机器上查找用户。
2. 若用户名是以本地方式定义的，则以本地方式认证该用户。
3. 如果以本地方式找不到该用户，则 DB2 UDB 尝试在它的域上查找该用户名，然后在可信域上查找。

如果 DB2 UDB 在作为资源域中主域控制器或备份域控制器的机器上运行，则它能够定位任何可信域中的任何域控制器。这样的原因是: 可信域中的备份域控制器的域的名称仅在域控制器上才能被识别。

如果 DB2 UDB 未在域控制器上运行，则应发出:

```
db2set -g DB2_GRP_LOOKUP=DOMAIN
```

此命令告诉 DB2 UDB 在其自己的域中使用域控制器来查找帐户域中域控制器的名称。即，当 DB2 UDB 发现特定用户帐户是在域 x 中定义的，它会将该请求发送至其自己域中的域控制器，而不是试图定位域 x 的域控制器。将找到帐户域中域控制器的名称，并返回至运行 DB2 UDB 的机器。此方法有两个优点:

1. 当主域控制器不可用时将会查找备份域控制器。
2. 当主域控制器地理上处于远程位置时备份域控制器将会是关闭的。

相关概念:

• 第 337 页的『Windows NT 上的组和用户认证』

使用排序域列表的认证

可信域林中可多次定义用户标识。可信域林是通过网络相关的域集合。某一域上的用户拥有的用户标识可能与不同域上另一用户的用户标识相同。尝试执行任何下列操作时，可能会引起麻烦:

- 认证拥有相同用户标识但位于不同域中的多个用户。
- 组查找，以便根据组授予和撤销特权。
- 验证密码。
- 控制网络流量。

过程:

为避免域林中可能会有多个用户具有相同用户标识而引起的麻烦，应使用采用 **db2set** 和注册表变量 DB2DOMAINLIST 定义的排序域列表。设置顺序时，用逗号隔开列表中将包含的域。认证用户时，必须就搜索域的顺序作出明智的决定。

如果要认证域列表下面域中出现的用户标识以便访问，则必须重命名它们。

可通过域列表控制访问。例如，如果用户的域不在列表中，则用户不能连接。

注: 仅当数据库管理器配置中设置 CLIENT 认证时，DB2DOMAINLIST 注册表变量才有效，如果 Windows NT 域环境中需要从 Windows NT 桌面单一注册，则需要 DB2DOMAINLIST 注册表变量。

相关概念:

- 第 333 页的『DB2 Windows NT 版和 Windows NT 安全性简介』

DB2 Windows NT 版对域安全性的支持

下列示例说明了 DB2 Windows NT 版对域安全性的支持。在第一个示例中，因为用户名和本地组在相同域上，所以可以进行连接。在第二个示例中，因为用户名与本地或全局组在不同的域上，所以不能进行连接。

成功连接的示例: 在以下方案中，因为用户名与本地或全局组在同一域上，所以可以进行连接。

注意，用户名与本地或全局组无需在运行数据库服务器的域上定义，但它们必须在同一个域上。

表 52. 使用域控制器的成功连接

Domain1	Domain2
存在与 Domain2 的信赖关系。	• 存在与 Domain1 的信赖关系。 • 定义了本地或全局组 grp2。 • 定义了用户名 id2。 • 用户名 id2 是 grp2 的一部分。

DB2 服务器在此域中运行。从它发出下列 DB2 命令:
`REVOKE CONNECT ON db FROM public`
`GRANT CONNECT ON db TO GROUP grp2`
`CONNECT TO db USER id2`	

扫描本地或全局域，但找不到 id2。扫描域安全性。	
	在此域中找到用户名 id2。DB2 获得关于此用户名的其它信息（即，它是组 grp2 的一部分）。
因为用户名与本地或全局组在同一域上，所以可以进行连接。	

相关概念:

- 第 337 页的『Windows NT 上的组和用户认证』

相关任务:

- 第 339 页的『使用组和域安全性的 DB2 Windows NT 版认证』

附录 G. 使用 “Windows 性能监视器”

Windows 性能监视器简介

当使用 DB2® 通用数据库（DB2 UDB）Windows® 版时，可以使用下列工具来监视性能:

- **DB2 性能专家**

  用于多平台的 DB2 性能专家 1.1 版根据与 DB2 UDB 性能有关的信息来合并、报告、分析并建议自我管理和资源调整更改。

- **DB2 UDB 健康中心**

  “健康中心”的功能提供不同方法以使用与性能有关的信息。这些功能在某种程度上可以替换“控制中心”的性能监视器能力。

- **Windows 性能监视器**

  “Windows 性能监视器”允许监视数据库和系统性能，从向系统注册的任何性能数据提供程序检索信息。Windows 还提供有关机器运行各方面的性能信息数据，包括:

  – CPU 的使用
  – 内存使用率
  – 磁盘活动
  – 网络活动

相关任务:

- 第 343 页的『向 Windows 性能监视器注册 DB2』
- 第 344 页的『启用对 DB2 性能信息的远程存取』
- 第 345 页的『显示 DB2 UDB 和 DB2 Connect 性能值』
- 第 346 页的『存取远程 DB2 UDB 性能信息』
- 第 346 页的『重新设置 DB2 性能值』

相关参考:

- 第 345 页的『Windows 性能对象』

向 Windows 性能监视器注册 DB2

过程:

安装程序自动向“Windows 性能监视器”注册 DB2。

要使“Windows 性能监视器”可存取 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）和 DB2 Connect 性能信息，必须注册“DB2 Windows 版性能计数器”的 DLL。这也就允许任何其它使用 Win32 性能 API 的 Windows 应用程序获取性能数据。

要安装并向“Windows 性能监视器”注册“DB2 Windows 版性能计数器”的 DLL（DB2Perf.DLL），输入:

```
db2perfi -i
```

注册该 DLL 的同时会在注册表的 services 选项中创建一个新键。其中一个条目给出 DLL 的名称，它提供计数器支持。另外三个条目给出该 DLL 中提供的函数的名称。这些函数包括:

- **Open**

  在一个进程中系统首次装入该 DLL 时调用。

- **Collect**

  从 DLL 请求性能信息时调用。

- **Close**

  卸装 DLL 时调用。

相关参考:

- 『db2perfi - Performance Counters Registration Utility Command』（*Command Reference*）

启用对 DB2 性能信息的远程存取

过程:

如果 DB2 Windows 版工作站与其它 Windows 机器联网，可使用本节中描述的功能部件。

要从另一台 DB2 Windows 版机器查看 Windows 性能对象，必须向 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）注册管理员用户名和密码。（“Windows 性能监视器”的缺省用户名 **SYSTEM** 是 DB2 UDB 保留字，因此不能使用。）要注册该用户名，输入:

```
db2perfr -r username password
```

注: 使用的 username 必须符合 DB2 UDB 命名规则。

用户名和密码数据保存在注册表内的一个键中，并设置了安全性，它只允许管理员和 SYSTEM 帐户存取。编码该数据，以避免在注册表中存储管理员密码出现安全性问题。

注:

1. 一旦将用户名与密码一起注册到 DB2 UDB 中，甚至“性能监视器”的本地实例也将使用该用户名和密码显式登录。这就表示，如果向 DB2 UDB 注册的用户名信息不匹配，“性能监视器”的本地会话将不显示 DB2 UDB 性能信息。
2. 必须维护该用户名与密码的组合，以便与 Windows 安全性数据库中存储的用户名和密码值相匹配。如果在 Windows 安全性数据库中更改了用户名或密码，必须重新设置用于远程性能监视的用户名和密码组合。
3. 要注销，输入:

   ```
   db2perfr -u <username> <password>
   ```

相关概念:

- 第 265 页的『通用命名规则』

相关参考:

- 『db2perfr - Performance Monitor Registration Tool Command』（*Command Reference*）

显示 DB2 UDB 和 DB2 Connect 性能值

过程:

要使用“性能监视器”显示 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）和 DB2 Connect 性能值，只需从**添加至**框中选择您想要显示其值的性能计数器。此框显示性能对象的列表及其性能数据。选择一个对象，查看该对象提供的计数器列表。

一个性能对象也可以有多个实例。例如，LogicalDisk 对象提供例如“% 磁盘读时间”和“磁盘字节／秒”这样的计数器；它还为机器上的每个逻辑驱动器提供一个实例，包括“C:”和“D:”。

相关概念:

- 第 343 页的『Windows 性能监视器简介』

相关参考:

- 第 345 页的『Windows 性能对象』

Windows 性能对象

Windows 提供了下列性能对象:

- **DB2 数据库管理器**

  此对象提供了单个 Windows 实例的常规信息。受到监视的 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）实例显示为对象实例。

  由于实际原因和性能原因，每次只能从一个 DB2 UDB 实例获取性能信息。“性能监视器”显示的 DB2 UDB 实例受“性能监视器”进程中的 `db2instance` 注册表变量控制。如果您有多个 DB2 UDB 实例在同时运行，并且想查看多个实例的性能信息，对于每个要监视的 DB2 UDB 实例，将 `db2instance` 设置为相关值，然后启动一个单独的“性能监视器”会话。

  若是在运行分区数据库系统，则一次只能从一个数据库分区服务器（节点）获取性能信息。在缺省情况下，显示缺省节点（即带有逻辑端口 0 的节点）的性能信息。要查看另一节点的性能信息，必须启动“性能监视器”的一个单独会话，并将 DB2NODE 环境变量设置为要监视的节点的节点号。

- **DB2 UDB 数据库**

  此对象提供特定数据库的信息。每个当前活动的数据库都有可用的信息。

- **DB2 应用程序**

  此对象提供特定 DB2 UDB 应用程序的信息。每个当前活动的 DB2 UDB 应用程序都有可用的信息。

- **DB2 DCS 数据库**

  此对象提供特定的 DCS 数据库的信息。每个当前活动的数据库都有可用的信息。

- **DB2 DCS 应用程序**

  此对象提供特定的 DB2 DCS 应用程序的信息。每个当前活动的 DB2 DCS 应用程序都有可用的信息。

“Windows 性能监视器”将列示的对象取决于 Windows 机器上安装了什么内容以及哪些应用程序是活动的。例如，若安装了 DB2 UDB 且已启动数据库管理器，将列出“DB2 数据库管理器”对象。如果此机器上还有一些 DB2 UDB 数据库和应用程序当前是活动的，也将列出那些 DB2 数据库和 DB2 应用程序对象。如果将 Windows 系统用作 DB2 Connect 网关，且有一些 DCS 数据库和应用程序当前是活动的，将列出 DB2 DCS 数据库和 DB2 DCS 应用程序对象。

相关概念:

- 『DB2 注册表和环境变量』（*《管理指南: 性能》*）

存取远程 DB2 UDB 性能信息

过程:

前面已讨论过允许远程存取“DB2 性能信息”。在**添加至**框中选择要监视的另一台计算机。这将显示一个列表，列出该计算机上所有可用的性能对象。

为了能够监视远程计算机上的 DB2 性能对象，安装在那台计算机上的 DB2 UDB 或 DB2 Connect 代码的级别必须是版本 6 或更高版本。

相关概念:

- 第 343 页的『Windows 性能监视器简介』

重新设置 DB2 性能值

过程:

当应用程序调用 DB2 监视器 API 时，由于启动了 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）服务器，因此返回的信息通常是累积值。但它经常可用于:

- 复位性能值
- 运行测试
- 再次复位值
- 重新运行测试

要复位数据库性能值，使用 **db2perfc** 程序。输入:

```
db2perfc
```

缺省情况下，此命令将复位所有活动 DB2 UDB 数据库的性能值。但也可指定要复位的数据库的列表。还可使用 -d 选项指定应复位 DCS 数据库的性能值。例如:

```
db2perfc
db2perfc dbalias1 dbalias2 ... dbaliasn

db2perfc  -d
db2perfc  -d  dbalias1 dbalias2 ... dbaliasn
```

第一个示例复位所有活动 DB2 UDB 数据库的性能值。第二个示例复位特定 DB2 UDB 数据库的性能值。第三个示例复位所有活动的 DB2 DCS 数据库的性能值。最后一个示例复位特定的 DB2 DCS 数据库的性能值。

db2perfc 程序复位当前存取相关 DB2 UDB 服务器实例（即在其中运行 **db2perfc** 的会话中的 DB2INSTANCE 中的服务器实例）的数据库性能信息的所有程序的值。

当执行 **db2perfc** 命令时，调用 **db2perfc** 也能复位远程存取 DB2 UDB 性能信息的人员所见到的值。

注: 有一个 DB2 UDB API sqlmrset，它允许应用程序复位其在本地（而非全局）看到的特定数据库的值。

相关参考:

- 『db2ResetMonitor - Reset Monitor』（*Administrative API Reference*）
- 『db2perfc - Reset Database Performance Values Command』（*Command Reference*）

附录 H. 使用 Windows 数据库分区服务器

在 Windows 环境中更改配置特征时，使用特定的实用程序执行所涉及的任务。

此处讨论的实用程序为:

- 『列出实例中的数据库分区服务器』
- 『将数据库分区服务器添加至实例（Windows）』
- 第 351 页的『更改数据库分区（Windows）』
- 第 352 页的『从实例中删除数据库分区（Windows）』

列出实例中的数据库分区服务器

过程:

在 Windows 上，使用 **db2nlist** 命令来获取参与实例的数据库分区服务器的列表。

按如下所示使用该命令:

```
db2nlist
```

当按以上所示使用此命令时，缺省实例是当前实例（由 DB2INSTANCE 环境变量设置）。要指定特定的实例，可使用以下命令指定该实例:

```
db2nlist /i:instName
```

其中 `instName` 是想要的特定实例名。

作为选项，也可使用以下命令请求每个分区服务器的状态:

```
db2nlist /s
```

每个数据库分区服务器的状态可能是：正在启动、正在运行、正在停止或已停止。

相关任务:

- 第 349 页的『将数据库分区服务器添加至实例（Windows）』
- 第 351 页的『更改数据库分区（Windows）』
- 第 352 页的『从实例中删除数据库分区（Windows）』

将数据库分区服务器添加至实例（Windows）

过程:

在 Windows 上，使用 **db2ncrt** 命令来将数据库分区服务器（节点）添加至实例。

注: 若此实例已包含数据库，则不要使用 **db2ncrt** 命令。而使用 **db2start addnode** 命令。这确保可正确地将该数据库添加至新的数据库分区服务器。**不要编辑** `db2nodes.cfg` 文件，因为更改文件可能导致分区数据库系统中的不一致性。

该命令具有下列必需参数:

```
db2ncrt /n:node_number
          /u:username,password
/p:logical_port
```

- /n:

  用于标识数据库分区服务器的唯一节点号。该号码可以是按递升顺序排列的 1 到 999 中的任何一个值。

- /u:

  DB2 服务的登录帐户名和密码。

- /p:logical_port

  用于数据库分区服务器的逻辑端口号（若逻辑端口不是零（0））。若不指定，则将逻辑端口号指定为 0。

仅当在机器上创建第一个节点时，逻辑端口参数才是可选的。若创建逻辑节点，则必须指定此参数并选择一个不在使用中的端口号。有几项限制:

- 在每台机器上，都必须要有一个逻辑端口为 0 的数据库分区服务器。
- 端口号不能超过 `%SystemRoot%\system32\drivers\etc` 目录中的 services 文件中为 FCM 通信保留的端口范围。例如，若为当前实例保留四个端口，则最大端口号将是 3（端口 1、2 和 3; 端口 0 用于缺省逻辑节点）。端口范围是在将 **db2icrt** 命令与 `/r:base_port, end_port` 参数配合使用时定义的。

还有几个可选的参数:

- /g:network_name

  指定数据库分区服务器的网络名。若不指定此参数，则 DB2 使用它在系统上检测到的第一个 IP 地址。

  若机器上有多个 IP 地址，且您想要对数据库分区服务器指定特定 IP 地址，则使用此参数。可输入使用网络名或 IP 地址的 *network_name* 参数。

- /h:host_name

  FCM 用于内部通信的 TCP/IP 主机名（若该主机名不是本地主机名）。若在远程机器上添加数据库分区服务器，则此参数是必需的。

- /i:instance_name

  实例名; 缺省值为当前实例。

- /m:machine_name

  该节点所驻留的 Windows 工作站的计算机名称; 缺省名称是本地机器的计算机名称。

- /o:instance_owning_machine

  该实例拥有的机器的计算机名称; 缺省值是本地机器的计算机名称。当在任何非该实例拥有的机器上调用 **db2ncrt** 命令时，此参数是必需的。

例如，若要向实例拥有的机器 MYMACHIN 上的实例 TESTMPP 添加新的数据库分区服务器（以运行多逻辑节点），且您想要让这个新节点成为使用逻辑端口 1 的节点 2，则输入:

```
db2ncrt /n:2 /p:1 /u:my_id,my_pword /i:TESTMPP
      /M:TEST /o:MYMACHIN
```

相关参考:

- 『db2start - Start DB2 Command』（*Command Reference*）
- 『db2icrt - Create Instance Command』（*Command Reference*）
- 『db2ncrt - Add Database Partition Server to an Instance Command』（*Command Reference*）

更改数据库分区（Windows）

过程:

在 Windows 上，使用 **db2nchg** 命令来执行下列各项:

- 将数据库分区从一台机器移至另一台机器。
- 更改机器的 TCP/IP 主机名。

  若计划使用多个网络适配器，则必须使用此命令来为 *db2nodes.cfg* 文件中的“netname”字段指定 TCP/IP 地址。
- 使用另一逻辑端口号。
- 对数据库分区服务器（节点）使用另一名称。

该命令具有下列必需参数:

```
db2nchg /n:node_number
```

参数 /n: 是您想要更改的数据库分区服务器的配置的节点号。此参数是必需的。

可选的参数包括:

- /i:instance_name

  指定此数据库分区服务器所参与的实例。若不指定此参数，则缺省值是当前实例。
- /u:username,password

  更改 DB2 Universal Database™（DB2 UDB）（DB2 通用数据库）服务的登录帐户名和密码。若不指定此参数，则登录帐户和密码保持不变。
- /p:logical_port

  更改数据库分区服务器的逻辑端口。若将数据库分区服务器移至另一机器，则必须指定此参数。若不指定此参数，则逻辑端口号保持不变。
- /h:host_name

  更改 FCM 用于内部通信的 TCP/IP 主机名。若不指定此参数，则主机名不更改。
- /m:machine_name

  将数据库分区服务器移至另一机器。仅当实例中没有现有数据库时，才可移动数据库分区服务器。
- /g:network_name

更改数据库分区服务器的网络名。

若机器上有多个 IP 地址，且您想要对数据库分区服务器使用特定 IP 地址，则使用此参数。可使用网络名或 IP 地址输入 network_name。

例如，要将分配给节点 2（它参与实例 TESTMPP）的逻辑端口更改为使用逻辑端口 3，则输入以下命令:

```
db2nchg /n:2 /i:TESTMPP /p:3
```

DB2 UDB 允许存取远程机器上实例级别的 DB2 UDB 注册表变量。当前，以三种不同的级别存储 DB2 UDB 注册表变量: 机器或全局级别、实例级别和节点级别。对于以实例级别（包括节点级别）存储的注册表变量，可使用 DB2REMOTEPREG 将其重定向到其它机器。如果设置了 DB2REMOTEPREG，DB2 UDB 将从 DB2REMOTEPREG 指向的机器存取 DB2 UDB 注册表变量。db2set 命令会出现为:

```
db2set DB2REMOTEPREG=<remote workstation>
```

其中 <remote workstation> 是远程工作站名称。

注: 应该小心设置此选项，因为所有的 DB2 UDB 实例概要文件和实例列表都将位于此指定的远程机器名上。

此功能可与设置 DBINSTPROF 结合起来使用，以指向同一台机器上包含该注册表的远程 LAN 驱动器。

相关概念:

- 『DB2 注册表和环境变量』（《*管理指南: 性能*》）

相关参考:

- 『db2nchg - Change Database Partition Server Configuration Command』（*Command Reference*）

从实例中删除数据库分区（Windows）

过程:

在 Windows 上，使用 **db2ndrop** 命令从没有数据库的实例中删除数据库分区服务器（节点）。若删除了一个数据库分区服务器，则它的节点号可以再次用于新的数据库分区服务器。

当从实例中删除数据库分区服务器时务必小心。若从该实例删除实例拥有的数据库分区服务器节点 0，该实例将变成不可用的。若要删除该实例，使用 **db2idrop** 命令。

注: 若此实例包含数据库，则不要使用 **db2ndrop** 命令。而使用 **db2stop drop nodenum** 命令。这确保可正确地从数据库分区中除去该数据库。**不要编辑** db2nodes.cfg 文件，因为更改文件可能导致分区数据库系统中的不一致性。

若要从运行多逻辑节点的机器中删除被分配了逻辑端口 0 的节点，则在删除被分配了逻辑端口 0 的节点之前，必须删除分配给其它逻辑端口的所有其它节点。每个数据库分区服务器都必须带有一个分配给逻辑端口 0 的节点。

该命令具有下列参数:

```
db2ndrop /n:node_number /i:instance_name
```

- /n:

  用于标识数据库分区服务器的唯一节点号。这是一个必需参数。该号码可以是按递升顺序排列的 0 到 999 中的任何一个值。记住节点 0 表示该实例拥有的机器。

- /i:instance_name

  实例名。这是一个可选的参数。若不给出，则缺省值是当前实例（由 DB2INSTANCE 注册表变量设置）。

相关概念:

- 『DB2 注册表和环境变量』（*《管理指南: 性能》*）

相关参考:

- 『db2stop - Stop DB2 Command』（*Command Reference*）
- 『db2idrop - Remove Instance Command』（*Command Reference*）
- 『db2ndrop - Drop Database Partition Server from an Instance Command』（*Command Reference*）

附录 I. 配置多逻辑节点

何时使用多逻辑节点

通常，将 DB2® 通用数据库（DB2 UDB）企业服务器版配置为对每台机器分配一个数据库分区服务器。但是，在几种情况下，在同一机器上运行数个数据库分区服务器很有好处。这意味着配置包含的节点数可以多于机器数。在这些情况下，我们称该机器正在运行*多逻辑节点*（若这些逻辑节点参与*同一*实例的话）。若它们参与不同的实例，则此机器*不*主管多逻辑节点。

借助多逻辑节点支持，您可以从三种类型的配置中进行选择:

- 标准配置，即每台机器只有一个数据库分区服务器。
- 多逻辑节点配置，即机器有多个数据库分区服务器。
- 在数台机器的每一台上运行数个逻辑节点的配置。

当系统在具有对称多处理器（SMP）体系结构的机器上运行查询时，使用多逻辑节点的配置非常有用。万一有机器故障，在机器上配置多逻辑节点的能力也是很有用的。若一台机器出现故障（导致其上的数据库分区服务器失败），则可以在另一机器上使用 DB2START NODENUM 命令重新启动数据库分区服务器。这确保用户数据保持可用。

另一个好处是多逻辑节点可利用 SMP 硬件配置。另外，因为数据库分区较小，所以当执行诸如备份和复原数据库和表空间以及创建索引之类的任务时，可以获得较佳的性能。

相关任务:

- 第 355 页的『配置多逻辑节点』

相关参考:

- 『db2start - Start DB2 Command』（*Command Reference*）

配置多逻辑节点

过程:

可以使用这两种方式中的一种方式配置多逻辑节点:

- 在 db2nodes.cfg 文件中配置逻辑节点（数据库分区）。然后，可使用 DB2START 命令或它的相关 API 启动所有逻辑节点和远程节点。

  注: 对于 Windows NT，若系统中没有数据库，则必须使用 *db2ncrt* 来添加节点，否则，若有一个或多个数据库，则应使用 DB2START ADDNODE 命令。在 Windows NT 中，绝对不应手工编辑 *db2nodes.cfg* 文件。
- 在另一个处理器上重新启动一个逻辑节点，其它逻辑数据库分区（节点）已在该处理器上运行。这允许您覆盖在 db2nodes.cfg 中为逻辑数据库分区指定的主机名和端口号。

要在 db2nodes.cfg 中配置一个逻辑数据库分区（节点），您必须在该文件中建立一个条目以便为该节点分配一个逻辑端口号。以下是应使用的语法:

```
nodenumber hostname logical-port netname
```

注: 对于 Windows NT，若系统中没有数据库，则必须使用 *db2ncrt* 来添加节点，否则，若有一个或多个数据库，则应使用 DB2START ADDNODE 命令。在 Windows NT 中，绝对不应手工编辑 *db2nodes.cfg* 文件。

Windows NT 上 *db2nodes.cfg* 文件的格式与 Unix 上同一文件的格式并不相同。在 Windows NT 上，列格式是:

```
nodenumber hostname computername logical_port netname
```

使用主机名的全限定名称。/etc/hosts 文件也应该使用全限定名称。如果未在 db2nodes.cfg 文件和 /etc/hosts 文件中使用全限定名称，则可能接收到错误消息 SQL30082N RC=3。

必须确保在 etc 目录的 services 文件中为 FCM 通信定义了足够的端口。

相关概念:

- 第 355 页的『何时使用多逻辑节点』

相关任务:

- 第 135 页的『更改节点和数据库配置文件』
- 第 31 页的『创建节点配置文件』

相关参考:

- 『db2start - Start DB2 Command』（*Command Reference*）
- 『db2ncrt - Add Database Partition Server to an Instance Command』（*Command Reference*）

附录 J. 扩展“控制中心”

引入“控制中心”的插件体系结构

可使用新的*插件*体系结构来扩展“DB2 通用数据库控制中心”，以提供附加功能。

*插件*体系结构能够向“控制中心”弹出菜单中的给定对象添加项、将对象添加至“控制中心”树以及向工具栏添加新按钮。这些工具在交付时附带有一组 Java™ 接口，您必须实现这些接口。这些接口用来与“控制中心”通信（包含那些附加的操作）。

在“控制中心”工具启动时，装入扩展插件（db2plug.zip）。根据压缩文件的大小，可能会延长工具的启动时间。但是，对于大多数用户，插件压缩文件属于小文件，因此影响也应最小。

相关概念:

- 第 358 页的『编译和运行示例插件』
- 第 359 页的『编写作为“控制中心”扩展的插件』
- 第 357 页的『“控制中心”插件开发者准则』

“控制中心”插件开发者准则

由于 db2plug.zip 文件中可包含多个插件，为“控制中心”创建插件时，插件开发者应遵循下列准则:

- 使用 Java™ 包来确保插件类具有唯一名称。遵循 Java 包命名约定。在程序包名称前加上因特网域的反转名称作为前缀（例如，com.companyname）。所有程序包名称或至少其唯一前缀应是小写字母。
- db2plug.zip 应安装在 `sqllib` 目录下的工具目录中。在 V8 之前，需要将 db2plug.zip 安装在 `sqllib` 目录下的 `cc` 目录中。
- 如果为“控制中心”创建插件时 db2plug.zip 文件已存在，则应将插件类添加至现有 db2plug.zip。不应使用自己的 db2plug.zip 文件覆盖现有 db2plug.zip 文件。要将插件添加至现有 db2plug.zip，应使用以下 zip 命令:

  ```
  zip -r0 db2plug.zip com\companyname\myplugin\*.class
  ```

  其中插件程序包名为 com.companyname.myplugin
- 当启动“控制中心”时，就会装入 db2plug.zip 中的所有类。db2plug.zip 文件应包含所有的 CCExtension 类文件和在 com.ibm.db2.tools.cc.navigator 程序包中扩展或实现的类。db2plug.zip 不必包含其它不被这些类直接使用的类。可将它们存储在单独的 jar 文件中以便在启动“控制中心”时将对性能的影响减至最小。如果其它类的数量很大，这是个好主意。应将 jar 文件放在 `sqllib` 目录下的 `tools` 目录中。当使用 **db2cc** 命令来启动“控制中心”时，会自动将 jar 文件包括在 *classpath* 中。
- 实现 CCObject 的插件类应提供无自变量的缺省构造函数以允许“控制中心”对 Class.newInstance() 的调用。

- 可能避免使用内部类的地方。一般来说，不将实现 CCTreeObject 以在“控制中心”创建新插件对象的插件类说明为内部类。这样可阻止“控制中心”将这些类实例化。
- 用 **db2cc -tf filename** 测试插件是否正确装入。这样将在指定文件名中装入“控制中心”跟踪信息。如果不提供完整路径名，将把跟踪文件写入到 sqllib 中的工具目录中。与插件相关的跟踪语句将包含“插件”一词。可通过寻找含有“插件装入器”文本的行来查看类是否已装入。

相关概念:

- 第 358 页的『编译和运行示例插件』
- 第 359 页的『编写作为“控制中心”扩展的插件』

相关参考:

- 『db2cc - Start Control Center Command』（*Command Reference*）

编译和运行示例插件

下面几节及其相应的插件样本程序演示了“控制中心插件”功能：Example1.java、Example2.java、Example3.java、Example3Folder.java 和 Example3Child.java。这些示例 java 文件是随 “DB2® 应用程序开发客户机”一起安装的。在 Windows® 平台上，这些样本程序在 DRIVE:\sqllib\samples\java\plugin 中，其中 DRIVE: 表示安装 DB2 的目录。在 UNIX® 平台上，这些样本在 /u/db2inst1/sqllib/samples/java/plugin 中，其中 /u/db2inst1 表示安装 DB2 的目录。

注: 插件样本程序中可能包含此处未反映的更新。当与此处显示的有所不同时，应考虑示例代码和 java 文档的最新信息。

要运行示例插件，根据 Java™ 归档文件的规则，必须压缩（ZIP）扩展类文件。ZIP 文件（db2plug.zip）必须放在 *classpath* 中。在 Windows 操作系统上，将 db2plug.zip 放在 DRIVE:\sqllib\tools 目录中，其中 DRIVE: 表示安装 DB2 的驱动器。在 UNIX 平台上，将 db2plug.zip 放在 /u/db2inst1/sqllib/tools 目录中，其中 /u/db2inst1 表示安装 DB2 的目录。

注: **db2cc** 命令将 *classpath* 设置为指向工具目录中的 db2plug.zip。

由于可能与其它示例存在冲突，因此这些示例（Example3、Example3Folder 和 Example3Child 除外，它们是放在一块的）不应压缩至同一 db2plug.zip 中。

要编译其中任何一个示例 java 文件，必须在 *classpath* 中包括下列各项:

- 在 Windows 平台上，使用:
  - DRIVE: \sqllib\java\Common.jar
  - DRIVE: \sqllib\tools\db2navplug.jar

  其中 DRIVE 表示安装 DB2 的驱动器。
- 在 UNIX 平台上，使用:
  - /u/db2inst1/sqllib/java/Common.jar
  - /u/db2inst1/sqllib/tools/db2navplug.jar

其中 /u/db2inst1 表示安装 DB2 的目录。

创建 db2plug.zip 以包括通过编译示例 java 文件生成的所有类。不应压缩该文件。例如，发出以下命令:

```
zip -r0 db2plug.zip *.class
```

此命令将所有类文件放在 db2plug.zip 文件中，并保留相对路径信息。

相关概念:

- 第 359 页的『编写作为“控制中心”扩展的插件』
- 第 357 页的『“控制中心”插件开发者准则』

相关参考:

- 『db2cc - Start Control Center Command』（*Command Reference*）

编写作为“控制中心”扩展的插件

第一步编写用于定义实现 CCExtension 接口的类的插件。此类将包含要由“控制中心”装入的插件类的列表。如果想要将菜单项添加至标准“控制中心”对象（如“数据库”和“表”），或想要创建自己的对象以显示在树中，则创建实现 CCObject 接口并在 getObjects 方法中返回 CCObjects 数组的类。如果想要添加工具栏按钮，实现 CCToolbarAction 并在 getToolbarActions 方法中返回 CCToolbarActions 数组。

每个接口都在以下位置作了说明:

- 在 Windows® 平台上，在 DRIVE:\sqllib\samples\java\plugin\doc 中，其中 DRIVE: 表示安装 DB2® 的驱动器。
- 在 UNIX® 平台上，在 /u/db2inst1/sqllib/samples/java/plugin/doc 中，其中 /u/db2inst1 表示安装 DB2 的目录。

相关任务:

- 第 360 页的『创建用来添加工具栏按钮的插件』
- 第 361 页的『创建基本菜单操作』
- 第 362 页的『确定菜单项的位置』
- 第 363 页的『创建基本菜单操作分隔符』
- 第 363 页的『创建子菜单』
- 第 364 页的『仅将菜单项添加至具有特定名称的对象』
- 第 365 页的『添加文件夹以容纳树中的多个对象』
- 第 367 页的『将示例对象添加至文件夹』
- 第 368 页的『设置插件树对象的属性』
- 第 370 页的『添加创建操作』
- 第 371 页的『添加具有多个选择支持的除去操作』
- 第 372 页的『添加改变操作』
- 第 373 页的『禁用具有 isConfigurable() 的配置功能部件』
- 第 373 页的『使用 isEditable() 禁用改变对象的能力』
- 第 374 页的『使用 hasConfigurationDefaults() 禁用配置对话框中的缺省按钮』

插件任务描述

讨论下列插件任务:

1. 创建用来添加工具栏按钮的插件
2. 创建用来将新菜单项添加到数据库对象的插件
3. 创建用来将插件对象添加到树中数据库下的插件
4. 通过 isConfigurable() 禁用配置功能
5. 使用 isEditable() 禁用改变对象的能力
6. 使用 hasConfigurationDefaults() 禁用配置对话框中的缺省按钮

创建用来添加工具栏按钮的插件

过程:

对于此示例，将仅添加一个工具栏按钮，因此 getObjects 应返回一个 null 数组，如下所示:

```
import com.ibm.db2.tools.cc.navigator.*;
import  java.awt.event.*;
import javax.swing.*;

public class Example1 implements CCExtension {

  public CCObject[] getObjects () {
      return null;
  }

}
```

注意，已导入 com.ibm.db2.tools.cc.navigator 程序包。此类将实现 CCToolbarAction 接口，该接口要求实现三个方法: getHoverHelpText、getIcon 和 actionPerformed。“控制中心”使用 getHoverHelpText 来显示一个小文本框，该文本框在用户将鼠标悬浮在工具栏按钮上时出现。使用 getIcon 来指定按钮的图标。“控制中心”在用户单击按钮时调用 actionPerformed。以下是添加名为 X 的按钮的示例，单击该按钮时，会将一条消息写至控制台。它使用“控制中心”的映像库类中的“刷新”图标。

```
class Example1ToolbarAction implements CCToolbarAction {

  public String getHoverHelpText() { return "X"; }

  public ImageIcon getIcon() {
   return CommonImageRepository.getCommonIcon(CommonImageRepository.WC_NV_
REFRESH);
  }

  public void actionPerformed(ActionEvent e) {
   System.out.println("I've been clicked");
  }

}
```

最后一步是实现 Example1 中的 getToolbarActions 方法以返回新类的实例，如下所示:

```
  public CCToolbarAction[] getToolbarActions () {
   return new CCToolbarAction[] { new Example1ToolbarAction() };
  }
```

相关概念:

• 第 358 页的『编译和运行示例插件』

创建用来将新菜单项添加到数据库对象的插件

下列过程概述如何创建用来将新菜单项添加到数据库对象的插件:

1. 创建基本菜单操作
2. 定位菜单项
3. 创建基本菜单操作分隔符
4. 创建子菜单
5. 仅将菜单项添加到具有特定名称的对象

创建基本菜单操作

过程:

在这一较为高级的主题中，会将新命令添加至“数据库”对象的弹出菜单。

与示例 1 中一样，第一步是编写扩展 CCExtension 的类。

```
import com.ibm.db2.tools.cc.navigator.*;
import  java.awt.event.*;
import javax.swing.*;

public class Example2 implements CCExtension {

  public CCToolbarAction[] getToolbarActions () {
      return null;
  }

}
```

第二步是在树中为“数据库”对象创建 CCObject，如下所示:

```
  class CCDatabase implements CCObject {

    public String getName () { return null; }
    public boolean isEditable () { return true; }
    public boolean isConfigurable () { return true; }

    public int getType () { return UDB_DATABASE; }
  }
```

因为未使用除向“控制中心”内置对象（例如，在此示例中为“数据库”对象）添加菜单项之外的任何功能，在返回 null 或 true 时，将实现大部分功能。要指定想要此对象表示“DB2 UDB 数据库”对象，将其类型指定为 CCObject 中的常量 UDB_DATABASE。在此示例中，将类命名为 CCDatabase，但应尽量使类名唯一，原因插件所在的 zip 文件中可能存在其他供应商的插件。应使用 Java 包来帮助确保唯一类名。

CCExtension 的 getObjects 方法应返回包含实例 CCDatabase 的数组，如下所示:

```
  public CCObject[] getObjects () {
    return new CCObject[] { new CCDatabase() };
  }
```

可以创建类型为 UDB_DATABASE 的多个 CCObject 子类，但如果从其 isEditable 或 isConfigurable 方法返回的值有冲突，则返回 false 的对象将覆盖返回 true 的对象。

余下要实现的唯一方法是 getMenuActions。这将返回 CCMenuActions 数组，因此，首先将编写实现此接口的类。

要实现以下两个方法：getMenuText 和 actionPerformed。菜单中显示的文本是使用 getMenuText 获取的。用户单击菜单项时，被触发的事件导致调用 actionPerformed。

选择单个数据库对象时，以下示例类显示称为 “Example2a Action” 的菜单项。当用户单击此菜单项时，会将消息 “Example2a menu item actionPerformed” 写至控制台。

```
class Example2AAction implements CCMenuAction {

  public String getMenuText () { return "Example2a Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2a menu item actionPerformed");
  }

}
```

最后，通过将以下内容添加至 CCObject 以将此菜单项连接至 UDB 数据库 CCObject。

```
  public CCMenuAction[] getMenuActions () {
     return new CCMenuAction[] { new Example2AAction() };
  }
```

相关概念:

- 第 358 页的『编译和运行示例插件』

相关任务:

- 第 362 页的『确定菜单项的位置』
- 第 363 页的『创建基本菜单操作分隔符』
- 第 363 页的『创建子菜单』
- 第 364 页的『仅将菜单项添加至具有特定名称的对象』

确定菜单项的位置

过程:

创建基本菜单项时，未指定菜单项在菜单中的位置。将插件菜单项添加至菜单时，缺省行为是在末尾添加它们，但应添加在所有 “刷新” 和 “过滤” 菜单项的前面。

可重设此行为以指定从零直到菜单中的项数的任何位置号，不计 “刷新” 和 “过滤” 菜单项。更改 CCMenuAction 子类以实现 Positionable 然后实现 getPosition 方法，如下所示:

```
class Example2BAction implements CCMenuAction, Positionable {

  public String getMenuText () { return "Example2B Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2B menu item actionPerformed");
  }

  public int getPosition() {
     return 0;
  }
}
```

指定位置号零会将菜单项置于列表中的第一位，指定等于菜单中的项数（不计插件菜单项）的位置号会将菜单项置于底部，但在“刷新”和“过滤”菜单项的前面。还可以返回 Positionable.POSITION_BOTTOM 值以获取缺省行为，即，让菜单项置于底部，但在所有“刷新”和“过滤”菜单项的前面。如果有多个类型为 UDB_DATABASE 的 CCObject 的菜单项位于 POSITION_BOTTOM，则会根据从 CCExtension 中的 getObjects 方法返回类型为 UDB_DATABASE 的 CCObjects 的顺序对菜单项进行排序。

更改 CCDatabase 以将 Example2BAction 添加至菜单，如下所示:

```
public CCMenuAction[] getMenuActions () {
  return new CCMenuAction[] { new Example2AAction(),
                              new Example2BAction() };
}
```

相关任务:

- 第 361 页的『创建基本菜单操作』
- 第 363 页的『创建基本菜单操作分隔符』
- 第 363 页的『创建子菜单』
- 第 364 页的『仅将菜单项添加至具有特定名称的对象』

创建基本菜单操作分隔符

过程:

要添加分隔符，创建实现 Separator 接口的 CCMenuAction。将忽略所有其它方法（如果是实现 Positionable，则 getPosition 除外）。

```
class Example2CSeparator implements CCMenuAction, Separator, Positionable {

  public String getMenuText () { return null; }

  public void actionPerformed (ActionEvent e) {}

  public int getPosition() {
     return 1;
  }
}

public CCMenuAction[] getMenuActions () {
  return new CCMenuAction[] { new Example2AAction(),
                              new Example2BAction(),
                              new Example2CSeparator() };
}
```

相关任务:

- 第 361 页的『创建基本菜单操作』
- 第 362 页的『确定菜单项的位置』
- 第 363 页的『创建子菜单』
- 第 364 页的『仅将菜单项添加至具有特定名称的对象』

创建子菜单

过程:

子菜单是一个 CCMenuActions 数组。要让菜单项包含子菜单，必须实现 SubMenuParent 接口。然后，为每个子菜单项创建 CCMenuAction 的实现并从 SubMenuParent 接口的 getSubMenuActions 方法以数组的形式返回它们。不支持将菜单项添加至非插件子菜单。还应注意，SubMenuParents 不会接收来自“控制中心”的 ActionEvents。以下是一个示例:

```
class Example2DAction implements CCMenuAction, SubMenuParent {

  public String getMenuText () { return "Example2D Action"; }

  public void actionPerformed (ActionEvent e) {}

  public CCMenuAction[] getSubMenuActions() {
     return new CCMenuAction[] { new Example2DSubMenuAction() };
  }

}

class Example2DSubMenuAction implements CCMenuAction {

  public String getMenuText () { return "Example2D Sub-Menu Action"; }

  public void actionPerformed (ActionEvent e) {
    System.out.println("Example2D sub-menu menu item actionPerformed");
  }

}
```

再次将此新菜单项添加至 CCDatabase。

```
  public CCMenuAction[] getMenuActions () }
    return new CCMenuAction[] { new Example2AAction(),
                                new Example2BAction(),
                                new Example2CSeparator(),
                                new Example2DAction() };
  }
```

相关任务:

- 第 361 页的『创建基本菜单操作』
- 第 362 页的『确定菜单项的位置』
- 第 363 页的『创建基本菜单操作分隔符』
- 第 364 页的『仅将菜单项添加至具有特定名称的对象』

仅将菜单项添加至具有特定名称的对象

过程:

当前，在“控制中心”显示的任何数据库都将显示已编写的插件菜单项。可通过在 CCDatabase 的 getName 方法中返回该名称，将这些菜单项限制为具有特定名称的数据库。这必须是全限定名称。在这种情况下，因为在引用数据库，所以必须包括在 getName 方法中返回的系统、实例和数据库的名称。这些名称用“–”分隔。以下是一个示例，其中系统名为 MYSYSTEM，实例名为 DB2，而数据库名为 SAMPLE。

```
class CCDatabase implements CCObject {
  ...
  public String getName () { return "MYSYSTEM – DB2 – SAMPLE"; }
  ...
}
```

相关任务:

- 第 361 页的『创建基本菜单操作』
- 第 362 页的『确定菜单项的位置』
- 第 363 页的『创建基本菜单操作分隔符』
- 第 363 页的『创建子菜单』

创建用来将插件对象添加到树中数据库下的插件

下列过程概述如何创建用来将插件对象添加到树中数据库下的插件:

1. 添加文件夹以容纳树中的多个对象
2. 将示例对象添加到文件夹下面
3. 设置插件树对象的属性
4. 添加创建操作
5. 添加除去操作
6. 添加改变操作

添加文件夹以容纳树中的多个对象

过程:

在本示例中，将实现 CCTreeObject 而不是 CCObject，以便让插件对象在“控制中心”树中显示在“数据库”下面。首先，需要为此对象创建 CCTreeObject 实现。如果要将多个对象放在树中而不是将它们直接放在“数据库”下面，通常会创建文件夹。以下是文件夹的初始版本:

```
  public class Example3Folder implements CCTreeObject {
    private String parentName = null;
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public CCTableObject getChildren () { return null; }
    public void setParentName(String name)
    {
      parentName = name;
    }
    public CCColumn[] getColumns () { return null; }
    public boolean isLeaf () { return false; }
    public CCMenuAction[] getMenuActions () { return null; }

    public String getName () { return "Example3 Folder"; }

    public void getData (Object[] data) {
       data[0] = this;
    }

    public int getType () { return CCTypeFactory.getTypeNumber
(this.getClass().getName()); }

    public Icon getIcon (int iconState) {
      switch (iconState) {
        case CLOSED_FOLDER:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_CLOSED_
FOLDER);

        case OPEN_FOLDER:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_OPEN_
FOLDER);

        default:
  return CommonImageRepository.getScaledIcon(CommonImageRepository.NV_CLOSED_
```

```
FOLDER);
      }
    }

  }
```

注意，现在 getType 使用类 CCTypeFactory。CCTypeFactory 用来防止两个对象使用同一个类型号，以便 “控制中心” 可以将插件标识为具有唯一类型。新文件夹不是一种内置 CC 对象类型，而是一个新类型且需要具有新类型号，该类型号一定不能与您创建的任何其它新类型号冲突，也不能与内置类型的类型号冲突。

getIcon 方法使用 iconState 作为参数，让您知道您是打开还是关闭文件夹。然后，可以使图标与状态相对应，如上所述。

当选择数据库且不只是在树中时，为在此详细视图中显示文件夹，getData 需要返回单列，列的值是插件对象本身。getData 方法将*此*引用指定给数据数组的第一个元素。这允许图标和名称出现在详细视图的同一列中。当看到您返回 CCTableObject 子类时，“控制中心” 就知道它可以对 Example3Folder 调用 getIcon 和 getName。

下一步是创建 CCDatabase 类以实现 CCTreeObject 并从其 getChildren 方法返回 CCTableObject 数组，该数组包含实例 Example3Folder，如下所示:

```
import java.util.*;

  class CCDatabase implements CCTreeObject {
    private String parentName = null;
    private Vector childVector;

    public CCDatabase() {
      childVector = new Vector();
      childVector.addElement(new Example3Folder());
    }

    public CCTableObject[] getChildren() {
      CCTableObject[] children = new CCTableObject[childVector.size()];
      childVector.copyInto(children);
      return children;
    }

    public void setParentName(String name)
    {
      parentName = name;
    }

    public String getName () { return null; }
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public void getData (Object[] data) { }
    public CCColumn[] getColumns () { return null; }
    public boolean isLeaf () { return false; }
    public int getType () { return UDB_DATABASE; }
    public Icon getIcon (int iconState) { return null; }
    public CCMenuAction[] getMenuActions () { return null; }
  }
```

相关概念:

- 第 358 页的『编译和运行示例插件』

相关任务:

- 第 367 页的『将示例对象添加至文件夹』

- 第 368 页的『设置插件树对象的属性』
- 第 370 页的『添加创建操作』
- 第 371 页的『添加具有多个选择支持的除去操作』
- 第 372 页的『添加改变操作』

将示例对象添加至文件夹

过程:

第一步是为子对象创建 CCObject 实现，如下所示:

```
  class Example3Child implements CCTableObject {
    private String parentName = null;
    public String getName () { return null; }
    public boolean isEditable () { return false; }
    public boolean isConfigurable () { return false; }
    public void getData (Object[] data) { }
    public CCColumn[] getColumns () { return null; }
    public Icon getIcon (int iconState) { return null; }
    public CCMenuAction[] getMenuActions () { return null; }
    public void setParentName(String name)
    {
      parentName = name;
    }
    public int getType () { return CCTypeFactory.getTypeNumber
(this.getClass().getName()); }
  }
```

下一步是修改 Example3Folder 以保存这些 Exercise3Child 对象的向量，如下所示:

```
  public class Example3Folder implements CCObject {
    private String parentName = null;
    private Vector childVector;
    ...

    public Example3Folder() {
      childVector = new Vector();
    }

    ...
    public CCTableObject[] getChildren () {
      CCTableObject[] children = new CCTableObject[childVector.size()];
      childVector.copyInto(children);
      return children;
    }
    public void setParentName(String name)
    {
      parentName = name;
    }
    ...
  }
```

为简单起见，在本示例中，getChildren 返回名为 childVector 的向量中存储的子代的数组。

真正的插件应该在调用 getChildren 时重新构造子代。这将刷新列表，此列表可能包含自从上次显示列表之后在“控制中心”外部创建或更改的新子对象或已更改的子对象。应该将子代存储在持久存储器中并从中读取，以防它们丢失。

并且，在真正的插件中，getChildren 返回的子代列表取决于在“控制中心”树中作为此对象父代的对象。父代信息位于由“控制中心”进行的 setParentName 方法调用提供的 parentName 字符串中。

注: 在此示例中，当在“控制中心”中从树中的“数据库”对象或更高级别执行刷新时，将丢失 Example3Folder 下的子代列表。这是因为新的 Example3Folder 是执行刷新时由“控制中心”构造的。如果此示例代码从持久存储器读取子代，则不会丢失子代。为了保持示例简单，我们没有执行此操作。

相关概念:

- 第 358 页的『编译和运行示例插件』

相关任务:

- 第 365 页的『添加文件夹以容纳树中的多个对象』
- 第 368 页的『设置插件树对象的属性』
- 第 370 页的『添加创建操作』
- 第 371 页的『添加具有多个选择支持的除去操作』
- 第 372 页的『添加改变操作』

设置插件树对象的属性

过程:

如果将树展开至插件文件夹并选择它，则将看到详细窗格中没有任何列。这是因为 getColumns 的 Example3Child 实现正在返回 null。要更改此项，首先创建一些 CCColumn 实现。我们将创建两列，因为接下来的示例将演示如何在运行时更改其中一个列的值，而每个对象所具有的一列决不能更改。将调用未更改列“Name”和更改列“State”。

```
class NameColumn implements CCColumn {
   getName() { return "Name"; }
   getColumnClass { return CCTableObject.class; }
}

class StateColumn implements CCColumn {
   getName() { return "State"; }
   getColumnClass { return String.class; }
}
```

受支持的类类型包括 Java 原语（例如 java.lang.Intger）的类等价项、java.util.Date 类和 CCTableObject 类。

将 Example3Child 的 getColumns 方法更改为包括这两列。

```
class Example3Child implements CCTableObject {
  ...
  public CCColumn[] getColumns () {
     return new CCColumn[] { new NameColumn(),
                             new StateColumn() };
  }
  ...
}
```

必须将父代更改为包括相同的列。

```
class Example3Folder implements CCTableObject {
  ...
  public CCColumn[] getColumns () {
     return new CCColumn[] { new NameColumn(),
                             new StateColumn() };
  }
  ...
}
```

现在，必须设置将在详细视图中对每个行显示的值。通过设置传送至 getData 的 Object 数组的元素来达到此目的。每个列数据的类必须与 getColumnClass 对相应的列返回的类相匹配。

```
class Example3Child implements CCTableObject {
  ...
  private String name;
  private String state;

  public Exampe3Child(String name, String state) {
    this.name = name;
    this.state = state;
  }
  ...
  public void getData (Object[] data) {
    data[0] = this;
    data[1] = state;
  }
  ...
}
```

在本例中，类 CCTableObject 的第一列将具有此值。这允许“控制中心”同时显示 getName 返回的文本和 getIcon 返回的图标。因此，下一步是实现这两个方法。将仅对工具栏按钮使用示例 1 中使用的相同刷新图标。

```
class Example3Child implements CCTableObject {
  ...
  public String getName () {
    return name;
  }
  public Icon getIcon () {
return CommonImageRepository.getScaledIcon(CommonImageRepository.WC_NV_
REFRESH);
  }
  ...
}
```

要查看至今为止的工作结果，可创建一个示例子对象，您将在下一个练习中除去它。构造 childVector 时，将实例 Example3Child 添加至 Example3Folder。

```
public class Example3Folder implements CCTreeObject {
  ...
  public Example3Folder() {
    childVector = new Vector();
    childVector.addElement(new Example3Child("Plugin1", "State1"));
  }
  ...
}
```

相关概念:

- 第 358 页的『编译和运行示例插件』

相关任务:

- 第 365 页的『添加文件夹以容纳树中的多个对象』

- 第 367 页的『将示例对象添加至文件夹』
- 第 370 页的『添加创建操作』
- 第 372 页的『添加改变操作』

添加创建操作

过程:

要允许用户在运行时在文件夹下面创建对象，只需更新向量、使类为 Observable 并且在用户触发事件时调用 notifyObservers。“控制中心”自动将其本身注册为作为 Observable 的任何 CCTableObjects 的 Observer。

首先，将方法添加至 Example3Folder 以将子对象添加至其子代的向量。

```
public class Example3Folder implements CCTreeObject, Observable {
  ...
  public void addChild(Example3Child child) {
    childVector.addElement(child);
    setChanged();
    notifyObservers(new CCObjectCollectionEvent(this,
                    CCObjectCollectionEvent.OBJECT_ADDED,
                    child));
  }
  ...
}
```

在以上的代码中，使用称为 CCObjectCollectionEvent 的新类作为 notifyObservers 的自变量。CCObjectCollectionEvent 是一个事件，它表示 CCObject 集合（例如“控制中心”树中的文件夹）中的更改。“控制中心”获取扩展 Observable 的所有 CCObject 并通过更新树和详细视图响应 CCObjectCollectionEvent。有三种类型的事件：添加、除去和改变。

CCObjectCollectionEvent 有三个自变量。第一个自变量是触发事件的对象。第二个自变量是事件的类型，可以是 OBJECT_ADDED、OBJECT_ALTERED 或 OBJECT_REMOVED。最后一个自变量是正在创建的新对象。

下一步，将菜单项添加至文件夹以允许用户触发对新 addChild 方法的调用。

```
class CreateAction implements CCMenuAction {
  private int pluginNumber = 0;
  public String getMenuText () { return "Create"; }

  public void actionPerformed (ActionEvent e) {
Example3Folder folder = (Example3Folder)((Vector)e.getSource()).elementAt(0);
    folder.addChild(new Example3Child("Plugin " + ++pluginNumber, "State1"));
  }
}
```

ActionEvent 将总是包含对其调用操作的所有对象的向量。由于将仅对 Example3Folder 调用此操作且只能有一个文件夹，所以仅强制转型向量中的第一个对象并对其调用 addChild。

最后一步是将菜单操作添加至文件夹，现在可以除去先前添加的样本对象。

```
public class Example3Folder extends Observable implements CCTreeObject {
  private CCMenuAction[] menuActions =
     new CCMenuAction[] { new CreateChildAction(); }
  ...
  public Example3Folder() {
```

```
    childVector = new Vector();
  }
  ...
  public CCMenuAction[] getMenuActions () {
    retun menuActions;
  }
  ...
}
```

相关概念:

- 第 358 页的『编译和运行示例插件』

相关任务:

- 第 365 页的『添加文件夹以容纳树中的多个对象』
- 第 367 页的『将示例对象添加至文件夹』
- 第 368 页的『设置插件树对象的属性』
- 第 371 页的『添加具有多个选择支持的除去操作』
- 第 372 页的『添加改变操作』

添加具有多个选择支持的除去操作

过程:

现在，用户可以随心所欲地为插件创建实例，您可能想要同时给予他们删除这些实例的能力。首先，将方法添加至 Example3Folder 以除去子代并通知 “控制中心”。

```
public class Example3Folder extends Observable implements CCTreeObject {

  public void removeChild(Example3Child child) {
    childVector.removeElement(child);
    setChanged();
    notifyObservers(new CCObjectCollectionEvent(this,
                    CCObjectCollectionEvent.OBJECT_REMOVED,
                    child));
  }
}
```

下一步是将菜单操作添加至 Example3Child。将使 CCMenuAction 实现 MultiSelectable 以便用户可以同时除去多个对象。因为此操作的源将是 Example3Child 对象而不是 Example3Folder 的向量，所以应使用其它方法（如在构造函数中）将 Example3Folder 传送至菜单操作。

```
class RemoveAction implements CCMenuAction, MultiSelectable {
  private Example3Folder folder;

  public RemoveAction(Example3Folder folder) {
     this.folder = folder;
  }

  public String getMenuText () { return "Remove"; }

  public int getSelectionMode () { return MultiSelectable.MULTI_HANDLE_ONE; }

  public void actionPerformed (ActionEvent e) {
    Vector childrenVector = (Vector)e.getSource();
    for (int i = 0; i < childrenVector.size(); i++) {
      folder.removeChild((Example3Child)childrenVector.elementAt(i));
    }
  }
}
```

实现 MultiSelectable 要求您实现 getSelectionMode。在本例中，会让它返回 MULTI_HANDLE_ONE，这表示此菜单项将出现在菜单上，即使选择了多个对象且对所有选择的对象调用单个 actionPerformed 方法也是如此。

现在，将新菜单操作添加至 Example3Child。这涉及将新参数添加至 Example3Child 构造函数以在文件夹中传送。

```
class Example3Child implements CCTableObject {
  ...
  private CCMenuAction[] menuActions;

  public Example3Child(Example3Folder folder, String name, String state) {
    ...
    menuActions = new CCMenuAction[] { new RemoveAction(folder) };
  }
  ...
  public CCMenuAction[] getMenuActions () {
    return menuActions;
  }
}
```

记住更改 CreateAction 以使用新构造函数。

```
class CreateAction implements CCMenuAction {
  ...
  public void actionPerformed (ActionEvent e) {
    ...
folder.addChild(new Example3Child(folder, "Plugin " + ++pluginNumber,
"State 1"));
  }
}
```

相关概念:

- 第 358 页的『编译和运行示例插件』

相关任务:

- 第 365 页的『添加文件夹以容纳树中的多个对象』
- 第 367 页的『将示例对象添加至文件夹』
- 第 368 页的『设置插件树对象的属性』
- 第 370 页的『添加创建操作』
- 第 372 页的『添加改变操作』

添加改变操作

过程:

对于插件，“控制中心”侦听的最后一种类型的事件是 OBJECT_ALTERED 事件。在前一示例中，我们创建了“State”列，所以可在本示例中演示此功能部件。调用“改变”操作时将增加状态值。

第一步是编写用来更改状态的方法，但此时它将在 Example3Child 上而不是在文件夹中。在本例中，第一个自变量和第三个自变量都是 Example3Child。记住扩展 Observable。

```
class Example3Child extends Observable implements CCTableObject {
  ...
  public void setState(String state) {
    this.state = state;
```

```
    setChanged();
    notifyObservers(new CCObjectCollectionEvent(this,
                        CCObjectCollectionEvent.OBJECT_ALTERED, this));
  }
  ...
}
```

下一步，为“改变”创建菜单操作并将其添加至 Example3Child 中的 CCMenuAction 数组。AlterAction 类还将实现 CCDefaultMenuAction 接口，将警报定义为缺省操作，这样当用户在“控制中心”双击 Example3Child 对象时，即可调用该操作。

```
class AlterAction implements CCMenuAction, CCDefaultMenuAction {
  private int stateNumber = 1;
  public String getMenuText () { return "Alter"; }

  public void actionPerformed (ActionEvent e) {
    ((Example3Child)((Vector)e.getSource()).elementAt(0)).setState("State "
        + ++stateNumber);
  }
}

class Example3Child implements CCTableObject {
  ...
  public Example3Child(Example3Folder folder, String name, String state) {
    ...
    menuActions = new CCMenuAction[] { new AlterAction(),
                                       new RemoveAction(folder) };
  }
  ...
}
```

相关概念:

- 第 358 页的『编译和运行示例插件』

相关任务:

- 第 365 页的『添加文件夹以容纳树中的多个对象』
- 第 367 页的『将示例对象添加至文件夹』
- 第 368 页的『设置插件树对象的属性』
- 第 370 页的『添加创建操作』
- 第 371 页的『添加具有多个选择支持的除去操作』

禁用具有 isConfigurable() 的配置功能部件

过程:

在类型为 UDB_DATABASE 或 UDB_INSTANCE 的 CCObject 的 isConfigurable 方法中返回 false 值时，将分别从“数据库”和“实例”弹出菜单中除去“配置”菜单项。

相关概念:

- 第 358 页的『编译和运行示例插件』

使用 isEditable() 禁用改变对象的能力

过程:

对于支持“改变”操作的任何“控制中心”对象，都可通过为该对象创建插件并从 isEditable 方法返回 false 来除去该操作。还可以指定仅对其全限定名称与该对象的 getName 方法返回的值相匹配的对象除去“改变”操作。

一个附加功能部件是添加“浏览”操作。仅可对“UDB 表”、“视图”和索引执行此操作。“浏览”操作是通过将称为 BrowseTable、BrowseView 或 BrowseIndex 的文件放在 db2plug.zip 中添加的。这些文件可以是空的，但它们的名称必须是 BrowseTable、BrowseView 或 BrowseIndex。添加“浏览”按钮不能限制为具有特定名称的对象。

相关概念:

- 第 358 页的『编译和运行示例插件』

使用 hasConfigurationDefaults() 禁用配置对话框中的缺省按钮

过程:

UDB 数据库和实例的配置对话框包含用于将值设置回 DB2 缺省值的按钮。如果有自己的一组缺省值且不想用户误击这些按钮，则可使用插件禁用这些按钮。创建实现 CCObject 的对象并将其类型设置为 UDB_DATABASE 或 UDB_INSTANCE。同时让它实现 Defaultable 接口。此接口包含称为 hasConfigurationDefaults 的方法。从此方法返回 false 将会禁用配置对话框的所有缺省值按钮。本示例禁用“UDB 数据库配置”的缺省值按钮。

相关概念:

- 第 358 页的『编译和运行示例插件』

附录 K. DB2 通用数据库技术信息

DB2 文档和帮助

DB2® 技术信息可通过下列工具和方法获得:

- DB2 信息中心
  - 主题
  - DB2 工具的帮助
  - 样本程序
  - 教程
- 可下载的 PDF 文件、CD 上的 PDF 文件和印刷书籍
  - 指南
  - 参考手册
- 命令行帮助
  - 命令帮助
  - 消息帮助
  - SQL 状态帮助
- 已安装的源代码
  - 样本程序

可以在线访问 ibm.com® 上的其它 DB2 Universal Database™（DB2 通用数据库）技术信息，例如，技术说明、白皮书和 Redbooks™（红皮书）。访问位于以下网址的 DB2 信息管理软件资料库站点: www.ibm.com/software/data/pubs/。

DB2 文档更新

IBM® 可能会定期提供 DB2 信息中心的文档修订包和其它文档更新。如果访问 http://publib.boulder.ibm.com/infocenter/db2help/ 网址中的 DB2 信息中心，则将始终可以查看最新的信息。如果本地安装了 DB2 信息中心，则需要手工安装所有更新才能查看它们。文档更新允许您在新信息可供使用时更新从 *DB2 信息中心 CD* 安装的信息。

信息中心的更新比 PDF 或硬拷贝书籍的更新要频繁。要获得最新的 DB2 技术信息，一提供文档更新时就安装它们，或者访问 www.ibm.com 站点上的 DB2 信息中心。

相关概念:

- 『CLI sample programs』（*CLI Guide and Reference, Volume 1*）
- 『Java 样本程序』（*《应用程序开发指南: 构建和运行应用程序》*）
- 第 376 页的『DB2 信息中心』

相关任务:

- 第 391 页的『从 DB2 工具调用上下文帮助』
- 第 384 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』
- 第 392 页的『从命令行处理器调用消息帮助』

- 第 392 页的『从命令行处理器调用命令帮助』
- 第 393 页的『从命令行处理器调用 SQL 状态帮助』

相关参考:

- 第 385 页的『DB2 PDF 和印刷文档』

DB2 信息中心

DB2® 信息中心使您可以访问充分利用 DB2 系列产品（包括 DB2 Universal Database™（DB2 通用数据库）、DB2 Connect™、DB2 Information Integrator 和 DB2 Query Patroller™）所需的所有信息。 DB2 信息中心还包含主要的 DB2 功能部件和组件（包括复制、数据仓储和 DB2 extender）的信息。

如果是在 Mozilla 1.0（或更新版本）或 Microsoft® Internet Explorer 5.5（或更新版本）中查看的话，则 DB2 信息中心具有下列功能部件。某些功能部件需要您启用对 JavaScript™ 的支持:

灵活安装选项

可选择使用最适合您的需要的选项来查看 DB2 文档:

- 要轻松确保文档始终是最新的，可直接从 IBM® Web 站点上的 DB2 信息中心访问所有文档，网址为: http://publib.boulder.ibm.com/infocenter/db2help/
- 要将更新工作量减至最少并使网络通信保持在内部网内，可将 DB2 文档安装在内部网上的单台服务器上
- 要使您有最大的灵活性并减少对网络连接的依赖，可将 DB2 文档安装在您自己的计算机上

搜索 可通过在**搜索**文本字段中输入搜索术语来搜索 DB2 信息中心中的所有主题。可通过用引号将术语括起来以检索确定匹配项，还可以使用通配运算符（* 和 ?）和布尔运算符（AND、NOT 和 OR）细化搜索。

面向任务的目录

可从单个目录查找 DB2 文档中的主题。目录主要是按想要执行的任务的种类组织的，同时也包括有关产品概述、目标、参考信息、索引和词汇表的条目。

- 产品概述描述 DB2 系列中的可用产品之间的关系、其中每个产品提供的功能部件以及其中每个产品的最新发行信息。
- 目标类别（例如，安装、管理和开发）包括一些主题，这些主题使您能够快速地完成任务并且更好地理解完成这些任务的背景信息。
- 参考主题提供有关主题的详细信息，包括语句和命令语法、消息帮助以及配置参数。

显示目录中的当前主题

可通过单击目录框架中的**刷新 / 显示当前主题**按钮或通过单击内容框架中的**在目录中显示**按钮来显示当前主题在目录中的位置。如果访问了指向若干个文件中的相关主题的若干个链接，或者是从搜索结果到达主题的，此功能会非常有用。

索引 可从索引访问全部文档。索引是按索引项的拼音顺序组织的。

词汇表 可使用词汇表来查找在 DB2 文档中使用的术语的定义。词汇表是按词汇表术语的拼音顺序组织的。

集成的本地化信息

DB2 信息中心以您在浏览器首选项中设置的首选语言显示信息。如果主题不是以首选语言提供的，则 DB2 信息中心将显示该主题的英文版。

有关 iSeries™ 技术信息，参阅 IBM eServer™ iSeries 信息中心，网址为 www.ibm.com/eserver/iseries/infocenter/。

相关概念:

- 第 377 页的『DB2 信息中心安装方案』

相关任务:

- 第 384 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』
- 第 384 页的『以首选语言显示 DB2 信息中心中的主题』
- 第 383 页的『调用 DB2 信息中心』
- 第 379 页的『使用 “DB2 安装” 向导来安装 DB2 信息中心（UNIX）』
- 第 381 页的『使用 “DB2 安装” 向导来安装 DB2 信息中心（Windows）』

DB2 信息中心安装方案

对于如何访问 DB2® 信息，不同的工作环境可有不同的需求。可以使用三种方法访问 DB2 信息中心: 从 IBM® Web 站点访问、从组织网络的服务器访问或从安装在计算机上的版本访问。在所有三种情况中，文档都包含在 DB2 信息中心中，DB2 信息中心是基于主题的信息的结构化 Web，可使用浏览器来查看。缺省情况下，DB2 产品从 IBM Web 站点访问 DB2 信息中心。但是，如果想要从内部网服务器或从您自己的计算机访问 DB2 信息中心，必须使用产品 “介质包” 中的 DB2 信息中心 CD 来安装 DB2 信息中心。参阅下面的用于访问 DB2 文档的选项的总结及三个方案来帮助确定访问 DB2 信息中心时使用哪个方法最适合您和您的工作环境，以及可能需要考虑哪些安装问题。

用于访问 DB2 文档的选项的总结:

下表提供了有关哪些选项可能用来在您的工作环境中访问 DB2 信息中心中的 DB2 产品文档的建议。

因特网访问	内部网访问	建议
是	是	访问 IBM Web 站点上的 DB2 信息中心，或者访问安装在内部网服务器上的 DB2 信息中心。
是	否	访问 IBM Web 站点上的 DB2 信息中心。
否	是	访问安装在内部网服务器上的 DB2 信息中心。
否	否	访问本地计算机上的 DB2 信息中心。

方案: 访问您的计算机上的 DB2 信息中心:

Tsu-Chen 在一个小镇上开了一家工厂，而小镇没有本地 ISP，所以他不能访问因特网。他购买了 DB2 Universal Database™（DB2 通用数据库）来管理库存、产品订单、银行帐户信息和业务开销。由于以前从未使用过 DB2 产品，所以 Tsu-Chen 需要从 DB2 产品文档了解产品的使用方法。

使用典型安装选项在计算机上安装了 DB2 通用数据库之后，Tsu-Chen 尝试访问 DB2 文档。但是，浏览器显示一条错误消息，提示找不到他尝试打开的页。Tsu-Chen 查阅了 DB2 产品的安装手册，发现如果想要访问计算机上的 DB2 文档，则必须安装 DB2 信息中心。他在介质包中找到了 *DB2 信息中心 CD* 并安装了它。

Tsu-Chen 现在能够从操作系统的应用程序启动程序访问 DB2 信息中心，也能够了解如何使用 DB2 产品来增加业务的成功率。

方案: 访问 IBM Web 站点上的 DB2 信息中心:

Colin 是培训公司的一名信息技术顾问。他精通数据库技术和 SQL 并对全北美使用 DB2 通用数据库的企业提供有关这些主题的讲座。Colin 的部分讲座包括将 DB2 文档用作教学工具。例如，在讲授有关 SQL 的课程时，Colin 使用有关 SQL 的 DB2 文档作为教授数据库查询的基本和高级语法的方法。

Colin 授课的大部分企业都访问因特网。当 Colin 安装了 DB2 通用数据库的最新版本时，他会决定配置其移动式计算机以访问 IBM Web 站点上的 DB2 信息中心。此配置允许 Colin 在授课期间在线访问最新的 DB2 文档。

但是，有时在旅行时 Colin 不能访问因特网。这对他来说是个问题，尤其是在他需要访问 DB2 文档来备课时。为避免类似情况，Colin 在他的移动式计算机上安装了 DB2 信息中心的副本。

Colin 可以随时很方便的获得 DB2 文档的副本。通过使用 **db2set** 命令，他可以根据所在位置很容易地将其移动式计算机上的注册表变量配置为访问 IBM Web 站点上或他的移动式计算机上的 DB2 信息中心。

方案: 访问内部网服务器上的 DB2 信息中心:

Eva 是一家人寿保险公司的高级数据库管理员。她的管理职责包括在公司的 UNIX® 数据库服务器上安装和配置 DB2 通用数据库的最新版本。她的公司最近通知职员，为安全起见，在工作时间将不提供因特网访问。因为公司具有联网环境，所以 Eva 决定在内部网服务器上安装 DB2 信息中心，以便公司中经常使用公司数据仓库的所有职员（销售代表、销售经理和业务分析人员）都可以访问 DB2 文档。

Eva 会指导她的数据库小组使用响应文件在所有职员的计算机上安装 DB2 通用数据库的最新版本，以确保每台计算机都配置为使用内部网服务器的主机名和端口号来访问 DB2 信息中心。

但是，Eva 的小组中的初级数据库管理员 Migual 误解了 Eva 的意思，她在一些职员的计算机上安装了 DB2 信息中心的副本，但没有将 DB2 通用数据库配置为访问内部网服务器上的 DB2 信息中心。为了更正这种情况，Eva 告诉 Migual 使用 **db2set** 命令在这些计算机中的每一台上更改 DB2 信息中心注册表变量（DB2_DOCHOST 表示主机名，而 DB2_DOCPORT 表示端口号）。现在，该网络上的所有相应计算机都可以访问 DB2 信息中心，而且职员可在 DB2 文档中查找有关 DB2 问题的答案。

相关概念:

- 第 376 页的『DB2 信息中心』

相关任务:

- 第 384 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』

- 第 379 页的『使用 “DB2 安装” 向导来安装 DB2 信息中心（UNIX）』
- 第 381 页的『使用 “DB2 安装” 向导来安装 DB2 信息中心（Windows）』
- 『设置访问 DB2 信息中心的位置: 公共 GUI 帮助』

相关参考:

- 『db2set - DB2 Profile Registry Command』（*Command Reference*）

使用 “DB2 安装” 向导来安装 DB2 信息中心（UNIX）

可使用三种方法访问 DB2 产品文档: 从 IBM Web 站点访问、从内部网服务器访问或从计算机上安装的版本访问。缺省情况下，DB2 产品从 IBM Web 站点访问 DB2 文档。如果想要从内部网服务器或您自己的计算机访问 DB2 文档，必须从 *DB2 信息中心 CD* 安装文档。使用 “DB2 安装” 向导，您可以定义安装首选项并在使用 UNIX 操作系统的计算机上安装 DB2 信息中心。

先决条件:

本节列示了在 UNIX 计算机上安装 DB2 信息中心的硬件、操作系统、软件以及通信需求。

- **硬件需求**

  需要下列其中一种处理器:

  – PowerPC（AIX）
  – HP 9000（HP-UX）
  – Intel 32 位（Linux）
  – Solaris UltraSPARC 计算机（Solaris Operating Environment）

- **操作系统需求**

  需要下列其中一个操作系统:

  – IBM AIX 5.1（在 PowerPC 上）
  – HP-UX 11i（在 HP 9000 上）
  – Red Hat Linux 8.0（在 Intel 32 位上）
  – SuSE Linux 8.1（在 Intel 32 位上）
  – Sun Solaris V8（在 Solaris Operating Environment UltraSPARC 计算机上）

  注: DB2 信息中心在支持 DB2 客户机的一部分 UNIX 操作系统上运行。因此，建议从 IBM Web 站点访问 DB2 信息中心，或者在内部网服务器上安装并访问 DB2 信息中心。

- **软件需求**

  – 支持下列浏览器:

    - Mozilla V1.0 或更高版本

- “DB2 安装” 向导是一个图形安装程序。必须实现能够呈示图形用户界面的 X Window System 软件才能使 “DB2 安装” 向导在计算机上运行。必须确保正确导出了显示内容，才能运行 “DB2 安装” 向导。例如，在命令提示符处输入以下命令:

  ```
  export DISPLAY=9.26.163.144:0.
  ```

- **通信需求**
  - TCP/IP

过程:

要使用“DB2 安装”向导安装 DB2 信息中心:

1. 登录系统。
2. 在系统上放入并装上 DB2 信息中心产品 CD。
3. 通过输入以下命令切换到装上 CD 的目录:

   ```
   cd /cd
   ```

   其中 */cd* 表示 CD 的安装点。
4. 输入 **./db2setup** 命令来启动“DB2 安装”向导。
5. “IBM DB2 安装启动板”打开。要直接进至 DB2 信息中心的安装，单击**安装产品**。联机帮助可指导您完成其余步骤。要调用联机帮助，单击**帮助**。可随时单击**取消**来结束安装。
6. 在**选择您想要安装的产品**页中，单击**下一步**。
7. 在**欢迎使用“DB2 安装”向导**页中，单击**下一步**。“DB2 安装”向导将指导您完成程序安装过程。
8. 要继续安装，必须接受许可协议。在**许可协议**页中，选择**我接受许可协议中的条款**，然后单击**下一步**。
9. 在**选择安装操作**页中，选择**在此计算机上安装 DB2 信息中心**。如果想要在稍后使用响应文件在此计算机或其它计算机上安装 DB2 信息中心，则选择**将设置保存在响应文件中**。单击**下一步**。
10. 在**选择要安装的语言**页中，选择将用来安装 DB2 信息中心的语言。单击**下一步**。
11. 在**指定 DB2 信息中心端口**页中，配置 DB2 信息中心的入局通信。单击**下一步**继续安装。
12. 在**开始复制文件**页中复查您作出的安装选择。要更改任何设置，单击**上一步**。单击**安装**以将 DB2 信息中心文件复制到计算机上。

还可以使用响应文件安装 DB2 信息中心。

缺省情况下，安装日志 `db2setup.his`、`db2setup.log` 和 `db2setup.err` 位于 `/tmp` 目录中。

`db2setup.log` 文件会捕获所有 DB2 产品安装信息（包括错误）。`db2setup.his` 文件会记录计算机上的所有 DB2 产品安装。DB2 将 `db2setup.log` 文件追加至 `db2setup.his` 文件。`db2setup.err` 文件捕获 Java 返回的任何错误输出，例如，异常和陷阱信息。

当安装完成后，DB2 信息中心将安装在下列其中一个目录中，这取决于您的 UNIX 操作系统:

- AIX: /usr/opt/db2_08_01
- HP-UX: /opt/IBM/db2/V8.1
- Linux: /opt/IBM/db2/V8.1
- Solaris Operating Environment: /opt/IBM/db2/V8.1

相关概念:

- 第 376 页的『DB2 信息中心』
- 第 377 页的『DB2 信息中心安装方案』

相关任务:

- 『使用响应文件安装 DB2（UNIX）』（《*安装与配置补充手册*》）
- 第 384 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』
- 第 384 页的『以首选语言显示 DB2 信息中心中的主题』
- 第 383 页的『调用 DB2 信息中心』
- 第 381 页的『使用“DB2 安装”向导来安装 DB2 信息中心（Windows）』

使用“DB2 安装”向导来安装 DB2 信息中心（Windows）

可使用三种方法访问 DB2 产品文档: 从 IBM Web 站点访问、从内部网服务器访问或从计算机上安装的版本访问。缺省情况下，DB2 产品从 IBM Web 站点访问 DB2 文档。如果想要从内部网服务器或您自己的计算机访问 DB2 文档，必须从 *DB2 信息中心 CD* 安装 DB2 文档。使用“DB2 安装”向导，可以定义安装首选项并在使用 Windows 操作系统的计算机上安装 DB2 信息中心。

先决条件:

本节列示了在 Windows 上安装 DB2 信息中心的硬件、操作系统、软件以及通信需求。

- **硬件需求**

  需要下列其中一种处理器:

  – 32 位计算机: 奔腾或与奔腾兼容的 CPU

- **操作系统需求**

  需要下列其中一个操作系统:

  – Windows 2000

  – Windows XP

  注: DB2 信息中心在支持 DB2 客户机的一部分 Windows 操作系统上运行。因此，建议从 IBM Web 站点访问 DB2 信息中心，或者在内部网服务器上安装并访问 DB2 信息中心。

- **软件需求**

  – 支持下列浏览器:

    - Mozilla 1.0 或更高版本

    - Internet Explorer V5.5 或 V6.0（对于 Windows XP，则为 Internet Explorer V6.0）

- **通信需求**

  – TCP/IP

限制:

- 需要具有安装 DB2 信息中心的管理特权的帐户。

过程:

要使用“DB2 安装”向导安装 DB2 信息中心:

1. 使用为 DB2 信息中心安装定义的帐户登录至系统。

2. 将 CD 插入到驱动器中。如果启用了自动运行功能，则它将启动“IBM DB2 安装启动板”。
3. “DB2 安装”向导会确定系统语言并启动该语言的安装程序。如果想要运行英语之外的语言的安装程序，或者安装程序无法自动启动，则可以手工启动“DB2 安装”向导。

   要手工启动“DB2 安装”向导:

   a. 单击**开始**并选择**运行**。
   b. 在**打开**字段中，输入以下命令:

      ```
      x:\setup.exe /i 2-letter language identifier
      ```

      其中 *x:* 表示 CD 驱动器，而 *2-letter language identifier* 表示将用来运行安装程序的语言。
   c. 单击**确定**。
4. “IBM DB2 安装启动板”打开。要直接进至 DB2 信息中心的安装，单击**安装产品**。联机帮助可指导您完成其余步骤。要调用联机帮助，单击**帮助**。可随时单击**取消**来结束安装。
5. 在**选择您想要安装的产品**页中，单击**下一步**。
6. 在**欢迎使用“DB2 安装”向导**页中，单击**下一步**。“DB2 安装”向导将指导您完成程序安装过程。
7. 要继续安装，必须接受许可协议。在**许可协议**页中，选择**我接受许可协议中的条款**，然后单击**下一步**。
8. 在**选择安装操作**页中，选择**在此计算机上安装 DB2 信息中心**。如果想要在稍后使用响应文件在此计算机或其它计算机上安装 DB2 信息中心，则选择**将设置保存在响应文件中**。单击**下一步**。
9. 在**选择要安装的语言**页中，选择将用来安装 DB2 信息中心的语言。单击**下一步**。
10. 在**指定 DB2 信息中心端口**页中，配置 DB2 信息中心的入局通信。单击**下一步**继续安装。
11. 在**开始复制文件**页中复查您作出的安装选择。要更改任何设置，单击**上一步**。单击**安装**以将 DB2 信息中心文件复制到计算机上。

可以使用响应文件来安装 DB2 信息中心。还可以使用 **db2rspgn** 命令来根据现有安装生成响应文件。

有关安装期间遇到的错误的信息，请参阅 'My Documents'\DB2LOG\ 目录中的 db2.log 和 db2wi.log 文件。'My Documents' 目录的位置将取决于计算机的设置。

db2wi.log 文件会捕获最新的 DB2 安装信息。db2.log 会捕获 DB2 产品安装的历史。

相关概念:

- 第 376 页的『DB2 信息中心』
- 第 377 页的『DB2 信息中心安装方案』

相关任务:

- 『使用响应文件安装 DB2 产品（Windows）』（*《安装与配置补充手册》*）
- 第 384 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』
- 第 384 页的『以首选语言显示 DB2 信息中心中的主题』

- 第 383 页的『调用 DB2 信息中心』
- 第 379 页的『使用 “DB2 安装” 向导来安装 DB2 信息中心（UNIX）』

相关参考:

- 『db2rspgn - Response File Generator Command (Windows)』（*Command Reference*）

调用 DB2 信息中心

DB2 信息中心允许您访问使用用于 Linux、UNIX 和 Windows 操作系统的 DB2 系列产品（例如，DB2 通用数据库、DB2 Connect、DB2 Information Integrator 和 DB2 Query Patroller）所需的所有信息。

可以从下列其中一个位置调用 DB2 信息中心;

- 安装了 DB2 UDB 客户机或服务器的计算机
- 安装了 DB2 信息中心的内部网服务器或本地计算机
- IBM Web 站点

先决条件:

在调用 DB2 信息中心之前:

- *可选*: 配置浏览器以便用首选语言来显示主题
- *可选*: 配置 DB2 客户机以使用安装在计算机或内部网服务器上的 DB2 信息中心

过程:

要调用安装了 DB2 UDB 客户机或服务器的计算机上的 DB2 信息中心:

- 从 “开始菜单”（Windows 操作系统）: 单击**开始 ⟶ 程序 ⟶ IBM DB2 ⟶ 信息 ⟶ 信息中心**。
- 从命令行提示符:
  - 对于 Linux 和 UNIX 操作系统，发出 **db2icdocs** 命令。
  - 对于 Windows 操作系统，发出 **db2icdocs.exe** 命令。

要在 Web 浏览器中打开安装在内部网服务器或本地计算机上的 DB2 信息中心:

- 打开 Web 页面（网址为: http://<host-name>:<port-number>/），其中 <host-name> 表示主机名，而 <port-number> 表示提供 DB2 信息中心的端口号。

要在 Web 浏览器中打开 IBM Web 站点上的 DB2 信息中心:

- 打开 Web 页面（网址为: publib.boulder.ibm.com/infocenter/db2help/）。

相关概念:

- 第 376 页的『DB2 信息中心』
- 第 377 页的『DB2 信息中心安装方案』

相关任务:

- 第 391 页的『从 DB2 工具调用上下文帮助』
- 第 384 页的『更新安装在计算机或内部网服务器上的 DB2 信息中心』
- 第 392 页的『从命令行处理器调用命令帮助』
- 『设置访问 DB2 信息中心的位置: 公共 GUI 帮助』

相关参考:

- 『HELP Command』（*Command Reference*）

更新安装在计算机或内部网服务器上的 DB2 信息中心

http://publib.boulder.ibm.com/infocenter/db2help/ 提供的 DB2 信息中心将会用新的或更改过的文档定期更新。IBM 还可能提供 DB2 信息中心更新，可以下载这些更新并将它们安装在计算机或内部网服务器上。更新 DB2 信息中心不会更新 DB2 客户机或服务器产品。

先决条件:

必须能够访问连接至因特网的计算机。

过程:

要更新安装在计算机或内部网服务器上的 DB2 信息中心:

1. 打开位于 IBM Web 站点上的 DB2 信息中心: http://publib.boulder.ibm.com/infocenter/db2help/
2. 在欢迎页面的“服务与支持”标题下面的“下载”部分，单击 **DB2 通用数据库文档**链接。
3. 通过将最新刷新的文档映像级别与已安装的文档级别进行比较来确定 DB2 信息中心的版本是否已过时。已安装的文档级别列示在 DB2 信息中心欢迎页面上。
4. 如果有较新版本的 DB2 信息中心可用，则下载适用于您的操作系统的最新刷新的 *DB2 信息中心*映像。
5. 要安装刷新过的 *DB2 信息中心*映像，遵循 Web 页面上提供的指示信息。

相关概念:

- 第 377 页的『DB2 信息中心安装方案』

相关任务:

- 第 383 页的『调用 DB2 信息中心』
- 第 379 页的『使用“DB2 安装”向导来安装 DB2 信息中心（UNIX）』
- 第 381 页的『使用“DB2 安装”向导来安装 DB2 信息中心（Windows）』

以首选语言显示 DB2 信息中心中的主题

DB2 信息中心尝试以您在浏览器首选项中指定的语言显示主题。如果主题未翻译为首选语言，则 DB2 信息中心将显示该主题的英文版。

过程:

要在 Internet Explorer 浏览器中以您的首选语言显示主题:

1. 在 Internet Explorer 中，单击**工具** —> **Internet 选项** —> **语言...** 按钮。“语言首选项”窗口打开。
2. 确保您的首选语言被指定为语言列表中的第一个条目。
   - 要将新语言添加至列表，单击**添加...** 按钮。

     注: 添加语言并不能保证计算机具有以首选语言显示主题所需的字体。

- 要将语言移至列表顶部，选择该语言并单击**上移**按钮直到该语言成为语言列表中的第一项。

3. 刷新该页面以便以首选语言显示 DB2 信息中心。

要在 Mozilla 浏览器中以首选语言显示主题:

1. 在 Mozilla 中，选择**编辑** —> **首选项** —> **语言**按钮。“语言”面板将显示在“首选项”窗口中。
2. 确保您的首选语言被指定为语言列表中的第一个条目。
   - 要将新语言添加至列表，单击**添加...** 按钮以从“添加语言”窗口中选择一种语言。
   - 要将语言移至列表顶部，选择该语言并单击**上移**按钮直到该语言成为语言列表中的第一项。
3. 刷新该页面以便以首选语言显示 DB2 信息中心。

相关概念:

- 第 376 页的『DB2 信息中心』

DB2 PDF 和印刷文档

下列各表提供正式书名、书号和 PDF 文件名。要订购硬拷贝书籍，必须知道正式书名。要打印 PDF 文件，必须知道 PDF 文件名。

DB2 文档按下列标题分类:

- 核心 DB2 信息
- 管理信息
- 应用程序开发信息
- 商业智能信息
- 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 资料库中的每本书籍，下表描述了订购该书籍的硬拷贝、打印或查看该书籍的 PDF 所需的信息。DB2 资料库中的每本书籍的完整描述可从 IBM 出版物中心（IBM Publications Center）获取，网址为 www.ibm.com/shop/publications/order。

核心 DB2 信息

这些书籍中的信息对所有 DB2 用户来说都是基础知识，不管您是程序员、数据库管理员或是使用 DB2 Connect、DB2 仓库管理器或其它 DB2 产品的人员，都将会发现此信息很有用。

表 53. 核心 DB2 信息

书名	书号	PDF 文件名
IBM DB2 Universal Database Command Reference	SC09-4828	db2n0x81
《IBM DB2 通用数据库词汇表》	无书号	db2t0c81

表 53. 核心 DB2 信息 (续)

书名	书号	PDF 文件名
《IBM DB2 通用数据库消息参考 第 1 卷》	G152-0177，未提供硬拷贝	db2m1c81
《IBM DB2 通用数据库消息参考 第 2 卷》	G152-0178，未提供硬拷贝	db2m2c81
《IBM DB2 通用数据库新增内容》	S152-0176	db2q0c81

管理信息

这些书籍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。

表 54. 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	db2d1c81
《IBM DB2 通用数据库管理指南: 实现》	S152-0165	db2d2c81
《IBM DB2 通用数据库管理指南: 性能》	S152-0166	db2d3c81
IBM DB2 Universal Database Administrative API Reference	SC09-4824	db2b0x81
IBM DB2 Universal Database Data Movement Utilities Guide and Reference	SC09-4830	db2dmx81
《IBM DB2 通用数据库数据恢复和高可用性指南与参考》	S152-0181	db2hac81
《IBM DB2 通用数据库数据仓库中心管理指南》	S152-0188	db2ddc81
IBM DB2 Universal Database SQL Reference, Volume 1	SC09-4844	db2s1x81
IBM DB2 Universal Database SQL Reference, Volume 2	SC09-4845	db2s2x81
IBM DB2 Universal Database System Monitor Guide and Reference	SC09-4847	db2f0x81

应用程序开发信息

这些书籍中的信息对于应用程序开发者或使用 DB2 通用数据库（DB2 UDB）的程序员特别有用。您将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（例如，嵌入式 SQL、ODBC、JDBC、SQLJ 和 CLI）访问 DB2 UDB 所需的文档。如果正在使用 DB2 信息中心，还可访问 HTML 版本的源代码以获取样本程序。

表 55. 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	db2axc81
IBM DB2 Universal Database Application Development Guide: Programming Client Applications	SC09-4826	db2a1x81
IBM DB2 Universal Database Application Development Guide: Programming Server Applications	SC09-4827	db2a2x81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 1	SC09-4849	db2l1x81
IBM DB2 Universal Database Call Level Interface Guide and Reference, Volume 2	SC09-4850	db2l2x81
IBM DB2 Universal Database Data Warehouse Center Application Integration Guide	SC27-1124	db2adx81
IBM DB2 XML Extender Administration and Programming	SC27-1234	db2sxx81

商业智能信息

这些书籍中的信息描述如何使用将增强 DB2 通用数据库的数据仓储功能和分析功能的组件。

表 56. 商业智能信息

书名	书号	PDF 文件名
IBM DB2 Warehouse Manager Standard Edition Information Catalog Center Administration Guide	SC27-1125	db2dix81
IBM DB2 Warehouse Manager Standard Edition Installation Guide	G152-0187	db2idc81
IBM DB2 Warehouse Manager Standard Edition Managing ETI Solution Conversion Programs with DB2 Warehouse Manager	SC18-7727	iwhe1mstx80

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 个人版来存取大型机和中型机服务器上的数据。

表 57. DB2 Connect 信息

书名	书号	PDF 文件名
IBM Connectivity Supplement	无书号	db2h1x81
《IBM DB2 Connect 快速入门 DB2 Connect 企业版》	G152-0271	db2c6c81
《IBM DB2 Connect 快速入门 DB2 Connect 个人版》	G152-0171	db2c1c81
《IBM DB2 Connect 用户指南》	S152-0172	db2c0c81

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。

表 58. 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，未提供硬拷贝	db2itc81
《IBM DB2 通用数据库快速入门 DB2 服务器版》	G152-0173	db2isc81
《IBM DB2 通用数据库快速入门 DB2 个人版》	G152-0175	db2i1c81
《IBM DB2 通用数据库安装与配置补充手册》	G152-0174，未提供硬拷贝	db2iyc81
《IBM DB2 通用数据库快速入门 DB2 Data Links Manager 版》	G152-0169	db2z6c81

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。

表 59. 教程信息

书名	书号	PDF 文件名
《商业智能教程：数据仓库简介》	无书号	db2tuc81
《商业智能教程：数据仓储扩展课程》	无书号	db2tac81
Information Catalog Center Tutorial	无书号	db2aix81
Video Central for e-business Tutorial	无书号	db2twx81
《Visual Explain 教程》	无书号	db2tvc81

可选组件信息

此类别中的信息描述如何使用可选 DB2 组件。

表 60. 可选组件信息

书名	书号	PDF 文件名
《IBM DB2 Cube Views 指南与参考》	S152–0532	db2aac81
IBM DB2 Query Patroller Guide: Installation, Administration and Usage Guide	GC09–7658	db2dwx81
IBM DB2 Spatial Extender and Geodetic Extender User's Guide and Reference	SC27-1226	db2sbx81
IBM DB2 Database Data Links Manager Administration Guide and Reference	SC27-1221	db2z0x82
《DB2 Net Search Extender 管理和用户指南》 **注**: 此文档的 HTML *不是*从 "HTML 文档" CD 安装的。	S152-0596	不适用

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。发行说明还提供了并入到每个发行版、更新和修订包中的文档更新的总结。

表 61. 发行说明

书名	书号	PDF 文件名
《DB2 发行说明》	请参阅 "注"。	请参阅 "注"。
《DB2 安装说明》	仅在产品 CD-ROM 上提供。	未提供。

注: 提供有下列格式的发行说明:

- XHTML 和文本格式（在产品 CD 上）
- PDF 格式（在 PDF 文档 CD 上）

此外，《发行说明》中讨论*已知问题和变通方法*和*发行版之间的不兼容性*的部分还会出现在 DB2 信息中心中。

要在基于 UNIX 的平台上查看文本格式的发行说明，请参阅 `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:

- 对于 AIX 操作系统: `/usr/opt/db2_08_01`
- 对于所有其它基于 UNIX 的操作系统: `/opt/IBM/db2/V8.1`

相关概念:

- 第 375 页的『DB2 文档和帮助』

相关任务:

- 第 390 页的『从 PDF 文件打印 DB2 书籍』
- 第 390 页的『订购印刷的 DB2 书籍』
- 第 391 页的『从 DB2 工具调用上下文帮助』

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:

确保安装了 Adobe Acrobat Reader。如果需要安装 Adobe Acrobat Reader，则可从 Adobe Web 站点获得它，网址为 www.adobe.com。

过程:

要从 PDF 文件打印 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，参阅《*快速入门*》一书。
2. 打开 index.htm。文件将在浏览器窗口中打开。
3. 单击想要查看的 PDF 的标题。该 PDF 将在 Acrobat Reader 中打开。
4. 选择**文件 → 打印**以打印想要的书籍的任何部分。

相关概念:

- 第 376 页的『DB2 信息中心』

相关任务:

- 『装上 CD-ROM（AIX）』（《*DB2 服务器快速入门*》）
- 『装上 CD-ROM（HP-UX）』（《*DB2 服务器快速入门*》）
- 『装上 CD-ROM（Linux）』（《*DB2 服务器快速入门*》）
- 第 390 页的『订购印刷的 DB2 书籍』
- 『安装 CD-ROM（Solaris Operating Environment）』（《*DB2 服务器快速入门*》）

相关参考:

- 第 385 页的『DB2 PDF 和印刷文档』

订购印刷的 DB2 书籍

如果喜欢使用硬拷贝书籍，可以用以下三种方式中的一种订购它们。

过程:

可在某些国家或地区订购印刷版书籍。访问您所在国家或地区的 IBM 出版物 Web 站点，以了解您所在国家或地区是否提供此项服务。如果可以订购这些出版物，则您可以:

- 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。

- 访问 IBM 出版物中心（IBM Publications Center），网址为 http://www.ibm.com/shop/publications/order。可能未在所有国家或地区提供从 IBM 出版物中心订购书籍这项功能。

DB2 产品可用时，印刷书籍与 *DB2 PDF 文档 CD* 上以 PDF 格式提供的那些书籍是相同的。印刷书籍中的内容出现在 *DB2 信息中心 CD* 中时也是相同的。但是，DB2 信息中心 CD 中有一些附加内容未出现在 PDF 书籍中的任何位置（例如，SQL 管理例程和 HTML 样本）。并非 DB2 PDF 文档 CD 上提供的所有书籍都可以订购硬拷贝。

注: DB2 信息中心的更新比 PDF 或硬拷贝书籍的更新要频繁得多；一提供文档更新就安装它们，或者参阅网址如下的 DB2 信息中心以获取最新信息：http://publib.boulder.ibm.com/infocenter/db2help/。

相关任务:

- 第 390 页的『从 PDF 文件打印 DB2 书籍』

相关参考:

- 第 385 页的『DB2 PDF 和印刷文档』

从 DB2 工具调用上下文帮助

上下文帮助提供有关与特定窗口、笔记本、向导或顾问程序相关联的任务或控件的信息。上下文帮助可从具有图形用户界面的 DB2 管理和开发工具获得。有两种类型的上下文帮助:

- 通过位于每个窗口或笔记本上的**帮助**按钮访问的帮助
- 弹出信息，即将鼠标光标放到字段或控件上或在窗口、笔记本、向导或顾问程序中选择了字段或控件并按 F1 键时显示的弹出信息窗口。

帮助按钮允许您访问概述、先决条件和任务信息。弹出信息描述各个字段和控件。

过程:

要调用上下文帮助:

- 要获取窗口和笔记本帮助，启动其中一个 DB2 工具，然后打开任意窗口或笔记本。单击窗口或笔记本右下角的**帮助**按钮以调用上下文帮助。

  还可从位于每个 DB2 工具中心上方的**帮助**菜单项访问上下文帮助。

  在向导和顾问程序中，单击第一页上的“任务概述”链接以查看上下文帮助。
- 要获取有关窗口或笔记本的各个控件的弹出信息帮助，单击该控件，然后按 **F1**。包含有关控件的详细信息的弹出信息将显示在黄色窗口中。

  注: 如果希望只要将鼠标光标放在字段或控件上就显示弹出信息，在“工具设置”笔记本的**文档**页上选择**自动显示弹出信息**复选框。

  与弹出信息类似，诊断弹出信息是另一种形式的上下文相关帮助；它们包含数据输入规则。诊断弹出信息显示在输入的数据无效或不充分时出现的紫色窗口中。会对以下各项显示诊断弹出信息:
  - 必填字段。
  - 其数据遵照精确格式的字段，例如，日期字段。

相关任务:

- 第 383 页的『调用 DB2 信息中心』
- 第 392 页的『从命令行处理器调用消息帮助』
- 第 392 页的『从命令行处理器调用命令帮助』
- 第 393 页的『从命令行处理器调用 SQL 状态帮助』
- 『访问 DB2 信息中心: 概念帮助』
- 『如何使用 DB2 UDB 帮助: 公共 GUI 帮助』
- 『设置访问 DB2 信息中心的位置: 公共 GUI 帮助』
- 『设置对 DB2 上下文帮助和文档的访问权: 公共 GUI 帮助』

从命令行处理器调用消息帮助

消息帮助描述产生消息的原因并描述为解决错误而应采取的任何操作。

过程:

要调用消息帮助，打开命令行处理器并输入:

```
? XXXnnnnn
```

其中 *XXXnnnnn* 表示有效的消息标识。

例如，`? SQL30081` 会显示有关 SQL30081 消息的帮助。

相关概念:

- 『消息介绍』（《*消息参考*》*第 1 卷*）

相关参考:

- 『db2 - Command Line Processor Invocation Command』（*Command Reference*）

从命令行处理器调用命令帮助

命令帮助说明命令行处理器中命令的语法。

过程:

要调用命令帮助，打开命令行处理器并输入:

```
? 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` 只显示 CATALOG DATABASE 命令的帮助。

相关任务:

- 第 391 页的『从 DB2 工具调用上下文帮助』
- 第 383 页的『调用 DB2 信息中心』
- 第 392 页的『从命令行处理器调用消息帮助』

- 第 393 页的『从命令行处理器调用 SQL 状态帮助』

相关参考:

- 『db2 - Command Line Processor Invocation Command』（*Command Reference*）

从命令行处理器调用 SQL 状态帮助

DB2 通用数据库返回可作为 SQL 语句结果的条件的 SQLSTATE 值。SQLSTATE 帮助说明 SQL 状态和 SQL 状态类代码的含义。

过程:

要调用 SQL 状态帮助，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 383 页的『调用 DB2 信息中心』
- 第 392 页的『从命令行处理器调用消息帮助』
- 第 392 页的『从命令行处理器调用命令帮助』

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:

可从“信息中心”查看 XHTML 版本的教程，网址如下: http://publib.boulder.ibm.com/infocenter/db2help/。

某些教程课程使用了样本数据或代码。有关其特定任务的任何先决条件的描述，请参阅每个教程。

DB2 通用数据库教程:

单击以下列表中的教程标题以查看该教程。

《商业智能教程: 数据仓库中心介绍》
使用“数据仓库中心”来执行介绍性的数据仓储任务。

《商业智能教程: 数据仓储扩展课程》
使用“数据仓库中心”来执行高级数据仓储任务。

《信息目录中心教程》
使用“信息目录中心”来创建和管理信息目录以查找并使用元数据。

《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。

DB2 故障诊断信息

提供有大量故障诊断和问题确定信息，可帮助您使用 DB2® 产品。

DB2 文档

DB2 信息中心以及构成 DB2 资料库的 PDF 书籍中处处可找到故障诊断信息。可参阅 DB2 信息中心导航树（在浏览器窗口的左窗格中）的“支持和故障诊断”分支以查看 DB2 故障诊断文档的完整列表。

DB2 技术支持 Web 站点

如果您遇到了问题并且想要获取查找可能的原因和解决方案的帮助，请参阅 DB2 技术支持（DB2 Technical Support）Web 站点。该“技术支持”站点具有指向最新 DB2 出版物、技术说明、授权程序分析报告（APAR）、修订包的链接、内部 DB2 错误代码的最新列表以及其它资源。可搜索此知识库并查找问题的可能解决方案。

访问位于以下网址的 DB2 技术支持 Web 站点：
http://www.ibm.com/software/data/db2/udb/winos2unix/support

DB2 问题确定教程系列

要查找有关如何快速标识和解决在使用 DB2 产品时可能遇到的问题的信息，参阅 DB2 问题确定教程系列 Web 站点。有一个教程介绍可用的 DB2 问题确定设施和工具并帮助您决定何时使用它们。其它教程处理相关主题，例如“数据库引擎问题确定”、“性能问题确定”和“应用程序问题确定”。

查看 DB2 技术支持站点上的 DB2 问题确定教程的完整集合，网址如下：
http://www.ibm.com/software/data/support/pdm/db2tutorials.html

相关概念:

- 第 376 页的『DB2 信息中心』
- 『问题确定 – DB2 技术支持教程简介』（*Troubleshooting Guide*）

辅助功能

辅助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）的用户成功地使用软件产品。以下列表指定 DB2® V8 产品中的主要辅助功能部件:

- 所有 DB2 功能可使用键盘（而不是鼠标）导航来实现。有关更多信息，请参阅第 395 页的『键盘输入和导航』。
- 可定制 DB2 用户界面上的字体大小和颜色。有关更多信息，请参阅第 395 页的『界面显示的辅助功能』。
- DB2 产品支持使用 Java™ Accessibility API 的辅助功能应用程序。有关更多信息，请参阅第 395 页的『与辅助技术的兼容性』。
- DB2 文档是以易使用格式提供的。有关更多信息，请参阅第 395 页的『文档的辅助功能』。

键盘输入和导航

键盘输入

只使用键盘就可以操作 DB2 工具。使用键或键组合就可以执行使用鼠标所能完成的操作。标准操作系统击键用于标准操作系统操作。

有关使用键或键组合执行操作的更多信息，请参阅 键盘快捷方式和加速键: 公共 GUI 帮助。

键盘导航

可使用键或键组合来导航 DB2 工具用户界面。

有关使用键或键组合来导航 DB2 工具的更多信息，请参阅 键盘快捷方式和加速键: 公共 GUI 帮助。

键盘焦点

在 UNIX® 操作系统中，击键操作起作用的活动窗口的区域将突出显示。

界面显示的辅助功能

DB2 工具所具有的功能部件使视力不太好的用户更易使用。这些辅助功能方面的增强包括了对可定制字体属性的支持。

字体设置

可使用“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。

有关指定字体设置的更多信息，请参阅 更改菜单和文本的字体: 公共 GUI 帮助。

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。

与辅助技术的兼容性

DB2 工具界面支持 Java Accessibility API，它使您能够将屏幕阅读器和其它辅助技术与 DB2 产品配合使用。

文档的辅助功能

DB2 的相关文档是以 XHTML 1.0 格式提供的，它在大部分 Web 浏览器中是可查看的。XHTML 允许您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技术。

语法图是以点分十进制格式提供的。仅当使用屏幕阅读器访问联机文档时，此格式才可用。

相关概念:

- 第 396 页的『点分十进制语法图』

相关任务:

- 『键盘快捷方式和加速键: 公共 GUI 帮助』
- 『更改菜单和文本的字体: 公共 GUI 帮助』

点分十进制语法图

语法图是以点分十进制的格式为使用屏幕阅读器访问信息中心的用户提供的。

在点分十进制格式中，每个语法元素写在单独的一行上。如果两个或多个语法元素总是一起出现（或总是一起不出现），它们可显示在同一行上，这是因为可将它们视作单个复合语法元素。

每一行以点分十进制编号开始；例如，3、3.1 或 3.1.1。要正确地听到这些数字，确保屏幕阅读器设置为读出标点。具有相同点分十进制编号的所有语法元素（例如，具有编号 3.1 的所有语法元素）是互斥的替代项。如果听到行 3.1 USERID 和 3.1 SYSTEMID，就知道语法可能包括 USERID 或 SYSTEMID，但不会同时包括这两者。

点分十进制编号级别表示嵌套级别。例如，如果具有点分十进制编号 3 的语法元素后跟点分十进制编号为 3.1 的一系列语法元素，则编号为 3.1 的所有语法元素是编号为 3 的语法元素的下级。

某些单词和符号用在点分十进制编号的旁边以添加有关这些语法元素的信息。这些单词和符号有时可能会出现在元素本身的开头。为易于识别，如果该单词或符号是语法元素的一部分，它的前面会加上反斜杠（\）字符。* 符号可用在点分十进制编号的旁边以指示该语法元素重复。例如，点分十进制编号为 3 的语法元素 *FILE 的格式为 3 * FILE。3* FILE 这一格式指示语法元素 FILE 重复。格式 3* * FILE 指示语法元素 * FILE 重复。

用来分隔一串语法元素的字符（例如，逗号）在语法中刚好显示在它们要分隔的项之前。这些字符可与每一项显示在同一行上，或显示在单独一行上并带有与相关项相同的点分十进制编号。该行还可显示另一个符号，该符号给出有关语法元素的信息。例如，行 5.1*、5.1 LASTRUN 和 5.1 DELETE 意味着如果使用多个 LASTRUN 和 DELETE 语法元素，必须用逗号分隔这些元素。如果未指定分隔符，则假定使用空格来分隔每个语法元素。

如果语法元素前面有 % 符号，这表示在别处定义的引用。% 符号之后的字符串是语法段的名称，而非文字。例如，行 2.1 %OP1 意味着您应引用单独的语法分段 OP1。

下列单词和符号用在点分十进制编号的旁边:

- ? 表示可选语法元素。后跟 ? 符号的点分十进制编号指示具有相应点分十进制编号的所有语法元素及任何下级语法元素都是可选的。如果只有一个带有点分十进制编号的语法元素，则 ? 符号与该语法元素显示在同一行上（例如，5? NOTIFY）。如果有多个带有点分十进制编号的语法元素，则 ? 符号单独显示在一行上，后跟可选语法元素。例如，如果您听到行 5 ?、5 NOTIFY 和 UPDATE，就知道语法元素 NOTIFY 和 UPDATE 是可选的；即，您可选择其中一项或全部都不选。? 符号相当于路线图中的支路。
- ! 表示缺省语法元素。后跟 ! 符号的点分十进制编号和语法元素指示该语法元素是共享同一点分十进制编号的所有语法元素的缺省选项。只有共享同一点分十进制编号的语法元素的其中一个可指定 ! 符号。例如，如果听到行 2? FILE、2.1! (KEEP) 和 2.1 (DELETE)，就知道 (KEEP) 是 FILE 关键字的缺省选项。在此示例中，如果包括 FILE 关键字但未指定选项，将应用缺省选项 KEEP。缺省选项还会应用于下一个较高的点分十进制编号。在此示例中，如果省略了 FILE 关键字，将使用缺省值 FILE(KEEP)。但是，如果听到行 2? FILE、2.1、2.1.1! (KEEP) 和 2.1.1 (DELETE)，

则缺省选项 KEEP 仅应用于下一个较高的点分十进制编号 2.1（它没有相关联的关键字），而不会应用于 2? FILE。如果省略了关键字 FILE，则不会使用任何值。

- * 表示可重复零次或多次的语法元素。后跟 * 符号的点分十进制编号指示此语法元素可使用零次或多次；即，它是可选的而且可以重复。例如，如果听到行 5.1* data area，就知道可以包括一个数据区、多个数据区或者不包括数据区。如果听到行 3*、3 HOST 和 3 STATE，就知道可包括 HOST 和／或 STATE 或者不包括任何内容。

  注:

  1. 如果点分十进制编号的旁边有星号（*）且只有一项带有该点分十进制编号，可重复同一项多次。
  2. 如果点分十进制编号的旁边有星号且有若干项带有该点分十进制编号，可使用列表中的多项，但每项只能使用一次。在先前示例中，可以写为 HOST STATE，但不能写为 HOST HOST。
  3. * 符号相当于路线语法图中的回路。

- + 表示必须被包括一次或多次的语法元素。后跟 + 符号的点分十进制编号指示此语法元素必须被包括一次或多次；即，它必须至少被包括一次，而且可以重复。例如，如果听到行 6.1+ data area，就知道必须至少包括一个数据区。如果听到行 2+、2 HOST 和 2 STATE，就知道必须包括 HOST 和／或 STATE。与 * 符号类似，如果 + 符号是带有该点分十进制编号的唯一项，则它只能重复特定项。与 * 符号一样，+ 符号相当于路线语法图中的回路。

相关概念:

- 第 394 页的『辅助功能』

相关任务:

- 『键盘快捷方式和加速键: 公共 GUI 帮助』

相关参考:

- 『How to read the syntax diagrams』（*SQL Reference, Volume 2*）

DB2 通用数据库产品的 Common Criteria 认证

DB2 通用数据库正在进行 Common Criteria 评估保证级别 4（EAL4）的评估认证。有关 Common Criteria 的更多信息，请参阅 Common Criteria Web 站点: http://niap.nist.gov/cc-scheme/

附录 L. 声明

IBM 可能在所有国家或地区不提供本文档中讨论的产品、服务或功能特性。有关您当前所在区域的产品和服务的信息，请向您当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并非意在明示或暗示只能使用 IBM 的产品、程序或服务。只要不侵犯 IBM 的知识产权，任何同等功能的产品、程序或服务，都可以代替 IBM 产品、程序或服务。但是，评估和验证任何非 IBM 产品、程序或服务，则由用户自行负责。

IBM 公司可能已拥有或正在申请与本文档内容有关的各项专利。提供本文档并未授予用户使用这些专利的任何许可。您可以用书面方式将许可查询寄往:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

有关双字节（DBCS）信息的许可查询，请与您所在国家或地区的 IBM 知识产权部门联系，或用书面方式将查询寄往:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

条款不适用英国或任何这样的条款与当地法律不一致的国家或地区: International Business Machines Corporation “按现状” 提供本出版物，不附有任何种类的（无论是明示的还是暗含的）保证，包括但不限于暗含的有关非侵权、适销和适用于某种特定用途的保证。某些国家或地区在某些交易中不允许免除明示或暗含的保证。因此本条款可能不适用于您。

本信息中可能包含技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些更改将编入本资料的新版本中。IBM 可以随时对本资料中描述的产品和／或程序进行改进和／或更改，而不另行通知。

本信息中对非 IBM Web 站点的任何引用都只是为了方便起见才提供的，不以任何方式充当对那些 Web 站点的保证。那些 Web 站点中的资料不是 IBM 产品资料的一部分，使用那些 Web 站点带来的风险将由您自行承担。

IBM 可以按它认为适当的任何方式使用或分发您所提供的任何信息而无须对您承担任何责任。

本程序的被许可方如果要了解有关程序的信息以达到如下目的: （i）允许在独立创建的程序和其他程序（包括本程序）之间进行信息交换，以及（ii）允许对已经交换的信息进行相互使用，请与下列地址联系:

IBM Canada Limited
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。

本资料中描述的许可程序及其所有可用的许可资料均由 IBM 依据 IBM 客户协议、IBM 国际软件许可协议或任何同等协议中的条款提供。

此处包含的任何性能数据都是在受控环境中测得的。因此，在其他操作环境中获得的数据可能会有明显的不同。有些测量可能是在开发级的系统上进行的，因此不保证与一般可用系统上进行的测量结果相同。此外，有些测量是通过推算而估计的，实际结果可能会有差异。本文档的用户应当验证其特定环境的适用数据。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版说明或其他可公开获得的资料中获取。IBM 没有对这些产品进行测试，也无法确认其性能的精确性、兼容性或任何其他关于非 IBM 产品的声明。有关非 IBM 产品性能的问题应当向这些产品的供应商提出。

所有关于 IBM 未来方向或意向的声明都可随时更改或收回，而不另行通知，它们仅仅表示了目标和意愿而已。

本信息可能包含在日常业务操作中使用的数据和报告的示例。为了尽可能完整地说明这些示例，示例中可能会包括个人、公司、品牌和产品的名称。所有这些名称都是虚构的，与实际商业企业所用的名称和地址的任何雷同纯属巧合。

版权许可:

本信息可能包括源语言形式的样本应用程序，这些样本说明不同操作平台上的编程方法。如果是为按照在编写样本程序的操作平台上的应用程序编程接口（API）进行应用程序的开发、使用、经销或分发为目的，您可以任何形式对这些样本程序进行复制、修改、分发，而无须向 IBM 付费。这些示例并未在所有条件下作全面测试。因此，IBM 不能担保或暗示这些程序的可靠性、可维护性或功能。

凡这些样本程序的每份拷贝或其任何部分或任何衍生产品，都必须包括如下版权声明:

©（*贵公司的名称*）（*年*）。此部分代码是根据 IBM 公司的样本程序衍生出来的。
© Copyright IBM Corp. （*输入年份*）. All rights reserved.

商标

下列各项是国际商业机器公司在美国和／或其他国家或地区的商标，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 Information Integrator
DB2 Query Patroller
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

下列各项是其他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／或其他国家或地区的商标。

Intel 和 Pentium 是 Intel Corporation 在美国和／或其他国家或地区的商标。

Java 和所有基于 Java 的商标是 Sun Microsystems, Inc. 在美国和／或其他国家或地区的商标。

UNIX 是 The Open Group 在美国和其他国家或地区的注册商标。

索引

[A]

[B]

[C]

[D]

[F]

[G]

[H]

[J]

[K]

[L]

[M]

[P]

[Q]

[R]

[S]

[T]

[W]

[X]

[Y]

[Z]

A

B

C

D

E

F

T

U

V

W

[特别字符]

与 IBM 联系

在中国，请致电下列其中一个号码以与 IBM 联系:

- 800-810-1818 或 (010) 84981188 分机 5151，可获得售前客户服务
- 800-810-1818 或 (010) 84981188 分机 5200，可获得售后客户服务
- 800-810-1818 或 (010) 84981188 分机 5017，可获得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，可在网上查看 IBM 全球联系人目录（Directory of Worldwide Contacts），网址为: http://www.ibm.com/planetwide

产品信息

有关 DB2 通用数据库产品的信息可通过万维网获取，网址为: http://www-900.ibm.com/cn/software/db2/

此站点包含有关 DB2 产品家族、DB2 解决方案、技术前沿与趋势、DB2 服务、成功案例、市场活动、培训与认证、DB2 开发者园地、合作伙伴、下载中心、资料库、第三方分析报告、殊荣与奖项、DB2 新闻以及如何购买 DB2 的最新信息。

有关如何在中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

中国印刷

S152-0165-01

Spine information:

IBM® DB2™ Universal Database
DB2 通用数据库

管理指南：实现

版本 8.2