DB2 Express-C

快 速 入 门

源于社区　贡献社区

RAUL CHONG, IAN HAKES, RAV AHUJA 著

DR. ARVIND KRISHNA 序

第 二 版

第二版 (2008 年 4 月)

本书适用于 Linux®、UNIX®、Windows®上的 IBM® DB2® Express-C Version 9.5

目录

关于本书

声明

IBM Canada
8200 Warden Avenue
Markham, ON
L6G 1C7
Canada

本书的读者对象

本书面向于那些从事或准备从事数据库相关工作的读者，比如数据库管理员（DBA）、数据库相关的开发人员、咨询人员、软件架构设计人员、产品管理人员、决策人员以及学生。

本书的架构

Part I，概览和安装。这一部分解释什么是 DB2 Express-C，介绍 DB2 产品家族的不同产品和各自的特性，并介绍 DB2 Express-C 的安装和数据库的创建，最后介绍 DB2 提供的各种工具。

Part II，阐述 DB2 的数据库管理，使您掌握关于 DB2 的环境、体系结构、远程连接、数据对象、数据迁移（import/export/load）、数据库安全、备份和恢复、并行和锁定以及其它常见的维护任务。

Part III，阐述 DB2 的应用开发。涵盖存储过程、用户定义函数、触发器、SQL/XML、XQuery 以及在使用 Java™, PHP 和 Ruby 开发过程中的 DB2 配置。

附录提供排除故障的有用信息。

许多章提供了实验，实验所需要的资料打包成压缩文件（expressc_book_quicklabs_9.5.zip）随书发布，也可以从 IBM® DB2 Express-C 网站上获得：http://www.ibm.com/db2/express。

书中的内容也用作为 DB2 on Campus 项目的课程材料，并与www.channelDB2.com/oncampus 上的学习视频相一致。同时，本书可以作为 DB2 on Campus 的考试复习资料，DB2 on Campus 考

试与一个 16 小时的 DB2 培训一同提供，考察对 DB2 的掌握程度。您可以在 DB2 Express-C 的网站上获得相关信息www.ibm.com/db2/express/students.html。

注意：
更多关于 DB2 on Campus 项目的信息，请参加下面视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:3902

一本属于社区的书

本书由 DB2 Express-C 团队创作，并免费向 DB2 Express-C 社区提供。社区中来自世界各地的许多成员将本书翻译成不同的语言，如果您想提供反馈信息、贡献新的材料、改进现有材料、或者帮助翻译本书到其它语言，请将您的计划发邮件到db2x@ca.ibm.com

本书的贡献者

我们要感谢来自社区的每位翻译志愿者，他们耗费了无数个日夜和周末，完成了不可思议的工作，使得本书成为现实。下表列出了每一位志愿贡献者。

贡献者的名字	所在公司/学校	职位名称	所作工作
Qijie Shen（申启杰）	广东工业大学, 广东, 中国	DB2 学生大使（DB2 Student Ambassador）	领导简体中文翻译团队，并且翻译了第 3 章、第 5 章、、第 6 章、第 9 章、第 10 章、第 15 章以及关于本书部分
Hanshu Shi（史晗枢）	吉林大学，吉林，中国	DB2 学生大使（DB2 Student Ambassador）	翻译了第 4 章、第 6 章、第 14 章
Xue Cao（曹雪）	厦门大学，福建，中国	DB2 学生大使（DB2 Student Ambassador）	翻译了第 7 章、第 13 章、第 16 章、第 17 章、附录
Dipu Zhang（张迪普）	厦门大学，福建，中国	DB2 学生大使（DB2 Student Ambassador）	翻译了第 11 章、第 12 章
Zhengang Liu（刘振刚）	广东工业大学, 广东, 中国	DB2 翻译志愿者	翻译了第 1 章、第 2 章、第 8 章
Xiaomei Wang	IBM 多伦多实验室	DB2 顾问（DB2 Consultant）	审校本书译文

致谢

我们真诚感谢下列的每个人，他们帮助和开发本书所引用的材料：

- 感谢来自 IBM 多伦多实验室的 Ted Wasserman, Clara Liu 和 Paul Yip 他们制作的材料成为本书的框架。
- 感谢 Don Chamberlin 和 Cindy Saracco 在 IBM developerWorks 关于 XQuery 的精彩文章,感谢 Matthias Nicola 关于 pureXML™的阐述。
- 感谢 Kevin Czap 和 Grant Hutchison 制作了 DB2 技术简报材料。
- 感谢 Katherine Boyachok 为本书设计了封面。
- 感谢 Susan Visser 对本书发布的帮助。

序

创新是技术进步的基石。在 IBM，创新是数据库服务器发展中极之重要的组成部分。早在上世纪六七十年代，我们作为这个领域的先锋，开拓了先进的数据管理技术，一直到现在，我们还一如既往的保持这种创新的精神，不断地在信息管理领域进行技术革新，单是数千个属于 IBM 技术专家的数据管理专利就很好的印证了这一点。经过我们不懈的努力，世界上的不少大型组织都依赖 IBM 的产品，比如 DB2，为他们提供满足最大需求的源动力和胜任严峻挑战的数据管理解决方案。

现在，DB2 并不在是一个只针对大型企业的数据库服务器。随着 DB2 Express-C 的发布，中小企业不需支付强制费用就能够享用 DB2 技术带来的丰厚回报。尽管当前还存在其它的免费或者开源数据库服务器，但 DB2 Express-C 所提供的独特优势使它成为最好的选择。

DB2 Express-C 为我们展现了很多技术创新，有些创新赋予了 DB2 新的能力，有些创新能减轻管理数据库的负担，有些创新改善数据库运行的性能，有些创新能减少基础设施建设的费用。我们不会在这里过多讨论它们，您能很容易在本书中发现它们。在此，我们只是简要地预告一下。

DB2 Express-C 基于 Viper 技术，它成为第一个能够以原始格式管理关系数据和 XML 数据的复合数据库。这使得 DB2 成为 SOA 和 Web 2.0 应用的理想选择，因为这些应用往往存在大量的 XML 数据流动。DB2 Express-C 不像其它的一些商业数据库，它不限制您存储在数据库中数据量的大小，也不限制您创建数据库数量的多少。而且，如果您需要 IBM 的帮助，那也会像点一下鼠标那样简单。

本书作为使用 DB2 Express-C 的起步学习和使用指南，能够很好地帮助您理解 DB2 的概念，并提高您管理 DB2 以及使用 DB2 进行应用开发的能力。本书所介绍的技巧和知识与 Linux、Unix、Windows 中高级版本 DB2 所需的知识和技巧是非常相近或一致的。

尽管 DB2 Express-C 不是一个开源的产品，在 IBM，我们深信支持和推动社区的首创精神。我很高兴能看到一本由 DB2 Express-C 社区成员创作的书，并且免费向社区所有成员提供。我非常感激你们，你们用所具有的知识和经验丰富和充实本书，并将本书翻译成多种语言，使得更多的人能够从你们的知识中获益。

Arvind Krishna
Vice President, Data Servers
Information Management, IBM Software Group

PART I – 概览

1

第 1 章 – DB2 Express-C 是什么?

DB2 Express-C 是 IBM DB2 系列产品的成员之一，DB2 是管理关系数据和 XML 数据的强大的数据库软件。DB2 Express-C 是则 DB2 的一个免费的、无限制且易操作的一个版本。DB2 Express-C 中的字母‘C’代表社区（Community）。DB2 Express-C 社区中的每位成员以各种方式相互帮助。DB2 Express-C 社区由各种设计、开发、部署或利用数据库解决方案的个人或公司组成。社区成员包括：

- 需要一个以建立独立的、C/S 结构的、基于 Web 的企业级应用为目标的开放标准数据库软件的应用程序开发者
- 想把一个功能齐全的数据库捆绑或嵌入其解决解决方案独立软件开发商（ISVs）、硬件提供商、基础设施提供商以及其它类型的方案提供者。
- 需要一个强壮的数据库服务器用来培训、开发技能、评估和制作原型的顾问、数据库管理员和 IT 架构师。
- 需要一个日常应用和操作的可靠数据库服务器的起步型、小型和中型企业。
- 为构建 Web 2.0 和下一代应用程序而寻找易用数据服务的数据库爱好者和先进技术热忠者
- 需要为教学、课程、工程和研究使用高度通用的数据库服务器的学者、教师和其它学术研究者

DB2 Express-C 具有与其它用于 Linux, UNIX, Windows 操作系统中非免费的 DB2 版本相同的核心功能和基本代码。DB2 Express-C 可以在 32 位或 64 位的 Linux 或 Windows 操作系统上运行。同时可以运行在拥有任何数量的处理器和存储器的系统上而没有任何特殊的存储器和系统安装需求。DB2 Express-C 也免费提供 pureXML。pureXML 是一项 DB2 原生地存储和处理 XML 文档的独特技术。

1.1 免费开发、部署和分发... 无限制!

这句话总结了 DB2 Express-C 的主要思想：

- **免费开发**：如果您是一个应用程序开发人员并且需要一个应用程序数据库，您可以使用 DB2 Express-C。
- **免费部署**：如果您工作在生产环境并且需要一个数据管理系统来存储您的重要记录，您可以使用 DB2 Express-C。
- **免费分发**：如果您正在开发一个应用程序或一个需要嵌入数据库的工具，您可以使用 DB2 Express-C。即使 DB2 Express-C 嵌入到您的应用程序中，出售您的应用程序并发布时它仍然是免费的。为了分发 DB2 Express-C 您必须在 IBM 注册，这个注册也是免费的。
- **无限制**：众多数据库竞争者提供数据库时，在数据库的规模、数量和用户数上都设有限制，而 DB2 Express-C 没有任何相关数据限制。您的数据库可以无限增长而不会违反协议许可，并且，关于连接数和每服务器上的用户数量也没有任何的限制。

注意：
更多关于 DB2 Express-C 以及它在信息需求世界和 Web2.0 中所扮演的角色，请参见以下视频：

http://www.channeldb2.com/video/video/show?id=807741:Video:3922

1.2 用户帮助和技术支持

有关 DB2 Express-C 的技术疑难，您可以在 DB2 Express-C 论坛上提交您的问题。这个免费论坛由一个专业的 DB2 Express-C 队伍管理，尽管这个论坛以自愿为基础向社区成员提供问题的答案。

IBM 也提供一个便宜的年度订购（也叫作 12 个月的许可和订购或 FTL，Fixed Term License）。这个 DB2 Express-C 的订购提供一天 24 小时，全年无休的技术支持和软件更新。一年的低价订购（例如在美国是$2,995/服务.年，价格会随国家而不同），除了技术支持和软件维护，同时也可以得到两个关键的附加功能：HADR（高可用性灾难恢复）和 SQL 复制（用于从其它的 DB2 服务器复制数据）。关于 DB2 Express-C 订购的进一步信息可以在下面网址找到：www.ibm.com/db2/express/support.html

1.3 DB2 服务器

所有的 DB2 服务器版本都包含相同的核心组件；用户可根据各自所需来选择不同价格的组件配置。图 1.1 图示了 DB2 产品的不同版本。

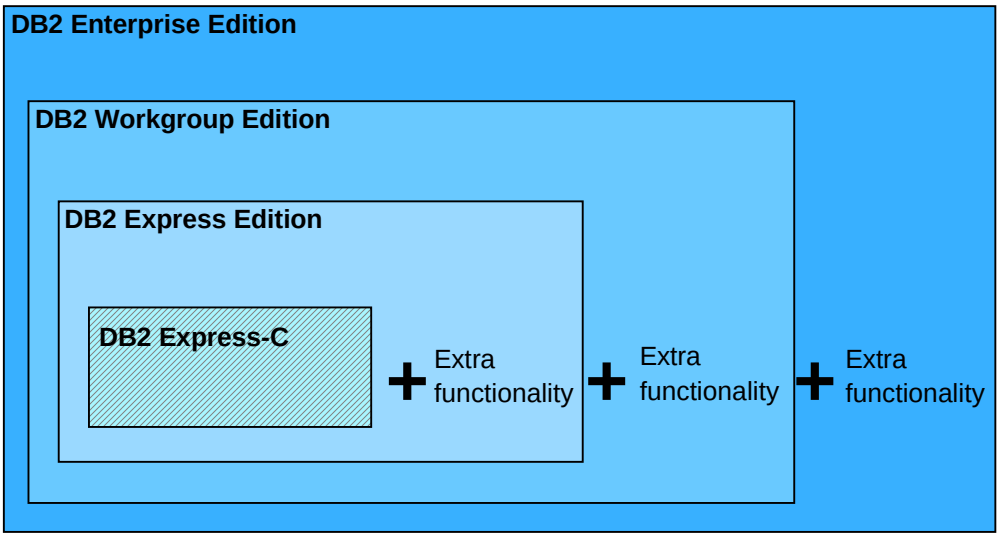


图 1.1 –DB2 服务器

如图 1.1 所示，DB2 Express-C 与去除某些组件的 DB2 Express 相同。DB2 Express-C 对社区是免费的。正如先前所提到的，可以通过免费论坛获得技术帮助或者如果您购买了 12 个月的订购许可就可以得到正式的 24 x 7 的 IBM DB2 技术支持.

图 1.1 也解释了容易从 DB2 Express-C 升级的原因。因为所有 DB2 服务器拥有相同的核心组件，如果您愿意，您可以将 DB2 Express-C 升级到任何 DB2 服务器版本。这也意味着基于某一 DB2 版本的应用程序可以无修改地运行在其它的高级的 DB2 版本上。而且从一个版本中所学到的技能同样适用于其它版本。

1.4 DB2 客户端和驱动

一个 DB2 客户端包含了连接到 DB2 服务器端的所需的功能；但是 DB2 客户端并不总是需要安装的。例如，JDBC Type 4 应用程序可以直接连接到 DB2 服务器端。DB2 客户端和驱动是以多种形式存在的：

- IBM 数据库服务器客户端：十分完善，包括 GUI 工具集和驱动程序集。
- IBM 数据库服务器运行时客户端：只拥有基本功能和驱动程序集的轻量级客户端。
- DB2 Windows 的运行时合并模块客户端：主要用来嵌入一个 DB2 运行时客户端以用作 Windows 安装应用程序的一部分。
- JDBC 和 SQLJ 的 IBM 数据库服务器驱动：允许 Java 应用程序连接到 DB2 服务器。
- ODBC 和 CLI 的 IBM 数据库服务器驱动：允许 ODBC 和 CLI 应用程序不用安装大型的客户端而直接连接到 DB2 服务器端。
- ODBC、CLI 和.NET 的 IBM 数据库服务器驱动：作为 ODBC 和 CLI 的补充，用来支持.NET 应用环境的 Windows 驱动。

图 1.2 展示了不同的 DB2 客户端和驱动

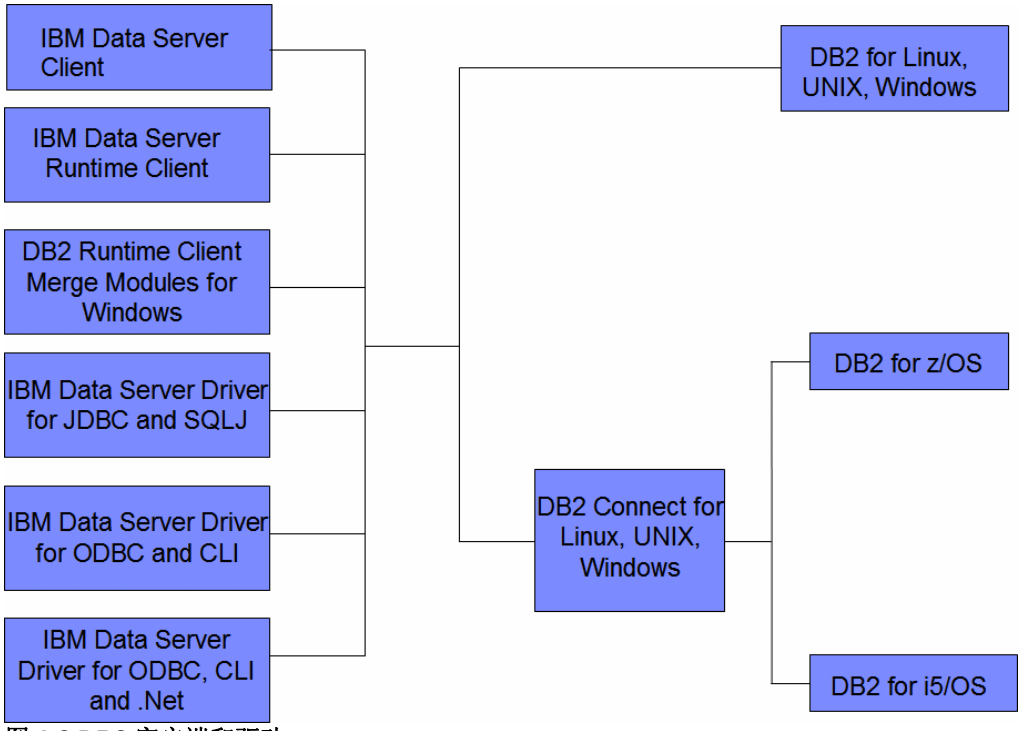


图 1.2 DB2 客户端和驱动

在图 1.2 的左边列出了所有 DB2 客户端和驱动。尽管所有的 DB2 客户端都包含了必需的驱动，但是从 DB2 V9 开始我们也提供了单独的驱动。DB2 客户端和驱动可以在 DB2 Epress-C 网站免费下载。客户端和驱动可以用来连接到 Linux,Unix 或 Windows 的 DB2 服务器上的。为了连接到 z/OS®或 i5/OS®的 DB2 服务器，则需要通过 DB2 连接服务器（DB2 Connect™ server，如图 1.2 中间部分所示）。我们将在第二章中讨论 DB2 的连接软件。

1.5 应用程序开发的自由性

DB2 提供了一个基于标准的和在 DB2 家族中是透明的应用程序开发环境。DB2 产品线中的 SQL 标准提供了通用的、用于数据存取的应用程序开发接口。另外，每个 DB2 产品都提供了 SQL 预编译器，它允许开发人员在移动应用程序开发中嵌入静态或动态 SQL。DB2 甚至有一个原生的、由.NET 管理的数据提供者（provider）并且与 Microsoft® Visual Studio 工具集成在一起。

在 DB2 开发中可以使用的语言和标准包括：

- SQL, XQuery, XPath
- C/C++ (CLI, ODBC and embedded SQL)
- Java (JDBC and SQLJ)
- COBOL
- PHP
- Perl
- Python
- Ruby on Rails
- .NET languages
- OLE-DB
- ADO
- MS Office: Excel, Access, Word
- Web services

1.6 DB2 版本号与 DB2 版本分类

如果您是一个 DB2 的初学者，您可能不太清楚 DB2 版本号与 DB2 版本分类之间的区别。

每几年，IBM 都会公开发布一个新的 DB2 版本号。新版本号包括新特性和对产品的重要改进。目前 DB2 Version 8 和 Version 9 均被 IBM 官方支持。一个版本号也许会发布一些虽新但不足以发布新版本号的新功能。例如 8.1 和 8.2 是 DB2 Version 8 的发布号。依过去来看，IBM 差不多每年都会发布一个新的 DB2，但是需要 2-3 年才有一个新的版本号。当前最新的发布号是 V9.5（先前以 DB2 'Viper 2'为代码编号）并在 2007 年十月普遍使用（Generally Available，GA）。每个发布号也会有一些修改号，但它们通常包含对缺陷的修复而不会有新的功能。当前最新的 DB2 Express-C 的版本好, 发布号，修改号 (V,R,M)是 9.5.0，表示代码级为 9.5，修复补丁为 0，也就是说这是一个普遍可以的版本（GA）。

另一方面，版本分类是每一个版本号中选取适当的组件进行打包以适应不同的功能需要。正如之前讨论，一个版本分类是一个依不同的价格和许可而打包成不同的功能集。DB2 V9.5（也叫做 DB2 9.5）有许多版本分类。例如，DB2 Express-C 9.5, DB2 Express 9.5, DB2 Workgroup 9.5, 和 DB2 Enterprise 9.5 (如图 1.1).

1.7 升级到其它的 DB2 版本

随着数据库需求的增加，您可能想升级到 DB2 的更高版本以支持更强大的硬件配置。这种情形下，它是很容易升级的:

- 如果您在相同的计算机系统中升级，请在 DB2 Express-C 之上安装新的 DB2 版本和相应的新的许可。您的数据库不会被删除（推荐先进行备份）。
- 如果升级时，要将新的版本安装到相同操作系统但硬件更强大的计算机时，请直接安装新的 DB2 版本在更强大的计算机上，然后从原来的计算机里备份您的数据库映像，并将其拷贝到新计算机上，再在新计算机中恢复数据备份镜像。您也需要从原计算机中保存实例配置设置（dbm cfg）并将这些设置应用到新系统中。备份和恢复命令的更多细节将在第 11 章-备份和恢复中讨论。dbm cfg 的更多细节将在第 5 章-DB2 环境中讨论。
- 不管是以上那一种情形您的程序都不需要修改。

1.8 DB2 Express-C 的维护

我们在前面曾经讨论过，DB2 Express-C 有两种技术支持选择：

1. 购买 12 个月的许可。将为您提供一个来自 IBM DB2 的全天候技术支持，同时能够安装 DB2 软件更新（也叫做 fixpacks）。

2. 使用 DB2 Express-C 的在线社区论坛。虽然完全免费但是得不到 IBM 的官方支持。同时 IBM 也不会负责定时提供新特性和修复补丁。关于补丁包（fixpack）的概念将在第 2 章中讨论。相反，IBM 会不时更新整个 DB2 Express-C 安装压缩包。当有新的版本发布时，您可以期待新的 DB2 Express-C 压缩包。

1.9 相关免费软件

从 DB2 Express-C 下载页(www.ibm.com/db2/express/download.html)下载的软件都是免费的。除了 DB2 Express-C(为 Linux 和 Windows, 32 位和 64 位系统结构均可)的压缩包之外，这里还有其它可以免费下载和使用的有用软件:

- IBM Data Studio
- DB2 Net Search Extender
- DB2 Spatial Extender

也有面向 DB2 Express-C 初学者的其他一些工具集，可以从 IBM Alphaworks 网络站点(www.alphaworks.ibm.com/datamgmt)上下载:

- Starter Toolkit for DB2 on Rails
- Web 2.0 Starter Toolkit for DB2

如果您正想要一个免费的轻量级的应用程序服务器，IBM 可提供：

- WebSphere® Application Server – Community Edition (WAS CE)

1.9.1 IBM 数据工作室（Data Studio）

IBM 数据工作室是一个基于 Eclipse 的工具，它允许您在整个数据管理生命周期里设计、开发、布署和管理您的数据、数据库以及数据库应用。数据工作室是以前 DB2 开发者工作台 9.1（Developer Workbench 9.1）解决方案的替代产品。

IBM 数据工作室帮助您开发用户定义函数、存储过程、XQuery、SQL 语句并提供一个集成的调试器。另外，数据工作室允许您用物理数据模型图来理解表之间的实体关系。数据工作室也可以帮您像 Web 服务一样开发和发布您的数据而不用编程。我们将在第 14 章-SQL PL 存储过程中讨论数据工作室。

1.9.2 DB2 Net Search Extender

用 DB2 Net Search Extender 可以对文本文件和原生存储在 DB2 中的 XML 文档执行快速和详细的全文本检索。

1.9.3 Starter Toolkit for DB2 on Rails

Starter Toolkit for DB2 on Rails是一系列产品和技术的便利包，它能够使用快速创建一个使用 Ruby on Rails 构建 DB2 Web 应用程序的环境。里面包括的的软件有：DB2 Express-C; DB2 driver for Ruby; DB2 adapter for Rails;同时有使用指南、示例和其它可学习材料。我们将在第 17 章- Java, PHP 和 Ruby 的开发中讨论 Ruby on Rails 的更多内容。

1.9.4 Web 2.0 Starter Toolkit for DB2

Web 2.0 Starter Toolkit for DB2是一个 DB2、PHP 和 Dojo 的上手工具。它帮助您配置必需的软件、链接到使用指南、并包含演示程序。两个演示程序分别 Atom Feed Control Panel 和 Web Services Control Panel。Atom Feed Control Panel 从 DB2 表中生成 Atom feeds；Web Services Control Panel 从 DB2 表中生成一个 REST 网络服务包。它们两个都依靠 Dojo 强大的 Ajax 和 widget 能力。

1.9.5 WebSphere Application Server – Community Edition

IBM WebSphere Application Server Community Edition 是一个轻量级的可免费获得的 Java EE 5 应用程序。基于 Apache Geronimo 技术。它利用开源社区的最新创新，来开发一个集成的、可用的和灵活的平台，用以开发和配置 Java 应用程序。可选的 WAS CE 技术支持可以从年度订制中获得。

2

第 2 章 – DB2 相关特性和产品

本章描述 DB2 的所有特性，如 DB2 Express-C 12 个月的订购许可，还有在其它 DB2 版本的其它特性，一般情况下，其它的这些特性需要支付额外费用。

DB2 Express-C 免费版本中包含的功能有:

- DB2 核心功能
- 控制中心, Data Studio 和其它的管理工具
- pureXML
- 达到 2GB 和 2 个 CPU 的资源利用
- 可用于 Linux、Windows、Solaris(x86)

在 DB2 Express-C 免费版中没有，但在 DB2 Express-C12 个月的订购许可版本中有的功能是：

- 补丁包
- 高可用性（High Availability）
- 数据复制 (Homogenous SQL)
- 达到 4GB 和 4 CPU 核心(2 插槽)的资源利用

下表列出了不同版本的 DB2 所具有的特性对比，您可以根据需要，选择下面所列出的不同 DB2 版本。

功能	Express-C fixed term license	DB2 Express Edition	DB2 Workgroup Server Edition	DB2 Enterprise Server Edition
Homogenous SQL Replication	Yes	Yes	Yes	Yes
Net Search Extender	Yes	Yes	Yes	Yes
Spatial Extender	Yes	Yes	Yes	Yes
pureXML™ technology	Yes	pureXML Feature	pureXML Feature	pureXML Feature
High availability disaster recovery	Yes	High Availability Feature	Yes	Yes

功能	Express-C fixed term license	DB2 Express Edition	DB2 Workgroup Server Edition	DB2 Enterprise Server Edition
Tivoli® System Automation	Yes	High Availability Feature	Yes	Yes
Advanced Copy Services	No	High Availability Feature	Yes	Yes
Online reorganization	No	High Availability Feature	Yes	Yes
Homogenous Federation	No	Homogeneous Federation Feature	Homogeneous Federation Feature	Homogeneous Federation Feature
MQT	No	No	Query Optimization Feature	Yes
MDC	No	No	Query Optimization Feature	Yes
Query parallelism	No	No	Query Optimization Feature	Yes
Connection concentrator	No	No	No	Yes
Table partitioning	No	No	No	Yes
DB2 Governor	No	No	No	Yes
Compression: row level	No	No	No	Storage Optimization Feature
Compression: backup	No	No	No	Storage Optimization Feature
Label-based access control (LBAC)	No	No	No	Advanced Access Control Feature
Geodetic Extender	No	No	No	Geodetic Data Management Feature
Query Patroller	No	No	No	Performance Optimization Feature

功能	Express-C fixed term license	DB2 Express Edition	DB2 Workgroup Server Edition	DB2 Enterprise Server Edition
DB2 workload management	No	No	No	Performance Optimization Feature
Performance Expert	No	No	No	Performance Optimization Feature
Homogenous Q Replication	No	No	No	Homogeneous Replication Feature
Database partitioning	No	No	No	No

表 2.1: DB2 9.5 版中，不同版本分类的特性和功能对比

在其它的 DB2 版本中可用的功能有：

付费的 DB2 Express 版功能

- pureXML
- 高可用性（High Availability）
- Homogenous Federation Feature

DB2 工作组版本免费提供的功能：

- 高可用性（High Availability）
- 可用于 AIX, Solaris,HP-UX，Linux，Windows

DB2 工作组版本付费提供的功能

- pureXML
- 查询优化特性(MQT、MDC、查询并行性)
- Homogenous Federation Feature

DB2 企业版提供免费的功能组件，如：

- Table (Range) Partitioning
- 具体化查询表(MQT，Materialized Query Tables)
- 多维群集(MDC，Multi-dimensional Clustering)
- 高可用性灾难恢复(HADR，High Availability and Disaster Recovery), 多机环境的系统自动化(Tivoli® System Automation)
- 连接集中器（Connection Concentrator）

DB2 企业版付费提供的功能

- pureXML
- 存储优化功能 (包括压缩)
- 高级存储控制 l (fine grained 和高级安全性)
- 性能优化 (Workload Management, Performance Expert, Query Patroller)
- Geodetic Data Management (geographical location analysis)
- Homogenous Federation Feature (access remote DB2 and Informix® data as local tables)

相关的 DB2 收费产品:

- DB2 Connect
- DB2 数据仓库版
- WebSphere® Federation Server
- WebSphere Replication Server

2.1 DB2 Express-C 订购中包含的功能

这一节主要介绍 DB2 补丁包（Fix pack）,HADR 和 SQL 复制。

2.1.1 Fix packs 补丁包

DB2补丁包（Fix pack）是一组附加到已安装DB2产品的代码集合，用于修复产品发布之后所发现的代码问题。如果已经安装了订购许可，DB2补丁包（Fix pack）可以免费下载和安装。通常每三个月更新一次。

欲下载最新的 DB2 补丁包（Fix pack），请查看 DB2 技术支持站点：
http://www.ibm.com/software/data/db2/support/db2_9/

2.1.2 高可用性灾难恢复（HADR）

高可用性灾难恢复（HADR）是一个提供数据库可靠性的功能，它为系统的部分或整个站点的故障提供一个高可靠的灾难恢复解决方案。一个 HADR 环境通常由两个数据服务器组成，一个主服务器和一个次服务器（它们可以是地理位置物理分开的）。主服务器中存有源数据库并可被客户端应用程序访问。当事务在主服务器中被处理时，数据库的日志记录就通过网络自动被记录到次服务器中。次服务器中有主数据库的完全克隆，这个克隆是由备份主数据库并在次服务器系统中恢复而产生的。当次数据库接收到主服务器日志记录时，它就将刷新并应用到次数据库中。通过这样不断应用日志记录对次服务器进行更新，在次数据库中保存有主数据库的同步复制数据库，这样，当主数据库出现故障时就可以由次数据库接管任务。

完全的DB2支持HADR解决方案有：

- 对用户和客户端应用程序来说透明的极速故障恢复
- 完全的原子事务防止数据丢失
- 不需要中断服务的升级系统和应用程序的能力
- 远程系统故障恢复,提供从破坏数据中心的本地灾难中完全恢复
- 用DB2 图形工具方便管理

> **注意：**
> 请连接一下视频来观看关于HADR如何工作的视频：
> http://www-306.ibm.com/software/data/db2/express/demo.html

- 以上所有这些对整体性能不会产生任何影响

2.1.3 数据复制（Data Replication）

这个功能允许在源服务器和目标服务器之间复制数据。源数据库的数据变化会被捕获，然后应用到目标服务器上。图 2.1 演示了复制工作的整个过程。

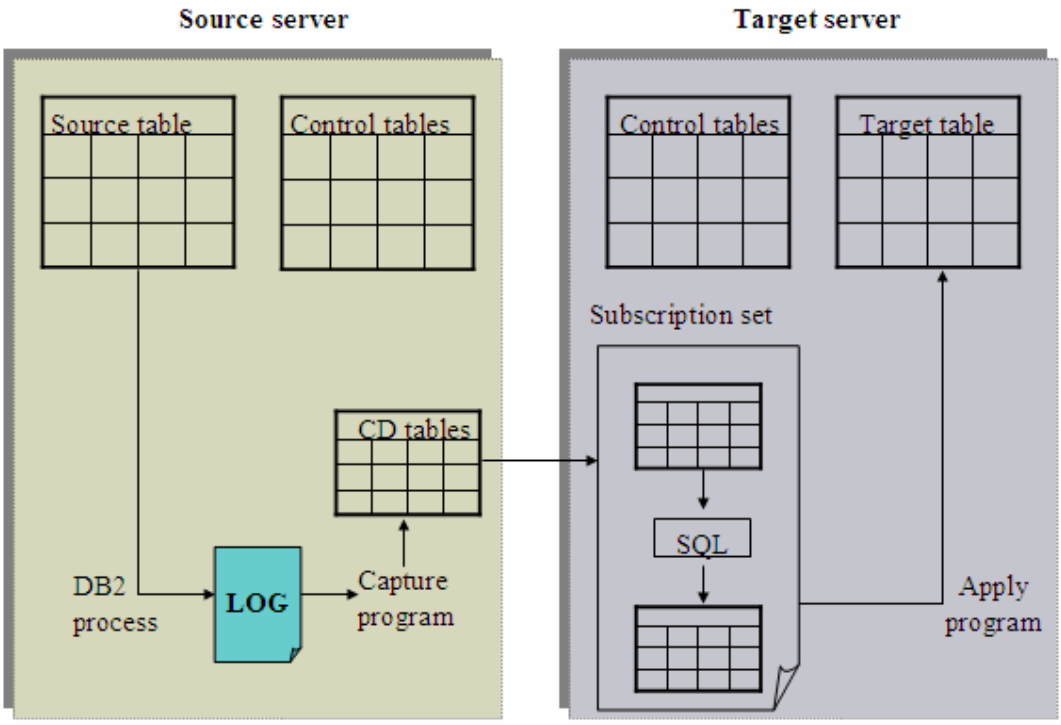


图 2.1 – SQL 复制

在图 2.1 中有两个服务器，一个源服务器一个目标服务器。在源服务器中，有一个捕捉程序可以捕捉到数据库中发生的数据改变。在目标服务器中有一个应用程序，它可以把数据改变应用到目标服务器中。不论是缓和压力、充填数据仓库或是数据集市还是审计数据历史，在需要复制数据的地方，这个功能都是非常有用的。使用 SQL 复制功能可以在 DB2 Express-C 和包括运行于 Linux, UNIX, z/OS, 和 i5/OS 系统的 其它 DB2 服务之间复制数据。

2.2 DB2 Express-C 所不具备的功能

这一节描述在 DB2 的其它版本有的而在 DB2 Express-C 中不具有的功能。

2.2.1 数据库分区

只有 DB2 的数据仓库版才具有数据库分区功能（DPF）。它允许数据库扩展到分布在不同计算机上的多个分区中。DPF 是基于无共享体系结构的。每个连接到分区组中的计算机需要用自己的 CPU 和内存来进行数据处理。DPF 在处理大量数据的环境下是非常有用的，如在运行决策支持系统（DSS）的数据仓库时。

2.2.2 连接集中器（Connection Concentrator ）

连接集中器支持同一时间的大量用户连接。以前，每一个数据库连接都需要一个数据库代理。数据库连接集中器引入“逻辑代理”的概念允许一个代理处理多个连接。代理的更详细内容将在第 6 章-DB2 体系结构中进行讨论。

2.2.3 Geodetic Extender

DB2 Geodetic Extender 是 DB2 企业版中可选的收费功能。这个扩展为需要更容易分析地理位置的商务智能和电子政务的应用提供了便利。DB2 Geodetic Extender 可以构建任何规模的虚拟地球。大部分的位置信息是由 GPS 等全球系统提供的，并可以表现为经度/纬度的形式（geocode）。

业务数据，如地址等，可以由 DB2 Geodetic Extender 转换为 geocode，企业应用在这些未规格化的数据上工作的更好，而不用在表现层规格化为地图数据（平面地图）以将它们显示和打印出来。

2.2.4 工作负载管理(Workload Management, WLM)

工作负载管理基于用户和应用优先度，并结合资源可用和工作量极限进行数据库范围的管理。它允许您调节自己的数据库工作量和查询，以使重要的和高优先级的查询可以得到及时的运行，并可以防止‘流氓’查询独占系统资源，确保系统有效运行。WLM 是 DB2 9.5 中提供的一个新功能，比 DB2 先前版本中 Query Patroller 和 DB2 Governor tools 更强大。

2.3 DB2 相关收费产品

2.3.1 DB2 连接（DB2 Connect）

DB2 连接是允许 Linux, UNIX 或 Windows 客户端连接连接到 z/OS 或 i5/OS 服务器端的收费产品，如图 2.2 所示。DB2 连接在相反的连接时是不需要的，即从 z/OS 或 i5/OS DB2 连接到 Linux, UNIX 或 Windows 的服务器端不需要 DB2 连接。DB2 连接有两个可选的主要版本：DB2 连接个人版和 DB2 连接企业版。

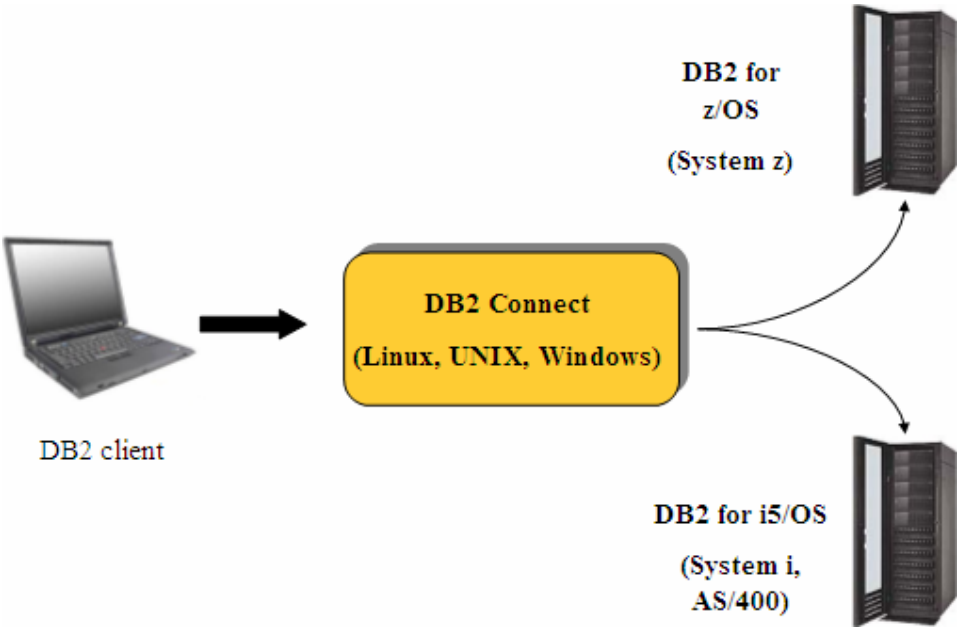


图 2.2 –DB2 连接

2.3.2 WebSphere Federation Server

以前称为 WebSphere Information Integrator(提供联合支持)，现在称为 WebSphere Federation Server，它允许数据联合，意味着可以对来自不同的关系数据库系统的进行数据库查询。例如，如果您购买了 WebSphere Federation Server，您可以运行下列查询语句：

```
SELECT  *
FROM      Oracle.Table1  A
             DB2.Table2  B
       SQLServer.Table3  C
WHERE
            A.col1 < 100
```

```
        and B.col5 = 1000
        and C.col2 = 'Test'
```

图 2.3 是一个 WebSphere Federation Server 演示

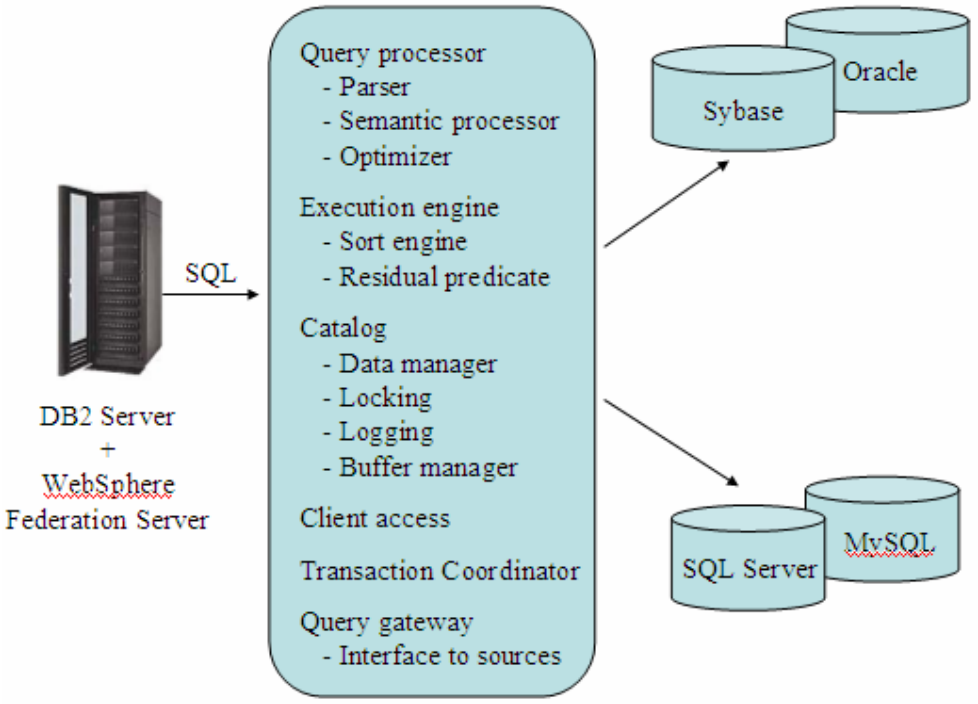


图 2.3 – WebSphere Federation Server

对于属于 IBM 的数据库管理系统，已经内建了联合支持，这意味着不需要安装 WebSphere Federation Server 就能够在不同的 IBM 数据库中进行联合操作。例如，欲在两个不同的 DB2 数据库间或一个 DB2 数据库和一个 Informix（Informix 是 IBM 家族的成员）数据库间执行查询时不需要 WebSphere Federation Server。

2.3.3 WebSphere Replication Server

以前称为 WebSphere Information Integrator（提供复制支持），现在称为 WebSphere Replication Server，它允许在涉及非 IBM 数据服务时复制数据库记录。它也包括著名 Q-Replication，Q-Replication 使用信息队列来复制数据。

3

第 3 章 – 安装 DB2

为了将 DB2 Express-C 安装到您的系统上（Linux 或 Windows），您的系统必须满足安装的前提条件。

3.1 安装前提条件

DB2 Express-C 能够满足大多数的应用场合，它有对应 Linux、Solaris（x64）、Windows 2000/2003/xp/Vista 的版本，而且不同的版本细分为 32bit、64bit、PowerPC（Linux）以对应不同的 CPU 架构。如果您想在其它的平台上（如 Unix）运行 DB2，您必须购买前面提到的其它版本的数据库服务器。下面链接的文档中详细描述了 DB2 不同版本对操作系统的要求。
http://www.ibm.com/software/data/db2/udb/sysreqs.html

在硬件方面，DB2 Express-C 能够安装在任意 CPU 核心和内存的系统上。但必须注意，免费版本的 DB2 Express-C 只能够利用最大 2 核的 CPU 和 2G 的内存，付费版本则能够使用最大 4 核的 CPU 和 4G 的内存。DB2 能够安装在实际的系统中，或者是虚拟机上。当然，您也可以在更少资源的系统上运行它，比如在单 CPU 和 1G 内存的机器上运行。

关于硬件要求的最新消息，请查阅 DB2 Express-C 的网站：
http://www-306.ibm.com/software/data/db2/express/getstarted.html

3.2 操作系统中的安装权限

要将 DB2 Express-C 安装到 Linux 或者 Windows 上，您必须拥有一个足够权限的帐户。

在 Linux 系统中，您必须是 root（超级用户）用户来安装 DB2 Express-C。当然，您可以使用其它的帐户来安装 DB2，但是在使用上会受到限制，比如，使用非 root 帐户安装的 DB2 Express-C 不能创建除安装时创建的默认实例以外的实例。

在 Windows 系统中，安装所用的帐户必须属于 Administrators 组。当然，非 Administrators 组的用户也能够安装 DB2 Express-C，只要系统的管理员授予了非管理员的软件安装权限。

对于 Windows 的域帐户，为了在 DB2 服务器上验证用户的 ID，安装所用的用户 ID 必须属于域中的 Domain Administrators 组。您也可以使用内建的本地帐户（Local System account）来安装。

安装所用的帐户必须有“从网络访问此计算机”的权限。

注意：
下面链接是一个关于 DB2 Express-C 安装的介绍
http://www.channeldb2.com/video/video/show?id=807741:Video:4442

3.3 安装向导

尽管安装 DB2 Express-C 有不同的方法，然而最简单的方法就是使用一个有界面的安装向导。在下载和解压 DB2 Express-C 后，您可以运行里面的安装向导：

- Windows: 运行在 EXP/image 文件夹中的 setup.exe

- Linux: 运行在 exp/disk1 文件夹中的 db2setup 命令

跟着 DB2 安装向导，整个安装过程是非常简单的。在很多时候，默认的设置已经足够了，所以您只需要接受协议，然后点击“Next”几次，最后点击“Finish”按键。几分钟后，安装完成，DB2 就安装好了，并且开始正常运行。

图 3.1 展示了 DB2 安装启动面板（DB2 Setup Launchpad）。点击“安装一个产品（Install a Product）”，然后选择“全新安装（Install New）”，就可以将一个新的 DB2 Express-C 安装到您的系统中。

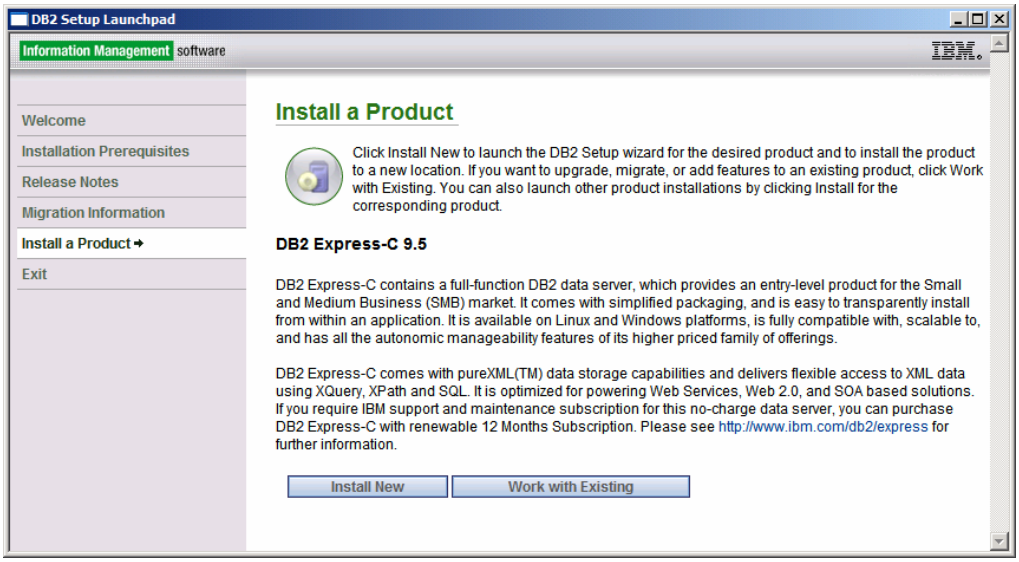


图 3.1 – DB2 安装启动面板（DB2 Setup Launchpad）

接受协议之后，出现安装类型，通常选择“典型（Typical）”安装（默认）就足够了，如图 3.2 所示。

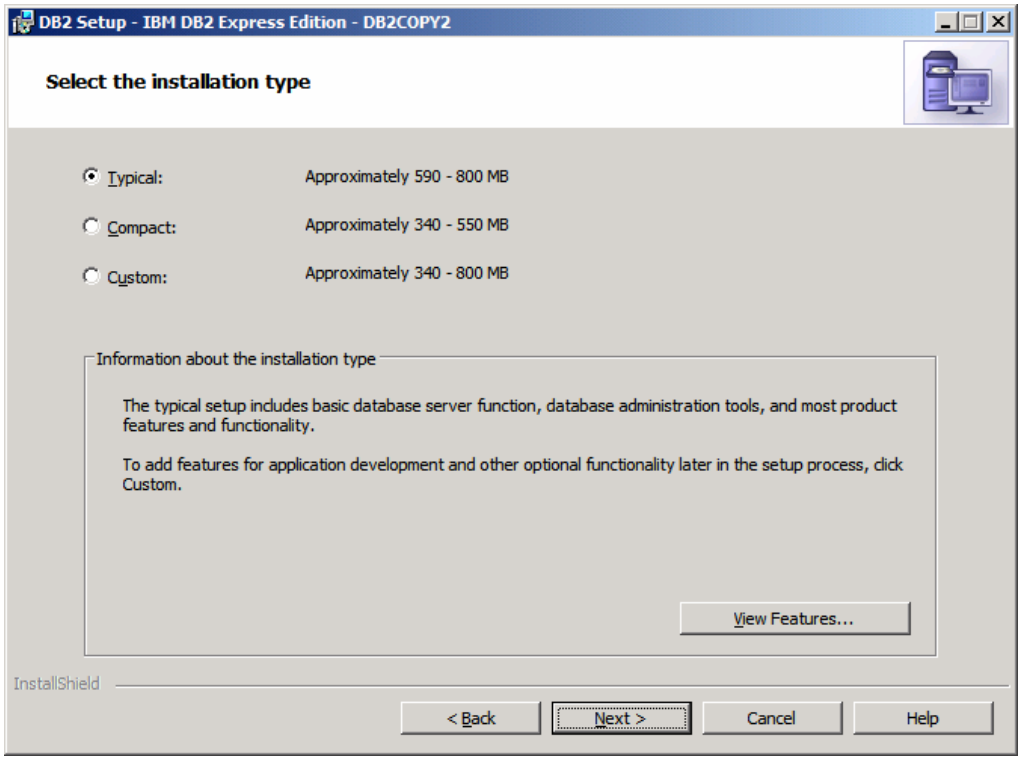


图 3.2 – 安装类型

在图 3.3 的画面中，您可以选择安装、建立响应文件、安装并建立响应文件。响应文件在 3.4 节讨论。在这里，默认的选项（Install IBM DB2 Express Edition on this computer and save my settings in a response file）已经足够。

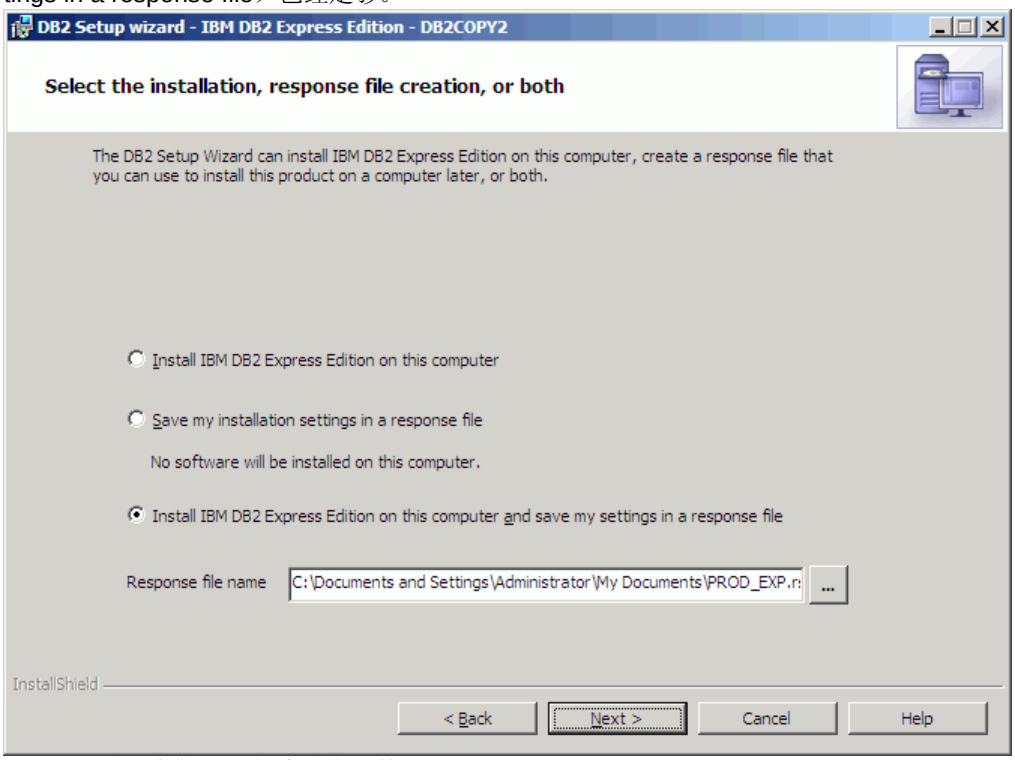


图 3.3 – 选择安装还是创建响应文件

在接下的几步都选择默认设置，当您到了如图 3.4 的窗口时，您可以输入一个存在的用户，这个用户将会使用 DB2 的实例和其它服务。这个用户必须是 windows 中本地管理员（Local Administrator）组的一员。如果您输入的用户 ID 不存在，这个用户 ID 就会被创建，并成为一个本地管理员。如果用户不属于任何的一个域，请将域（domain）留空。Windows 中，默认创建的用户名是 db2admin，在 Linux 中，默认创建的用户名是 db2inst1。

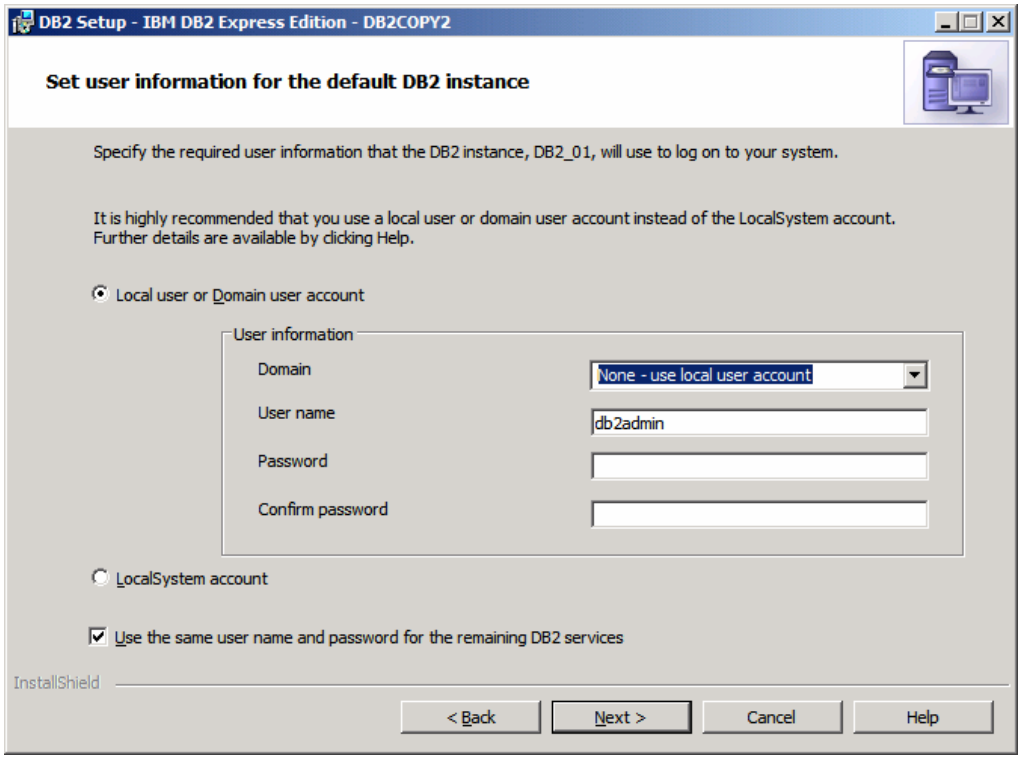


图 3.4 – 为默认的 DB2 实例配置用户信息

最后，如图 3.5，安装向导显示安装摘要，摘要中显示了所要安装的内容以及每一个阶段的设置信息。单击“Finish”执行安装，程序文件将会安装到您的系统中。

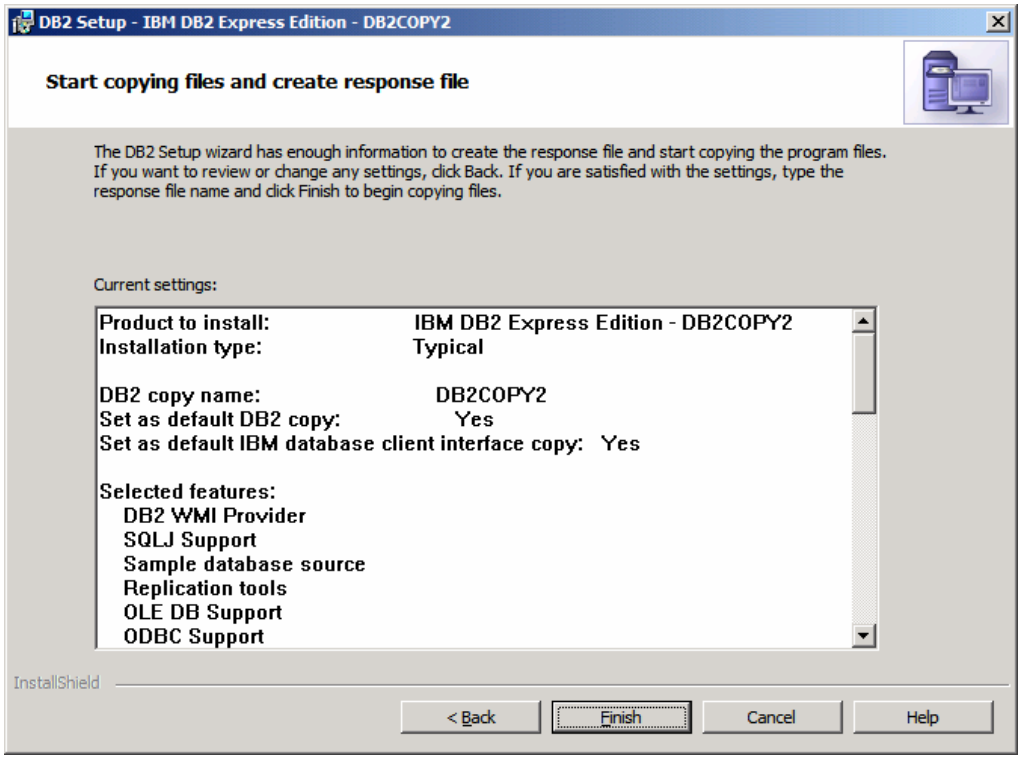


图 3.5 – 安装摘要

3.4 自动安装

有时，您需要将 DB2 客户端安装到多台机器上，又或者您需要将 DB2 数据库服务器嵌入到您的应用程序，并在安装这个应用程序同时安装 DB2 数据库服务器。这些情况下，DB2 的自动安装是一个理想的方法。

DB2 利用响应文件来进行自动安装，响应文件是一个文本文件，它保存了安装所需的信息。下面展示了响应文件的一小部分。

```
PROD=UDB_EXPRESS_EDITION
LIC_AGREEMENT=ACCEPT
FILE=C:\Program Files\IBM\SQLLIB\
INSTALL_TYPE=TYPICAL

LANG=EN

INSTANCE=DB2
DB2.NAME=DB2
DEFAULT_INSTANCE=DB2
DB2.SVCENAME=db2c_DB2
DB2.DB2COMM=TCPIP
...
```

有许多方法创建响应文件：

- 在安装 DB2 Express-C 过程中，第一步允许您选择将安装信息保存到响应文件中，如图 3.3 所示，您可以设定响应文件的保存路径和文件名。在向导结束后会在您指定的路径中生成响应文件。响应文件是一个文本文件，您可以用文本编辑器对它进行编辑。
- 修改 DB2 中提供的响应文件样例。响应文件样例（.rsp 扩展名）在 db2/platform/samples/文件夹中。
- 在 Windows 中，您还可以使用相应文件生成器来生成响应文件:

  db2rspgn –d <output directory>

有了响应文件后，可以用响应文件来自动安装 DB2，在 Windows 中，您运行下面的命令来执行安装：

```
setup -u <response filename>
```

在 Linux 中，您运行下面的命令进行自动安装：

```
db2setup -r <response filename>
```

实验 #1 安装 DB2 Express-C，创建 SAMPLE 数据库

实验目标：

在您开始探索 DB2 Express-C 的各项特性，使用各种工具之前，您必须将 DB2 Express-C 安装到系统中。在本实验，您将会在 Windows 系统中进行基本的 DB2 Express-C 安装，每一步都是非常简单的。

实验过程

1. 获得 DB2 Express-C 安装压缩包。可以在 DB2 Express-C 网站（ibm.com/db2/express）上下载合适的 DB2 Express-C 安装压缩包或者订购包含安装程序压缩包的 Discovery Kit DVD。然后将获得的安装压缩包解压缩到自定义的路径。

2. 定位文件。浏览到第一步解压的路径，找到 DB2 的安装文件。

3. 运行 Launchpad。双击 setup.exe 运行 DB2 Launchpad。如果是 Linux 系统，请以 root 身份运行 db2setup 命令。在 Launchpad 窗口左边的面板中，单击 Install Product 选项。

4. 运行 DB2 安装向导。DB2 安装向导会先检查系统是否满足安装条件，并检查系统是否已经安装过 DB2。单击 Next 按钮进入下一步。

5. 查看许可证。查阅并接受许可协议（选择“我同意（I Accept）”），然后单击 Next。

6. 选择安装类型。本实验中，选择典型(Typical)选项（默认的选项）。精简 Compact 选项会执行一个最基本的安装，而自定义 Custom 选项则允许您自己定义安装的细节。点击 Next 进入下一步。

7. 选择安装的文件夹。这里允许您制定 DB2 在本机上安装的盘符和路径。请确保目标路径有足够的空间。本例中使用默认的驱动器和路径。如下所示：

   Drive: C:
   Directory: C:\Program Files\IBM\SQLLIB

   点击 Next 进入下一步。

8. 设置用户信息。当 DB2 Express-C 安装完成后，一些 DB2 进程会作为系统服务运行。因为这些服务需要一个操作系统帐户来运行，所以必须设置必要的用户信息。在 Windows 系统中，推荐使用默认的 db2admin。如果指定的用户不存在，DB2 则会在系统中新建一个用户。您可以指定一个已经存在的系统帐户，不过这个系统帐户必须有本地管理员权限。我们推荐您使用默认的用户名，并确保输入了密码。在 Linux 中推荐使用默认的 db2inst1 作为实例所有者，db2fenc1 作为执行存储过程的隔离用户，dasusr1 作为 DB2 管理服务器 DAS 用户。单击 Next 进入下一步。

9. 配置 DB2 实例。一个 DB2 实例可以看成是数据库的容器。实例必须先于数据库存在，数据库是在实例中创建的。Windows 版本的 DB2 Express-C 安装时，会自动创建名为 DB2 的实例。在 Linux 中，默认的实例名字为 db2inst1。我们会在后面的本书章节详细阐述 DB2 的实例。默认的情况中，DB2 实例监听的 TCP/IP 连接端口是 50000。默认的协议和端口号可以通过单击协议（*Protocols*）和开始（*Startup*）按键来修改。我们推荐使用默认的设置。单击 Next 进入下一步。

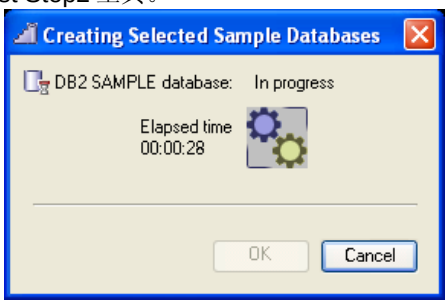
10. 开始安装。在摘要中检查之前每一步的选项，单击 Install 按钮开始将文件复制到安装路径中。DB2 同时会执行一些初始化的配置。

11. First Steps。当安装完成后，会显示一个叫 First Steps 的工具。First Steps 工具也可通过 db2fs 来启动。

12. SAMPLE 数据库是本书用于测试和实验目的的数据库。它会在 DB2 安装完成后自动创建。可以在 DB2 控制中心中检查这个数据库时候存在。要打开 DB2 控制中心，从 Windows 的开始菜单开始依次点击：*Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> General Administration Tools -> Control Center*。

13. 如果 SAMPLE 数据库出现在控制中心里面，您可以直接跳到第 16 步继续实验。如果控制中心里没有显示 SAMPLE，请您选择视图（View）菜单中的刷新（Refresh）选项，确保您查看的信息是最新的信息。如果 SAMPLE 数据库依然没有显示，它可能没有被创建。您可以通过 First Steps 工具手工创建它。在 First Steps 中选择“创建数据库（Database Create）”标签，然后根据向导来创建 SAMPLE 数据库。创建过程中确保 XML 和 SQL 对象和数据的选项被选中，并点“OK”，最后的一个选项会创建一个 UNICODE 数据库，这个数据库用在 DB2 V9，以支持 pureXML，但是在 DB2 V9.5 中并不需要。

14. 下面的进度窗口显示数据库正在创建（这个过程可能会花几分钟）。当数据库创建成功后，单击 OK 按键并关闭 First Step2 工具。



15. 回到控制中心，在对象树面板中再次检查 SAMPLE 数据库，您可能需要刷新控制中心来得到最新的更改。要刷新控制中心，请您选择视图（View）菜单中的刷新（Refresh）选项。

16. 重启系统。尽管这一步并没有在 DB2 官方安装文档中提及，我们还是推荐您重启系统（在可能的情况下）来确保所有的进程都成功启动，并且清除安装过程中可能没有正确释放的内存占用。这一步是**可选**的。

4

第 4 章 – DB2 的应用环境

本章讨论 DB2 的应用环境。图 4.1 是对 DB2 的综合性概述，红色椭圆区域标注了本章重点关注的内容。图的左侧部分介绍了不同的 DB2 命令: SQL，SQL/XML，和能与 DB2 数据服务器相交互的 XQuery 语句。图表的中间部分显示了与 DB2 数据服务器相交互所应用的不同工具。图表的右侧部分显示了基本的 DB2 环境所涵盖的内容：实例，数据库和相关的配置文件。

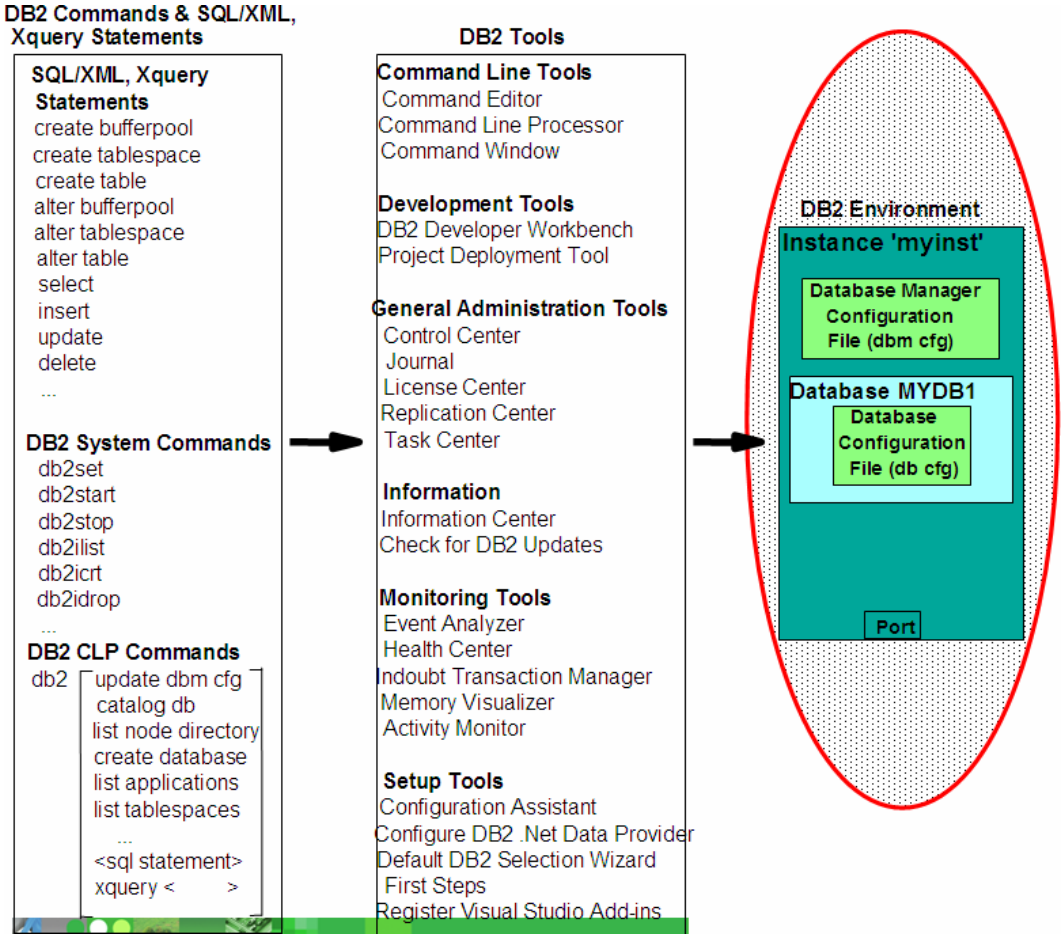


图 4.1– DB2 概览图：DB2 应用环境

注意：
更多关于 DB2 应用环境的信息，请参见以下视频
http://www.channeldb2.com/video/video/show?id=807741:Video:4029

http://www.channeldb2.com/video/video/show?id=807741:Video:4042

为了讲述 DB2 应用环境，我们可以将会逐步介绍 DB2 的每个组件。
图 4.2 的灰色方框表示安装完 DB2 Express-C 9.5 之后的 DB2 数据服务器。

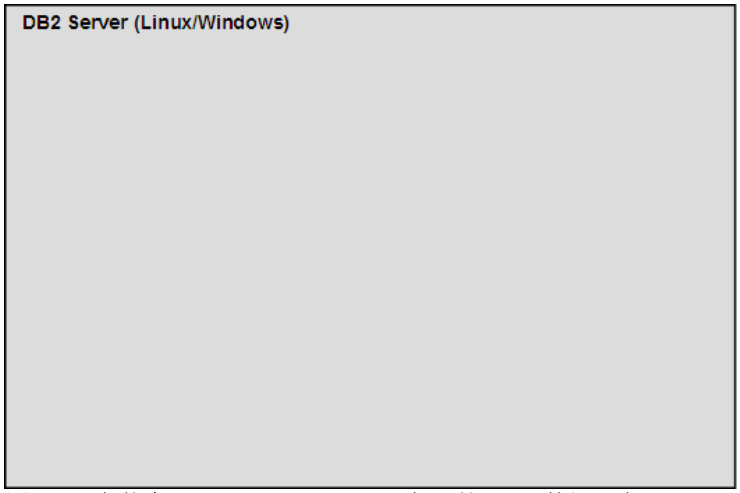


图 4.2 - 安装完 DB2 Express-C 9.5 之后的 DB2 数据服务器

在 Windows 系统中，安装时默认创建一个实例名称为“DB2”（Linux 系统下创建的实例名称为“db2inst1”）。如图 4.3 的绿色部分。实例是应用程序运行和创建数据库所必须的独立环境。一个数据库服务器上可以创建多个用于不同目的实例。例如，一个实例用于针对产品数据库的保存，再有一个实例用于测试环境数据库，还可以有一个实例用于开发环境。所有的这些实例都是相互独立，也就是说在一个实例上实现的操作不会影响到其它实例。

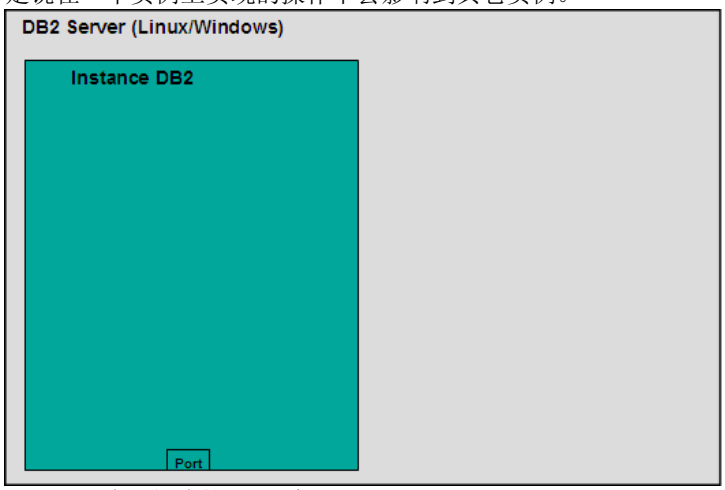


图 4.3 – 默认创建的 DB2 实例

可以用如下命令新建数据库实例，db2icrt <实例名称>，这里的<实例名称>可以用任意 8 个字符代替。例如，创建一个名为 *myinst* 的实例其命令为：db2icrt myinst。
图 4.4 分隔的绿色区域显示了一个名为 myinst 的新数据库实例。

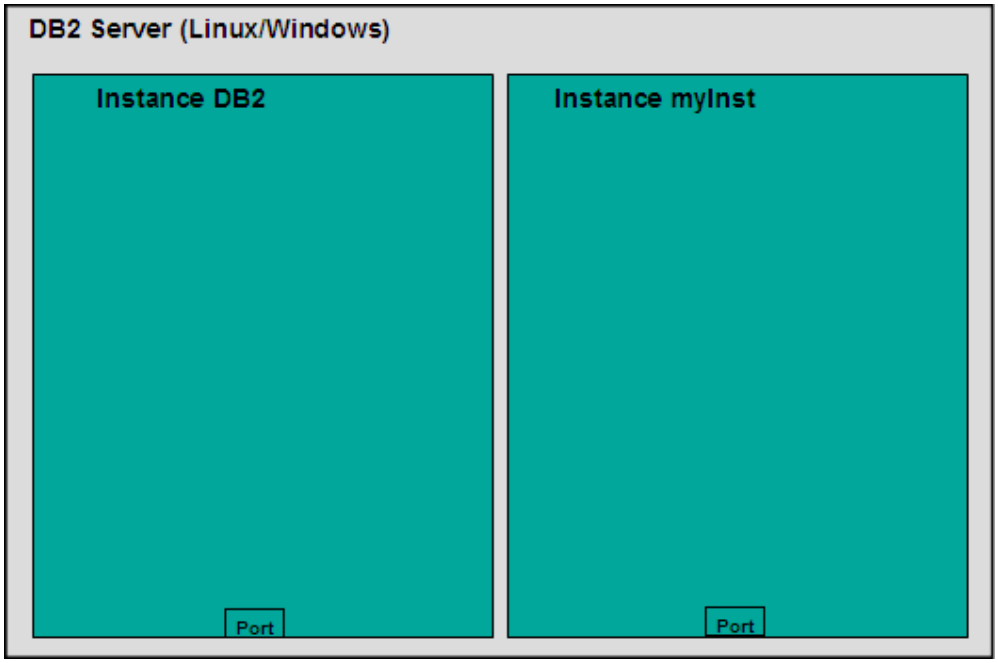


图 4.4 – 一个包含两个数据库实例的 DB2 服务器

必须注意的是每一个实例的端口号必须是唯一的。这样在您远程连接数据库时，此端口号可以保证您能够正确的连接实例。在 Windows 中可以使用 DB2 命令窗口，输入如下命令来激活任意 DB2 实例：

```
set db2instance=myinst
```

这样，如果您现在从命令窗口创建数据库，数据库会在 myinst 实例中被创建。运行如下命令显示所有实例：

```
db2ilist
```

在 Linux 操作系统上，每一个实例必须对应一个 Linux 操作系统用户，因此，两个实例的转换可以通过转换用户得以简单的实现（用 su 命令）。

表 4.1 显示了一些常用的实例层命令。

命令	描述
db2start	启动当前实例
db2stop	停止当前实例
db2icrt	创建一个新的实例
db2idrop	删除一个实例
db2ilist	显示系统您当前的所有实例清单
db2 get instance	显示当前运行的实例

表 4.1 – 实例层上常用的 DB2 命令

以上一些命令也可以通过控制中心（Control Center）来执行。例如，在控制中心，展开实例文件夹（*Instances*），右键单击所要操作的实例，接着选择 *Start* 来启动该实例，这一操作就等同于在 DB2 命令窗口中执行 db2start 命令；或者选择 *Stop* 来停止该实例，这一操作则等同于在 DB2 命令窗口中执行 db2stop 命令，如图 4.5 所示。

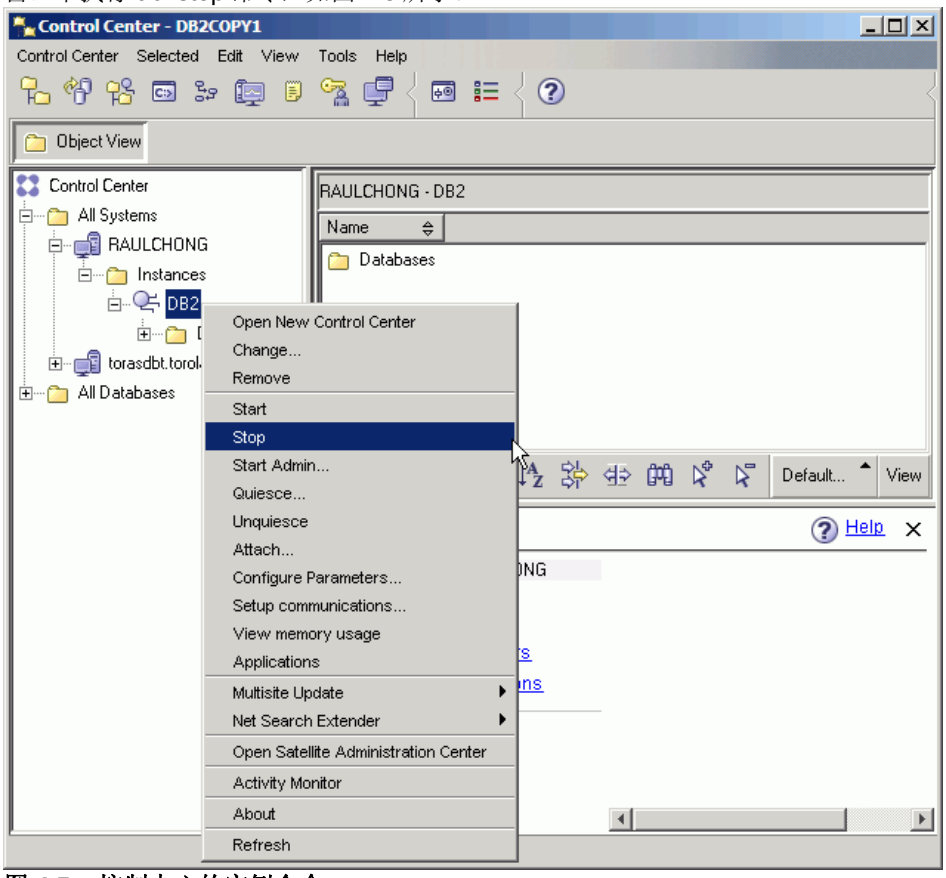


图 4.5 – 控制中心的实例命令

在当前运行实例中创建一个数据库，可以在 DB2 命令窗口执行如下命令：

```
db2 create database mydb1
```

若要显示所有已创建的数据库可以执行如下命令：

```
db2 list db directory
```

在任一实例中都可以创建多个数据库。数据库是诸如表，视图，索引等对象的集合。数据库之间是相互独立的单元，因此一个数据库并不与其他数据库共享内部对象。图 4.6 显示了在“DB2”实例中创建的数据库“MYDB1”。

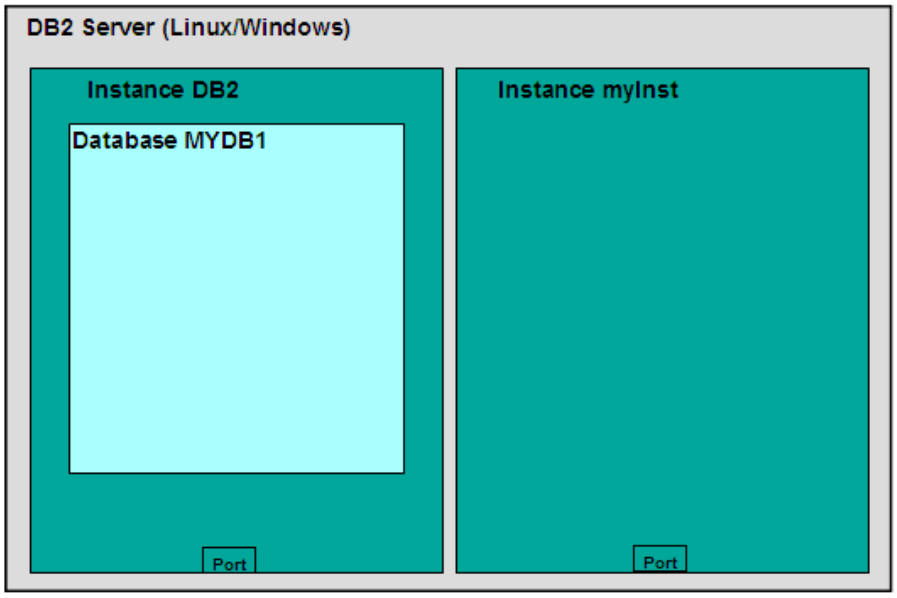


图 4.6 – 在实例“DB2”中创建的数据库“MYDB1”。

表 4.2 显示了一些应用于数据库层的命令。

命令/SQL 语句	**描述**
db2 create database	创建一个新的数据库
db2 drop database	删除一个数据库
db2 connect to <database_name>	连接数据库
db2 create table/create view/create index	分别创建表，视图，和索引的 SQL 语句。

表 4.2 – 数据库层的命令/SQL 语句。

如果想要在“myinst”实例中创建相同名称（MYDB1）的数据库，可以在 DB2 命令窗口执行如下的命令：

```
db2 list db directory
set db2instance=myinst
db2 create database mydb1
set db2instance=db2
```

图 4.7 展示了在“myinst”实例中创建的新数据库“MYDB1”。

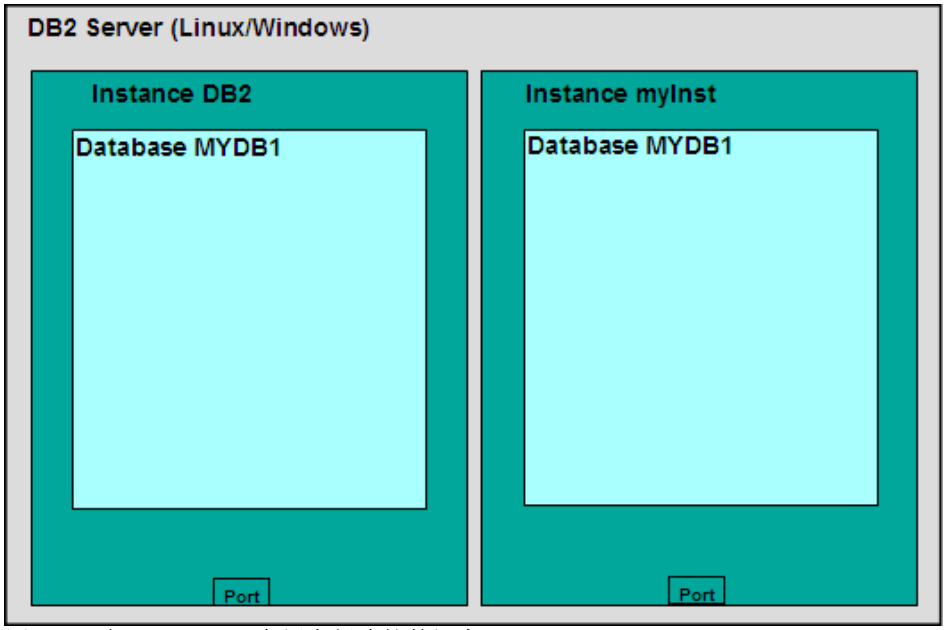


图 4.7 – 在“myInst”实例中创建的数据库“MYDB1”。

随着数据库的创建，有几个默认的对象也同时被创建：表空间，表，缓冲池，日志文件。因为创建这些对象需要一点时间，所以执行数据库创建过程需要几分钟。图 4.8 显示了三个默认被创建的表空间。在第 6 章 DB2 体系结构中会详细介绍表空间的相关细节，现在，可以把表空间看成是处于逻辑表和物理资源之间（如类似硬盘、内存等）的逻辑层。

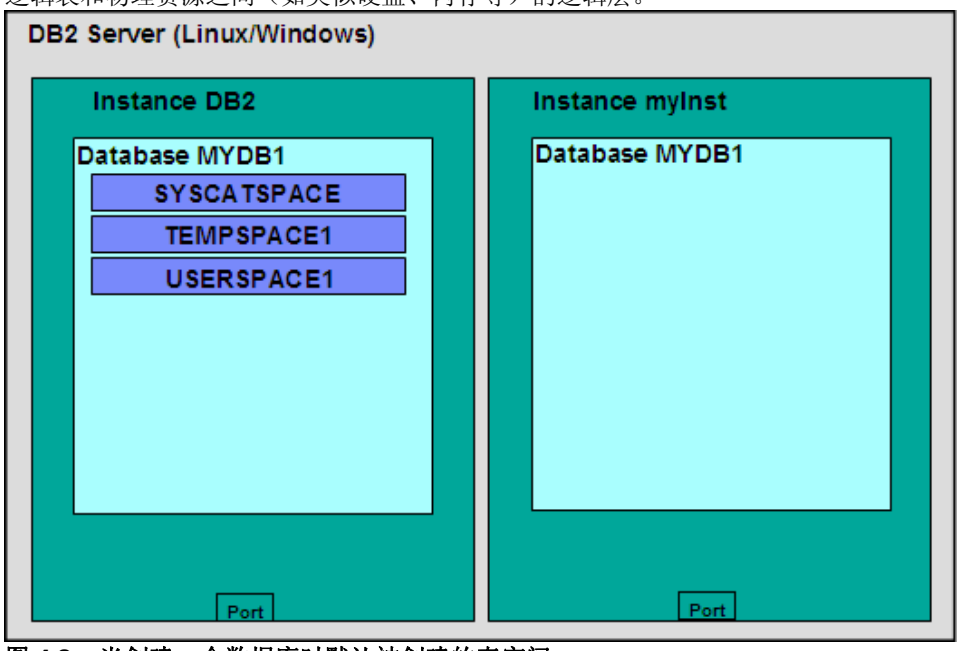


图 4.8 – 当创建一个数据库时默认被创建的表空间

SYSCATSPACE 表空间包含目录表（Catalog）。这个目录也以数据字典的形式存在于其他关系数据库管理系统中。它包含不可更改和不可删除的系统信息，否则数据库无法正常工作。当 DB2 实现一些类似排序等需要额外空间的操作时，就会用到表空间 TEMPSPACE1。如果创建一个表的时候没有指定空间，系统通常会使用 USERSPACE1 来存储您的数据库表。

您也可以使用 CREATE TABLESPACE 语句创建自己的表空间。图 4.9 显示了一个表空间 MYTBLS1，它创建在 DB2 实例的 MYDB1 数据库中。您在创建表空间时需要指定所用的硬盘和内存（缓冲池）。因此，如果您的某个表使用频繁，那么您可以将此表定位在具有最快的硬盘和最大内存的表空间上。

图 4.9 展示了另两个被默认创建的对象：名为 IBMDEFAULTBP 的缓冲池和日志文件。

缓冲池是数据库使用的高速缓冲存储器。您可以创建多个缓冲池，但是至少应该有一个缓冲池，它的大小与现存的表空间页的大小相同。数据页和页的大小将在第六章 DB2 体系结构中详细介绍。

日志文件用于恢复操作。数据库运行过程中，不仅仅是数据库信息被存储在硬盘上，而且日志文件会存储所有针对数据的操作。可以把日志文件看成是一个在“autosave”操作产生的临时文件。第十一章(备份和恢复)将对日志加以详细讨论。

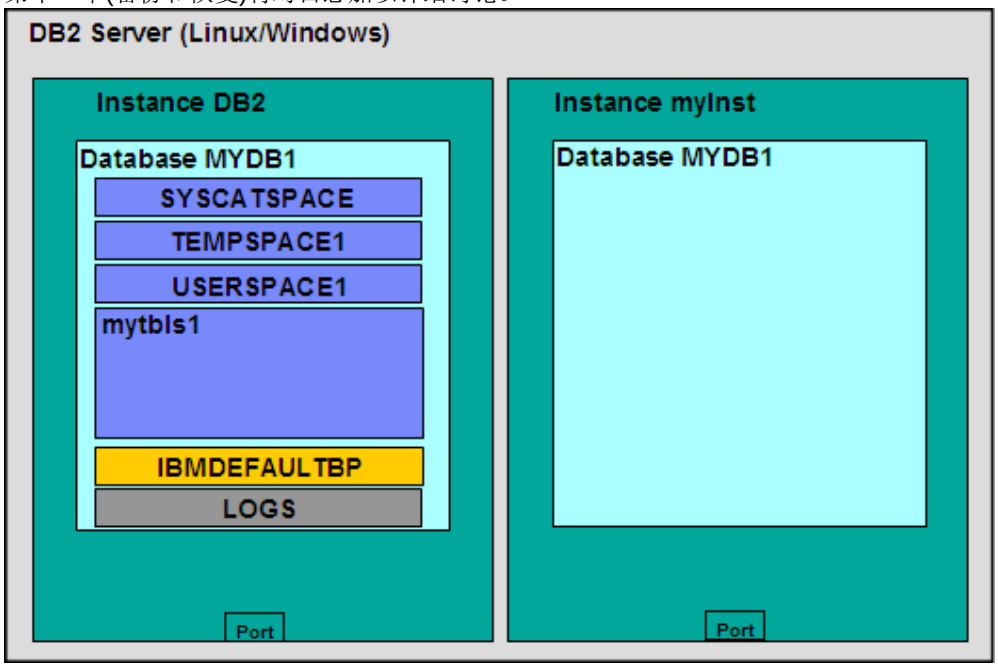


图 4.9 – 默认创建的缓冲池和日志

前面我们讨论了实例间是相互独立的，因此，几个不同实例内可以创建同名数据库。与实例类似，数据库之间也是相对独立的单元，所以，一个数据库内的对象和另一个数据库内的对象毫无关系。图 4.10 展示了存在于实例 DB2 的两个数据库 MYDB1 和 SAMPLE 中同名为“mytbls1”的表空间。注意由于受图片空间大小限制图 4.10 并没有显示数据库 SAMPLE 中其他被默认创建的对象。

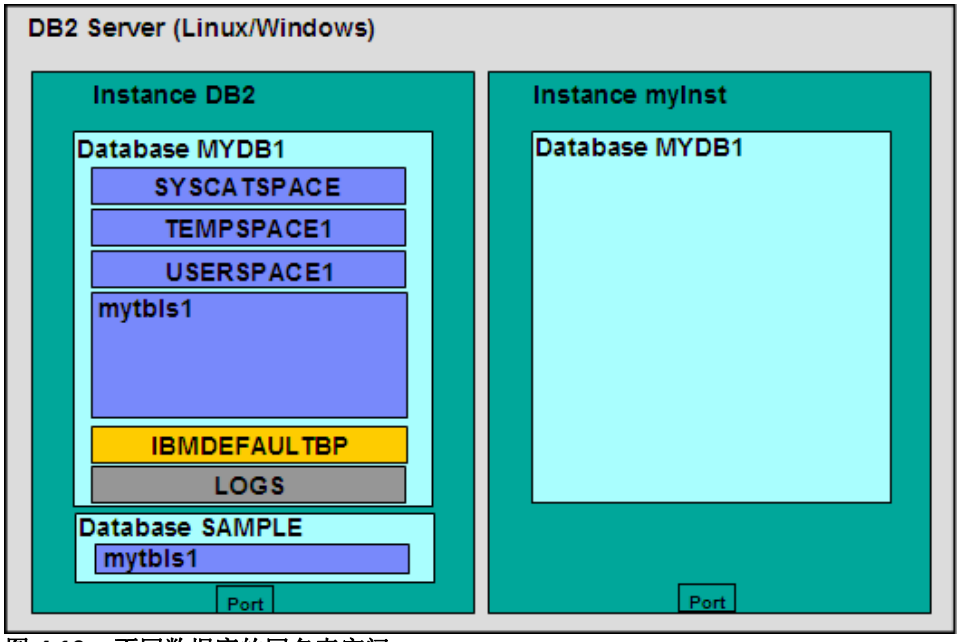


图 4.10 – 不同数据库的同名表空间

当您创建了表空间后，就可以在表空间中创建其它数据库对象，例如数据表，视图和索引。如图 4.11 所示。

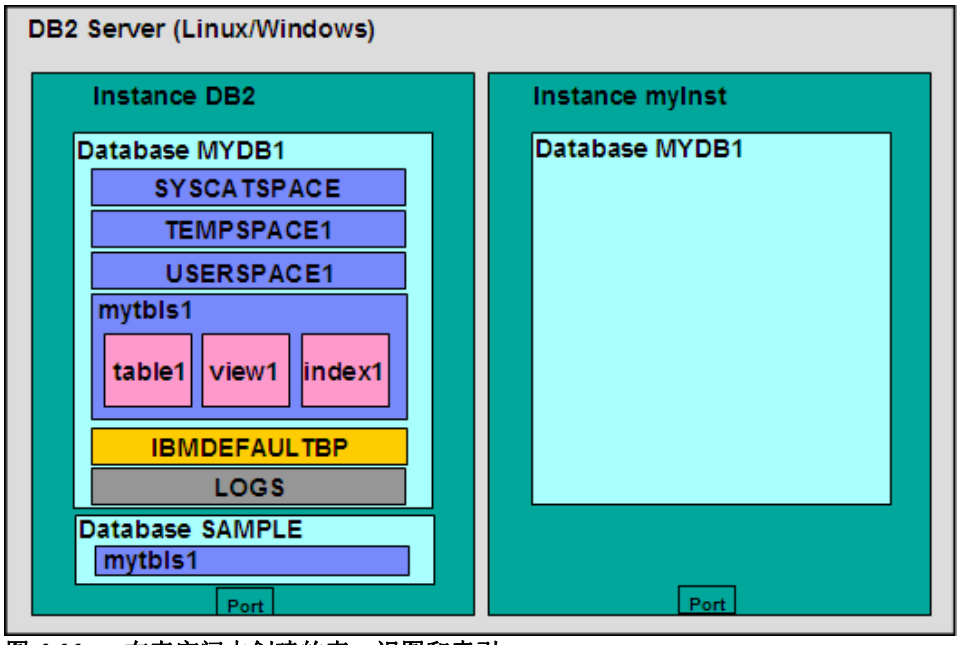


图 4.11 – 在表空间中创建的表，视图和索引

实验 #2 - 创建一个新的数据库

实验目标

本实验中您会在控制中心用创建数据库向导（Create Database wizard）创建一个新的数据库。

实验过程

1. 在控制中心左侧对象树状面板中，右键点击 *All Databases* 文件夹，选择 *Create Database* 项，然后选择自动维护 *With Automatic Maintenance* 项。运行“创建数据库向导 Create Database Wizard”。

2. 在向导的 *Name* 页面上指定数据库名和存储位置。可以使用如下值：

   数据库名： EXPRESS
   默认驱动器（windows）：C
   默认路径（Linux）： /home/db2inst1
   别名： 如果置空则默认规定为 EXPRESS
   备注： 此项可选且可为空

   然后点 *Next* 按钮进入到下一页。

3. 在指定数据存储 *Specify where to store your data* 页面，不要做任何改动，直接点 *Next*。

4. 在选择维护策略 *Select your maintenance strategy* 页面，保留默认（*Yes， I can specify an offline …*），然后点 *Next*。

5. 在向导的 *Timing* 页制定离线维护时间窗口期。确保每星期有两个小时或更长时间 DB2 可以进行自动维护任务，以确保数据库的健康运行。此实验中将窗口期设置为周一至周四每天早上 1：00am 起的 6 小时。然后点 *Next* 按钮继续。

6. 在向导的 *Mail Server* 页面配置通知。如果 DB2 监测到问题或者异常情况，它可以自动发送邮件或者呼叫您。如果想配置此项，则需为 DB2 提供可用的 SMTP 服务器。在本实验中，我们没有 SMTP 服务器，所以此项置空并单击 *Next*。

7. 在向导的 *Summary* 页面复查所有的已选项。点击 *Finish* 按钮开始数据库创建进程。数据库创建通常要花费几分钟时间，这期间可以看到执行的进度条。

4.1 DB2 配置

使用配置向导工具(Configuration Advisor Tool)可以设置 DB2 参数。在控制中心右键单击数据库并选择“Configuration Advisor”，根据您对系统资源和工作载荷的描述，配置向导会提供一个 DB2 推荐参数列表，您可以阅读它们来获得更对关于 DB2 配置的细节信息，也可直接使用配置向导提供的数值。

一个 DB2 服务器可以在四个不同层面上加以配置：

- 环境变量
- 数据库管理器配置文件(dbm cfg)
- 数据库配置文件(db cfg)
- DB2 概要文件注册表

如图 4.12 所示。注意图中每个方框所处的不同位置。例如，环境变量是在服务器的操作系统层设置的，数据库管理器配置文件变量是在实例层设置的。数据库配置变量是在数据库层设置的，DB2 概要文件注册表则可以在操作系统层或实例层设置。

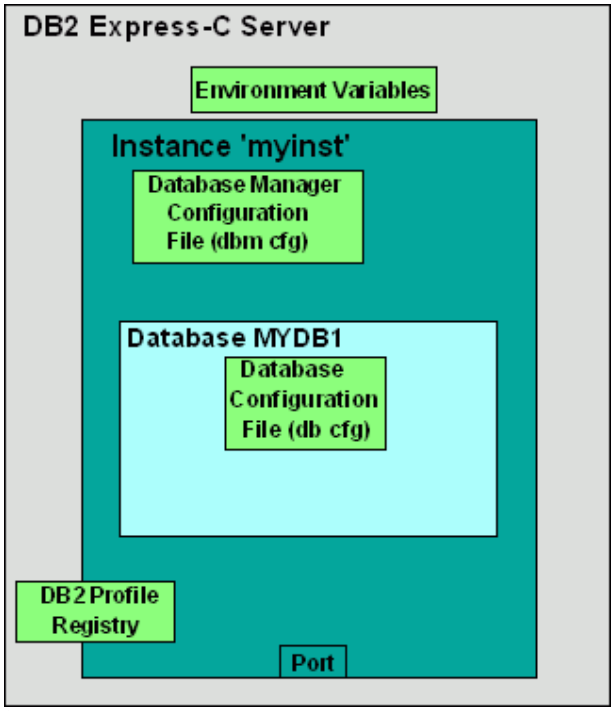


图 4.12 – DB2 配置

4.1.1 环境变量

环境变量是在操作系统层设置的变量。DB2INSTANCE 是其中一个主要的环境变量。这个变量显示了当前活动的实例——即您的 DB2 命令对之执行操作的实例。例如，在命令窗口中设置一个活动的实例“myinst”，您可以运行如下操作系统命令：set db2instance=myinst

4.1.2 数据库管理器配置文件(dbm cfg)

数据库管理器配置文件(dbm cfg)包含一些参数，这些参数影响对应的实例和及其数据库。您可以通过命令行或者 DB2 控制中心查看和修改数据库管理器配置文件。

想要从控制中心启动 DBM CFG，可以在控制中心实例（instance）文件夹内选择实例对象，然后右键单击，在弹出菜单上选择配置参数（Configure Parameters）。如图 4.13 所示。

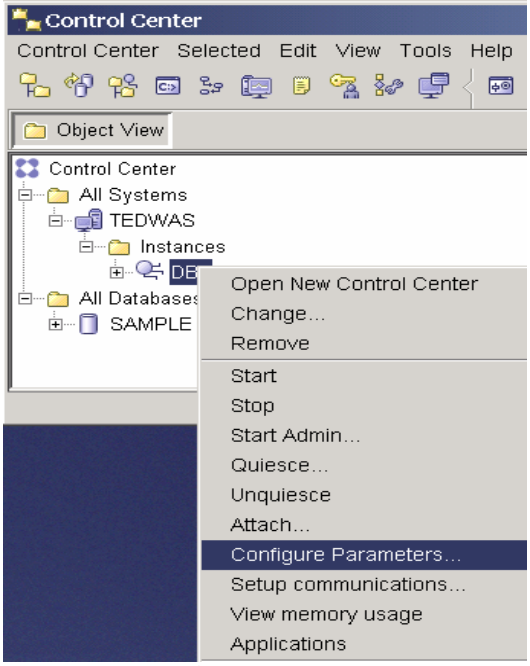


图 4.13 – 在控制中心配置 dbm cfg

在选择配置参数（Configure Parameters）之后，屏幕会显示 dbm cfg 参数列表，如图 4.14 所示。

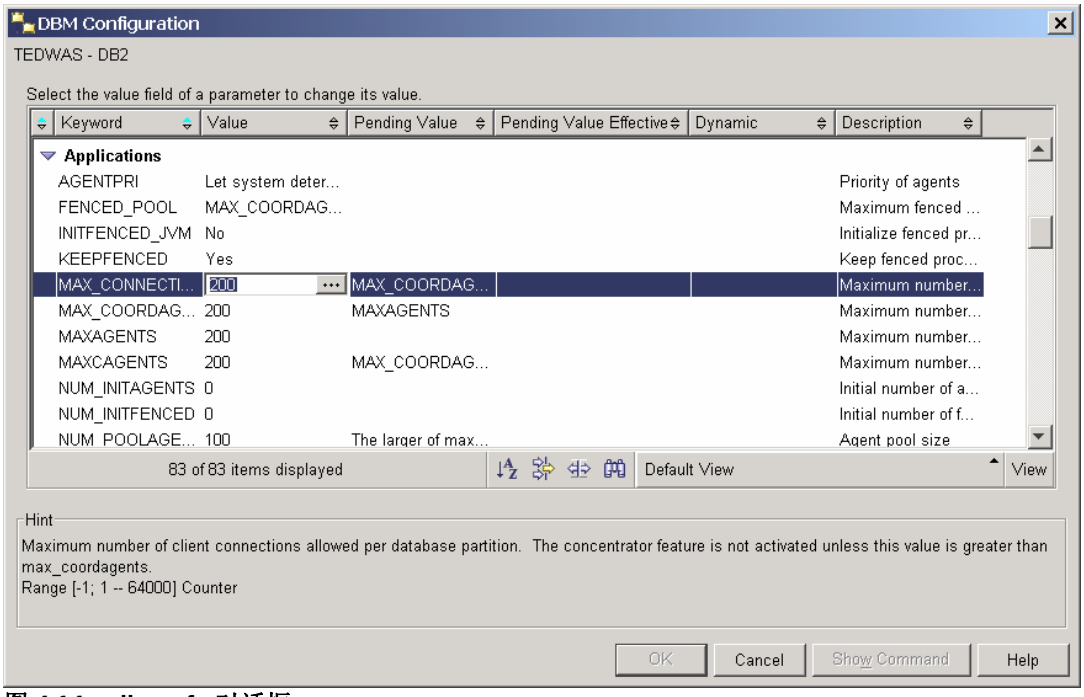


图 4.14 – dbm cfg 对话框

很多参数都是动态的，立即生效；但是，有一些参数的修改需要重启实例。可以在命令行使用 db2stop 和 db2start 命令来实现。

在停止一个实例之前，必须断开所有应用与实例的连接。如果您希望强制停止实例，可以使用 db2stop force 命令。

还可以通过控制中心来停止和启动实例——单击实例对象选择 Stop 或 Start 即可。

表 4.3 显示了一些用命令行管理 dbm cfg 的命令。

命令	描述
db2 get dbm cfg	重新得到 dbm cfg 的信息
db2 update dbm cfg using <parameter_name> <value>	更新 dbm cfg 的参数信息

表 4.3 – 操作 dbm cfg 的命令

4.1.3 数据库配置文件（db cfg）

数据库配置文件（db cfg）包含影响对应数据库的参数。数据库配置文件也可以通过命令行或者控制中心来查看或者修改。

从控制中心启动 DB CFG，可以在数据库（database）文件夹中选择数据库对象，并右键单击弹出菜单，在菜单中选择配置参数（Configure Parameters）。如图 4.15 所示。

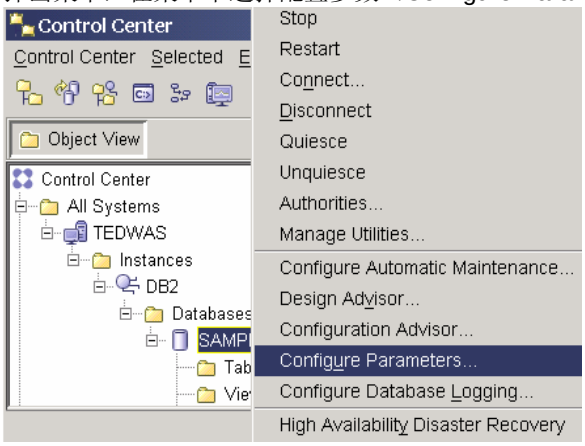


图 4.15 – 通过控制中心配置 db cfg。

选择配置参数后（Configure Parameters），屏幕会显示一个 db cfg 参数的清单。如图 4.16 所示。

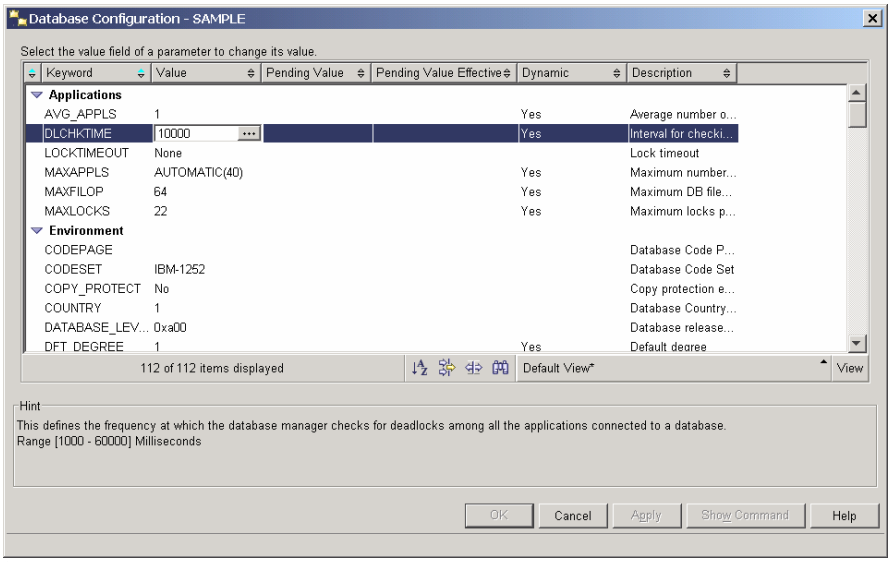


图 4.16 – db cfg

表 4.4 包含了命令行中使用的管理 db cfg 的命令。

命令	描述
get db cfg for <database_name>	重新得到指定数据库的 db cfg 信息
update db cfg for <database_name> using <parameter_name> <value>	更新 db cfg 参数的值

图 4.4 – 操作 db cfg 的命令

4.1.4 DB2 概要文件注册表

DB2 概要文件注册表包含了与平台相关的全局（影响所有实例）或者实力层次（只影响某个实例）的参数。

表 4.5 显示了操作 DB2 概要文件注册表的一些命令。

命令	描述
db2set –all	列表显示当前设置的所有 DB2 概要文件注册表变量
db2set –lr	列表显示所有 DB2 概要文件注册表变量
db2set <parameter>=<value>	把一个参数设置为指定值

表 4.5 - 操作 DB2 文档注册的一些命令

表 4.6 展示一些最常用的 DB2 注册变量

Registry Variable	Description
DB2COMM	指定数据库管理器启动后的通讯管理器。
DB2_EXTSECURITY	Windows 操作系统上，通过锁住 DB2 系统文件来阻止无权限用户对 DB2 的访问
DB2_COPY_NAME	存储当前使用的 DB2 副本的名称。

	要转换到不同的 DB2 副本，请运行 installpath\bin\db2envars.bat 命令进行转换。本变量不能被用于这种转换副本的目的。

表 4.6 – 经常使用的 DB2 文档注册变量

例如，若想用 TCPIP 协议实现通讯，可以将 DB2COMM 注册变量用命令设置为 TCPIP，如下所示：

```
db2set db2comm=tcpip
```

4.2 DB2 管理服务器

DB2 管理服务器（DAS）是一个运行在 DB2 服务器上的守护进程，此进程允许远程您连接并通过图形化管理器管理 DB2 服务器。每一台计算机只有一个 DAS，如图 4.16 所示。

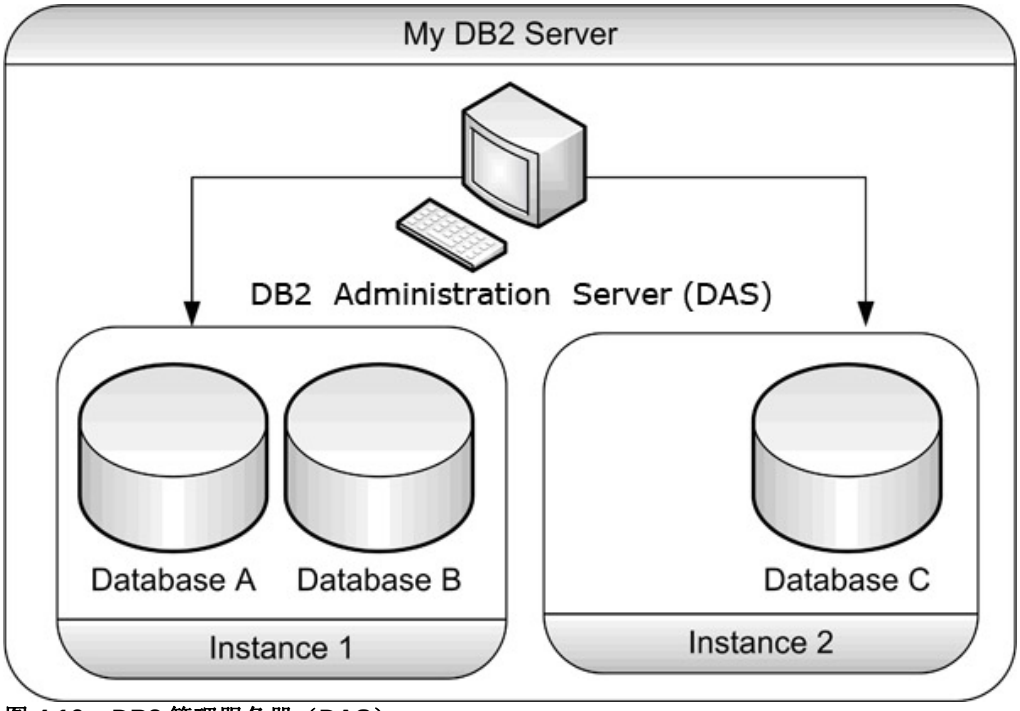


图 4.16 – DB2 管理服务器（DAS）

实验 #3 – 实例、数据库和配置管理

实验目的

在本实验中，您将在 Windows 系统下创建一个新的实例和一个新的数据库，然后更改 DB2 服务器的配置参数。您可以通过控制中心或者命令窗口来实现这些操作。在这里我们以使用命令窗口为例。

实验过程

1. 从命令窗口创建一个名为 *newinst* 新实例
   db2icrt newinst

2. 在这个新实例 *newinst* 中，使用默认值创建一个名为 *newdb* 的数据库
   set db2instance=newinst
   db2start
   db2 create database newdb

3. 列出服务器上所有的实例
   db2ilist

4. 转换到 DB2 实例，并确认转换成功
   set db2instance=db2
   db2 get instance

5. 把 dbm cfg 的 FEDERATED 参数值由 NO 改为 YES 并验证修改的结果。
   db2 update dbm cfg using FEDERATED YES
   db2 force applications all
   db2 terminate
   db2stop
   db2start
   db2 get dbm cfg

6. 使用登录操作系统的用户名和密码连接 SAMPLE 数据库
   db2 connect to sample user <userID> using <psw>

7. 查看当前实例上有多少应用程序在运行
   db2 list applications show detail

8. 打开另一个 DB2 命令窗口，不指明用户名和密码连接到 SAMPLE 数据库。然后查看当前有多少应用连接到该实例上。
   db2 connect to sample
   db2 list applications

9. 强制关闭一个 DB2 命令窗口
   db2 force <application> (<application> 填入程序“db2bp.exe”的句柄，该值由 db2 list applications 命令获得)

10. 删除实例 *newinst*
    db2idrop newinst

11. 删除并重新创建 DAS，然后启动 DAS。
    db2admin stop
    db2admin drop

```
db2admin create
db2admin start
```

12. 在您的实例中设置 DB2 注册变量 DB2COMM，使其值为 tcpip，npipe。

```
db2set db2comm=tcpip，npipe
db2stop
 db2start
```

13. 置空 DB2COMM 注册变量

```
db2set db2comm=
db2stop
db2start
```

14. 检查当前 db cfg 的 LOGSECOND 参数，然后将其值设置成 5 并验证新值

```
db2 connect to sample
db2 get db cfg
db2 update db cfg using LOGSECOND 5
db2 get db cf
```

5

第 5 章 – DB2 工具

本章我们会讨论在 DB2 中会使用到的工具，图 5.1 中的红色椭圆标识的部分是本章关注的内容。

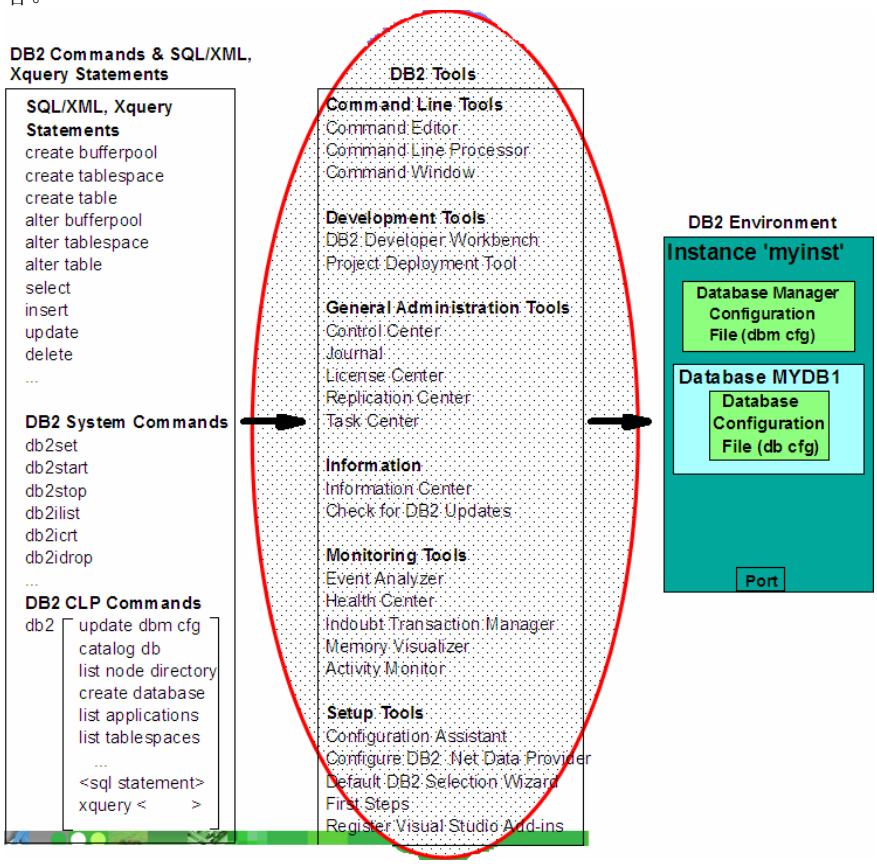


Figure 5.1 – The DB2 big picture: DB2 tools
图 5.1 – DB2 概览图：DB2 工具

注意:
更多关于 DB2 工具和脚本的信息，请参见以下视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:4202
http://www.channeldb2.com/video/video/show?id=807741:Video:4182

图 5.2 列出了 IBM DB2 在开始菜单快捷方式能找到的所有工具。无论在 Linux 还是 Windows 操作系统，它们大部分都是相同的。

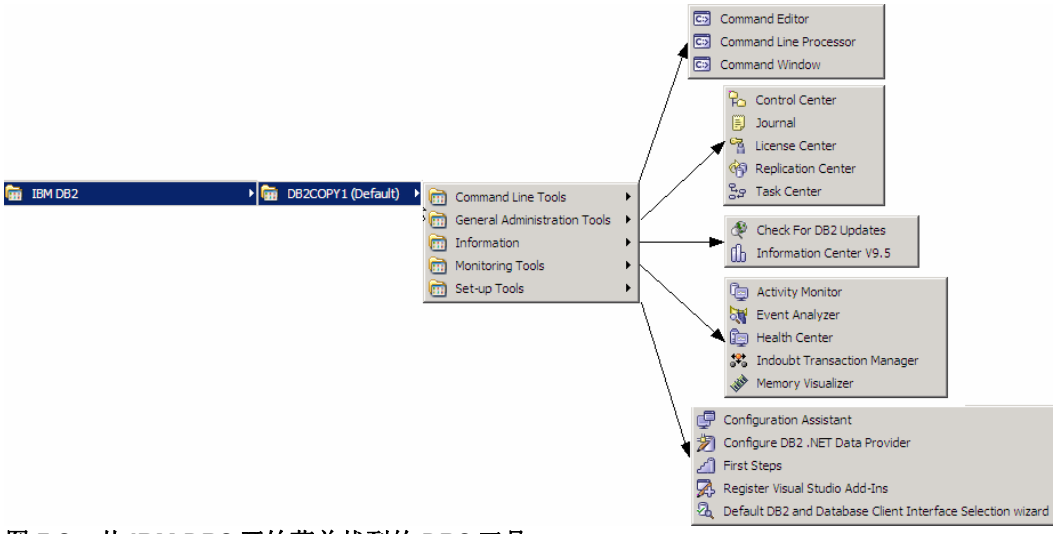


图 5.2 – 从 IBM DB2 开始菜单找到的 DB2 工具

表 5.1 提供了一系列在 Linux 或 Windows 操作系统上应用最频繁工具所对应的快捷命令.

工具名称	命令
Command Editor	db2ce
Command Line processor	db2
Command Window (Only on Windows platforms)	db2cmd
Control Center	db2cc
Task Center	db2tc
Health Center	db2hc
Configuration Assistant	db2ca
First Steps	db2fs

表 5.1 – DB2 工具对应的快捷命令

5.1 控制中心（Control Center）

DB2 中，主要的数据库管理工具是控制中心(Control Center)，如图 5.3 所示。

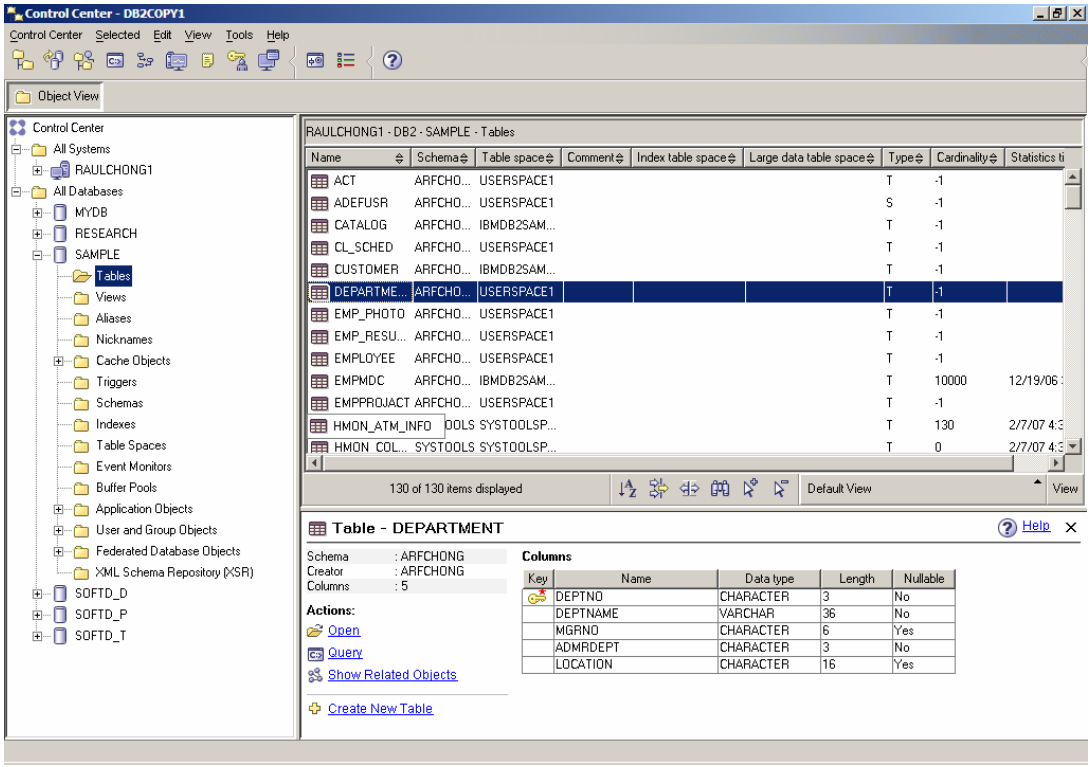


图 5.3 – DB2 控制中心(Control Center)

控制中心(Control Center)是一个集中式的管理工具，使用它可以实现如下功能：

- 查看您的系统、实例、数据库、数据库对象
- 创建、修改、管理数据库和数据库对象
- 启动其它的图形化 DB2 工具

控制中心左边的面板提供了系统中数据库对象的层次表示，用文件夹表示数据表或者视等。当您双击一个文件夹时，例如双击 Table 文件夹，如图 5.3 所示，控制中心右上方的窗口中会列出所含的相关对象，在本例中显示的在 SAMPLE 数据库中所有的表。如果您在右上的窗口中选中某一个数据表，在右下的窗口会显示出关于这个数据表更多的详细信息。

在左边层次树的不同文件夹/对象上单击右键，会弹出对应于这个文件夹/对象的菜单。例如，右键单击一个数据库实例，然后选择“Configure parameters”，就可以弹出窗口让您查看或者设置数据库管理配置文件。相似的，当您右键单击一个数据库然后选择“Configure parameters”，就可以弹出窗口让您查看或者设置该数据库的配置文件。DB2 环境和设置参数在第五章——DB2 环境中详细阐述。

当您第一次启动控制中心(Control Center)时，您会要求选择控制中心的视图，不同的视图会显示不同的数据库类型和对象。图 5.4 展示了控制中心视图对话框。

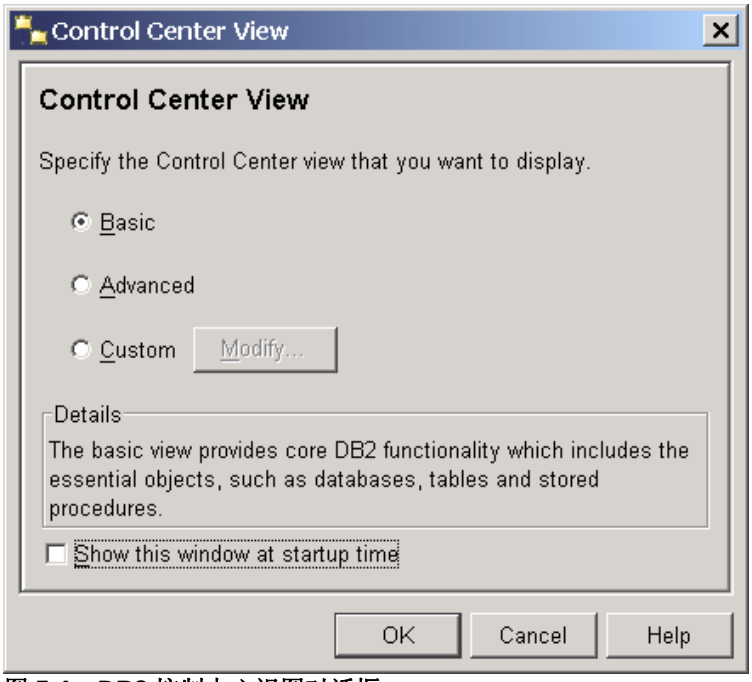


图 5.4 – DB2 控制中心视图对话框

基本视图（basic）提供了 DB2 的主要核心功能。

高级视图（advanced）提供了关于 DB2 更多的选项和特性

自定义视图（custom）可以让您自己定义控制中心所显示的选项、对象和数据库特性。

此后，您可以在控制中心的“工具”菜单中选择“自定义控制中心 Customize Control Center”选项来弹出 DB2 控制中心视图对话框，如图 5.5 所示。

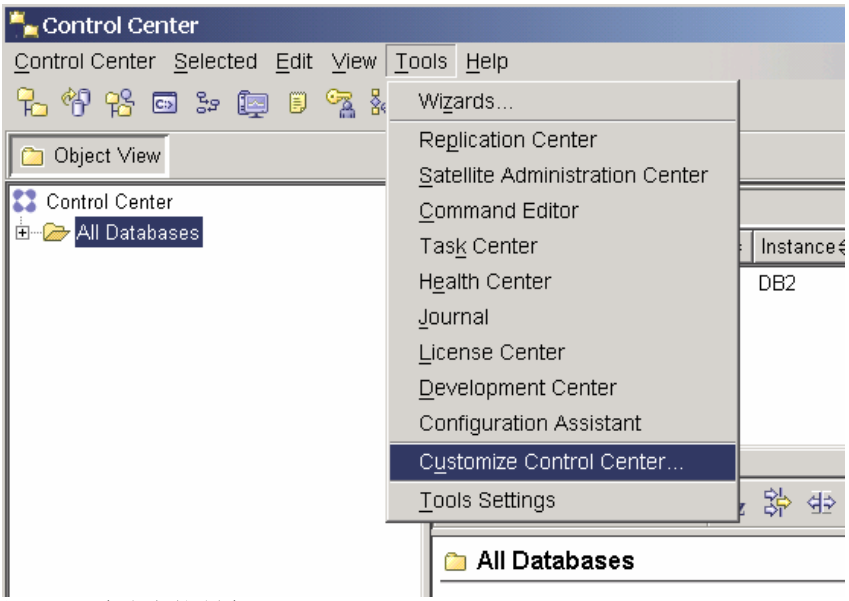


图 5.5 – 自定义控制中心

运行控制中心（Control Center）

有很多种方法来启动控制中心 Control Center

- 在 Windows 开始菜单中找到控制中心的快捷方式来启动。

- 在命令提示符中输入 db2cc 来启动。
- 在任意的 DB2 GUI 工具中点击控制中心的图标 来启动。
- 从 Windows 的系统任务栏的 DB2 图标中启动，如图 5.6 所示，右键单击 DB2 的绿色图标，然后在弹出的菜单中选择 DB2 Control Center 选项。

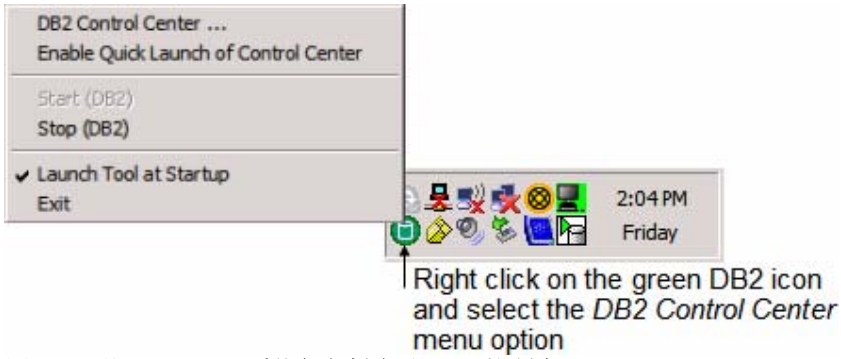


图 5.6 – 从 Windows 系统任务栏启动 DB2 控制中心（Control Center）

5.2 命令编辑器（Command Editor）

您可以使用 DB2 命令编辑器来执行 DB2 命令、SQL 和 XQuery 语句、分析语句的执行流程、查看或者更新查询结果集。

图 5.7 展示了命令编辑器 Command Editor 并进行了解释。

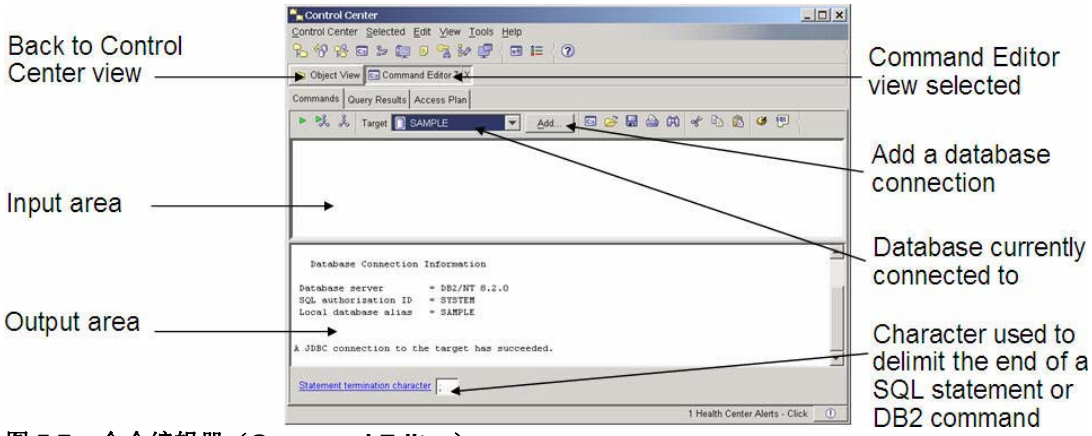


图 5.7 – 命令编辑器（Command Editor）

在输入框中，您可以输入若干条语句，每个语句后面使用一个终结字符结束。当您点击“执行 execute”按钮（如图 5.8 所示），语句会一条一条按顺序执行。当您高亮了某些语句时，只有选定高亮的语句会被执行。在任何 SQL 语句执行前必须连接到数据库，当然，也可以把数据库的连接语句放到要执行的 SQL 语句中。

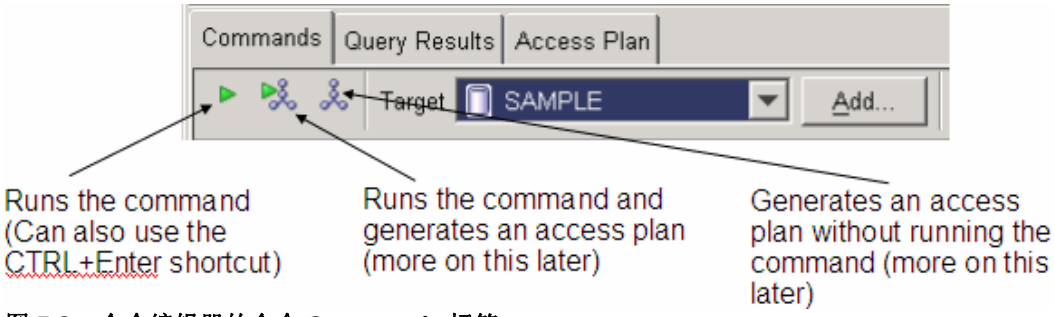


图 5.8 – 命令编辑器的命令 Commands 标签

运行命令编辑器（Command Editor）

您可以通过几种方法类运行命令编辑器:

1、从 windows 开始菜单启动：Start -> Programs -> IBM DB2 -> DB2COPY1 (Default) -> Command Line Tools -> Command Editor

2、在命令提示符中输入 db2ce

3、从控制中心的“工具 tools”菜单中运行

4、将命令编辑器嵌入到控制中心

- 在控制中心的对象树窗口中右键单击 SAMPLE 数据库，然后选择 Query 选项。
- 在任何时候，如果能够被查询的对象（如数据库、表、等等）被选中，您都可以单击在控制中心的对象明细 Object Detail 窗口中的 *Query* 链接来运行命令编辑器。

5、在控制中心的工具栏中单击命令编辑器的图标，如图 5.9 所示，可以启动命令编辑器。

图 5.9 – 控制中心的命令编辑器图标

添加一个数据库的连接

为了给连接到一个数据库，单击命令编辑器的 Add 按钮（请查看图 5.7），会出现图 5.10 的对话框。

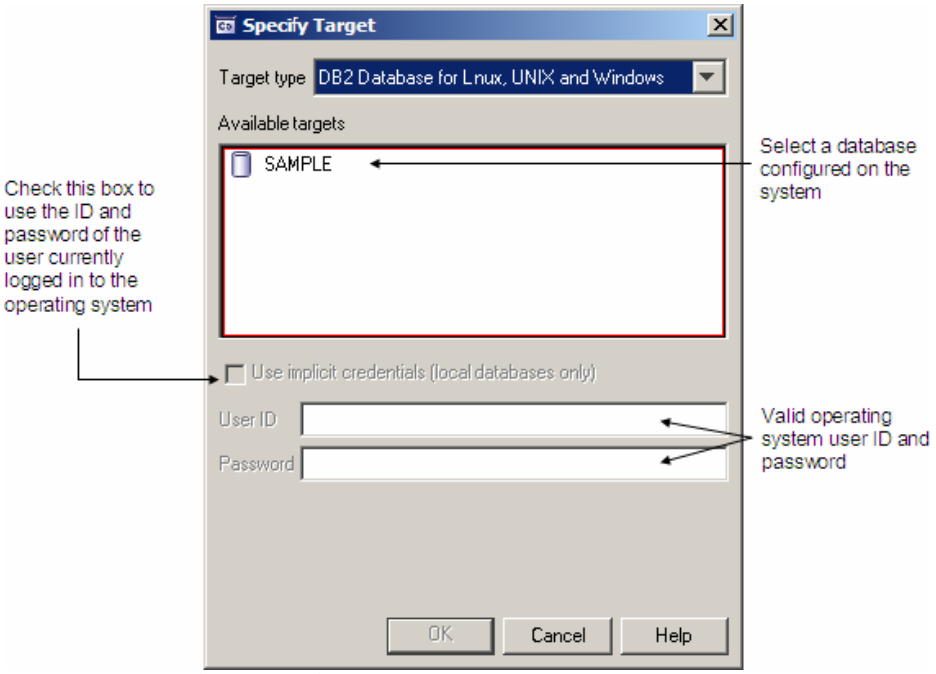


图 5.10 – 添加一个数据库连接

5.3 SQL 帮助向导（SQL Assist Wizard ）

如果您不熟悉 SQL 语言，您可以使用 SQL 助手或者向导来生成所需要的 SQL 代码。SQL 帮助向导（SQL Assist Wizard）可以在命令编辑器中启动，如图 5.11 所示，您可以通过单击 Commands 标签中的工具栏最后一个有 SQL 字样的按钮来启动它。

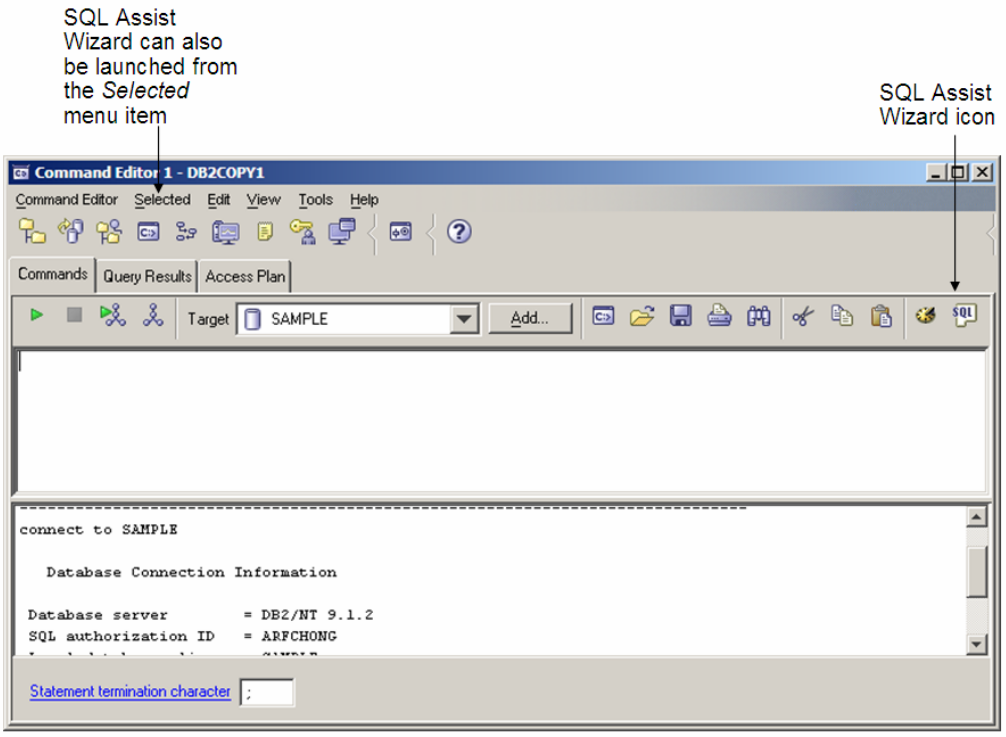


图 5.11 – 运行 SQL 帮助向导（SQL Assist Wizard）

图 5.12 展示了 SQL 帮助向导（SQL Assist Wizard），它是非常简单易用的。首先指明您需要帮助的 SQL 语句类型（SELECT， INSERT， UPDATE， DELETE）。根据您选的语句，会出现不同的选项，在向导对话框的底部窗口中，您可以看到根据您的选项而产生的 SQL 语句。

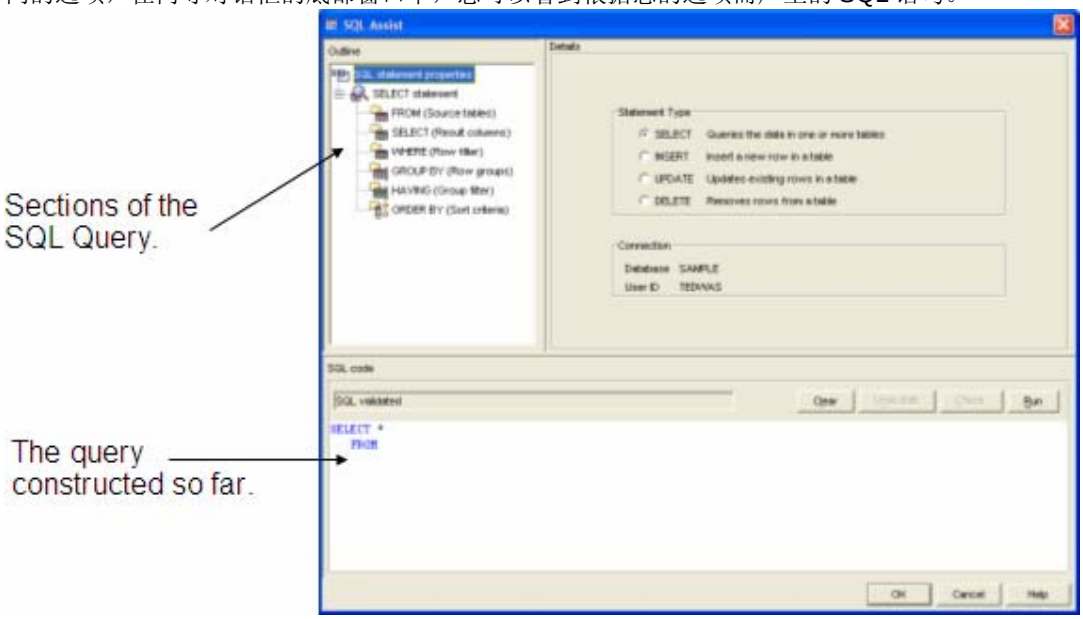


图 5.12 – SQL 帮助向导（SQL Assist wizard）

5.4 显示 SQL 按钮

在 DB2 中，大多数的 GUI 工具或者向导允许您查看当前所产生的 SQL 语句。为了看到 SQL 语句，在当前使用的工具中单击显示 SQL（Show SQL）按钮，如图 5.13 和 5.14 所示。

图 5.13 – 显示 SQL（Show SQL）按钮

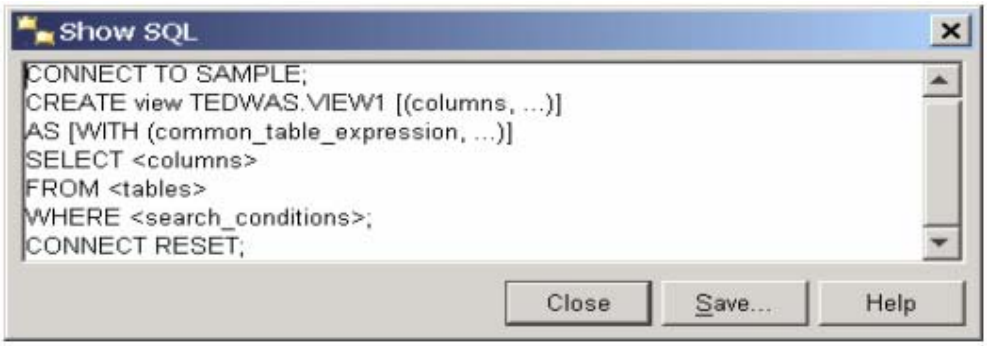


图 5.14 – 显示 SQL（Show SQL）的输出

查看 SQL 语句和命令的功能使您能够非常地方便学习 SQL 的语法，您也可以将这些命令或者语句保存下来，以便将来使用，也可以使用所生成的这些命令和语句来创建自己的脚本。

实验 #4 使用脚本填充 EXPRESS 数据库

实验目标

在本实验，您会使用命令编辑器和两个脚本来填充 EXPRESS 数据库。

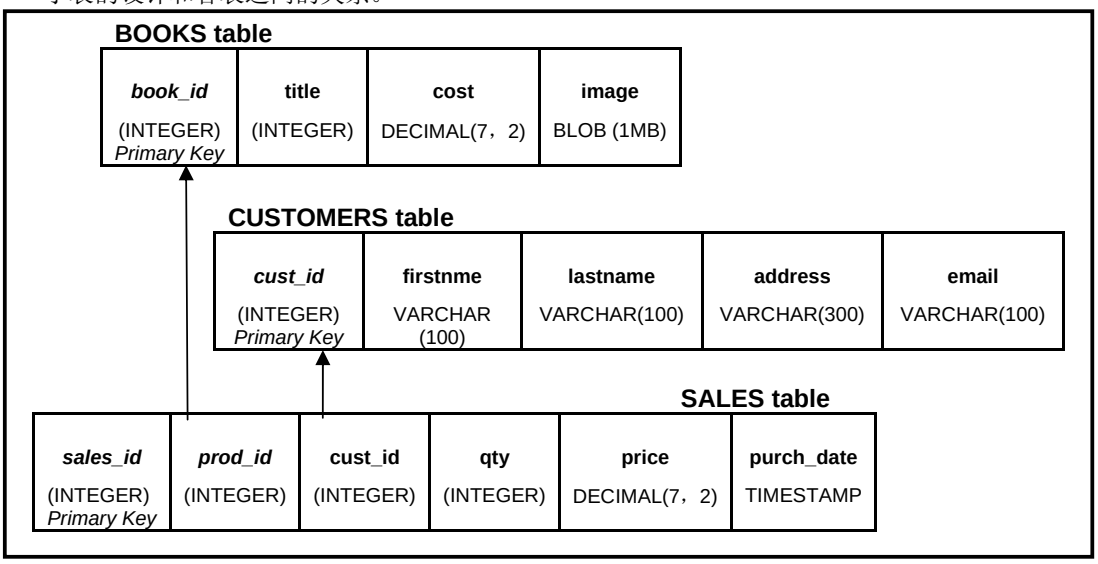
实验过程

1、在实验#2 中您创建了 EXPRESS 数据库，现在您需要用一些数据表和数据来填充这个数据库。为了您的方便，我们预先建立好了两个脚本（`quicklab4.db2` 和 `quicklab4.dat`）供您在本实验使用。quicklab4.db2 包含了创建数据表的命令，所以必须先运行这个脚本。quicklab4.dat 脚本包含了将数据插入数据表的语句。在 quicklabs.zip（随书发布）文件中可以找到这两个脚本。使用命令编辑器可以运行这两个脚本。在运行这两段脚本前，请确保在工具栏的下拉列表中选中那个新建的数据库 EXPRESS。如果新建的数据库没有出现在下拉列表中，单击 Add 按钮来添加一个到数据库的连接。

2、在命令编辑器中单击 *Selected → Open*，找到 *quicklab4.db2* 并单击 OK 按钮。这时，在命令编辑器的输入区域会显示 *quicklab4.db2* 文件的所有内容。单击运行（Run）按钮来执行这个脚本。检查在脚本运行过程中是否有出现什么错误。

3、按照步骤 2 来执行 `quicklab4.dat` 文件。

您新建的这个数据库是用于一个非常简单的网上书店，BOOKS 表保存了书店中所有库存书目信息。CUSTOMERS 表保存了每一个顾客的信息。SALES 表保存了销售的数据。下面的图表显示了表的设计和各表之间的关系。



5.5 脚本

将常用的一连串 DB2 命令或者 SQL 语句编写成脚本是非常方便和有用的。例如，一个数据库管理员每天会运行一个脚本来检查重要数据表的行数。

常用的有两种脚本类型：

1、SQL 脚本

2、操作系统（shell）脚本

5.5.1 SQL 脚本

SQL 脚本包括查询语句和数据库命令。这类脚本比较简单易懂，并且具有平台独立性。但是 SQL 脚本不支持变量和参数。

例如，下面是保存在 script1.db2 文件中的命令。

```
CONNECT TO EXPRESS;
CREATE TABLE user1.mytable
          ( col1 INTEGER NOT NULL,
            col2 VARCHAR(40),
            col3 DECIMAL(9, 2));
SELECT * FROM user1.mytable FETCH FIRST 10 ROWS ONLY;
COMMIT;
```

File script1.db2

在上面的脚本中，所有的语句都是 SQL 语句，每条语句都使用分号来隔开，文件扩展名不一定是 db2，任何扩展名都是可以的。

执行 SQL 脚本

可以在命令编辑器或者 Windows 的 DB2 命令窗口或者 Linux 的 shell 中执行 SQL 脚本。在 windows 的 DB2 命令窗口或者 linux 的 shell 中执行 SQL 语句，您可以使用下面的命令：

```
db2 -t -v -f script1.db2 -z script1.log
```

或者：

```
db2 –tvf script1.db2 –z script1.log
```

在上面的命令中，

-t 表示语句使用默认的语句终结符——分号

-v 表示使用冗长模式，这样 DB2 会显示每一条正在执行命令的信息。

-f 表示其后就是脚本文件

-z 表示其后的信息记录文件用于记录屏幕的输出，方便以后的分析（这是可选的，但我们建议使用该选项）

当使用了-t 选项而没有标明语句终结符，则分号（；）会默认为语句的终结符。有时可能会出现使用另外的终结符的情况，例如用 SQL PL 编写的的脚本使用其它的符号而不是默认的分号，因为分号在 SQL PL 是用于定义数据库对象过程中的语句结束。

例如，如下名为 functions.db2 的脚本包含了创建一个函数所需要的数据定义语言（DDL），在 SELECT 语句后有一个分号作为该语句的结束。而整个 CREATE FUNCTION 语句，则使用了叹号（！）作为终结符，因为如果我们依然使用分号作为这个 FUNCTION 对象的终结符，会在运行时产生冲突，DB2 将为此返回一个错误。

```
CREATE FUNCTION f1()
```

```
   SELECT …    ;
   …
END!
```

File functions.db2

为了使 DB2 使用其它的语句终结符，使用-d 选项结合-t 来声明其它的终结符如下的-td！所示：

```
db2 –td! –v –f functions.db2 –z functions.log
```

至于其它选项的信息，可以在命令窗口或者 Linux 的 shell 中输入如下的命令来查询：

```
db2 list command options
```

5.5.2 操作系统（shell）脚本

操作系统脚本更加灵活和强大，因为您可以添加其它的程序逻辑。它们依赖于平台，不同的操作系统的脚本可能不相同，但是可以它们支持变量和使用参数。下面是一个 Windows 操作系统的脚本。

```
set DBPATH=c:
set DBNAME=PRODEXPR
set MEMORY=25
db2 CREATE DATABASE %DBNAME% ON %DBPATH% AUTOCONFIGURE USING
      MEM_PERCENT %MEMORY% APPLY DB AND DBM
db2 CONNECT TO %DBNAME% USER %1 USING %2
del schema.log triggers.log app_objects.log
db2 set schema user1
db2 –t –v –f schema.db2 –z schema.log
db2 –td@ -v –f triggers.db2 –z triggers.log
db2 –td@ -v –f functions.db2 –z functions.log
```

File create_database.bat

为了运行这段脚本，您可以在 Windows 的命令行中输入下面的语句。

```
create_database.bat db2admin ibmdb2
```

在 Windows 中使用“bat”作为扩展名，表示这是一个可执行的批处理命令。

在 Linux 中，您需要改变文件的属性，使用 `chmod +x` 命令来为文件添加可执行的属性。添加可执行属性后，您可以像上面那样将它运行。

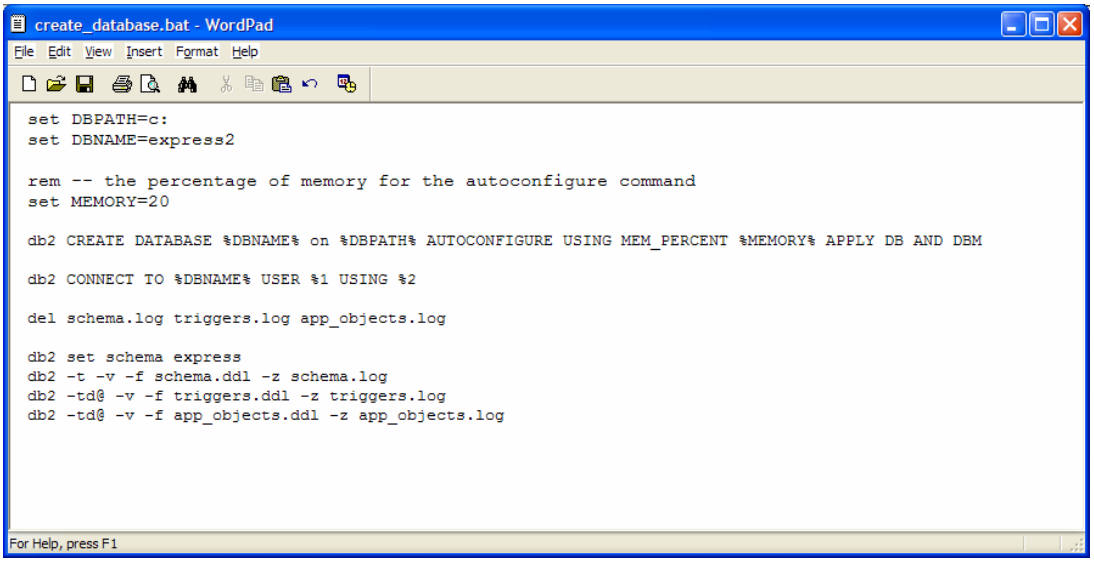
实验 #5 为 EXPRESS 数据库创建一个安装脚本

实验目标

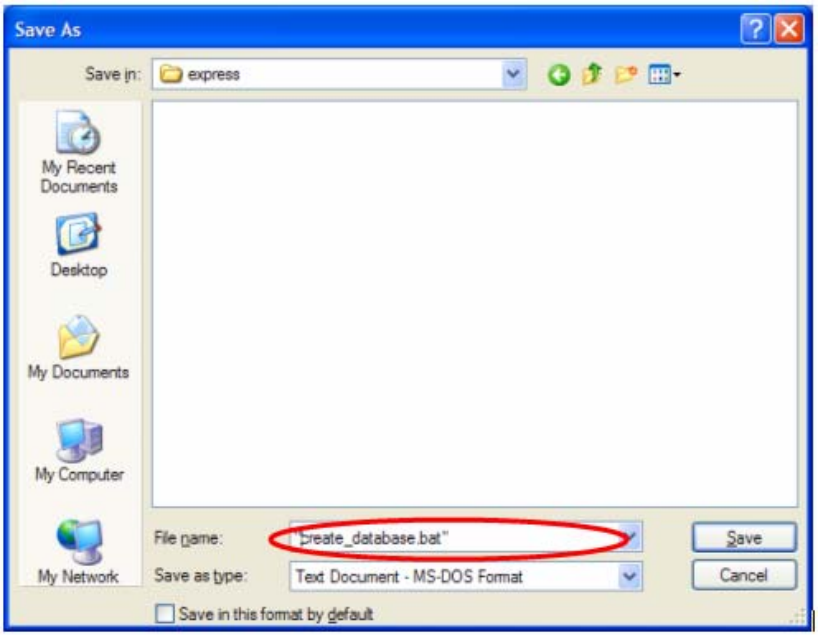
脚本是处理重复工作的强大工具，例如收集数据库统计数据、数据库备份、数据库发布配置等等存在大量重复劳动的工作可以使用脚本来方便的将它们解决。操作系统脚本支持脚本参数，使得它更加灵活。在本实验中，您将会建立一个操作系统脚本来发布和配置 EXPRESS 数据库（将其配置为 EXPRESS2）。这个脚本会调用先前创建的 SQL 脚本来处理数据库对象。本实验的所有脚本和命令都是基于 Windows 平台，如果您想在 Linux 上进行实验，请将脚本和命令进行适当的修改。

实验过程

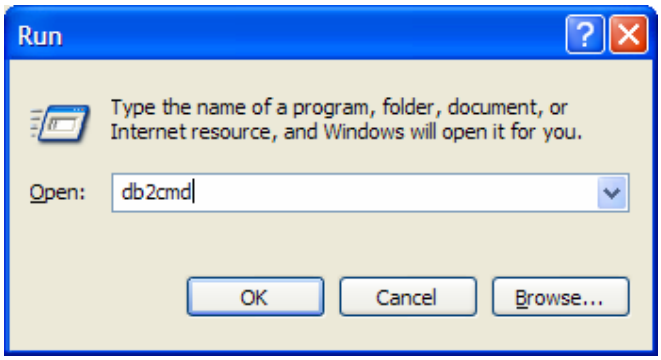
1、打开文本编辑器，如记事本或者 word，然后输入下图所示的内容：



2、将上面所输入的内容命名为 `create_database.bat` 并保存到文件夹中。在 word 的保存对话框中，请选择保存类型为 *MS-DOS* 格式，如果您选择了其它的格式，会在文件中插入一些不可见的字符，这会导致该脚本在运行中出错。同时，使用双引号将文件名括起来，这样 Windows 就不会将.txt 的扩展名添加到脚本文件名后面，如下图所示：

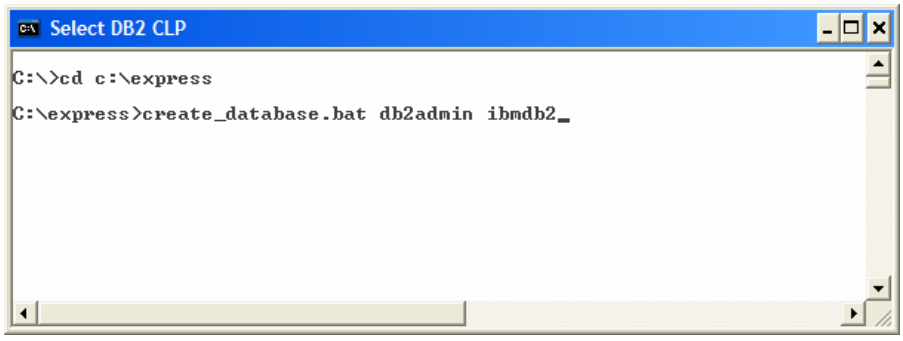


3、为了与 DB2 交互地运行这段脚本，您必须先打开 DB2 命令行环境。您可以从 *Start* > *Program Files* > *IBM DB2* > *DB2COPY1 (default)* > *Command Line Tools* > *Command Window* 打开 DB2 命令行窗口。
您也可以使用 Windows 的“运行”命令（*Start* > *Run*），用 db2cmd 来打开 DB2 命令行窗口。如下图所示：



4、输入下面的命令来执行脚本

```
cd C:\express
create_database.bat db2admin ibmdb2
```



5、请花些时间来理解上面创建的脚本，您能说出每一行的作用是什么吗？

6、尝试回答下面的问题：

A、数据库在哪里建立连接？

B、%1 和%2 是什么意思?

C、下面的语句起什么作用？作用在哪里？为什么要这样做？

```
SET DBPATH=C:
```

D、下面的语句有什么作用?

```
del schema.log， triggers.log， app_objects.log
```

E、如果无参执行上面的脚本会发生什么事情？

F、为什么 SQL 脚本没有包含 CONNECT TO 语句？它们是怎么样连接到数据库的？

5.6 任务中心（Task Center ）

GUI 的任务中心用于建立任务，任务指的是一连串操作（如运行 DB2 命令、操作系统命令或脚本）的集合。在任务成功或者失败时，任务中心会执行相应的动作。例如，有一项任务是在 3：00am 备份一个重要的数据库，如果这个任务执行成功，则向数据库管理员发送一封包含成功执行信息的电子邮件；如果备份失败，任务中心能够呼叫数据库管理员。任务中心如图 5.15 所示。

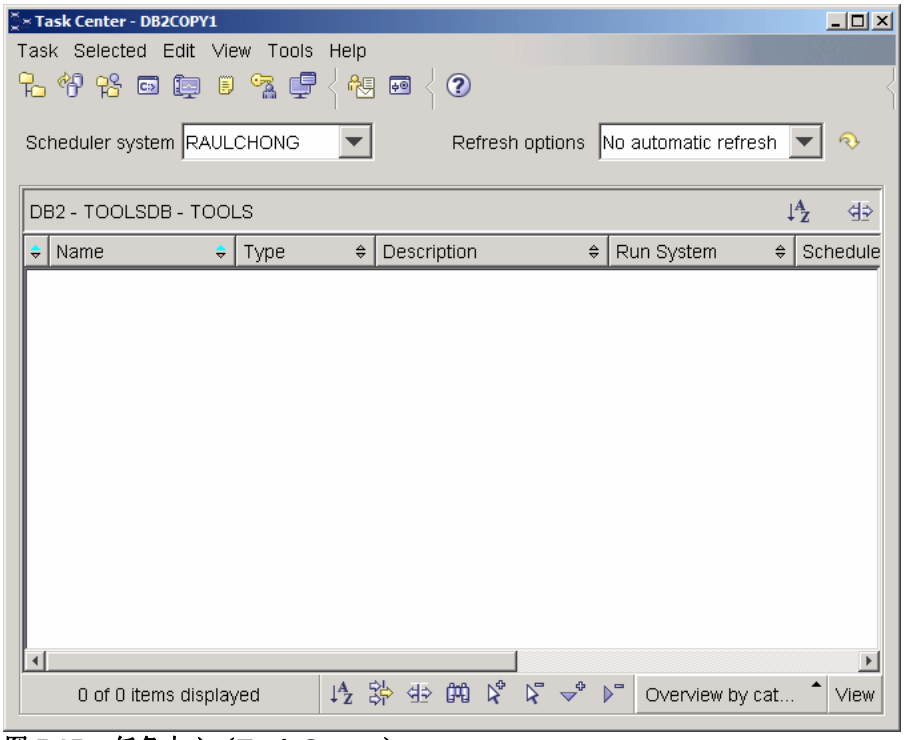


图 5.15 – 任务中心（Task Center）

5.6.1 工具目录（Tools Catalog）数据库

关于任务的所有详细信息都保存在 DB2 中叫 Tools Catalog 的数据库里。在安排任务前，这个数据库必须存在。您可以用下面的命令来创建 Tools Catalog 数据库。

```
CREATE TOOLS CATALOG systools CREATE NEW DATABASE toolsdb
```

在上面的例子中，systools 是数据库中所有数据表的模式名字，toolsdb 是数据库的名字。我们会在第 8 章对模式进行详细讲解。

运行任务中心

您可以在控制中心中启动任务中心。如图 5.16 所示（*Tools* > *Task Center*）。同样，可以从 Windows 的开始菜单中启动任务中心（*Start* > *Programs* > *IBM DB2* > *DB2COPY1* > *General Administration Tools* > *Task Center*）。

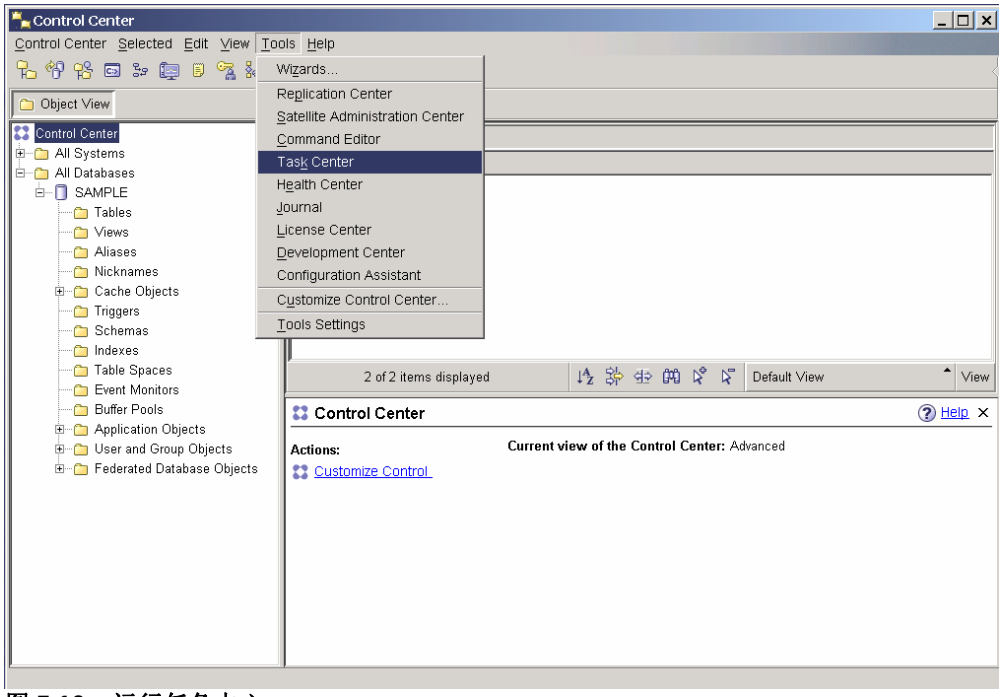


图 5.16 – 运行任务中心

计划任务

可以使用任务中心来计划各种任务的运行安排。各种任务按照它们的时间表来运行。我们建议您自行研究一下任务中心，建立一项任务是非常简单明了的。

5.7 日志（Journal）

DB2 GUI 的日志工具以在线的形式为管理员提供了数据库各种活动的日志。图 5.17 展示了日志工具，表 5.2 展示了您能够从日志工具获得的各种信息。

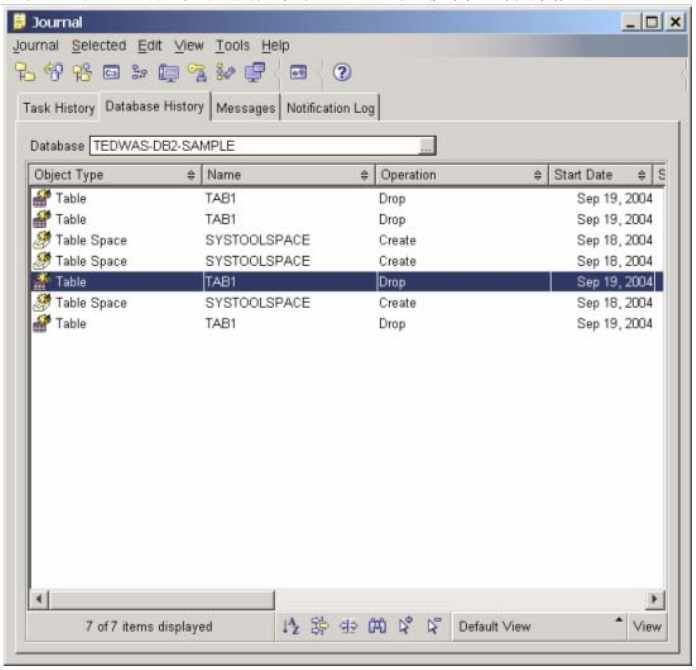


图 5.17 – 日志工具

Type of Information	Description
Task History	所有执行的计划任务和它们的完成情况
Database History	数据库所有活动的记录，如备份、恢复、重组等等
Message	保存 DB2 工具返回的信息，这是非常有用的。如果您想重新执行某项操作并比较新旧的出错信息，或者有意无意关闭了一个对话框而您还来不及看清里面的信息，您可以在这里找到这些信息。
Notification Log	包含系统级别的信息，严重错误信息记录在这里

表 5.2 – 日志工具提供的信息

运行日志工具

您可以在控制中心中启动日志工具。如图 5.18 所示（*Tools > Journal*）。同样，可以从 Windows 的开始菜单中启动日志工具（*Start > Programs > IBM DB2 > DB2COPY1 > General Administration Tools > Journal*）。

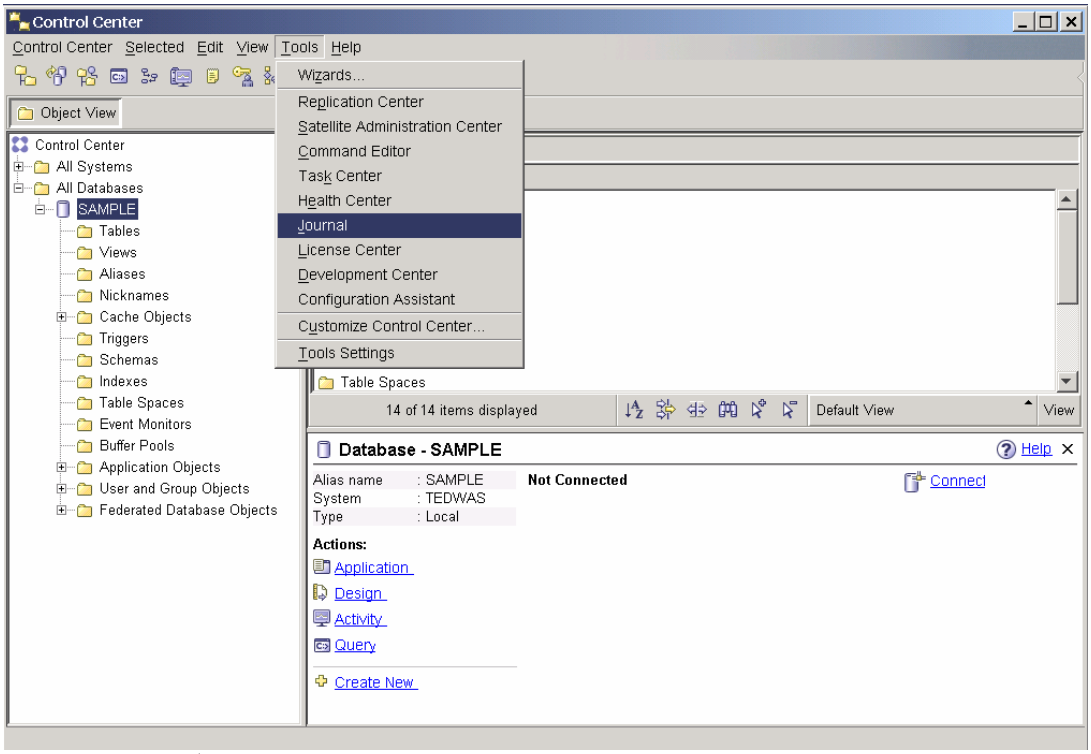


图 5.18 – 运行日志工具

5.8 运行状况监视器（Health Monitor ）

运行状况监视器（Health Monitor）是在 DB2 引擎中默认的状况监视工具，它全方位监控数据库的健康状况（包括内存、磁盘空间、预定义的活动等等）。如果某些方面超出健康阈值，则会抛出异常并引起数据库管理员的注意。运行状况监视器有三种报警等级：

- 注意（Attention）：系统轻微不正常状态
- 警告（Warning）：一个中等严重的状态，并不需要马上的处理，但是整个系统性能受到影响。

- 严重警告（Alarm）：数据库受到严重影响，需要马上进行处理。

可以使用数据库管理配置参数 HEALTH_MON 来打开或者关闭运行状况监视器。

5.8.1 运行状况中心（Health Center ）

运行状况中心是一个与运行状况监视器交互的图形化工具。它按照实例、数据库、表空间三个等级来显示数据库环境的总体状态以及当前的所有报警，如图 5. 19 所示。

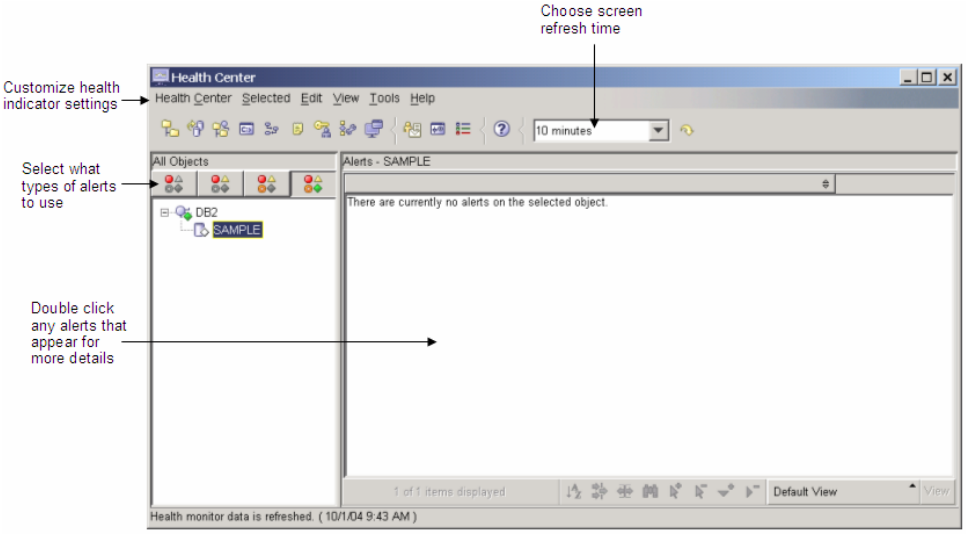


图 5.19 – 运行状况中心

运行运行状况中心

您可以在控制中心中启动运行状况中心。如图 5.20 所示（*Tools* > Health Center）。同样，可以从 Windows 的开始菜单中启动运行状况中心（*Start > Programs > IBM DB2 > DB2COPY1 >Monitoring Tools > Health Center*）。

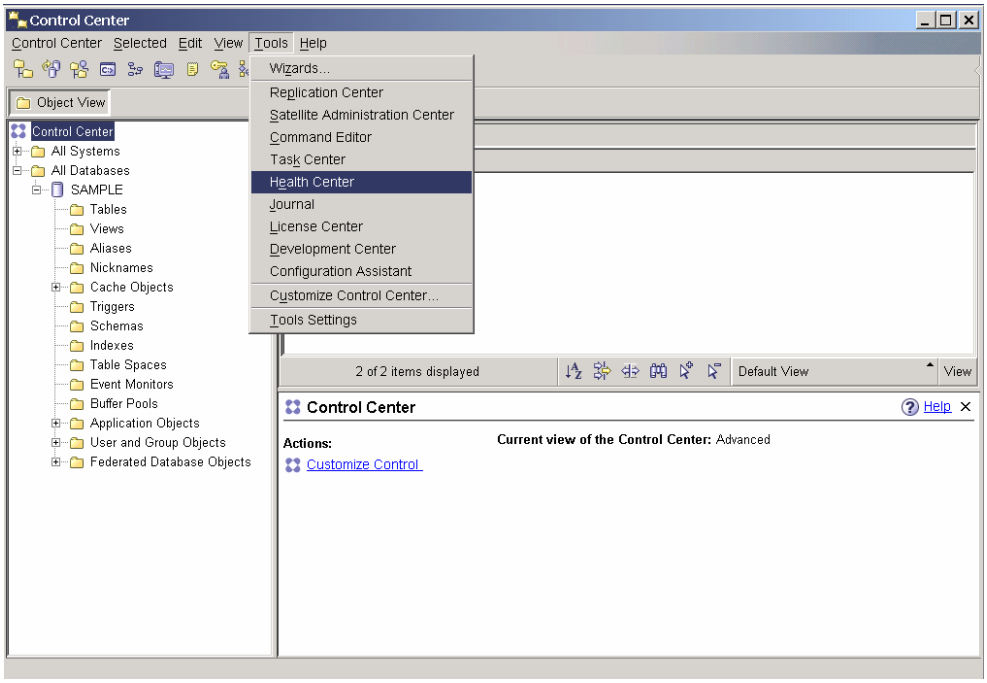


图 5.20 – 启动运行状态中心

配置运行状态报警通知

如果您启动了运行状态中心，您可以通过菜单 *Health Center> Configure > Alert Notification* 来配置报警通知，如图 5.21 所示。报警通知允许您输入出现报警时要联系的名字和 email 地址或者呼叫号码。

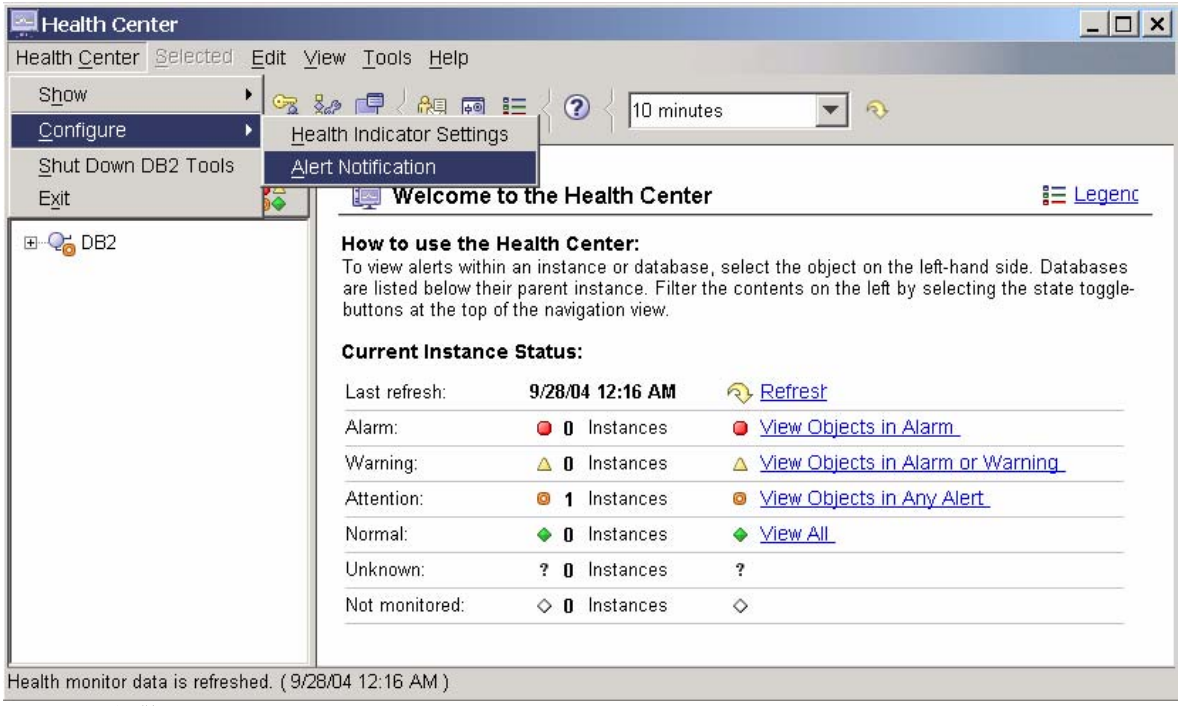


图 5.21 – 报警通知

PART II – DB2 Express-C 数据库管

6

第 6 章 – DB2 体系结构

这章我们简要介绍 DB2 体系结构：

- DB2 进程模型
- DB2 内存模型
- DB2 存储模型

注意：
更多有关 DB2 体系结构的信息请参看以下视频文件：
http://www.channeldb2.com/video/video/show?id=807741:Video:4482

6.1 DB2 进程模型

在图 6.1 描述了 DB2 的进程模型，在这个图中，长方形代表了处理进程，而椭圆形代表了处理线程。DB2 中主进程是 db2sysc。在这个处理进程下有许多的线程，最主要的也称为 db2sysc。这个主要的线程派生了其他的子线程。当一个远程的应用程序尝试使用 SQL CONNECT 语句连接服务器时，通讯协议的远程监听器将接收这个请求，并联系 DB2 协调代理（db2agent）。DB2 代理是一个代表 DB2 实现一些操作的小处理器。当发出请求的应用程序是本地的，也就是说，它与 DB2 在同一个服务器上运行，那么，整个过程中的步骤也是十分类似的，唯一不同的是 db2ipccm 代理取代了 db2tcpcm 线程处理本地应用程序请求。有一些情况，比如，如果发生并行情况时，db2agent 也许会生成其他代理诸如 db2agntp 线程等。其他的一些代理，如图所示：db2pfchr, db2loggr, db2dlock，它们都将应用于不同的目的。表 6.1 描述了大部分公用的进程，表 6.2 描述了大部分公用的线程。

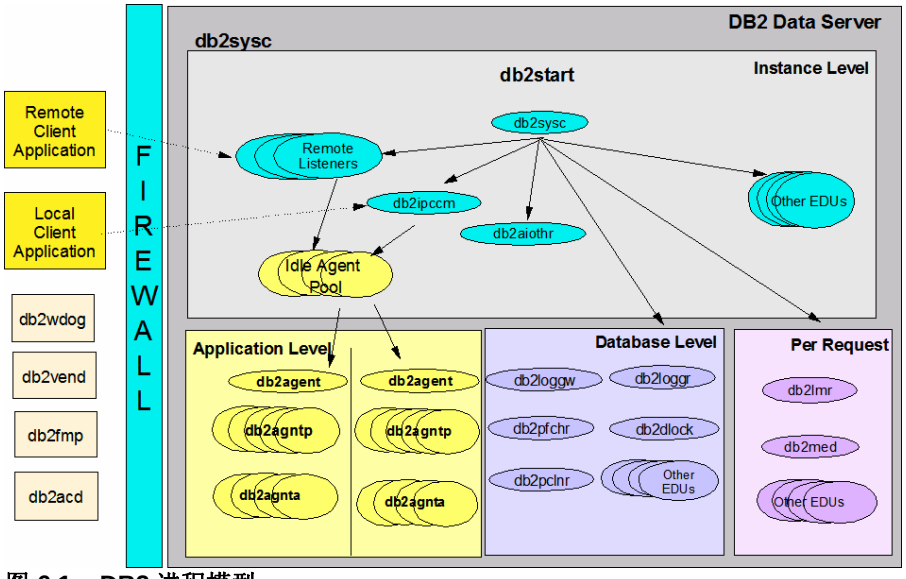


图 6.1 – DB2 进程模型

进程名称	描述
db2sysc (Linux) db2syscs (Win)	DB2 的主系统控制器或者引擎。在 DB2 9.5 中，对于一个完整的分区其中只有一个包含多线程的主引擎进程。所有的引擎可分派单元（EDU）都是进程中的线程。没有这个进程数据库服务器是无法工作的。
db2acd	主管运行状况监视器和自动维护实用程序的自主计算守护程序。此进程以前称为 db2hmon。
db2wdog	DB2 的看门狗。看门狗是主引擎进程的父进程。如果 db2sysc 进程非正常终止，它将清除其所占用的资源。
db2vend	在主引擎进程之外的围栏进程，DB29.5 中，所有的第三方代码都在这个进程中运行。第三方的程序指那些非 IBM 的程序，它们可以和 DB2 交互，例如可以使用第三方程序执行记录归档功能，只需要将用户的出口参数指向这个程序。
db2fmp	围栏进程，在防火墙外运行用户的存储程序和用户定义函数代码。此进程代替了 DB2 老版本中使用的 db2udf 和 db2dari 进程。

表 6.1 – 常见 DB2 进程

线程名称	描述
db2sysc	系统控制线程。这个线程主要负责启动实例、关闭和管理正在运行的实例。
db2tcpcm	TCP/IP 交互监听器
db2agent	协调代理代表应用程序实现数据库操作（如果使用连接集中器，那么至少有一个连接）
db2agntp	如果 INTRA_PARALLEL 的属性设置为 YES。那么会产生活动的副代理。它会为应用程序执行数据库操作。db2agent 将协调不同 db2agntp 副代理之间的工作
db2pfchr	DB2 异步 I/O 数据预取(NUM_IOSERVERS)。
db2pclnr	DB2 异步 I/O 数据写入(NUM_IOCLEANERS)

表 6.2 – 常见 DB2 线程

6.2 DB2 内存模型

DB2 内存模型由内存的不同区域构成，这里的内存是指存在于实例层、数据库层、应用程序和代理层中的内存，如图 6.2 所示。本书中不详细解释每个不同的内存区域，取而代之的，我们提供概要性的总结。

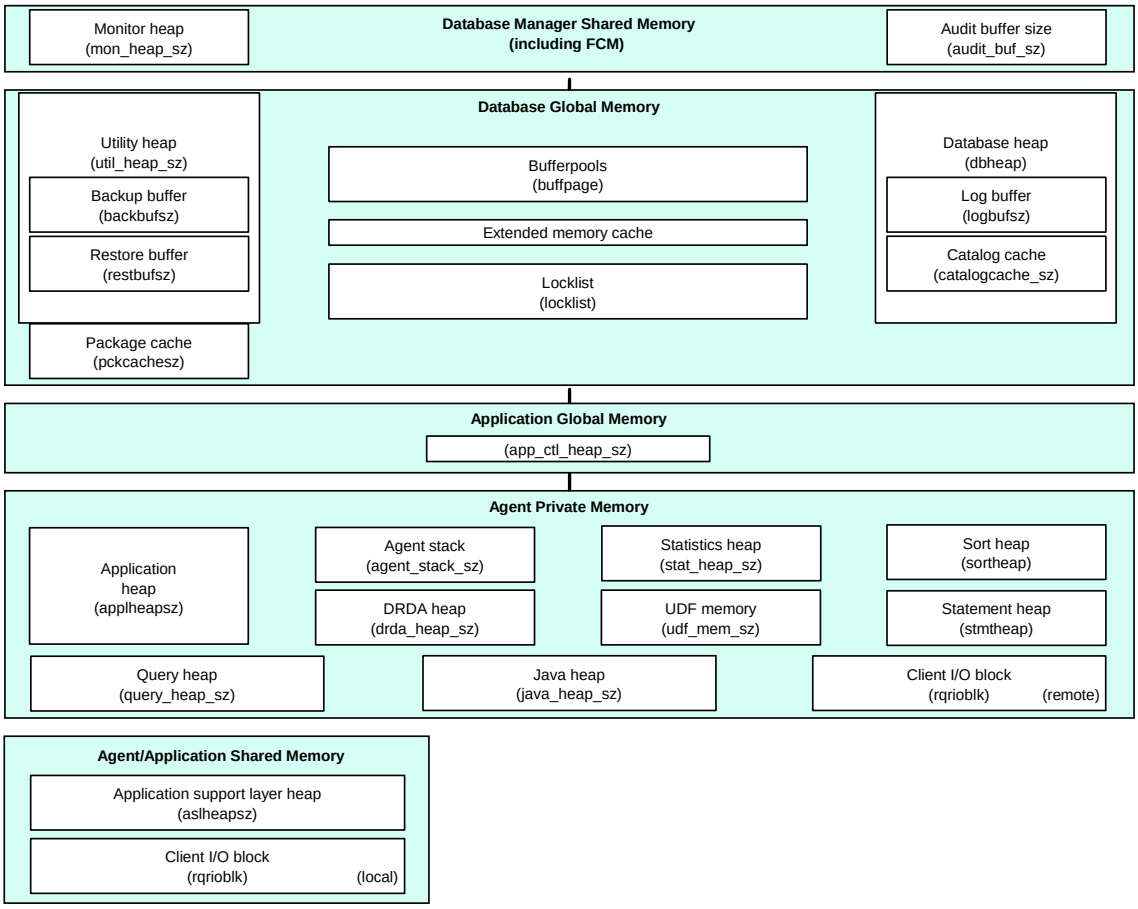


图 6.2 – DB2 内存模型

当一个实例启动后，系统就为数据库管理器分配共享的内存。这不需要花费很大的空间。当您第一次连接数据库时，就会分配 Database Global Memory（数据库全局内存）。在此区域(内存块)内，缓冲池是最重要的部分之一，尤其体现在改善系统查询的操作时。缓冲池的大小将决定整个 Database Global Memory（数据库全局内存）的大小。

每个 DB2 代理都使用一个代理私有内存。在没有连接集中器的情况下，每一个连接都需要一个代理。典型的一个代理可以使用大约 3-5MB 内存。在有连接集中器的情况下，几个链接可以使用一个代理，因此减少了对物理内存的需求。

6.3 DB2 存储模型

本节将会讨论以下概念:

- 数据页和扩展数据块
- 缓冲池
- 表空间

6.3.1 数据页和扩展数据块

页是在 DB2 中的最小存储单元。允许的数据页大小为：4K、8K、16K 和 32K。扩展数据块是一组数据页。DB2 中，每次处理一页会影响数据库的性能，所以 DB2 以扩展数据块为单位进行处理。页大小和扩展数据块大小在使用缓冲池和表空间的时候定义。

6.3.2 缓冲池

缓冲池是表和索引数据在内存中的缓存。它减少持续直接的 IO 存取，提供异步读取（预取）和写入来提高系统的性能。也就是说，DB2 预测那些所要用到的数据页，然后再使用它们之前预先将它们读取到缓冲池中。

缓冲池在内存中以 4K、8K、16K、32K 的页面大小为单位。每一个数据库至少需要一个缓冲池，而且对于每个表空间，至少必须存在一个符合制定大小的缓冲池。

创建缓冲池

您可以使用 CREATE BUFFERPOOL 语句来创建一个缓冲池。当然，您也可以使用控制中心来创建缓冲池。在控制中心中，右键单击某一数据库的缓冲池文件夹，然后选择 Create，如图 6.3 所示。

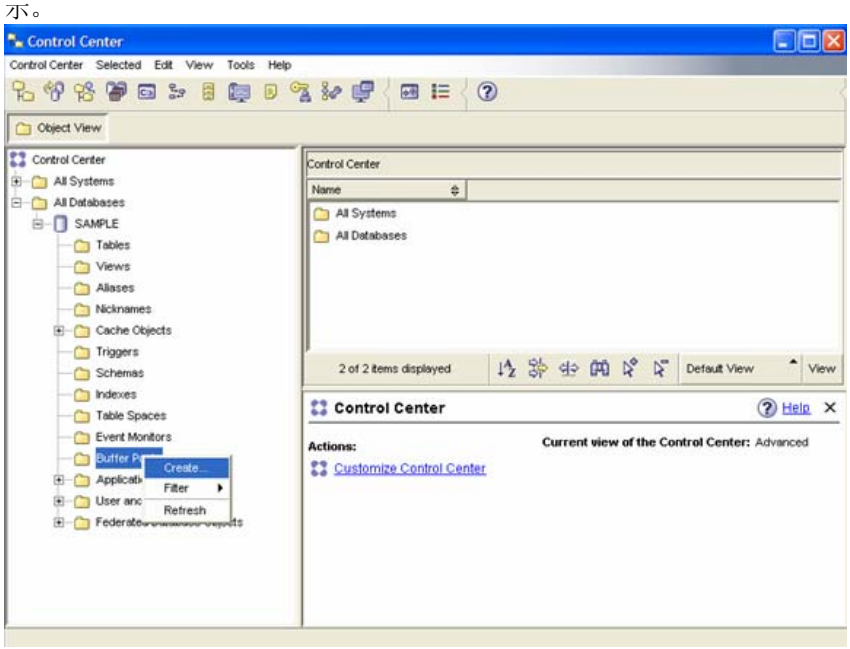


图 6.3 –创建一个缓冲池

点击 Create 后，会出现如图 6.4 所示的创建缓冲池对话框。

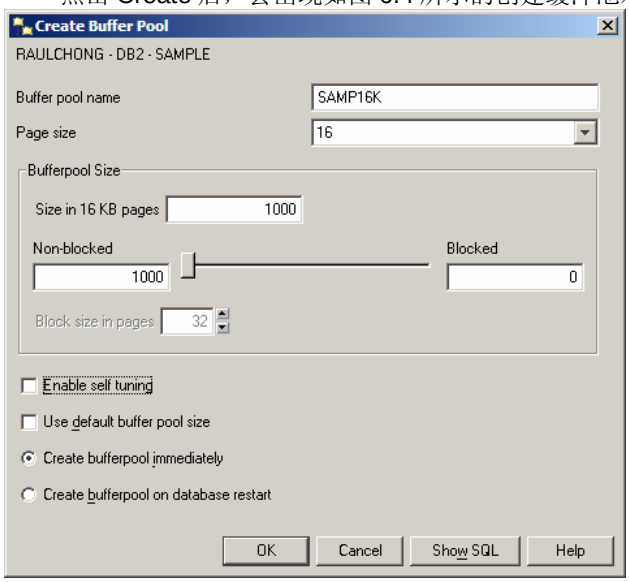


图 6.4 – 创建缓冲池对话框

图 6.4 中的大多数条目都很简单明了，Non-blocked 和 Blocked 区域表示数据页以块或者非块存在的数目。基于块的缓冲池保证硬盘上连续的页面也会连续地移动到块区域中，着能够提高性能。页面数必须小于 98%的缓冲池页面数目。

当缓冲池创建成功后，它就会显示在控制中心里，如图 6.5 所示。

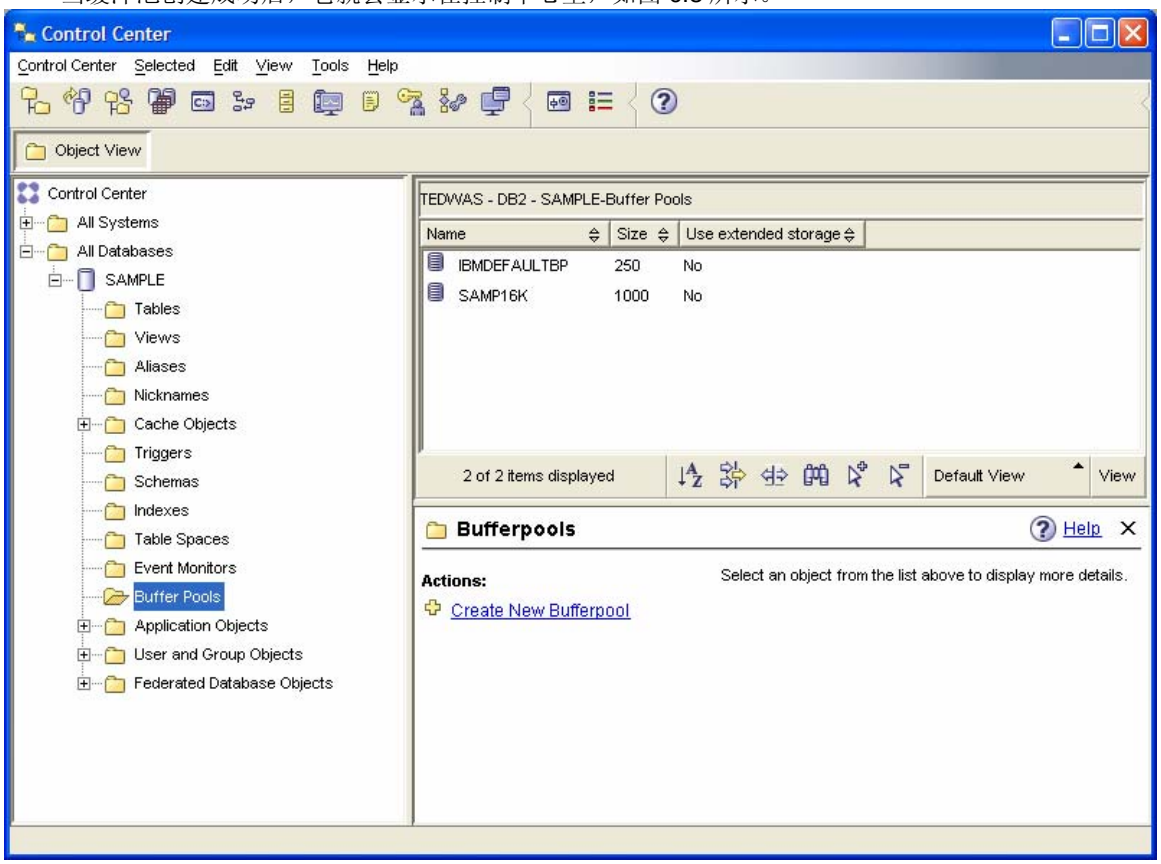


图 6.5 – 创建名为“SAMP16K”的缓冲池后的控制中心示意图

6.3.3 表空间

表空间是处于逻辑的数据表和系统物理内存（缓冲池）以及容器（硬盘）之间的逻辑接口。用 CREATE TABLESPACE 语句来创建一个表空间。创建时您能够制定如下的参数：

- 表空间的页大小 (4KB, 8KB, 16KB, or 32KB)。页大小必须与其关联的缓冲池的大小相同。
- 分配给这个表空间的缓冲池名字。
- 扩展的大小。
- 预取的大小。

表空间类型

表空间有三种类型：

- 常规表空间
  这种表空间用于用户的数据表。比如，默认创建的 USERSPACE1 表空间就是一个常规表空间。
- 大型表空间

这种表空间可作为在原有数据表中分离 LOB 数据的可选表空间。如果创建时指明 pureXML 的支持（数据库以 UNICODE 编码创建，并且使用 XML 数据类型作为列属性），它也能用作存储 XML 数据，这种情况下，大型表空间是默认的表空间。

- 临时表空间
  临时表空间有两种：
  - 系统临时表空间
    系统临时表空间用于 DB2 的内部操作，比如排序等。TEMPSPACE1 就是一个默认创建的系统临时表空间，它在您创建一个数据库时自动建立。

  - 用户临时表空间
    用户临时表空间用于创建用户定义全局临时表（在内存中的临时表）。它们通常和系统临时表空间想混。

表空间的管理

根据表空间的管理方式，能将它们分类。管理方式能在使用 CREATE TABLESPACE 创建表空间时指定。

由系统管理

这种类型的表空间也称为系统管理存储（System Managed Storage，SMS）表空间。这意味着由操作系统来管理表空间的存储。这种方式的表空间是很容易管理的，表空间的容器就是 OS 文件系统的文件夹。存储空间不是预先分配的，但是表空间文件能够动态增长。当您指定容器后，它就会在创建表空间时固定，以后不能再添加其它的表空间容器，除非使用转向存储。当使用 SMS 表空间，表中的数据、索引、LOB 数据不能够跨越不同的表空间。

由数据库管理

这种类型的表空间也称为数据库管理存储（Database Managed Storage，DMS）表空间。这意味着由 DB2 管理表空间的存储。这种表空间的管理需要数据库管理员（DBA）的人工干预，能够预先分配容器或者裸设备。使用裸设备时，数据是直接写入到设备中而没有使用 I/O 缓存。

这种管理方式能够增加、删除容器，也能够改变容器大小。DMS 表空间能够提高性能表现，数据表的数据、索引、LOB 数据能够分割到不同的表空间中，这样能够提高性能。

由自动存储管理

这种管理方式由自动存储来管理，它具有 SMS 表空间类似的自主管理性，又具有 DMS 数据表空间的灵活性和高性能表现。所以，从 DB2 V9.5 开始，这是默认的表空间类型。使用这种表空间时，用户首先指明一个逻辑存储设备组，不需指明其容器。容器会在存储路径里自动创建。容器的增长和新建都是有 DB2 来管理的。

为了使用自动存储管理，您在创建一个数据库时必须允许自动存储管理（这是默认的行为），然后为其分配一个存储路径集合。这个数据库创建之后，如果您需要，可以重新定义存储的路径，这只需使用 RESTORE 操作即可。然后，您就可以创建自动存储管理的表空间（同样，这也是默认的行为）。

自动存储管理举例

首先创建允许自动存储管理的数据库，如下面的例子所示：

创建数据库时默认使用的就是自动存储管理。

```
CREATE DATABASE DB1
```

显式声明使用自动存储管理

```
CREATE DATABASE DB1 AUTOMATIC STORAGE YES
```

默认使用自动存储管理，并且指明存储路径。

```
CREATE DATABASE DB1 ON /data/path1, /data/path2

显式禁用自动存储管理
CREATE DATABASE DB1 AUTOMATIC STORAGE NO
```

然后，创建使用自动存储管理的表空间，如下的例子所示：

```
创建表空间时默认使用的也是自动存储管理：
CREATE TEMPORARY TABLESPACE TEMPTS

显式声明使用自动存储管理的表空间
CREATE TABLESPACE TS2 MANAGED BY AUTOMATIC STORAGE

隐式声明适用自动存储管理，并且指明初始化分配的大小、增长的大小、所能达到的最大容量
CREATE TABLESPACE TS1
        INITIALSIZE 500 K
        INCREASESIZE 100 K
        MAXSIZE 100 M
```

数据是怎么样存储在表空间中的

默认情况下，DB2 每次写入数据到硬盘都会跨越若干个容器。例如，您使用 4K 大小 DMS 表空间，扩展数据块数页数为 8，使用 3 个容器，这就是说，在写入下一个容器之前，会将 32K 数据（4K x 8 页/扩展块 = 32K）写入当前容器中。图 6.6 展示了这种情况。注意，不同表不会共享相同的扩展数据块。

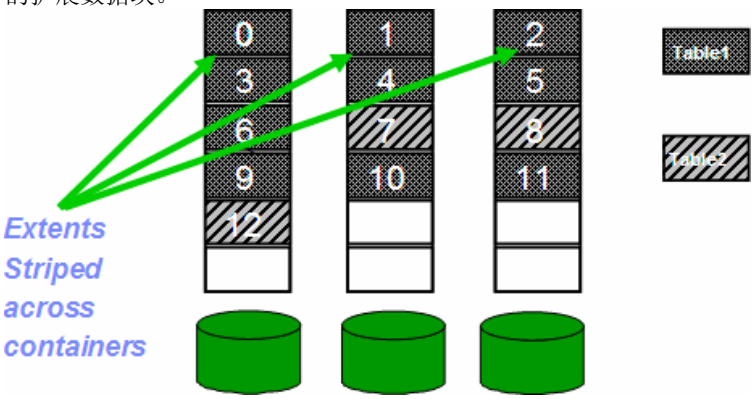


图 6.6 –在表空间中写入数据

使用控制中心创建表空间

使用控制中心创建表空间，首先在某个数据库上右键单击表空间（Table Spaces）文件夹，然后选择 Create，如图 6.7 所示。这是会出现创建表空间向导（Create table space wizard），如图 6.8 所示。根据向导提示，即可完成表空间的创建。

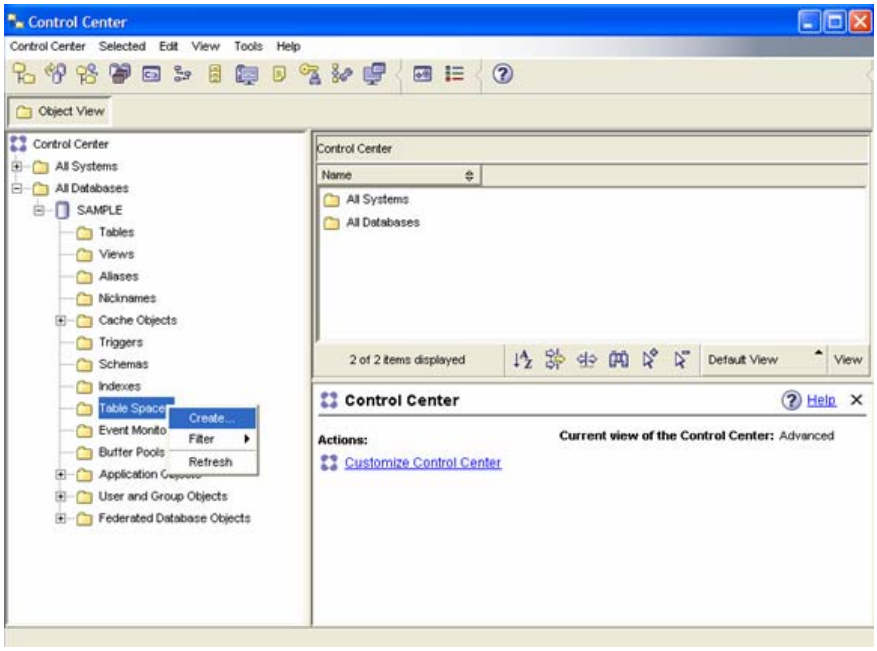


图 6.7 – 从控制中心创建表空间

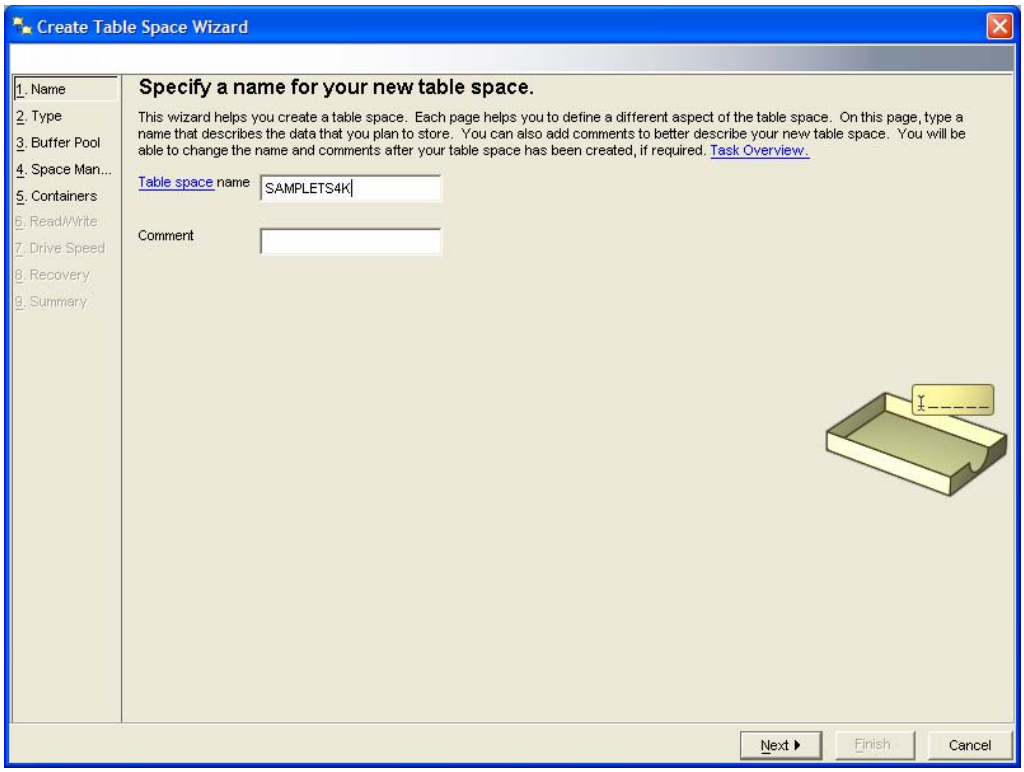


图 6.8 – 创建表空间向导

7

第 7 章 – DB2 客户端的连接

本章将介绍使用 TCP/IP 协议从 DB2 的客户端连接到 DB2 服务器端所需的步骤。因为 DB2 服务器自带了客户端组件，所以 DB2 服务器也可作为一个客户端与其它的 DB2 服务器相连。DB2 客户端连接的设置有几种方式，不过在本章中我们只讨论最简单的一种——使用配置助手。

注意：
更多有关 DB2 结构的信息，请参见一下视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:4222

7.1 DB2 目录

DB2 目录是一些二进制文件，文件中存储的信息是关于从您的机器可以访问到哪些数据库。DB2 共有四个不同的目录：

1. 系统数据库目录(System database directory)

这个目录就像一本书的目录。它显示了所有您可以连接到的本地或远程数据库。对于本地数据库，将会有一个指针来指向这个本地数据库目录（Local database directory）。对于远程的数据库，它将有一个指向结点目录（Node directory）的指针 . 可以使用如下命令查看目录内容：

```
list db directory
```

2. 本地数据库目录(Local database directory)

这个目录包含的信息是关于您可以连接到的本地数据库。可以用如下命令查看目录内容：

```
list db directory on <drive/path>
```

3. 结点目录(Node directory)

这个目录包含的信息是关于如何连接到一个给定的数据库。例如，如果使用的是 TCP/IP 协议，一个 TCP/IP 结点项就将包含您试图连接的 DB2 数据库所在服务器的 IP 地址，以及这个数据库的实例所用的端口。可以用如下命令查看此目录的内容：

```
list node directory
```

4. 数据库连接服务目录（DCS directory , DCS：Database Connection Services directory）

这个目录只有在您已经安装了 DB2 的连接软件（用于连接到 DB2 z/OS（大型机）版或 DB2 i5/OS 版）时才会出现。查看此目录要用到如下命令：

```
list dcs directory
```

以上所有目录的内容都可以通过 GUI 配置助手（Configuration Assistant）工具来进行查看和修改.

7.2 配置助手（Configuration Assistant ）

使用配置助手，您可以轻松地配置 DB2 客户端与服务器端之间的连接.

在 Windows 系统中运行配置助手，您只要选择"开始>程序>IBM DB2>DB2COPY1>设置工具>配置助手"（ *Start > Programs > IBM DB2 > DB2COPY1 > Set-up Tools > Configuration Assistant*）

您也可以用 db2ca 命令从命令行启动这个工具。配置助手如图 7.1 所示：

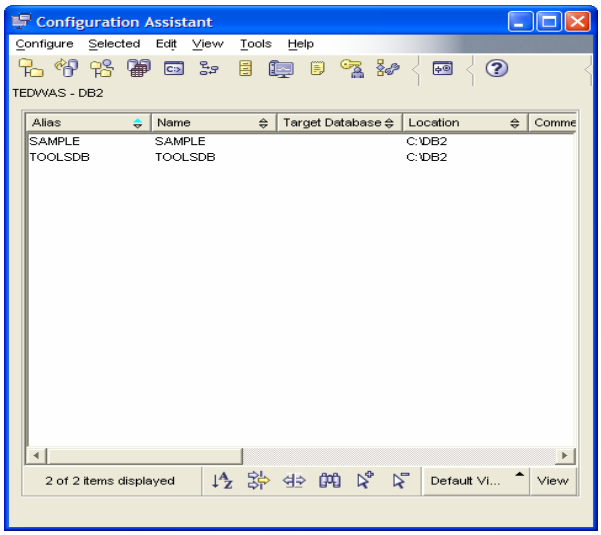


图 7.1 – 配置助手

7.2.1 服务器端的安装要求

服务器端有两处需要设置：

1) DB2COMM

这个注册值决定了应当由哪一个通信协议控制器来监听来自客户端的请求。通常 TCP/IP 是最常用的通信协议。改变这个参数需要重启实例。如需在配置助手中查看或改变 DB2COMM 的值，请选择 配置->DB2 注册表（*Configure -> DB2 Registry*）(如图 7.2，图 7.3 所示).

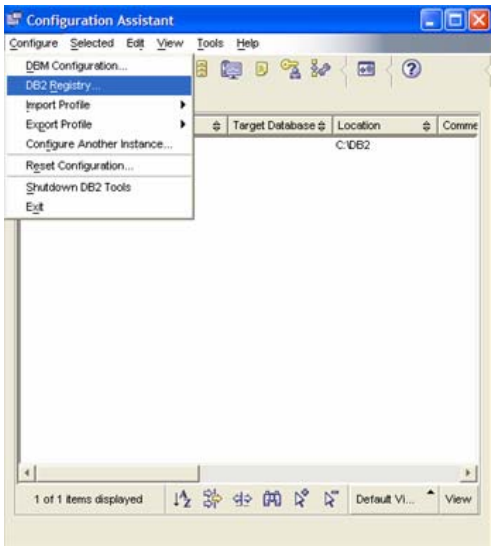


图 7.2 – 访问 DB2 注册表

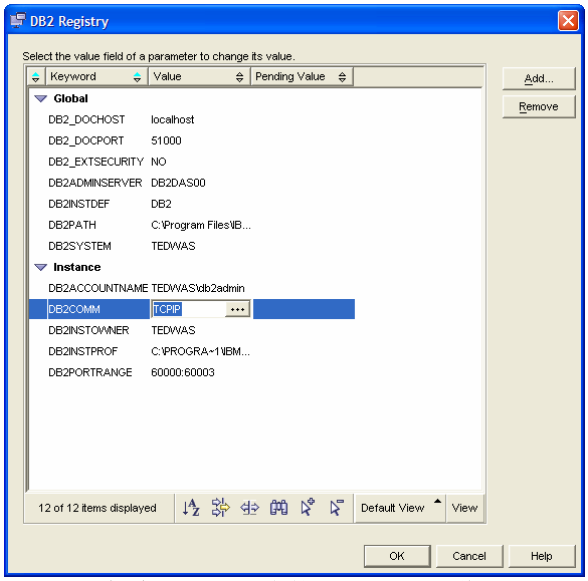


图 7.3 – 查看 DB2 注册表的 DB2COMM 值

2) SVCENAME

这个 DBM 配置 参数应当被设为服务器名(与在 TCP/IP 服务文件中的定义一致) 或是您想在这个实例中访问数据库时所使用的端口号。如图 7.4 所示，在配置助手中，选择 配置>DBM 配置（*Configure > DBM configuration*）。

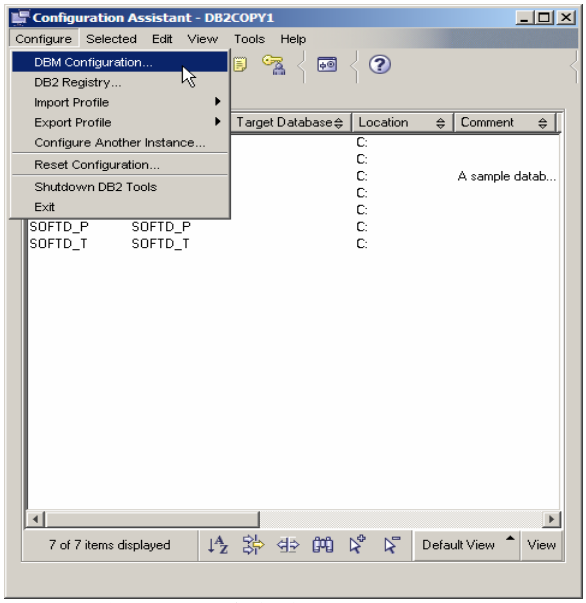


图 7.4 – 在配置助手中查看 DBM 配置

进入 DBM 配置窗口，找到 “Communications”项，然后寻找 SVCENAME.。如有必要，您可以把这个值改为一个与端口号相同的字符串.。如图 7.5。

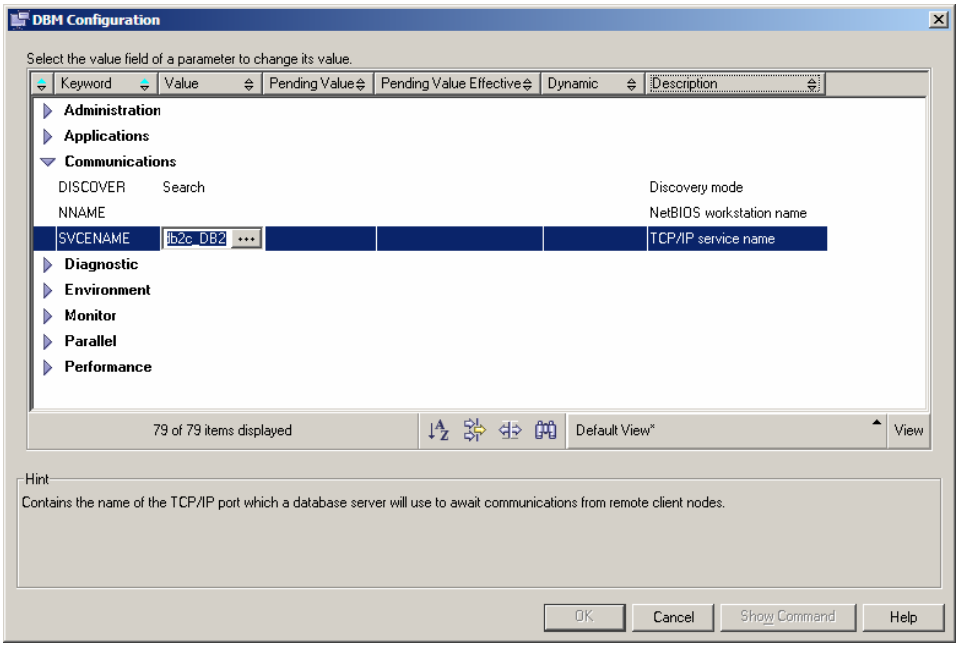


图 7.5 – 查看 DBM 配置的 SVCENAME 参数

7.2.2 Setup required at the client 客户端的安装要求

在客户端，您需要事先了解以下信息：

1. 您要连接的数据库的名称
2. 数据库所在的服务器上 DB2 实例所使用的端口号。您也可以用服务名，只要它与 TCP/IP 服务文件中的项相匹配。
3. 知道操作系统用户名和密码以便连接到数据库时使用。用户名必须已经在服务器端建立。

以上信息可以使用配置助手从 DB2 客户端输入。首先启动添加数据库向导：选择“所选>使用向导来添加数据库”（Selected -> Add Database Using Wizard）选项。(如图 7.6)。

图 7.6 – 运行“使用向导来添加数据库”

您也可以用右键单击配置助手的空白处并选择“使用向导来添加数据库”的方式打开这个向导。

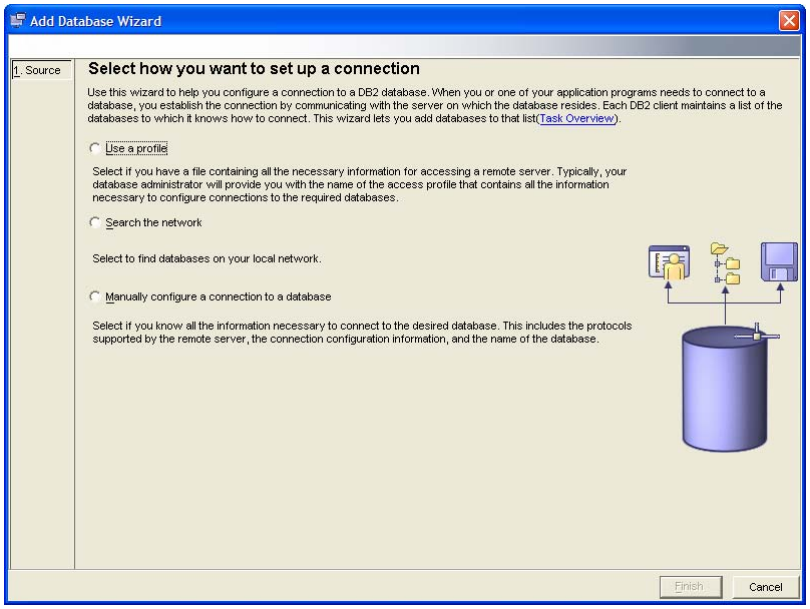


图 7.7 – 添加数据库向导

在添加数据库向导中，有三个选项：

使用概要文件

可能在有些情况下，您需要配置多个客户端到同一个服务器的连接。在这样的情况下，一个方便的做法是在一个客户端上进行所有的配置，并且把这些配置存入一个“概要文件”，然后再其它客户端上直接使用该文件进行配置。 如果您选择了”使用概要文件”， 您将从一个已经存在的概要文件加载信息..。本章稍后将介绍具体该如何建立客户端与服务器的概要文件。

搜索网络

这一方式也称做”探索(Discovery)”， 令 DB2 在网络中搜索给定的服务器、实例和数据库。为使这种方式正常运行，每一个 DB2 服务器上必须运行 DAS（DB2 管理服务器,DB2 Administration Server），使得服务器能够被发现。通常有两种方法来执行搜索：

- 搜索网络 (Search)
  搜索整个网络。如果您的网络很大并且有许多的集线器，那么不推荐您这样做。因为检索每一个系统的数据将会耗费相当长的的时间。
- 已知系统(Known)
  根据您提供的地址在网络中搜索已知的系统。

图 7.8 描述了这两种方式。

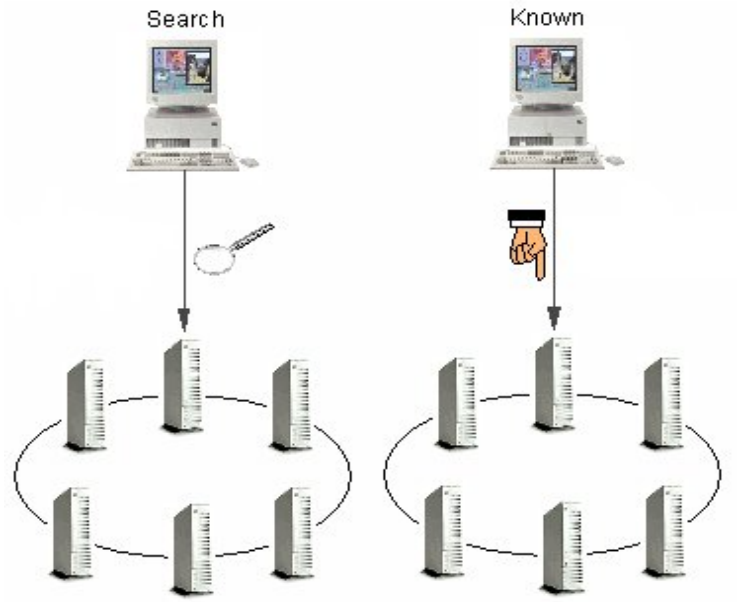


图 7.8 – “搜索网络”与“已知系统” 的搜索(或 探索)方式

可能在一些情况下，管理员并不希望客户端的用户搜索到网络中含有保密信息的数据库。DAS 可以在实例级(Instance level)或者数据库级(database level)上做到这一点。详见图 7.9

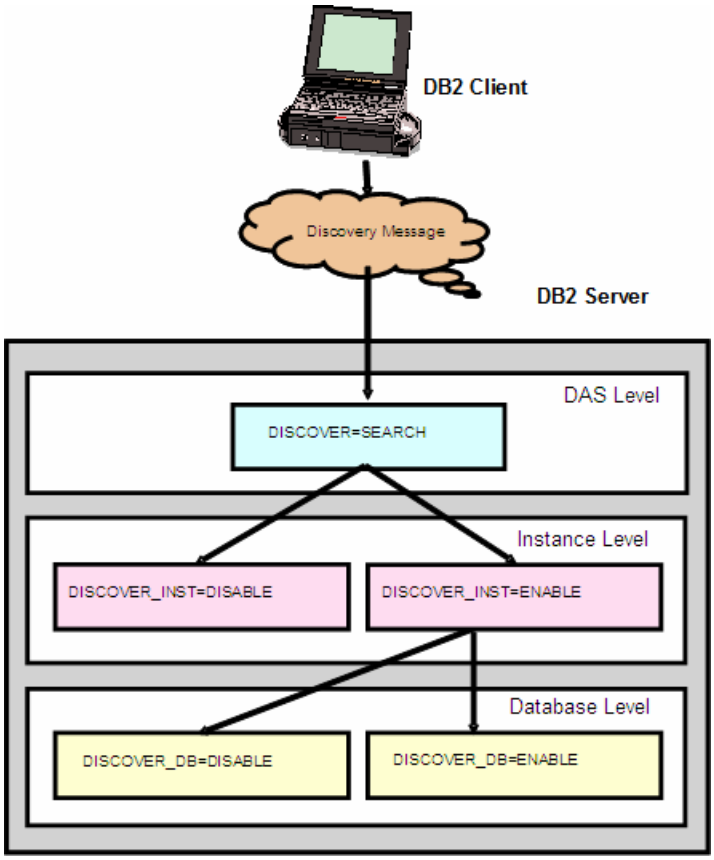


图 7.9 – 为搜索设置参数

如图 7.9 所示，您可以在不同的级别上允许或禁止搜索。在 DAS 级，您可以将 DISCOVER 赋值为 SEARCH 或 KNOWN。在实例级，DBM 配置参数 DISCOVER_INST 可以设为 ENABLE(允许)或 DISABLE(禁止).。这些参数的设定为您提供了所需的数据库搜索的粒度。

手动配置与数据库的连接

使用这一方式，您将手动把主机名、端口号和数据库信息添加到配置助手中，随后会生成一条“catalog”命令来执行这些连接配置。配置助手将不会检查信息的正确性。只有当您无法连接到服务器时才会意识到错误的存在。所以您要确保用以连接到选程数据库的用户名和密码的正确性。在默认情况下，身份验证是在试图连接的 DB2 服务器端进行，所以，您必须提供该服务器上的用户名和密码。

7.2.3 建立客户端与服务器端概要文件

如果您正在配置大量的服务器(或者客户端)， 而又不想独立地配置每一个服务器端(或客户端)。您可以先设定一个服务器(或者客服端)的配置，然后将其导出为概要文件(即配置文件)，再将此概要文件应用到其它的客户端或服务器上。这在设置运行环境时会节省管理员大量的时间。

如图 7.10 所示，要新建一个定制的概要文件，只需在配置助手中点击“配置（Configure）”菜单，然后选择“（导出概要文件=>定制）Export Profile => Customize”。

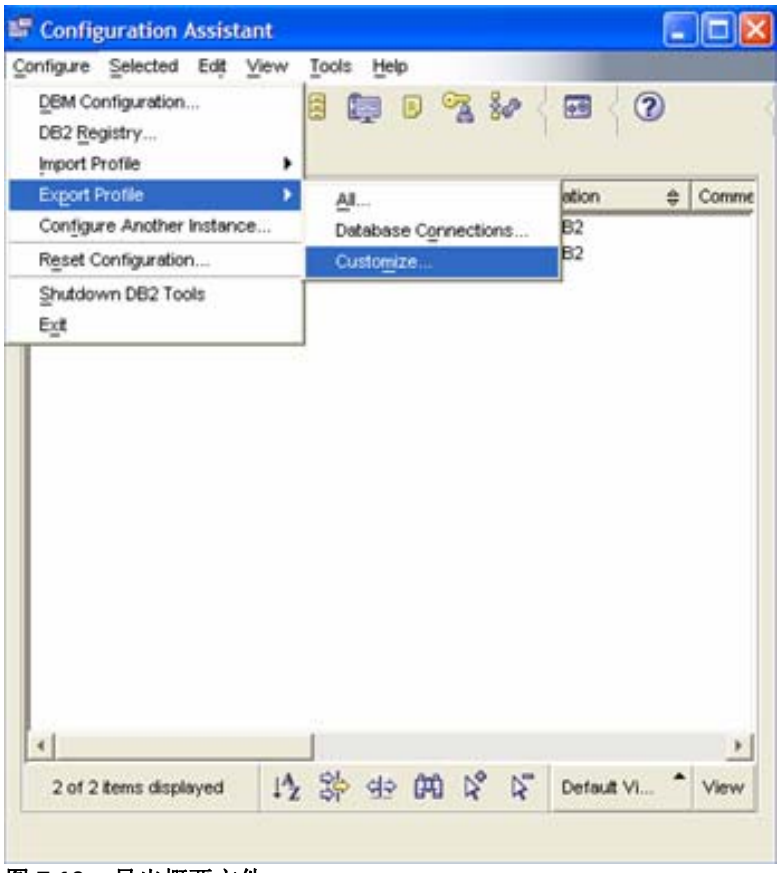


图 7.10 – 导出概要文件

图 7.11 所示为导出概要文件时需要填写的字段。

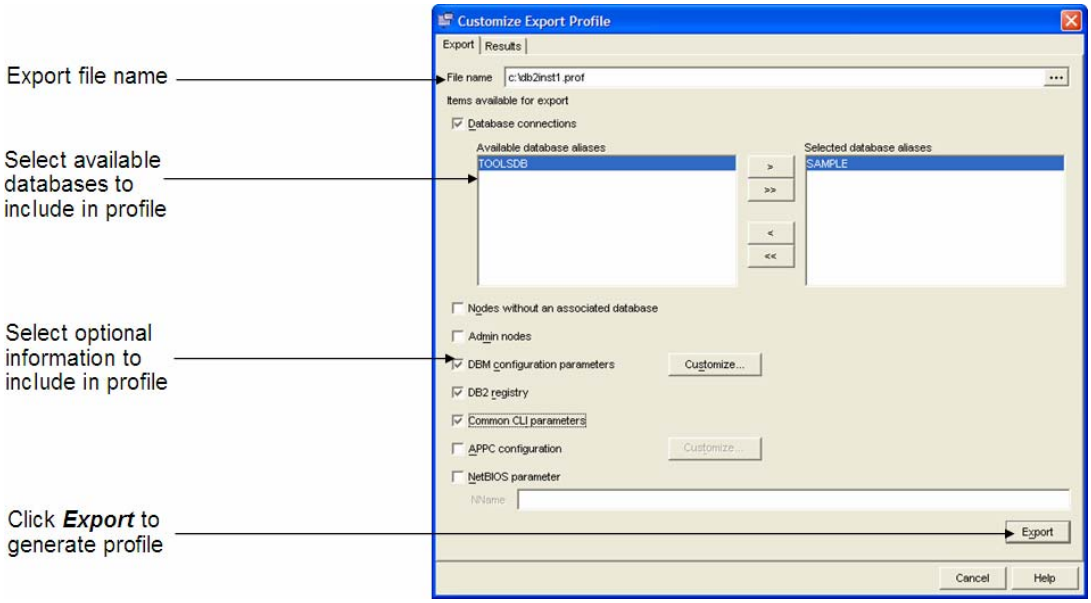


图 7.11 – 定制导出概要文件对话框

图 7.12 所示为点击定制概要文件（Customize Export Profile）对话框的“导出”按钮后的结果。

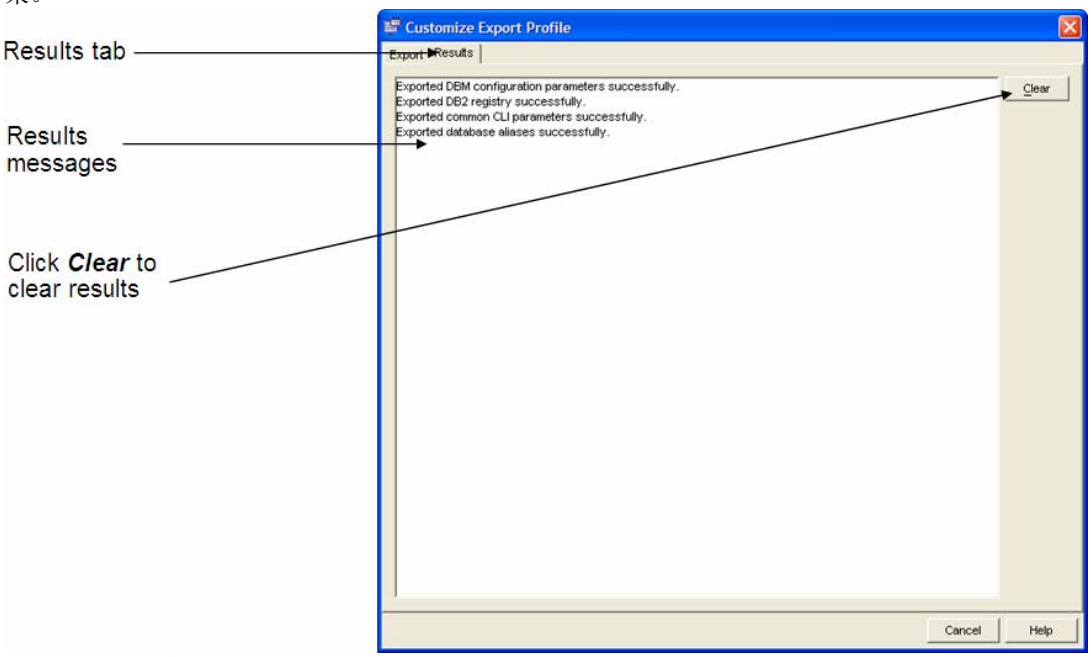


图 7.12 – 导出概要文件的结果

如要从配置助手导入一个定制概要文件，就点击配置菜单（Configure），然后选中“导入概要文件=>定制”（Import Profile => Customize），如图 7.13 所示.

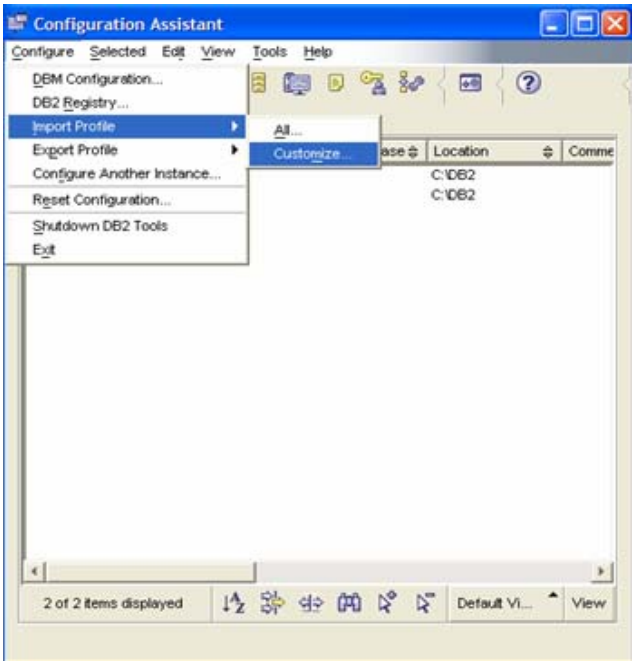


图 7.13 – 导入概要文件

图 7.14 所示为导入概要文件时需填写的字段.

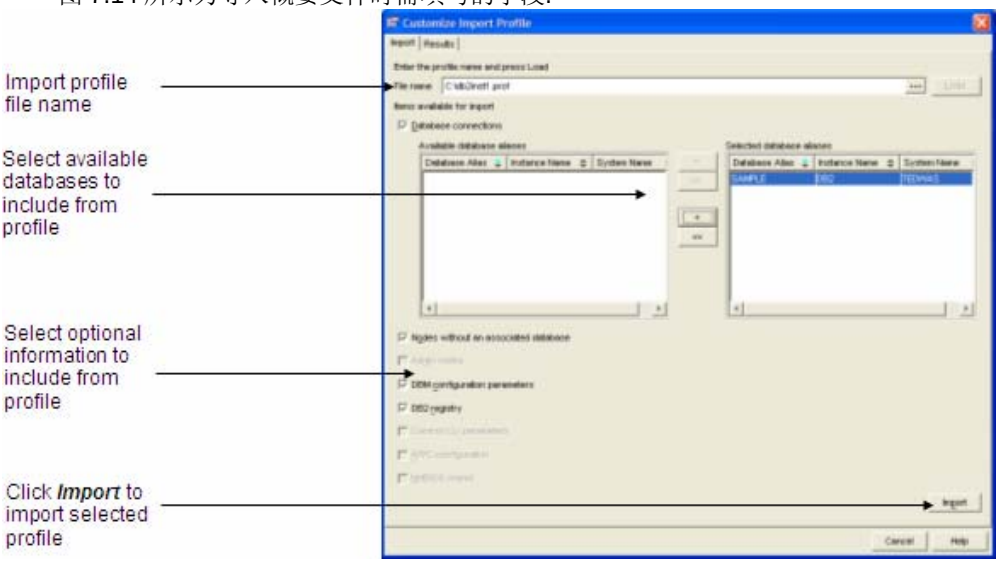


图 7.14 – 定制导入概要文件

实验 #6 使用配置助手

实验目标

配置助手可以用来简便地设置远程数据库连接。 在本次练习中，您将把一个位于远程 DB2 服务器(用另一台工作站代替)上的数据加入目录，使用搜索网络和已知系统搜索两种方式添加目录。一经加入目录，您就可以像访问本地系统一样访问它。DB2 在后台完成所有的通信过程。

注:

本实验假设您是工作在一个网络内。如果不是，您可以同时把您的计算机用作客户端和服务器，然后按照下面配置说明连接到您自己的系统。

实验过程

1. 向另一台工作站用户询问以下的信息:

   远程数据库信息:

(PR)	Protocol 协议	TCPIP
(IP)	IP Address or hostname IP 地址或主机名	_____
(PN)	Instance Port Number 实例的端口号	_____
(DB)	Database Name 数据库名	SAMPLE

   提示:

   - 在 Windows 系统命令行键入 hostname 来获取主机名
   - 在 Windows 系统的命令行键入 ipconfig 来获取 IP 地址

2. 打开配置助手。(提示:可以从”开始”菜单访问)

3. 打开“所选（Selected）”菜单并选择“使用向导来添加数据库（Add Database Using Wizard）”。

4. 在向导的“源（Source）”页面中选择“手工配置与数据库的连接 Manually Configure a Connection to a Database”。

5. 在向导的“协议 Protocol”页面，选择 TCP/IP 选项。点击“下一步”按钮进入向导的下一页。

6. 在向导的”TCP/IP”页，输入您在第(1)步中记下的完整的主机名或 IP 地址、端口号，点击”下一步”按钮进入下一页。

   注意: 仅在您的本地服务文件有一个与远程服务器上实例监听的端口一致的端口号的条目时，才可以用“服务名称”选项。当您使用这个选项时，DB2 将会在本机上而不是服务器上查找服务文件。如果您想要用此选项，必须向本地服务文件中添加一个项。

7. 在向导的“数据库（Database）”页上，在“数据库名称”处输入您第（1）步记录的定义在远程服务器上的数据库名。注意“数据库别名”会自动填充相同的值。数据库别名通常是指本地应用程序用来与这个数据库连接时所使用的名称。既然您已经定义了一个叫作“SAMPLE”的数据库，DB2 将不会允许您在目录中以同样的名称登记一个数据库。因此，您必须使用一个不同的别名。在这个例子中，我们把数据库别名改为 SAMPLE1。有必要的话您可以添加一个注释。点击“下一步”按钮进入向导的下一页。

8. 在“数据源 Data Source”页上，您可以选择把这个新的数据库(数据源)注册为一个 ODBC 数据源。它会自动地在 Windows 的 ODBC 管理器中为您进行注册。在这里，取消“为 CLI/ODBC 注册此数据库”的勾选。点击 Next 按钮进入向导的下一页。

9. 在向导的“节点选项 Node”页，指定数据库所在的远程服务所用的操作系统。因为这个实验中所有的工作站都使用微软的 Windows 系统，所以确认您选中了下拉菜单里的“Windows”项。把这里的“实例名”字段设为“DB2”。点击 Next 按钮进入向导的下一页。

10. 在向导的“系统选项 System Options”页，您将会确认系统和主机名是否正确，并核对操作系统的设置。点击 Next 按钮进入向导的下一页。

11. 向导的“安全性选项 Security Options”页面，您可以指定用户认证在何处进行，以及您想采用的认证方式。选中“使用服务'DBM'配置中的认证值 Use authentication value in server's DBM Configuration”。这样就将采用在远程实例配置文件中 AUTHENTICATION 参数指定的认证方式。点击“完成 Finish”按钮，将远程数据库加入目录并关闭向导。这时应该出现一个确认对话框。点击“测试连接 Test Connection”按钮来确认是否可以成功连接该数据库，请确认您提供的定义在远程服务器上的用户名和密码是有效的(前提是服务器的 AUTHENTICATION 参数值被设为 SERVER)。如果测试成功，意味着您已成功地将远程数据库加入目录。否则，需要返回向导确认所有指定的值都是正确的(点击”修改”按钮回顾向导中的设置)。

12. 打开控制中心，尝试查看新加入目录的远程数据库的各个表。

13. 回到配置助手，尝试向目录中添加一个不同的数据库。这次选用“搜索网络 Search the Network”选项。在向导中采用与刚才同样的方式进行设置。注意，在比较大的网络中，搜索可能会耗较长的时间才返回结果。

8

第 8 章 – 数据库对象

本章讨论模式、数据表、视图、索引、序列等数据库对象。一些高级的数据库应用对象诸如触发器、用户定义函数和存储过程将在第 14 章讨论，SQL PL 存储过程、直接插入 SQL PL、触发器在第 15 章讨论。

注意：
更多关于数据库对象的信息，请参见以下视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:4242

8.1 模式

模式是一组数据库对象的命名空间。它主要用于：

- 提供对象所有权的标识或某个应用的关系，
- 合理地将相关对象组合在一起。

所有的 DB 数据库对象标识名都有可以一分为二，模式就是此名字的前半部分。

<模式名>.<对象名>

一个标识名称必须是独一无二的。在没有指定模式的情况下连接数据库并创建或引用数据库对象时，DB2 使用用户 ID 作为模式名称来连接数据库。例如，使用 arfchong 用户名连接 SAMPLE 数据库，然后用 CREATE TABLE 语句创建表：

```
CREATE TABLE artists …
```

所建的数据库的标识名称即为 `arfchong.artists`。

8.2 表

表是一组由相关数据逻辑安排的行和列。下面是一个用 CREATE TABLE 语句创建表的例子。

```
CREATE TABLE artists
 (artno           SMALLINT    not null,
  name            VARCHAR(50) with default 'abc',
  classification  CHAR(1)     not null,
  bio             CLOB(100K)  logged,
  picture         BLOB(2M)    not logged compact
 )
 IN mytbls1
```

在接下来的部分，我们将阐述这个 CREATE TABLE 语句的主要部分。

8.2.1 数据类型

图 8.1 列出 DB2 支持的数据类型。

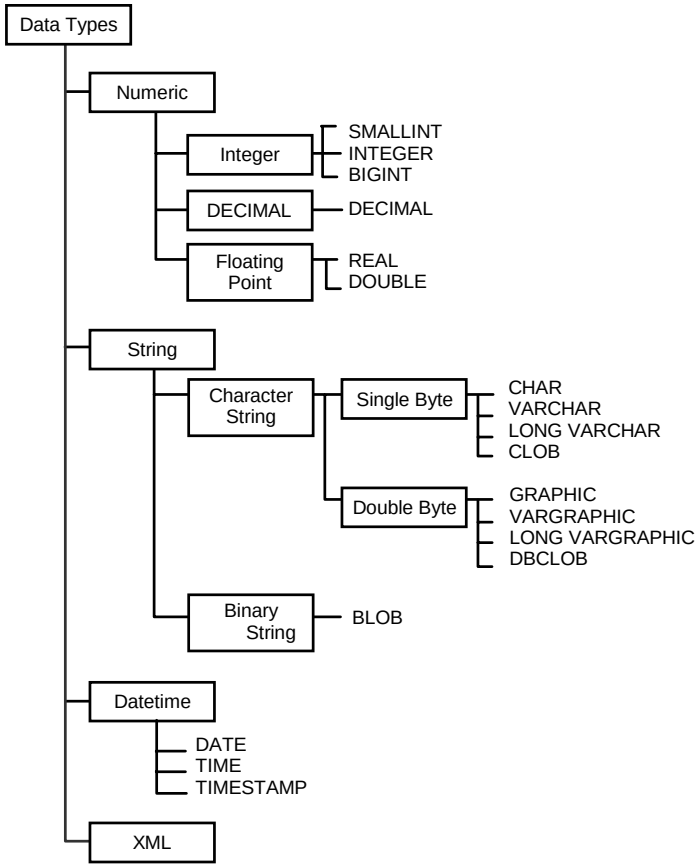


Figure 8.1 – DB2 内置数据类型

大对象数据类型（LOB）

大对象数据类型（Large Object，LOB）用来存储大字符串、大二进制串或文件，如图 8.2 所示。

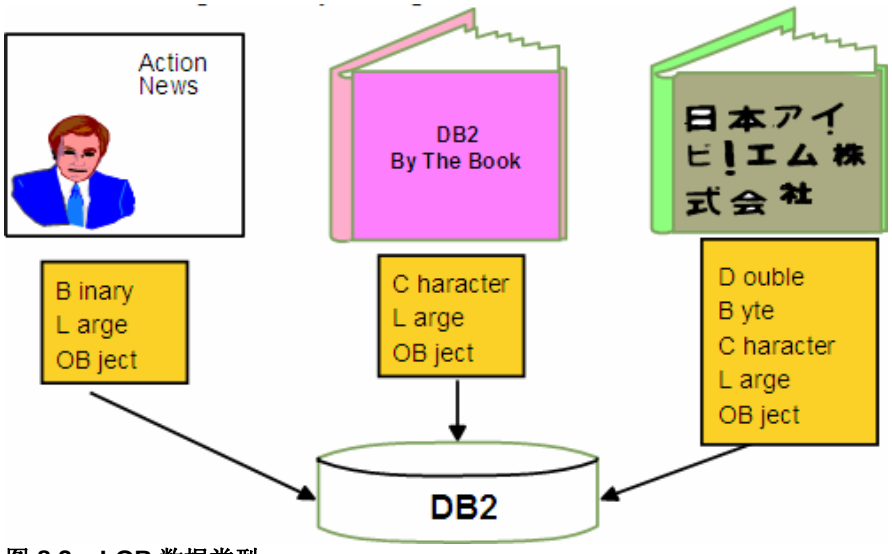


图 8.2 – LOB 数据类型

这些大对象二进制类型常简写为下面的形式：二进制大对象简写为 BLOB，字符大对象简写为 CLOB，双字节字符大对象简写为 DBCLOB。

用户定义类型

DB2 允许您使用内置数据类型定义自己的数据类型，它们称为用户定义类型（UDT），UDTs 常出现在以下情况：

- 需要为值建立上下文
- 需要使 DB2 执行的数据类型

下面的语句序列阐述了怎样和何时使用 UDT。

```
CREATE DISTINCT TYPE POUND AS INTEGER WITH COMPARISONS

CREATE DISTINCT TYPE KILOGRAM AS INTEGER WITH COMPARISONS

CREATE TABLE person
   (f_name    VARCHAR(30),
    weight_p  POUND NOT NULL,
    weight_k  KILOGRAM NOT NULL )
```

在这个例子中，有两个 UDT 被创建：POUND 和 KILOGRAM。这两个都是基于 INTEGER 内置类型。WITH COMPARISONS 子句表明作用于该数据类型的比较转换函数也将会被分别创建。

表 person 在列 weight_p 和 weight_k 分别使用这两个新的 UDT。如果我们使用以下语句。

```
SELECT F_NAME FROM PERSON
  WHERE weight_p > weight_k
```

您会得到一个错误信息，因为两个正在作比较的列使用的是不同的数据类型。即使 weight_p 和 weight_k 分别使用 POUND 和 KILOGRAM，并且两个 UDT 均是基于 INTEGER 数据类型，这种类型的比较也是不可行的。这其是正是我们所对面要的，因为在现实生活中，磅和公斤之间的比较会什么意义呢？这种比较不具有任何意义。

在接下来的例子中，您可能想将 weight_p 和一个触发器进行比较，但是这两个数据类型是不同的，因此除非使用一个转换函数，否则也将会收到一个错误信息。

正如下面的语句所示，进行我们想要的比较，要使用转换函数 POUND()进行转换。正如早先说明的，当调用 CREATE DISTINCT TYPE 语句的 WITH COMPARISONS 子句时 POUND()转换函数与 UDT 一同被创建。

```
SELECT F_NAME FROM PERSON
  WHERE weight_p > POUND(30)
```

零值（Null）

一个零值代表一个示知的状态。不过，CREATE TABLE 语句可以使用 NOT NULL 子句定义一列。这就保证了该列都包含已知的数据值。如果某列定义了 NOT NULL，您也可以为该列赋予一个默认值。如下示例：

```
CREATE TABLE Staff (
     ID       SMALLINT NOT NULL,
     NAME     VARCHAR(9),
     DEPT     SMALLINT NOT NULL with default 10,
     JOB      CHAR(5),
     YEARS    SMALLINT,
```

```
SALARY   DECIMAL(7，2)，
COMM     DECIMAL(7，2) with default 15
)
```

8.2.2 标识列

一个标识列就是一个可以为每个插入行自动产生唯一值的数字列。每个表的标识列不得多于一个。

有两种根据定义为标识列产生数值的方法：

- **常规产生:**值常被 DB2 产生。应用程序不允许提供确切值
- **默认产生:** 值可以明确地由应用程序提供，或者应用程序不提供时由 DB2 产生。DB2 不能保证单值性。这个选项是为生成数据、卸载或重载表提供的。

让我们看一看接下来的例子：

```
CREATE TABLE subscriber(subscriberID INTEGER GENERATED ALWAYS AS
                                     IDENTITY (START WITH 100
                                     INCREMENT BY 100)，
                        firstname VARCHAR(50)，
                        lastname  VARCHAR(50) )
```

在这个例子中，列 subscriberID 是 INTEGER 类型，被定义为标识列并且是常规定义。其值将从 100 开始，按增量 100 增加。

8.2.3 序列对象

序列对象产生整个数据库的一个唯一值。跟标识列不一样的是，序列独立于数据表。下面提供一个例子：

```
CREATE TABLE t1 (salary int)

CREATE SEQUENCE myseq
   START WITH 10
   INCREMENT BY 1
   NO CYCLE

INSERT INTO t1 VALUES (nextval for myseq)
INSERT INTO t1 VALUES (nextval for myseq)
INSERT INTO t1 VALUES (nextval for myseq)

SELECT * FROM t1

SALARY
-----------
         10
         11
         12
  3 record(s) selected.


SELECT prevval for myseq FROM sysibm.sysdummy1

1
-----------
```

```
          12
  1 record(s) selected
```

PREVVA 提供序列的一个当前值，而 NEXTVAL 提供下一个值。

上面的例子也使用 SYSIBM.SYSDUMMY1。这是一个包含一行一列的系统目录表。如果一个查询要求只得到一个输出，可以使用该表。系统目录表在下节进行阐述。

8.2.4 系统目录表

每个数据库都有它自己的系统目录表和视图。它们存储关于数据对象的元数据。你可以像使用普通的数据库表一样查询这些表。有三种模式用来识别系统目录表：

- SYSIBM: 基本表， 对 DB2 使用进行最优化
- SYSCAT: 基于 SYSIBM 表的视图， 对平常轻负荷使用进行优化
- SYSSTAT: 数据库分析

下面是一些目录视图的例子

- SYSCAT.TABLES
- SYSCAT.INDEXES
- SYSCAT.COLUMNS
- SYSCAT.FUNCTIONS
- SYSCAT.PROCEDURES

8.2.5 已声明临时表

已声明临时表是内存中创建的表，供应用程序使用，当应用程序终止时就会自动删除。这些表只可以被创建它们的应用程序访问。它们在 DB2 目录表中并不存在相应记录条目。由于没有目录的争夺，没有行的锁定，也没有默认日志（日志是可选的）和没有权限检查，存取这些表是非常高效的。临时表支持索引，可以在临时表上建立标准的索引，也可以在临时表上运行 RUNSTATS 收集统计信息。

已声明临时表创建在用户的临时表空间中，这个空间必须在创建任何已声明临时表之前被创建。以下语句创建 3 个已声明临时表。

```
CREATE USER TEMPORARY TABLESPACE apptemps
     MANAGED BY SYSTEM USING ('apptemps');

DECLARE GLOBAL TEMPORARY TABLE temployees
     LIKE employee NOT LOGGED;

DECLARE GLOBAL TEMPORARY TABLE tempdept
      (deptid CHAR(6),  deptname CHAR(20))
       ON COMMIT DELETE ROWS NOT LOGGED;

DECLARE GLOBAL TEMPORARY TABLE tempprojects
     AS ( fullselect ) DEFINITION ONLY
     ON COMMIT PRESERVE ROWS NOT LOGGED
     WITH REPLACE IN TABLESPACE apptemps;
```

当一个已声明临时表被创建，它的模式必须被指定为 SESSION。用来创建临时表的用户 ID 必须具有在表上所有的权限。每个创建临时表的应用程序都具有它自己的独立拷贝，如图 8.5 所示。

图 8.5 – 全局临时表的范围

实验 #7 创建一个数据表

实验目的

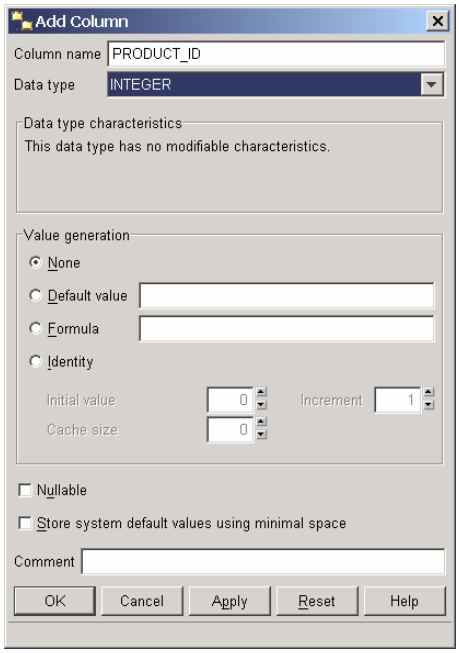
到目前为止，您已经使用过 *SAMPLE* 数据库中的现存数据库。本实验，需要创建自己的数据库和表。您可以使用创建数据表向导（Create Table Wizard） 在 SAMPLE 数据库中创建两个新表。

实验过程

1. 按之前介绍的方法，打开 Create Table Wizard (*Control Center -> All Databases -> SAMPLE -> (right-click) Tables object -> Create …* option
2. 定义表名，列定义和其它的约束。这个表存储在 SAMPLE 数据库中，记录项目的办公用具采购信息。每次采购，表中都相应该增加一行。这个表有六列：
   - product_id: 被购买商品的唯一标识
   - description: 商品描述
   - quantity: 采购量
   - cost: 花费
   - image:所购买商品的图片 (可为空)
   - project_num: 购买这些商品的目标工程

3. 在这个向导的第一页，在模式名中，输入当前登录的用户 ID，并且使用表名：*SUPPLIES*。您也可以输入注释。点击 *Next* 按钮继续向导的下一页。

4. 在这一页，可以为表增加一列。点击 *ADD* 按钮增加一列。



输入列名“*product_id*”，并选择数据类型：*INTEGER*。*Nullable* 置空，点击 *Apply* 按钮来定义此列。

使用下图所示的选项重复这一步骤完成此表中其它列。当所有的列都已经添加完成，点击 *OK* 按钮，得到刚才所创建列的总结。点击 *Next* 按钮继续向导的下一步。

列名	列属性
product_id (completed)	INTEGER， NOT NULL
description	VARCHAR， length 40， NOT NULL
quantity	INTEGER， NOT NULL
cost	DECIMAL， Precision 7， Scale 2， NOT NULL
image	BLOB， 1MB， NULLABLE， NOT LOGGED
project_num	CHAR， length 6， NOT NULL

注意：当声明 LOB 列时选项 NOT LOGGED 可以被指定。它强制约束列可以大于 1GB。对大于 10MB 的 LOB 进行更改会很快添满日志文件，这种情况下，常常推荐使用此选项。即使使用 NOT LOGGED 选项，事务过程中对 LOB 类型的更改仍可以成功回滚。同时注意 image 列是唯一一个被定义为 NULLABLE 的列。你认为为什么这个列要这样定义呢？

5. 在这一点上，提供创建表的所有的强制约束信息。可以直接跳过其它的页，以使用系统的默认选项。表建成以后也可以增加键和约束。点击 *Next* 按钮继续向导的下一步。

6. 为表增加一外约束条件来限制 quantity 列的值。在向导的 Constraint 页，点击 ADD 按钮。在 *Check Name* 部分输入：*valid_quantities*.在 *Check Condition* 部分输入：*quantity > 0*。点击 OK 按钮。就可以在向导的 Constraint 页看到刚才所设置的约束信息列表。

7. 继续操作向导来改变表的其它参数。也可以直接跳到 *Summary* 页或简单地点击 *Finish* 按钮来创建表。

8. 在控制中心，点击 *SAMPLE* 数据库下的 *Tables* 文件夹。刚才所创建的表现在可以在列表中找到。有时候需要刷新控制中心来查看其中的改变。

8.3 视图

视图是表中多个数据的一次描述。视图中的数据不是单独存储在一起的，但当视图被调用时就可以被获得。嵌套视图是一基于其它视图创建的视图。所有有关视图的信息保存在 DB2 的目录视图中：SYSCAT.VIEWS，SYSCAT.VIEWDEP 和 SYSCAT.TABLES。下面是一个创建视图和使用视图的例子：

```
CONNECT TO MYDB1;

CREATE VIEW MYVIEW1
  AS SELECT ARTNO,  NAME,  CLASSIFICATION
  FROM ARTISTS;

SELECT * FROM MYVIEW1;

Output:

ARTNO            NAME                 CLASSIFICATION
------           -----------------    --------------
  10             HUMAN                  A
  20             MY PLANT               C
  30             THE STORE              E
  ...
```

8.4 索引

索引是有序键值的集合，每一个键值指向表的一行。索引的值可以唯一，它改善了数据库的性能。在索引上可以定义如下的一些特性:

- 索引顺序可以递增也可以递减速
- 索引键可以是独值的也可以不是独值
- 一些列可以一起用作索引 (这被称作混合索引)
- 如果索引和物理数据串在聚集在一个相似的索引序列中，它们就成为簇索引。

例如：

```
CREATE UNIQUE INDEX artno_ix ON artists (artno)
```

8.4.1 Design Advisor

Design Advisor 是在给定 SQL 工作量的前提下如何优化数据库设计的绝好工具。Design Advisor 可以帮助设计索引、具体化查询表（MQT），多维集群（MDC）和数据库分区特性。Design Advisor 可以从控制中心中调用，右键点击数据库并如图 8.6 所示，选择 Design Advisor。

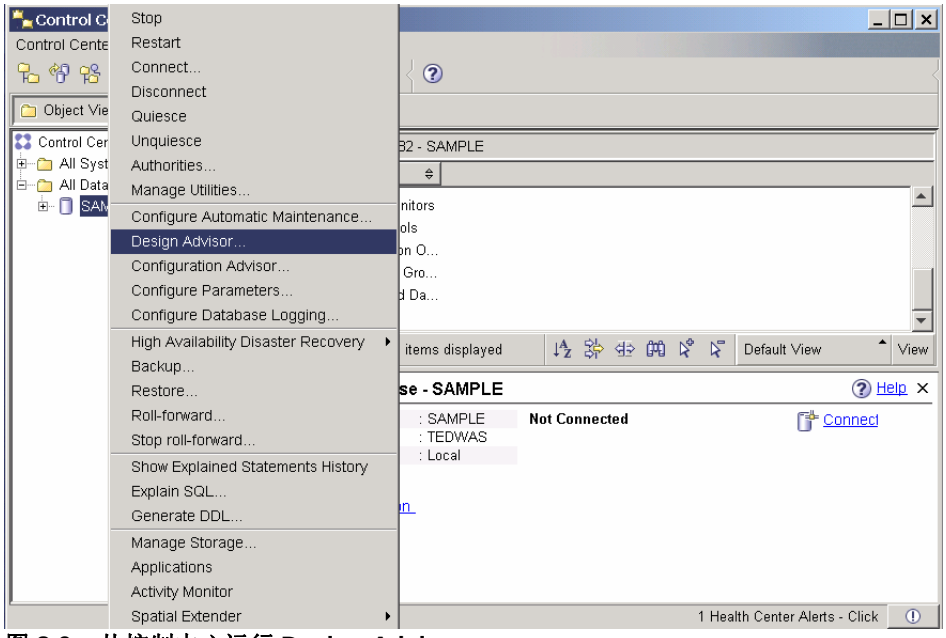


图 8.6 – 从控制中心运行 Design Advisor

图 8.7 展示了 Design Advisor。在 DB2 中，跟随向导获得设计建议。

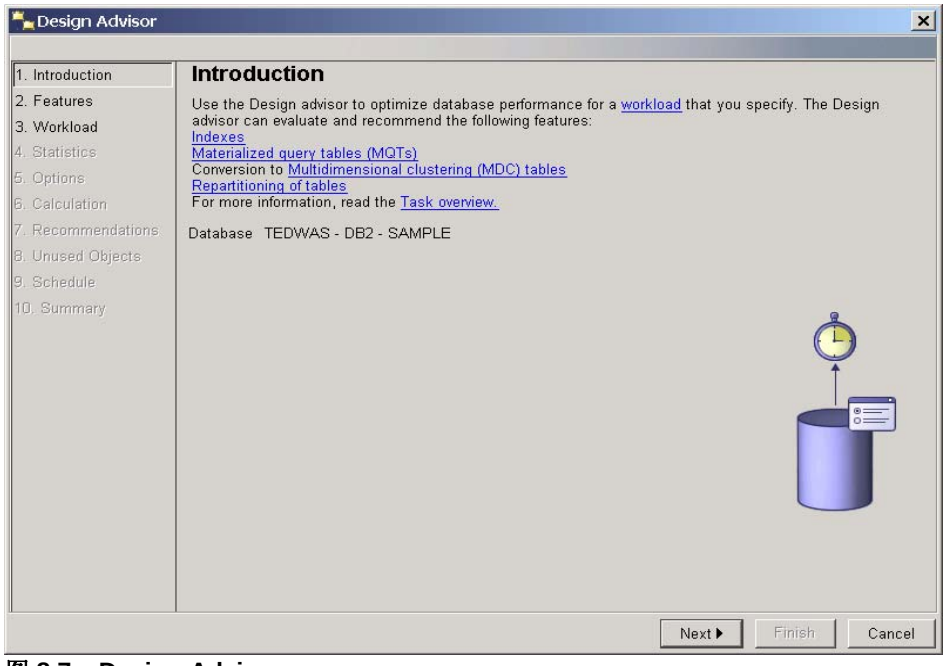


图 8.7 – Design Advisor

8.5 参照完整性

参照完整性允许您管理数据库表之间的关系。例如可以建立如图 8.8 所示的父子表关系。在这个图中，有两个通过部门号相关联的表，DEPARTMENT 和 EMPLOYEE。EMPLOYEE 表的 WORKDEPT 列仅能包含已经存在于 DEPARTMENT 表中的部门号。这个关系中，DEPARTMENT 是父

表，EMPLOYEE 表是子表或者依赖表。图中也显示了 EMPLOYEE 表的 CREATE TABLE 语句必须包含这种关系的定义。

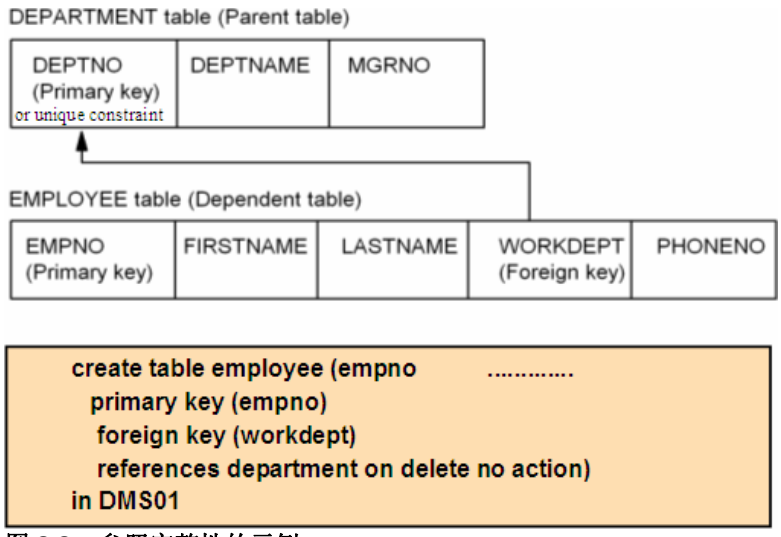


图 8.8 – 参照完整性的示例

关于参照完照性，常用概念如下所示

概念	描述
父表	父键所在的数据表
从属表	由父表中数据决定的表。它包含外键。为了在从属表中存储一行，在父表中必须已经存在相匹配的行.
主键	定义父表中的父键。不可以是空值，并且值必须具有唯一性。一个主键由表中的一列或多列组成。
外键	引用父表的外键

使用参照完整性可以将一个表中的数据与另外的一个或多个表中的数据相关联。为了符合某些属性或事务规则约束也可以对数据值添加约束。例如，如果一个表列存储人物性别，则可添加约束，使此列只可以取值“M”代表男性，和“F”代表女性。

9

第 9 章 – 数据迁移工具

本章介绍的工具或者命令用于在不同情况中的数据迁移，这些情况包括在同一个数据库中的数据迁移，同一个平台上不同数据库的数据迁移以及在不同平台之间的数据迁移。图 9.1 展示了这些数据迁移工具。

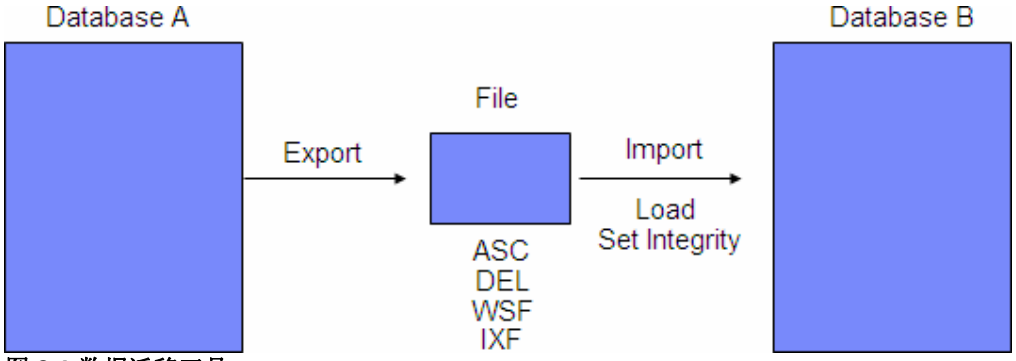


图 9.1 数据迁移工具

图 9.1 中有两个数据库，分别为 A 和 B，使用 Export 工具，可以将数据库中的表导出为文件，导出的文件格式可以为：

ASC = ASCII
DEL = Delimited ASCII
WSF = Worksheet format
IXF = Integrated Exchange Format

ASC 和 DEL 格式的文件是文本文件，可以用任何文本编辑器打开。WSF 格式的文件可以将数据迁移到电子表格软件中，例如 Excel，Lotus® 1-2-3。IXF 格式文件包括了数据表的数据描述语言（DDL）和里面的数据。使用 IXF 格式是非常方便的，利用它可以重建数据表，而其他格式则没有办法这么做。

当数据导出到文件后，使用 Import 可以将数据由文件导入到数据表中。如果使用 ASC，DEL 和 WSF 格式的文件作为中间文件，在它们导入之前数据表必须存在。而使用 IXF 格式的文件在导入前不需要存在相应的数据表。将数据导入到表的另外一个方法就是使用 Load 语句。Load 语句比 Import 更加快，因为它不与 DB2 数据引擎交互而直接操作数据数据库页面。然而，使用 Load 的方法导入数据不会检查数据的约束，也不会引发触发器。为了保证所导入数据的一致性，在使用了 Load 之后通常会运行 Set Integrity 命令。

下一节会详细介绍 Export，Import 和 Load。

注意：
更多关于数据迁移工具的信息，请参见以下视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:4262

9.1 导出（EXPORT）工具

Export 用于将数据从数据表中导出到前面介绍那几种格式的文件中。其实，它执行了一个 SQL 的 SELECT 操作。下面的例子将一个 employee 数据表中的 10 行数据导出到 employee.ixf 文件中。

```
EXPORT TO employee.ixf OF IXF
   SELECT * FROM employee
   FETCH FIRST 10 ROWS ONLY
```

我们建议您尝试操作上面的例子。employee 表是 SAMPLE 数据库（在前面的章节中建立）其中的一个表，所以您必须先连接这个数据库。

如果您选择使用 GUI 工具。Export 工具可以在控制中心中通过点击右键来调用，如图 9.2 所示。

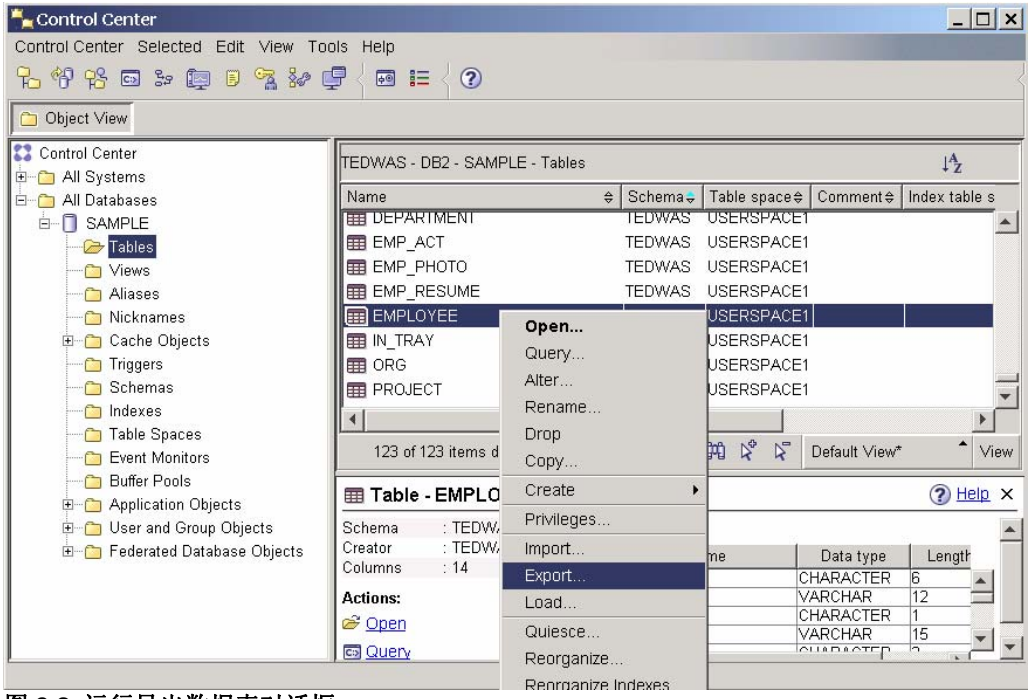


图 9.2 运行导出数据表对话框

如图所示，您首先单击选中 employee 表，然后点击右键，弹出菜单，在里面您 可以看到 Export 选项。点击 Export 选项之后会出现一个向导，根据向导提示操作即可一步步的完成导出操作。

9.2 导入（IMPORT）工具

Import 用于将前面介绍的数据文件导入到数据表中。它其实执行了 SQL 的 INSERT 操作。和 INSERT 操作一样，Import 执行的时候会激活触发器，所有的约束会强制实现，而且会使用数据库的缓冲池。下面的例子将 IXF 格式的 employee.ixf 中所有的数据导入到 employee_copy 数据表中。REPLACE_CREATE 操作是 Import 工具提供的众多参数之一。这个参数表示如果 employee_copy 数据表已经存在，则先清空数据表中的数据然后将 ixf 中的数据导入，如果 employee_copy 数据表不存在，则会先建立该表，然后将数据导入。我们建议您尝试操作上面的例子，但是您必须先像前面一节那样执行 Export 工具将数据导出。

```
IMPORT FROM employee.ixf OF IXF
   REPLACE_CREATE
```

```
    INTO employee_copy
```

如果您选择使用控制中心，您可以选定任意表然后单击右键，选择 Import 选项来调用 Import 工具，如图 9.3 所示

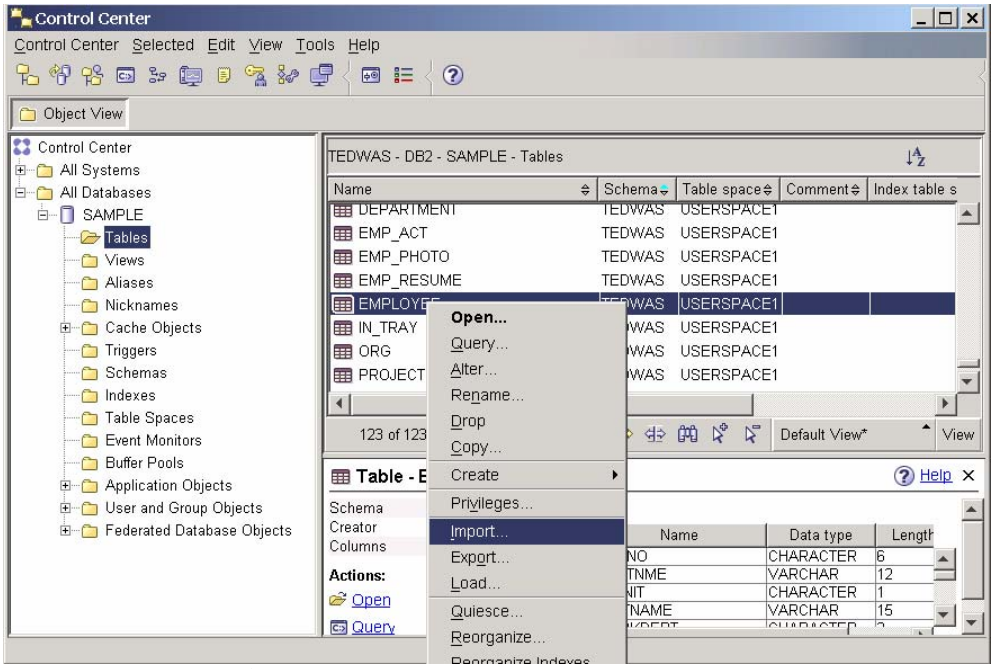


图 9.3 运行导入对话框

9.3 使用 LOAD 来导入

Load 工具可以更快的将数据文件导入到数据表中。正如前面讨论过的那样，Load 工具不会与 DB2 数据引擎发生交互，所以当使用 Load 工具时，不会触发触发器也不会使用缓冲池，而且必须单独实现数据表的约束。Import 工具执行起来比 Load 慢是因为它是低层次的数据操作工具，它分 LOAD，BUILD，DELETE 三个阶段对硬盘上的数据页面来进行直接的处理。

下面的例子将 IXF 格式的 employee.ixf 文件里面的所有数据导入到表 employee_copy。REPLACE 是 LOAD 工具所提供的众多选项之一。它表示将替换 employee_copy 表中的所有数据。

```
LOAD FROM employee.ixf OF IXF
   REPLACE INTO employee_copy
```

执行完上面的命令后，该表进入检查暂挂状态。这时您必须运行 SET INTEGRITY 命令来检查数据的一致性，下面是执行 SET INTEGRITY 的例子：

```
SET INTEGRITY FOR employee_copy
   ALL IMMEDIATE UNCHECKED
```

如果您选择使用控制中心，您可以任意选定一个表, 然后单击右键，依次选择 Load 和 Set Integrity 选项来调用它们，分别如图 9.4 和 9.5 所示。

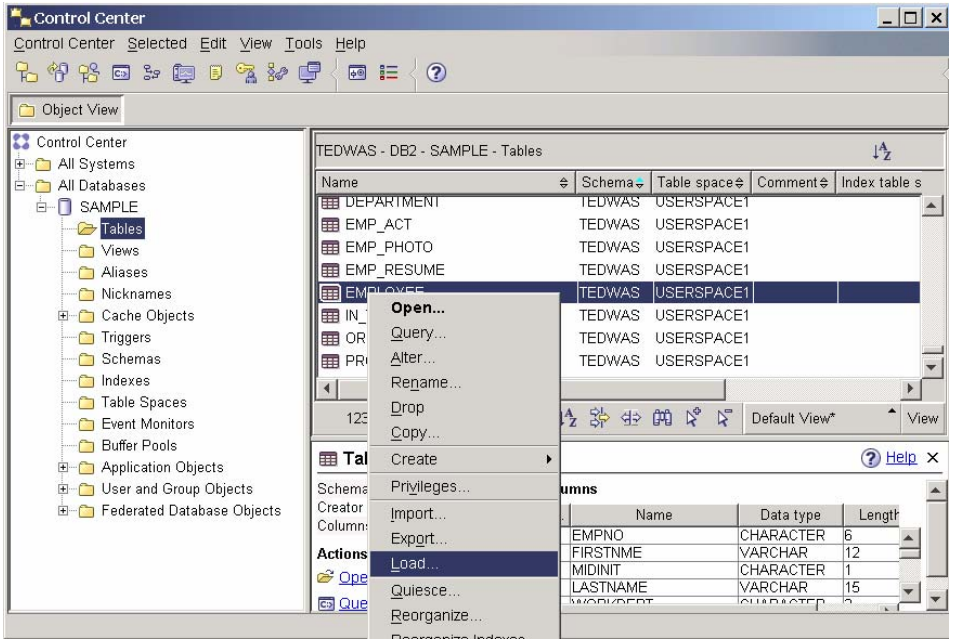


图 9.4 运行 LOAD 向导

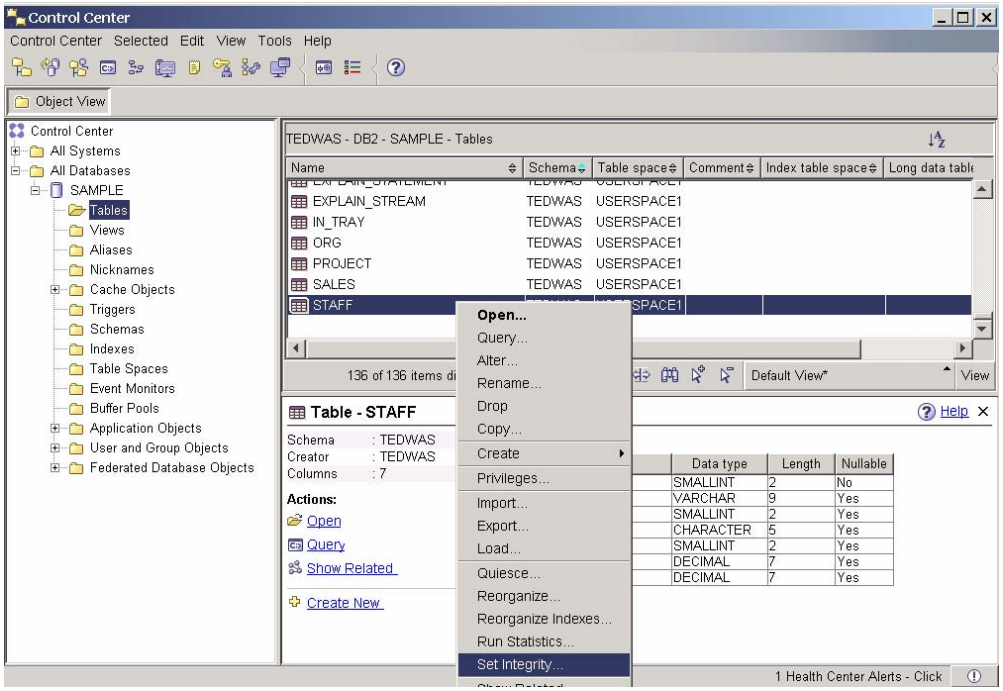


图 9.5 运行 Set Integrity 向导

9.4 db2move 工具

Export，Import 和 Load 每次都只对一个表进行操作。但是您若使用它们来写一段脚本，就可对一个数据库中的所有的表进行操作。另一个工具 db2move 可以更方便地完成同样的工作。db2move 工具只兼容 IXF 格式的文件，而且 IXF 文件的名字由 db2move 自动生成。下面的例子展示怎么样使用 db2move 并结合 export 和 import 选项来对 SAMPLE 数据库进行整体操作。

```
db2move sample export
```

```
db2move sample import
```

db2mov 不可以通过控制中心来调用。

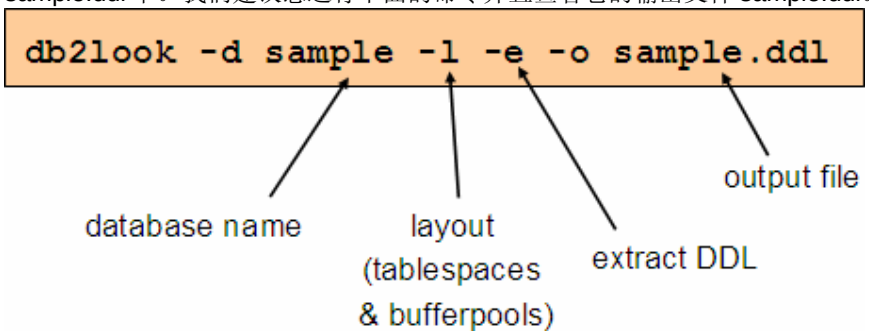
9.5 db2look 工具

由前面可以知道 Export、Import、Load 和 db2move 工具可以在同一个数据库或者跨数据库的数据表之间进行数据的迁移。而 db2look 工具则可以将 DDL 语句、数据库统计状态、表空间参数导出到一个脚本文件中，这个文件可以用于不同系统的数据库。

例如，您想将 Linux 系统上 DB2 服务器的一个数据库克隆到 Windows 平台的 DB2 服务器，第一步，您在 Linux 系统中的 DB2 服务器上运行 db2look 工具，得到数据库的结构并将这个结构存储到脚本文件中，然后将这个脚本文件拷贝到 Windows 平台上的 DB2 服务器上运行，运行后得到一个克隆的数据库。这时，Windows 平台上的这个克隆数据库和 Linux 平台上的源数据库有着相同的结构。第二步，在 DB2 Linux 服务器上运行带 export 选项的 db2move 工具，并将所产生的所有文件拷贝到 DB2 Windows 服务器上，再次运行 db2move 工具，利用 import 或者 load 选项将所有的文件导入到克隆的数据库中，执行完这个步骤后，这两个不同平台上的数据库就是一模一样的了。

当您的工作需要处理不同操作系统上的数据库的时候，您可能会进行上面的步骤。如果数据库服务器运行在同样的平台上，您很可能倾向于使用备份和恢复命令，这两个命令使这个整个数据迁移过程更加简单和直接。备份和恢复系统将会在后面的章节中进行详细的讨论。

下面的例子使用 db2look 工具，将 SAMPLE 数据库的表空间、缓冲池、DDL 语句导出并存储到 sample.ddl 中。我们建议您运行下面的命令并且查看它的输出文件 sample.ddl。



db2look 命令有非常多的选项，本书不能一一列举，您可以通过-h 选项来获得关于所有可用选项的简要介绍。

```
db2look -h
```

db2look 工具也可以在控制中心启动，如图 9.6 所示。

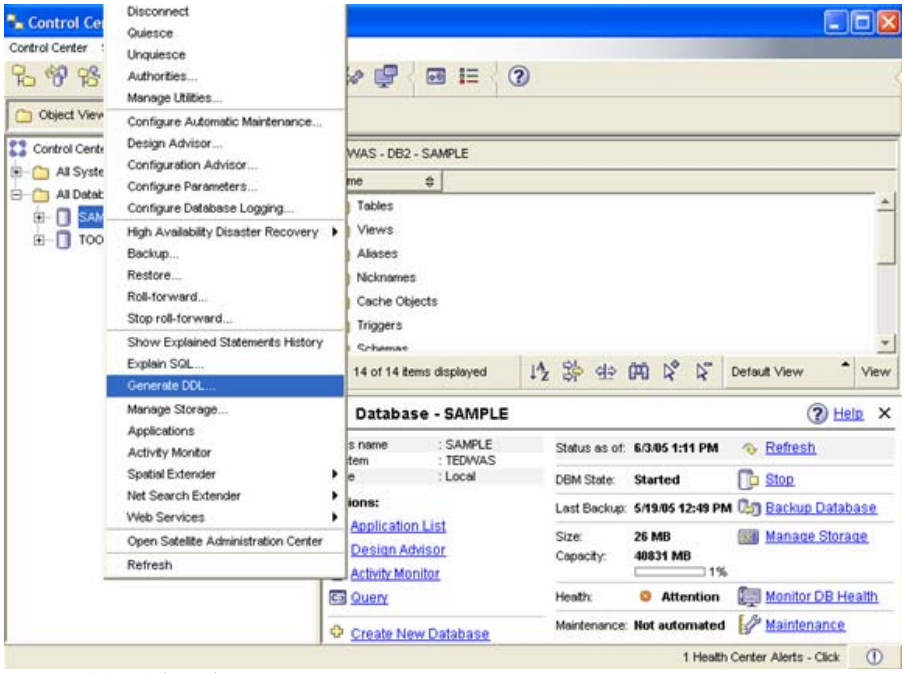


图 9.6 在控制中心中导出 DDL

在图 9.6 中，选择您想要生成 DDL 的数据库，用右键点击它，然后选择“Generate DDL”选项，这时会弹出“Generate DDL”窗口，里面有若干选项，如图 9.7 所示。

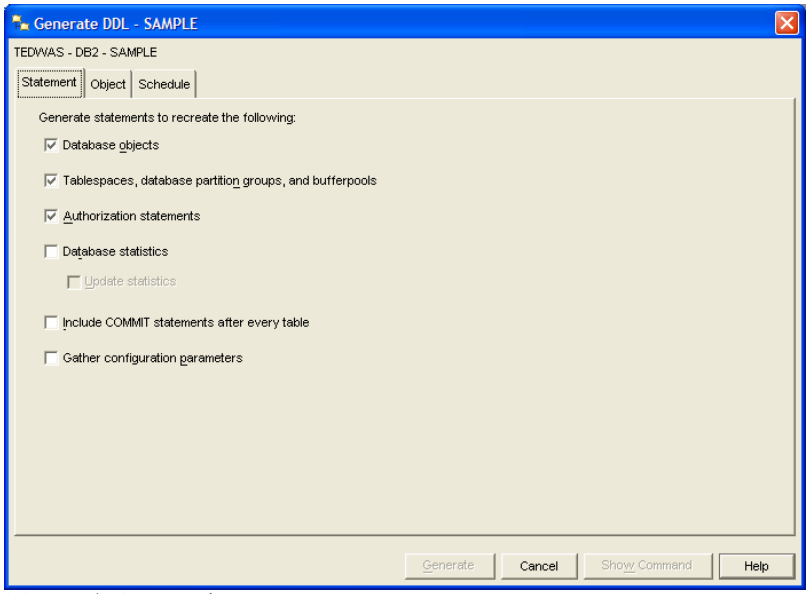


图 9.7 生成 DDL 窗口

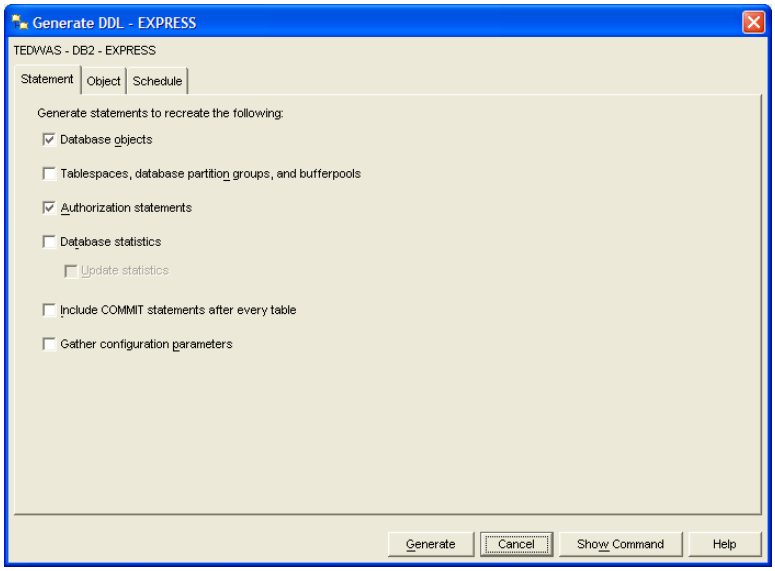
实验 #8 导出 EXPRESS 数据库的 DDL

实验目标：

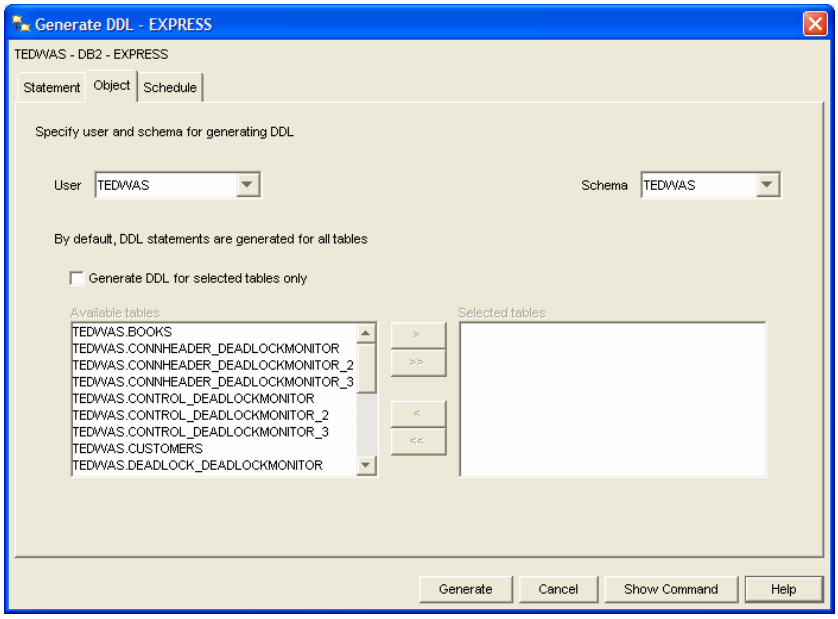
当您克隆一个数据库时，您希望将这个重建数据库的工作能够尽量简单明了和具有重复性。通常，使用 SQL 脚本能够达到这个目的，只要安装好 DB2，SQL 脚本就能够执行来进行一系列的数据库操作。在本实验中，将会使用控制中心来进行这些工作，这里要求您使用控制中心来导出 EXPRESS 数据库（在实验#2 建立）的 DDL。

实验过程：

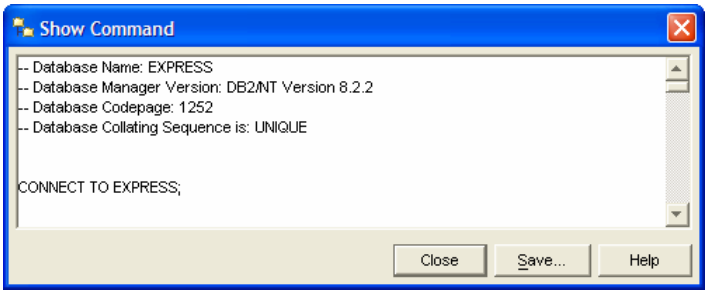
1. 打开控制中心。
2. 在左边窗口的所有数据库中右键单击 EXPRESS 数据库，在弹出的菜单中选择“Generate DDL”项，这会启动“Generate DDL”窗口。
3. “Generate DDL”窗口如下所示，里面是导出 DDL 的选项。如果您在数据库中创建了其它的对象，比如表 空间、缓冲池等等，您可以在这里选择您需要导出的对象。如果您没有创建这些对象，那么请不要选中相应的项。这里没有选中“导出数据库的统计信息”项，因为生产环境中的数据库可能和开发环境中的数据库有不同的统计信息。同样，不同环境下的设置参数也很可能不同，所以也没有选中“Gather configuration parameters”项，如果在您的环境中，所有的设置都和即将发布的设置一样，您可以选中这两个选项。



4. 单击“*Object*”标签，您可以指定要产生 DDL 的对象。在本实验中，选择您创建数据库所使用的用户和模式，然后选择生成这个数据库模式中所有对象的 DDL，最后单击“生成”按钮即可开始生成 DDL。



5. 查看生成的 DDL。上一步的生成结果是一个单独的脚本，里面记录了所选择生成对象的全部 SQL 语句。现在您可以将这个脚本进行逻辑归类。
6. 新建一个文件夹 C:\express，单击“保存”按钮，将文件命名为 schema.ddl 并保存在这个新建的文件夹中。



7. 在命令编辑器中打开这个新保存的文件。（提示：在命令编辑器中选择“文件”，然后点选“打开”，在对话框中选择要打开的文件）。
8. 虽然我们需要的只是数据表的 DDL，但是您可以看到这个文件中包括了所有数据库对象的 DDL。这时，将所有的 CREATE TRIGGER 语句移动到一个新文件中，并将这个新文件命名为 triggers.ddl。这里尽管我们只生成了一个触发器，但是将不同的对象分开是一个最好的工程实践。
9. 现在，我们建议删除下面的语句：
   数据库连接语句 CONNEXT TO
   断开连接语句 DISCONNECT
   这时，您有两个脚本了， 一个保存了所有数据表、视图、索引、约束的 DDL
   `C:\express\schema.ddl`
   一个保存了触发器的 DDL
   `C:\express\triggers.ddl`

10. 整理脚本:
    - 删除不需要的注释，例如 –CONNECT TO…

- 将函数和过程分离到独立的文件中。尤其在有很多函数和过程的时候特别有用。您可以根据他们的功能或者应用将它们分类，例如分成 billing.ddl，math.ddl，stringfunc.ddl 等等

11.您可能会注意到，触发器、函数、过程的结尾都使用了一个特殊的字符@。这是为了区分CREATE <object>语句和在这些对象中的过程性语句。

10

第 10 章 – 数据库安全

本章讨论 DB2 的安全性，从图 10.1 可以大概的看出 DB2 对于安全性的处理。

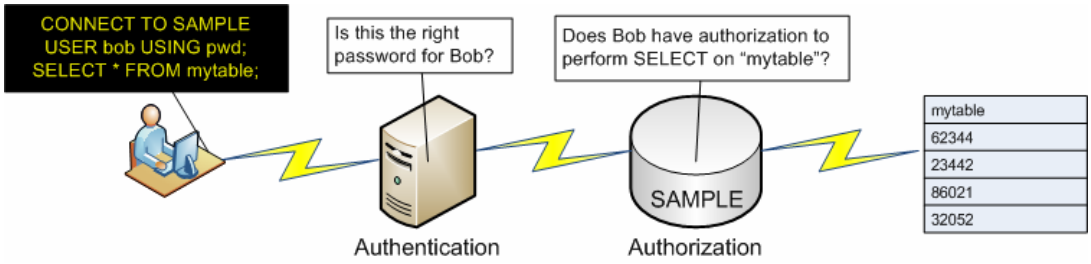


图 10.1 DB2 安全性概览

正如图 10.1 所示，DB2 的安全性由两方面组成：

认证

系统验证用户标识的过程。用户认证是通过 DB2 数据库系统外部的安全性工具来完成的，此安全性工具通常是操作系统的一部分或是一个单独的产品，比如一些网络认证方式或者第三方的认证产品。默认情况下使用的是基于操作系统的用户认证。当使用基于操作系统的用户认证时，先将用户的 id 和密码发送到数据库的服务器（例如，作为连接数据库语句的一部分），然后数据库服务器调用操作系统的用户认证来验证用户 id 和密码的正确性。

授权

在这一阶段，DB2 获取有关已认证的用户的信息，验证该用户是否有权限执行其请求的操作。授权的信息存储在一个 DB2 数据库目录和一个 DBM 设置文件中。

例如，在图 10.1， 用户“bob”使用下面的语句连接到 SAMPLE 数据库

```
CONNECT TO sample USER bob USING pwd
```

用户名“bob”和密码 pwd 会发送到操作系统或者外部的认证设施来执行用户认证，用户认证首先会检验用户名“bob”是否已经定义，然后检测所发送过来的密码是否和这个用户名相对应。如果用户认证通过，操作系统会将安全控制权交给 DB2，接着，当用户“bob”执行一条语句，如：

```
SELECT * FROM mytable
```

此时 DB2 会接管安全控制，对用户进行授权验证，确定用户“bob”是否有权限在数据表 mytable 上执行 SELECT 操作。如果授权验证成功，那么 DB2 就会在 mytable 上执行该操作，否则，DB2 会返回一个出错信息。

> **注意：**
> 更多关于 DB2 安全性的信息，请参见以下视频：
> http://www.channeldb2.com/video/video/show?id=807741:Video:4267

10.1 认证

尽管实际的认证过程是由操作系统或者外部的安全工具来执行，但是 DB2 可以决定在哪里进行用户认证。

DB2 服务器中，DBM CFG 的 AUTHENTICATION 参数能设定为一系列的值，比如当这个值设为 SERVER(默认)时，认证过程由在服务器端的操作系统/外部安全工具来执行，如果这个值设置为 CLIENT 时，认证过程在客户端执行，如图 10.2 所示。

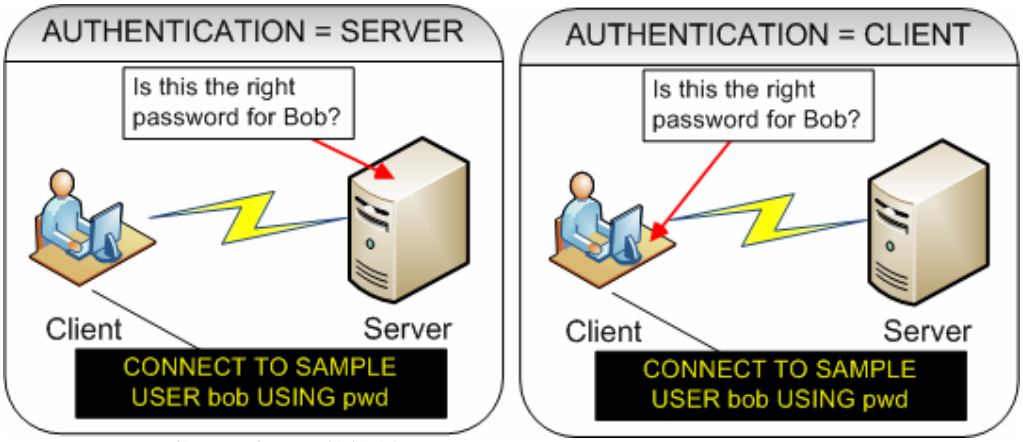


图 10.2 DB2 对执行用户认证的控制

AUTHENTICATION 参数允许值列表在表 10.1 中

Command	Description
SERVER (default)	认证在服务器端执行
CLIENT	认证在客户端执行
SERVER_ENCRYPT	和 SERVER 参数相似，而且用户的 id 和密码都经过加密
KERBEROS	认证使用 Kerberos 安全机制
SQL_AUTHENTICATION_DATAENC	在服务器端进行认证，数据库连接时必须使用数据加密
SQL_AUTHENTICATION_DATAENC_CMP	与上面类似，但当条件不允许的情况下，可以不对数据进行加密
GSSPLUGIN	使用外部的基于 GSS API 插件的安全工具进行认证

表 10.1 AUTHENTICATION 参数的允许值

10.2 授权

授权由权限和特权组成，授权信息存储在 DB2 的系统表中，并由 DB2 来管理。

特权允许用户对数据库执行单一的操作，比如 CREATE，UPDATE，DELETE，INSERT 等等。

权限是预定义的规则，由若干的特权组成，图 10.3 展示了 DB2 的不同特权和权限。

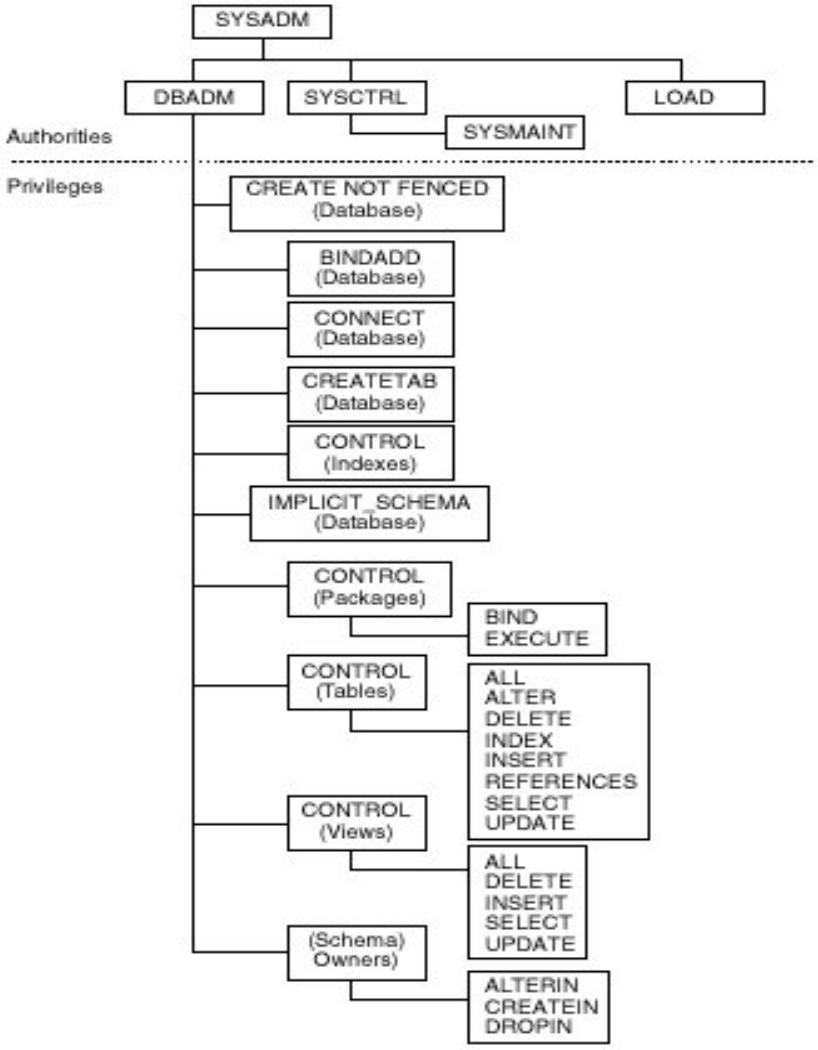

SYSADM
DBADM
SYSCTRL
LOAD
SYSMAINT
Authorities
Privileges
CREATE NOT FENCED (Database)
BINDADD (Database)
CONNECT (Database)
CREATETAB (Database)
CONTROL (Indexes)
IMPLICIT_SCHEMA (Database)
CONTROL (Packages)
BIND
EXECUTE
CONTROL (Tables)
ALL
ALTER
DELETE
INDEX
INSERT
REFERENCES
SELECT
UPDATE
CONTROL (Views)
ALL
DELETE
INSERT
SELECT
UPDATE
(Schema) Owners)
ALTERIN
CREATEIN
DROPIN

图 10.3 权限和特权

表 10.2 展示了不同权限级别所具有的特权。您可以看到，SYSADM 所具有的特权最多，而 SYSMON 所具有的特权最少。

Function	*SYSADM*	*SYSCTRL*	*SYSMAINT*	*SYSMON*	*DBADM*	*LOAD*
Update DBM CFG	**Y**					
Grant/revoke DBADM	**Y**					
Establish/change SYSCTRL	**Y**					
Establish/change SYSMAINT	**Y**					
Establish/change SYSMON	**Y**					
Force users off database	**Y**	**Y**				
Create/drop database	**Y**	**Y**				
Restore to new database	**Y**	**Y**				
Update DB CFG	**Y**	**Y**	**Y**			
Backup database/table space	**Y**	**Y**	**Y**			
Restore to existing database	**Y**	**Y**	**Y**			
Perform roll-forward recovery	**Y**	**Y**	**Y**			
Start/stop instance	**Y**	**Y**	**Y**			
Restore table space	**Y**	**Y**	**Y**			
Run trace	**Y**	**Y**	**Y**	**Y**		
Obtain monitor snapshots	**Y**	**Y**	**Y**			
Query table space state	**Y**	**Y**	**Y**			
Prune log history files	**Y**	**Y**	**Y**			
Quiesce table space	**Y**	**Y**	**Y**		**Y**	**Y**
LOAD tables	**Y**				**Y**	**Y**
Set/unset check pending state	**Y**				**Y**	
Create/drop event monitors	**Y**				**Y**	

表 10.2 DB2 的权限和特权

为了将 SYSADM，SYSCTRL，SYSMAINT 的权限授予一个组，可以在 DBM CFG 中，将 SYSAMD_GROUP，SYSCTRL_GROUP，SYSMAINT_GROUP 赋值为操作系统的用户组。例如，将 SYSADM 权限赋予 db2admns 这个组，您可以输入如下的命令：

```
update dbm cfg using SYSADM_GROUP db2admns
```

每一个 DB2 实例都有自己的权限定义。

在 Windows 系统中，这些参数默认为空，这意味着本地的 Administrators 组具有 SYSADM 的权限级别。在 Linux 系统中，SYSADM 组默认是 owner 组。

10.3 DBADM 权限

DBADM（DataBase ADMinistrator）是数据库的超级用户。它不是实例层次的权限，所以它没有在前一节中列出，要想授予 DBADM 权限，使用 GRANT 语句，如下面的例子所示：

```
connect to sample
```

```
grant DBADM on database to user <userid>
```

在上面的例子中，您首先要连接到数据库，本例中为 sample 数据库，然后，您可以将 DBADM 权限授予用户。DBADM 权限的授予者必须有 SYSADM 权限。

请注意:DBADM 权限不能够建立表空间，尽管表空间是数据库里面的对象。因为表空间与其提供者（磁盘）和缓冲池（内存）相关联，而这些是系统的物理资源。

10.4 PUBLIC 组

DB2 定义了一个内部的组，叫 PUBLIC。操作系统或者网络认证服务定义的用户都隐式的属于 PUBLIC 组。当一个数据库建立时，下面的特权都会自动的授予 PUBLIC 组：

- CONNECT，
- CREATETAB，
- IMPLICIT SCHEMA，
- BINDADD

为了提高安全性，我们建议撤回 PUBLIC 组以下的权限：

```
REVOKE CONNECT              ON DATABASE FROM PUBLIC
REVOKE CREATETAB            ON DATABASE FROM PUBLIC
REVOKE IMPLICIT_SCHEMA      ON DATABASE FROM PUBLIC
REVOKE BINDADD              ON DATABASE FROM PUBLIC
```

10.5 GRANT 和 REVOKE 语句

GRANT 和 REVOKE 语句是标准 SQL 的语句，它们用于授予/撤销用户或者用户组的特权。下面是一些例子：

赋予 USER1 T1 表的 SELECT 特权:

```
GRANT SELECT ON TABLE T1 TO USER user1
```

赋予 GROUP1 对于 T1 表的所有权限:

```
GRANT ALL ON TABLE T1 TO GROUP group1
```

撤销 GROUP1 对于 T1 表的所有权限:

```
REVOKE ALL ON TABLE T1 FROM GROUP group1
```

赋予 USER1 对于存储过程 p1 的 EXECUTE 特权:

```
GRANT EXECUTE ON PROCEDURE p1 TO USER user1
```

撤销 USER1 对于存储过程 p1 的 EXECUTE 特权:

```
REVOKE EXECUTE ON PROCEDURE p1 FROM USER user1
```

10.6 查看授权和特权

查看授权和权限的最简单方法就是通过控制中心来查看。图 10.4 展示了怎样在控制中心里启动数据表 EMPLOYEE 的表特权（Table Privileges）对话框。

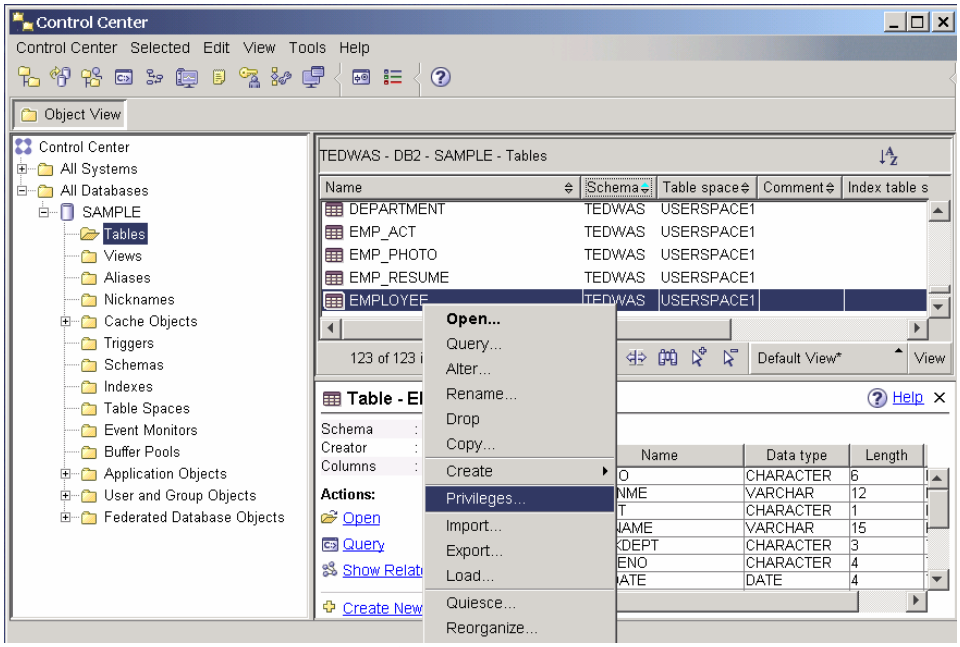


图 10.4 运行 Table Privileges 对话框

如图 10.4 所示，您首先选择目的数据表，右键点击它，然后选择“Privileges”，选择后将会出现表权限 Table Privileges 对话框，如图 10.5 所示，图中的文字对不同区域和窗体部件进行了解释。

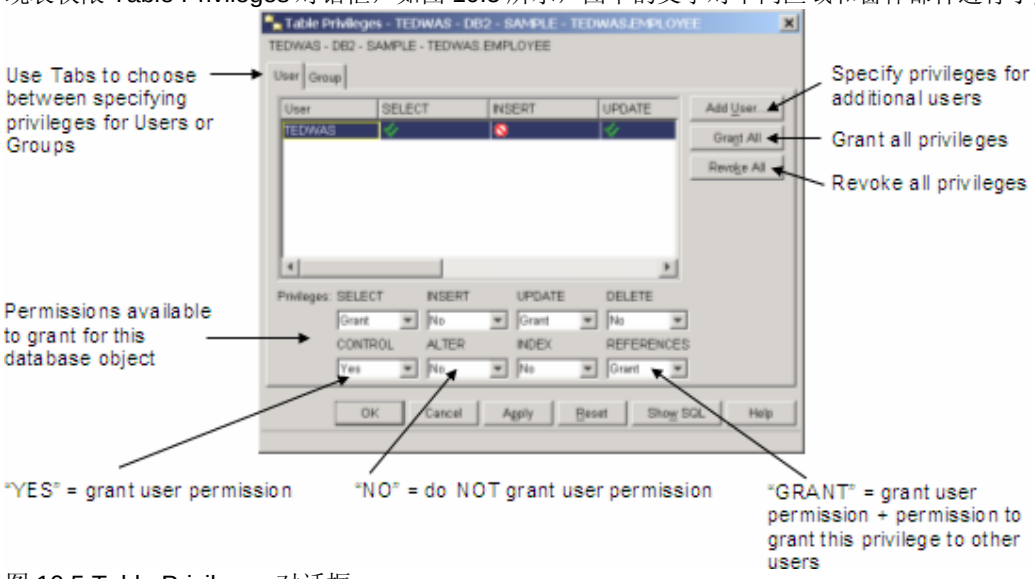


图 10.5 Table Privileges 对话框

另一方面，您可以查询 DB2 SYSCAT 目录视图，它保存了所有授权的信息。例如您想知道用户 DB2ADMIN 是否有表 T2 的 SELECT 权限，如果有，是哪位用户将这个特权授予 DB2ADMIN，您可以像下面例子这样查询：

```
SELECT grantor, grantee, selectauth
  FROM syscat.tabauth
 WHERE tabname = 'T2'

GRANTOR                GRANTEE                SELECTAUTH
```

```
------------------------------------------------
ARFCHONG            DB2ADMIN                    Y
```

在上面的例子中，用户 ARFCHONG 将这个 SELECT 特权授予用户 DB2ADMIN。

10.7 关于组特权

为了更方便管理 DB2，可以将用户放到用户组中，然后向用户组授予所需的特权。

当一个用户组获得特权后，它的所有成员也会隐式地获得这些特权。

当一个用户从用户组中移除后，这个用户会失去从用户组隐式获得的那些特权，但是对于原先显式赋予的特权依然保留。显式地将特权赋予一个用户，则必须显式地将这个特权撤销。

实验 #9 授予和撤销用户的权限

实验目标

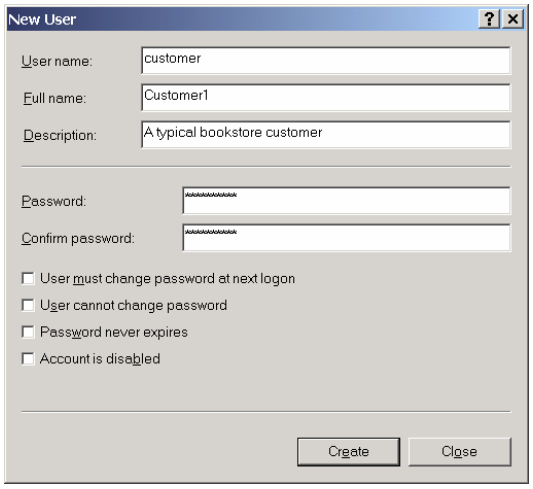
目前为止，您一直都是使用 DB2 的管理员帐户（SYSADM）来运行所有的数据库命令。该管理员帐户拥有对所有的数据库工具、数据、数据库对象的控制权限，所以这个账户必须小心保护好以避免有意或者无意的数据丢失。在大多数情况下，您希望另外建立一个新的帐号或者用户组，并赋予它们有限的权限。在本实验中，您将会建立一个新的用户帐号，然后赋予它特定的一些特权。

实验过程

1. 在桌面上右键单击“我的电脑”图标，在弹出菜单中选择“管理”项，出现“计算机管理”窗口（Windows Computer Management）。

2. 在“计算机管理”的左边窗口的控制树中展开“系统工具”，然后再其下展开“本地用户和组”文件夹，右键单击用户文件夹，然后选择“新用户..”。

3. 在“新用户”对话窗中进行以下步骤：在“用户名”一栏中输入“customer”；在“全名”一栏中输入“Customer1”；在“描述”栏中输入“*A typical bookstore customer*”；在“密码”和“确认密码”栏中输入“ibmdb2”；去掉“用户下次登陆时必须修改密码”前面的钩；最后单击“创建”按钮，这样就可以创建一个新的用户。



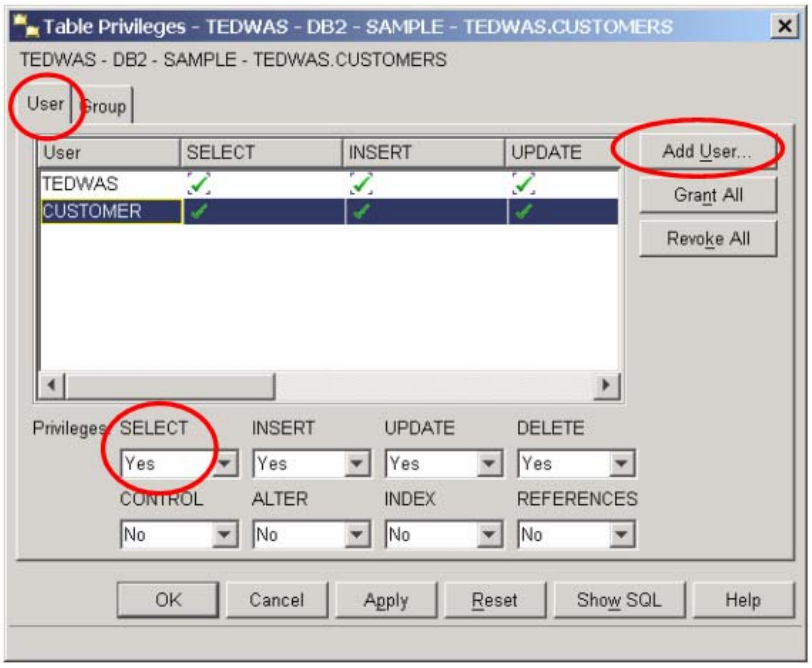
4. 确保 DB2 控制中心使用高级视图。为了转换到高级视图，在控制中心工具 *Tools* 菜单里选择“*Customize Control Center*”，然后设置“Advanced”选项并单击“OK”按键。

5. 将控制中心左边的对象树展开为：*All Databases* > *EXPRESS* > *Tables*。

6. 授予新建用户所需的特权。在 EXPRESS 数据库中右键选择 CUSTOMERS 表，然后在弹出菜单中选择 Pririleges 项，这时会弹出 Table Privileges 对话框。

7. 单击“Add User”按钮，然后选择刚才新建立的“customer”用户。单击“OK”关闭“Add User”对话框。

8. 您会注意到，用户 customer 已经添加到用户列表中，但是并没有分配相关特权。为了将 SELECT、INSERT、UPDATE、DELETE 特权赋予用户，可以在对应的下拉列表中选择 Yes。网上的顾客必须能够浏览、添加、更新、删除与他们账户相关的数据。我们不必给他们其他的权限因为他们并不需要。单击“OK”确定您做的更改并关闭 Table Privileges 对话框。



9. 在 BOOKS 和 SALES 表上重复第 6～8 步。对于 BOOKS 表，只需要赋予 customer 账户 SELECT 权限，因为顾客不能够更改商店的存货信息。对于 SALES 表，只需要赋予 customer 账户 SELECT 和 INSERT 特权而没有 DELETE 和 UPDATE 特权，因为只有商店的雇员才可以修改销售交易信息。

10. 用上面建立的用户 id 连接数据库，尝试从 customer 表选择（SELECT）数据，发现了什么？尝试删除（DELETE）或者更新(UPDATE)SALES 表，又发现了什么？

在本实验，我们只创建了一个用户，在您实际的应用环境中可能需要许多不同类型的用户，请自行实验创建其他的用户并赋予它们不同的特权。同时，您也可以创建用户组，然后赋予整个用户组相应的特权而不是单独给每一个用户赋予特权。

11

第 11 章 – 备份和恢复

本章我们将讨论 DB2 数据库的日志记录、如何使用 BACKUP 功能制作一个数据库的完整或部分拷贝、以及如何使用 RESTORE 功能恢复您的数据。

注意：
更多关于日志记录、备份、恢复的信息，请参见以下视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:4282

11.1 数据库的日志记录

当您使用一个文本编辑器进行编辑工作时，每次都需要点击“save”按钮以确保您的文档被成功保存。而在数据库的世界里，一个“COMMIT”语句正是完成这同样的工作。每次当一个“COMMIT”语句被执行的时候，就保证了所有已经对数据做出的修改都被保存在某个特定位置。

同样的，当您使用文本编辑器时，有时您会看到窗口的右下角显示一条“auto-saving”的短消息。在数据库里，这同样会发生，因为您对数据做出的任何操作，如“UPDATE”，“INSERT”或者“DELETE”，都会在您执行该操作的时候被保存在某个特定位置。

上文提到的所谓“特定位置”是指数据库的日志。数据库日志存储在磁盘上，用来记录事务（transactions）的处理行为。如果发生系统崩溃或者数据库崩溃，日志被用来在恢复过程中恢复崩溃前被提交的事务。

图 11.1 形象地介绍了使用一个采用日志记录的数据库时发生的情况。

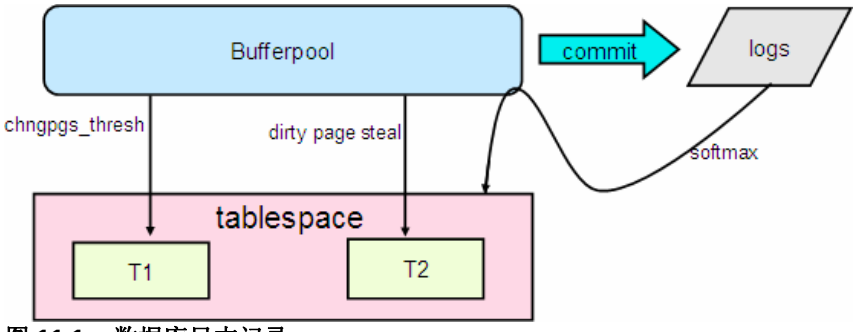


图 11.1 – 数据库日志记录

在图 11.1 中，我们看到一个表空间和数据库的日志。它们都被存储在磁盘上，我们建议它们应该被存储在不同的磁盘上。举例来说，当一个 UPDATE 操作发生的时候，当前被操作的数据行记录将会被移动到缓冲池（位于内存中）。UPDATE 操作将会在缓冲池中执行，而同时该行的旧值和新值会被存储到日志文件中。有时候这个存储是立即进行的，有时候是当一个日志缓冲区满了以后才进行。而当在 UPDATE 操作后立即执行一个 COMMIT 操作时，该数据行的旧值和新值就会被立即存储到日志文件中。这样的过程在数据库的很多其他 SQL 操作上反复发生。只有当碰到某些情况的时候，比如被修改的数据页数目达到了 CHNGPGS_THRES 数值，缓冲池中的这些修改的数据页记录

就会被“具体化”或者被写到表空间磁盘上。参数 CHNGPGS_THRES 表示了缓冲池中所谓“dirty”脏记录的百分比，也就是被修改了的记录的百分比。

从性能的角度来看，为每一个 COMMIT 操作执行两次磁盘写入是毫无意义的：一次写到日志中，另一次写到表空间所在磁盘中；这就是为什么数据“具体化”到表空间所在磁盘的操作仅仅发生在到达阈值参数的时候，如修改的数据页达到 CHNGPGS_THRES 的时候。

11.2 日志的类型

有两种日志类型：

主日志（Primary logs）

主日志是预分配的，可用的数目由 db cfg 的 LOGPRIMARY 参数规定。

辅助日志（Secondary logs）

辅助日志是根据 DB2 的需要动态分配的。辅助日志的最大数目由 db cfg 的 LOGSECOND 参数规定。动态分配日志是很消耗系统资源的；所以，为了保证每日使用的性能，应确保日志都在所分配的主日志范围内。辅助日志文件会在所有到数据库的连接全部关闭后被删除。

如果设定 LOGSECOND 为-1，则不限制记录日志的大小；然而，我们不推荐这样做，因为您的文件系统空间会因此而耗尽。

11.3 日志记录的类型

有两种日志记录类型：循环日志记录（默认）和档案日志记录。

11.3.1 循环日志记录

图 11.2 演示了循环日志记录是如何工作的。

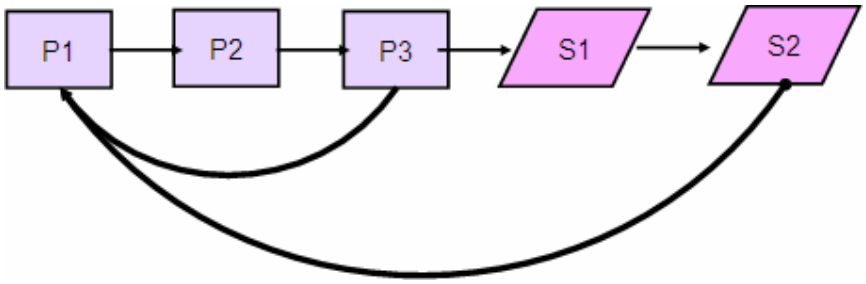


图 11.2 - 主日志和辅助日志的使用情况

在图 11.2 中，我们可以假设 LOGPRIMARY 参数被置为 3，于是图中有三个主日志。为简单起见，我们在这个例子中只讨论仅有一个事务被执行的情况。当这个事务被执行时，开始占用日志文件 P1 的空间，接着是 P2 的空间。如果一个提交操作发生，接着信息被具体化到表空间磁盘中，那么 P1 和 P2 就可以被覆盖了，这是因为崩溃系统恢复（本章的后面将更详细的谈到）已经不再需要这些信息了。另一方面，如果事务很长，以至于它占用了 P1，P2，P3 的空间后，由于该事务既没有被提交也没有被具体化而需要更多的日志空间，那么辅助日志（图中的 S1）就会被动态的分配。如果该事务依然在继续，更多的辅助日志就会被分配，直到 LOGSECOND 所规定的最大辅助日志数目都被分配完毕。这时候如果还需要更多的日志，用户就会看到一个显示日志已满的错误信息，并且当前事务将会被回滚。

11.3.2 档案日志记录和日志保留

在档案日志记录（也被称为日志保留记录）中，日志文件不会被覆盖，而会被在线或离线保存。在线档案日志与系统恢复所需的活动日志保存在一起，离线档案日志被移动到另外一种媒介中，例如磁带，这个操作可以由 USEREXIT 程序完成。要使用档案日志记录，就把 LOGRETAIN 参数设定为 YES。

档案日志记录通常被用在生产系统中，并且由于日志都被保存，使得数据库可以在大多数情况下被恢复到最早的日志状态。有了档案日志记录，DBA（数据库管理员）可以（从一定程度上）使数据库从人为造成的错误中恢复。举例来说，如果一个系统用户不经意的执行了一个错误的事务，几天后，当这个问题被检测到时，DBA 可以使系统恢复到这个问题发生以前的状态。然而，为了保证正确恢复这个事务，可能需要一些手动处理的过程。

在需要向前回滚恢复和在线备份的时候需要档案日志记录。图 11.3 描述了档案文件记录的过程。

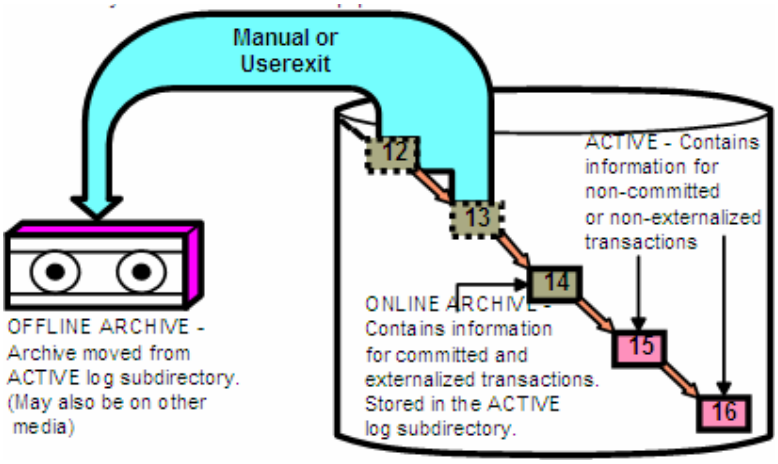


图 11.3 - 档案日志记录

11.4 从控制中心进行数据库日志记录

在控制中心，您可以通过右键单击数据库，选择“配置数据库日志 Configure Database Logging”来配置数据库的日志记录。图 11.4 进行了详细描述。

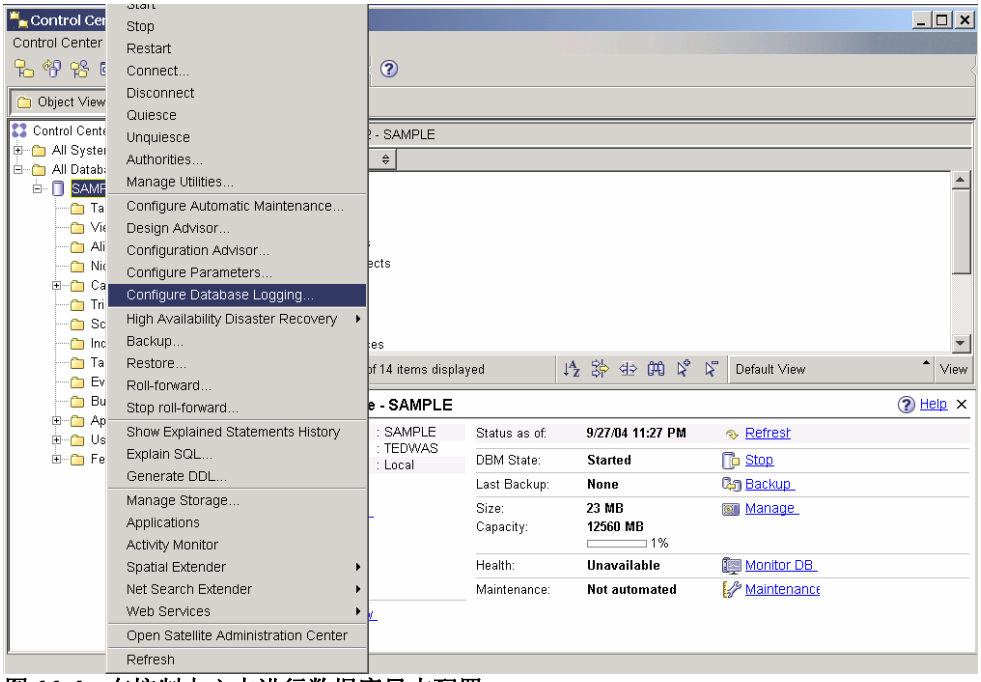


图 11.4 - 在控制中心中进行数据库日志配置

图 11.5 中显示了数据库的日志记录配置向导，在这里您可以选择循环日志记录或者档案日志记录。

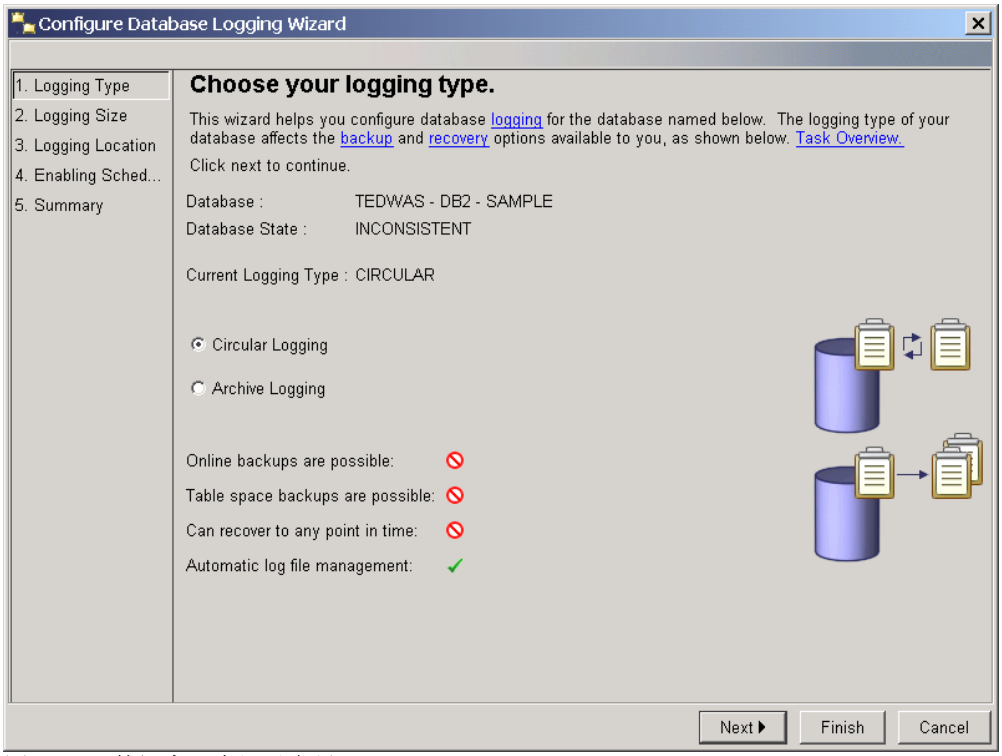


图 11.5 – 数据库日志记录向导

11.5 日志记录的参数

有一些关于日志记录的数据库配置参数。表 11.1 列出了其中主要的参数。

Parameter	Description
logbufsz	在日志记录被写到磁盘之前，被用来给日志记录做缓冲的内存大小
logfilsz	每个设定的日志的大小，以 4KB 页为单位
logprimary	被创建的大小为 logfilsz 的主日志的数目
logsecond	在需要时被创建的用于系统恢复的辅助日志的数目
log-path/newlogpath	活动日志和将来的档案日志被放置的位置
mirrorlogpath	为了保护主日志路径上的日志不受到磁盘错误和意外的删除操作的影响，您可以指定主日志存放点的第二（镜像）路径。
loghead	当前活动日志文件的文件名
userexit	允许 userexit 离线拷贝日志
softmax	限制系统崩溃恢复的开销
logretain	允许档案日志记录模式
overflowlogpath	与 ROLLFORWARD 命令的 OVERFLOW LOG PATH 选项相似；然而，您可以只用设定一次该配置参数，而不用为了每一个 ROLLFORWARD 命令都指定一次 OVERFLOW LOG PATH 选项。
blk_log_dsk_ful	设定该参数以在 DB2 不能够在档案日志路径创建一个新的日志文件时避免磁盘严重错误的产生。取而代之的是，DB2 会尝试着每五分钟创建新日志文件，直到成功。只读的 SQL 就可以继续执行。
max_log	事务的最大活动日志空间所占的百分比
num_log_span	一个活动的工作单元（UOW）的活动日志文件数目

表 11.1 – 日志记录的参数

11.6 数据库备份

DB2 备份命令允许您在该命令执行时获取一个您的数据库的快照拷贝。您执行这条命令的最简单的语法是：

```
BACKUP DATABASE <dbname> [ TO <path> ]
```

大多数命令和工具可以在线或者离线执行。在线意味着其它的用户可能在您执行您的命令时正连接到数据库并执行其他数据库上的操作。离线表示当您执行您的操作时，没有其他的用户连接到数据库。要允许一个在线操作，把关键词 ONLINE 加在命令语法里，否则，默认情况下该命令会假定您在离线执行它。

比如，如果您想要备份 *sample* 数据库到路径 C:\BACKUPS，您可以在 DB2 Window/Linux 命令行解释器中执行这条命令：

```
db2 BACKUP DB sample TO C:\BACKUPS
```

注意，C:\BACKUPS 目录必须在执行这条命令之前就已经存在。同时要确定在您执行上面这条指令时没有其他到数据库的连接，否则您就会收到一条错误信息，因为一次离线的备份不能在有其他连接时执行。

要找出当前是否有其它到数据库的连接，在 DB2 命令行解释器或者 Linux 的 shell 中执行下面命令：

```
db2 list applications
```

要立刻夺取到全部数据库的所有连接，在 DB2 命令行解释器或者 Linux 的 shell 中执行下面命令：

```
db2 force applications all
```

在一个有很多用户的生产环境中，您可能并不想运行这最后一条命令，否则您会收到许多来自您的生气同事们的电话！同时也要注意到，最后一条命令是异步运行的。这意味着当您以后尝试着执行备份命令时，刚才那条命令可能仍然没有起作用。如果您第一次收到一条错误，等待几秒钟，重复备份命令。

在成功执行备份命令之后，一个包含备份好的数据库镜像的文件被生成。该文件的文件名遵循图 11.6 中所示的规则。

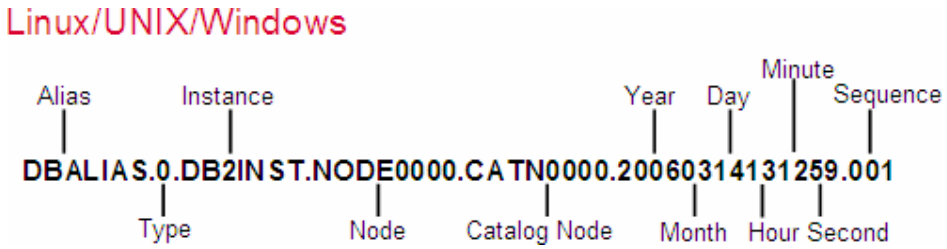


图 11.6 – 备份镜像的命名规则

类型“0”表示这个备份是一个完全备份。类型“3”则表示这仅仅是一个表空间备份。节点被定为 NODE0000，表示没有被分割的数据库，这正是带有 DPF 特性的除 DB2 企业版之外的所有 DB2 版本的情形。目录节点也被定为 CATN0000。您可以参考 DB2 手册获取更多细节。

当获取了几个备份且都存储在同一个路径下的时候，文件名最后的时间标记部分被用来区分这些备份镜像。正如我们将在下一部分看到的，RESTORE 命令可以利用这个时间标记来从其中的特定备份恢复。

实验 #10 – 安排一个备份计划

实验目标

尽管 DB2 可以自动进行一些数据库的维护行为，但有时候有些情况需要您特别定制。在本实验中，您将会为 *EXPRESS* 数据库创建一个定制的每晚备份安排。

实验过程

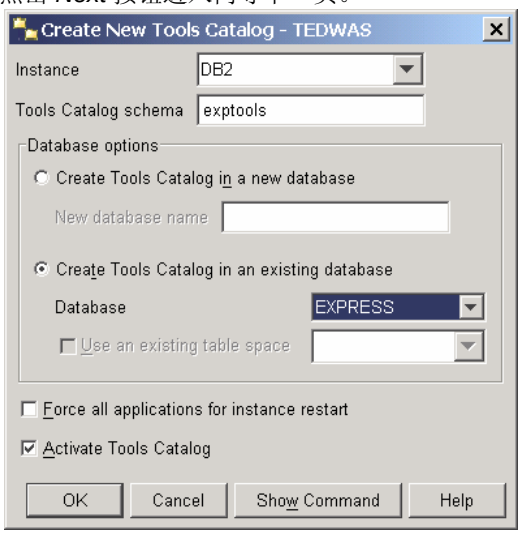
1、在控制中心左边的对象树中找到 Control Center => All Databases。右键单击 *EXPRESS* 数据库并选择 *Backup* 选项。这会启动备份向导 *Backup Wizard*。

2、这个向导的 *Introduction* 页概述了当前数据库的状态，包括最后一次备份的时间和日志记录的方法。点击 *Next* 按钮进入下一页。

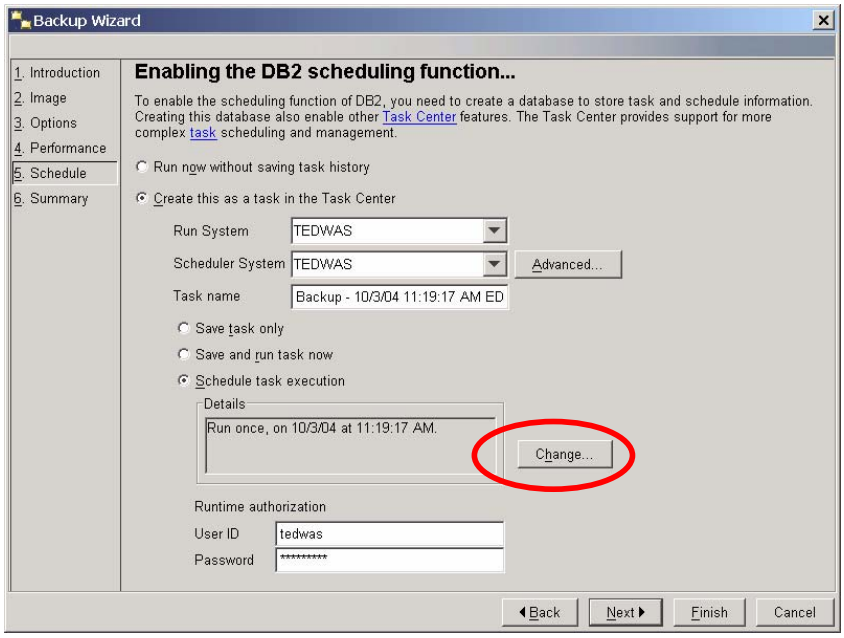
3、在该向导的 *Image* 页，选择镜像文件要存储的目标位置。通常，您应该选择一个与现存数据库所在位置不同的物理驱动器。现在，在文件系统中创建一个新文件夹 `C:\db2backup`，并设定该文件夹为备份位置。在向导中，从媒介类型 *Media Type* 下拉列表里选择文件系统 *File System*。点击 *Add* 按钮，选择您刚刚创建的文件夹，并点击 *OK*。点击 *Next* 进入下一页。

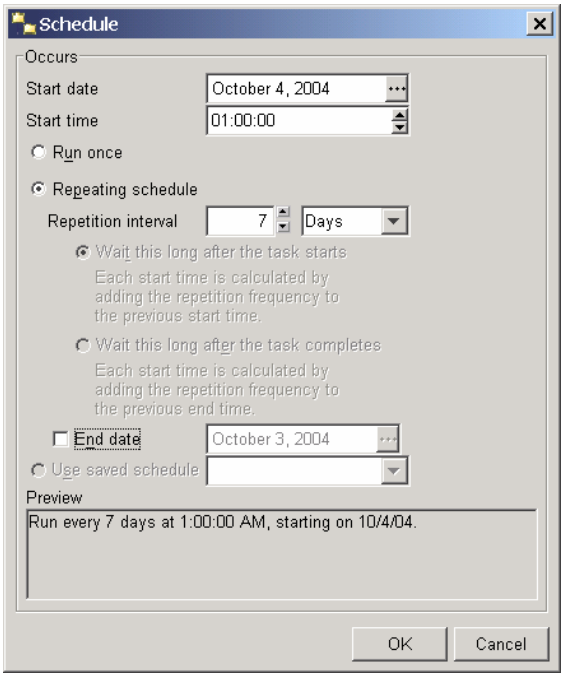
4、您可以仔细研究 *Options* 页和 *Performance* 页，但是默认的选项常常足够了，因为 DB2 自动地以最佳方式进行数据库的备份。在您设置完毕后，请导航至 *Schedule* 页。

5、在计划 *Schedule* 页中，如果计划任务没有被激活，现在就选择激活它。选择要创建工具目录的系统并创建一个新工具目录。给工具目录指定一种模式并且选择在已有的 *EXPRESS* 数据库中创建它。这个工具目录包含有关每个计划任务的元数据。点击 *OK* 按钮继续。当工具目录被创建后，点击 *Next* 按钮进入向导下一页。



6、在 *Schedule* 页上，选择为任务的执行创建一个计划。安排备份每天从凌晨 1 点开始执行。点击 *Next* 按钮进入下一页。





7、在总结页面，您可以检查一下将要被创建的计划任务。一切妥当后，点击 *Finish* 按钮创建这个任务。

8、启动任务中心 Task Center 查看或者修改刚刚创建的备份任务。

11.7 数据库恢复

一次数据库恢复是指从一个备份和（或者）日志中重建您的数据库。如果您刚从一个备份中重建数据库，您所做的就相当于重新创建一个与您备份的时候一模一样的数据库。

如果档案日志记录在备份前被激活，您就不能仅仅是使用一个备份镜像重建数据库，您还要使用日志。正如我们将要在下一部分看到的一样，一个向前回滚的恢复允许您从备份中重建数据库，然后再应用（向前回滚）全部日志进行恢复，或者恢复到某一个特定的时间点。

注意术语“恢复”在这一部分使用很频繁，但是用于恢复的命令却是“RESTORE”。

11.7.1 恢复类型

有三种恢复类型：

- **崩溃或重启动恢复**

  假设您正在使用桌面电脑运行重要的 DB2 数据库事务。突然系统掉电，或者某些人不小心拔去了电源线：那么数据库会发生什么呢？

  下一次您启动电脑并启动 DB2 时，崩溃恢复就会自动执行。在崩溃恢复中，DB2 会自动运行命令 RESTART DATABASE，并且基于活动日志，重新进行或者取消事务。当这条命令完成后，DB2 保证您的数据库将会被维持在与以前一致的状态，也就是说，任何被提交的操作都会被保存，任何没被提交的操作都会被回滚。

- **版本或镜像恢复**

  这种类型的恢复意味着您仅从备份镜像中恢复数据库；所以，您的数据库会被置于与备份的时候相一致的状态。任何在备份后进行的事务都会丢失。

- **向前回滚恢复**

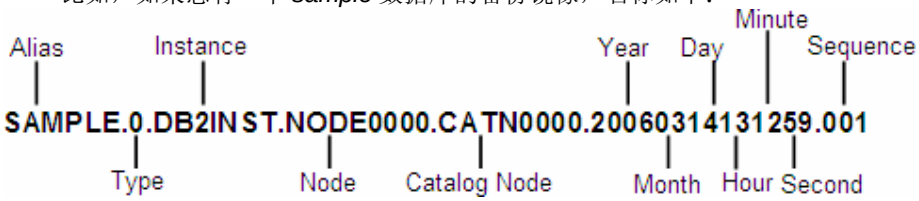
  使用这种类型的恢复，您不仅仅从备份镜像中恢复数据库，同时还将运行 ROLLFORWARD 命令在备份恢复的基础上利用日志来恢复数据，这样一来，您就可以恢复数据库到一个特定的时间点。这种类型的恢复使数据库损失降到最低。

11.7.2 数据库恢复

使用 RESTORE 命令来从一个备份镜像中恢复数据库。以下语法是这个命令的最简单应用：

```
RESTORE DATABASE <dbname> [from <path>] [taken at <timestamp>]
```

比如，如果您有一个 *sample* 数据库的备份镜像，名称如下：



您可以执行下面的命令：

```
RESTORE DB sample FROM <path> TAKEN AT 20060314131259
```

11.8 其他关于备份和恢复的操作

以下列出了使用备份和恢复命令还能做的一些事情。我们建议您查阅 DB2 手册以获取更多信息。

- 在一个 32 位 DB2 上备份数据库，并在一个 64 位 DB2 上重建它
- 恢复并覆盖一个已存数据库
- 使用重定向恢复把数据库恢复到一个与备份镜像中所描述的磁盘数目不同的系统
- 仅备份或者恢复表空间，而不是整个数据库
- 可以进行三角备份或者增量备份；三角备份仅仅记录从上一次备份到下一次备份之间的改动，而增量备份记录所有改动并把它们追加到每一个备份镜像上。
- 从闪存拷贝中备份（需要相关硬件支持）
- 恢复删除的表（如果给定表的该选项被激活）
- 不能从一个操作系统中备份（比如 windows）再到在另一个操作系统（如 Linux）中重建。在这种情况下应使用 db2look 和 db2move 工具。

12

第 12 章 – 维护任务

本章讨论维护数据库的各种任务。一般来说，DB2 会自动执行其中大部分任务的。DB2 Express-C 版本，和其他所有现存 DB2 版本一样，拥有这些自动维护的能力。这种自我管理能力对于不能雇用一个全职数据库管理员来管理数据服务器的中小型公司是十分有好处的。另一方面，假使雇用了数据库管理员，他或她也会因此而获得更多的时间，来处理其他对公司有益的高级活动。

注意：
更多关于数据库维护的信息，请参见以下视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:4302

12.1 重组（REORG）、运行统计（RUNSTATS）、重绑定（REBIND）

DB2 有三个主要的维护任务，在图 12.1 中给出了描述：重组（REORG）、运行统计（RUNSTATS）、重绑定（REBIND）。

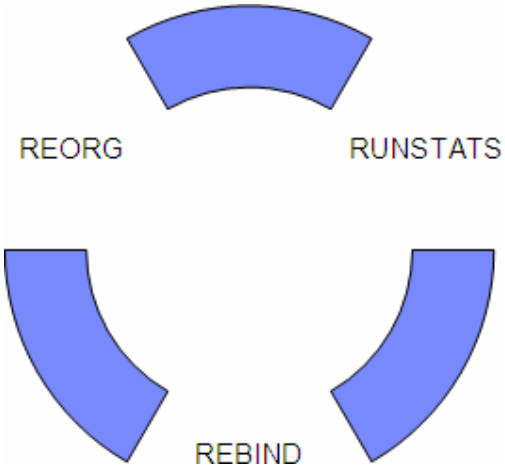


图 12.1 – 维护任务：重组（REORG）、运行统计（RUNSTATS）、重绑定（REBIND）

图 12.1 显示了各种维护任务是以一种循环的方式进行的。如果执行了 REORG，那么建议您接着运行 RUNSTATS 和 REBIND。一段时间之后，数据库中的表会因 UPDATE，DELETE 和 INSERT 等操作被修改。这时候这个循环就会从 REORG 开始重新启动。

12.1.1 重组（REORG）命令

伴随着时间流逝，您不断在数据库上执行 UPDATE，DELETE 和 INSERT 等操作，您的数据在数据库页之间变得越来越支离破碎。REORG 命令回收浪费的空间并重新组织数据，从而获得更高的

运行效率。被频繁修改的表能从 REORG 命令中获得最大的利益。您也可以 R 重组索引，并且 REORG 命令在线或者离线都可以执行。

离线 REORG 命令更快更有效，但不允许访问数据表。而在线的 REORG 命令允许对表的访问，但会消耗大量系统资源，这对于小型的表最为有利。

语法：

```
REORG TABLE <tablename>
```

例子：

```
REORG TABLE employee
```

REORGCHK 命令可以在 REORG 之前使用以检查一个表或者一个索引是否需要被修复。

12.1.2 运行统计（RUNSTATS）命令

DB2 优化器是 DB2 的“大脑”。它为定位或者获取数据找到最有效的路径。优化器是系统的价值所在，它使用存储在目录表中的数据库对象统计信息来最优化数据库的性能。目录表存有关于一个表中当前有多少列，多少行，表有多少个索引，索引是什么类型之类的统计信息。

统计信息不是动态更新的。因为您不会想要 DB2 在每次数据库操作后都更新统计信息，这有可能对整个数据库性能产生不利的影响。取而代之的是，DB2 提供了 RUNSTATS 命令来更新统计信息。保持数据库统计信息的更新十分重要。如果 DB2 优化器认为一个表有一行而不是一百万行，就可能从根本上改变访问路径。如果数据库的统计信息是最新的话，DB2 可以选择一个更好的访问策略。更新统计数据的频率应该由表格中数据的变动频率决定。

语法：

```
RUNSTATS ON TABLE <schema.tablename>
```

例子：

```
RUNSTATS ON TABLE myschema.employee
```

12.1.3 绑定 / 重新绑定

在成功执行 RUNSTATS 命令之后，并不是所有的查询都会使用最新的统计信息。静态的 SQL 访问策略在执行 BIND 命令时确定，因此这时候使用的统计信息有可能并不是和当前一致的。图 12.2 举例说明了这个问题。

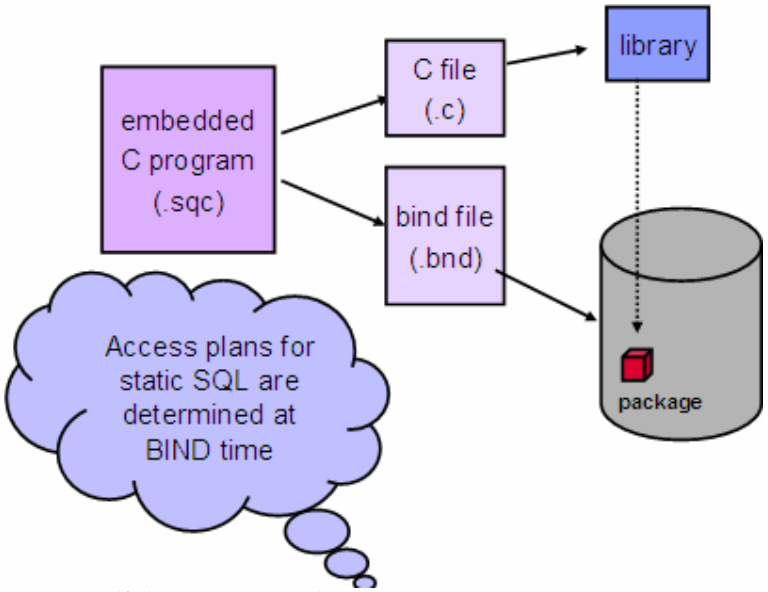


图 12.2 – 静态的 SQL 绑定过程

在图 12.2 中一段嵌入的 C 程序（存储在“sqc”文件类型中）被预编译。预编译后，会产生两个文件，一个“.c”文件包含了 C 代码以及所有的 SQL 注释；一个“.bnd”文件包含了所有的 SQL 语句。扩展名为“.c”的 C 语言文件像以往一样由 C 编译器编译，创建一个如图中右上角所示的“库”。“.bnd”文件进行类似的处理，生成一个存储在数据库中的包。绑定相当于以当时可用的统计信息为基础，用一种最快的访问策略编译 SQL 语句，并把它们存在一个包里。

那么，用个嵌入的 C 程序调用 SQL 把一百万行插入到一个正在使用中的表中时会发生什么呢？在插入之后，若执行了 RUNSTATS 命令，统计信息就会被更新，然而这个绑定的包是不会利用更新的统计信息来重新计算访问路径的。

语法：

```
db2rbind database_alias -l <logfile>
```

例子：

```
db2rbind sample -l mylog.txt
```

要重新绑定 `sample` 数据库里的所有包，并把输出日志存储在一个 `log.txt` 文件中，执行这条命令：

```
db2rbind sample -l mylog.txt
```

12.1.4 在控制中心执行维护工作

在控制中心中您可以执行 REORG 和 RUNSTATS。图 12.3 显示了如何来操作。

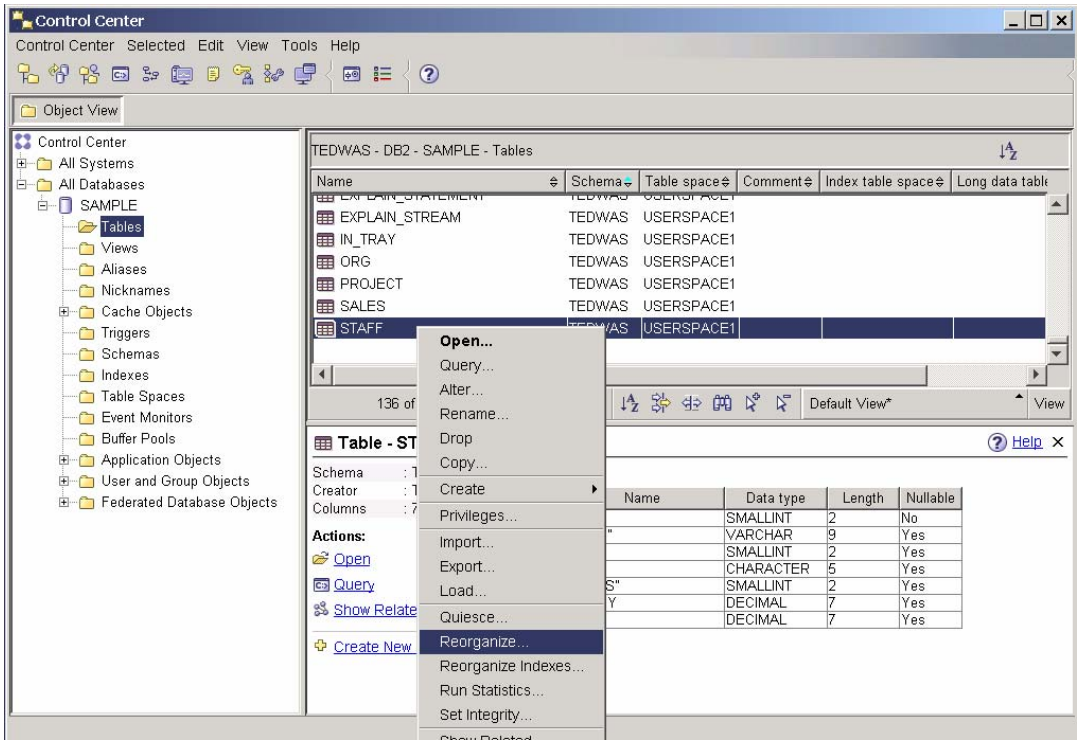


图 12.3 – 控制中心里的 REORG 和 RUNSTATS

选择您要操作的表，右键单击它并选择 Reorganize（即 REORG）或者 Run Statistics（即 RUNSTATS）。

数据库的操作视图

当您选中一个数据库，控制中心右下角的数据库操作视图就会提供有关数据库的信息，比如它的大小，最后一次备份是在什么时候，是否设定了自动维护等等。这个视图允许您快速识别出您的数据库需要的维护。图 12.4 显示了这些信息。

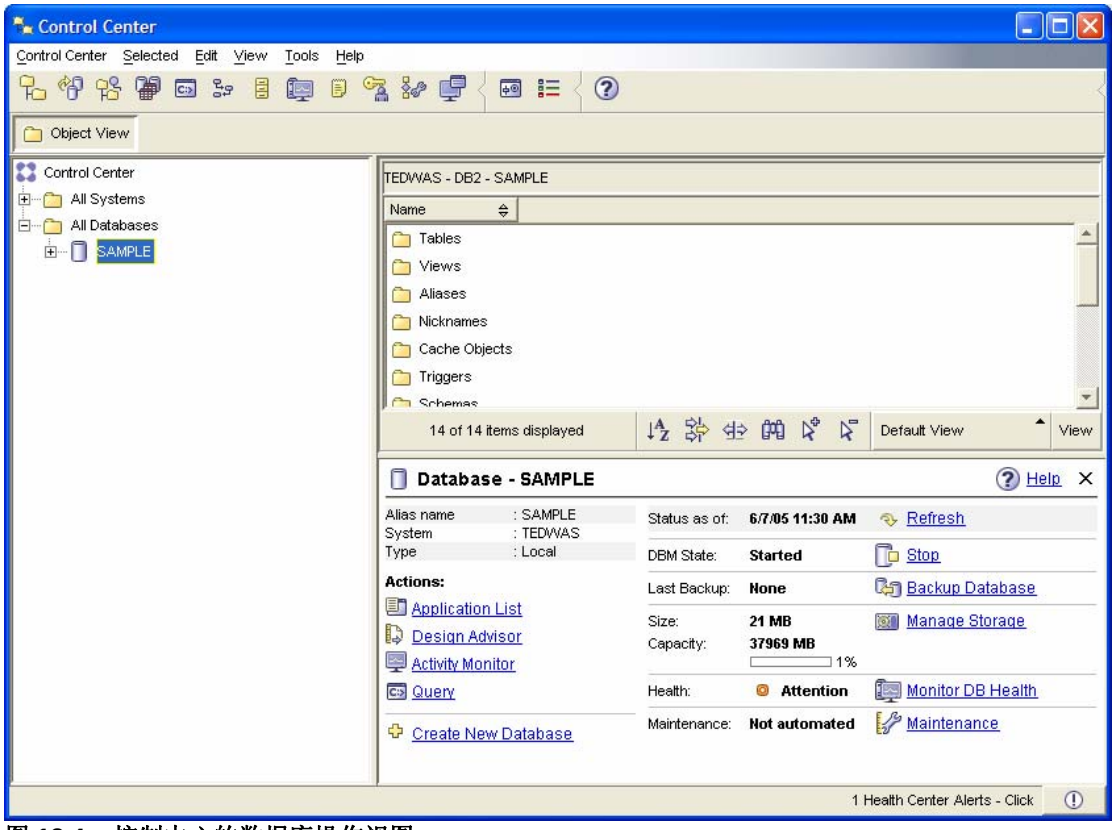


图 12.4 – 控制中心的数据库操作视图

12.2 维护方式

有三种方式进行数据库的维护：

1、手动维护

在需要的时候进行手动维护。

2、创建维护脚本

您可以用维护命令来创建脚本，并让它们按时间表执行。

3、自动维护

让 DB2 自动的为您执行维护(REORG， RUNSTATS， BACKUP) 。

这部分我们主要关注自动维护。

自动维护由以下组成：

- 用户定义一个维护窗口期，在其中可以最小干扰的执行维护任务。比如，如果系统在周日凌晨 2 点到 4 点活动最少，这个时间段就可以用作一个维护窗口期。
- 有两种维护窗口期，一种适用于线上操作，一种适用于线下操作。
  在维护窗口期，仅当有维护需要时 DB2 才自动执行维护操作。

在控制中心中，您可以启动图 12.5 所示的自动维护配置向导（Configure Automated Maintenance Wizard）。

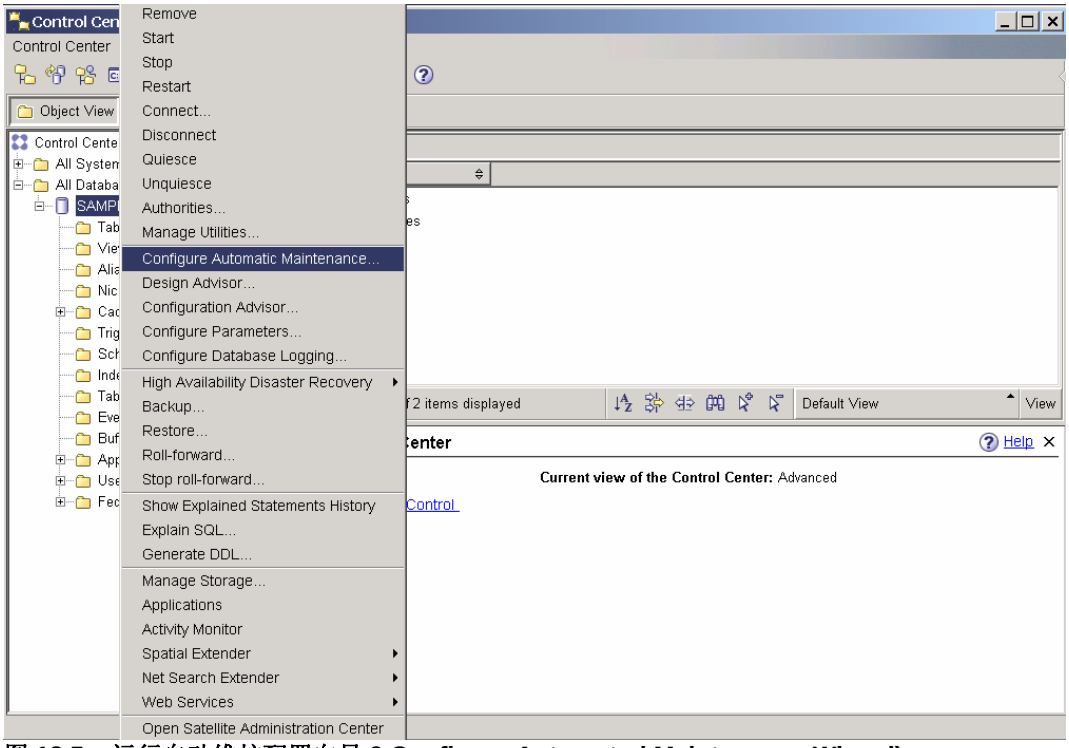


图 12.5 – 运行自动维护配置向导 9 Configure Automated Maintenance Wizard)

图 12.6 展示了自动维护配置向导（Configure Automated Maintenance Wizard）。

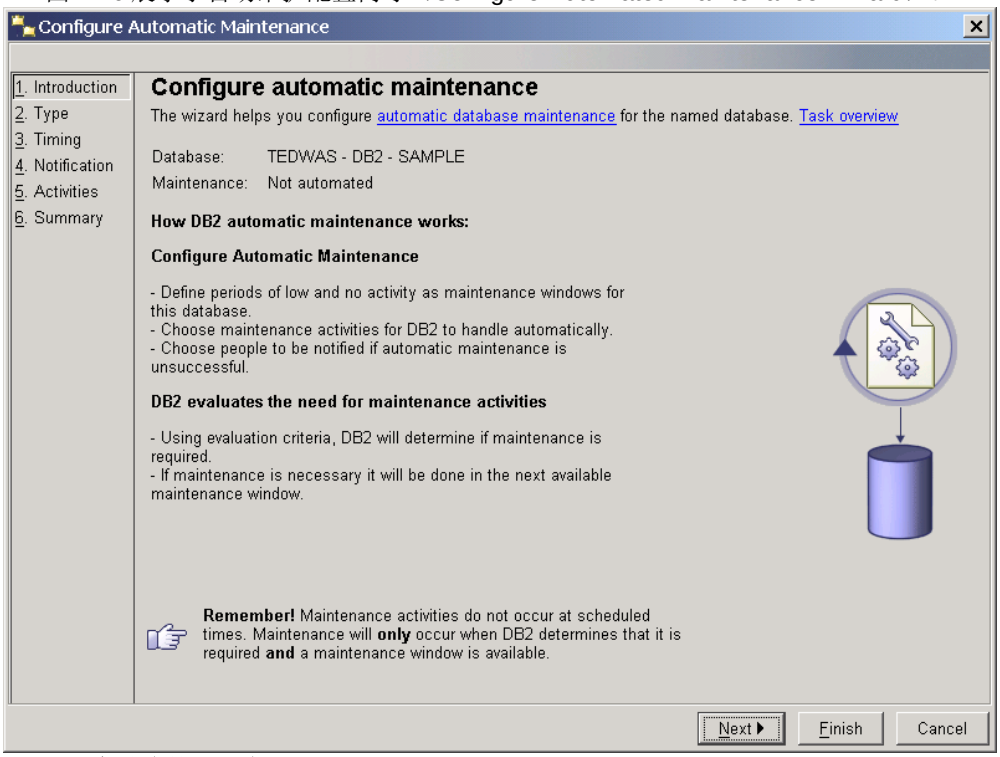


图 12.6 自动维护配置向导（Configure Automated Maintenance Wizard）

实验#11 – 配置自动维护

实验目标

在本实验中，您将对 DB2 的 sample 数据库配置自动维护。

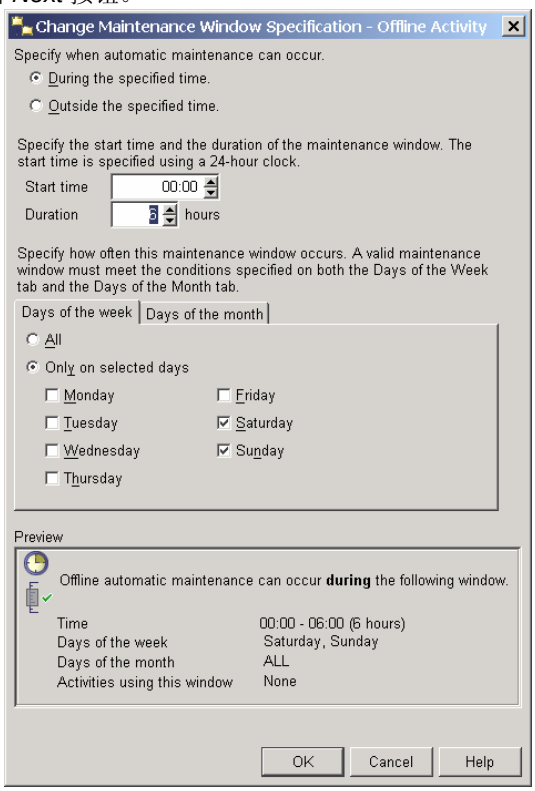
实验过程

1、在控制中心对象树中，右键单击 sample 数据库并选择配置自动维护（*Configure Automatic Maintenance*）菜单项。这将运行自动维护配置向导。

2、向导的起始页（*Introduction*）显示了当前自动维护的设定。如果您创建数据库时使用了自动维护选项，自动维护就已经配置过了。您可以使用该向导重新配置自动维护的选项。点击 *Next* 按钮进入下一页。

3、向导的类型页（*Type*）让您选择关闭所有自动维护还是变更自动维护选项。选择变更自动维护选项。点击 *Next*。

4、时间选择页（*Timing*）让您设定维护窗口期。按照下图设定离线窗口期为每周六和周日的午夜到早上六点。点击离线维护窗口期预览面板旁边的 *Change* 按钮并设定想要的时间。在设定这些必要信息后，点击 *OK* 按钮回到向导。不改变在线窗口期设定（在线维护随时都可以进行）。点击 *Next* 按钮。



5、在向导的通知页（*Notification*），您可以建立一个联系方式，当自动维护失败时会联系您。现在跳过这一步，点击 *Next* 按钮。

6、在向导的活动页（*Activities*），您可以选择某些活动为自动或者某些活动不自动，也可以选择进行某些活动要通报。在这个例子中，请确保所有的自动（*Automate*）选框都被选中，所有的通报选项都被取消。点击 *Next* 按钮。

7、在进入下一页之前，您应该要配置数据库备份的地址。理论上，您应当把备份存储在一个不同的物理磁盘上，防止磁盘错误。在活动页（*Activities*），选择数据库备份选项（*Backup database*），点击配置设定 *Configure Settings* 按钮。

8、在配置设定（*Configure Settings*）对话框的备份标准（*Backup Criteria*）标签，选择平衡数据库恢复和运行性能（*Balance Database Recoverability with Performance*）选项。在备份地址栏，选择当前地址并点击 *Change* 按钮。设定一个不同的地址进行备份（请确保该磁盘有足够空间）。在备份模式（*Backup Mode*）标签，请确保选择了离线备份（*Offline Backup*）。点击 *OK* 按钮关闭备份标准标签。点击 *Next* 按钮。

9、 自动维护配置向导的总结页包含了您的所有选择的总结。点击 *Finish* 按钮同意并执行这些改变。

13

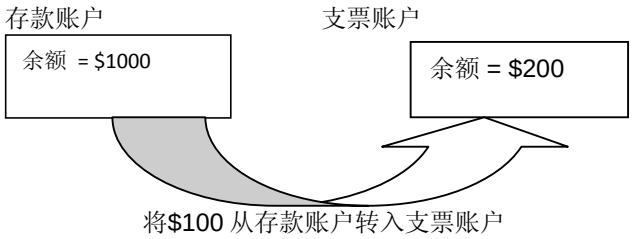
第 13 章 – 并行与锁定

本章讨论如何允许多用户互不干扰地同时访问一个数据库，并且确保他们操作的完整性。我们将会讨论事务、并行和锁定的概念。

注意:
更多关于并行与锁定的信息，请参见以下视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:4322

13.1 事务（Transactions）

一个由若干条 SQL 语句组成的事务或者工作单元在执行时应被视为一个独立的整体；即，如果事务中有一条语句执行出错，整个事务都会失败，在故障点前执行的语句也会回滚。COMMIT 语句是一个事务的结束，也标志着一个新事务的开始。图 13.1 是一个事务的例子。



- 从存款账户取出$100
- 向支票账户存入$100

图 13.1 – 一个事务的例子

如图 13.1 所示，您想要把 100 元从您的储蓄账户转入您的支票账户。为了完成这项工作，需要以下的事件序列：

从储蓄账户取出$100
向支票账户存入$100

如果不将上面的事件序列看作一个单独的事件单元，想象一下，在把$100 从存款账户取出后，若在存入支票账户前突然断电，会发生什么？您将损失$100!

13.2 并行（Concurrency）

并行意味着多个用户可以在同一时间操作同一个数据库对象。DB2 是一个多用户数据库。它使用一种机制合理协调多个用户对数据库访问，每一个用户不会感觉到其他用户的存在，并能确保数据的完整性与一致性。如图 13.2 所示：

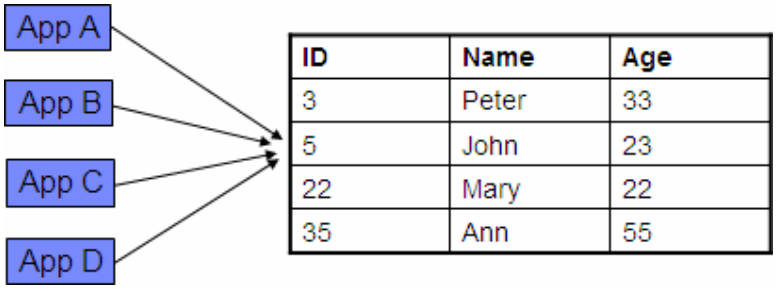


图 13.2 – 一个并行的例子，需要并发控制

在图 13.2 中，有四个应用程序（App A， App B， App C， 和 App D）试图访问表中的同一行（第二行）。如果没有并发控制，所有的程序可能会对同一行进行操作。假设所有的程序都想把第二行中的“Age”项改为不同的值，那么最后一个更新 Age 列的程序似乎将为这种情形下的“胜出者”。很明显地，在本例中，需要某种并发控制来保证结果的一致性。这个并发控制是基于使用锁定的。

锁定与并行的概念是紧密关联的。锁定暂停其它程序的操作直到当前操作完成。一个系统里的锁定越多，并行的可能性也就越小。反之，锁定越少，并行的可能性越大。

锁定会在需要一个事务时被自动获取，在事务终止时被释放（使用一条 COMMIT 或 ROLLBACK 命令）。锁定可以是针对行或列的。锁定有两种：

- 共享锁定（S 锁定）- 当程序企图读并且禁止其它程序修改同一行时被获取的锁定
- 互斥锁定（X 锁定）- 当一个程序修改，插入或删除一行时被获取

现在考虑图 13.3，它与图 13.2 很接近，但它现在展示的是一个锁定。

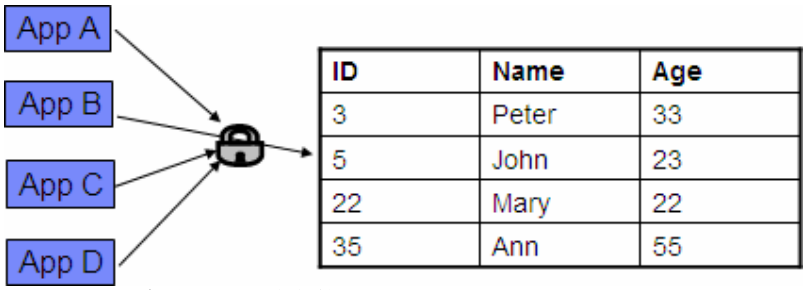


图 13.3 – 一个并行需要锁定的例子

例如在图 13.2 中，如果 App B 最先访问第二行并且要进行修改，App B 就掌握着对该行的 X 锁定。当 App A， App C 和 App D 试图访问同一行时，由于 X 锁定的缘故，它们就无法对其进行修改。这项控制可以保证数据的一致性与完整性。

13.3 无并行控制导致的问题

没有了一些并发控制的格式，可能会遇到以下问题:

- 丢失更新（Lost update）
- 未落实的读（Uncommitted read）
- 不可重复读(Non-repeatable read)
- 幻象（Phantom read）

13.3.1 丢失更新（Lost update）

本节中曾解释了在何种情况下最后进行修改操作的程序会成为“胜出者”，丢失更新的问题与此相似。

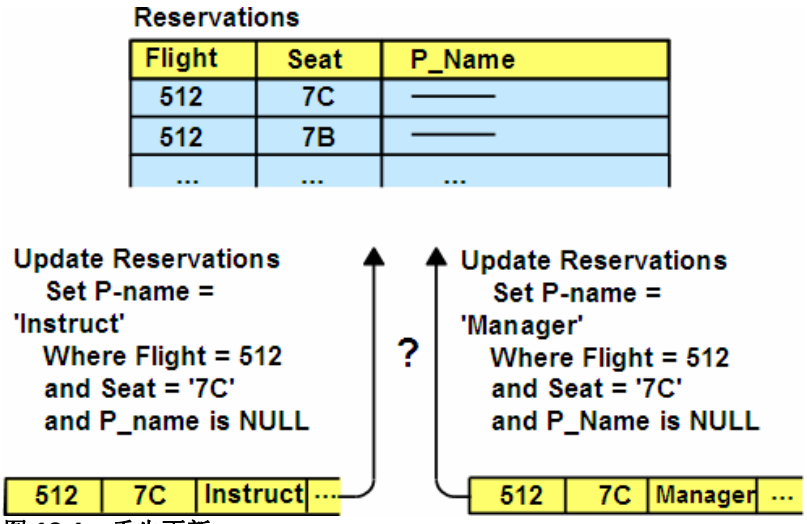


图 13.4 – 丢失更新

在图 13.4 中，有两个程序企图修改同一行。左边的一个是程序 App1， 右边的是程序 App2。事件发生的序列如下：

1、App1 修改一行
2、App2 修改同一行
3、App1 提交
4、App2 提交

App2 进行更新时，App1 的更新就丢失了。所以它们“丢失了更新”。

13.3.2 未落实的读（Uncommitted read）

一个未落实的读，或谓之“脏读”允许一个程序读取未提交的信息，因此读出的数据不保证准确性。

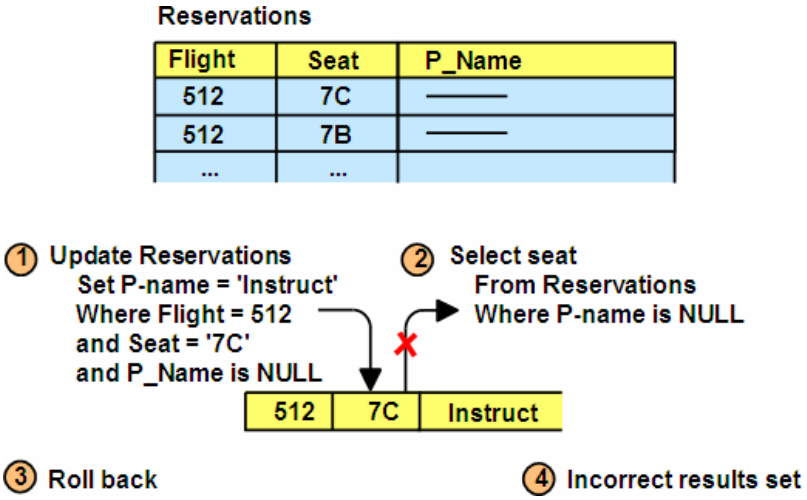


图 13.5 – 未落实的读

图 13.5 依照的是如下的事件序列：

1、App1 修改一行

2、App2 读取该行的新值

3、App1 回滚它对该行的更改

App2 读取的是未提交的数据，所以该数据是无效的，这便是为什么这个问题会被称作“未落实的读”。

13.3.3 不可重复读（Non-repeatable read）

一个不可重复的读意味着您无法在执行完这一读操作这后，再得到同样的结果。

FLIGHT	SEAT	NAME	DESTINATION	ORIGIN
512	7B	———	DENVER	DALLAS
....				
....				
814	8A	———	SAN JOSE	DENVER
....				
134	1C	———	HONOLULU	SAN JOSE
....				....

Figure 13.6 –不可重复读

在图 13.6 中，考虑如果您正试图订购从达拉斯(Dallas)到火奴鲁鲁(Honolulu)的机票，事件的序列是：

1、App1 打开一个游标（亦即结果集）获取您在图 13.6 中看到的内容。

2、App2 删除游标限定的一行（例如目的地（destination）是“San Jose” 一行）。

3、App2 提交更改。

4、App1 关闭并重新开启游标。

在此情况下，因为 App1 在一次重复的读中不会得到同一份的数据，所以它不能够重新产生这个数据集；这就是为何这个问题被称作“不可重复读”。

13.3.4 幻象（Phantom read）

幻象的问题与不可重复读相似，不同之处在于随后的行读取上。您可能会获取额外一些行。图 13.7 展示了这个问题的一个例子：

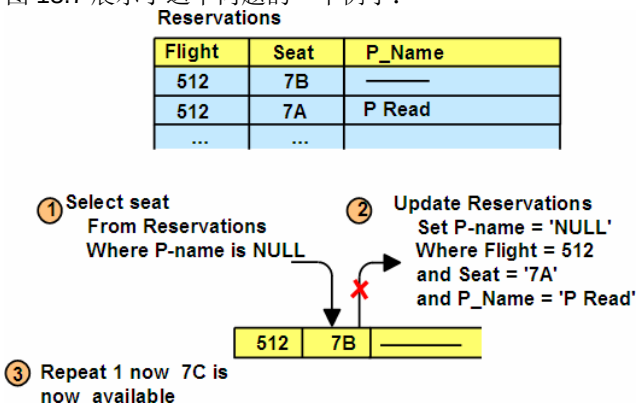


Figure 13.7 –幻象

图 13.7 展示了如下的事件序列：

1、App1 开启一个游标

2、App2 向数据库中添加了一个行，该行与游标匹配

3、App2 提交更改

4、App1 关闭并且重新开启游标

在此情况下，App1 在重复的读取中获取的将不是同一数据，它会得到更多的行。这就是为何此问题被称作“幻象”。

13.4 隔离级别（Isolation Levels）

您可以把隔离级别设想为这样一些锁定政策：有赖于隔离级别的选择，您可能在一个程序中收到不同的数据库锁定行为。

DB2 提供了不同的保护级别来隔离数据:

- 未落实的读（UR）
- 游标稳定性（CS）
- 读稳定性（RS）
- 可重复读（RR）

13.4.1 未落实的读

未落实的读亦称“脏读”。它是最低的隔离级别，并且提供最高的并行性。除非另一个程序企图删除(drop)或者更改(alter)整个表，否则读操作时没有行锁定；而修改（update）操作与游标稳定性级别相同。

此隔离级别仍存在的问题：

- 未落实的读
- 不可重复读
- 幻象

此隔离级别所防止的问题：

- 丢失更新

13.4.2 游标稳定性

游标稳定是默认的隔离级别。它提了低程度的锁定。在这一隔离级别中，游标的“当前”行是锁定的。如果该行只是被读的，锁定会一直持续到一个新行被访问或者该工作单元终止。如果该行被修改，锁定会一直持续到该工作单元终止。

此隔离级别仍存在的的问题:

- 不可重复读
- 幻象

此隔离级别所解决的问题:

- 丢失更新
- 未落实的读

13.4.3 读稳定性

使用读稳定性，在同一个工作单元中的一个程序进程所检索的全部行都会被锁定。对于一个给定的游标，它要锁定所有与结果集匹配的行。例如，如果您有一个含 1000 行的表并且查询返回 10 行，那么只有那 10 行会被锁定。读稳定性使用中等级别的锁定。

此隔离级别仍存在的问题：
- 幻象

引隔离级别所解决的问题：
- 丢失更新
- 未落实的读
- 不可重复的读

13.4.4 可重复读

可重复读是最高的隔离级别。它提供了最大程度的锁定和最少的并行。产生结果集的所有行都会被锁定，也就是说，即使不必出现在最终结果集中的行也会被锁定。在此该工作单元结束前，任何其它的程序都不能修改，删除或插入一个会影响结果集的行。重复读确保程序在一个工作单元中多次进行的同一项查询都返回结果。

此隔离级别仍未解决的问题：
- 无

此隔离级别解决了的问题
- 丢失更新
- 未落实的读
- 不可重复的读
- 幻象

13.4.5 隔离级别对比

图 13.8 比较了对于一次访存，不同隔离级别的作用。如图所示，未落实的读（UR）级别没有锁定。游标稳定性（CS）级别在访问行 1 时将它锁定，但是在访问行 2 时即将此锁定释放，等等。对于读稳定性（RS）级别或是可重复读（RR）级别，任何被访问题的行都将被锁定，并且直到事务结束（一个提交点）时才会被释放。

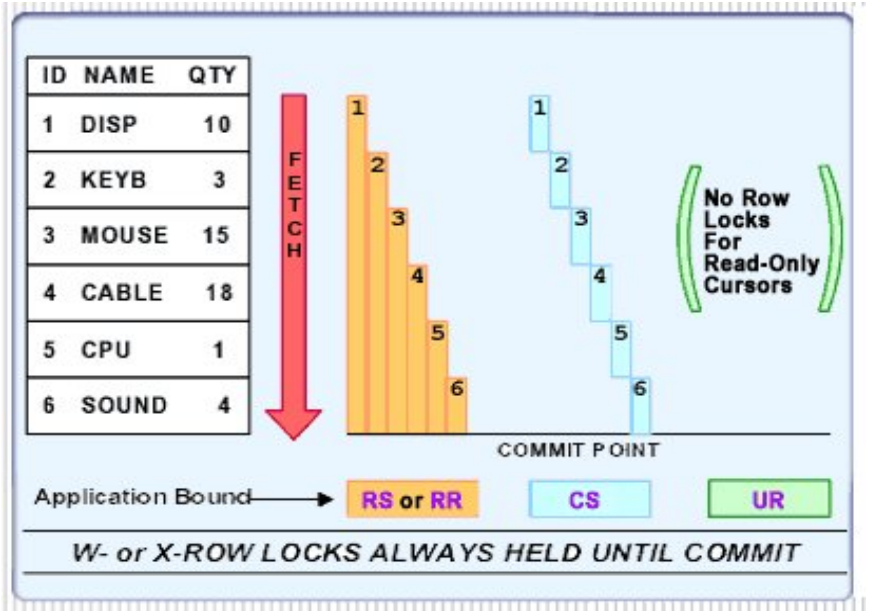


图 13.8 – 隔离级别对比

13.4.6 设定隔离级别

隔离级别可以作用于不同的级别:

- Session (application)会话级（程序级）
- Connection 连接级
- Statement 语句级

隔离级别通常被定义在会话或程序级别。在您的程序中如果没有定义隔离级别，它的默认值为游标稳定性等级。例如，表 13.1 展示了对于.NET 和 JDBC 程序而言可能的隔离级别以及这些设定属性如何匹配 DB2 的隔离级别。

DB2	.NET	JDBC
Uncommitted Read 未落实的读(UR)	ReadUncommitted	TRANSACTION_READ_UNCOMMITTED
Cursor Stability 游标稳定性(CS)	ReadCommitted	TRANSACTION_READ_COMMITTED
Read Stability 读稳定性(RS)	RepeatableRead	TRANSACTION_REPEATABLE_READ
Repeatable Read 可重复读(RR)	Serializable	TRANSACTION_SERIALIZABLE

表达 13.1 – 隔离级别术语对比

语句级隔离级别可以通过 WITH{隔离级别}子句设定。例如：

```
SELECT ... WITH {UR | CS | RS | RR}
```

例如有如下的情形：

一个程序需要粗略地知道表中的行数。性能最为重要。游标稳定性级别要求一条 SQL 异常语句的配合。

```
SELECT COUNT(*) FROM tab1 WITH UR
```

对于嵌入式的 SQL，该级别是在绑定时设定的，对于动态 SQL，该级别是在运行时设定的。

隔离级别的选择取决于您的程序。如果您的程序像例子中那样不需要确切计数，就选用未落实的读（UR）级别。如果您的程序要求严格控制它作用的数据，选择可重复读（RR）级别。

13.5 锁定升级

DB2 做的每次锁定都要消耗一定量的存储空间。当优化器认为一次锁定整个表格比锁定多个行更好时，就出现了锁定升级。

图 13.9 描述了这样情况。

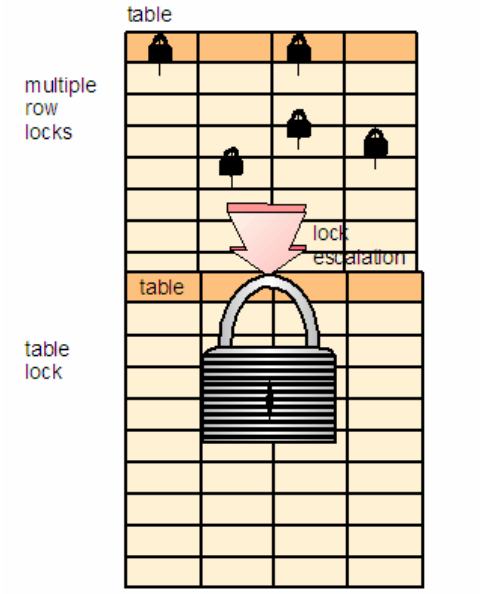


图 13.9 – 锁定升级

有两个主要的数据库配置参数与锁定升级有关：

LOCKLIST – 预留存储空间的数量（以 4k 页为单位），该空间用于管理所有连接的进程的锁定。Windows 中的默认值 50，即 50 个 4K，总共有 50X4K=200K 个页。

MAXLOCKS – 单个程序在整个锁定列表（lock list）中可以占用的最大比例。默认值为 22%。

因此，如果使用默认值，当单个程序请求多于 44K(200 K * 22% = 44K)的空间用于保存锁定，就会发生锁定升级。如果在默认配置下经常发生锁定升级，请增大 LOCKLIST 和 MAXLOCKS 的值。锁定升级会减少并行，因而导致性能的下降。DB2 诊断记录文件（db2diag.log，通常保存在 C:\Program Files\IBM\SQLLIB\DB2 文件夹中）能够用于监测锁定升级的发生情况。

13.6 锁定监视

您可以使用 DB2 程序锁定快照来监视锁定的使用情况。使用如下命令来为锁定开启快照:

```
UPDATE MONITOR SWITCHES USING LOCK ON
```

在这个开关被打开后，监视信息会被收集起来。使用如下命令来获得给定时间的锁定情况报告:

```
GET SNAPSHOT FOR LOCKS FOR APPLICATION AGENTID <handle>
```

图 13.9 展示了一个程序的锁定快照输出的内容。

```
             Application Lock Snapshot

Snapshot timestamp                          = 11-05-2002 00:09:08.672586

Application handle                          = 9
Application ID                              = *LOCAL.DB2.00B9C5050843
Sequence number                             = 0001
Application name                            = db2bp.exe
Authorization ID                            = ADMINISTRATOR
Application status                          = UOW Waiting
Status change time                          = Not Collected
Application code page                       = 1252
Locks held                                  = 4
Total wait time (ms)                        = 0

List Of Locks
 Lock Name                  = 0x05000700048001000000000052
 Lock Attributes            = 0x00000000
 Release Flags              = 0x40000000
 Lock Count                 = 255
 Hold Count                 = 0
 Lock Object Name           = 98308
 Object Type                = Row
 Tablespace Name            = TEST4K
 Table Schema               = ADMINISTRATOR
 Table Name                 = T2
 Mode                       = X
```

图 13.9 – 程序的锁定情况快照

13.7 锁定等待

当两个或以上程序需要对同一个对象进行操作，它们中的一个可能不得不等待以获取它所需要的锁定。默认情况下，程序会无限期地等待。一个程序等待锁定的时间是由数据库配置参数 LOCKTIMEOUT 来决定的。默认值是-1(无限期地等待)。

CURRENT LOCK TIMEOUT 寄存器可以用于设置给定连接的锁定等待时间。这个寄存器默认情况下会被设为 LOCKTIMEOUT 的值。它的值用 SET LOCK TIMEOUTWG 语句来修改。该寄存器的值一旦被设定到一个连接，它会一直持续于整个事务，如：

```
SET LOCK TIMEOUT=WAIT n
```

13.8 死锁的引发与侦测

当连接到同一个数据库的两个以上的进程无限期地等待某一资源时，便可能发生死锁。因为每个进程都保持着另一个进程需要的资源，所以这种等待永远都无法解除。死锁是大多数程序设计中都要解决的问题。图 13.10 描述了一个死锁。

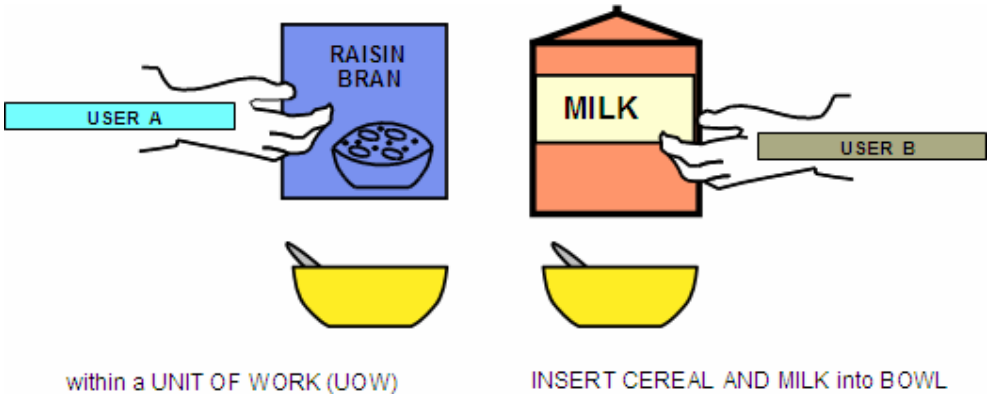


图 13.10 – 死锁的场景

在图 13.10 中，用户 A 拿着葡萄干麦片，直到他拿到牛奶才肯释放。另一方面，用户 B 拿着牛奶，直到他拿到葡萄干麦片才肯释放。于是，我们就有一个死锁的情况。

按以下步骤可以在 DB2 中模拟死锁:

1、打开两个 DB2 命令窗口(我们分别称为”CLP1”和”CLP2”)，代表连接到数据库的两个不同程序。

2、在 CLP1 中执行以下命令:

```
db2 connect to sample
db2 +c update employee set firstnme = 'Mary' where empno = '000050'
```

首先我们连接到了数据库 SAMPLE，在 employee 表对所有”empno = 50000”的行执行了一条更新(update)语句。语句中的”+c”选项表示我们不希望 DB2 命令窗口自动提交语句。我们这样做的目的是为了保持锁定。

3、在 CLP2 中执行如下命令:

```
db2 connect to sample
db2 +c update employee set firstnme = 'Tom' where empno = '000030'
```

在表示第二个程序的 CLP2 窗口中，我们也连接到 SAMPLE 数据库，但是要更新 employee 表中的另一行。

4、在 CLP1 中执行:

```
db2 +c select firstnme from employee where empno = '000030'
```

在键入回车键执行上述 SELECT 语句后，SELECT 仿佛被挂起了。它实际上不是挂起，而是在等待 CLP2 在步骤 3 中对这一行锁定的释放。在此处，如果 LOCKTIMEOUT 被保留为默认值-1，CLP1 进程就会一直等待下去。

5、在 CLP2 中执行:

```
db2 +c select firstnme from employee where empno = '000050'
```

通过执行上述 SELECT 语句，我们现在制造出了一个死锁。这个 SELECT 语句也会仿佛被挂起了，而事实上它是在等待 CLP1 在步骤 2 中对该行锁定的释放。

在上面有死锁场景中，DB2 将会核对数据库的配置参数 DLCHKTIME。这个参数设置的是检查死锁等待的间隔时间。例如，如果这个参数被设为 10 秒，DB2 将每 10 秒检查一次是否存在死锁。如果确实发生了死锁，DB2 会用一个内部算法来决定两个事务中的哪一个应该被回滚，另一个该继续。

如果您正在体验数目众多的死锁，您应该重新检查现存的事务并且检查是否可能重新进行构造。

13.9 并行与锁定的最佳实践：

遵循以下建议有助于最佳地实现可能的并行

1、使事务尽可能地短小。这可以通过在您的程序逻辑允许的情况下频繁地使用 COMMIT 语句（即使对只读的事务也一样）来实现。

2、只有必要时才记录事务信息。

3、快速清理数据：

```
ALTER TABLE ACTIVATE NOT LOGGED INITIALLY WITH EMPTY TABLE
```

4、成批或成组地修改数据，例如：

```
DELETE FROM (
   SELECT * FROM tedwas.t1 WHERE c1 = … FETCH FIRST 3000 ROWS ONLY)
```

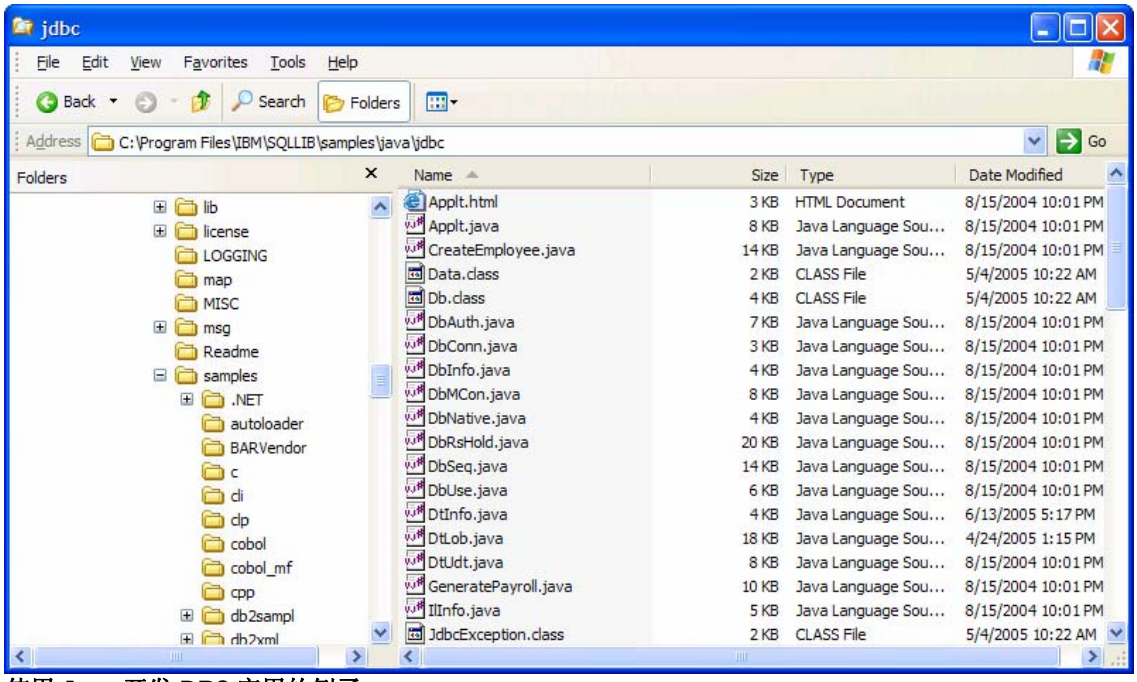
5、利用 DB2 的数据转移工具中的并行特性。

6、设置数据库级的 LOCKTIMEOUT 参数（推荐值为 30~120 秒之间）。不要保留为默认值-1。您也可以使用基于会话的超时锁定。

7、不要检索不需要的数据。例如，在 SELECT 语句中使用 FETCH FIRST n ROWS ONLY 子句。

PART III – DB2 Express-C 应用程序开发

在第三本书部分，我们会深入讨论应用中的数据库对象，如存储过程、用户自定义函数（UDF）、触发器等。请注意，安装完 DB2 数据库服务器后，您可以在安装目录的 SQLLIB\samples 文件夹中找到使用 DB2 开发不同应用的例子。下图展示了 DB2 在 Windows 平台下提供的 Java 开发例子。



使用 Java 开发 DB2 应用的例子

14

第 14 章 –SQL PL 存储过程

本章主要讲述存储过程。存储过程是一个能够封装 SQL 语句和业务逻辑的数据库应用对象。应用程序和数据库的大量交互会产生网络堵塞，而将应用逻辑的一部分保存在数据库中会使这一情况得到相当程度的改善。另外，存储过程提供一个集中的位置存储您的代码，因此其他的应用可以重用相同的程序。

DB2 存储过程可以用以下语言来表达：SQL PL, C/C++, Java, Cobol, CLR(Common Language Runtime)支持的语言, OLE。因为 SQL PL 具有流行性和简单性，在本章主要关注 SQL PL 存储过程。

图 14.1 展示了存储过程是如何工作的。

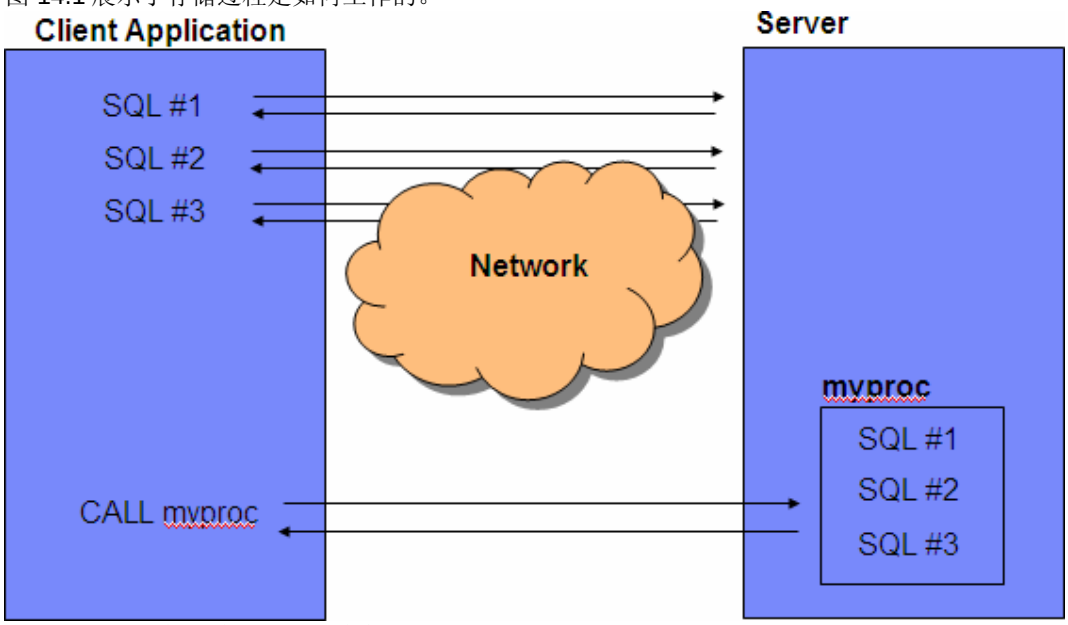


图 14.1 –应用存储过程后网络堵塞减少

在图的左上角，您可以看见几条 SQL 语句顺序执行。每一条 SQL 语句都是从客户端发送至服务器端，然后再由服务器端将结果返回给客户端。如果很多 SQL 语句都按这样的规则执行，网络阻塞可能性就会增加。相反，在图的右下角，您可以看到保存在服务器端，名为“myproc”的存储过程，其内的 SQL 语句与左上角的相同；同时，客户端（在图的左下脚）调用右下角的存储过程的 CALL 语句。执行调用程序的第二种方法更有效率，因为只有一个调用语句和一条返回结果通过网络。

在数据库范围内采用存储过程对于其安全性也很有帮助。例如，您可以限制用户只能通过存储过程访问表和视图；这样可以锁定数据库而防止用户存取无权操作的那部分数据。用户通过存储过程存取数据表或者视图时不需要显式赋予权限，而只需要得到运行存储过程的权限。

注意：
更多关于 SQL PL 存储过程的信息，请参加以下视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:4343

14.1 IBM 数据工作室（Data Studio）

IBM Data Studio 是一个帮助您在数据管理生命周期内开发和管理数据库应用程序的全面解决方案。下面列举了它的一些特征：

- 创建，转换和删除数据库对象
- 浏览、编辑关系数据和 XML
- 可视化创建 SQL、XQuery 语句
- 用 Visual Explain 优化查询
- 开发、调试部署 SQL 和 Java™版的存储过程
- 开发用户定义函数（UDFs）
- 开发 SQLJ 应用程序
- 为 pureXML 应用程序开发查询和常用功能
- 进行数据迁移工作
- 与团队成员合作和分享项目
- 快速建立 SOAP 和 REST Web Services
- 根据物理数据模型（图表）发现数据库对象关系
- 呈现跨表数据分布

IBM Data Studio 是基于 Eclipse 平台的。它是一个单独且免费的部分（不是 DB2 安装程序的一部分）。Data Studio 安装文件可以在 ibm.com/db2/express 页面点“Download”下载。图 14.2 显示了 IBM Data Studio。

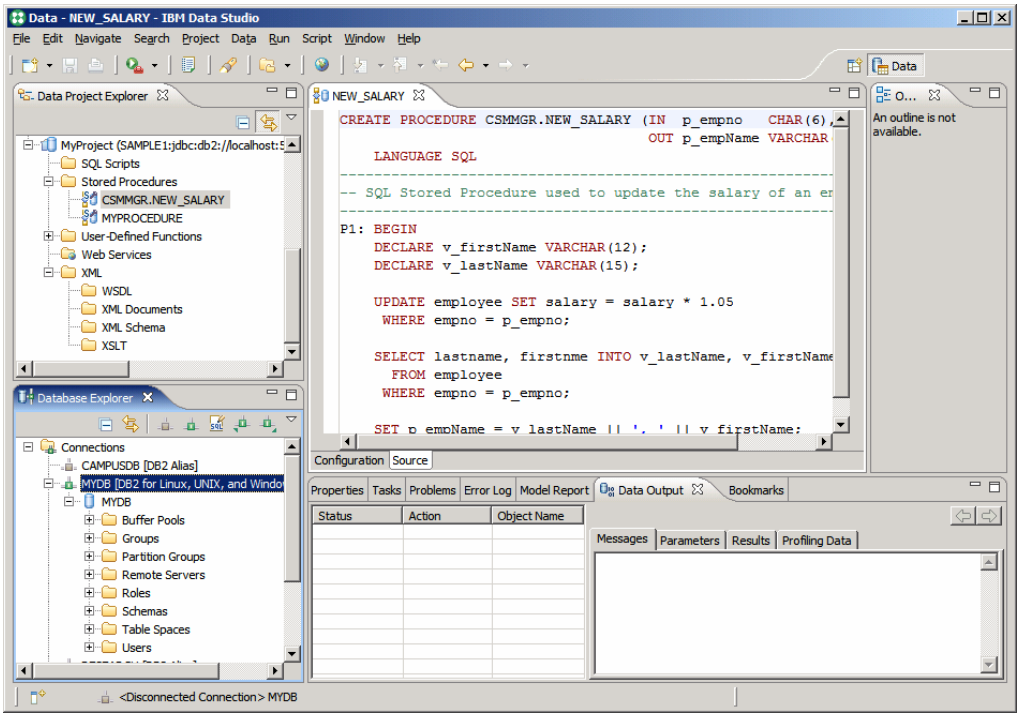


图 14.2 –IBM Data Studio

14.1.2 在 Data Studio 中创建一个存储过程

在 Data Studio 中创建一个 Java 或者 SQL PL 存储过程，可以按如下步骤操作。注意不能从 Data Studio 创建用其它语言编写的存储过程。

第一步：创建一个 Data Studio 项目

在 Data Studio 菜单中，选择 *File -> New -> Project* 然后选择 *Data Development Project*。如图 14.3 所示。

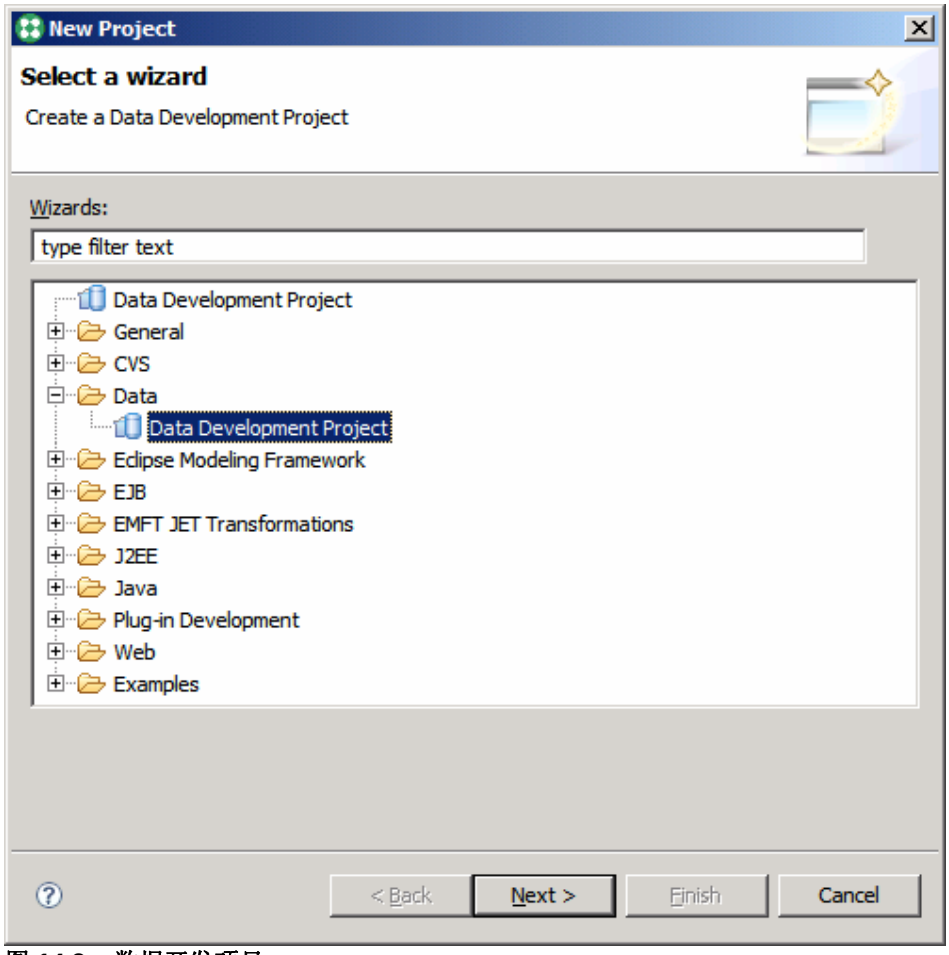


图 14.3 – 数据开发项目

按照向导提示的步骤，输入您的项目名称，指出您想要链接的数据库使之成为您项目的一部分，并且指明 JDK 的目录（通常默认提供的目录就是正确的）。

第二步：创建一个存储过程

当一个项目被创建，数据窗口左侧会显示您的项目。在图 14.4 中您可以看到已创建和展开的“myProject”项目。

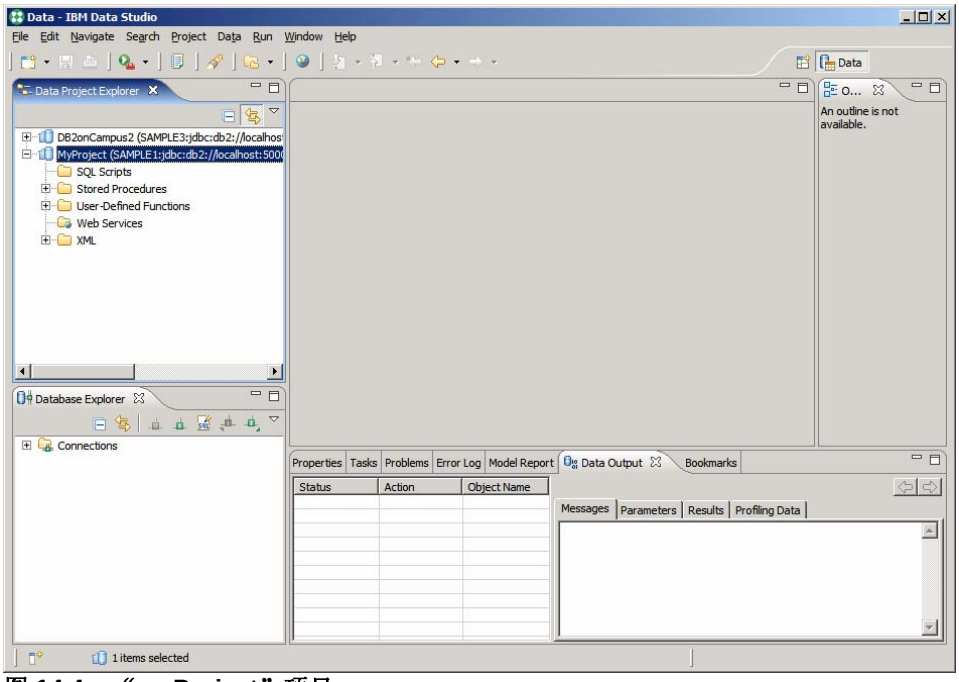


图 14.4 – “myProject”项目

图 14.4 显示了项目的不同文件夹。当您想要创建一个存储过程时，右键点击 Stored Procedures 文件夹选择 *New -> Stored Procedure*，填写所有“新建存储过程（New Stored Procedure）”向导提示的信息，例如相关的项目名称，存储过程的名字和所用的语言以及程序中的 SQL 语句（注意在 IBM Data Studio 只支持用 SQL PL 和 Java 语言编写存储过程）。默认情况下，Data Studio 会给出一个 SQL 语句的例子。这时，可以点击 *Finish*，应用这个模板代码和 SQL 语句就创建了如前面例子中给出的存储过程。如图 14.5 所示。

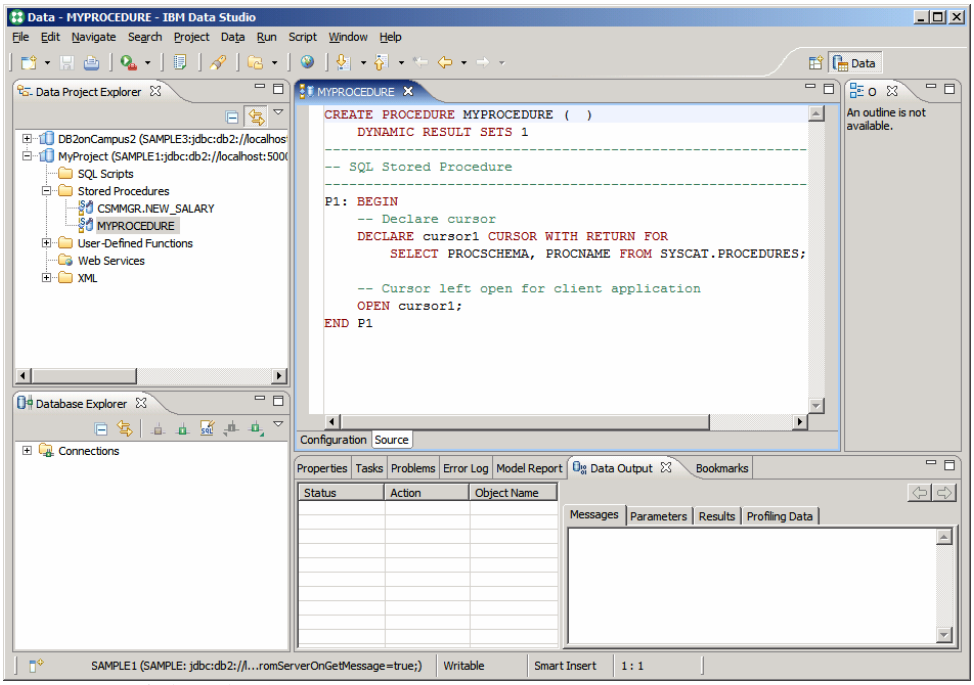


图 14.5 – 一个存储过程的例子

图 14.5 中，“MYPROCEDURE”存储过程的代码是已经创建好了。您可以用自己的存储过程代码来替代它。为方便起见，在这本书中我们把上面这个样例存储过程作为我们自己写的来应用。

第三步：编译（部署）存储过程

存储过程被创建之后，您就可以在左侧的面板上通过右键单击选择“Deploy”来对其加以编译和部署。图 14.6 展示了这一过程。

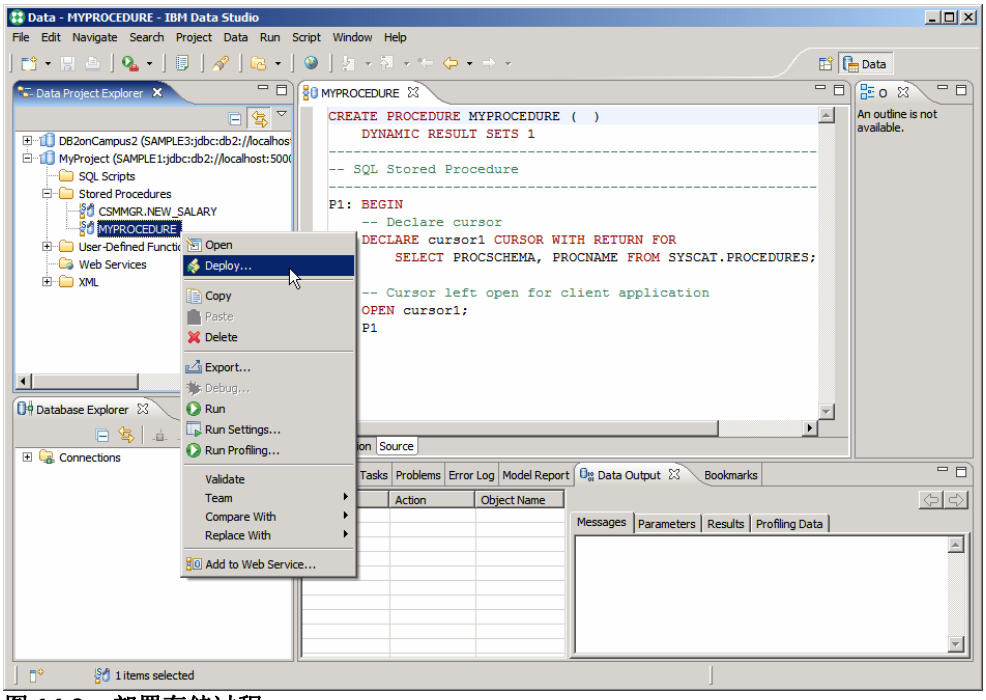


图 14.6 – 部署存储过程

第四步：运行存储过程

一旦存储过程部署完毕，您就可以通过右键点击它再选择“Run”来运行它。运行结果会出现在窗口的右下角“Result”标签标示的区域内。

想从 Command Window 或者 Command Editor 运行存储过程，您可以使用

```
CALL <procedure name>
```

14.2 SQL PL 存储过程基础

SQL 过程语言（SQL PL，SQL Procedural Language）存储过程是很容易创建和学习的。在 DB2 中 SQL PL 具有最佳表现。SQL PL 存储过程（或者简称为“SQL 存储过程”）是本章关注的焦点。

14.2.1 存储过程的结构

如下所示为最基本的存储过程语法构造：

```
CREATE PROCEDURE 存储过程名称 [( {可选变量} )]
                    [可选存储过程属性]        <语句>
```

这里的<statement>是一条单独的语句或者是一组由 BEGIN [ATOMIC] ... END 归结的句群。

14.2.2 可选的存储过程属性

下面是一些可选的存储过程属性的描述:

- LANGUAGE SQL
  这一属性显示了存数过程所使用的语言。LANGUAGE SQL 是其默认值。还有其它的语言供选择，比如 Java 或者 C，可以将这一属性值分别设置为 LANGUAGE JAVA 或者 LANGUAGE C。

- RESULT SETS <n>
  如果您的存储过程将返回 n 个结果集，那么需要填写这一选项。

- SPECIFIC my_unique_name
  这是赋给一个存储过程的唯一名称。一个存储过程是可以被重载的，也就是说许多个不同的存储过程可以使用同一个名字，但这些存储过程所包含的参数数量不同。通过使用 SPECIFIC 关键字，您可以给每一个存储过程起一个唯一的名字，这可以使得我们对于存储过程的管理更加容易。例如，要使用 SPECIFIC 关键字来删除一个存储过程，您可以运行这样的命令：DROP SPECIFIC PROCEDURE。 如果没有使用 SPECIFIC 这个关键字，您将不得不使用 DROP PROCEDURE 命令，并且指明存储过程的名字及其参数，这样 DB2 才能知道哪个被重载的存储过程是您想删除的。

14.2.3 参数

在 SQL PL 存储过程中有三种类型的参数：

- IN – 输入参数
- OUT – 输出参数
- INOUT – 输入和输出参数

例如：

```
CREATE PROCEDURE proc(IN p1 INT, OUT p2 INT, INOUT p3 INT)
CREATE PROCEDURE proc(IN p1 INT, OUT p2 INT, INOUT p3 INT)
```

当调用存储过程，所有的参数都必须提供给 CALL 语句。例如，调用上面的存储过程您需要指明：

```
CALL proc (10,?,4)
CALL proc (10,?,4)
```

在 Call 语句中（？）问号标志是用来设置输出参数的。

下面是另一个可以用来试验的带有参数的存储过程。

```
CREATE PROCEDURE P2 ( IN    v_p1 INT,
                      INOUT v_p2 INT,
                      OUT   v_p3 INT)
LANGUAGE SQL
SPECIFIC myP2
BEGIN
    -- my second SQL procedure
    SET v_p2 = v_p2 + v_p1;
    SET v_p3 = v_p1;
END
```

可以用如下命令在命令编辑器（Command Editor）中调用上面的存储储过程：

```
call P2 (3, 4, ?)
call P2 (3, 4, ?)
```

14.2.4 SQL PL 存储过程中的注释

有两种给 SQL PL 存储过程添加注释的方法，如下所示：

- 使用两个“–”例如：
  `-- 这是一个 SQL-类型的注释`
- 使用类似 C 语言中注释的格式。例如：
  `/* 这是一个 C-类型的注释 */`

14.2.5 复合语句

存储过程中的复合语句是由 BEGIN 和 END 封装起来的几个过程指令和 SQL 语句所组成的。当 BEGIN 关键字后紧随 ATOMIC 关键字时，其封装的复合语句就被当作一个处理单元，也就是说，复合语句中的所有程序指令和语句都必须成功运行，以保证整个复合语句的成功运行。如果其中的一个语句发生错误，那么这整个存储过程所执行的结果都将回滚。图 14.7 展示了一个复合语句的结构。

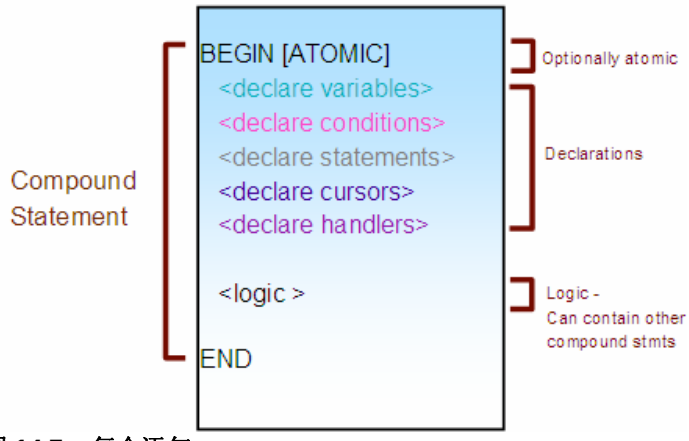


图 14.7 – 复合语句

14.2.6 变量声明

可以用 DECLARE 语句声明变量：

```
DECLARE 变量名 <数据类型> [DEFAULT 值];
```

下面是一些例子：

```
DECLARE temp1 SMALLINT DEFAULT 0;
DECLARE temp2 INTEGER DEFAULT 10;
DECLARE temp3 DECIMAL(10,2) DEFAULT 100.10;
DECLARE temp4 REAL DEFAULT 10.1;
DECLARE temp5 DOUBLE DEFAULT 10000.1001;
DECLARE temp6 BIGINT DEFAULT 10000;
DECLARE temp7 CHAR(10) DEFAULT 'yes';
DECLARE temp8 VARCHAR(10) DEFAULT 'hello';
DECLARE temp9 DATE DEFAULT '1998-12-25';
DECLARE temp10 TIME DEFAULT '1:50 PM';
DECLARE temp11 TIMESTAMP DEFAULT '2001-01-05-12.00.00';
DECLARE temp12 CLOB(2G);
DECLARE temp13 BLOB(2G);
```

14.2.7 赋值语句

用 SET 语句给变量赋值。 例如：

```
SET total = 100;
```

上面的语句等同于下面的：

```
VALUES(100) INTO total;
```

除此之外，所有的变量都可以赋 NULL 值：

```
SET total = NULL;
```

选择表中的数据给变量赋值时，可能会将一个多行的集合赋予变量，如果只想取选择结果的第一行数据时，可以使用

```
SET total = (select sum(c1) from T1);
SET first_val = (select c1 from T1 fetch first 1 row only)
```

您也可以根据外部数据库属性设置变量

```
SET sch = CURRENT SCHEMA;
```

14.3 游标

游标是保持 SELECT 语句执行结果的集合。下面是声明，打开，读取和关闭游标的语法：

```
DECLARE <游标名称> CURSOR [WITH RETURN <返回目标>]
      <SELECT 语句>;
OPEN  <游标名称>;
FETCH <游标名称> INTO <变量>;
CLOSE <游标名称>;
```

游标被声明之后，可以对 WITH RETURN 子句使用如下值：

- CLIENT：把结果集返回给客户端应用程序
- CALLER：把结果集返回给客户端或者发出调用命令的存储过程。

下面是一个使用游标的存储过程的例子：

```
CREATE PROCEDURE set()
DYNAMIC RESULT SETS 1
LANGUAGE SQL
BEGIN
DECLARE cur CURSOR WITH RETURN TO CLIENT
        FOR SELECT name, dept, job
         FROM staff
         WHERE salary > 20000;
OPEN cur;
END
```

14.4 流控制

同其它语言相类似，SQL PL 有几个用于控制逻辑流的语句。下面列出了一些可使用的流控制语句：

```
CASE (选择执行路径 (简单搜索))
IF
FOR (执行表中的每一行)
WHILE
```

ITERATE (强制下一轮循环. 类似于 C 中的 CONTINUE)
LEAVE (跳出一个块或循环体，类似结构语言的 Goto)
LOOP (无限循环)
REPEAT
GOTO
RETURN
CALL (程序调用)

14.5 调用存储过程

下面的代码片段显示了如何用不同的程序语言来调用存储过程。

在 CLI/ODBC 应用程序中调用一个存储过程的例子

```
SQLCHAR *stmt = (SQLCHAR *)
"CALL MEDIAN_RESULT_SET( ? )" ;
  SQLDOUBLE   sal = 20000.0;  /* Bound to parameter marker in stmt
*/
  SQLINTEGER  salind  = 0;    /* Indicator variable for sal */

  sqlrc = SQLPrepare(hstmt, stmt, SQL_NTS);
  sqlrc = SQLBindParameter(hstmt, 1, SQL_PARAM_OUTPUT,
    SQL_C_DOUBLE, SQL_DOUBLE, 0, 0, &sal, 0, &salind);
  SQLExecute(hstmt);


        if (salind == SQL_NULL_DATA)
            printf("Median Salary = NULL\n");
          else
            printf("Median Salary = %.2f\n\n", sal );

        /* Get first result set */
            sqlrc = StmtResultPrint(hstmt);
        /* Check for another result set */
            sqlrc = SQLMoreResults(hstmt);
          if (sqlrc == SQL_SUCCESS)  {
            /* There is another result set */
            sqlrc = StmtResultPrint(hstmt);
          }
```

想要知道更详细的信息，可以参见的例子文件：sqllib/samples/sqlproc/rsultset.c

在 VB.NET 应用程序中调用一个存储过程的例子

```
        Try
          ‘ Create a DB2Command to run the stored procedure
          Dim procName As String = “TRUNC_DEMO”
          Dim cmd As DB2Command = conn.CreateCommand()
          Dim parm As DB2Parameter

          cmd.CommandType = CommandType.StoredProcedure
          cmd.CommandText = procName

          ‘ Register the output parameters for the DB2Command
          parm           = cmd.Parameters.Add(“v_lastname”,
        DB2Type.VarChar)
          parm.Direction = ParameterDirection.Output
          parm           = cmd.Parameters.Add(“v_msg”,
        DB2Type.VarChar)
          parm.Direction = ParameterDirection.Output
```

```
    ‘ Call the stored procedure
    Dim reader As DB2DataReader = cmd.ExecuteReader

  Catch myException As DB2Exception
    DB2ExceptionHandler(myException)
  Catch
    UnhandledExceptionHandler()
  End Try
```

在一个 Java 应用程序中调用一个存储过程的例子

```
  try
  {
    // Connect to sample database
    String url = “jdbc:db2:sample”;
    con = DriverManager.getConnection(url);


    CallableStatement cs = con.prepareCall(“CALL trunc_demo(?,
  ?)”);

    // register the output parameters
    callStmt.registerOutParameter(1, Types.VARCHAR);
    callStmt.registerOutParameter(2, Types.VARCHAR);

    cs.execute();
    con.close();
  }
  catch (Exception e)
  {
    /* exception handling logic goes here */
  }
```

14.6 错误和情况处理器

在 DB2 中，SQL CODE 和 SQLSTATE 关键字是用来判定 SQL 语句执行成功与否的。必须在存储过程之外对这两个关键字加以明确声明，如下：

```
DECLARE SQLSTATE CHAR(5);
DECLARE SQLCODE INT;
```

DB2 在每个 SQL 操作之后自动设置上面几个关键字，对于 SQLCODE，是这样被设置其值的，如：

- = 0, successful.
- > 0, successful with warning
- < 0, unsuccessful
- = 100, no data was found. (i.e.: FETCH statement returned no data)
- = 0, 成功
- > 0, 成功但有警告
- < 0, 失败
- = 100, 未找到指定值.（例如： FETCH 语句无返回值）

对于 SQLSTATE，值按如下规则设置：

- success: SQLSTATE '00000'
- not found: SQLSTATE '02000'

- warning: SQLSTATE '01XXX'
- exception: all other values
- 成功：SQLSTATE '00000'
- 未找到：SQLSTATE '02000'
- 警告：SQLSTATE '01XXX'
- 异常：所有其它值

SQLCODE 是 RDBMS（关系数据库管理系统）特有的，它比 SQLSTATE 更具体。SQLSTATE 在各 RDBMS 间是相同的标准，但一般比较笼统。通常，几个 SQLCODE 可能对应一个 SQLSTATE.。SQLCODE 和 SQLSTATE 的更多细节将在第 18 章，疑难解答 介绍。

任何的 SQL 语句都会产生一个状态，这个状态会和一个 SQLSTATE 匹配。例如当一个值在 SQL 操作中被截取时，会产生 SQLSTATE“01004”。胜于用 SQLSTATE'01004'测试这种情形，这个情形可以被赋予名字。在这个特殊的例子中，此状态被取名为“trunc”，如下所示：

```
DECLARE trunc CONDITION FOR SQLSTATE '01004'
```

其他预定义的一般条件如下：

- SQLWARNING
- SQLEXCEPTION
- NOT FOUND

情况处理

想要解决某一条件引发的状况，您可以创建一个情况处理器，它必须指定：

- 它解决哪种情况引发的问题
- 从哪里开始恢复执行（基于处理器的类型：CONTINUE，EXIT，或者 UNDO）。
- 处理问题的动作，这个动作可以是任意的语句，包括控制结构语句。

如果一个 SQLEXCEPTION 情况发生了，并且没有针对这一情况的处理器，程序就会终止并返回客户端一个错误信息。

处理器类型

有三种类型的处理器：

CUNTINUE – 这个处理器表示：当抛出异常后，由对应的情况处理器解决异常，工作流会继续执行（CONTINUE）抛出异常语句的下一个语句。

EXIT – 这个处理器表示，当异常被抛出，相应的情况处理器解决该异常，工作流会直接到程序的末尾。

UNDO – 这个处理器表示：在异常抛出以后，对应的情况处理器解决此异常情况，工作流直接到达程序的末尾并且撤销所有已实现的操作，或者回滚所有已执行的语句。

图 14.8 显示了情况处理器间的差异和它们的行为。

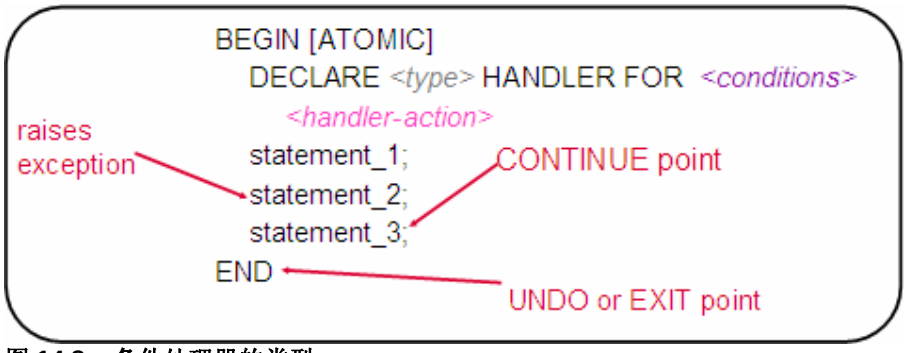


图 14.8 – 条件处理器的类型

14.7 动态 SQL

相对于静态 SQL 语句，动态 SQL 在运行时是无法得知其全部的 SQL 语句的。例如如果 col1 和 tabname 是语句中的两个变量，那么我们正在处理动态 SQL：

```
'SELECT ' || col1 || ' FROM ' || tabname;
```

在 DDL 中建议使用动态 SQL，以避免从属问题和包无效问题。执行递归操作时也需要动态 SQL.

可以通过以下两种方法执行动态 SQL：

- 使用 EXECUTE IMMEDATE 语句– 对于单独执行的 SQL 这是理想的方法
- 在 EXECUTE 语句之前用 PREPARE 语句 – 对于多重执行的 SQL 语句此方法比较理想。

下面的代码片段提供了用两种方法执行动态 SQL 的例子。这个例子的前提是假设了表 T2 是用如下语句创建：

```
CREATE TABLE T2 (c1 INT, c2 INT)

            CREATE PROCEDURE dyn1 (IN value1 INT, IN value2 INT)
            SPECIFIC dyn1
            BEGIN
            DECLARE stmt varchar(255);
            DECLARE st STATEMENT;

            SET stmt = 'INSERT INTO T2 VALUES (?, ?)';

            PREPARE st FROM stmt;

            EXECUTE st USING value1, value1;
            EXECUTE st USING value2, value2;

            SET stmt = INSERT INTO T2 VALUES (9,9)';
            EXECUTE IMMEDIATE stmt;
            END
```

15

第 15 章 – 直接插入 SQL 过程语言、触发器、用户定义函数（UDF）

在本章，我们会讨论直接插入 SQL PL以及其它数据库应用对象，如用户定义函数（UDF）和触发器。（SQL PL全称是SQL procedural language, 表示SQL过程语言。）

注意:
更多关于触发器和用户定义函数（UDF）的信息，请参见以下视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:4367
http://www.channeldb2.com/video/video/show?id=807741:Video:4362

15.1 直接插入 SQL PL

在第 14 章，我们讨论过如何使用 SQL PL 来创建存储过程。SQL PL 能够直接插入，意味着它可以视为一个独立的整体。它是可以在 SQL 函数、触发器和动态复合语句中使用的 SQL 过程语言的子集，所以它只支持一部分 SQL PL 参数，下面是直接插入 SQL PL 所支持的关键字：

```
DECLARE <variable>
SET
CASE
FOR
GET DIAGNOSTICS
GOTO
IF
RETURN
SIGNAL
WHILE
ITERATE
LEAVE
```

下面是直接插入 SQL PL 不支持的关键字：

```
ALLOCATE CURSOR
ASSOCIATE LOCATORS
DECLARE <cursor>
DECLARE ...HANDLER
PREPARE
EXECUTE
EXECUTE IMMEDIATE
LOOP
REPEAT
RESIGNAL
CALL
COMMIT/ROLLBACK
```

下面是一个在动态复合语句中使用直接插入 SQL PL 的例子，如果您想运行它，您可以将它保存到脚本文件中，并保证建立了下面两个数据表：

```
CREATE TABLE T1 (c1 INT)
CREATE TABLE T3 (c1 INT)
```

```
BEGIN ATOMIC
  DECLARE cnt      INT DEFAULT 0;
  DECLARE sumevens INT DEFAULT 0;
  DECLARE err_msg  VARCHAR(1000) DEFAULT '';
  WHILE (cnt < 100) DO
    IF mod(cnt, 2) = 0 THEN
       SET sumevens = sumevens + cnt;
    END IF;
    SET cnt=cnt+1;
  END WHILE;
  INSERT INTO T3 values (sumevens);
  SET cnt = (SELECT 0 FROM SYSIBM.SYSDUMMY1);
  FOR cur1 AS SELECT * FROM T1 DO
    IF cur1.c1 > 100 THEN
      SET cnt = cnt + 1;
    END IF;
  END FOR;

  SET err_msg = 'Rows with values > 100 is:' || char(cnt);
  SIGNAL SQLSTATE '80000' SET MESSAGE_TEXT = err_msg;
END!
```

如果您将上面的直接插入 SQL PL 保存到文件“myScript.txt”中，您可以直接运行它，命令如下所示：

```
db2 -td! –vf myScript.txt
```

15.2 触发器（Trigger）

触发器是与定义规则的单个基本表或视图相关联的数据库对象，当在该表上发生 INSERT、UPDATE、或 DELETE 等操作时，它们就会自动激活并执行定义的操作。激活触发器的语句称为触发 SQL 语句。

15.2.1 触发器的类型

触发器有三种类型：前触发器、后触发器、替代触发器。

前触发器（“Before” trigger）

前触发器，被指定为在发生所定义的触发器事件（即在触发器定义中指定的对表的插入、更新或删除操作）之前激活的一种触发器。前触发器的行为不会激活其它触发器，所以前触发器不支持 INSERT、UPDATE、DELETE 语句。

图 15.1 展示了前触发器的一个简单例子。

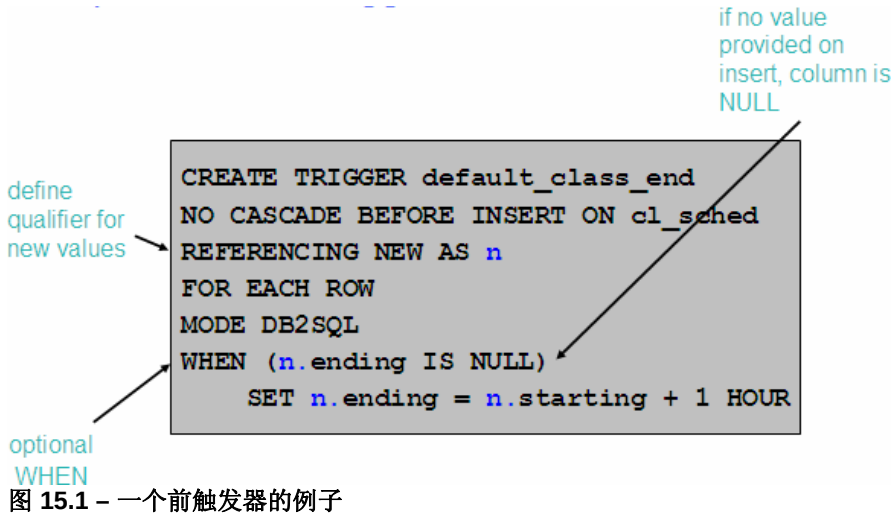


图 15.1 – 一个前触发器的例子

在图 15.1 中，触发器“default_class_end”会在插入（INSERT）数据到表 cl_sched 之前激活。表 cl_sched 是 SAMPLE 数据库中的一个数据表，所以当您连接到数据库的时候，可以自行创建并测试这个触发器。触发器定义中的变量 n 表示要插入的值，也即是 INSERT 中的新值。该触发器会检查插入值的正确性，如果在插入中列“ending”没有值，那么这个触发器会将“starting”列的值加 1 小时插入到“ending”列。

下面的语句展示如何使用触发器：

```
C:\Program Files\IBM\SQLLIB\BIN>db2 insert into cl_sched (class_code,
day,  starting) values ('abc', 1, current time)
DB20000I  The SQL command completed successfully.

C:\Program Files\IBM\SQLLIB\BIN>db2 select * from cl_sched

CLASS_CODE DAY    STARTING ENDING
---------- ------ -------- --------
042:BF          4 12:10:00 14:00:00
553:MJA         1 10:30:00 11:00:00
543:CWM         3 09:10:00 10:30:00
778:RES         2 12:10:00 14:00:00
044:HD          3 17:12:30 18:00:00
abc             1 11:06:53 12:06:53

6 record(s) selected.
```

下面展示的触发器“validate_sched”扩展了上例触发器“default_class_end”的功能。同样，您可以在 SAMPLE 数据库上创建并测试它。

```
CREATE TRIGGER validate_sched
NO CASCADE BEFORE INSERT ON cl_sched
REFERENCING NEW AS n
FOR EACH ROW
MODE DB2SQL
BEGIN ATOMIC
-- supply default value for ending time if null
IF (n.ending IS NULL) THEN
    SET n.ending = n.starting + 1 HOUR;
END IF;
```

```
-- ensure that class does not end beyond 9pm
IF (n.ending > '21:00') THEN
    SIGNAL SQLSTATE '80000'
    SET MESSAGE_TEXT='class ending time is beyond 9pm';
ELSEIF (n.DAY=1 or n.DAY=7) THEN
    SIGNAL SQLSTATE '80001'
        SET MESSAGE_TEXT='class cannot be scheduled on a
weekend';
END IF;
END
```

后触发器（“After” trigger）

后触发器在触发 SQL 语句成功执行后激活。后触发器的行为可以激活其它触发器（支持 16 级级连）。后触发器支持 INSERT、UPDATE、DELETE 语句，下面是一个后触发器的例子：

```
CREATE TRIGGER audit_emp_sal
AFTER UPDATE OF salary ON employee
REFERENCING OLD AS o NEW AS n
FOR EACH ROW
MODE DB2SQL
  INSERT INTO audit VALUES (
    CURRENT TIMESTAMP,  ' Employee ' || o.empno || ' salary
changed from ' || CHAR(o.salary) || ' to ' || CHAR(n.salary)
|| ' by ' || USER)
```

在本例中，触发器 audit_emp_sal 用于数据表“employee”中对列“salary”的审计。当某人对该列作出修改后，触发器会激活并把修改的信息存到“audit”数据表中。“OLD as o NEW as n ”语句表示用“o”代表数据表中旧的或现存的数据，用“n”代表 UPDATE 语句中的新数据。所以，“o.salary ”表示旧的或现存的薪金数值，“n.salary”表示新的薪金数值。

替代触发器（“INSTEAD OF” Trigger）

替代触发器在视图中定义，该触发器中定义的逻辑会替代视图中的触发 SQL 语句来执行。例如，您在一个视图上执行更新操作，替代触发器会在这个视图所对应的基础表上实际地运行这个更新操作。

触发器不能从 IBM Data Studio 创建。但能通过控制中心（Control Center）或者命令行工具（Command Window、Command Line Processor、 Command Editor)创建它们。

实验 #12 从控制中心创建一个触发器

实验目标

触发器一种数据库对象，它会在数据表的数据发生改变时激发并执行指定的逻辑操作。在本实验中，您会使用控制中心创建一个触发器，这个触发器出于审计的需要，会记录 SALES 表的更改。它将记录更改这个表的用户 ID 和更改时间。

实验过程

1. 打开 控制中心。
2. 创建一个数据表用于记录信息。这个表的信息如下：
   表名：`saleslog`

   第一列：
   属性名：userid
   数据类型：VARCHAR(128)
   其它属性： NOT NULL

   第二列
   属性名：daytime
   数据类型： TIMESTAMP
   其它属性： NOT NULL

   提示： 在命令编辑器（Command Editor）中使用 CREATE TABLE 语句或者控制中心里的创建表向导（Create Table wizard）来创建这个表。

3. 在控制中心中展开 EXPRESS 数据库文件夹，右键单击 Triggers 文件夹，然后选择创建（Create）选项，打开创建触发器（Create Trigger）对话框。
4. 在对话框中输入下面的信息：
   触发器模式（Trigger schema）：您登录数据库的用户 ID（应当为默认的设置）

   触发器名字（Trigger name）：audit_sales

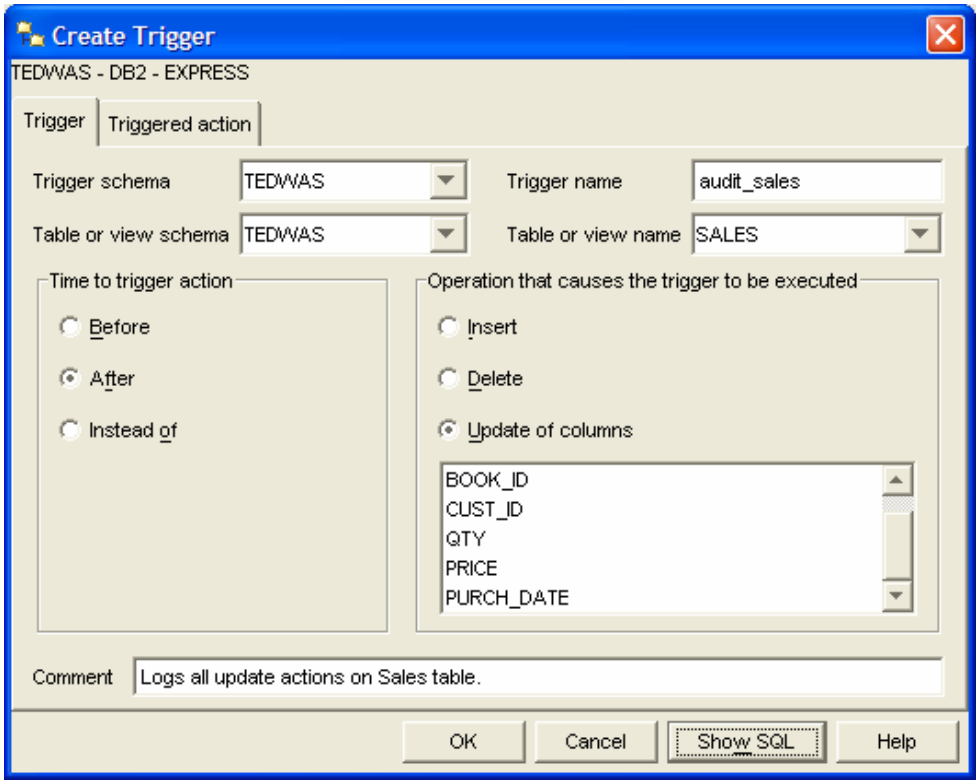
   数据表或者试图模式（Table or view name）：您登录数据库的用户 ID（应当为默认的设置）

   表或视图名字（Table or view name）：SALES

   触发器激活时机（Time to trigger action）：After

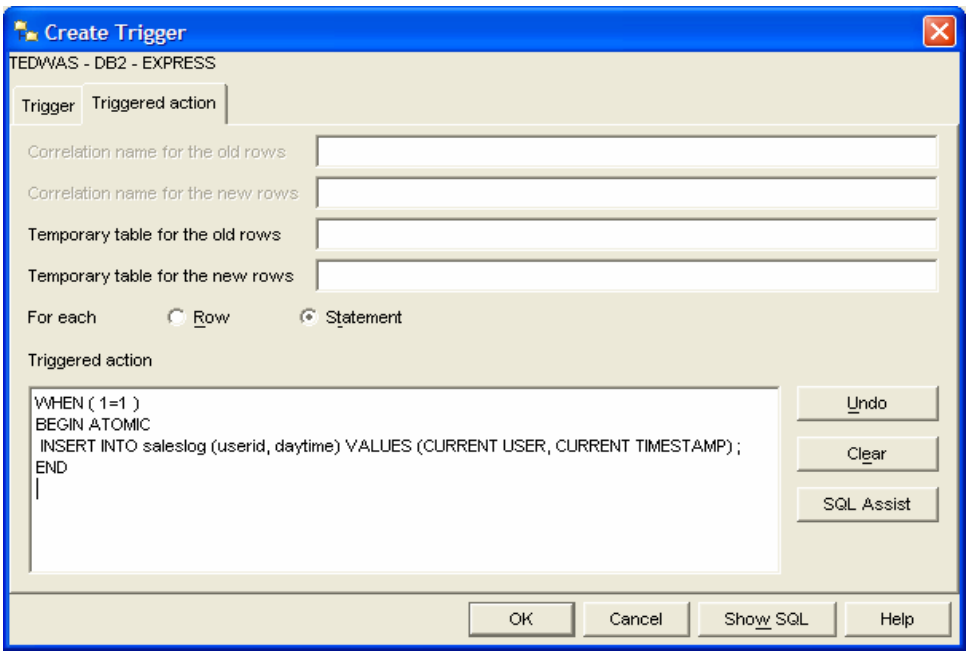
   激活触发器的操作（Operation that causes the trigger to be executed）：*Update of columns*（不要指明属性列，因为我们想记录这个表任何的更新）
   Comment: *Logs all update actions on Sales table.*

   备注（Comment）：填入“记录 Sales 表的所有更新”



5. 在触发器动作（Triggered action）标签页，选择 *For Each **STATEMENT*** 选项，在触发器动作输入框中输入下面的语句：

```
WHEN ( 1=1 )
BEGIN ATOMIC
  INSERT INTO saleslog (userid,  daytime) VALUES (CURRENT USER, CURRENT TIMESTAMP);
END
```



点击 OK 来创建一个触发器。
（注意：对于语句的触发器会在关联的触发语句执行完成后激活，对于行的触发器，会在关联的触发语句操作完任意一行的时候激活。）

6. 现在您可以在控制中心的触发器（Triggers）文件夹中看到刚才创建的触发器。
7. 检索 saleslog 表，确保里面没有数据，如果里面有数据，请将它们全部删除（DELETE FORM salelog）。
8. 尝试更新 sale 表的记录。（提示：使用命令编辑器 Command Editor 或者 SQL 帮助向导 SQL Assist Wizard）
9. 再次检索 saleslog 表的数据，您发现里面有什么？

15.3 用户定义函数 (UDF)

用户定义函数（UDF）是一个数据库应用对象，它在输入数据集和输出数据集中建立映射关系。例如，一个函数输入以寸为单位的测量值，输出以厘米为单位的结果。

DB2 支持使用 SQL PL，C/C++，Java，CLR（Common Language Runtime），OLE（Object Linking and Embedding）来创建函数。本书中我们关注 SQL PL 创建的函数，因为它们简单、普及、有较佳的性能。

函数有 4 种类型：标量函数、表函数、行函数、列函数。本章我们只涉及标量函数和表函数。

15.3.1 标量函数（Scalar function）

标量函数只返回一个值，标量函数中不能有更改数据库状态的 SQL 语句，例如不允许 INSERT、UPDATE、DELETE 语句。DB2 中有些内建的标量函数，如 SUM()，AVG()，COALESCE()，SUBSTR()。

DB2 允许您将常用的程序逻辑封装成用户自定义函数。例如，您从 ORACLE 将数据应用迁移到 DB2 上，而在您的程序中广泛使用了 ORACLE 的 NVL()函数。在 DB2 中与此相对应的函数却叫 COALESE，这时您可以在 DB2 中创建用户定义函数，将函数命名为 NVL，然后让这个自定义的 NVL 函数调用 DB2 内置的 COALESCE 函数，如下所示：

```
CREATE FUNCTION NVL (p_var1 VARCHAR(30),
                     p_var2 VARCHAR(30))
SPECIFIC nvlvarchar30
RETURNS VARCHAR(30)
RETURN COALESCE(p_var1,  p_var2)
```

COALESE 函数会返回第一个非空的参数。

下面是另一个标量函数的例子，函数名为：deptname，它根据雇员的 id 返回雇员所在部门的编号。

```
CREATE FUNCTION deptname(p_empid VARCHAR(6))
RETURNS VARCHAR(30)
SPECIFIC deptname
BEGIN ATOMIC
  DECLARE v_department_name VARCHAR(30);
  DECLARE v_err VARCHAR(70);
  SET v_department_name = (
    SELECT d.deptname FROM department d,  employee e
      WHERE e.workdept=d.deptno AND e.empno= p_empid);
  SET v_err = 'Error: employee ' || p_empid || ' was not
found';
  IF v_department_name IS NULL THEN
    SIGNAL SQLSTATE '80000' SET MESSAGE_TEXT=v_err;
  END IF;
RETURN v_department_name;
END
```

在命令窗口或者 Linux/UNIX 的 shell 中输入下面命令进行测试：

```
db2 "values (deptname ('000300'))"
```

调用标量用户定义函数

标量 UDF 能够在 SQL 语句中调用，用在 SQL 语句需要一个标量或者 VALUES 子句中。下面是调用 COALESCE 标量函数的例子：

```
SELECT DEPTNAME,  COALESCE(MGRNO,  'ABSENT') FROM DEPARTMENT

VALUES COALESCE('A', 'B')
```

15.3.2 表函数（Table function）

表函数返回一个表或者若干行，您可以在查询中使用 FROM 子句来调用它们。表函数与标量函数不同，它能够改变数据库的状态，因此，它能够使用 INSERT、UPDATE、DELETE 语句。DB2 内建的表函数有 SNAPSHOT_DYN_SQL()、MQREADALL()等等。表函数和视图相似，但是表函数能够改变数据库的数据（INSERT、UPDATE、DELETE），所以它更加强大。通常，它们用来返回一个表或者保存审计信息。

下面是一个表函数的例子，它枚举部门的雇员。

```
CREATE FUNCTION getEnumEmployee(p_dept VARCHAR(3))
RETURNS TABLE
  (empno CHAR(6),
   lastname VARCHAR(15),
   firstnme VARCHAR(12))
SPECIFIC getEnumEmployee
RETURN
  SELECT e.empno,  e.lastname,  e.firstnme
    FROM employee e
    WHERE e.workdept=p_dept
```

输入下面的命令来测试这个函数：

```
db2 "SELECT * FROM table(getEnumEmployee('D11')) AS t"
```

调用表用户定义函数

表用户定义函数能够在 SQL 语句的 FROM 子句中调用，并且需要应用 TABLE()函数和别名，如图 15.2 所示。图 15.2 展示了如何调用我们上面定义的“getEnumEmployee”函数。

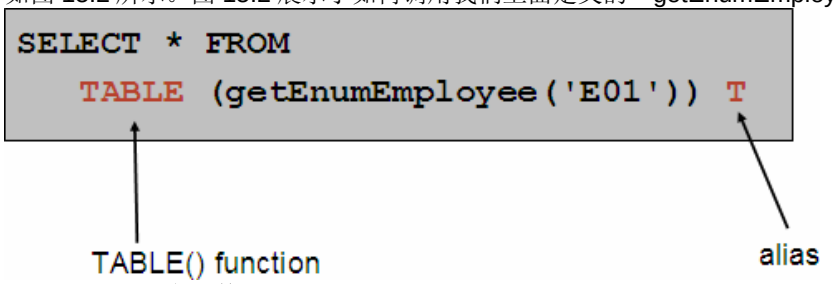


图 15.2 – 调用表函数

实验 #13 使用 IBM Data Studio 创建用户定义函数（UDF）

实验目标

在本实验中，您要使用 IBM Data Studio 创建一个标量用户定义函数。您能够获得更多的使用 Data Studio 经验，同样也能够提高您使用 SQL PL 创建用户定义函数的熟练程度。

实验过程

1. 打开 IBM Data Studio（提示：从开始菜单启动）。
2. 在 Data Project Explorer 窗口，选择在前面实验中创建的项目，然后选择打开项目（Open Project）。
3. 右键单击用户定义函数（User-Defined Functions）文件夹，选择新建（New）项，选择 User-Defined Function 菜单项。这是会出现“New User-Defined Function”向导，在向导中填入下面的值：

   Project：确保是您刚才创建的那个项目名字。
   Name：FUNCTION1
   Language：SQL

   上面的项目填写完毕后，点击 FINISH

4. 此时编辑器窗口打开了一个函数的骨架。如下所示：

```
CREATE FUNCTION FUNCTION1( )
        RETURNS INTEGER
        NO EXTERNAL ACTION
------------------------------------------------------------------------
-- SQL UDF (Scalar)
------------------------------------------------------------------------
F1: BEGIN ATOMIC
        RETURN SELECT count(*) FROM SYSCAT.FUNCTIONS;
END
```

将其改为如下代码：

```
CREATE FUNCTION booktitle(p_bid INTEGER)
RETURNS VARCHAR(300)
---------------------------------------------------------
SQL UDF (Scalar)
---------------------------------------------------------
SPECIFIC booktitle
F1: BEGIN ATOMIC
DECLARE v_book_title VARCHAR(300);
DECLARE v_err VARCHAR(70);
SET v_book_title = (SELECT title FROM books WHERE p_bid = book_id);
SET v_err = 'Error: The book with ID ' || CHAR(p_bid) || '
was not found.';
IF v_book_title IS NULL THEN SIGNAL SQLSTATE '80000' SET
MESSAGE_TEXT=v_err;
END IF;
RETURN v_book_title;
END
```

5. 右键单击函数，然后选择部署（Deploy）来构建这个函数。如果出现一个对话框提示“是否保存更改”，请选择 Yes。之后出现部署选项（Deploy Options）面板。保留其中的默认值，单击 Finish。
6. 右键单击函数，并选择 Run 选项来运行这个函数。
7. 因为该函数要输入参数，所以会出现一个对话框要求您输入参数。
   输入值：80002
   此时出现什么结果？
   尝试另一个值：1002
   又得到什么结果？（提示：查看 Output 视图的 Message）
8. 实验完毕，关闭 IBM Data Studio。

16

第 16 章 – DB2 pureXML

本章我们将讨论 pureXML，DB2 9 引入这一新技术来支持 XML 原生存储。本章中很多示例和概念皆引自于 IBM 红宝书：《DB2 9：pureXML 概观和快速启航》。更多信息请参阅“参考资源”这一节。图 16.1 的 DB2 概览图指示出我们在本章要讨论内容。

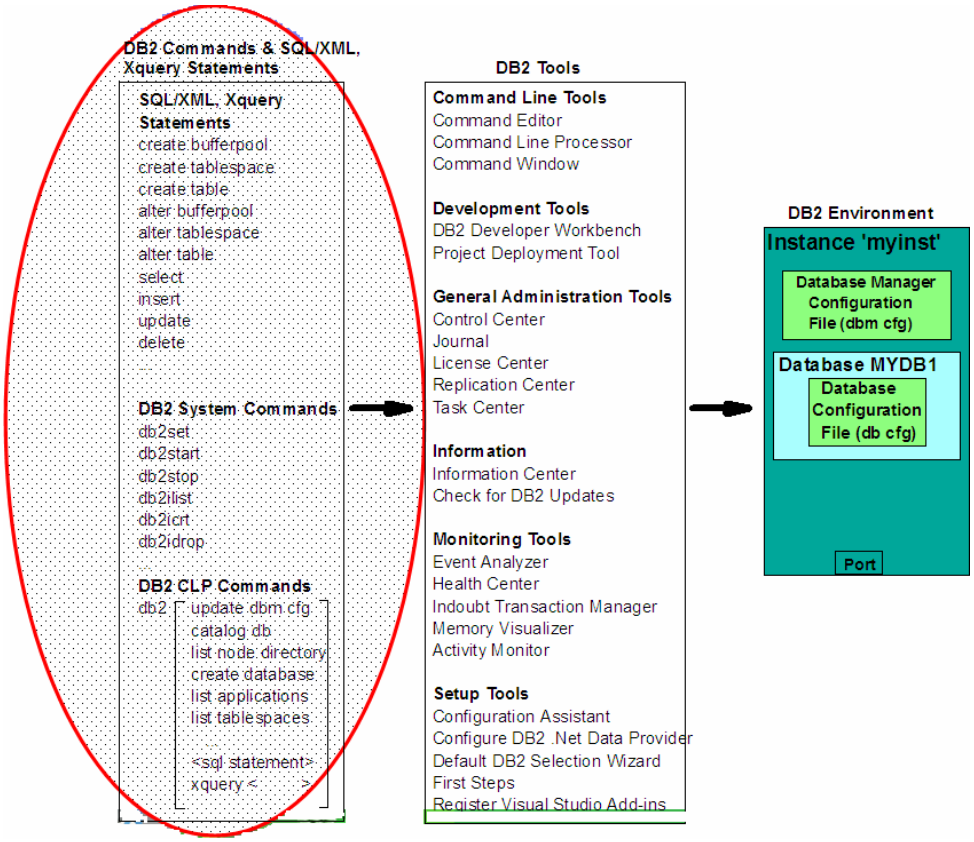


图 16.1 – DB2 概览图：DB2 命令, SQL/XML 和 XQuery

注意：
更多关于 pureXML 的信息，请参见以下视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:4382

16.1 在数据库中使用 XML

XML 文档可以存储在文本文件、XML 库或者数据库中。许多公司更倾向于把它们存储在数据库中, 这主要基于两个原因：

- 管理海量 XML 数据是一个数据库的问题。XML 与其他数据相似，只是格式不同罢了。使用数据库存储关系型数据的原因也同样适用于 XML 数据：数据库提供了高效的数据检索和获取途径，对于数据持久化的健壮支持，数据备份和恢复，事务处理支持以及性能保证和可扩展性。
- 集成：通过将关系型数据和 XML 数据存储在一起，您可以把新的 XML 数据和现有的关系型数据集成起来，同时把 SQL 和 XPath 或者 XQuery 组合成为单一查询。另外，关系型数据可以用 XML 格式发布，反之亦然。通过集成，数据库可以更好的支持 Web 应用、SOA 和 Web 服务。

16.2 XML 数据库

有两种类型数据库可以存储 XML 数据：
- 启用 XML 的数据库
- 原生 XML 数据库

16.2.1 启用 XML 的数据库

启用 XML 的数据库使用关系模型作为其核心数据存储模型。这需要在 XML（层次型）数据模型和关系数据模型之间建立映射，或者把 XML 数据存储为字符大对象。尽管稍显“陈旧”，很多数据库厂商依然在使用这一技术。图 16.2 更细致的解释了启用 XML 的数据库存储数据的两种方式。

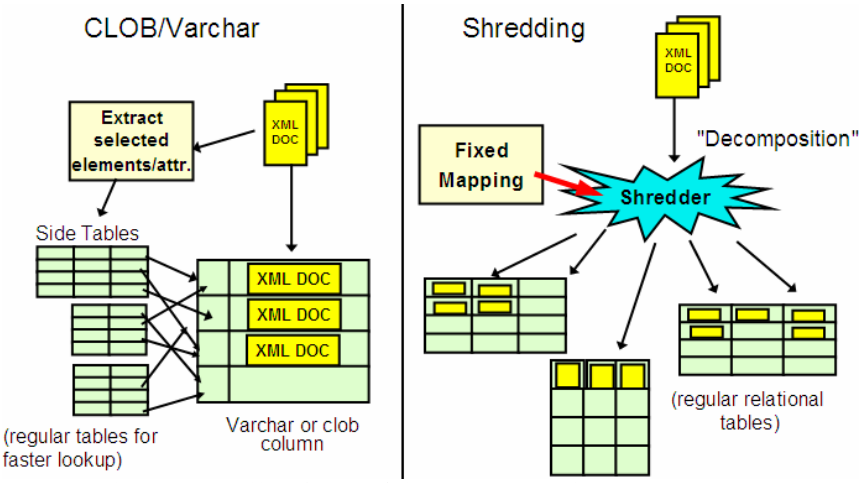


图 16.2 –启用 XML 的数据库的两种存储 XML 的方式

图 16.2 左半部分描述了使用“字符大对象和变长字符”存储 XML 的方法。这种方法以字符大对象或者变长字符的形式把 XML 文档存储为未经解析的字符串。如果 XML 文档以字符串形式存储，那么欲获取其部分内容，应用程序就必须获取整个字符串，然后完成解析的工作。显然，这种方法缺少灵活性。

图 16.2 右半部分描述了启用 XML 的数据库的另一种存储 XML 数据的方法，称为“裂分或者解构”，整个 XML 文档会被裂分作存储在数据表中的小块。通过这种方法，XML 文档的层次模型被转化成关系模型。这样同样无益于灵活性：XML 文档的一处改动不能轻易传递到相应的数据表，而且很可能需要创建额外的许多表。这种方法对性能的影响同样不能让人满意：当您需要取回原始 XML 文档时，您必须执行代价可观的 SQL 操作，而当牵涉到更多的表时，这一代价将更加惊人。

16.2.2 原生 XML 数据库

原生 XML 数据库在内部使用层次型数据模型存储、处理 XML 数据。存储格式与处理格式一致，即层次型数据模型不再映射为关系数据模型，XML 文档也不再存储为镜像。XPath 或者 XQuery 交由本地引擎直接处理，而不再转化成 SQL。“原生 XML 数据库”由此得名。自 DB2 V9 起，这是目前提供这一支持的唯一商业数据服务器。

16.3 DB2 中的 XML

图 16.3 大致描述了自 DB2 V9 起，DB2 是如何将关系型数据和层次型数据（XML 文档）存储在一起。假定图中 dept 表定义如下：

CREATE TABLE dept (deptID CHAR(8),…, deptdoc XML);

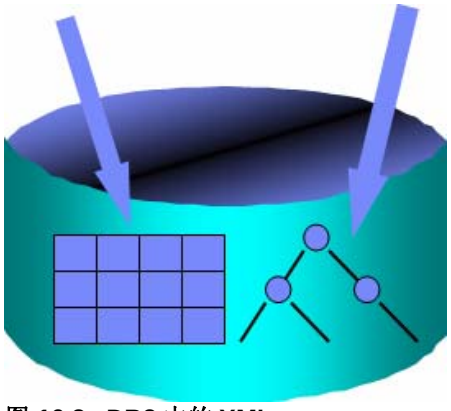


图 16.3 –DB2 中的 XML

注意 deptdoc 列采用了新的数据类型，XML。图中左侧箭头表明 deptID 列存储在关系表中，而 deptdoc 列以经解析的层次型格式存储。

图 16.4 说明从 DB2 9 开始，有四种方式访问数据：
- 使用 SQL 访问关系型数据
- 使用带有 XML 扩展的 SQL 访问 XML 数据
- 使用 XQuery 访问 XML 数据
- 使用 XQuery 访问关系型数据

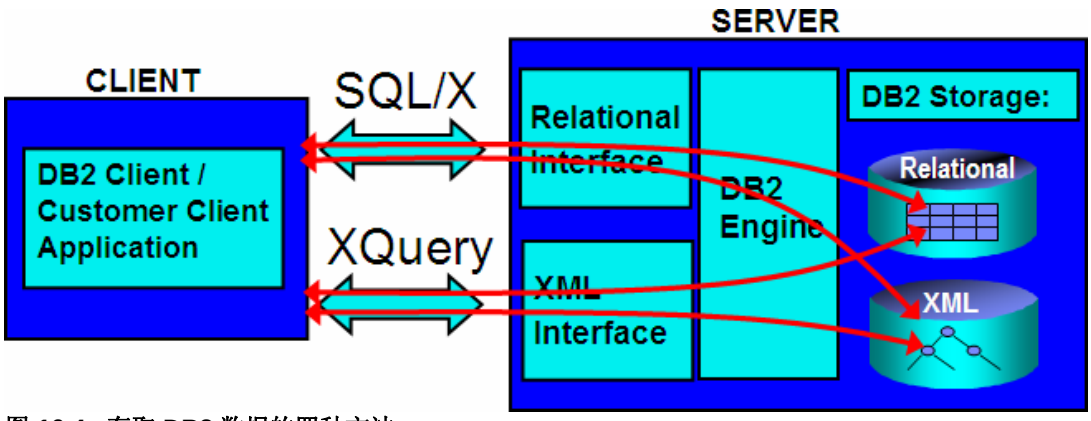


图 16.4 –存取 DB2 数据的四种方法

因此，惯用 SQL 的人会把 DB2 看作世界级的关系型数据库管理系统，同时扩展了对于 XML 的支持；而 XML 的使用者会认为 DB2 是世界级的 XML 数据库，同时对 SQL 给予了支持。

请注意 IBM 使用 pureXML 而不是“native XML”来定义该创新技术。因为其他厂商使用“字符大对象/变长字符”或者“解构”的早期技术来存储 XML 文档，并且称之为“native XML”。为了避免混淆，

IBM 决定使用 pureXML 并且注册了商标，这样其他数据库或者 XML 技术厂商不能再使用这一术语来为某个相异的技术命名。在 DB2 9.1 中，pureXML 仅支持 Unicode 数据库。DB2 9.5 解除了这个限制，无论是 unicode 还是非 unicode 环境，您都可以自由使用 pureXML。

16.3.1 pureXML 技术优势

pureXML 提供了许多技术上的优势。

1. 考虑到关系表使用新的 XML 数据类型存储 XML 文档，这种一致性使得先前在关系型数据库上的投资不会贬值。

2. 减少代码复杂性。例如，图 16.5 展示了使用和不使用 pureXML 的同一段 PHP 脚本的变化。使用 pureXML(绿色方框)减少了代码行数。这不仅降低了代码的复杂性，而且因为解析和维护的代码更少，应用的整体性能得到提升。

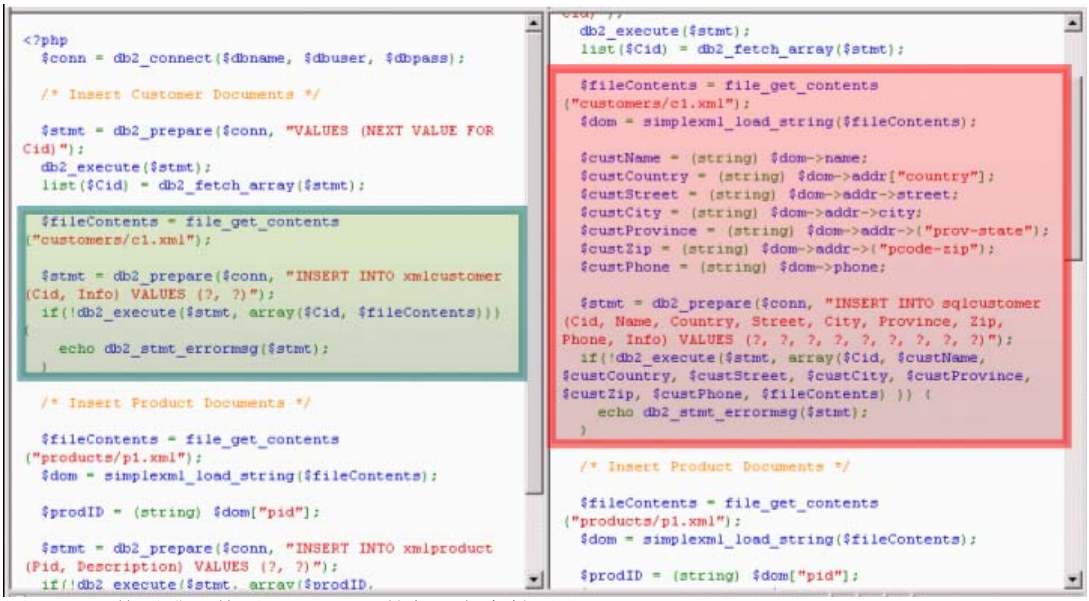


图 16.5 – 使用或不使用 pureXML 的代码复杂性对比

3. 使用 XML 和 pureXML 技术使变更数据库模式更加容易。图 16.6 是 pureXML 带来更多灵活性的例子。图中假定数据库由一张雇员表和一张部门表组成。通常在非 XML 数据库中，如果经理让您为每一位雇员存储不止一个电话号码（除了家庭电话还有手机号码），那么您会在雇员表中增加一列来存储手机号码。然而，这种做法违背了关系型数据库的规则。如果不愿违背规则，那么您只能另外创建一个电话表，把所有电话信息迁移到这张表中。然后您可以把手机号码添加到这张表中。创建这张新表的代价很高，不只由于需要移动之前存在的大量数据，同时应用程序中所有相关的 SQL 都必须修改指向这张新表。

   而图的左侧展示了如何通过 XML 解决这个问题。如果雇员“Christine”同时有一个手机号码，那么可以添加一个新的标签来对应这条信息。“Michael”没有手机号码，那么我们什么都不需要做。

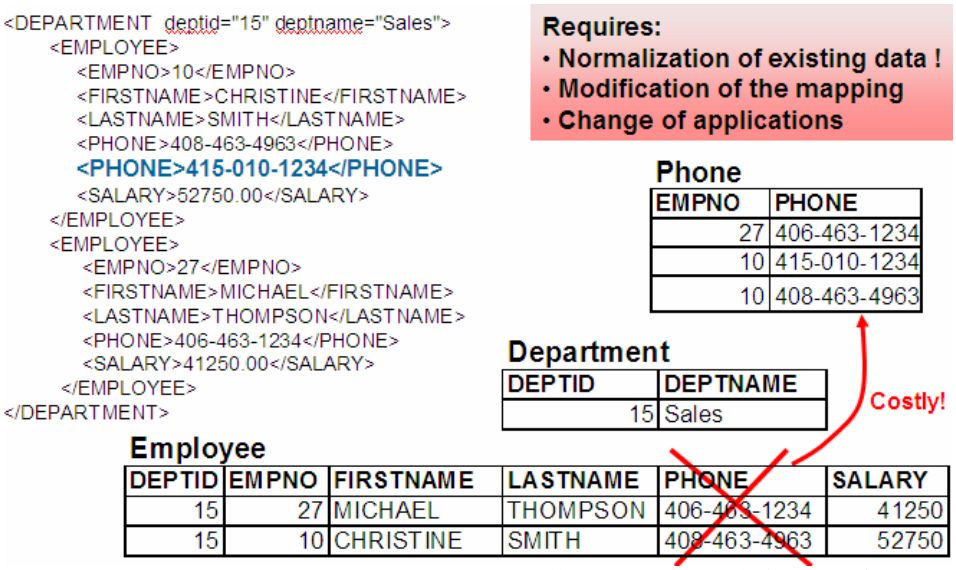


图 16.6 – Increased data flexibility using XML 使用 XML 可以增加代码灵活性

4. 提升 XML 应用程序的性能。对比测试表明，使用 pureXML 技术后几种不同类型 XML 应用程序的性能都会得到显著改善。测试针对一家由其他早期技术迁移到 pureXML 的公司，表 16.7 给出了测试结果。中间列对应早期 XML 处理技术，第三列是使用 pureXML 的效果。

Task	Other relational DB	DB2 with pureXML
Development of search and retrieval business processes	CLOB: 8 hrs Shred: 2 hrs	30 min.
Relative lines of I/O code	100	35 (65% reduction)
Add field to schema	1 week	5 min.
Queries	24 - 36 hrs	20 sec - 10 min

图 16.7 – 使用 pureXML 技术能够提升应用程序的性能

16.3.2 XPath 基础

XPath （XML Path Language， XML 路径语言）是一种用来查询 XML 文档的语言。图 16.8 是一个 XML 文档的例子，图 16.9 经过解析后的层次结构（也称作“节点”或“叶片”）。我们将用图 16.9 显示的层次结构解释 XPath 如何工作。

```
<dept bldg="101">
  <employee id="901">
    <name>John Doe</name>
      <phone>408 555 1212</phone>
      <office>344</office>
  </employee>
  <employee id="902">
      <name>Peter Pan</name>
      <phone>408 555 9918</phone>
      <office>216</office>
  </employee>
</dept>
```

图 16.8 – 一个 XML 文档

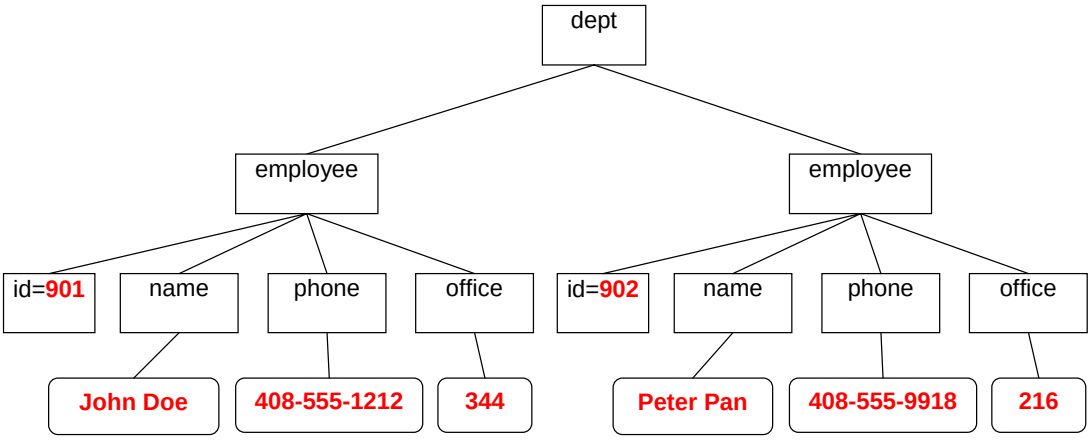


图 16.9 – 图 16.8 的 XML 文档经过解析后的层次结构

比较 XPath 和 MS-DOS 或者 Linux/UNIX 中的更改目录命令是学习 XPath 的快捷途径。在 MS-DOS 或者 Linux/UNIX 中，使用如下 cd 命令展开目录树：

cd /directory1/directory2/…

同样，XPATH 使用斜线"/"从 XML 文档的一个元素定位到另一个。例如，对于图 16.9 所示的文档可以使用如下查询获得所有雇员的姓名：

/dept/employee/name

XPath 表达式

XPath 表达式使用完整限定路径指定元素和（或者）属性。符号"@"用来指定属性。若仅仅需要获得一个元素的值(文本节点)，则使用"text()"函数。表 16.1 列出了针对图 16.9 中 XML 文档的 XPath 查询和相应的结果。

XPath	Result
/dept/@bldg	**101**
/dept/employee/@id	**901** **902**
/dept/employee/name	**<name>Peter Pan</name>** **<name>John Doe</name>**
/dept/employee/name/text()	**Peter Pan** **John Doe**

表 16.1 – XPath 表达式举例

XPath 通配符

XPath 中有两个通配符：

- "*"匹配任意标签名

- “//”是“子或自身”通配符

表 16.2 列出了更多使用通配符的例子。

XPath	Result
/dept/employee/*/text()	**John Doe** **408 555 1212** **344** **Peter Pan** **408 555 9918** **216**
/dept/*/@id	**901** **902**
//name/text()	**Peter Pan** **John Doe**
/dept**//**phone	**<phone>408 555 1212</phone>** **<phone>408 555 9918</phone>**

表 16.2 – XPath 通配符举例

XPath 谓词

谓词部分放置在[]中。作为对照，可以认为它们等价于 SQL 中的 WHERE 子句。例如[@id=”902”]会被解释成“WHERE 属性 id 值等于 902”。在一个 XPath 表达式中可以有多个谓词。要指定特定位置的项，使用 [n]，这表明同一级元素的第 n 项会被选中。例如，employee[2]表示第二个雇员被选中。表 16.3 列出了更多示例。

XPath	Result
/dept/employee[@id=“902”]/name	**<name>Peter Pan</name>**
/dept[@bldg=“101”]/employee[office >“300”]/name	**<name>John Doe</name>**
//employee[office=“344” OR office=“216”]/@id	**901** **902**
/dept/employee[2]/@id	**902**

表 16.3 –XPath 谓词举例

XPath 的父元素

与 MS-DOS 或者 Linux/UNIX 相似，在表达式中使用“.”（读作 dot）表示引用当前上下文，使用“..”表示引用上一级上下文。更多示例参见表 16.4。

XPath	Result
/dept/employee/name[../@id=“902”]	**<name>Peter Pan</name>**
/dept/employee/office[.>“300”]	**<office>344</office>**
/dept/employee[office > “300”]/office	**<office>344</office>**
/dept/employee[name=“John Doe”]/../@bldg	**101**

/dept/employee/name[.="John Doe"]/../../@bldg	**101**

表 16.4 – XPath 的父元素

16.3.3 XQuery 的定义

XQuery 是为 XML 创建的查询语言。XQuery 支持路径表达式从而为 XML 层次结构提供导航。事实上，XPath 是 XQuery 的子集；因而先前关于 XPath 的所有知识也适用于 XQuery。XQuery 同时支持类型化和无类型数据。因为 XML 文档不包含遗失的或者未知数据所以 XQuery 不提供空值。XQuery 返回 XML 数据序列。请注意 XQuery 和 XPath 表达式都区分大小写，这一点非常重要。

XQuery 支持 FLWOR 表达式。FLWOR 与 SQL 中 SELECT-FROM-WHERE 表达式相似。下一节将会探讨 FLWOR 的更多细节。

XQuery 的 FLWOR 表达式

FLWOR 是几个词首字母的缩写：

- FOR：对序列进行迭代
- LET：绑定变量
- WHERE：定义过滤器
- ORDER：将过滤结果排序
- RETURN：返回查询结果

该表达式允许对 XML 文档进行操作，同时返回另一个表达式。例如，假定一张表定义如下：

```
CREATE TABLE dept(deptID CHAR(8),deptdoc XML);
```

将如下 XML 文档插入到 deptdoc 列：

```
<dept bldg="101">
   <employee id="901">
         <name>John Doe</name>
         <phone>408 555 1212</phone>
         <office>344</office>
    </employee>
    <employee id="902">
         <name>Peter Pan</name>
         <phone>408 555 9918</phone>
         <office>216</office>
     </employee>
</dept>
```

然后可以像如下使用 FLWOR 表达式：

```
xquery
for $d in db2-fn:xmlcolumn('dept.deptdoc')/dept
let $emp := $d//employee/name
where $d/@bldg > 95
order by $d/@bldg
return
  <EmpList>
  {$d/@bldg, $emp}
  </EmpList>
```

返回结果如下：

```
<EmpList bldg="101">
  <name>
        John Doe
  </name>
  <name>
        Peter Pan
  </name>
</EmpList>
```

16.3.4 插入 XML 文档

可以使用 SQL INSERT 语句或者导入工具向 DB2 数据库插入 XML 文档。此处 XQuery 无能为力因为在 XQuery 标准中没有定义这项操作。

```
db2 –tvf table_creation.txt
```

可以使用 db2 –tvf table_creation.txt 命令从 DB2 的命令窗口或者 Linux Shell 中运行下面这段脚本：

table_creation.txt

```
-- (1)
drop database mydb
;

-- (2)
create database mydb using codeset UTF-8 territory US
;

-- (3)
connect to mydb
;

-- (4)
create table items (
  id          int primary key not null,
  brandname   varchar(30),
  itemname    varchar(30),
  sku         int,
  srp         decimal(7,2),
  comments    xml
);

-- (5)
create table clients(
  id          int primary key not null,
  name        varchar(50),
  status      varchar(10),
  contact     xml
);

-- (6)
insert into clients values (77, 'John Smith', 'Gold',
   '<addr>111 Main St., Dallas, TX, 00112</addr>')
;

-- (7)
```

```
IMPORT FROM "D:\Raul\clients.del" of del xml from "D:\Raul" INSERT INTO CLIENTS (ID,
NAME, STATUS, CONTACT)
;

-- (8)
IMPORT FROM "D:\Raul\items.del" of del xml from "D:\Raul" INSERT INTO ITEMS (ID,
BRANDNAME, ITEMNAME, SKU, SRP, COMMENTS)
;
```

注意这个脚本和其他相关文件位于随书光盘的 **expressc_book_quicklabs.zip** 文件中。接下来我们逐行解释这段脚本。

1. 删除“mydb”数据库。这一步执行清理。如果“mydb”数据库不存在，会得到错误提示，忽略即可。

2. 使用创建 UTF-8 字符集创建“mydb”数据库。只有 UNICODE 数据库才能为 pureXML 提供支持，因此创建 UNICODE 数据库是必需的。

3. 连接新建立的“mydb”数据库。接下来才可以在数据库中创建对象。

4. 建立“items”表。注意表的最后一列“comments”被定义为 XML 列，使用了新的 XML 数据类型。

5. 建立“clients”表。最后一列同样使用新的 XML 数据类型进行定义。

6. 用 SQL 的 INSERT 语句将 XML 文档插入 XML 列。在 INSERT 语句中 XML 文档被放在一对单引号里当作字符串来传递。

7. 用 IMPORT 命令可以把几个 XML 文档连同关系型数据一起导入到数据库中。第(7)部分导入的数据来自 clients.del 文件（一个有界 ascii 文件），该行同时指明了 clients.del 文件引用的 XML 数据存放何处。

首先来看目录 D:\Raul 下有什么内容（图 16.10），clients.del 文件我们稍后关注。

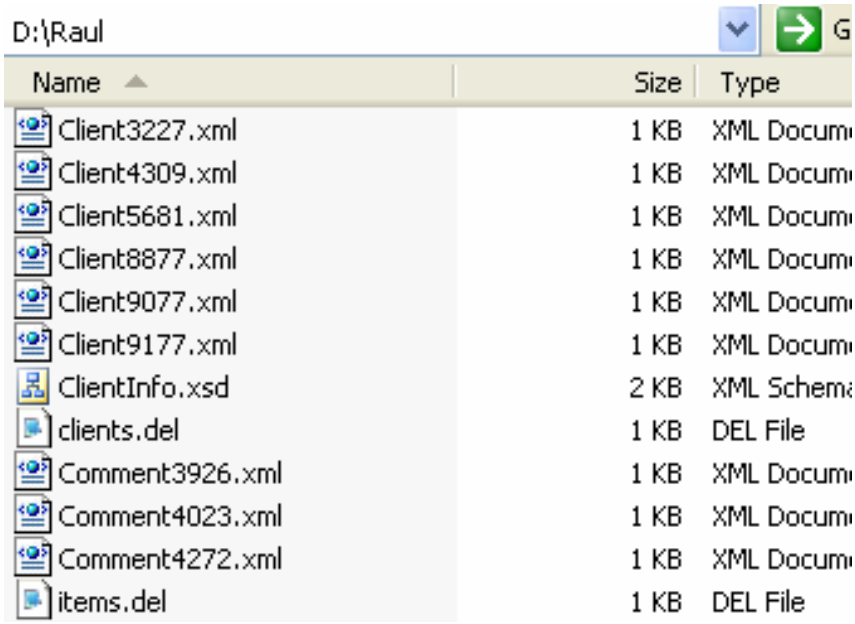


图 16.10 - D:\Raul 目录的内容，里面包含了 del 文件和 xml 文件

这是 clients.del 文件的内容

clients.del

```
3227,Ella Kimpton,Gold,<XDS FIL='Client3227.xml' />,
8877,Chris Bontempo,Gold,<XDS FIL='Client8877.xml'/>,
9077,Lisa Hansen,Silver,<XDS FIL='Client9077.xml' />
9177,Rita Gomez,Standard,<XDS FIL='Client9177.xml'/>,
5681,Paula Lipenski,Standard,<XDS FIL='Client5681.xml' />,
4309,Tina Wang,Standard,<XDS FIL='Client4309.xml'/>
```

在这个文件里，“XDS FIL=”用来指向某一 XML 文档。

运行上面的脚本后，控制中心如图 16.11 所示。

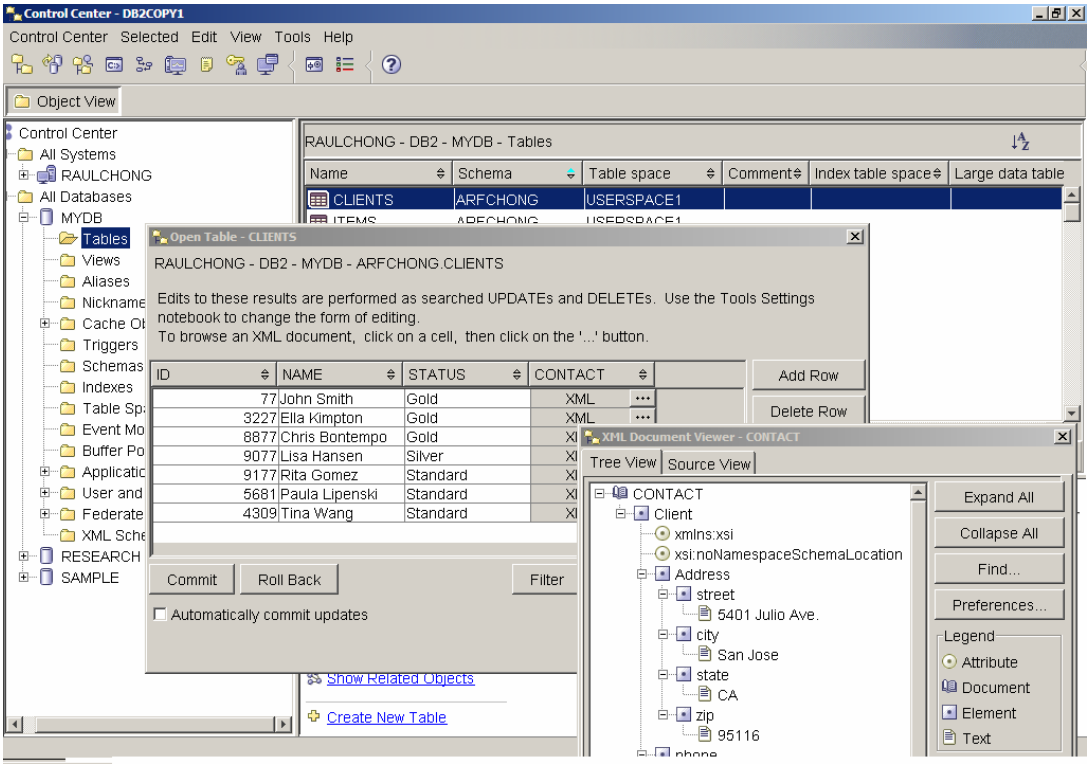


图 16.11 –执行完 table_creation.txt 后的控制中心视图

注意图中显示了 clients 表的内容。最后一列“Contact”是 XML 列。点击有三个点的按钮，将会在另一个窗口显示 XML 文档的内容。图中该窗口位于右下角。

16.3.5 查询 XML 数据

在 DB2 中有两种方式查询 XML 数据：

- 使用带有 XML 扩展的 SQL(SQL/XML)
- 使用 XQuery

无论那一种方式，DB2 都遵循 XML 的国际标准。

SQL/XML 查询 XML 数据

普通的 SQL 语句可以处理行和列。处理完整的 XML 文档 SQL 语句也堪此任；然而，当要获取文档的部分内容时，普通的 SQL 语句就捉襟见肘了。在这种情形下，需要使用带有 XML 扩展的 SQL(SQL/XML)。

表 16.5 列举了符合 SQL2006 标准的部分 SQL/XML 函数。

函数名	描述
XMLPARSE	解析字符或者大对象二进制数据，产生 XML
XMLSERIALIZE	将 XML 值转换为字符或者大对象二进制数据
XMLVALIDATE	根据 XML schema 校验 XML 值的有效性
XMLEXISTS	检测 XQuery 是否返回结果（如由一个或多个项所组成的序列）
XMLQUERY	执行一个 XQuery 并以一个序列返回结果
XMLTABLE	执行一个 XQuery，以关系数据表形式返回结果（如果可能的话）
XMLCAST	XML 类型转换

表 16.5 – SQL/XML 函数

可以使用先前创建的“mydb”数据库测试下面的示例。

例一

这是一个查询示例。设想您要获得居住在某一地区的所有客户的姓名。clients 表在 XML 列里存储了客户的地址和邮编。使用 XMLEXISTS 函数，可以在 XML 列搜索目标邮编，根据条件返回结果集。

```
SELECT name FROM clients
 WHERE xmlexists(
    '$c/Client/Address[zip="95116"]'
    passing clients.contact as "c"
    )
```

第一行 SQL 语句指定要从“clients”表的 name 列获取信息。

在 WHERE 子句中调用 XMLEXISTS 函数，参数指定了 XML 路径表达式，DB2 据此导航到 XML 文档邮编元素并且查找值 95116。

“$c/Client/Address”指定了 DB2 在 XML 文档中定位邮编元素的路径。美元符号($)用来指定变量。passing clients.contact as "c"对变量 c 进行定义。这里，“clients”是表名，“contact”是 XML 列的列名。换言之，XML 文档传递给了变量 c。

DB2 检索“contact”列中的 XML 数据，从根节点“client”向下，通过节点“Address”到节点“zip”，经过目标邮编匹配检查判断该客户是否居住在目标地区。若 XMLEXISTS 函数返回“true”，则 DB2 返回该行的客户姓名。

DB2 9.5 简化了上面的查询：

```
SELECT name FROM clients
 WHERE xmlexists(
    '$CONTACT/Client/Address[zip="95116"]'
    )
```

DB2 自动生成与 XML 列同名的变量。在上例中，CONTACT 变量是 DB2 自动生成的。变量名与 CONTACT 列名相同。

例二

考虑一下如何生成一份列举了所有金牌顾客邮件地址的报告。运行下面的查询：

```
SELECT xmlquery('$c/Client/email' passing contact as "c")
  FROM  clients
  WHERE status = 'Gold'
```

第一行表明我们要从 XML 文档的元素中（不是关系列）提取邮件地址。同上例，“$c”是指向 XML 文档的变量。XMLEXISTS 函数可以用在 WHERE 后面，类似的可以在 SELECT 后面使用 XMLQUERY 函数。

例三

有时您想用表来呈现 XML 数据。XMLTABLE 函数帮助完成这一工作，演示如下。

```
SELECT t.comment#, i.itemname, t.customerID, Message
  FROM items i,
  xmltable('$c/Comments/Comment' passing i.comments as "c"
     columns Comment# integer path 'CommentID',
     CustomerID integer path 'CustomerID',
     Message varchar(100) path 'Message') AS t
```

第 1 行指定了包含在结果集中的列。以变量 t 作为前缀表明该列的值是 XML 文档的元素值。

第 2 行调用 XMLTABLE 函数，参数指定了包含目标数据的 DB2 XML 列，以及目标数据在 XML 文档中的路径。

第 4 行至第 6 行的“columns”语句将 XML 元素映射到结果集中相应各列。映射还指定了 XML 元素值将要转换成的数据类型。示例中，所有 XML 数据都被转换成传统 SQL 数据类型。

例四

现在我们看一个简单的例子，该例子在 XMLQUERY 函数中使用 XQuery FLWOR 表达式。

```
SELECT name, xmlquery(
    ‘for $e in $c/Client/email[1] return $e’
    passing contact as “c”
    )
  FROM clients
  WHERE status = ‘Gold’
```

第 1 行指定在结果集中包含顾客姓名和 XMLQUERY 函数的输出。第 2 行指出“Client”元素的第一个“email”子元素将被返回。第三行指定 XML 数据源(“contact” 列)。第四行告诉我们该列来自“clients”表；第五行指出查询条件是金牌顾客。

例五

此例再一次演示了使用 XQuery FLWOR 表达式的 XMLQUERY 函数的用法，但请注意这一次返回的不仅是 XML，而且还有 HTML。

```
SELECT xmlquery('for $e in $c/Client/email[1]/text()
    return <p>{$e}</p>'
    passing contact as "c")
  FROM clients
  WHERE status = 'Gold'
```

XQuery 的 return 语句确保 XML 输出按需转换。第一行使用 text()函数仅获取金牌顾客的第一个邮件地址的元素值部分。第二行把这个信息放在了 HTML 段标签中。

例六

这个例子使用 XMLELEMENT 函数创建一系列 XML item 元素，每一项都包含 ID、brand name 和 stock keeping unit(SKU)三个子元素，它们的值由“items”表相应列获取。基本上，当把关系数据转化成 XML 数据时都可以使用 XMLELEMENT 函数。

```
SELECT
  xmlelement (name "item", itemname),
  xmlelement (name "id", id),
  xmlelement (name "brand", brandname),
  xmlelement (name "sku", sku)
  FROM items
  WHERE srp < 100
```

上面的查询输出如下:

```
<item>
        <id>4272</id>
        <brand>Classy</brand>
        <sku>981140</sku>
</item>
…
<item>
        <id>1193</id>
        <brand>Natural</brand
        <sku>557813</sku>
</item>
```

使用 XQuery 查询 XML 数据

前一节，我们叙述了如何使用带有 XML 扩展的 SQL 进行 XML 数据查询。一直以来 SQL 都是最主要的查询手段，XPath 也被嵌入其中。本节，我们讨论如何使用 XQuery 查询 XML 数据，XQuery 将会是主要的查询手段，一些例子也使用了嵌入 SQL 的 XQuery (使用“db2-fn:sqlquery”函数)。使用 XQuery 时，我们将会调用几个函数，同时使用 FLWOR 表达式。

例一

返回顾客联系方式的简单 XQuery 示例

```
xquery db2-fn:xmlcolumn('CLIENTS.CONTACT')
```

给 XQuery 表达式加上“xquery”命令前缀，这样 DB2 才能调用 XQuery 解析器，否则 DB2 假定您尝试运行的是 SQL 表达式。**db2-fn:xmlcolumn** 函数从由参数指定的列获取整个 XML 文档。这与下面的 SQL 语句等价，此时获取的是整列内容：

```
SELECT contact FROM clients
```

例二

本例使用 FLWOR 表达式获取客户传真数据

```
xquery
  for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/fax
  return $y
```

第一行调用 XQuery 解析器。第二行告诉 DB2 在 CLIENTS.CONTACT 列包含的传真子元素中进行遍历。每一个传真元素都绑定到变量$y。第三行返回每一次遍历的 “$y”值。

查询的输出与此类似（默认情况下输出可能会包含命名空间，但我们在此没有显示它，否则输出将会跨越多行而变得难以阅读）：

```
<fax>4081112222</fax>
<fax>5559998888</fax>
```

例三

这个例子查询 XML 数据并以 HTML 形式返回结果。

```
xquery
  <ul> {
    for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/Address
    order by $y/zip
    return <li>{$y}</li>
  }
  </ul>
```

示例返回的 HTML 如下所示：

```
<ul>
<li>
<address>
    <street>9407 Los Gatos Blvd.</street>
    <city>Los Gatos</city>
    <state>ca</state>
    <zip>95302</zip>
</address>
</li>
<address>
<street>4209 El Camino Real</street>
    <city>Mountain View</city>
    <state>CA</state>
    <zip>95302</zip>
</address>
</li>
...
</ul>
```

例四

这个例子演示了如何使用 db2-fn:sqlquery 函数将 SQL 嵌入到 XQuery 中。db2-fn:sqlquery 函数执行 SQL 查询并返回选中的 XML 数据。传递给 db2-fn:sqlquery 函数的 SQL 查询必须只返回 XML 数据。返回的 XML 数据可以被 XQuery 进一步处理。

```
xquery
  for $y in
  db2-fn:sqlquery(
    'select comments from items where srp > 100'
  )/Comments/Comment
  where $y/ResponseRequested='Yes'
  return (
    <action>
      {$y/ProductID
       $y/CustomerID
       $y/Message}
    </action>
  )
```

示例中，SQL 查询根据条件“srp”列的值大于 100 过滤行。在经过过滤的行中，XML 列“comments”会被选出来。下一行 XQuery（或者说 XPath）用来遍历 Comment 子元素。

注意：DB2 不区分大小写，它将所有表名、列名都当作大写来处理，而 XQuery 是大小写敏感的。示例中使用的都是 XQuery 接口函数，因此传递给函数的表名和列名都应该是大写的。传递小写的对象名可能会导致一个“未定义的对象名”错误。

16.3.6 使用 SQL/XML 执行联合操作

这一节讨论如何在不同表的两个 XML 列之间或者 XML 列与关系数据列之间执行联合操作。假定已用如下语句创建了两张表：

```
CREATE TABLE dept (unitID CHAR(8), deptdoc XML)

CREATE TABLE unit (unitID  CHAR(8) primary key not null,
                   name    CHAR(20),
                   manager VARCHAR(20),
                   ...
                   )
```

可以选择如下两种方法之一执行联合操作：

方法一：

```
SELECT u.unitID
  FROM dept d, unit u
  WHERE XMLEXISTS (
    '$e//employee[name = $m]'
    passing d.deptdoc as “e”, u.manager as “m”)
```

name 是表“dept”中“deptdoc”列 XML 文档的子元素，同时 manager 是表 unit 的列，语句在 dept 表和 unit 表上执行了联合操作。

方法二：

```
SELECT u.unitID
  FROM dept d, unit u
  WHERE u.manager = XMLCAST(
    XMLQUERY('$e//employee/name '
      passing d.deptdoc as “e”)
    AS char(20))
```

在这个方法中，关系列位于联合操作表达式的左侧。此情形允许使用关系索引替代 XML 索引。

16.3.7 使用 XQuery 执行联合操作

假定已经创建了两张表：

```
CREATE TABLE dept(unitID CHAR(8), deptdoc XML)
CREATE TABLE project(projectDoc XML)
```

如果使用 SQL/XML，查询如下所示：

```
SELECT XMLQUERY (
 '$d/dept/employee' passing d.deptdoc as "d")
 FROM dept d, project p
 WHERE XMLEXISTS (
  '$e/dept[@deptID=$p/project/deptID]'
    passing d.deptdoc as "e", p.projectDoc as "p")
```

使用 XQuery 的相同查询如下：

```
xquery
 for $dept in db2-fn:xmlcolumn("DEPT.DEPTDOC")/dept
 for $proj in db2-fn:xmlcolumn("PROJECT.PROJECTDOC")/project
 where $dept/@deptID = $proj/deptID
 return $dept/employee
```

第二种方法更容易解释：变量"$dept"保存了"dept"表中"deptdoc"列的 XML 文档。变量"$proj"保存了"project"表中"projectdoc"列的 XML 文档。语句第四行在第一个 XML 文档的属性与第二个 XML 文档的元素间执行联合操作。

16.3.8 更新与删除操作

有两种方式可以对 XML 数据执行更新和删除操作：

- 使用 SQL UPDATE 和 DELETE 语句
- 使用 TRANSFORM 表达式

第一种方式，更新和删除是文档级的，即整个 XML 文档都被更新的文档替换。例如，在下面的例子中我们想改变的仅仅是<state>元素，然而实际上整个文档已经被替换。

```
UPDATE clients SET contact=(
 xmlparse(document
  '<Client>
    <address>
     <street>5401 Julio ave.</street>
     <city>San Jose</city>
     <state>CA</state>
     <zip>95116</zip>
    </address>
    <phone>
     <work>4084633000</work>
     <home>4081111111</home>
     <cell>4082222222</cell>
    </phone>
    <fax>4087776666</fax>
    <email>newemail@someplace.com</email>
  </Client>')
 )
 WHERE id = 3227
```

第二种方式用 TRANSFORM 表达式执行子文档更新，效率高很多。这种方法可以替换、插入、删除或者重命名 XML 文档节点。同样也可以更改节点值而不用替换节点本身，典型的应用便是更改元素或者属性值，这类更新相当常用。这是 DB2 9.5 新添加的特性。

TRANSFORM 表达式是 XQuery 语言的一部分，通常可以在任何使用 XQuery 的地方使用它，譬如在 FLWOR 表达式或者 SQL/XML 语句的 XMLQUERY 函数中。当需要修改 XML 列中的一个 XML 文档时，将 TRANSFORM 表达式用在 SQL UPDATE 语句中，这是最典型的应用。

TRANSFORM 表达式的语法如下：

```
>>-transform--| copy clause |--| modify clause |--| return clause |-><

copy clause

       .-,----------------------------------------.
       V                                        |
|--copy----$VariableName--:=--CopySourceExpression-+------------|

modify clause

|--modify--ModifyExpression-------------------------------------|

return clause

|--return--ReturnExpression--------------------------------------|
```

Copy 语句用来把要处理的 XML 文档赋值给变量。在 modify 语句中，可以使用 insert, delete, rename 或者 replace 表达式执行操作。这些表达式可以对 XML 文档执行更新。例如，使用 insert 表达式向文档添加新的节点，使用 delete 表达式删除节点，使用 rename 表达式重命名元素或者属性，使用 replace 表达式把已有节点替换为新的节点。表达式的替换值只能用以更改元素或者属性值。Return 语句返回 transform 表达式的值。

这是一条 UPDATE 语句，语句使用了 tranform 表达式。

```
(1)-- UPDATE customers
(2)-- SET contactinfo = xmlquery( 'declare default element namespace
(3)--                          "http://posample.org";
(4)--    transform
(5)--    copy $newinfo := $c
(6)--          modify do insert <email2>my2email.gm.com</email2>
(7)--                   as last into $newinfo/customerinfo
(8)--    return $newinfo' passing contactinfo as "c")
(9)-- WHERE id = 100
```

在上面的例子中，第(1), (2), (9)行使用的依然是 SQL UPDATE 语句的语法。第(2)行的 XMLQUERY 函数调用第(4)行的 transform 表达式。(4)至(8)行是 transform 表达式，它向 XML 文档插入一个新节点 email2。注意 DB2 不支持通过视图更新 XML 文档的元素。

从数据表删除整个 XML 文档一如在 SQL/XML 中使用 SELECT 语句一样简单。使用 SQL DELETE 语句同时指定需要的条件即可。

16.3.9 XML 索引

在 XML 文档中，可以为元素、属性或者值（文本节点）创建索引。示例如下，假定已经创建下面的表：

```
CREATE TABLE customer(info XML)
```

同时假定这是存储的 XML 文档之一：

```
<customerinfo Cid="1004">
  <name>Matt Foreman</name>
  <addr country="Canada">
    <street>1596 Baseline</street>
    <city>Toronto</city>
    <state>Ontario</state>
    <pcode>M3Z-5H9</pcode>
  </addr>
  <phone type="work">905-555-4789</phone>
  <phone type="home">416-555-3376</phone>
  <assistant>
    <name>Peter Smith</name>
    <phone type="home">416-555-3426</phone>
  </assistant>
</customerinfo>
```

1) 这条语句为属性“Cid”创建索引

```
CREATE UNIQUE INDEX idx1 ON customer(info)
  GENERATE KEY USING
  xmlpattern '/customerinfo/@Cid'
  AS sql DOUBLE
```

2) 这条语句为“name”元素创建索引

```
CREATE INDEX idx2 ON customer(info)
  GENERATE KEY USING
  xmlpattern '/customerinfo/name'
  AS sql VARCHAR(40)
```

3) 这条语句为所有“name”元素创建索引

```
CREATE INDEX idx3 ON customer(info)
  GENERATE KEY USING
  xmlpattern '//name'
  AS sql VARCHAR(40);
```

4) 这条语句为所有文本节点（值）创建索引。这种做法并不提倡，因为在 update, delete 和 insert 操作中维护索引代价高昂，同时索引也未免太大了。

```
CREATE INDEX idx4 ON customer(info)
  GENERATE KEY USING
  xmlpattern '//text()'
  AS sql VARCHAR(40);
```

实验 #14 - SQL/XML 和 XQuery

实验目标

至此，您已经看到为数不少的 SQL/XML 和 XQuery 语法示例，DB2 命令编辑器和 IBM Data Studio 也已经做过介绍。本实验中，您可以测试您对 SQL/XML 和 XQuery 掌握的程度，同时获得这些工具的动手经验。请使用“mydb”数据库完成这些练习，本章前面解释了创建这个数据库的 **table_creation.txt** 脚本文件。

实验过程

1. 如本章前面讨论的那样，创建“mydb”数据库并且加载 XML 数据。

2. 使用命令编辑器或者 IBM Data Studio：
   a)用两种方法获取 items 表中 XML 文档的所有 comments，注意仅使用 XQuery。
   b)为什么下面这条 SQL 语句不会返回与 XQuery 同样的结果？
   SELECT comments FROM items
   c)返回满足 XML 文档中 ResponseRequested 元素值为 No 的 ID 和 BRANDNAME 记录。

解答：

2a)

```
xquery db2-fn:xmlcolumn('ITEMS.COMMENTS')
xquery db2-fn:sqlquery("select comments from items")
```

2b)当值不存在时 SQL 返回 NULL 而 XQuery 什么也不返回，因而输出不同。

2c)

```
SELECT  id, brandname FROM items WHERE
XMLEXISTS('$c/Comments/Comment[ResponseRequested="No"]'
passing ITEMS.COMMENTS as "c
```

17

第 17 章 – 使用 Java、PHP 和 Ruby 进行数据库应用开发

本章介绍使用 Java、PHP、Ruby on Rails 进行 DB2 数据库服务器应用程序的开发基础知识。本章的目的不在于讲解这些语言，而是提供与 DB2 一起使用这些语言直接相关的信息。

注意：
更多关于这个主题的更多信息，请观看视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:4402

17.1 Java 应用程序开发

IBM DB2 JDBC 驱动程序（也称作 JCC 驱动程序）已经为所有平台上的 DB2 服务器做了优化。db2jcc.jar（com.ibm.db2.jcc）文件包括类型 2 和类型 4 驱动程序。任何 DB2 客户端都包含 db2jcc.jar 文件，该文件（IBM DB2 Driver for JDBC and SQLJ）也可以从 DB2 Express-C 网站单独获得(ibm.com/db2/express)。

17.1.1 JDBC 类型 2 驱动程序

JDBC 类型 2 驱动需要在运行 JDBC 应用程序的机器上安装 DB2 客户端。使用类型 2 驱动的 JDBC 应用程序结构简图如图 17.1 所示。

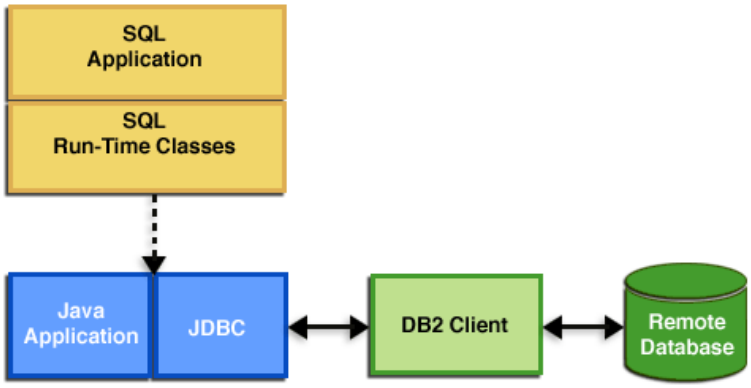


图 17.1 – JDBC 类型 2 驱动程序

图 17.2 展示了如何使用 JDBC 类型 2 驱动建立数据库连接。注意 URL 中不包括主机名和端口信息，这些信息将从 DB2 客户端提取。

```
...
  public static final String DB_URL = "jdbc:db2:sample";
  Properties connectProperties = new Properties();
  connectProperties.put("user", "db2admin");
  connectProperties.put("password", "ibmdb2");
  Connection connection = null
  try
  {
     Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
     connection = DriverManager.getConnection(url, connectProperties)
  }
  catch (Exception e)
  throw e;
  }
...
```

图 17.2 –使用 JDBC 类型 2 驱动程序建立数据库连接

17.1.2 JDBC 类型 4 驱动程序

JDBC 类型 4 驱动不需要 DB2 客户端来连接 DB2 数据库服务器。图 17.3 是使用类型 4 驱动的 JDBC 应用程序结构。

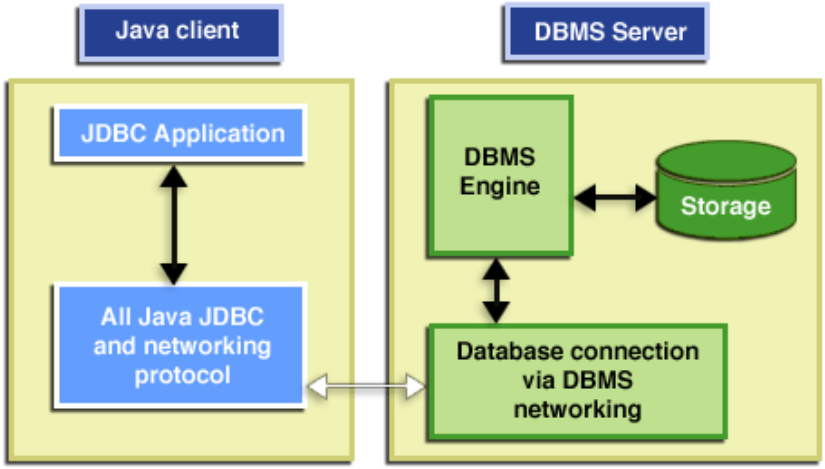


图 17.3 –JDBC 类型 4 驱动程序

图 17.4 说明了如何使用 JDBC 类型 4 驱动程序建立数据库连接。注意 URL 包括主机名和端口信息。

```
...
  public static final String DB_URL = "jdbc:db2://server1:50000/sample";
  Properties connectProperties = new Properties();
  connectProperties.put("user", "db2admin");
  connectProperties.put("password", "ibmdb2");
  Connection connection = null
  try
  {
     Class.forName("com.ibm.db2.jcc.DB2Driver").newInstance();
     connection = DriverManager.getConnection(url,connectProperties)
  }
  catch (Exception e)
    throw e;
  }
...
```

图 17.4 – 使用 JDBC 类型 4 驱动程序建立数据库连接

17.2 PHP 应用程序开发

PHP 是开源的，平台无关的专为 web 应用开发设计的脚本语言。今天它是世界上部署最为广泛的 web 语言之一。PHP 的风行建立在如下语言特征之上：

- 较平缓的学习曲线；快速可循环的开发周期
- 健壮、高性能和可扩展性
- 稳定、安全
- J2EE™ and .NET 平台之外的另一 Web 开发利器
- 复杂环境/系统的易集成性
- 现实中广泛部署的检验
- 完善并且活跃的社区支持

PHP 是 LAMP (Linux, Apache HTTP Server, MySQL, **PHP** / Perl / Python)组合的一部分，LAMP 是开源的网络技术组合，通常在每月交纳合适的费用后由网络服务提供商提供。

17.2.1 DB2 为 PHP 提供的连接选项

IBM 通过两项扩展支持 PHP 应用程序访问 DB2 数据库。

ibm_db2:

ibm_db2 扩展提供了过程化应用程序编程接口来满足创建、读取、更新和写入数据库操作以及大量数据库元数据访问的需要。编译后可用于 PHP4 或 PHP5。在 Apache2.0 许可协议下该扩展可以从 PECL 库获取。IBM 开发了它并一直提供支持。它全面支持存储过程和大对象，并且由于已经为 DB2 做过优化所以它的速度令人满意。

PDO_ODBC：

PDO_ODBC 是用于 PHP 数据对象扩展（PDO）的驱动，它通过由 PHP5.1 引入的标准面向对象数据库接口提供对 DB2 数据库的访问。该扩展可以针对 DB2 库直接编译。它为 PHP 提供了标准数据访问接口。同时它是快速、轻量和面向对象的。PDO_ODBC 扩展使用 DB2 库提供本地访问并且已经被内置到 PHP5.1 中。更多信息，请访问以下网站：

- http://pecl.php.net/package/pdo
- http://pecl.php.net/package/PDO_ODBC

连接未列入目录的 DB2 数据库

如何使用之前讲解的两种扩展连接 DB2 数据库示于列表 17.1。

```
$host = 'localhost';
$port = 50000;
$DSN = "DRIVER={IBM DB2 ODBC DRIVER}; PORT=$port;
      HOSTNAME=$host; DATABASE=$database; PROTOCOL=TCPIP;
      USER=$user; PWD=$password";

-- If using the ibm_db2 extension --
$uconn = db2_connect($DSN, null, null);

-- If using the PDO_ODBC extension --
try {
  $uconn = new PDO("odbc:$DSN", null, null);
}
```

```
catch (PDOException $e) { print $e->errmsg(); }
```

列表 17.1 –连接到未列入目录的 DB2 数据库

列表 17.2 是一个使用 ibm_db2 扩展的简单 PHP 应用程序示例。

```
<?php
$sql = "SELECT name, breed FROM ANIMALS WHERE weight < ?";
$conn = db2_connect($database, $user, $password);
$stmt = db2_prepare($conn, $sql);
$res = db2_execute($stmt, array(10));
while ($row = db2_fetch_assoc($stmt)) {
   print "{$row['NAME']} is a {$row['BREED']}.\n";
}
?>
```

列表 17.2 – 使用 ibm_db2 扩展的简单 PHP 应用程序示例

为 PHP 配置 ibm_db2

在 Linux 或者 UNIX 环境您可能需要修改 php.ini 文件：

```
extension=ibm_db2.so
ibm_db2.instance_name=<instance name>
```

在 Windows 中，php.ini 文件修改如下：

```
extension=php_ibm_db2.dll
```

作为替代方案，可以下载并安装 **Zend Core for IBM** 应用程序套件，使用它可以轻松完成这些配置工作。该套件将在下一节讲述。

17.2.2 Zend Core for IBM

Zend Core 是无缝集成的全面的商业级 web 应用程序开发和生产环境。它为运行 PHP 应用带来可靠性，开发效率和灵活性。可以在下面的站点免费下载它:
http://ibm.com/software/data/info/zendcore

Zend Core for IBM 安装 DB2 和 IDS 客户端、一个可选的 Apache HTTP 服务器、PHP5 和流行的 PHP 扩展，包括 ibm_db2 和 PDO_INFORMIX。在该套件中还可以选择安装 DB2 Express-C 服务器、IBM Cloudscape™服务器、完整的 PHP 手册和 DB2 示例应用程序。这是一个易于使用和配置 PHP 开发环境，详情见图 17.5、17.6 和 17.7。

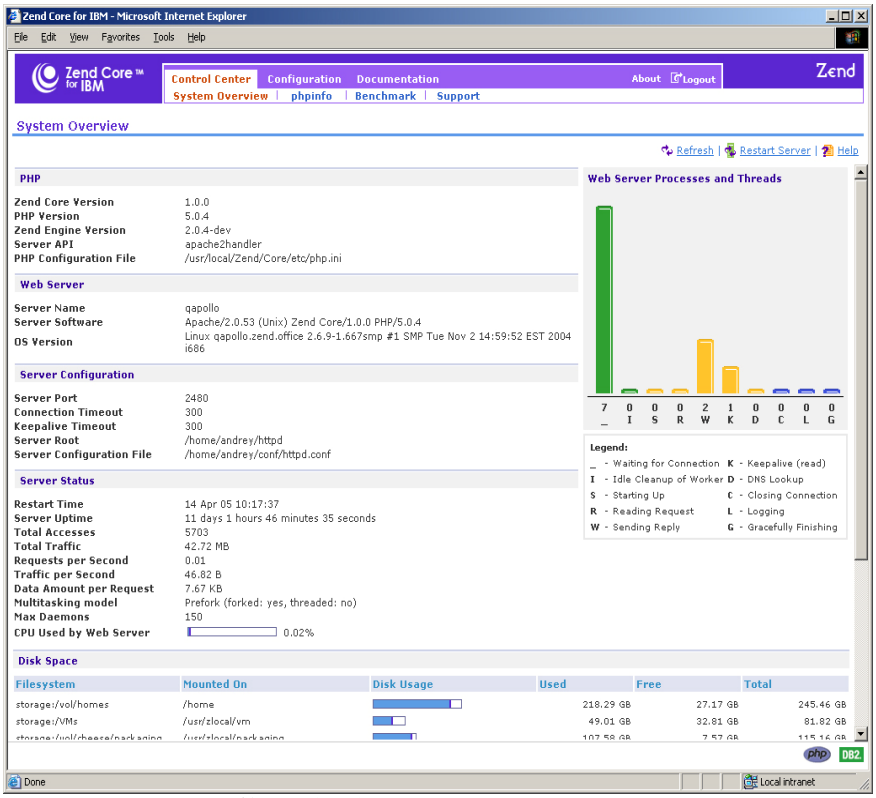


图 17.5 - Zend Core 管理和控制界面

图 17.6 - Zend Core PHP 配置界面

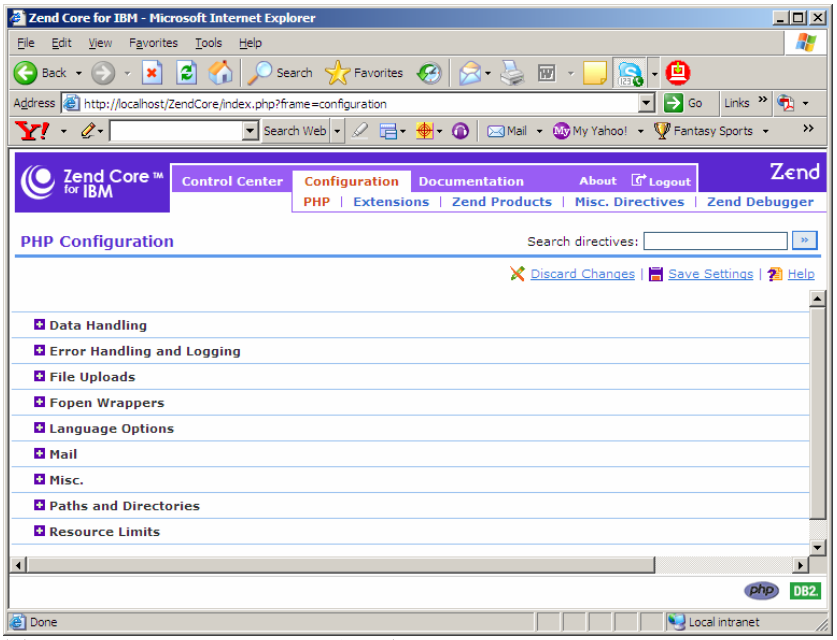


图 17.7 - Zend Core PHP 配置界面

17.3 Ruby on Rails 应用程序开发

Ruby 是面向对象、动态、跨平台的脚本语言。它提供了快速应用程序开发，并且包含了丰富的库。1995 年 Ruby 由 Yukihiro Matsumoto (“Matz”)开发，它简单而且有趣。

Rails 用 Ruby 写成，是一套数据库支持的 web 应用程序完整框架。它采用了模型-视图-控制(MVC)架构，带来了很高的开发效率并且易于使用。Rails 是 2004 年以来出现的开发效率最高的 web 框架之一，它由 David Heinemeier Hansson 开发。

17.3.1 Startup Toolkit for DB2 on Rails

IBM 意识到了 Ruby on Rails 在开发社区的重要性；因此开发了 **Startup Toolkit for DB2 on Rails**。这一集成的安装工具会建立起一个完整的 DB2 Ruby on Rails 开发环境。可以从 IBM alpha-Works 网站免费下载和使用这个工具包：

Startup Toolkit for DB2 on Rails：

- 包括一个集成安装文件
- 使 Ruby on Rails 易于安装和配置
- 安装 DB2 Express – C 9 和各种工具
- 包括 IBM 开发的一个 DB2 Ruby 驱动程序和一个 DB2 Rails 适配器
- 包括各种示例和教程

附录 A — 排除故障

此附录要讨论的时如何检测并维护使用 DB2 中可能遇到的问题。图 A.1 展示了发生问题时通常采取的措施。

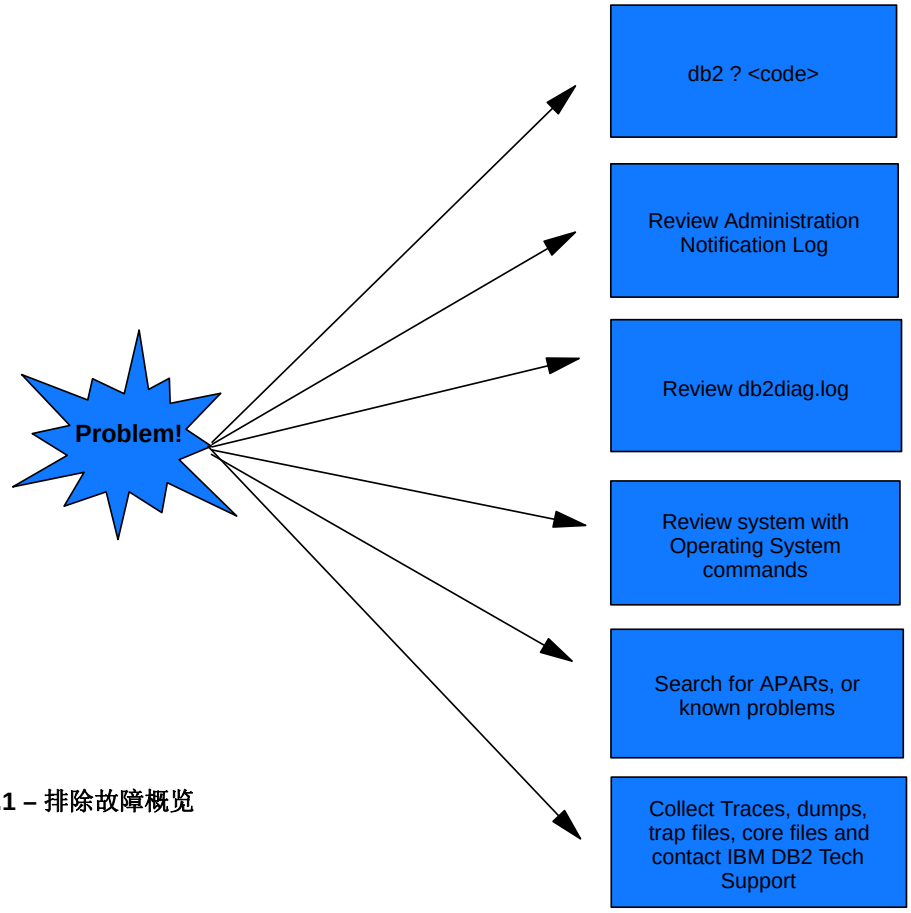


图 A.1 – 排除故障概览

注意;
更多有关排除故障的信息，请参见以下视频：
http://www.channeldb2.com/video/video/show?id=807741:Video:4462

A.1 查找错误代码的更多信息

想了获取更多有关故障代码的信息，在命令编辑器处输入该代码，并在前面加上问号。如图 A.2 所示

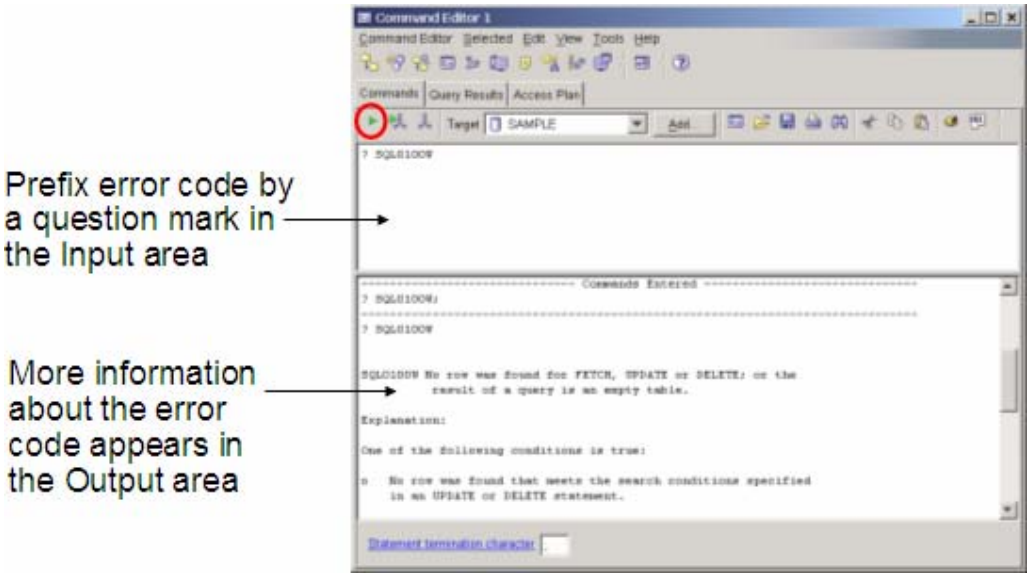


图 A.2 – 查找 DB2 错误代码的信息

问号（？）会调用 DB2 的帮助命令。以下是一些调用它来获取帮助的例子。例如：您想知道 SQL 错误代码“-104”的信息，可使用如下语句查询（以下例子都是等价的）：

```
db2 ? SQL0104N
db2 ? SQL104N
db2 ? SQL-0104
db2 ? SQL-104
db2 ? SQL-104N
```

A.2 SQLCODE 与 SQLSTATE

SQLCODE 是在每一条 SQL 语句执行后收到的代码。这些值的意义如下：

SQLCODE = 0; the command was successful 该命令执行成功。

SQLCODE > 0; the command was successful， but returned a warning 该命令执行成功，但返回了一个警告。

SQLCODE < 0; the command was unsuccessful and returned an error 该命令没有成功执行，并且返回了一个错误。

SQLSTATE 是一个遵守 ISO/ANSI SQL92 标准的长为 5 个字符的字符串。前两个字符就是 SQLSTATE 类代码:

- 类代码 00 表示该命令执行成功。
- 类代码 01 意味着有一个警告。
- 类代码 02 意味着有未检测到的情况。
- 其它的类代码都被认为是错误。

A.3 DB2 管理通知日志

DB2 管理通知日志提供了关于故障点的错误诊断信息。在 Linux/UNIX 平台上，管理通知日志是一个名为<实例名>.nfy 的文本文件(如”db2inst.nfy”)。 在 Windows 平台，所有的管理通知消息都被写入 Windows 事件日志（Windows Event Log）。

DBM 配置参数 notifylevel 允许管理员指定要记录的信息的级别:

0 --不捕获任何管理通知消息
1 --致命的或不可恢复的错误
2 --要求立即采取措施
3 --重要信息，不要求立即响应(默认级别)
4 --普通消息

A.4 db2diag.log

db2diag.log 提供了比 DB2 管理通知日志更多的信息。它通常由 IBM DB2 技术支持或者经验丰富的数据库管理员使用。在 db2diag.log 中包含了如下内容：

- 出现错误的 DB2 代码的位置
- 应用程序标识符，它能让您将事件记录与服务器端和客户端的应用程序相对应
- 诊断信息（以“DIA”开头），提供了对错误的解释说明
- 其它的支持信息，例如 SQLCA 数据结构以及指向外部堆或者异常捕捉文件的指针

在 Windows 系统中，db2diag.log 默认保存在下面的路径：

```
C:\Program Files\IBM\sqllib\<instance name>\db2diag.log
```

在 Linux/UNIX 系统中，db2diag.log 默认保存在下面的路径：

```
/home/<instance_owner>/sqllib/db2dump/db2diag.log
```

诊断信息文字的长短和详细程度由 dbm cfg 的 DIAGLEVEL 参数决定，参数值可取 0～4，0 表示最短信息记录，而 4 表示最长最详细的信息记录。默认等级为 3。

A.5 CLI 追踪

对于 CLI 和 Java 应用程序，您可以打开 CLI 追踪来捕获和解决故障。在承载应用程序的服务器端修改 db2cli.ini 文件可以打开 CLI 追踪。典型的 db2cli.ini 文件如下所示：

```
[common]
trace=0
tracerefreshinterval=300
tracepathname=/path/to/writeable/directory
traceflush=1
```

当然也可以使用底层的追踪（db2trc），但是通常这只对 DB2 技术支持人员有用。

A.6 DB2 缺陷与补丁

有时候您遇到的问题可能是由 DB2 的缺陷引起，IBM 会定期发布补丁包来修复这些缺陷（APARs）。补丁包的文档描述了对应补丁包所修复的缺陷。在开发新应用时，我们建议您安装好最新的补丁包。您可以用两种方式查看当前的版本和补丁包信息：1、在控制中心中点击“Help”菜单，然后选择“About”项；2、在命令行窗口输入“db2level”命令。注意：要获得 DB2 Express—C 的补丁包和 IBM DB2 技术支持，必须购买 12 个月使用许可证。

参考资源

网站

1. DB2 Express-C 网站：
   www.ibm.com/db2/express
   可以从该网站上面下载 DB2 Express-C servers、DB2 clients、DB2 drivers、用户手册，也可以查阅开发团队 blog，注册邮件列表等等。

2. DB2Express 论坛：
   www.ibm.com/developerworks/forums/dw_forum.jsp?forum=805&cat=19
   当您在用户手册上找不到答案时，请使用该论坛来张贴技术问题。

3. DB2 信息中心（DB2 Information Center ）
   http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp
   信息中心提供在线用户手册，它们时刻保持更新。

4. developerWorks
   http://www-128.ibm.com/developerworks/db2
   这个网站免费为开发人员和数据库管理员提供优质的资源，如技术文章、使用指南等等。

5. alphaWorks
   http://www.alphaworks.ibm.com/
   在这个网站能找到 IBM 的新兴技术，它们都是 IBM 当前研究的最新技术

6. planetDB2
   www.planetDB2.com
   这里收集了大量贡献者关于 DB2 的博客文章

7. DB2 技术支持（DB2 Technical Support）
   http://www.ibm.com/software/data/db2/support/db2_9/
   如果您购买了 DB2 Express-C 的 12 个月使用许可证，您可以在这个网站上下载补丁。

8. ChannelDB2
   ChannelDB2 是 DB2 社区的社会网络，它提供了与 DB2 相关的视频、demo、博客、播客、讨论、其它资源等。涵盖的范围包括 Linux、UNIX、Windows、z/OS、i5/OS 等方面。
   http://www.ChannelDB2.com/

书籍

1. 免费红皮书：《DB2 Express-C: The Developer Handbook for XML， PHP， C/C++，Java， and .NET》
   Whei-Jen Chen， John Chun， Naomi Ngan， Rakesh Ranjan， Manoj K. Sardana，August 2006 - SG24-7301-00
   http://www.redbooks.ibm.com/abstracts/sg247301.html?Open

2. 《Understanding DB2 – Learning Visually with Examples V9.5》
   Raul F. Chong， et all. January 2008
   ISBN-10: 0131580183

3. 《DB2 9: pureXML overview and fast start》
   Cynthia M. Saracco， Don Chamberlin， Rav Ahuja June 2006 SG24-7298
   http://www.redbooks.ibm.com/abstracts/sg247298.html?Open

4. 《DB2® SQL PL: Essential Guide for DB2® UDB on Linux™， UNIX®， Windows™，i5/OS™， and z/OS®》第二版
   Zamil Janmohamed， Clara Liu， Drew Bradstock， Raul Chong， Michael Gao， Fraser McArthur， Paul Yip
   ISBN: 0-13-100772-6

5. 免费红皮书：《DB2 pureXML Guide》
   Whei-Jen Chen， Art Sammartino， Dobromir Goutev， Felicity Hendricks， Ippei Komi，Ming-Pang Wei， Rav Ahuja， Matthias Nicola. August 2007
   http://www.redbooks.ibm.com/abstracts/sg247315.html?Open

6. 《Information on Demand - Introduction to DB2 9 New Features》
   Paul Zikopoulos， George Baklarz， Chris Eaton， Leon Katsnelson
   ISBN-10: 0071487832
   ISBN-13: 978-0071487832

7. 红皮书：《Developing PHP Applications for IBM Data Servers》
   Whei-Jen Chen， Holger Kirstein， Daniel Krook， Kiran H Nair， Piotr Pietrzak
   May 2006 - SG24-7218-00
   http://www.redbooks.ibm.com/abstracts/sg247218.html?Open

联系 email

DB2 Express-C 邮箱： db2x@ca.ibm.com
DB2 校园计划邮箱： db2univ@ca.ibm.com

《DB2 Express-C 快速入门》 并不是一本十分容易的书。 阅读本书可以：

- **了解关于DB2 Express-C的所有信息**
- **理解DB2 体系结构、工具、安全性**
- **学习如何管理DB2数据库**
- **编写SQL、XQuery、存储过程**
- **使用DB2开发数据库应用程序**
- **亲身实践每个练习**
- **准备DB2 on Campus考试**

XML在应用程序集成、Web2.0、SOA中的迅速采用，推动了复合数据库的革新。来自IBM的DB2 Express-C是一款免费、无限制的复合数据库，它能够同时轻松管理XML和传统的关系型数据。免费意味着DB2 Express-C可以免费下载、可以使用它免费创建应用程序、免费配置到产品中、也可以随着您的解决方案免费发放。同时、DB2不会人为地添加任何数据库大小、数据库数量、用户数量上的限制。

DB2 Express-C能够运行在Windows和Linux上，并且为各种编程语言（包括C/C++、Java、.NET、PHP、Perl、Ruby）提供了应用程序驱动。同时，DB2提供可选的、低费用的订购，以提供额外的需要。如果您需要更大的可扩展性或者更先进的功能，您能够无缝地将基于DB2 Ecpress-C的应用程序配置到其它的DB2版本（如DB2企业版）上。

免费的DB2适合于开发人员、技术顾问、独立软件开发商（ISV）、数据库管理员以及其他想开发、测试、配置、分发数据库应用程序的人员。请加入到成长中的DB2 Express-C社区，领略DB2 Express-C的强大力量，开始体验如何创建下一代应用程序以及创新的解决方案。