



VMware® 基础和入门

信息管理云计算能力中心

IBM（加拿大）研究院

目录

1. VMWARE 基础和入门	2
2. 如何获得 VMWARE 软件?	3
3. 解压缩影像	4
4. 使用 VMWARE 虚拟机	4
4.1 在 VMWARE 中打开虚拟机.....	4
4.2 启动虚拟机.....	5
4.3 登录到虚拟机中并接受许可协议.....	6
4.4 打开终端窗口.....	8
4.5 关闭终端窗口.....	10

1. VMware 基础和入门

VMware® Player 和 VMware Workstation 在IT行业中对于测试台和开发者环境而言是同义词。虽然就此具体目的而言还有许多其他功能，但它特别有利于在任何人的计算机上发布含有最新 DB2® 9.7 及 WebSphere® 应用服务器技术的“安装即运行”的 Linux® 系统 – 无论这种计算机是笔记本、台式机或服务器。

VMware 影像可以为简单的演示及教育目的而部署，它也可以成为在给定的环境中自行进行开发及试验的基础。

什么是 VMware 影像?

VMware 能够在基于 Intel® 或 AMD™ 处理器的系统上的现有操作系统中提供一种虚拟的计算机环境。这种虚拟的计算机具有所有通常的组件，例如 CPU、内存和磁盘以及网络、USB 驱动器，甚至声音。CPU 和内存只是基础性的操作系统所提供的现有资源（你可以把它们视为随同“vmware...”启动的进程。磁盘是不同的。对于宿主操作系统而言，它们显示为一系列能够在任何系统之间复制的文件 – 甚至可以在 Windows® 和 Linux 之间复制。这些虚拟的磁盘文件构成了影像的大部分内容，而对虚拟机的实际描述文件则非常小。

下面将介绍如何获得 VMware Player。随后再向你介绍如何启动 VMware 影像，以便用于本技术部分所采用的亲自动手实验室。

2. 如何获得 VMware 软件？

打开 web 浏览器并访问 www.vmware.com
 点击“下载”链接。在页面的右上角寻找“下载”链接。

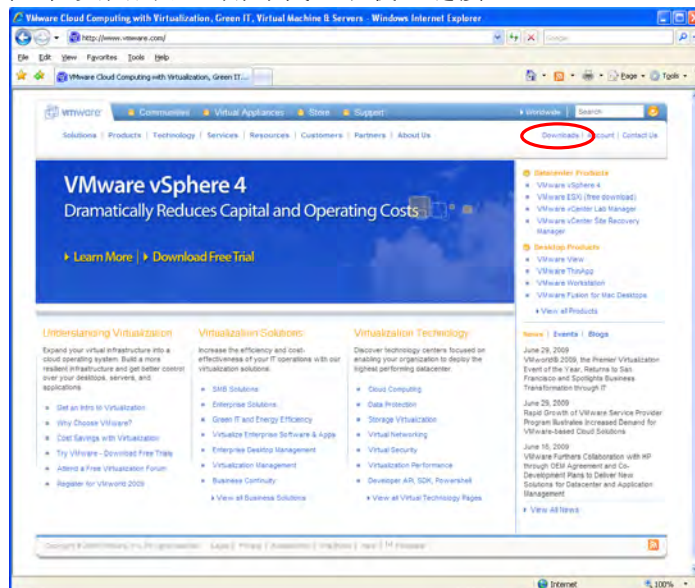


图 1 – 获得 VMware 软件

点击“台式机下载”选项卡。

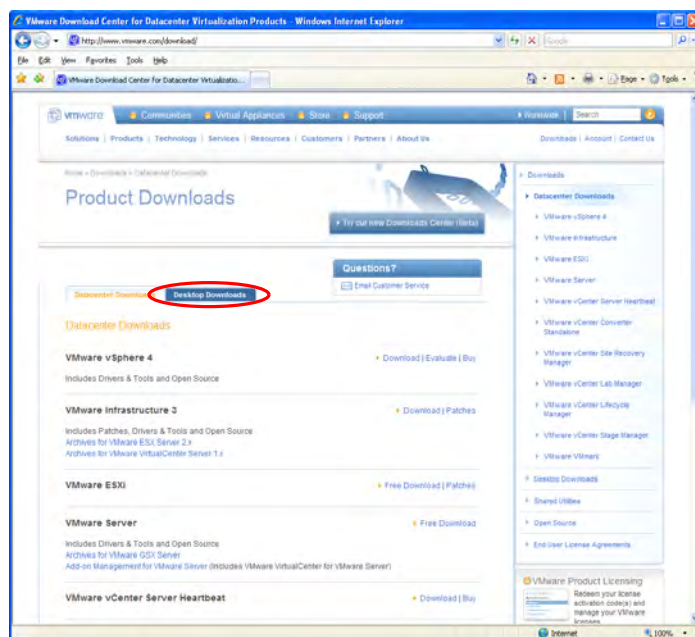


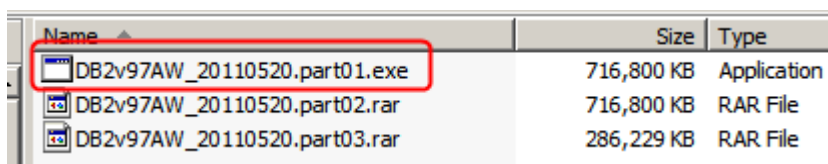
图 2 – 获得 VMware 软件（续）

点击你打算选择的磁盘。我们建议选择 VMware Player 或 VMware Workstation。根据屏幕提示完成注册和下载。

3. 解压缩影像

该影像是由一系列自解压的 rar 文件构成的。为便于处理，文件被压缩多个为 700MB 大小的卷。把所有的卷下载至同一目录中。

双击可执行文件（下图中突出显示的）并选择目标文件夹。



Name	Size	Type
DB2v97AW_20110520.part01.exe	716,800 KB	Application
DB2v97AW_20110520.part02.rar	716,800 KB	RAR File
DB2v97AW_20110520.part03.rar	286,229 KB	RAR File

4. 使用 VMware 虚拟机

4.1 在 VMware 中打开虚拟机

可以用下述方式之一启动 VMware 虚拟机：

- 在 Windows Explorer 或 Linux 文件浏览器中双击文件 **“DB2 Express-C 9.7 32-bit.vmx”**。

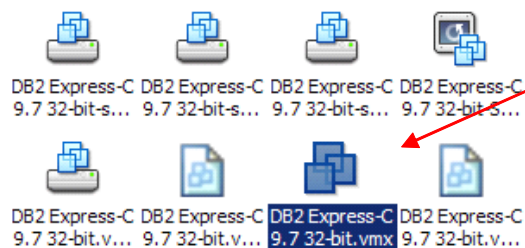


图 3 – 通过双击 vmx 文件启动虚拟机

或者：

- 在 VMware 控制台中通过文件 > 打开... 图标选择该文件

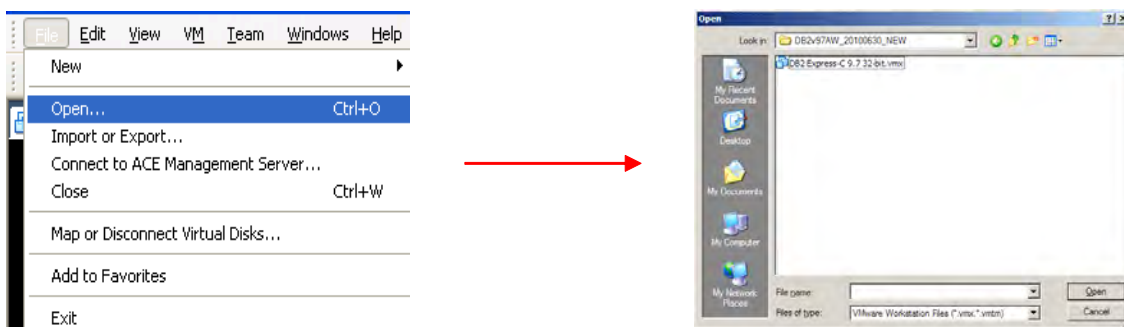


图 4 – 利用文件 -> 打开启动虚拟机

以上任何方法都能够得出下面所示的屏幕：

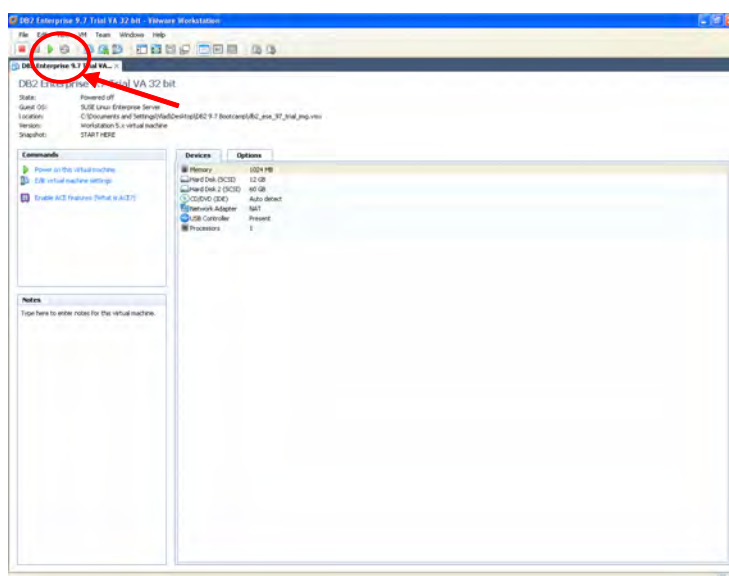



图 5 – 启动虚拟机

4.2 启动虚拟机

下一步，点击左上侧的“开机”  按钮（上图中红圈所示），即可以启动该影像。

该系统将会像任何其他 Linux 系统一样启动，并进入下图所示的状态：

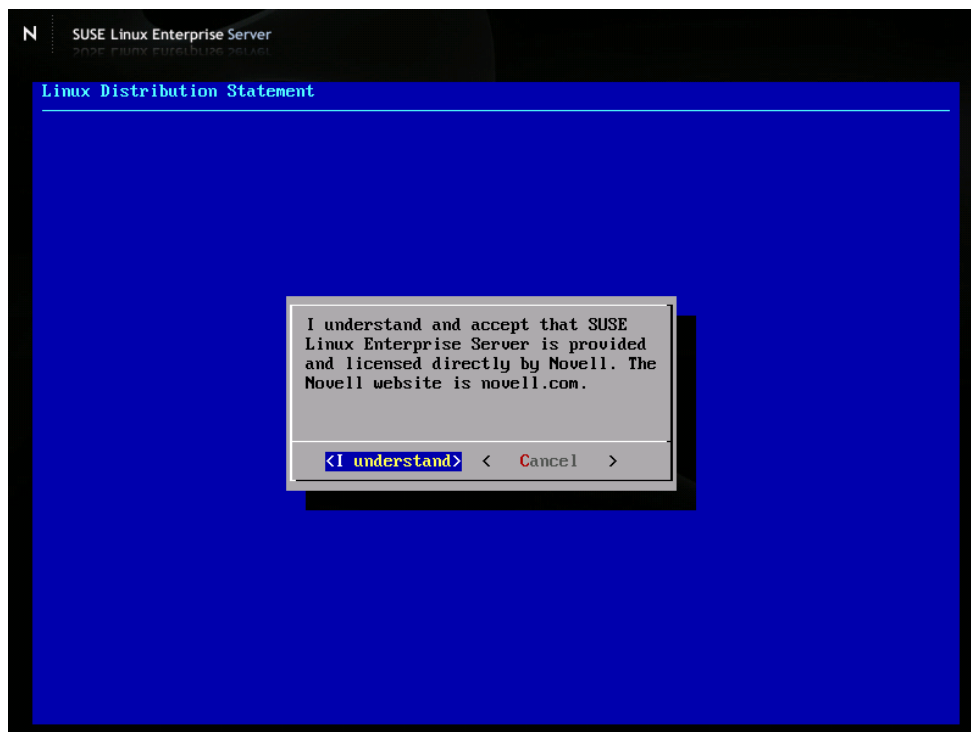


图 7 – 接受许可证

为了使用该影像，你必须接受所有列出的协议以及显示出来的声明。选择“**我接受**”进入下一个屏幕。如果你不同意该许可证，请选择“终止”（“**Abort**”），虚拟机将自动关闭。在选择所有窗格中的“**我接受**”或者“**我理解**”之后，会显示几秒钟黑色窗口。此时不要触动任何东西！图形界面最终会自动启动，并提示你输入用户 ID，如下图所示。

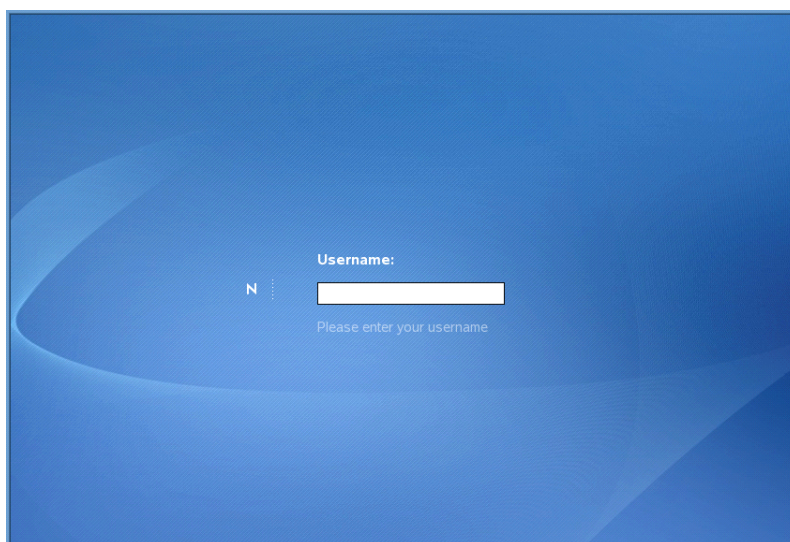


图 8 – 登录到你的 Linux 系统

在提示中，输入以下证书：

用户 ID: **db2inst1**

密码 **password**

这是从现在起你将在所有练习中使用的用户 ID 及密码。下一次你再使用这个虚拟机时，将**不会**再提示你接受许可证。

如果登录成功，你会看到类似下图的桌面。

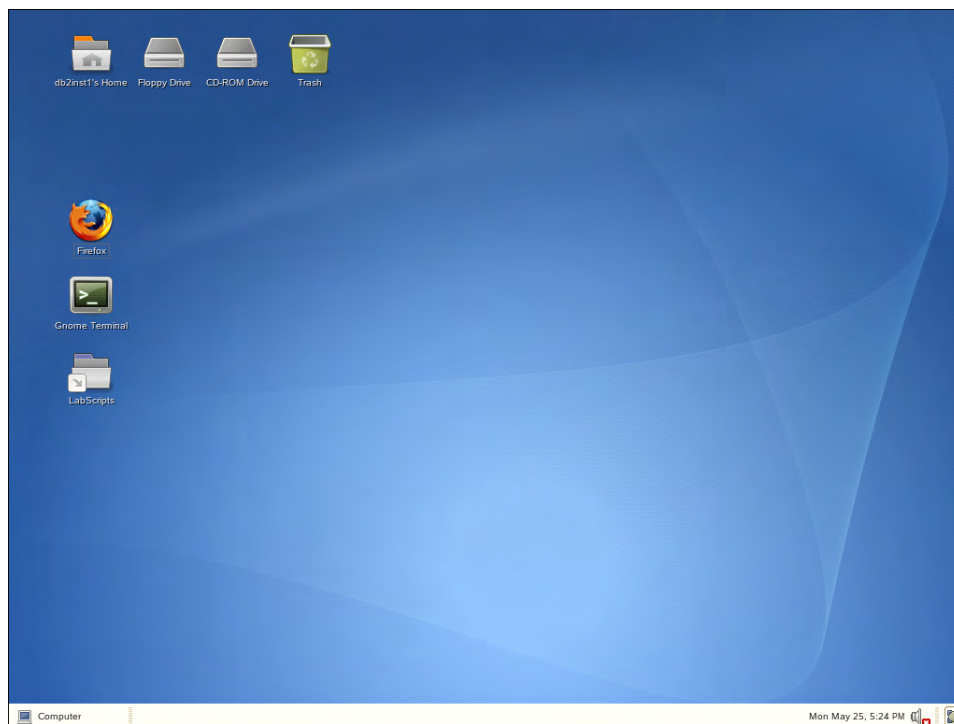


图 9 – 你的桌面

4.4 打开终端窗口

为了执行命令，我们将使用**命令行终端**。为了启动终端窗口，请点击屏幕左下角的



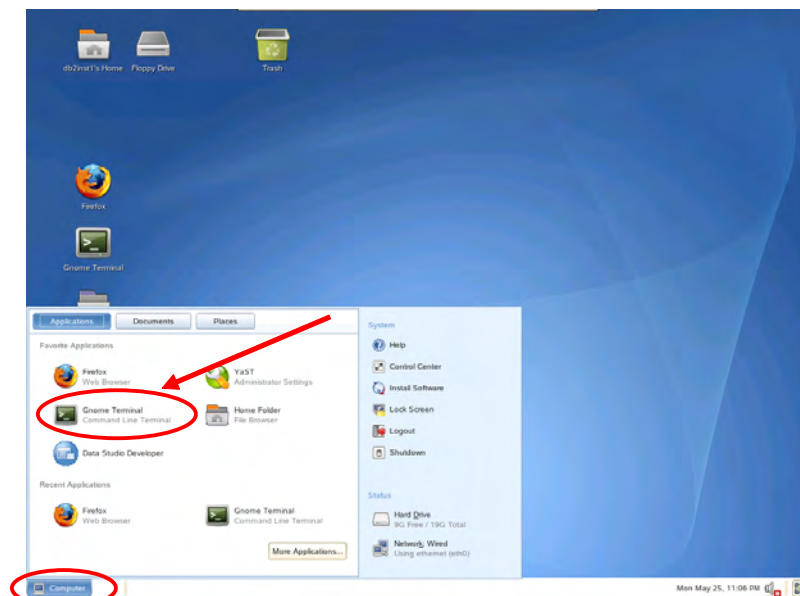


图 10 – 启动终端

点击该图标之后，将会弹出一个类似于上面的终端窗口。

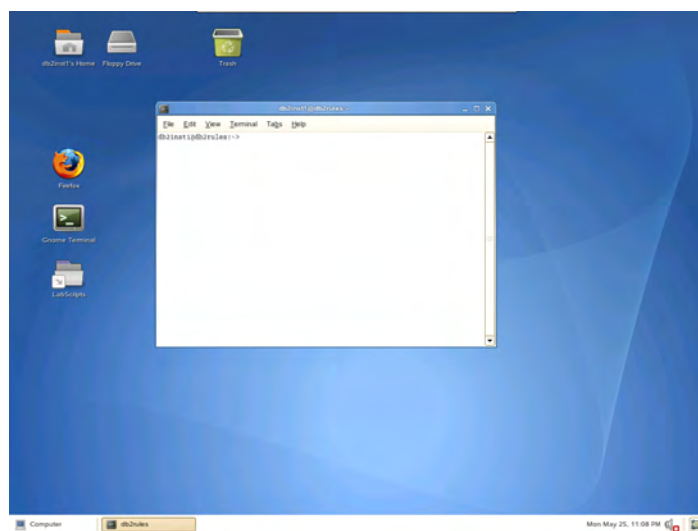


图 11 – 启动终端（续）

该终端为你提供了一个命令行提示符，你可以通过该提示符执行任何命令。

4.5 关闭终端窗口

为了关闭终端窗口，只需点击终端窗口右上角的“X”按钮，或者在命令行提示符下输入“**exit**”，即可退出已经登录的终端。（**注意**，如果你采用了远程登录，或者你从本终端窗口中登录至不同的用户，有可能会多次“**exit**”命令才能退出所有已经登录的会话并关闭终端窗口）。

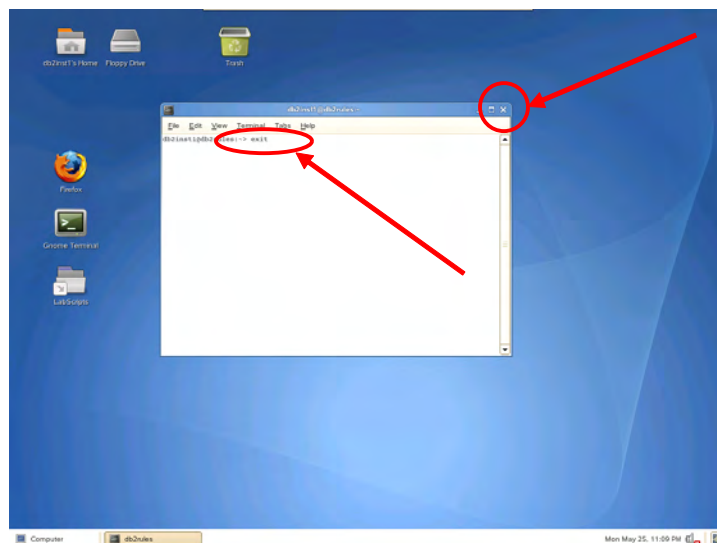


图 12 – 关闭你的终端

简短介绍到此结束。



© Copyright IBM Corporation 2011
All Rights Reserved.

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

IBM, IBM (logo), and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

VMware is a trademark of VMware Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this publication is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of July 2010, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.



IBM DB2® 9.7

关系数据库概念 亲自动手实验室

信息管理云计算能力中心

IBM（加拿大）研究院

目录

1. 前言	3
2. 目标	3
3. 推荐读物	3
4. 使用 ER 图	3
5. 关系的类型	4
6. 把实体映射为表	5
7. 关系模型的概念	5
8. 答案	6
使用 ER 图.....	6
关系的类型.....	7
把实体映射为表.....	8
关系模型的概念.....	8

1. 前言

在本实验中，你将学习关系数据库概念。本实验由四部分构成：

1. 使用 ER（实体-关系）图
2. 关系的类型
3. 把实体映射为表
4. 关系模型的概念

2. 目标

本实验结束时，你将能够：

- ▶ 使用 ER 图
- ▶ 理解不同类型的关系
- ▶ 把实体映射为表
- ▶ 区别关系的属性、度（或者目，degree）以及基数 (cardinality)。

3. 推荐读物

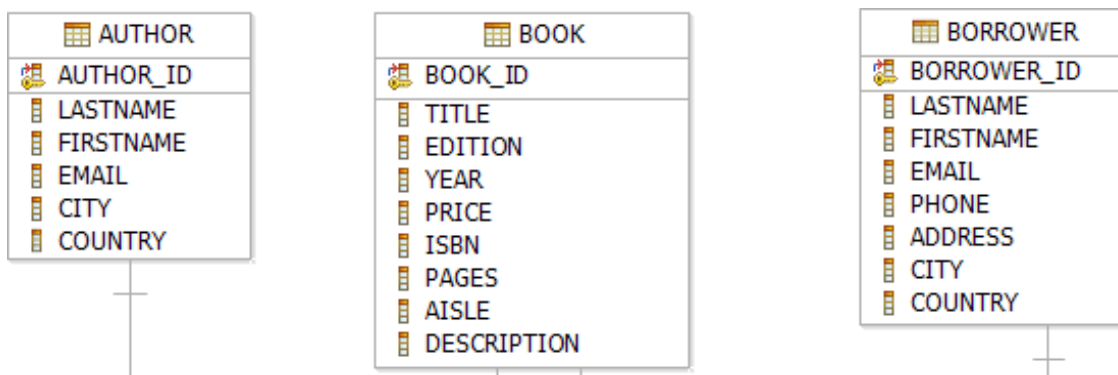
Database Fundamentals eBook（数据库原理电子书）（第 1 – 4 章）

<https://www.ibm.com/developerworks/wikis/display/db2oncampus/FREE+ebook+-+Database+fundamentals>

这是一本免费的电子书，能够让你熟悉数据库的概念以及 SQL 语言。

4. 使用 ER 图

假设有下图：

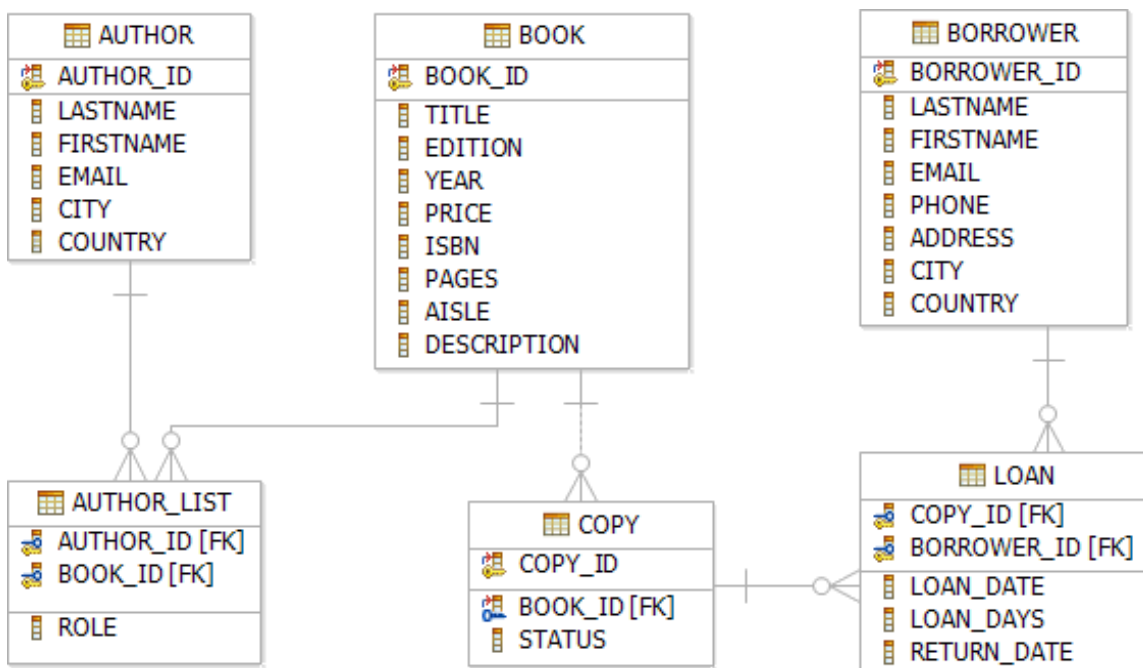


4.1 为实体 AUTHOR 绘制 ERD (实体-关系图)

4.2 为实体 BORROWER 绘制 ERD

5. 关系的类型

假设有下图:

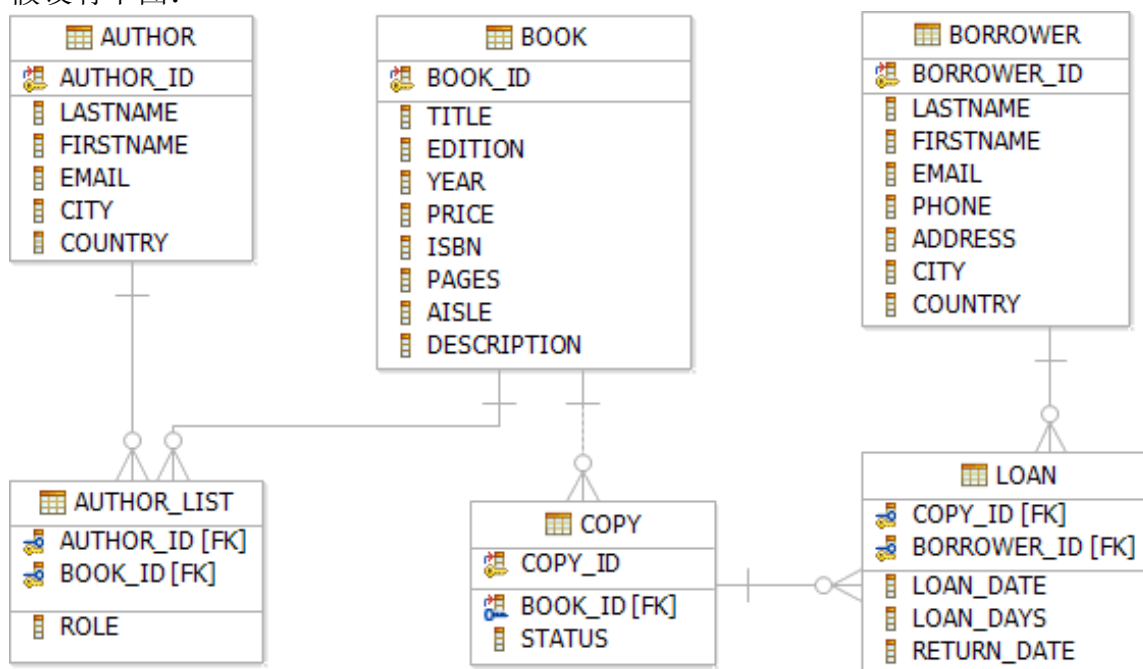


为以下两个场景绘制关系图:

- 5.1 一种书籍可能只有一册，也可能有很多册（提示：应当使用的关系 - 书籍的数量）
- 5.2 许多书可能有许多作者（提示：应当使用的关系 - 作者）

6. 把实体映射为表

假设有下图：



- 6.1 把实体 BORROWER 映射为表 BORROWER
- 6.2 把实体 AUTHOR_LIST 映射为表 AUTHOR_LIST

7. 关系模型的概念

考察下面的 BOOK 关系并回答问题：

关系：**BOOK**

BOOK_ID	TITLE (名称)	EDITION (版本)	YEAR (年)	PRICE (价)	ISBN	PAGES (页)	AISLE (通)	DESCRIPTION (描述)
---------	------------	--------------	----------	-----------	------	-----------	-----------	------------------

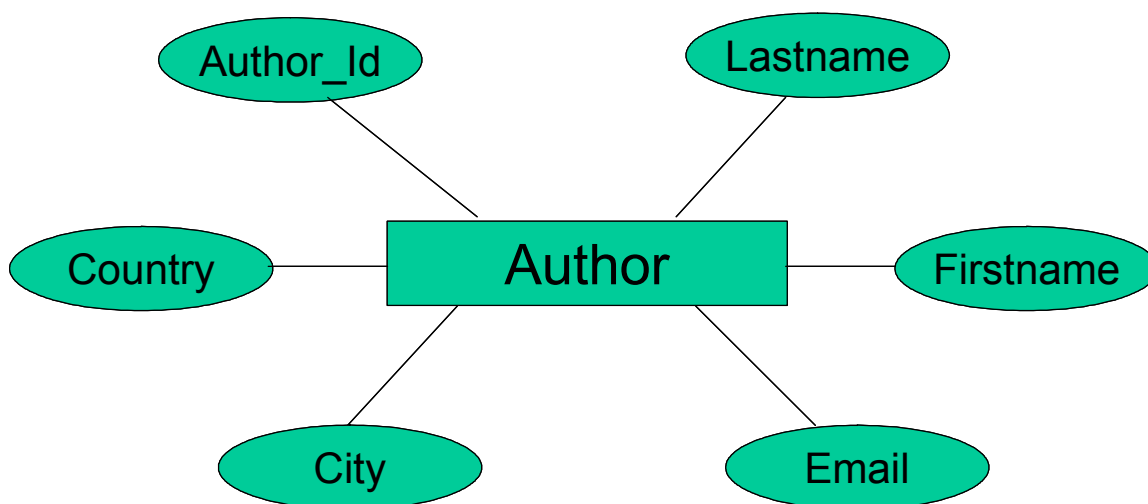
			代)	格)		数)	道)	
B1	DB2 Express-C 入门	3	2009	24.99	978-0-9866283-5-1	280	DB-A01	利用 DB2 Express-C (DB2 的免费版本) 向你讲解 DB2 的原理
B2	数据库原理	1	2010	24.99	978-0-9866283-1-1	300	DB-A02	向你讲解数据库的原理
B3	DB2 应用开发入门	1	2010	24.99	978-0-9866283-1-2	298	DB-A01	向你讲解 DB2 应用开发的概要
B4	WAS CE 入门	1	2010	24.99	978-0-9866283-1-3	278	DB-A01	向你讲解 WebSphere 应用服务器社区版的概要

- 7.1 确定关系 BOOK 的属性。
- 7.2 这个关系的度是多少?
- 7.3 这个关系的基数是多少?
- 7.4 这个关系有多少元组 (tuple)?
- 7.5 确定该表 BOOK 的列数。
- 7.6 这个表有多少字段?
- 7.7 这个表有多少记录?
- 7.8 不同的行会存在不同的列数吗? 为什么?
- 7.9 “页数”列应当有什么适当的数据类型?
- 7.10 能够在一列中存储超过一种数据类型的数据吗?
- 7.11 哪个(些)列适合用作主键?
- 7.12 把 YEAR 和 PRICE 用作复合主键是个好主意吗? 为什么?

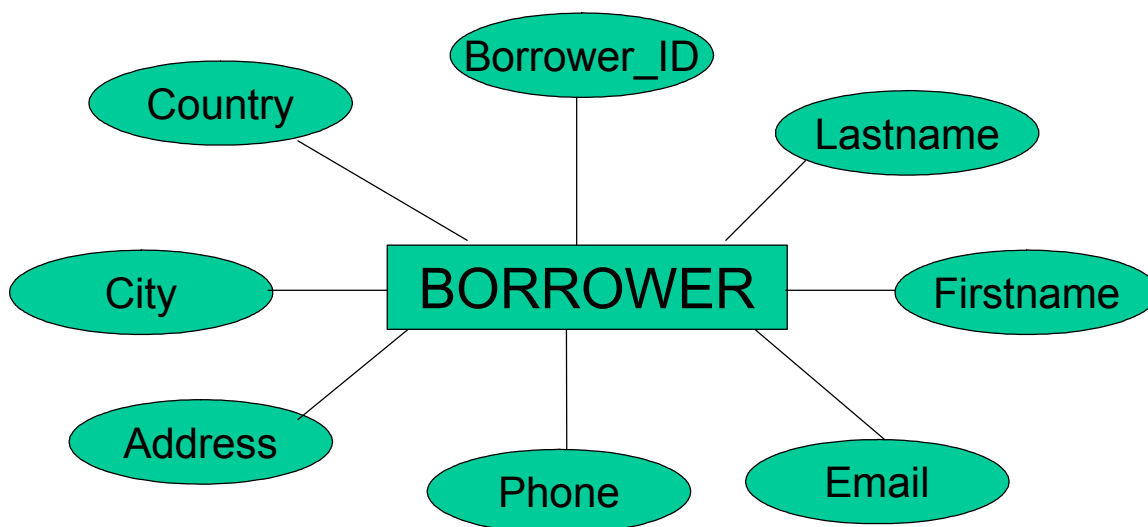
8. 答案

使用 ER 图

4.1 实体的 AUTHOR 的 ERD



4.2 实体 BORROWER 的 ERD



关系的类型

5.1 一种书籍可能只有一册，也可能有很多册



5.2 许多书可能有许多作者



把实体映射为表

6.1. 表: BORROWER

BORROWER_ID	LASTNAME	FIRSTNAME	EMAIL	PHONE	ADDRESS	CITY	COUNTRY

6.2. 表: AUTHOR_LIST

AUTHOR_ID	BOOK_ID	ROLE

关系模型的概念

7.1 关系 BOOK 的属性是:

- a. BOOK_ID
- b. TITLE
- c. EDITION
- d. YEAR
- e. PRICE
- f. ISBN
- g. PAGES
- h. AISLE
- i. DESCRIPTION

7.2 关系 BOOK 的度数为 9

7.3 关系 **BOOK** 的基数为 4

7.4 四个元组（即基数）

7.5 与 7.1 中的答案相同

7.6 九个字段。字段、列或属性指的是同一种东西

7.7 四条记录。元组、列或记录指的是同一种东西

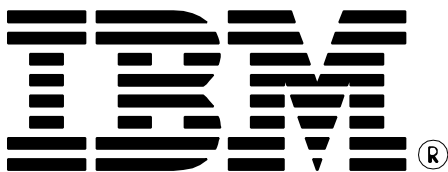
7.8 不能。观察一张表，你会看到，所有的行必须有相同的列数。

7.9 “**Pages**” 列存储了书籍的页数；因此，**INTEGER** 或许是恰当的数据类型。

7.10 不能。一列中只能存储一种类型的数据。

7.11 **BOOK_ID** 是不错的选择，因为它唯一地指出了一列。**ISBN** 也可以被选作主键，因为它也唯一地指出了一列。

7.12 把 **YEAR** 和 **PRICE** 结合起来作为复合主键（在本例中，就是由两列构成的一个键）不是个好主意，因为可能有多个记录具有相同的 **YEAR-PRICE** 组合，从而无法唯一地指出一条记录。



© Copyright IBM Corporation 2011
All Rights Reserved.

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

IBM, IBM (logo), and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States, other countries, or both.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

VMware is a trademark or VMware Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this publication is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of July 2009, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.



IBM DB2® 9.7

DB2 入门 亲自动手实验

信息管理云计算能力中心

IBM（加拿大）研究院

目录

1. 前言	3
2. 目标	3
3. 推荐读物	3
4. 开始	3
4.1 环境设置要求.....	3
4.2 初始步骤.....	3
5. 使用 DB2 数据库	5
5.1 FIRST STEPS (初始步骤)	5
5.2 使用 DB2SAMPL 命令.....	7
5.3 试试看：练习 DB2 命令.....	8
5.4 使用脚本.....	9
5.4.1 创建 SQL 脚本.....	9
5.4.2 创建操作系统脚本.....	10
5.5 接入 DB2 数据库.....	11
5.5.1 使用 CONNECT 语句.....	11
5.5.2 对本地 DB2 数据库进行编目.....	11
5.5.3 对远程 DB2 数据库进行编目.....	14
6. 答案	15
7. 小结	17

1. 前言

本模块旨在向你介绍实例、数据库，并练习 DB2 命令，包括与 DB2 服务器的连接。

2. 目标

本实验结束时，你将能够：

- ▶ 创建 DB2 数据库
- ▶ 使用 DB2 命令
- ▶ 接入远程数据库

3. 推荐读物

Getting started with DB2 Express-C eBook (DB2 Express-C 入门电子书) (第 1、3-7、9 章)

<https://www.ibm.com/developerworks/wikis/display/DB2/FREE+Book-+Getting+Started+with+DB2+Express-C>

这是一本免费的电子书，能够让你快速了解 DB2。


4. 开始

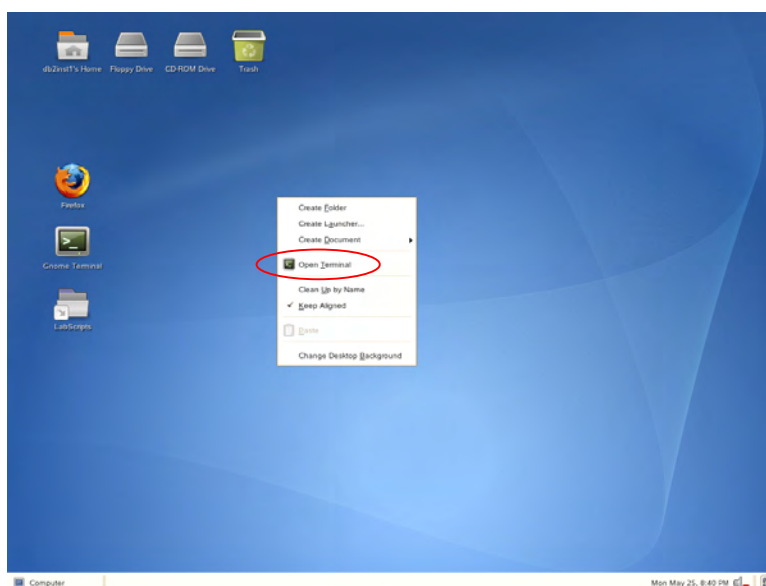
4.1 环境设置要求

要完成本实验，你需要以下软件：

- DB2 Academic Associate Bootcamp VMware 影像 (DB2 学术联合训练营 VMware 影像)
- VMware Player 2.x 或 VMware Workstation 5.x 或者更高版本

4.2 初始步骤

1. 在 VMware 中点击  Power On 按钮，启动 VMware 影像。
2. 在登录提示符下，以下列证书登录：
 - ▶ 用户名: **db2inst1**
 - ▶ 密码: **password**
3. 在桌面上右击鼠标，选择打开终端项，即可打开终端窗口。



4. 在提示符下输入以下命令，启动 DB2 Database Manager:

```
db2inst1@db2rules:~> db2start
```

注：此命令只有在以用户 db2inst1 用户的身份登录时才会起作用。如果你偶尔以其他用户的身份登录了，请在命令行提示符下键入 `su - db2inst1`，密码为：`password`。

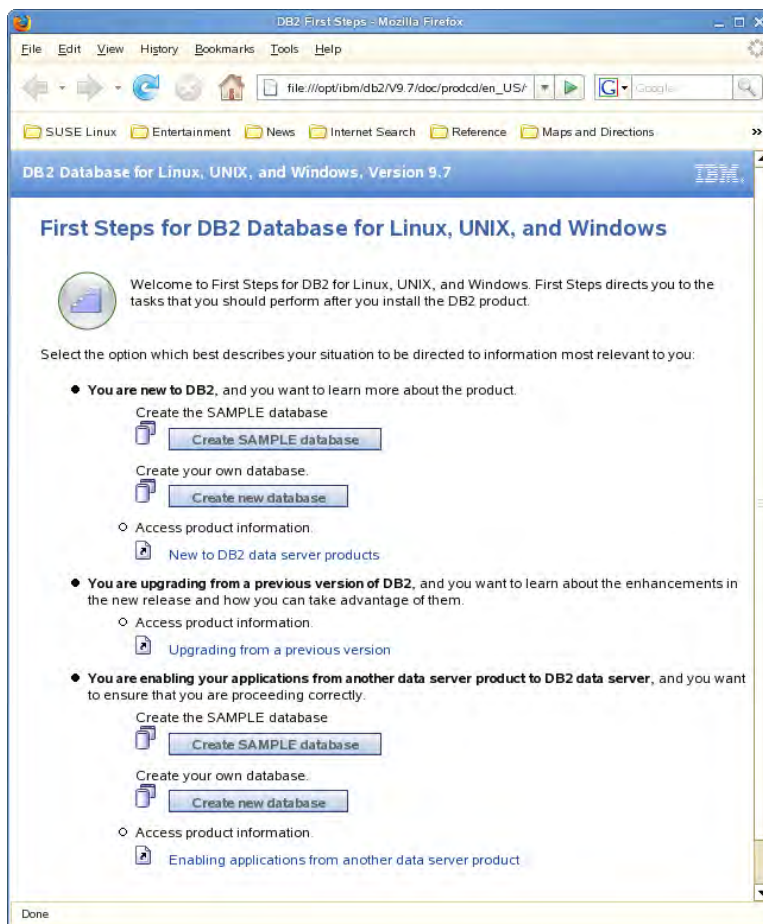
5. 在整个实验中，将使用 SAMPLE 数据库来探索 DB2 的功能。要创建 SAMPLE 数据库，我们需要首先删除现有的 SAMPLE 数据库。输入以下命令：

```
db2inst1@db2rules:~> db2 drop db sample
```

5. 使用 DB2 数据库

5.1 First Steps（初始步骤）

First Steps 是一个帮助你开始使用 DB2 的图形界面工具。作为 DB2 安装过程的一部分，First Steps 面板将显示出来，让用户生成多个能够使用的数据库示例：



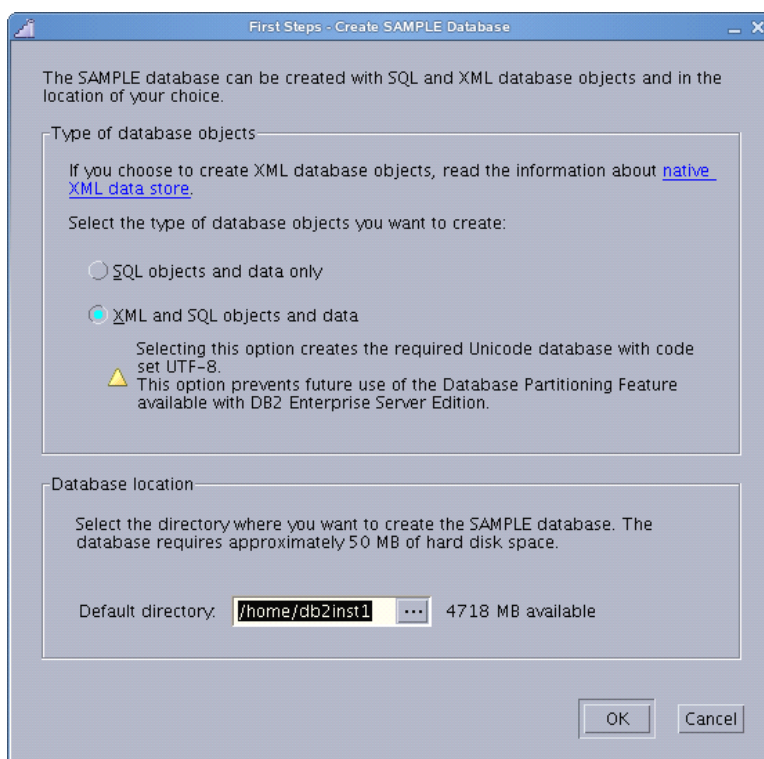
多数用户希望创建 **SAMPLE** 数据库并使用它来探索 DB2 的功能。可以通过在命令行提示符下输入命令 `db2fs` 来调用此面板。

```
db2inst1@db2rules:~> db2fs
```

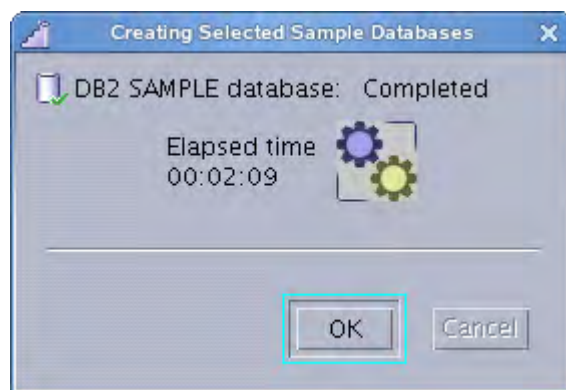
First Steps 需要一个浏览器，浏览器配置文件应当能够正常运行及工作。选择 **Yes**（是）来创建浏览器配置文件，并选择 **OK**（确定）继续。



此外，在命令行提示符下发出命令 `db2samp1` 也能够生成 **SAMPLE** 数据库。一旦选择了 **SAMPLE** 按钮，就会显示出另外一个面板来确定在什么地方创建 **SAMPLE** 数据库。



在创建 **SAMPLE** 数据库时，我们建议你选择“XML and SQL objects and data”（XML 和 SQL 对象及数据）选项。此选项将以 UTF-8 (Unicode) 格式生成数据库，这将能够让你操纵 XML 对象。如果你未选择 XML 选项，你将无法向你的 **SAMPLE** 数据库中添加 XML 对象。



下面，我们在不使用图形用户界面的情况下创建 **DB2** 数据库。

5.2 使用 **db2sampl** 命令

如果你利用上述 **First Steps** 方法创建了 **SAMPLE** 数据库，请删除此数据库示例，以便我们能够研究如何利用命令行来创建它。

```
db2inst1@db2rules:~> db2 drop db sample
```

如在前面章节所述，我们也可以在命令行提示符下输入 `db2sampl` 命令来创建 **SAMPLE** 数据库。

```
db2inst1@db2rules:~> db2sampl
```

我们来看一看在创建 **SAMPLE** 数据库时 **DB2** 创建的这些表空间。通过以下命令，连接至该示例数据库（将在后面详细讨论）并列出生数据库的表空间：

```
db2inst1@db2rules:~> db2 connect to sample
```

```
db2inst1@db2rules:~> db2 list tablespaces
```

你会观察到类似于下面内容的输出：

```
db2inst1@db2rules:~> db2 list tablespaces

          Tablespaces for Current Database

Tablespace ID          = 0
Name                   = SYSCATSPACE
Type                   = Database managed space
Contents               = All permanent data. Regular table space.
State                  = 0x0000
  Detailed explanation:
    Normal

Tablespace ID          = 1
Name                   = TEMPSPACE1
Type                   = System managed space
Contents               = System Temporary data
State                  = 0x0000
  Detailed explanation:
    Normal

Tablespace ID          = 2
Name                   = USERSPACE1
Type                   = Database managed space
Contents               = All permanent data. Large table space.
State                  = 0x0000
  Detailed explanation:
    Normal
```

5.3 试试看：练习 **DB2** 命令

在本节中，请遵照所提供的指示练习 **DB2** 命令。这部分的答案可以在本实验的末尾找到，但我们建议你在自己做题之后再查看答案！

1.

在 Windows 或 Linux 中打开命令行处理器 (CLP)。你看到的提示符应当是 "db2 =>"。如果不是这样的话，你就没有处于 CLP 中。

2. 进行这些操作：

- a) 以默认值创建数据库 "DB101"。在 DB2 创建本课程中所讨论的数据库对象时，可能需要花费几分钟的时间才能完成。
- b) 连接到数据库中。
- c) 关闭本数据库的“自优化内存” ("Self tuning memory")。提示：查看 db cfg
- d) 显示 db cfg，确认“自优化内存”已经关闭。
- e) 终止该实例。你可能会收到一条错误提示，指出与数据库之间有连接。如果是这样的话，请使用以下命令列出存在哪些连接：

```
db2=> list applications
```

运行下面的命令来强制关闭所有的应用（连接）：

```
db2=> force applications all
```

现在再次尝试停止该实例。

- f) 启动该实例
- g) 列出 DB2 配置注册表 (Profile Registry) 的内容。你需要从 Linux 外壳或者 DB2 命令窗口中完成该工作。

5.4 使用脚本

5.4.1 创建 SQL 脚本

1) 利用以下条件创建 SQL 脚本：

- a) 脚本名称： myscript1.db2
- b) SQL 脚本运行：

```
select * from department

create table tbl1 (name varchar(30), phone varchar(20))

insert into tbl1 values

    ('Tom', '123456789'), ('Mary', '987654321')

select * from tbl1

drop table tbl1
```

c) 使用 "#" 作为语句结束符

2)

从 DB2 命令窗口或者 Linux 外壳中运行该脚本。你首先需要进入 SAMPLE 数据库。

5.4.2 创建操作系统脚本

1) 利用下面的条件创建操作系统脚本:

a) 脚本名称: myscript2

b) 该脚本应当调用前面章节中创建的 myscript1.db2。

c) 该脚本在调用时应当使用三个参数, 如下所示:

```
myscript2 <dbname> <userID> <password>
```

其中:

- <dbname> 是数据库名称

- <userID> 和 <password> 是用于连接数据库的参数

2) 利用 SAMPLE 数据库运行 myscript2, 如下所示:

a) 授予执行期限:

```
chmod +x myscript2
```

b) 利用 SAMPLE 数据库执行:

```
db2inst1@db2rules:~> ./myscript2 sample db2inst1 password
```

如果在 Windows 中运行, 则无需使用 chmod 命令; 相反, 应当添加 “.bat” 后缀, 并以如下方式执行:

```
C:\> myscript2.bat sample <userID> <password>
```

5.5 接入 DB2 数据库

在用户或者应用程序使用数据库之前, 必须首先与该数据库建立连接。你可以使用 CONNECT 语句与数据库建立连接。

5.5.1 使用 CONNECT 语句

在执行 SQL 语句之前, 你必须首先接入数据库。

为了接入我们的示例数据库, 请输入以下命令:

```
db2inst1@db2rules:~> db2 CONNECT TO sample USER db2inst1 USING password
```

你也可以接入数据库并让 DB2 提示你输入密码, 命令如下:

```
db2inst1@db2rules:~> db2 CONNECT TO sample USER db2inst1
```

或者, 如果你打算接入的数据库是本地数据库, 而且你只打算利用默认的用户 ID 接入该数据库, 则执行如下命令:

```
db2inst1@db2rules:~> db2 CONNECT TO sample
```

任何时候当你需要终止与数据库的连接时, 你可以使用 TERMINATE 命令:

```
db2inst1@db2rules:~> db2 terminate
```

5.5.2 对本地 DB2 数据库进行编目

为什么要对数据库编目呢? 因为, 如果没有该信息的话, 应用就无法接入数据库!

当你创建一个数据库的时候，该数据库就会自动在本地及系统数据库目录中进行编目。这样，一项本地连接就能够发挥作用，就像我们在前面章节中所介绍的那样。

对于远程连接，也就是当 DB2 客户端和 DB2 服务器不处在同一个系统上的时候，你就需要在 DB2 客户端上运行编目命令。这将在下面介绍。

DB2 有多个目录可以被用来访问数据库。这些目录能够让 DB2 找到它已知的数据库，无论这些数据库是在本地系统还是在远程系统。系统数据库目录中包含一份列表和指针指示，在这里可以找到每个已知的数据库。

为了把某个条项输入任何此类目录，可以使用 CATALOG 命令。为了删除某个条目，可以使用 UNCATALOG 命令。

要查看系统数据库目录中的条目，可以执行下面的命令：

```
db2inst1@db2rules:~> db2 list database directory
```

其输出应当是类似于下面的内容：

```
System Database Directory
Number of entries in the directory = 1
Database 1 entry:
Database alias           = SAMPLE
Database name           = SAMPLE
Local database directory = /home/db2inst1
Database release level  = d.00
Comment                 =
Directory entry type    = Indirect
Catalog database partition number = 0
Alternate server hostname =
Alternate server port number =
```

这里可以看到，示例数据库已经在系统中编目了。这些信息可以被用于接入此数据库。

但是，如果这些信息没有在这里列出（即：数据库在创建时没有被编目），我们就无法接入该数据库。

我们来看看如果数据库未被编目将会发生什么情况。对 SAMPLE 数据库发出 UNCATALOG 命令：

```
db2inst1@db2rules:~> db2 uncatalog database sample
```

然后再尝试接入 SAMPLE 数据库。

```
db2inst1@db2rules:~> db2 connect to sample
```

可以看到，这是不可能的。你可能会接到一条 SQL1013N 消息：

```
SQL1013N  The database alias name or database name "SAMPLE"
could not be found.  SQLSTATE=42705
```

事实上，该示例数据库以及与之相关的文件仍然存在于我们的系统中，但在数据库目录中不存在该信息，无法供 DB2 客户端来建立连接。可以利用前面检查系统数据库目录的方法来验证这一点：

```
db2inst1@db2rules:~> db2 list database directory
```

```
SQL1057W  The system database directory is empty.  SQLSTATE=01606
```

可以通过在命令行处理器中输入以下命令对数据库编目：

```
db2 catalog database database_name as database_alias on path/drive
```

其中：

- *database_name* 是你打算编目的数据库的名称。
- *database_alias* 是你打算编目的数据库的本地昵称。
- *path/drive* 是待编目的数据库驻留的路径。

为了对名为 *sample* 的数据库编目并使其具有本地数据库别名 *mysample*，可以输入以下命令：

```
db2inst1@db2rules:~> db2 catalog database sample as mysample
```

输入以下命令，检查这一新条项在数据库目录中的信息：

```
db2inst1@db2rules:~> db2 list database directory
```

```
System Database Directory
```

```
Number of entries in the directory = 1
```

```
Database 1 entry:
```

```
Database alias                = MYSAMPLE
Database name                 = SAMPLE
Local database directory     = /home/db2inst1
Database release level         = d.00
Comment                         =
```

Directory entry type	= Indirect
Catalog database partition number	= 0
Alternate server hostname	=
Alternate server port number	=

现在，由于该数据库已经被编目了，其信息已经存在与数据库目录中，我们可以使用在上面的编目语句中指定的别名来连接它，并可以执行我们的 SQL 语句了。

```
db2inst1@db2rules:~> db2 connect to mysample
```

发出 **TERMINATE** 命令，以终止与 **SAMPLE** 数据库之间的连接。

```
db2inst1@db2rules:~> db2 terminate
```

5.5.3 对远程 DB2 数据库进行编目

在本节中，请遵照所提供的指示练习 DB2 命令。这部分的答案可以在本实验的末尾找到，但我们建议你在自己做题之后再查看答案！

对远程 DB2 数据库进行编目意味着你手边已经具备了如下信息：

- 远程 DB2 服务器的 IP 地址或主机名
- 数据库所驻留的实例的端口号码
- 待接入的数据库的名称
- 远程服务器上定义的用户 ID / 密码。

我们来模拟一下你如何接入远程数据库，在这里，通过使用 **LOCALHOST** 把你自己的服务器当作远程服务器。使用如下关于你的远程服务器的信息：

IP 地址或主机名	<u>LOCALHOST</u>
实例端口号码	<u>50001</u>
数据库名称	<u>SAMPLE</u>
用户 ID	<u>db2inst1</u>
密码	<u>password</u>

使用 **ALIAS** “**SAMPLE2**” 指向该数据库。

6. 答案

第 5.3 节的答案

- a) 使用默认值创建数据库 "DB101"。

```
db2=> create database db101
```

- b) 接入数据库

```
db2=> connect to db101
```

- c) 关闭数据库的 "Self tuning memory" (自优化内存)。

```
db2=> update db cfg using SELF_TUNING_MEM off
```

- d) 显示 db cfg, 以确认 "Self tuning memory" 现在确实已经关闭。

```
db2=> get db cfg
```

- e) 终止该实例。

```
db2=> list applications
db2=> force applications all
db2=> db2stop
```

- f) 启动该实例

```
db2=> db2start
```

- g) 列出 DB2 配置注册表 (Profile Registry) 的内容。你需要从 Linux 外壳或者 DB2 命令窗口中完成该工作。

```
db2=> quit
db2set -all
```

第 5.5.3 节的答案

- a) 对 TCPIP 节点编目

```
db2=> catalog tcpip node mynode remote localhost server
50001
```

- b) 对该数据库编目，以确保指向前一步中的 TCPIP 节点

```
db2=> catalog db sample as sample2 at node mynode
```

- c) 测试是否能够连接。注意，你**必须**按照语法输入用户 ID/密码，因为在该设置中，我们是利用 TCPIP 访问数据库的，这意味着 DB2 认为它是一个远程数据库。

```
db2=> connect to sample2 user db2inst1 using password
```

- d) 如果无法连接，请检查服务器上的 db2comm（在本例中，它恰好是与你的客户端相同的系统）是否设置为 TCPIP:

```
db2inst1@db2rules:~> db2set -all
```

- e) 如果未如此设置的话，请按照下述方式设置它:

```
db2inst1@db2rules:~> db2set db2comm=tcpip
db2inst1@db2rules:~> db2stop
db2inst1@db2rules:~> db2start
```

请注意，对 DB2 配置注册表变量进行修改时，只有执行 db2stop/db2start 命令之后，修改才会生效。

- f) 如果仍然无法建立连接的话，请在你的 DB2 服务器上检查该端口是否设置为 50001:

```
db2inst1@db2rules:~> db2 get dbm cfg | grep SVCENAME
```

- g) 如果 SVCENAME 是一个字符串而不是一个数值（例如 50001），请在 /etc/services 中查找该字符串。例如，如果 SVCENAME 具有值 db2c_DB2，则按照如下方式查找它:

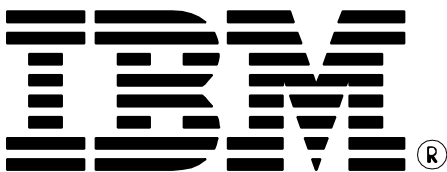
```
db2inst1@db2rules:~> cat /etc/services | grep db2c_DB2
```

应当有这样的条目： db2c_DB2 50001/tcp

如果没有的话，添加之。

7. 小结

本练习向你介绍了构成 DB2 数据库的对象以及影响数据库的创建的各种因素。你也学习了如何检查和修改 DB2 配置参数以及如何建立连接。



© Copyright IBM Corporation 2011
All Rights Reserved.

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

IBM, IBM (logo), and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both

UNIX is a registered trademark of The Open Group in the United States, other countries, or both

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this publication is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of April 2010, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.



IBM DB2® 9.7

Data Studio 入门
亲自动手实验

信息管理云计算能力中心

IBM（加拿大）研究院

目录

1. 前言	2
2. 目标	2
3. 推荐读物	3
4. 开始: IBM DATA STUDIO 基础	3
4.1 ECLIPSE 基本原理.....	3
4.1.1 数据组织 – 工作区和工程.....	3
4.1.2 用户界面 – 视图和视角 (PERSPECTIVE).....	5
5. 环境设置要求	6
6. 启动 DATA STUDIO	8
6.1 数据库连接.....	10
6.1.1 接入 SAMPLE 数据库.....	10
6.1.2 修改数据库参数.....	12
6.1.3 停止和启动你的 DB2 实例.....	14
7. 小结	16

1. 前言

Data Studio 是一套基于 Eclipse 的免费工具，可以用来进行数据库管理和开发。

2. 目标

本实验结束时，你将能够：

- ▶ 了解 Eclipse 环境的基础知识
- ▶ 建立数据库连接
- ▶ 修改数据库参数
- ▶ 启动和终止 DB2 实例

3. 推荐读物

Getting started with IBM Data Studio for DB2（面向 DB2 的 IBM Data Studio 入门） （第 1-3 章）

<https://www.ibm.com/developerworks/wikis/display/db2oncampus/FREE+ebook+-+Getting+started+with+IBM+Data+Studio+for+DB2>

这是一本免费的电子书，能够让你快速了解 IBM Data Studio

IBM Data Studio Information Center（IBM Data Studio 信息中心）

<http://publib.boulder.ibm.com/infocenter/idm/v2r2/index.jsp>

这是一部资料库，含有利用 IBM Data Studio 进行开发和管理的教程。

4. 开始：IBM Data Studio 基础

本节实验向你介绍 IBM Data Studio 的基础以及如何快速使其正常运行。

完成本节内容之后，你将能够：

- ▶ 启动 Data Studio
- ▶ 创建新的数据库连接
- ▶ 断开及重新连接数据库

4.1 Eclipse 基本原理

IBM Data Studio 以 Eclipse 平台为基础构建而成，因此被称为基于 Eclipse 的开发环境。Eclipse 平台是一套框架，能够支持创建集成的开发环境 (IDE)；有许多插件可以支持以 Java、C/C++、PHP、COBOL、Ruby 等语言进行开发。使用 Eclipse 的开发人员将会鉴赏 IBM Data Studio 所提供的熟悉的感觉和外观。

4.1.1 数据组织 – 工作区和工程

在基于 Eclipse 的环境中，所有的开发都在某个工程中进行，所谓“工程”就是一个含有全部源代码、图形及其他辅助材料的目录。这个概念许多人在使用其他 IDE 时都已经熟悉了。在 Data Studio 中，你通常使用数据开发工程，但也有其他工程类型用于 Java 开发、web 开发，等等。

你所创建的每项工程必须包含在某个工作区中，所谓“工作区”也是你的文件系统中的—一个目录。工作区目录中含有多个子目录，用于存放在工作区中创建的各个不同工程。例

如，图 4 1 展示了一个这样的场景：在路径 /workspace 中创建了一个工作区，在该工作区中又创建了三个工程 – BankApp、BookStore 和 WebSite。请注意，这些工程都是作为 /workspace 的子目录来创建的。

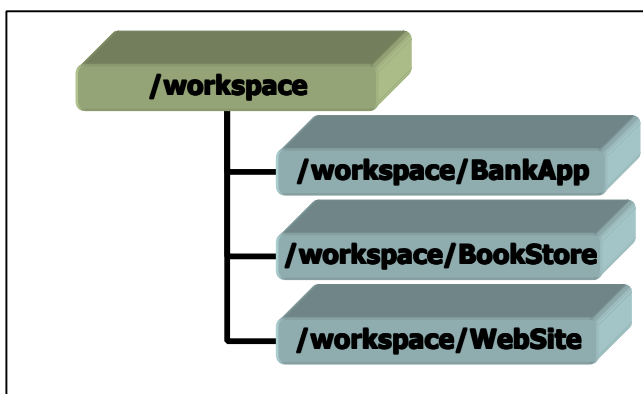


图 4 1-工作区-工程分级体系

在基于 Eclipse 的环境中，可以通过 *Project Explorer* 视图很方便地浏览工作区和工程（我们将在后面的章节中讨论视图）。

在打开了基于 Eclipse 的环境之后，用户可以在图 4 2 所示的对话中选择使用哪个工作区。通过输入新的、不存在的路径，也可以创建新的工作区，或者，可以通过指定现有的路径来使用现有的工作区。

此外，用户也可以决定仅使用某个特定的工作区（从而再不会被打扰！），方法是选中 **“Use this as the default and do not ask again”**（把它作为默认工作区，而且不要再询问）选择框。当然，可以随时撤销该选择，方法是修改程序首选项中的设置。

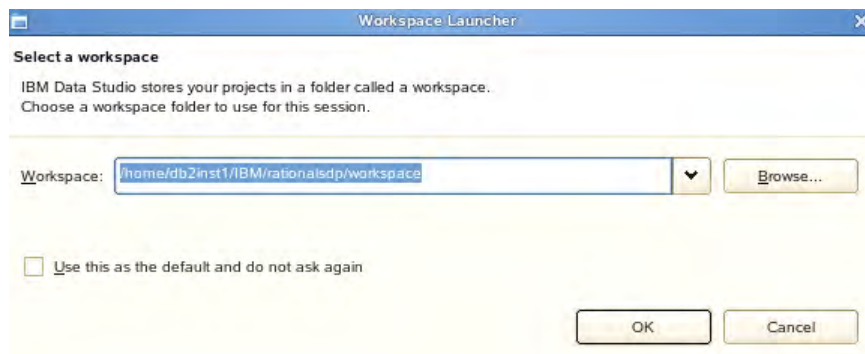


图 4 2-工作区选择

图 4 3 示出了图 4 1 的工作区等级结构在 Project Explorer 的显示情况。

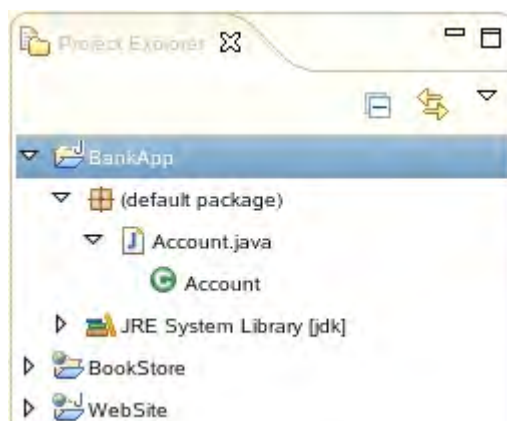


图 4 3 – *Project Explorer* 视图

4.1.2 用户界面 – 视图和视角 (Perspective)


基于 **Eclipse** 的环境通过 *视图* 和 *视角* 提供了易于使用的、可以定制的图形界面。就像工作区包含工程一样，**Eclipse** 视角包含了视图。实际上，我们已经看见过一个视图的例子。在图 4 3 中，我们看到，*Project Explorer* 视图显示出工作区中的所有工程以及这些工程中包含的文件。视图只不过是一个任务面板 – 即一个停驻的窗口，可以用来查看对象，或许还可以操纵对象。

基于 **Eclipse** 的环境把视角定义为适合于特定任务或一系列工作的多个视图的集合。当打开一个视角时，所有与之相关的视图都在该环境打开，而先前打开的任何其他视图则都被隐藏起来。

在 **IBM Data Studio** 中，你一般会使用两种视角：*数据库管理* 视角和 *数据* 视角。

为了在视角之间切换，请点击工具栏上所期望的名称：



如果你在寻找的视角没有显示出来，只需点击  工具栏图标即可出现可用视角的列表。

基于 **Eclipse** 的环境能够通过指定加载哪些视图来创建定制的视角。

下图示出了“数据库管理”视角。

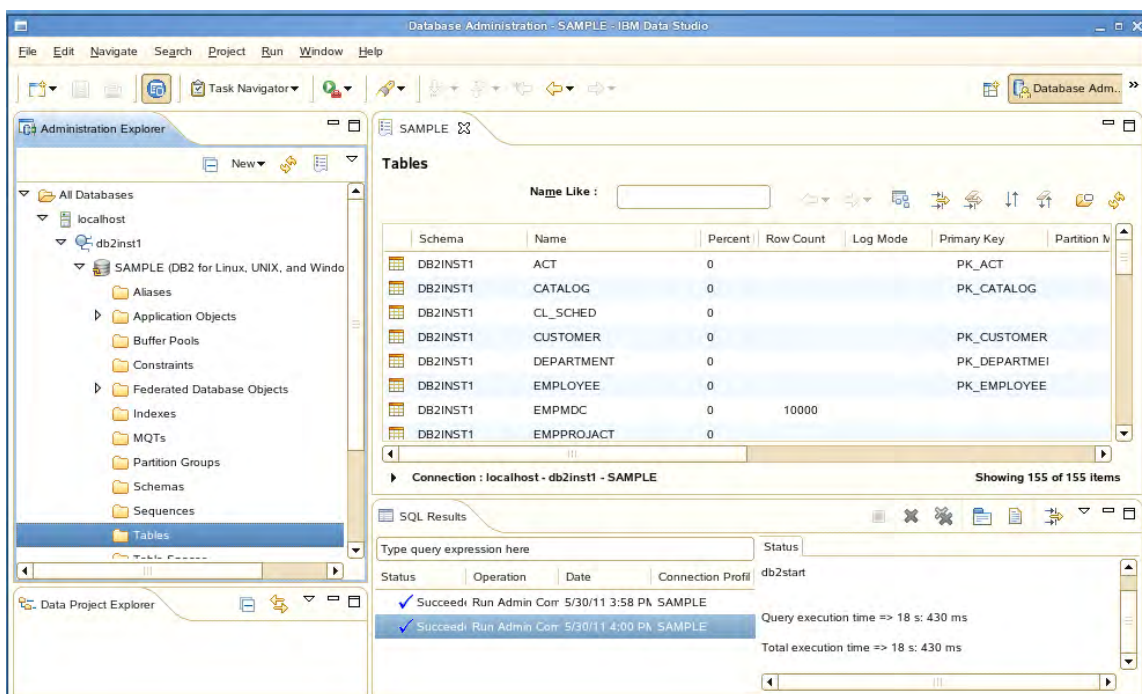


图 4-4-数据库管理视角

5. 环境设置要求

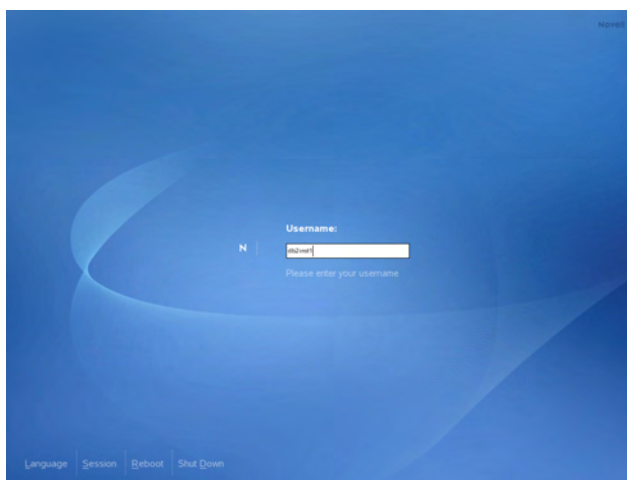
要完成本实验，你需要以下软件：

- DB2 Academic Training VMware® 影像（DB2 学术培训 VMware® 影像）
- VMware Player 2.x 或 VMware Workstation 6.x 或更高版本

关于如何获得这些软件以及如何使用 VMware 影像的帮助信息，请遵循“**VMware 基础和入门**”实验中给出的指示。

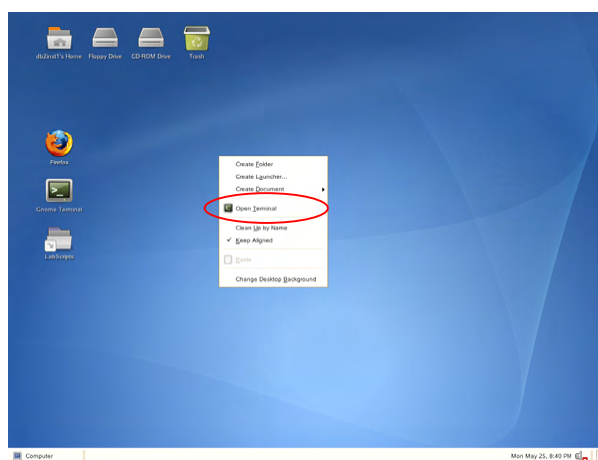
1. 在登录提示符下，以 db2inst1 证书登录：

- ▶ 用户名：**db2inst1**
- ▶ 密码：**password**



注：非常重要的一点是，此时不要以根用户的身份登录。

2. 在桌面上右击鼠标，选择打开终端项，即可打开终端窗口。



3. 在提示符下输入以下命令，启动 DB2 Database Manager:

```
db2inst1@db2rules:~> db2start
```

注：此命令只有在以用户 db2inst1 用户的身份登录时才会起作用。如果你偶尔以其他用户的身份登录了，请在命令行提示符下键入 `su - db2inst1`，密码为：`password`。

4. 本实验假设你已经创建了 **SAMPLE** 数据库。你可以使用下面的命令检查现有数据库的列表:

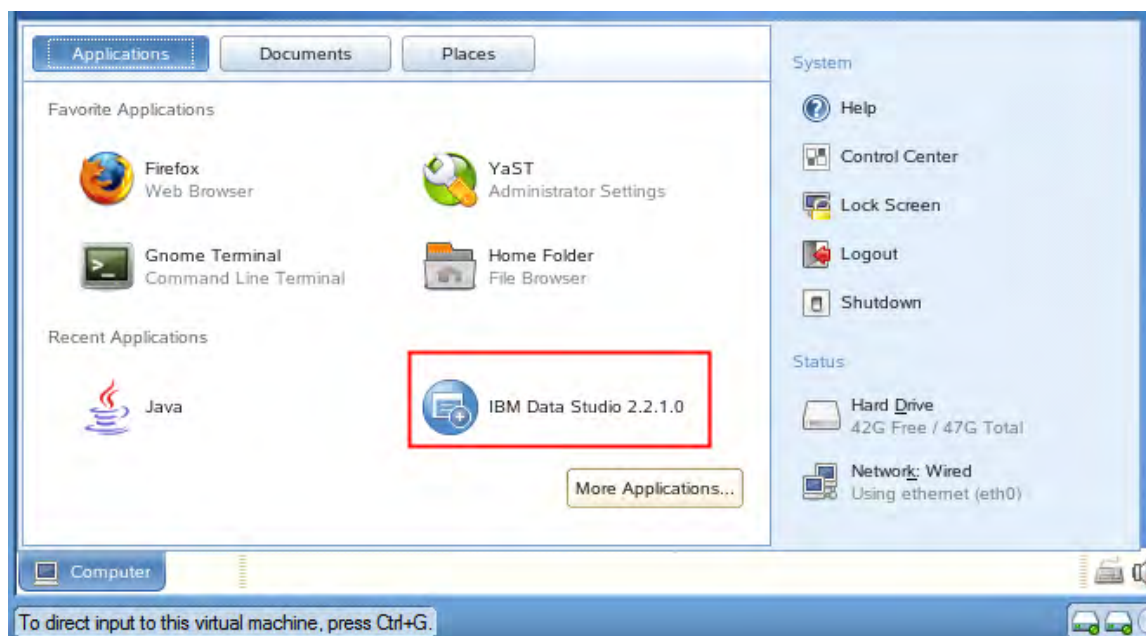
```
db2inst1@db2rules:~> db2 list db directory
```

5. 如果 **SAMPLE** 数据库未出现在列表中，你可以利用下面的命令创建它:

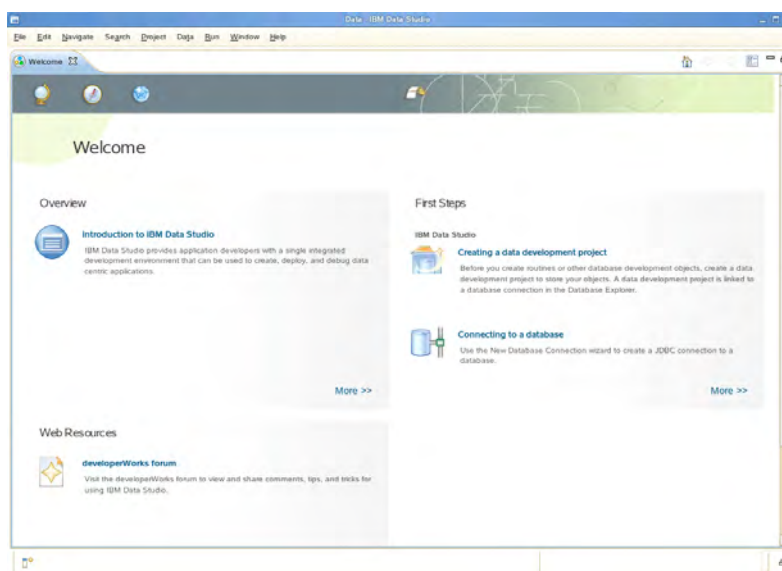
```
db2inst1@db2rules:~> db2sampl
```


6. 启动 Data Studio

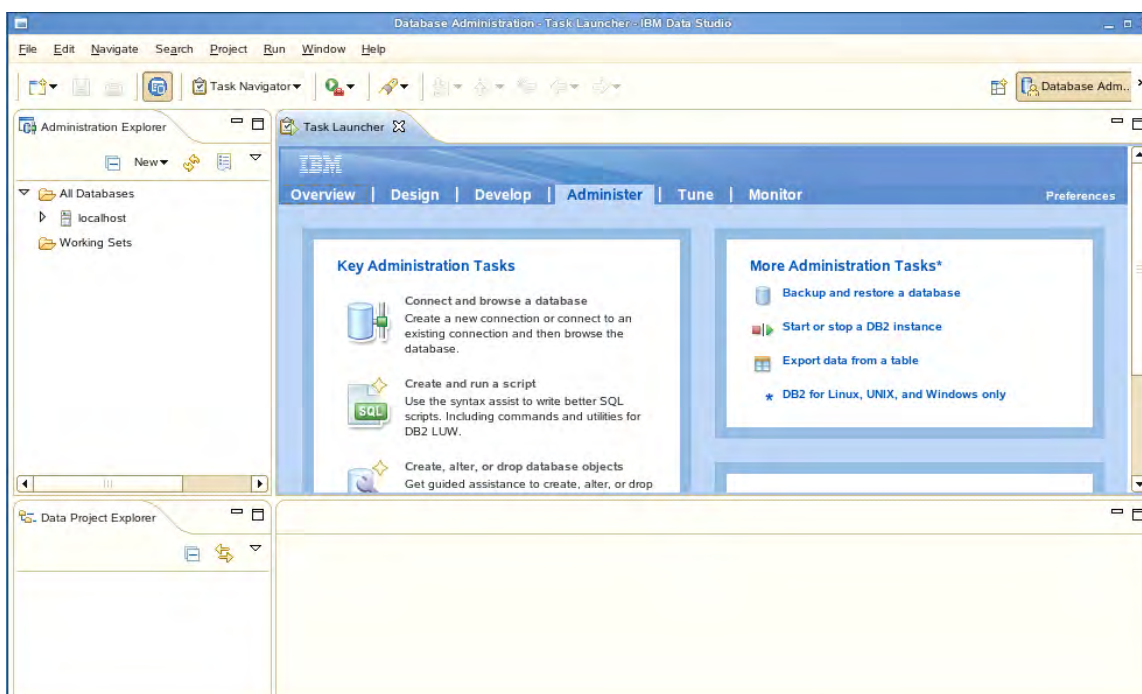
1. 在屏幕左下角点击**计算机**按钮，然后选择 **Data Studio 2.2.1.0**



2. 在“选择工作区”对话框中，接受默认路径。点击“确定”。
3. Data Studio 现在启动，显示欢迎页面。



4. 点击左上侧的关闭按钮 (), 关闭本窗口, 你将进入下面所示的**数据库管理**视角。如果你看到的屏幕与下面的不同, 请点击 *Window* (窗口) -> *Open perspective* (打开视角) -> *Other* (其他) -> *Database Administration* (数据库管理), 以确保打开了**数据库管理**视角。

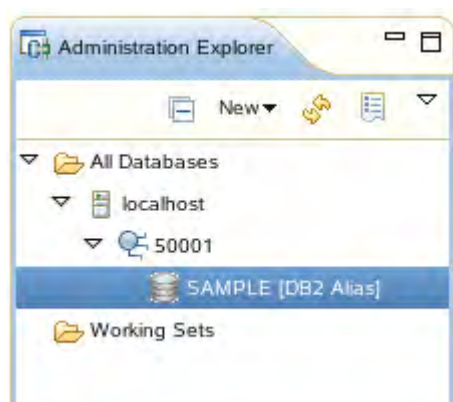


6.1 数据库连接

在你能够利用 **Data Studio** 开展任何生产性工作之前，必须首先建立与数据库之间的连接。**Data Studio** 中的 **Administration Explorer**（管理浏览器）视图能够让你完成该工作。从该视图中，可以与数据库工件进行交互并对其进行操纵。由于我们将使用 **SAMPLE** 数据库，现在连接该数据库。

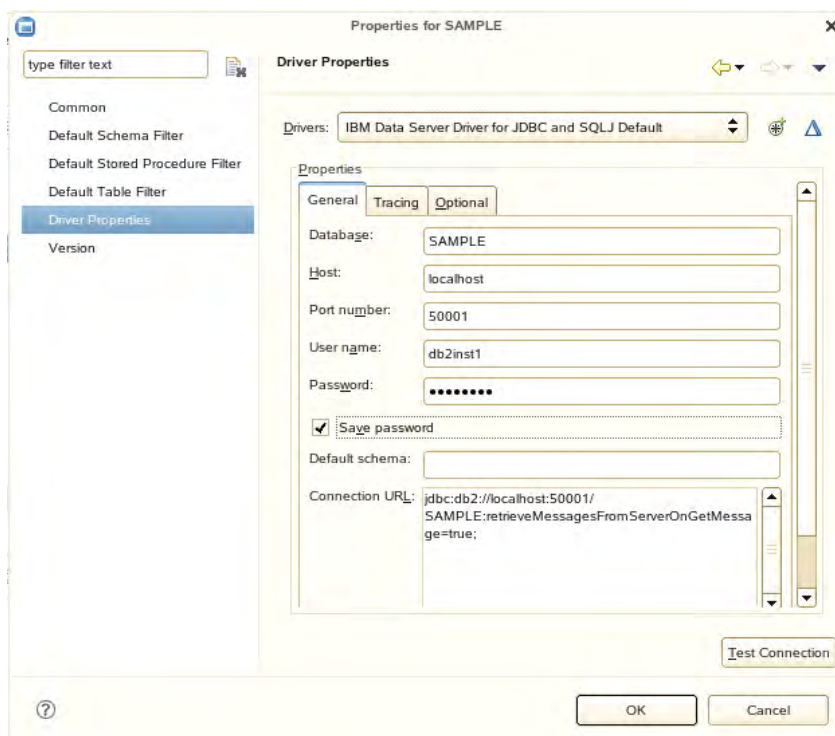
6.1.1 接入 **SAMPLE** 数据库

1. 由于 **SAMPLE** 数据库是在调用 **Data Studio** 之前创建的，因此，**Data Studio** 应当能够探测到该数据库。在 **Administration Explorer** 中，可以一种沿着树结构向下探究，如下所述：

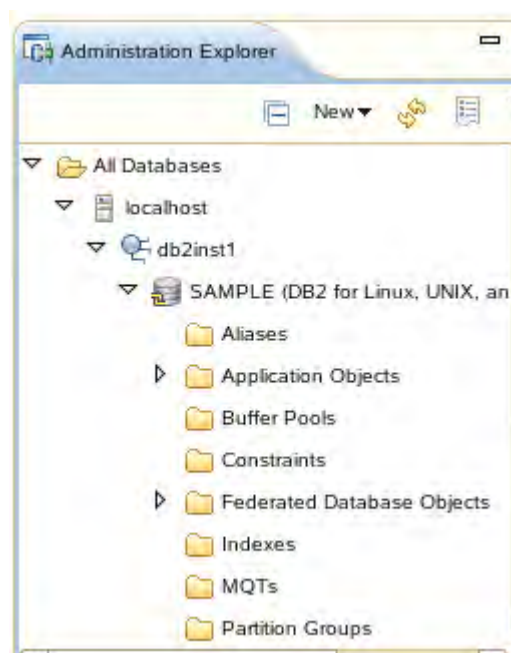


2. 右击 **SAMPLE [DB2 Alias]** 并选择 **Properties**（属性）。然后再点击 **Driver properties**（驱动器属性）选项卡。
3. 在 **Properties** 面板中，提供如下信息：
 - ▶ 数据库：SAMPLE
 - ▶ 主机：localhost
 - ▶ 端口号码：50001
 - ▶ 用户名：db2inst1
 - ▶ 密码：password

一定要点击“**Save password**”（保存密码）检查框，这样你就不必反复输入密码了。如下图所示。



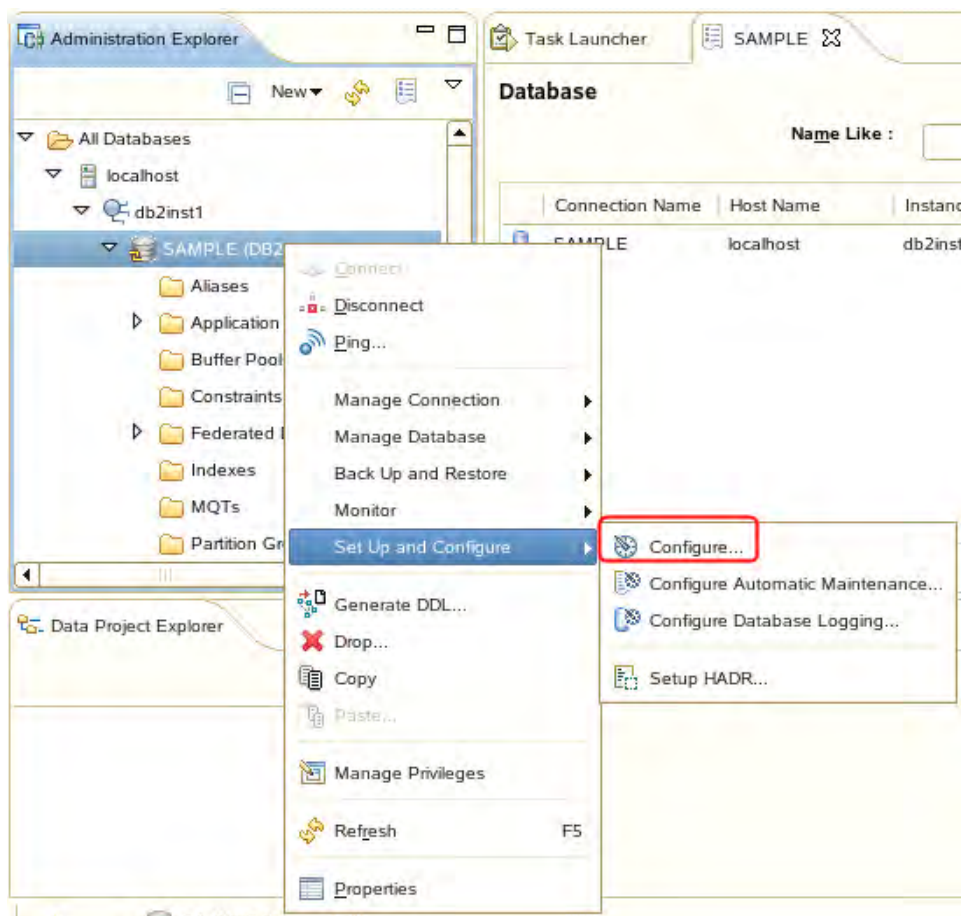
4. 点击右下角的 **Test Connection**（测试连接）按钮。你应当收到“Ping 成功完成”的消息。如果没有收到的话，请检查你是否输入了正确的信息，然后再试一次。在测试成功后点击 **OK**（确认）。
5. 现在已经验证了你的连接属性，可以从 Administration Explorer 中再次右击 **SAMPLE [DB2 Alias]** 并选择 **Connect**（连接）。
6. 几秒钟之后，与数据库 **SAMPLE** 之间的连接应当建立起来。请注意，**SAMPLE** 旁边的连接图标现在显示为互锁的双手。这表明，你已经接入数据库。



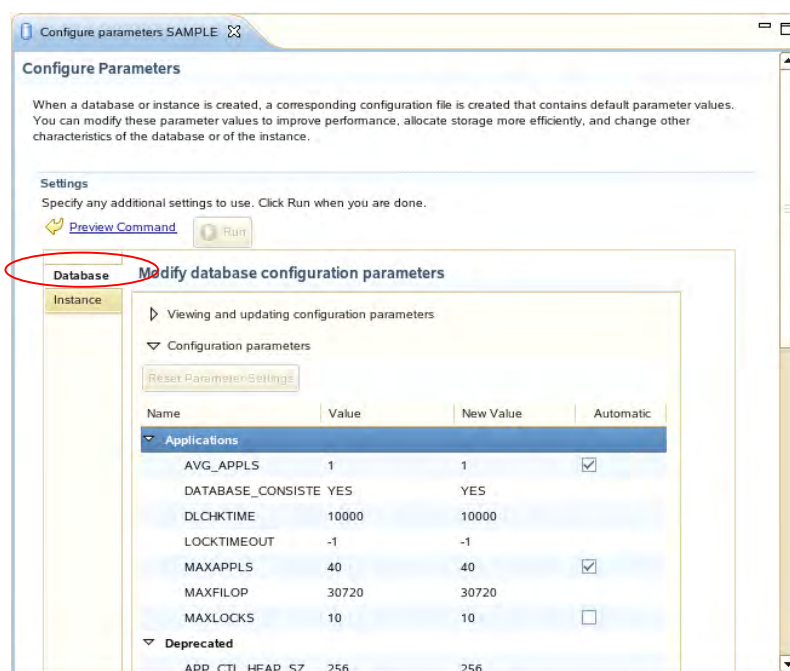
6.1.2 修改数据库参数

Data Studio 可以执行 DB2 中的多种管理功能。其中之一是能够操纵数据库参数。

1. 在 Administration Explorer 中，右击 **SAMPLE** 数据库，然后选择 **Set up and Configure**（设置及配置）-> **Configure**（配置）。



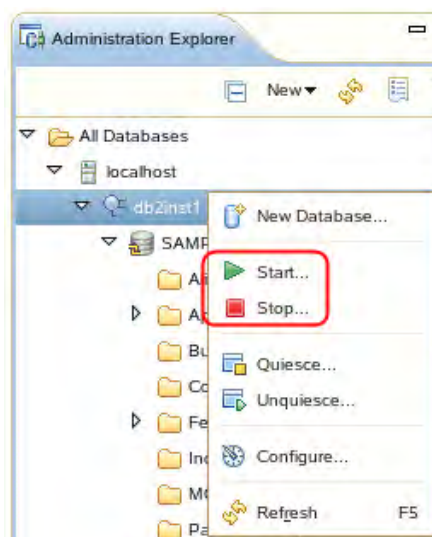
2. 这将打开一个新的视图，以供配置 **SAMPLE** 数据库的参数。在这个视图中，可以修改与数据库配置相关的多项参数，也可以修改与数据库所属的实例相关的参数。我们目前不修改任何参数，因此在浏览之后关闭该视图即可。



6.1.3 停止和启动你的 DB2 实例

在上一节，你已经看到，可以修改实例级的参数。实际上，有些修改需要重启实例才会生效。这就是在 Data Studio 中允许你停止和启动实例的原因。

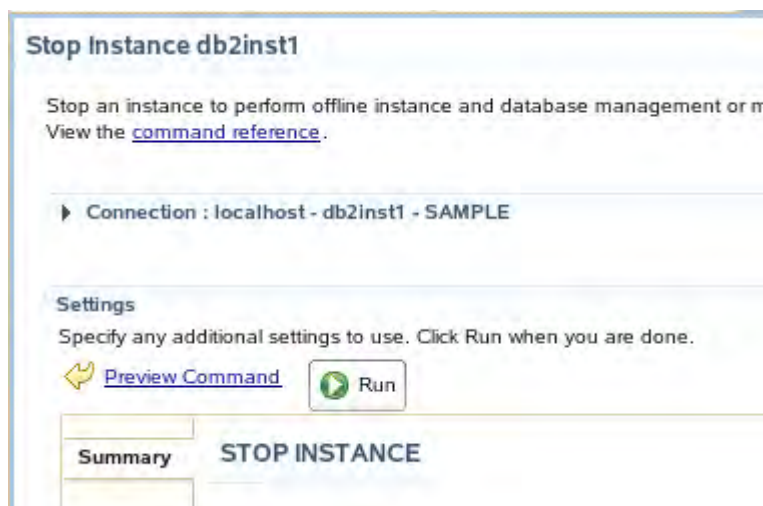
1. 要终止实例，可以在 Administration Explorer 中右击实例名称（本例中是 db2inst1）并选择停止。



2. 然后弹出一个窗口要求你选择连接配置文件。Administration Explorer 实际上显示出多个连接配置文件，其中包括数据库、用户 ID、密码以及你在前面输入的其他信息。你可以创建不同的连接配置文件。例如，对于相同的 SAMPLE 数据库，你可以创建称为 'SAMPLE1' 的连接配置文件，它使用不同的用户 ID 和密码接入数据库。当你停止该实例时，Data Studio 会询问你使用哪个连接配置文件，这实际上是说将利用哪个用户 ID 来执行此操作。在我们的例子中，我们只有一个连接配置文件，即 'SAMPLE'。因此，选择该连接配置文件，然后单击 OK（确定）。



3. 接着，你需要单击 Run（执行）按钮来停止该实例。



4. 在“SQL 结果”视图中，你会看到你的命令的状态，它从处理时的“正在运行”转变为完成后的“成功”。在“状态”面板中，你也能看到所执行的命令以及控制台的输出。
5. 现在启动该实例，方法与上面的步骤一样，但这次选择 **Start**（开始）。

6. 退出 Data Studio。

7. 小结

现在你已经看到，IBM Data Studio 为 DB2 的开发和管理提供了非常具有生产力的环境。在下面的实验过程中，我们还将看到能够多么快捷及简便地创建和执行 SQL 和 XQuery 脚本；开发和测试用 SQL 和 Java 编写的存储过程；创建和修改数据库对象；分析查询执行；等等。



© Copyright IBM Corporation 2011
All Rights Reserved.

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

IBM, IBM (logo), and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States, other countries, or both.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

VMware is a trademark or VMware Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this publication is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of July 2009, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.



IBM DB2® 9.7

SQL 和数据库对象简介 亲自动手实验

信息管理云计算能力中心

IBM（加拿大）研究院

目录

目录.....	2
1. 前言.....	3
2. 目标.....	3
3. 推荐读物.....	3
4. 开始.....	4
4.1 环境设置要求.....	4
4.2 初始步骤.....	4
5. 使用 DB2 数据对象.....	5
5.1 表.....	5
5.1.1 模式.....	9
5.2 视图.....	11
5.3 别名.....	13
5.4 索引.....	15
5.5 序列.....	17
6. 使用 SQL.....	19
6.1 查询数据.....	19
6.1.1 利用 DATA STUDIO 从表中检索所有的行.....	20
6.1.2 利用 SELECT 语句检索各行.....	21
6.1.3 对结果排序.....	24
6.1.4 对信息进行聚集.....	25
6.1.5 从多个表在检索数据（连接）.....	26
6.2 插入、更新和删除.....	29
6.2.1 插入.....	29
6.2.2 更新.....	31
6.2.3 删除.....	32
7. 小结.....	32
8. 答案.....	33

1. 前言

本模块旨在向你概要介绍在创建了数据库之后可以开发的各种对象。而且还向你介绍 SQL，并让你在 Data Studio 中练习 SQL 语句。

2. 目标

本实验结束时，你将能够：

- ▶ 检查和操纵数据库中的对象
- ▶ 练习 SQL 语句

3. 推荐读物

Getting started with DB2 Express-C eBook (DB2 Express-C 入门电子书) (第 8 章)

<https://www.ibm.com/developerworks/wikis/display/DB2/FREE+Book+-+Getting+Started+with+DB2+Express-C>

这是一本免费的电子书，能够让你快速了解 DB2。

Getting started with IBM Data Studio for DB2 (面向 DB2 的 IBM Data Studio 入门) (第 1 – 3 章)

<https://www.ibm.com/developerworks/wikis/display/db2oncampus/FREE+ebook+-+Getting+started+with+IBM+Data+Studio+for+DB2>

这是一本免费的电子书，能够让你快速了解 IBM Data Studio

Database Fundamentals (数据库原理) (第 5 章)

<https://www.ibm.com/developerworks/wikis/display/db2oncampus/FREE+ebook+-+Database+fundamentals>

这是一本免费的电子书，能够让你熟悉关系模型以及 SQL 语言。


4. 开始

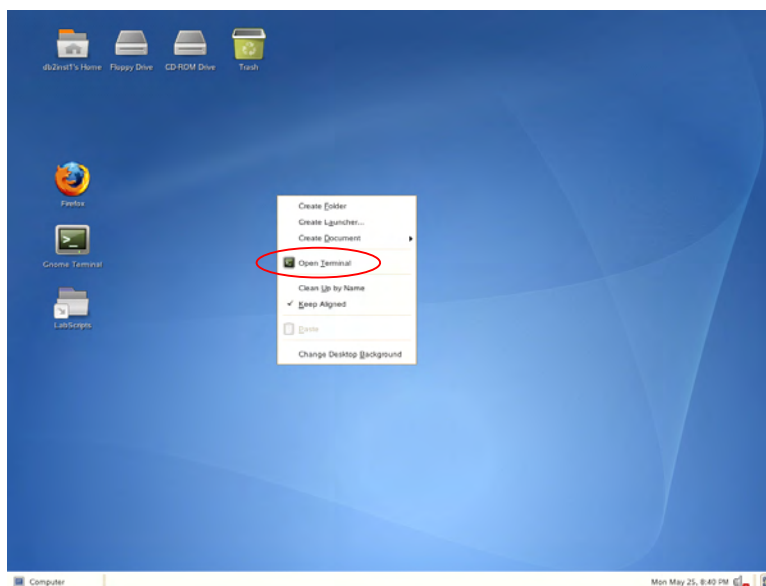
4.1 环境设置要求

要完成本实验，你需要以下软件：

- DB2 Academic Associate Bootcamp VMware 影像（DB2 学术联合训练营 VMware 影像）
- VMware Player 2.x 或 VMware Workstation 5.x 或者更高版本

4.2 初始步骤

1. 在 VMware 中点击  按钮，启动 VMware 影像。
2. 在登录提示符下，以下列证书登录：
 - ▶ 用户名：**db2inst1**
 - ▶ 密码：**password**
3. 在桌面上右击鼠标，选择**打开终端**项，即可打开终端窗口。



4. 在提示符下输入以下命令，启动 DB2 Database Manager:

```
db2inst1@db2rules:~> db2start
```

注：此命令只有在以用户 `db2inst1` 用户的身份登录时才会起作用。如果你偶然以其他用户的身份登录了，请在命令行提示符下键入 `su - db2inst1`，密码为：`password`。

5. 使用 DB2 数据对象

在我们着手认识和创建某些基本的、基础性的数据库对象之前，我们首先创建一个新的数据库，并利用它来强调本节中的某些概念。

```
db2inst1@db2rules:~> db2 create db testdb
```

创建 **TESTDB** 数据库之后，如下所示发出 **CONNECT** 语句，以便建立与新创建的数据库之间的连接。

```
db2inst1@db2rules:~> db2 connect to testdb
```

5.1 表

关系数据库以一系列数据表的方式展示数据。表由逻辑上按列及按行排列的数据构成（每一行一般称为一条记录）。

通过执行 **CREATE TABLE SQL** 语句来创建表。在最简单的格式中，该语句的语法为：

```
CREATE TABLE [TableName]  
    ([ColumnName] [DataType], ...)
```

其中：

- `TableName` 是指定给待创建的表的名称。
- `ColumnName` 是指定给待创建的列的唯一名称。
- `DataType` 是指定给待创建的列的数据类型；所规定的数据类型决定了在该列中可以存储的数据值的种类。

因此，如果你打算创建一张名为 **EMPLOYEES** 的表，其中含有三列，一列用来存储数值，另两列用来存储字符串值（如下所示），

列	类型
empid	INTEGER
name	CHAR(50)
Dept	CHAR(9)

那么，你应当执行类似下面的 `CREATE TABLE SQL` 语句：

```
db2inst1@db2rules:~> db2 "CREATE TABLE employees
      (empid INTEGER,
       name CHAR(50),
       dept INTEGER)"
```

你可以执行 `DESCRIBE` 命令来查看该表的基本属性：

```
db2inst1@db2rules:~> db2 describe table employees
```

Column name	Data type		Column Length	Scale	Nulls
	schema	Data type name			
EMPID	SYSIBM	INTEGER	4	0	Yes
NAME	SYSIBM	CHARACTER	50	0	Yes
DEPT	SYSIBM	INTEGER	4	0	Yes

3 record(s) selected.

但我们此时发现，部门的数据类型被定义为 `INTEGER`，而不是原来设想的 `CHAR`。因此，我们需要有某种方法把该数据类型从 `INTEGER` 改为 `CHARACTER`。我们可以使用 `alter`（修改）语句实现这一点。

Alter

```
db2inst1@db2rules:~> db2 "alter table employees alter column dept
set data type char(9)"
```

现在再次使用 `DESCRIBE` 命令来查看所做的修改：

```
db2inst1@db2rules:~> db2 describe table employees
```

Column name	Data type		Column Length	Scale	Nulls
	schema	Data type name			
EMPID	SYSIBM	INTEGER	4	0	Yes
NAME	SYSIBM	CHARACTER	50	0	Yes
DEPT	SYSIBM	CHARACTER	9	0	Yes

3 record(s) selected.

可以看到，`DEPT` 列现在使用 `CHARACTER` 数据类型，而不是 `INTEGER` 数据类型。

既然我们现在已经按照我们的期望创建了数据表，我们就可以开始向该表中输出待保存的数据了。我们可以联系一些非常简单的数据操纵语句，例如 `insert`（插入）、`update`（更新）及 `delete`（删除）。

插入

我们可以使用下面的语句把一些基本数据插入到我们的表中：

```
db2inst1@db2rules:~> db2 "INSERT INTO employees (EMPID, NAME, DEPT)
VALUES (1, 'Adam', 'A01 '),
(2, 'John', 'B01'),
(3, 'Peter', 'B01'),
(4, 'William', 'A01')"
```

你会收到一条 **SQL0668N** 消息：

```
SQL0668N Operation not allowed for reason code "7" on table
"DB2INST1.EMPLOYEES". SQLSTATE=57016
```

这是什么意思？如果我们发出“? SQL0668N”命令，我们就可以看到关于这条消息的问题解释和用户响应。

```
db2inst1@db2rules:~> db2 "? SQL0668N"
```

```
SQL0668N Operation not allowed for reason code "<reason-code>" on table
"<table-name>".
```

Explanation:

Access to table "<table-name>" is restricted. The cause is based on the following reason codes "<reason-code>":

...

7

The table is in the reorg pending state. This can occur after an ALTER TABLE statement containing a REORG-recommended operation.

...

User response:

...

7

Reorganize the table using the REORG TABLE command.

...

因此，我们可以得出结论，我们无法输入数据的原因在于：我们在前面对该表执行了 **ALTER** 语句，这使得该表处于 **reorg** 暂挂状态。为解决该问题，我们应当遵照所建议的方式，“使用 **REORG TABLE** 命令对该表进行重组”：

```
db2inst1@db2rules:~> db2 reorg table employees
```

再次尝试像前面那样发出 `insert` 语句:

```
db2inst1@db2rules:~> db2 "INSERT INTO employees (EMPID, NAME, DEPT)
VALUES (1, 'Adam', 'A01 '),
(2, 'John', 'B01'),
(3, 'Peter', 'B01'),
(4, 'William', 'A01')"
```

为了验证数据是否已经插入, 可以对该表执行非常基本的 `SELECT` 语句:

```
db2inst1@db2rules:~> db2 "select * from employees"
```

EMPID	NAME	DEPT
1	Adam	A01
2	John	B01
3	Peter	B01
4	William	A01

4 record(s) selected.

更新

我们也可以对该表进行更新操作。例如, 假设需要把 **Peter** 从部门 **B01** 调到部门 **A01**。我们可以使用下面的更新语句修改此表:

```
db2inst1@db2rules:~> db2 "update employees set dept='A01' where
name='Peter'"
```

再验证一下是否进行了更新。

```
db2inst1@db2rules:~> db2 "select * from employees"
```

EMPID	NAME	DEPT
1	Adam	A01
2	John	B01
3	Peter	A01
4	William	A01

4 record(s) selected.

删除

最后, 我们再对该表进行一次操作: 删除一条记录。例如, 假设 **William** 不再是我们的雇员了; 因此, 我们应当把他从我们的表中删除。我们可以使用下面的 `DELETE` 语句:

```
db2inst1@db2rules:~> db2 "delete employees where name='William'"
```


再次验证一下是否进行了删除。

```
db2inst1@db2rules:~> db2 "select * from employees"
```

EMPID	NAME	DEPT
1	Adam	A01
2	John	B01
3	Peter	A01

3 record(s) selected.

5.1.1 模式

模式 (schema) 是多个已命名对象的集合。模式为数据库的对象提供了一种逻辑分类方式。模式中 can 包含表、视图、昵称、触发器、函数、程序包 (package) 以及其他对象。

数据库中的大多数对象都采用两部分命名规约来命名。名称的第一部分 (最左侧部分) 称为模式名或者限定符, 而第二部分 (最右侧部分) 称为对象名。在句法构成上, 这两部分联接在一起并以点号 (.) 分隔:

```
schema_name.object_name
```

模式本身也是数据库中的一种对象。模式主要通过两种途径来创建:

1. 可以在创建另一个对象时隐式创建模式, 但用户必须具有 `IMPLICIT_SCHEMA` 数据库权限。
2. 当前用户可以利用 `CREATE SCHEMA` 语句显式创建模式。

我们来看看 `EMPLOYEES` 表所在的模式。在建立与数据库之间的连接之后, 列出数据库中的表, 就可以实现这一点。

```
db2inst1@db2rules:~> db2 list tables
```

在输出结果中有一列, 它指出了每个具体的表属于哪个模式:

Table/View	Schema	Type	Creation time
EMPLOYEES	DB2INST1	T	2010-03-30-16.37.05.046385

1 record(s) selected.

这是在创建另一个对象时隐式创建模式的例子。因此, 这意味着, 在创建 `EMPLOYEES` 表时, 也同时隐式创建了一个模式名, 因为我们并未规定任何模式名。默认情况下, `DB2` 将使用创建某对象的用户的 `ID` 作为模式名。这在上面的输出中可以看到。

如前所述, 我们也可以显式创建模式, 然后在创建别的对象时把这些对象赋给此模式。我们来看一个例子。假设我们打算创建一个名为 `MYSHEMA` 的新模式, 并在新创建的这个模式中创建一个新表 `STORE`。我们还希望该模式拥有当前用户 (`db2inst1`) 的权限 `ID`。

首先，我们需要利用创建 **CREATE SCHEMA** 命令创建模式：

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

在我们的例子中，就是：

```
db2inst1@db2rules:~> db2 CREATE SCHEMA myschema AUTHORIZATION db2inst1
```

要列出相应的数据库中所有可用的模式，可能在与数据库建立连接之后发出以下命令：

```
db2inst1@db2rules:~> db2 select schemaname from syscat.schemata
```

SCHEMANAME
DB2INST1
MYSHEMA
NULLID
SQLJ
SYSCAT
SYSFUN
SYSIBM
SYSIBMADM
SYSIBMINTERNAL
SYSIBMTS
SYSPROC
SYSPUBLIC
SYSSTAT
SYSTOOLS

14 record(s) selected.

下面，我们来创建一个属于 **MYSHEMA**，而不是属于 **DB2INST1** 的表。我们可以使用下面的语句来完成：

```
db2inst1@db2rules:~> db2 "CREATE TABLE myschema.store
                           (storeid INTEGER,
                            address CHAR(50))"
```

（注意，所指定的表名称在其被创建的模式中必须是唯一的。）

现在可以列出这一新模式中的表：

```
db2inst1@db2rules:~> db2 list tables for schema myschema
```

Table/View	Schema	Type	Creation time
STORE	MYSHEMA	T	2010-03-31-00.16.27.223473

1 record(s) selected.

我们也可以通过下面的命令来查看所有模式中的表：

```
db2inst1@db2rules:~> db2 list tables for all
```

现在你也许会奇怪为什么会有人希望使用 **CREATE SCHEMA** 语句来创建模式。显式创建模式的主要原因在于进行 **访问控制**。显式创建的模式有一个所有者，它既可以通过执行 **CREATE SCHEMA** 语句的用户的权限ID来确定，也可以通过在创建模式时提供用来识别所有者的权限ID来确定（在我们的例子就是 **db2inst1**）。模式所有者有权创建、改变及删除在模式中存储的任何对象；有权删除 (**drop**) 模式本身；也有权把这些特权授予其他用户。

最后，除了访问控制方面的好处之外，我们还可以在同一个数据库中拥有具有相同名称的表。这是因为各个对象的名称只需在其模式中是唯一的即可。我们来看一下。

在我们的 **db2inst1** 模式中已经有一个称为 **EMPLOYEES** 的表；现在我们创建另一个名为 **employees** 的表，但是在 **myschema** 模式之中：

```
db2inst1@db2rules:~> db2 "CREATE TABLE myschema.employees
                          (storeid INTEGER,
                           address CHAR(50) )"

```

这成功了，因为它在不同的模式之中，虽然是在同一个数据库之中！

在上面这些例子中，我都是以数据库来举例的，但模式也可以用于其他对象，例如：视图、索引、用户定义的数据类型、用户定义的函数、昵称、程序包、触发器，等等。

5.2 视图

*视图*是另一种展示存在于一个或多个表中的数据的方式。一个视图中可以包含来自一个或多个基本表的全部或部分列。

在本节中，我们将创建一份视图，它省略了表中的某些数据，因此可以对最终用户屏蔽某些表数据。

在本例中，我们将创建一份 **EMPLOYEES** 视图，其中将省略部门雇员信息，并对前两列重命名。

就是说，这是我们打算获得的东西：

列	类型
employee_id	INTEGER
first_name	CHAR(50)

Dept	CHAR(9)
-------------	----------------

为了定义视图，我们必须使用 **CREATE VIEW** 语句，如下所示：

```
db2inst1@db2rules:~> db2 "CREATE VIEW empview (employee_id, first_name)
AS SELECT EMPID, NAME
FROM employees"
```

检验是否创建了视图：

```
db2inst1@db2rules:~> db2 list tables
```

Table/View	Schema	Type	Creation time
EMPLOYEES	DB2INST1	T	2010-03-30-16.37.05.046385
EMPVIEW	DB2INST1	V	2010-03-31-21.22.26.130570

2 record(s) selected.

现在，描绘该视图，以确保它是按照我们原来设想的方式建立的：

```
db2inst1@db2rules:~> db2 describe table empview
```

Column name	Data type schema	Data type name	Column Length	Scale	Nulls
EMPLOYEE_ID	SYSTEM	INTEGER		4	0 Yes
FIRST_NAME	SYSTEM	CHARACTER		50	0 Yes

2 record(s) selected.

这符合我们原来的设想。作为最后测试，我们发出一个 **SELECT *** 语句，从该视图中检索所有数据：

```
db2inst1@db2rules:~> db2 "select * from empview"
```

EMPLOYEE_ID	FIRST_NAME
1	Adam
2	John
3	Peter

3 record(s) selected.

请注意，列名称已经按照我们期望的方式改变了，我们无法获得任何来自部门列的数据，因为我们没有在视图中包含它。

利用类似的方法，我们还可以创建把来自不同基本表的数据组合到一起的视图，也可以根据其他视图或者根据视图和表的组合来创建视图。我们现在就不对这些内容举例了，但应当知道可以实现这些功能。

虽然视图看起来与基本表很相似，但视图中并不包含真实的数据。相反，视图只是引用存储在其他基本表中的数据。只有视图定义本身实际存储在数据库中。（实际上，当修改视图中展示的数据时，修改实际上是针对存储在视图所引用的基本表中的数据进行的）。

例如，下面我们更新该视图，然后检验基础表中是否包含了相应的修改。

```
db2inst1@db2rules:~> db2 "update empview
      SET FIRST_NAME='Piotr'
      WHERE employee_id=3"
```

检验一下。

```
db2inst1@db2rules:~> db2 "SELECT * FROM employees"
```

EMPID	NAME	DEPT
1	Adam	A01
2	John	B01
3	Piotr	A01

3 record(s) selected.

5.3 别名

别名是表或视图的替代名称。引用别名的方式，与引用别名所指代的表或视图相同。

别名是公开引用的名称，因此，无需特别的权限或特权就可以使用它们。但是，访问别名所指代的表或视图仍然需要适当的权限。

可以通过执行 **CREATE ALIAS SQL** 语句来创建别名。

我们来看看如何为我们在本节中使用的 **EMPLOYEES** 表创建别名（名为 **EMPINFO**）。

```
db2inst1@db2rules:~> db2 CREATE ALIAS empinfo FOR employees
```

现在我们拥有了这个 **empinfo** 别名，我们可以用它来引用基础性的 **employees** 表，而不是直接使用表名称。

为了查看该别名，可以使用以下命令列出这些表：

```
db2inst1@db2rules:~> db2 list tables
```

Table/View	Schema	Type	Creation time
EMPINFO	DB2INST1	A	2010-07-05-11.05.48.124653
EMPLOYEES	DB2INST1	T	2010-07-05-16.37.05.046385
EMPVIEW	DB2INST1	V	2010-07-05-16.30.40.431174

```
3 record(s) selected.
```

我们可以用一个简单的查询语句来试试我们新创建的别名。

```
db2inst1@db2rules:~> db2 "SELECT * FROM empinfo "
```

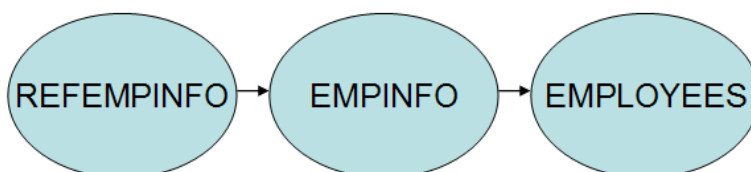
EMPID	NAME	DEPT
1	Adam	A01
2	John	B01
3	Peter	A01

3 record(s) selected.

如你所看到的那样，它输出的结果与对原始表名称进行查询的结果是一样的，因为归根结底，它们引用的是同一个表。

与表和视图相似的是，别名也可以创建、删除并拥有与之关联的注释。而与表不同的是（但与视图类似），别名可以引用其他的别名 — 这个过程称为 **链接 (chaining)**。我们来看看这样的例子。

现在我们有 **EMPLOYEES** 表和 **EMPINFO** 别名。我们再创建另一个别名 (**REFEMPINFO**)，它引用 **EMPINFO** 别名，以此方式创建这种链接。这种情况可以由下图来表达：



对此，只需再次执行 **CREATE ALIAS** 命令即可，但这一次将引用 **EMPINFO** 别名，而不是基础性的 **EMPLOYEES** 基本表名称：

```
db2inst1@db2rules:~> db2 CREATE ALIAS refempinfo FOR empinfo
```

下面列出所有的表，以查看这个新的别名：

```
db2inst1@db2rules:~> db2 list tables
```

Table/View	Schema	Type	Creation time
EMPINFO	DB2INST1	A	2010-07-05-11.05.48.124653
EMPLOYEES	DB2INST1	T	2010-07-05-16.37.05.046385
EMPVIEW	DB2INST1	V	2010-07-05-16.30.40.431174
REFEMPINFO	DB2INST1	A	2010-07-05-16.42.36.059937

4 record(s) selected.

我们再次查询这个别名，它将从该别名所引用的基础表中检索数据：

```
db2inst1@db2rules:~> db2 "SELECT * FROM refempinfo"
```

EMPID	NAME	DEPT
1	Adam	A01
2	John	B01
3	Peter	A01

3 record(s) selected.

总之，通过使用别名，可以使 SQL 语句独立于别名所指代的、能够识别出基本表或视图的限定名称。

5.4 索引

索引是关于数据表中各行的一系列有序指针。DB2 能够使用指针来确保唯一性并提高性能，这是通过对数据进行集群、对数据进行分区以及为数据查询提供高效的访问路径来实现的。在多数情况下，对有索引的数据进行访问的速度要快于对数据进行扫描。

索引有三个主要目的：

- 提高性能。
- 确保每一行是唯一的。
- 对数据进行集群。

索引在存储时是与表中的数据分开的。各个索引在物理上存储在各自的索引空间中。

在本节中我们将讨论两个索引的例子。一个例子用来说明索引如何有利于保证每一行是唯一的，另一个例子用来说明我们如何提高数据库的查询性能。

在前面的例子中，我们创建了具有下面结构的 EMPLOYEES 表：

列	类型
empid	INTEGER
name	CHAR(50)
Dept	CHAR(9)

在创建该表时，我们未定义任何主键或唯一约束。这样一来，我们就可能把多个具有相同员工 ID 的项目输入到该表中，而这不是我们希望看到的情况。对此，我们可以使用索引来确保在该表中不会出现两个具有相同 EMPID 值的项目：

```
db2inst1@db2rules:~> db2 "CREATE UNIQUE INDEX unique_id
ON employees(empid) "
```

注：在你打算创建索引的键值上，如果其中存在重复的输入项，而你打算利用该输入项创建唯一索引，那么你就会收到下面的消息。在创建唯一索引时，不能存在重复的输入项：

```
SQL0603N  A unique index cannot be created because the table
contains data that would result in duplicate index entries.
SQLSTATE=23515
```

利用 **DESCRIBE** 命令验证是否创建了索引：

```
db2inst1@db2rules:~> db2 DESCRIBE INDEXES FOR TABLE employees
```

输出结果将类似于下面的内容：

Index schema	Index name	Unique rule	Number of columns	Index type	Index partitioning
DB2INST1	UNIQUE_ID	U		1 RELATIONAL DATA	-

1 record(s) selected.

“唯一规则” (unique rule) 列指明该索引是否为唯一的。有三种不同的唯一规则类型：

- **D** = 意味着允许重复
- **P** = 意味着主索引
- **U** = 意味着唯一索引

现在，我们试试用一个已经存在的员工 ID 插入一行（例如，EMPID=3）

```
db2inst1@db2rules:~> db2 "INSERT INTO employees VALUES(3, 'William',
'A01')"
```

你将会收到一条错误消息，它指出：

```
SQL0803N  One or more values in the INSERT statement, UPDATE
statement, or foreign key update caused by a DELETE statement are
not valid because the primary key, unique constraint or unique
index identified by "1" constrains table "DB2INST1.EMPLOYEES"
from having duplicate values for the index key.
SQLSTATE=23505
```

这意味着我们的唯一索引发挥作用了，因为我们无法用一个已经存在的员工 ID 插入一行（我们根据该属性定义了我们的索引）。

此外，如前所述，索引还有助于提高性能。我们还可以利用索引收集统计信息。收集索引统计信息能够让优化器评估评估是否应当使用索引来完成查询。

我们可以创建能够自动收集统计信息索引：

```
db2inst1@db2rules:~> db2 "CREATE INDEX idx
                           ON employees(dept) COLLECT STATISTICS "
```

可以像前面一样利用 **DESCRIBE** 命令来查看索引及其属性：

```
db2inst1@db2rules:~> db2 DESCRIBE INDEXES FOR TABLE employees
```

除了体验到性能上的变化之外，表的用户不会知道是否在使用索引。**DB2** 自行决定是否使用索引来访问某个表。

要知道，索引是把双刃剑。大量的索引一方面可以提高某个特定事务的访问性能，但同时也需要对插入、更新及删除索引键进行额外的处理。在你创建索引之后，**DB2** 将维护该索引，但你也可以进行必要的维护，例如在必要时对索引进行重组或恢复。

5.5 序列

序列是用来自动生成数据值的对象。

序列具有如下特征：

- 所生成的值可以是任何准确的数字数值类型，其小数部分为零。
- 相邻值可以相差任何规定的增量值。
- 计数器值是可以恢复的（在必要时可以从日志中重新构建）。
- 所生成的值可以缓存，以提高性能。

此外，序列可以采用下述方式之一生成值：

- 按照规定的值递增或递减，没有边界
- 按照规定的值递增或递减至某个用户定义的限制位置，然后停止
- 按照规定的值递增或递减至某个用户定义的限制位置，然后循环会起点并重新开始

我们现在就开始创建一个名为 **emp_id** 的序列，它起始于 **4**，增量为 **1**，不循环，而且每次缓存 **5** 个值。为此，我们必须发出以下语句：

```
db2inst1@db2rules:~> db2 "CREATE SEQUENCE emp_id
                           START WITH 4
                           INCREMENT BY 1
                           NO CYCLE
                           CACHE 5"
```

我们将使用该序列把一个新员工插入到我们的表中，但无需显示指明具体的员工 **ID**；该序列会替我们把这些事情处理好。

为便于在 SQL 操作中使用序列，提供了两个表达式：**PREVIOUS VALUE** 和 **NEXT VALUE**。**PREVIOUS VALUE** 表达式返回为指定的序列最近生成的值，而 **NEXT VALUE** 表达式返回下一个值。

下面利用我们新创建的序列中的 **NEXT VALUE** 在部门 **B01** 中插入一个名为 **Daniel** 的新雇员：

```
db2inst1@db2rules:~> db2 "INSERT INTO EMPLOYEES
                           VALUES (NEXT VALUE FOR emp_id, 'Daniel',
                                   'B01')"
```

对该表进行一次查询，看看我们的序列是否发挥了作用。

```
db2inst1@db2rules:~> db2 "SELECT * FROM employees"
```

EMPID	NAME	DEPT
1	Adam	A01
2	John	B01
3	Piotr	A01
4	Daniel	B01

4 record(s) selected.

我们看到，该序列能够正常工作。可以继续使用 **NEXT VALUE** 语句，增量为 1。例如：

```
db2inst1@db2rules:~> db2 "INSERT INTO EMPLOYEES
                           VALUES (NEXT VALUE FOR emp_id, 'Stan',
                                   'B01')"
```

```
db2inst1@db2rules:~> db2 "SELECT * FROM employees"
```

EMPID	NAME	DEPT
1	Adam	A01
2	John	B01
3	Piotr	A01
4	Daniel	B01
5	Stan	B01

5 record(s) selected.

但是，由于我们每次缓存 5 个值，我们必须小心，因为该值规定了将提前生成并保存在内存中的标识序列中值的个数。（在为序列生成值时，预先生成并缓存这些值可以减少对日志的同步 I/O。但是，如果系统发生故障，那么，所有已缓存的，但并未被已落实的语句使用的序列值均会丢失；就是说，它们永远也不会被使用了。）

我们来看看这种情况。请终止与数据库的连接，然后重新连接。

```
db2inst1@db2rules:~> db2 terminate
```

```
db2inst1@db2rules:~> db2 connect to testdb
```

现在像我们前面的操作那样，利用序列添加一个项目，然后使用 **SELECT *** 进行验证。

```
db2inst1@db2rules:~> db2 "INSERT INTO EMPLOYEES
VALUES (NEXT VALUE FOR emp_id, 'Bill',
'B01')"
```

```
db2inst1@db2rules:~> db2 "SELECT * FROM employees"
```

EMPID	NAME	DEPT
1	Adam	A01
2	John	B01
9	Bill	B01
3	Piotr	A01
4	Daniel	B01
5	Stan	B01

6 record(s) selected.

请注意，下一个值是 **9**，而不是 **6**。为什么？因为我们规定，在终止连接之前，在内存中缓存后面的 **5** 个值。就是说，我们在内存中有了下面几个值：**4, 5, 6, 7, 8**。当连接丢失之后（系统故障），我们就丢失了这 **5** 个值，现在无法再使用它们了，于是我们只能用这些被缓存的值之后的下一个值来继续该序列。

6. 使用 SQL

在本节中你将练习 SQL 语句。你将使用 IBM Data Studio 和 SAMPLE 数据库。为了确保你拥有一部“干净的”SAMPLE 数据库，我们先删除它，然后再重新创建（从 DB2 命令窗口中或者从 Linux 外壳中）：

```
db2 force applications all
db2 drop db sample
db2 drop db mysample
db2sampl
```

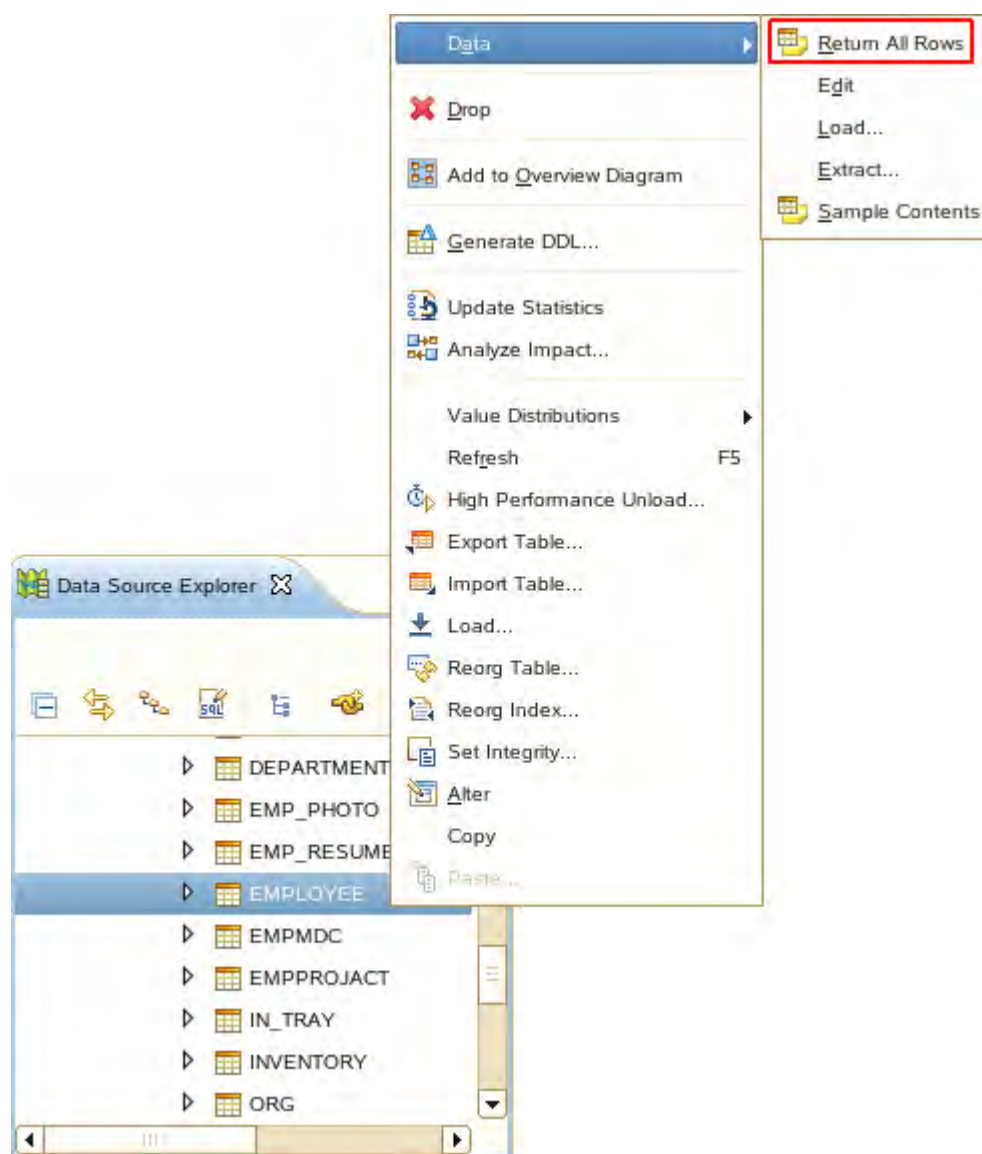
6.1 查询数据

如果无法从数据库中获得数据的话，数据库就没有太大用途，因此，我们必须掌握如何使用 **SELECT SQL** 语句从表中检索数据。

6.1.1 利用 **Data Studio** 从表中检索所有的行

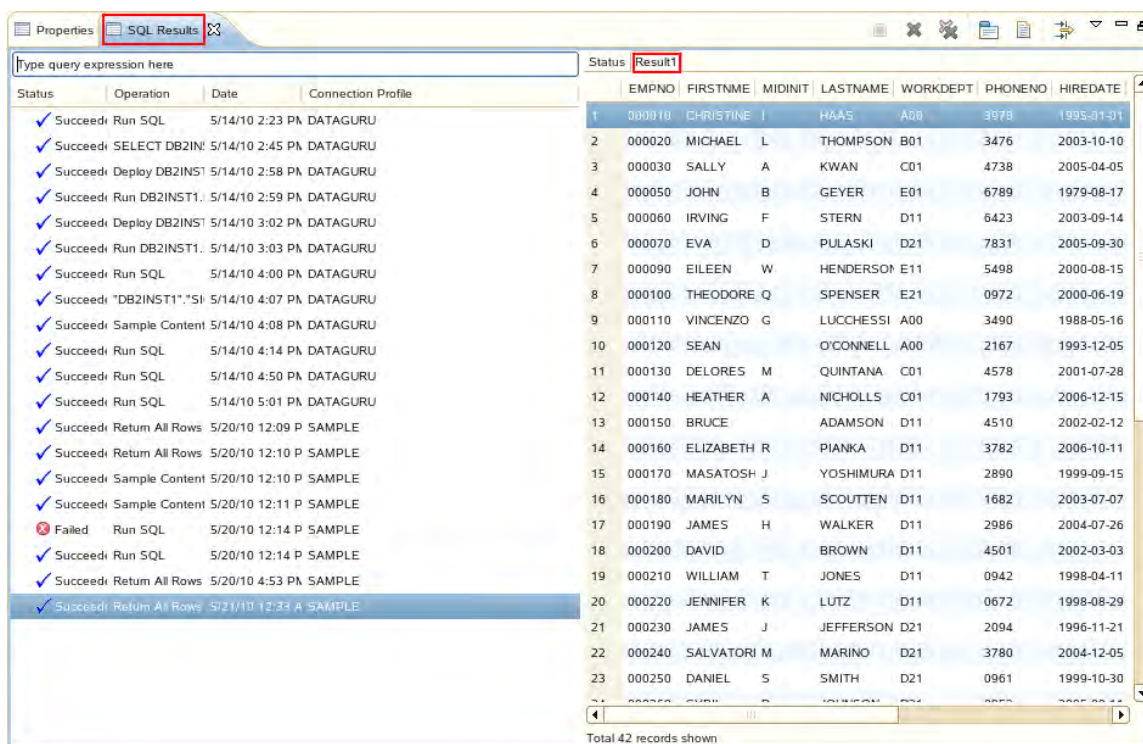
在使用 **SQL** 之前，我们来快速介绍一下如何利用 **Data Studio** 选项从表中检索各行，而无需编写 **SQL** 代码。

1. 在 **Data Source Explorer** 视图中，定位到你打算从其中返回所有行的表。例如：**SAMPLE [DB2 for Linux...] > SAMPLE > Schemas [Filtered] > DB2INST1 > Tables > EMPLOYEE.**
2. 右击表 **EMPLOYEE**，选择 **Data**（数据），然后点击 **Return All Rows**（返回所有行）。



3. 正如我们在 **SQL Results (SQL 结果)** 选项卡中所看到的那样，该操作成功完成。**EMPLOYEE** 表中的各行都显示在右侧的 **Result1** 选项卡中。

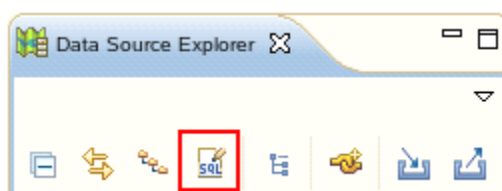
通过点击各个视图右上角相应的图标，你可以随时展开或者恢复视图。



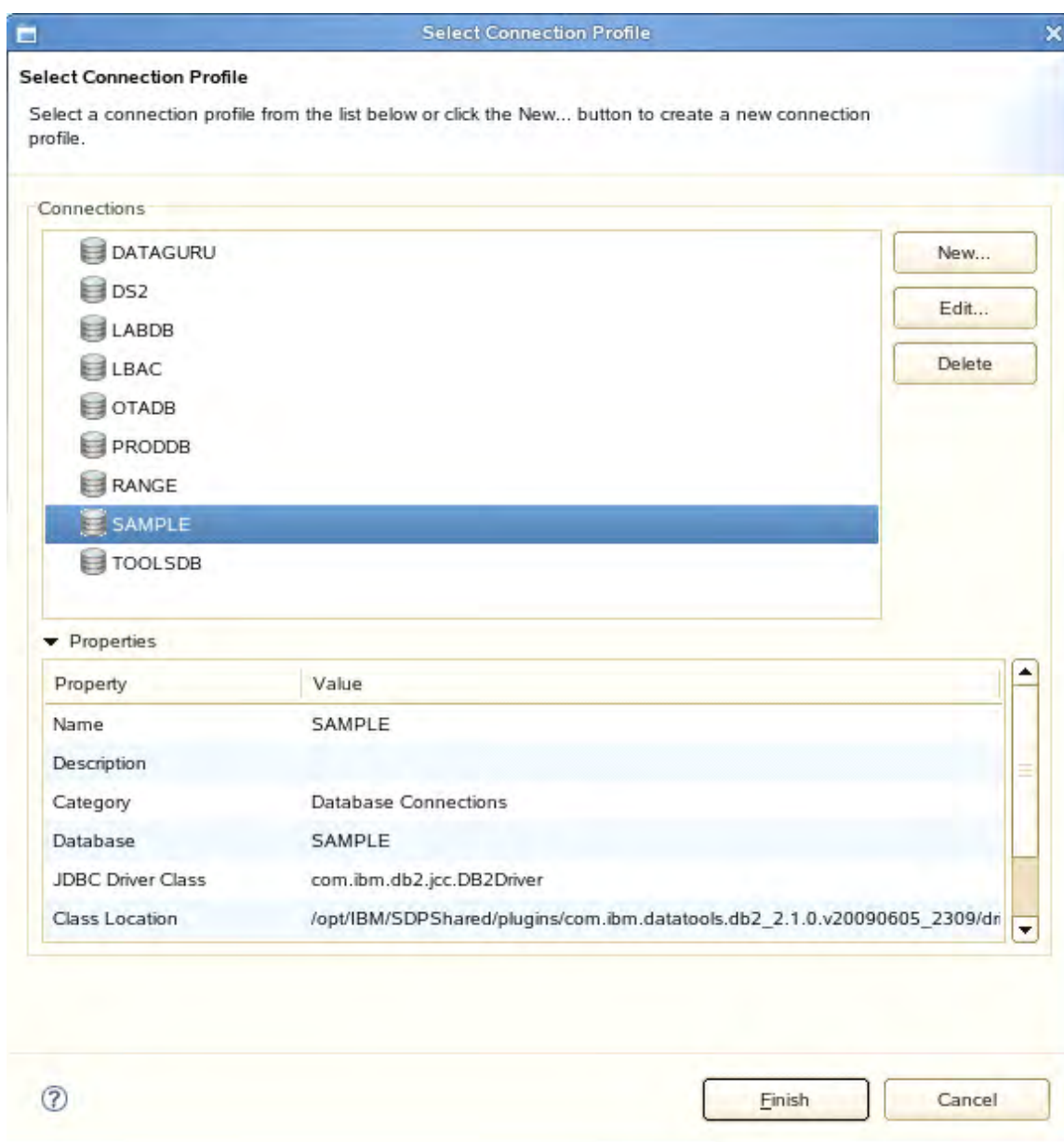
6.1.2 利用 SELECT 语句检索各行

请遵循下面的步骤来学习如何在 Data Studio 中执行 SELECT 语句。

1. 在 **Data Source Explorer** 工具栏中，点击图标 **New SQL Script**（新建 SQL 脚本）。



2. 在所出现的 **Select Connection Profile**（选择连接配置文件）窗口中，选择 **SAMPLE** 并点击 **Finish**（完成）。

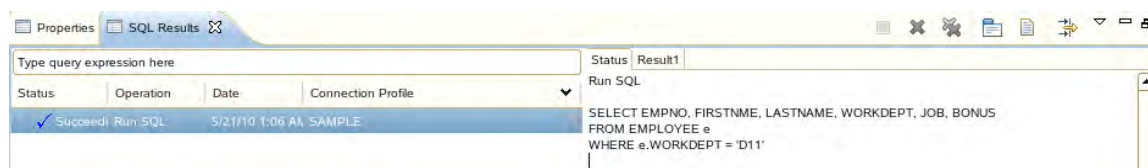


3. 主视图将出现一个新选项卡。现在我们使用 **WHERE** 子句编写一条SQL查询，比如说，我们想知道 部门 D11 的经理得到多少BONUS（奖金）。可以在该新出现的选项卡中输入下面的查询：

```
SELECT EMPNO, FIRSTNME, LASTNAME, WORKDEPT, JOB, BONUS
FROM EMPLOYEE e
WHERE e. WORKDEPT = 'D11'
```

4. 在主菜单中，选择 **Run > Run SQL**

5. 请注意，**SQL Results**（SQL 结果）视图在屏幕底部进入前景。点击图标使该视图最大化。SQL 结果视图能够说明指出该 SQL 脚本是否得到成功执行。在右侧的**Status**（状态）选项卡中，列出了该脚本文件中语句的摘要。



6. 要查看我们的 SQL 查询语句的结果，可以点击右侧的 **Result1** 选项卡。

	EMPNO	FIRSTNME	LASTNAME	WORKDEPT	JOB	BONUS
1	000060	IRVING	STERN	D11	MANAGER	500.00
2	000150	BRUCE	ADAMSON	D11	DESIGNER	500.00
3	000160	ELIZABETH	PIANKA	D11	DESIGNER	400.00
4	000170	MASATOSHI	YOSHIMURA	D11	DESIGNER	500.00
5	000180	MARILYN	SCOUTTEN	D11	DESIGNER	500.00
6	000190	JAMES	WALKER	D11	DESIGNER	400.00
7	000200	DAVID	BROWN	D11	DESIGNER	600.00
8	000210	WILLIAM	JONES	D11	DESIGNER	400.00
9	000220	JENNIFER	LUTZ	D11	DESIGNER	600.00
10	200170	KIYOSHI	YAMAMOTO	D11	DESIGNER	500.00
11	200220	REBA	JOHN	D11	DESIGNER	600.00

7. 把 **SQL Results** 视图恢复至原始状态，通过点击 **X** 图标关闭主窗口中的 **Script.sql** 选项卡。

6.1.2.1 试试看：练习 **SELECT** 语句

为下面所述的查询创建 SQL 语句。然后你可以把你的答案与本实验末尾给出的参考答案相比较。

- 从 **SALES** 表中寻找“Ontario-South”地区名为“*LEE*”的销售人员的全部销售信息。
- 从 **DEPARTMENT** 表中寻找所有已经安排了经理的部门的名称。

小提示：没有经理的部门在其 **MGRNO** 列中显示 NULL。

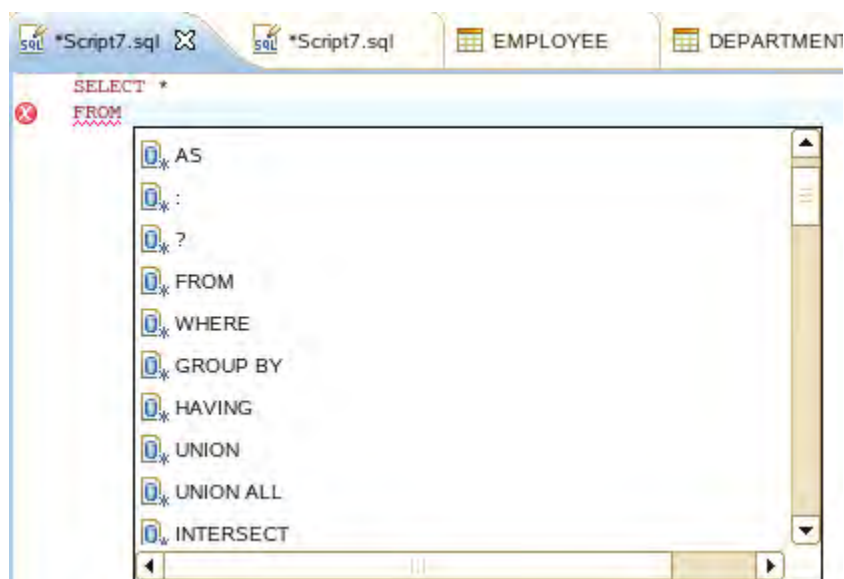
6.1.3 对结果排序

ORDER BY 语句能够根据一列或多列对结果集合进行排序。在默认情况下，执行 SQL 语句后所返回的记录是按照升序排序的，但我们可以通过关键字 **DESC** 把它改为降序。

```
SELECT column_name(s)
FROM table_name
ORDER BY column_name(s) ASC|DESC
```

现在我们在 **SAMPLE** 数据库中运行一个例子。在表 **STAFF** 中，对部门 66 的所有人员依据其工资按照降序排序。

小提示：通过同时按下 **Ctrl + Space**，你可以调用代码助手功能。通过该功能，你无需再键入整个单词，而是可以从预定义的关键字列表中进行选择。



现在运行下面的查询：

```
SELECT *
FROM STAFF
WHERE DEPT = '66'
ORDER BY SALARY DESC
```


从上面返回的表中可以看到，在 **JOB** 列之后显示出一个新建的称为 **AVG_SALARY** 的列，其中含有我们所要寻找的结果。

该 SQL 背后的运行逻辑是：**GROUP BY JOB** 子句指示 DB2 对 **JOB** 列中具有相同值的各行进行分组，例如，划分为 *Clerk*、*Manager*，等等。然后，再利用 **AVG** 函数对各个组的 **SALARY** 列计算平均工资。

6.1.4.1 试试看：练习聚集函数

- 寻找 **SALES** 表中每名销售人员的合计销售额。
- 统计各种职位中男性员工的人数。

6.1.5 从多个表在检索数据（连接）

SQL 的连接语句可以用来从两个或更多个表中查询数据，其根据是这些表中某些列之间的关系。

数据库中的表通常通过使用外键 (FK) 相互建立联系；所谓外键，就是第二个表中的主键 (PK)。当我们使用 **JOIN** 查询数据时，我们最有可能是根据 PK 和 FK 之间的这些关系把两个或多个表连接起来。

在继续讨论更多例子之前，我们先列出你可能遇到的各类 **JOIN** 以及它们之间的差异。

1. **(INNER) JOIN**（（内）连接）：返回具有相应 PK 及 FK 的所有行。
2. **LEFT (OUTER) JOIN**（左（外）连接）：返回具有相应 PK 及 FK 的所有行，另加左侧表中所有与右侧表不相匹配的行。
3. **RIGHT (OUTER) JOIN**（右（外）连接）：返回具有相应 PK 及 FK 的所有行，另加右侧表中所有与左侧表不相匹配的行。
4. **FULL (OUTER) JOIN**（全部（外）连接）：返回具有相应 PK 及 FK 的所有行，另加左侧表中所有与右侧表不相匹配的行，再加上右侧表中所有与左侧表不相匹配的行。

现在，我们从 **EMP_PHOTO** 表和 **EMPLOYEE** 表中找出谁以位图格式上载了员工图片。试试下面的查询：

```
SELECT e.EMPNO, p.PHOTO_FORMAT, e.FIRSTNAME, e.LASTNAME
FROM EMPLOYEE e, EMP_PHOTO p
```

```
WHERE e.EMPNO = p.EMPNO AND p.PHOTO_FORMAT = 'bitmap'
```

注：**EMP_PHOTO** 中的 *p.EMPNO* 实际上是一个 **FK**，它指向 *e.EMPNO*，即 **EMPLOYEE** 表中的 **PK**。

Status	Result1	EMPNO	PHOTO_FORMAT	FIRSTNME	LASTNAME
1		000130	bitmap	DELORES	QUINTANA
2		000140	bitmap	HEATHER	NICHOLLS
3		000150	bitmap	BRUCE	ADAMSON
4		000190	bitmap	JAMES	WALKER

上述 SQL 语句背后的原理是这样的：首先，**FROM** 子句中的表结合到一起形成一张大表。然后，**WHERE** 子句负责过滤这种大表中的行。最后，由 **SELECT** 子句返回所有匹配行的列。这被称为 **INNER JOIN**（内连接）的“隐式连接表达”。具有“显式连接表达”的等价查询如下所示：

```
SELECT e.EMPNO, p.PHOTO_FORMAT, e.FIRSTNME, e.LASTNAME
FROM   EMPLOYEE e INNER JOIN EMP_PHOTO p
      ON e.EMPNO = p.EMPNO
      AND p.PHOTO_FORMAT = 'bitmap'
```

现在，我们来运行一个类似的查询，但采用 **LEFT OUTER JOIN**（左外连接），而不是上面的 **INNER JOIN**（内连接）。运行下面的查询：

```
SELECT e.EMPNO, p.PHOTO_FORMAT, e.FIRSTNME, e.LASTNAME
FROM   EMPLOYEE e LEFT OUTER JOIN EMP_PHOTO p
      ON e.EMPNO = p.EMPNO
      AND p.PHOTO_FORMAT = 'bitmap'
```

这一次，得到的结果长了许多。根据前面的解释可以看出，外连接并不要求两个被连接的表中的每条记录具有匹配的记录。表 **EMPLOYEE** 和表 **EMP_PHOTO** 左（外）连接的结果总是含有“左侧”表 (**EMPLOYEE**) 中的所有记录，即使连接条件没有在“右侧”表 (**EMP_PHOTO**) 中找到任何匹配的记录，这正是 *PHOTO_FORMAT* 列中 *NULL* 值的来历。

Status	Result1			
	EMPNO	PHOTO_FORMAT	FIRSTNME	LASTNAME
1	000130	bitmap	DELORES	QUINTANA
2	000140	bitmap	HEATHER	NICHOLLS
3	000150	bitmap	BRUCE	ADAMSON
4	000190	bitmap	JAMES	WALKER
5	000110	NULL	VINCENZO	LUCCHESSI
6	000180	NULL	MARILYN	SCOUTTEN
7	200010	NULL	DIAN	HEMMINGER
8	200330	NULL	HELENA	WONG
9	000200	NULL	DAVID	BROWN
10	000280	NULL	ETHEL	SCHNEIDER
11	000310	NULL	MAUDE	SETRIGHT
12	300001	NULL	Paul	Pierce
13	000070	NULL	EVA	PULASKI
14	000170	NULL	MASATOSHI	YOSHIMURA
15	000240	NULL	SALVATORE	MARINO
16	000260	NULL	SYBIL	JOHNSON
17	000270	NULL	MARIA	PEREZ
18	200280	NULL	EILEEN	SCHWARTZ
19	000100	NULL	THEODORE	SPENSER
20	000230	NULL	JAMES	JEFFERSON
21	000250	NULL	DANIEL	SMITH
22	000320	NULL	RAMLAL	MEHTA
23	200240	NULL	ROBERT	MONTEVERDE
24	000020	NULL	MICHAEL	THOMPSON
25	000050	NULL	JOHN	GEYER
26	000210	NULL	WILLIAM	JONES
27	000290	NULL	JOHN	PARKER
28	200120	NULL	GREG	ORLANDO
29	200310	NULL	MICHELLE	SPRINGER
30	000010	NULL	CHRISTINE	HAAS
31	000060	NULL	IRVING	STERN
32	000220	NULL	JENNIFER	LUTZ
33	000330	NULL	WING	LEE
34	200220	NULL	REBA	JOHN
35	200340	NULL	ROY	ALONZO
36	000030	NULL	SALLY	KWAN
37	000120	NULL	SEAN	O'CONNELL
38	000300	NULL	PHILIP	SMITH
39	000090	NULL	EILEEN	HENDERSON
40	000160	NULL	ELIZABETH	PIANKA
41	000340	NULL	JASON	GOUNOT
42	200170	NULL	KIYOSHI	YAMAMOTO
43	200140	NULL	KIM	NATZ

6.1.5.1 试试看：练习连接

- 假设你想知道行动信息（其 *ACTNO* 大于 100）以及各个项目行动的设计人姓名（从 **PROJECT** 表中查找）。请列出这些信息，并按照设计人的姓名以字母顺序对结果排序。
- 把 **EMPLOYEE** 表和 **DEPARTMENT** 表连接起来，假设 **EMPLOYEE** 表中的 *WORKDEPT* 是 **FK**，它指向 *DEPTNO*，即 **DEPARTMENT** 表中的 **PK**。保存结果，然后再重复该查询，但分别使用 **LEFT OUTER JOIN**（左外连接）、**RIGHT OUTER JOIN**（右外连接）以及 **FULL OUTER JOIN**（全部外连接）。比较这些结果。

6.2 插入、更新和删除

假设现在我们需要把新产品信息输入的数据库中；或者李女士被提拔了，因此她的 **JOB** 和 **SALARY** 都需要相应地修改；或者 Bryant 先生在项目工作中表现不积极，因此被辞退了，我们应当把他继续保留在员工表中？

对于这些情况，我们可以使用 SQL 的 **INSERT**、**UPDATE** 及 **DELETE** 语句对表数据进行操纵。

6.2.1 插入

INSERT（插入）语句被用于向表中添加新行。例如，假设 NBA 球员 Paul Pierce 从其篮球职业生涯中退役了，并成功地在你的公司中找到了职位。现在你需要把他的信息添加到 **EMPLOYEE** 表中。

运行下面的查询：

```
INSERT INTO EMPLOYEE(EMPNO, FIRSTNME, LASTNAME, EDLEVEL)
VALUES (300001, 'Paul', 'Pierce', 18)
```

```
Status |
Run SQL

INSERT INTO EMPLOYEE(EMPNO, FIRSTNME, LASTNAME, EDLEVEL)
VALUES (300001, 'Paul', 'Pierce', 18)
Updated 1 rows.
|
```

如果我们运行 **SELECT * FROM EMPLOYEE**，我们就可以看到 Paul Pierce 现在已经顺利地成为 **EMPLOYEE** 表的一部分，所有未指明的列均填写默认值，在本例中为 *NULL*。

Status	Result1													
	EMP	FIRSTNME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE	JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM
1	300001	Paul	NULL	Pierce	NULL	NULL	NULL	NULL 18	NULL	NULL	NULL	NULL	NULL	NULL
2	200340	ROY	R	ALONZO	E21	5698	1997-07-05	FIELI 16		M	1956-05-17	31840.00	500.00	1907.00
3	200330	HELENA		WONG	E21	2103	2006-02-23	FIELI 14		F	1971-07-18	35370.00	500.00	2030.00
4	200310	MICHELLE	F	SPRINGER	E11	3332	1994-09-12	OPEF 12		F	1961-04-21	35900.00	300.00	1272.00
5	200280	EILEEN	R	SCHWARTZ	E11	8997	1997-03-24	OPEF 17		F	1966-03-28	46250.00	500.00	2100.00
6	200240	ROBERT	M	MONTEVERI	D21	3780	2004-12-05	CLEF 17		M	1984-03-31	37760.00	600.00	2301.00
7	200220	REBA	K	JOHN	D11	0672	2005-08-29	DESI 18		F	1978-03-19	69840.00	600.00	2387.00
8	200170	KIYOSHI		YAMAMOTO	D11	2890	2005-09-15	DESI 16		M	1981-01-05	64680.00	500.00	1974.00
9	200140	KIM	N	NATZ	C01	1793	2006-12-15	ANAL 18		F	1976-01-19	68420.00	600.00	2274.00
10	200120	GREG		ORLANDO	A00	2167	2002-05-05	CLEF 14		M	1972-10-18	39250.00	600.00	2340.00
11	200010	DIAN	J	HEMMINGE	A00	3978	1995-01-01	SALE 18		F	1973-08-14	46500.00	1000.00	4220.00
12	000340	JASON	R	GOUNOT	E21	5698	1977-05-05	FIELI 16		M	1956-05-17	43840.00	500.00	1907.00

INSERT 语句中还可以含有子查询，例如，利用 **SELECT** 子句同时插入多项记录。我们来试试。首先执行下面的 DDL 语句：

```
CREATE TABLE MNG_PEOPLE LIKE EMPLOYEE
```

该语句创建了一个称为 **MNG_PEOPLE** 的新表，它继承了 **EMPLOYEE** 表的所有属性/列定义。

然后我们从 **EMPLOYEE** 表中 **SELECT** 所有的经理，并把他们插入到新创建的 **MNG_PEOPLE** 表中。

```
INSERT INTO MNG_PEOPLE SELECT * FROM EMPLOYEE WHERE JOB = 'MANAGER'
```

为了检查该操纵是否成功，可以检索 **MNG_PEOPLE** 表的所有列。你会看到有 7 条记录被成功插入到该新表中。

Status	Result1													
	EMPNO	FIRSTNME	MI	LASTNAME	WOR	PHONE	HIRE	JOB	EDL	SEX	BIRTHDATE	SALARY	BONUS	COMM
1	000020	MICHAEL	L	THOMPSON	B01	3476	2003-	MANAGER 18		M	1978-02-02	94250.00	800.00	3300.00
2	000030	SALLY	A	KWAN	C01	4738	2005-	MANAGER 20		F	1971-05-11	98250.00	800.00	3060.00
3	000050	JOHN	B	GEYER	E01	6789	1979-	MANAGER 16		M	1955-09-15	80175.00	800.00	3214.00
4	000060	IRVING	F	STERN	D11	6423	2003-	MANAGER 16		M	1975-07-07	72250.00	500.00	2580.00
5	000070	EVA	D	PULASKI	D21	7831	2005-	MANAGER 16		F	2003-05-26	96170.00	700.00	2893.00
6	000090	EILEEN	W	HENDERSON	E11	5498	2000-	MANAGER 16		F	1971-05-15	89750.00	600.00	2380.00
7	000100	THEODORE	Q	SPENSER	E21	0972	2000-	MANAGER 14		M	1980-12-18	86150.00	500.00	2092.00

6.2.1.1 试试看：练习 INSERT 语句

1. 本公司刚设立了一个新部门，名为 *FOO*，部门代码为 *K47*，并以 *'E01'* 作为 **ADMRDEPT**。请把这条记录插入到 **DEPARTMENT** 表中。
2. 创建一个称为 **D11_PROJ** 的新表，它具有与 **PROJECT** 表相同的结构，然后把来自部门 *D11* 的、与所有项目相关的数据添加到其中。

6.2.2 更新

UPDATE（更新）语句被用来更新表中现有的记录。其语法如下：

```
UPDATE table_name
SET column1=value, column2=value2,...,column = valueN
WHERE some_column=some_value
```

注：这里的 **WHERE** 子句特别指明哪些记录将被更新。如果没有 **WHERE** 子句，本表中的所有记录都将被修改！

假设人力资源部的工作人员刚刚交给你一份关于 **Paul Pierce** 的详细信息，并要求你利用这些数据更新 **EMPLOYEE** 表中的以下各列：

```
HIREDATE: 2010-01-01
JOB:      DESIGNER
SEX:      M
BIRTHDATE: 1977-10-13
```

我们可以通过运行下面的查询来更新其个人信息：

```
UPDATE EMPLOYEE
SET HIREDATE = '2010-01-01', JOB = 'DESIGNER', SEX = 'M', BIRTHDATE =
'1977-10-13'
WHERE EMPNO = 300001
```

你可以在状态选项卡中看到，有一行被更新了。

```
Status
Run SQL

UPDATE EMPLOYEE
SET HIREDATE = '2010-01-01', JOB = 'DESIGNER', SEX = 'M', BIRTHDATE = '1977-10-13'
WHERE EMPNO = 300001
Updated 1 rows.
```


同样，如果我们运行 `SELECT * FROM EMPLOYEE`，我们会看到 Paul 的数据在这四个列中被更新了。

EMF ▲	FIRSTNAME	MIDINIT	LASTNAME	WORKDEPT	PHONENO	HIREDATE	JOB	EDLEVEL	SEX	BIRTHDATE	SALARY	BONUS	COMM	
1	300001	Paul	NULL	Pierce	NULL	NULL	2010-01-01	DESIGNER	18	M	1977-10-13	NULL	NULL	NULL

6.2.2.1 试试看：练习 UPDATE 语句

我们来看看利用 Paul 的信息还能做些什么：

- 试试把 Paul 的 EDLEVEL 更新为 'NULL'，看看会发生什么事情。
- 试试把 Paul 的 WORKDEPT 更新为 'Z11'，看看会发生什么事情。

6.2.3 删除

DELETE（删除）语句能够删除表中的各行。其语法为：

```
DELETE FROM table_name
      WHERE some_column=some_value
```

重要提示：与 UPDATE 语句的情况一样，如果你省略了 **WHERE** 子句，所有的记录都将被删除。

现在，我们从数据库中删除 Paul Pierce 的记录，因为他改变主意，又返回球场了。运行下面的查询：

```
DELETE FROM EMPLOYEE
      WHERE EMPNO = 300001
```

检查 **EMPLOYEE** 表中的内容（到目前为止，你应当已经知道至少两种检查方法了）。如果你成功地执行了 **DELETE** 语句，Paul 的记录就不会出现在结果列表中。

6.2.3.1 试试看：练习 DELETE 语句

- ▶ 试试从 **DEPARTMENT** 表中删除部门 'E21'

7. 小结

本实验向你介绍了构成 DB2 数据库的各种对象。你现在对 DB2 对象应当具有翔实知识，了解了我们为什么要使用它们以及如何创建它们。你还学习及练习了 SQL 语句。

8. 答案

第 6.1.2.1 节

查询 1

- 从 **SALES** 表中寻找“Ontario-South”地区名为“*LEE*”的销售人员的全部销售信息。

```
SELECT *  
FROM SALES  
WHERE SALES_PERSON = 'LEE'  
AND REGION = 'Ontario-South'
```

Status	Result1	SALES_DATE	SALES_PERSON	REGION	SALES
1		2005-12-31	LEE	Ontario-South	3
2		2006-03-29	LEE	Ontario-South	2
3		2006-03-30	LEE	Ontario-South	7
4		2006-03-31	LEE	Ontario-South	14
5		2006-04-01	LEE	Ontario-South	8

查询 2

- 从 **DEPARTMENT** 表中寻找所有已经安排了经理的部门的名称。

```
SELECT DEPTNAME  
FROM DEPARTMENT  
WHERE MGRNO is not NULL
```

Status	Result1
	DEPTNAME
1	SPIFFY COMPUTER SERVICE DIV.
2	PLANNING
3	INFORMATION CENTER
4	MANUFACTURING SYSTEMS
5	ADMINISTRATION SYSTEMS
6	SUPPORT SERVICES
7	OPERATIONS
8	SOFTWARE SUPPORT

第 6.1.3.1 节

查询 1

- ▶ 利用 **STAFF** 表，对所有人员按照其经验年限以降序排序。对于具有相同 **YEARS** 的人员，再按照其工资以升序排序。

```
SELECT *  
FROM STAFF  
WHERE YEARS is not NULL  
ORDER BY YEARS DESC, SALARY ASC
```

Status	Result1	ID	NAME	DEPT	JOB	YEARS	SALARY	COMM
1		310	Graham	66	Sales	13	71000.00	200.30
2		260	Jones	10	Mgr	12	81234.00	NULL
3		50	Hanes	15	Mgr	10	80659.80	NULL
4		290	Quill	84	Mgr	10	89818.00	NULL
5		210	Lu	10	Mgr	10	90010.00	NULL
6		280	Wilson	66	Sales	9	78674.50	811.50
7		270	Lea	66	Mgr	9	88555.50	NULL
8		190	Sneider	20	Clerk	8	34252.75	126.50
9		20	Pemal	20	Sales	8	78171.25	612.45
10		340	Edwards	84	Sales	7	67844.00	1285.00
11		70	Rothman	15	Sales	7	76502.83	1152.00
12		100	Plotz	42	Mgr	7	78352.80	NULL
13		160	Molinare	10	Mgr	7	82959.20	NULL
14		220	Smith	51	Sales	7	87654.50	992.80
15		10	Sanders	20	Mgr	7	98357.50	NULL
16		90	Koonitz	42	Sales	6	38001.75	1386.70
17		130	Yamaguch	42	Clerk	6	40505.90	75.60
18		250	Wheeler	51	Clerk	6	74460.00	513.30
19		40	O'Brien	38	Sales	6	78006.00	846.55
20		150	Williams	51	Sales	6	79456.50	637.65
21		140	Fraye	51	Mgr	6	91150.00	NULL
22		110	Ngan	15	Clerk	5	42508.20	206.60
23		350	Gafney	84	Clerk	5	43030.50	188.00
24		300	Davis	84	Sales	5	65454.50	806.10
25		30	Marenghi	38	Mgr	5	77506.75	NULL
26		240	Daniels	10	Mgr	5	79260.25	NULL
27		170	Kermisch	15	Clerk	4	42258.50	110.10
28		320	Gonzales	66	Sales	4	76858.20	844.00
29		180	Abrahams	38	Clerk	3	37009.75	236.50
30		230	Lundquist	51	Clerk	3	83369.80	189.65
31		330	Burke	66	Clerk	1	49988.00	55.50

Total 31 records shown

第 6.1.4.1 节

查询 1

- 寻找 **SALES** 表中每名销售人员的合计销售额。

```
SELECT SALES_PERSON, SUM(SALES) AS total_sales
FROM SALES
GROUP BY SALES_PERSON
```

Status	Result1
	SALES_PERSON TOTAL_SALES
1	GOUNOT 50
2	LEE 91
3	LUCCHESI 14

查询 2

- 统计各种职位中男性员工的人数。

```
SELECT JOB, COUNT(*) as TOTAL_NUM
FROM EMPLOYEE
WHERE SEX = 'M'
GROUP BY JOB
```

Status	Result1
	JOB TOTAL_NUM
1	CLERK 6
2	DESIGNER 6
3	FIELDREP 4
4	MANAGER 4
5	OPERATOR 2
6	SALESREP 1

第 6.1.5.1 节

查询 1

- 假设你想知道行动信息（其 *ACTNO* 大于 100）以及各个项目行动的设计人姓名（从 **PROJECT** 表中查找）。请列出这些信息，并按照设计人的姓名以字母顺序对结果排序。

```
SELECT DISTINCT p.PROJNO, FIRSTNME, LASTNAME, p.ACSTDATE, ep.EMSTDATE
FROM EMPLOYEE e, EMPPROJECT ep, PROJECT p
WHERE e.EMPNO = ep.EMPNO
AND ep.PROJNO = p.PROJNO
AND e.JOB = 'DESIGNER'
AND p.ACTNO > 100
ORDER BY FIRSTNME, LASTNAME
```

Status	Result1	PROJNO	FIRSTNME	LASTNAME	ACSTDATE	EMSTDATE
1		MA2112	BRUCE	ADAMSON	2002-07-15	2002-01-01
2		MA2112	BRUCE	ADAMSON	2002-07-15	2002-07-15
3		MA2113	ELIZABETH	PIANKA	2002-10-01	2002-07-15
4		MA2112	JAMES	WALKER	2002-07-15	2002-01-01
5		MA2112	JAMES	WALKER	2002-07-15	2002-10-01
6		MA2113	MARILYN	SCOUTTEN	2002-10-01	2002-04-01
7		MA2112	MASATOSHI	YOSHIMURA	2002-07-15	2002-01-01
8		MA2113	MASATOSHI	YOSHIMURA	2002-10-01	2002-01-01
9		MA2112	MASATOSHI	YOSHIMURA	2002-07-15	2002-06-01
10		MA2113	WILLIAM	JONES	2002-10-01	2002-10-01

查询 2

- 把 **EMPLOYEE** 表和 **DEPARTMENT** 表连接起来，假设 **EMPLOYEE** 表中的 *WORKDEPT* 是 **FK**，它指向 *DEPTNO*，即 **DEPARTMENT** 表中的 **PK**。保存结果，然后再重复该查询，但分别使用 **LEFT OUTER JOIN**（左外连接）、**RIGHT OUTER JOIN**（右外连接）以及 **FULL OUTER JOIN**（全部外连接）。比较这些结果。

```
SELECT *
FROM EMPLOYEE e
      INNER
      | LEFT (OUTER)
      | RIGHT (OUTER)
```

```

        | FULL (OUTER)
    JOIN DEPARTMENT d
    ON e.WORKDEPT = d.DEPTNO

```

第 6.2.1.1 节

查询 1

1. 本公司刚设立了一个新部门，名为 *FOO*，部门代码为 *K47*，并以 *'E01'* 作为 ADMRDEPT。请把这条记录插入到 **DEPARTMENT** 表中。

```

INSERT INTO DEPARTMENT (DEPTNO, DEPTNAME, ADMRDEPT)
VALUES ('K47', 'FOO', 'E01')

```

Status		Result1				
	DEPTNO	DEPTNAME	MGRNO	ADMRDEPT	LOCATION	
1	K47	FOO	NULL	E01	NULL	

查询 2

2. 创建一个称为 **D11_PROJ** 的新表，它具有与 **PROJECT** 表相同的结构，然后把来自部门 *D11* 的、与所有项目相关的数据添加到其中。

```

CREATE TABLE D11_PROJ LIKE PROJECT

```

```

INSERT INTO D11_PROJ
SELECT *
FROM PROJECT
WHERE DEPTNO = 'D11'

```

Status		Result1							
	PROJNO	PROJNAME	DEPTNO	RESPEMP	PRSTAFF	PRSTDATE	PRENDATE	MAJPROJ	
1	MA2110	W L PROGRAMMING	D11	000060	9.00	2002-01-01	2003-02-01	MA2100	
2	MA2111	W L PROGRAM DESIGN	D11	000220	2.00	2002-01-01	1982-12-01	MA2110	
3	MA2112	W L ROBOT DESIGN	D11	000150	3.00	2002-01-01	1982-12-01	MA2110	
4	MA2113	W L PROD CONT PROGS	D11	000160	3.00	2002-02-15	1982-12-01	MA2110	

第 6.2.2.1 节

查询 1

- 试试把 Paul 的 EDLEVEL 更新为 *'NULL'*，看看会发生什么事情。

```
UPDATE EMPLOYEE
SET EDLEVEL = NULL
WHERE EMPNO = 300001
```

在 SQL 结果选项卡中，提示本次查询运行失败。右侧的“状态”选项卡称，这是因为我们企图把一个 NULL 值赋给 NOT NULL 列，这是非法的。

Status	Operation	Date	Connection Profile
Failed	Run SQL	5/26/10 2:11 AM	SAMPLE

```
Status
Run SQL

UPDATE EMPLOYEE
SET EDLEVEL = NULL
WHERE EMPNO = 300001
Assignment of a NULL value to a NOT NULL column "TBSPACEID=2, TABLEID=6, COLNO=8" is not allowed.. SQLC
```

查询 2

- 试试把 Paul 的 WORKDEPT 更新为 'Z11'，看看会发生什么事情。

```
UPDATE EMPLOYEE
SET WORKDEPT = 'Z11'
WHERE EMPNO = 300001
```

本查询也失败了，因为它企图更新一个外键 (FK) 列（在本例中即 WORKDEPT），但所采用的值在主键/父键列（即 DEPARTMENT 表中的 DEPTNO 列，这也是本 FK 所引用的内容）中并不存在。这也是非法的。

```
Status
Run SQL

UPDATE EMPLOYEE
SET WORKDEPT = 'Z11'
WHERE EMPNO = 300001
The insert or update value of the FOREIGN KEY "DB2INST1.EMPLOYEE.REDE" is not equal to any value of the parent key of the parent table.. SQLC
```

第 6.2.3.1 节

查询 1

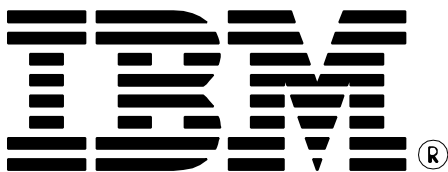
试试从 **DEPARTMENT** 表中删除部门 'E21'

```
DELETE FROM DEPARTMENT
WHERE DEPTNO = 'E21'
```


错误消息提示说，我们不能删除这些行，因为它们所包含的主键/父键列当前正在被某些其他列所引用。这种操作也是非法的。

```
Status 
Run SQL

DELETE FROM DEPARTMENT
WHERE DEPTNO = 'E21'
A parent row cannot be deleted because the relationship "DB2INST1.PROJECT.FK_PROJECT_1" restricts the deletion..
```



© Copyright IBM Corporation 2011
All Rights Reserved.

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

IBM, IBM (logo), and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both

UNIX is a registered trademark of The Open Group in the United States, other countries, or both

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this publication is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of April 2010, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.



IBM DB2® 9.7

数据一致性
亲自动手实验

目录

1. 数据一致性简介.....	4
2. 本实验的目标.....	4
3. 安装和启动 DB2.....	4
3.1 ENVIRONMENT SETUP REQUIREMENTS.....	4
3.2 登录到虚拟机.....	4
3.3 SAMPLE 数据库.....	5
3.4 创建并填充数据表.....	5
4. 具有“当前落实”的“游标稳定性”.....	6
4.1 “之前”场景：不具有“当前落实”.....	6
4.1.1 关闭“当前落实”.....	6
4.1.2 在终端 A 中执行一次写入查询.....	7
4.1.3 在终端 B 中执行一次读查询.....	9
4.1.4 释放锁.....	10
4.2 “之后”场景：具有“当前落实”.....	11
4.2.1 打开“当前落实”.....	12
4.2.2 在终端 A 中执行一次写查询.....	12
4.2.3 在终端 B 中执行读查询.....	12
5. 可重复读.....	14
5.1 “幻影读”场景：可重复读.....	15
5.1.1 在终端 A 中执行读查询.....	15
5.1.2 在终端 B 中执行写查询.....	15
5.1.3 释放锁.....	16
6. 读稳定性.....	18
6.1 “幻影读”场景：读稳定性.....	18
6.1.1 在终端 A 中执行读查询.....	18
6.1.2 在终端 B 中执行写查询.....	19
6.1.3 在终端 A 中执行另一次读查询.....	20
7. 未落实的读.....	21
7.1 “未落实的读”场景：游标稳定性.....	22
7.1.1 在终端 A 中执行一次更新查询.....	22

7.1.2	在终端 B 中执行读查询.....	22
7.1.3	释放锁.....	23
7.2	“未落实的读”场景：“未落实的读”	25
7.2.1	在终端 A 中执行一次更新查询.....	25
7.2.2	在终端 B 中执行读查询.....	25

1. 数据一致性简介

在本实验中，我们将讨论数据一致性以及 DB2 中的一致性控制。

2. 本实验的目标

本实验结束时，学员们将能够：

- 理解游标稳定性 (Cursor Stability) 和当前落实 (Currently Committed) 之间的差异。
- 理解可重复读 (Repeatable Read)、读稳定性 (Read Stability)、游标稳定性 (Cursor Stability) 和未落实的读 (Uncommitted Read) 之间的差异。
- 能够利用 CLP 在运行时为数据库指定不同的隔离等级。

3. 安装和启动 DB2

3.1 环境设置要求

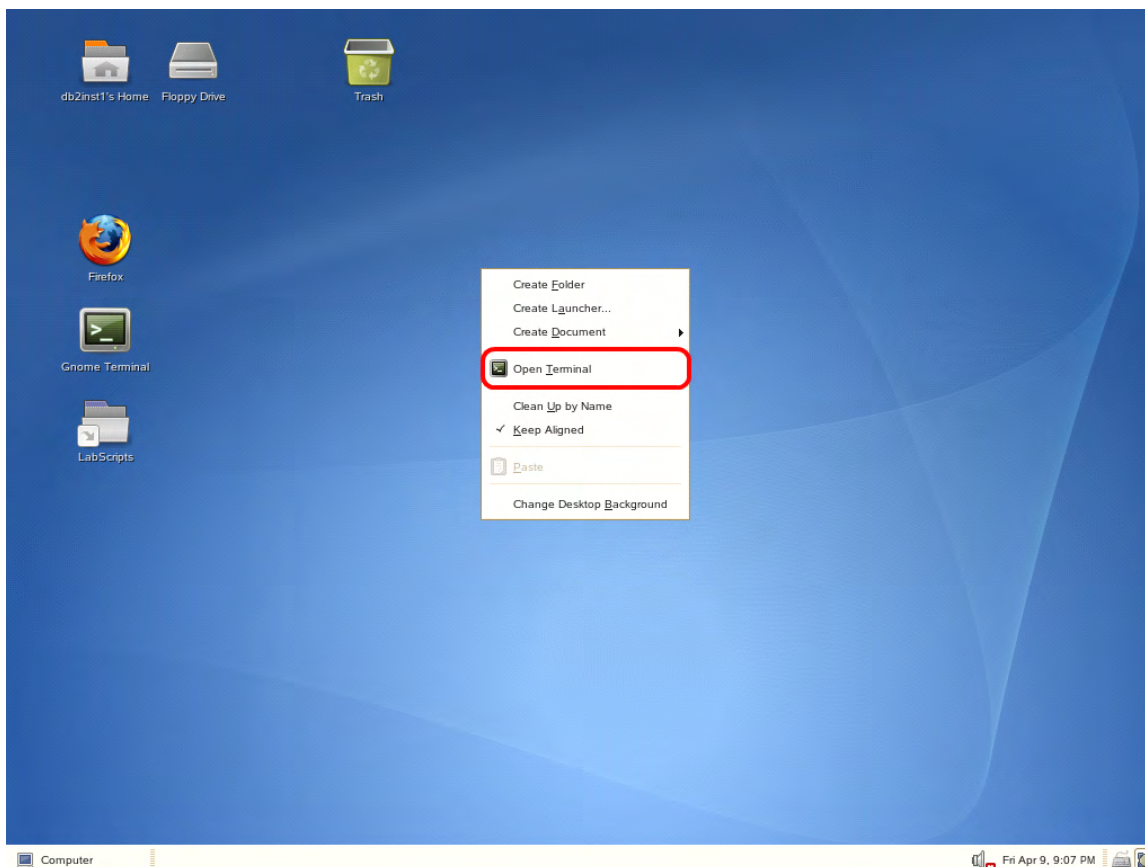
要完成本实验，你需要以下软件：

- DB2 Academic Workshop VMware® 影像（DB2 学术研讨班 VMware® 影像）
- VMware Player 2.x 或 VMware Workstation 5.x 或者更高版本

关于如何获得这些组件的帮助信息，请遵循“**VMware 基础和入门**”模块中给出的指示。

3.2 登录到虚拟机

1. 利用下述信息登录到 VMware 虚拟机：
用户名：**db2inst1**
密码：**password**
2. 在**桌面**上右击鼠标，选择“**打开终端**”项，即可打开终端窗口。



3. 在终端窗口中输入“**db2start**”，启动 DB2 服务器。

```
db2start
```

3.3 SAMPLE 数据库

为执行本实验，你需要以其原始格式创建的 DB2 示例数据库。

执行下面的命令，删除（如果已经存在的话）并重建 **SAMPLE** 数据库：

```
db2 force applications all
db2 drop db sample
db2saml
```

3.4 创建并填充数据表

我们将创建一个简单的数据表，在该实验会话过程中对其进行更新。名为“**tb1**”的表仅含有一列，名为“**column1**”。然后我们为其填充 9 行，内容均为“**10**”。

1. 运行以下命令。

```
db2 connect to SAMPLE
db2 "create table TB1 (COLUMN1 integer)"
db2 "insert into TB1 (select 10 from syscat.tables fetch first 9 rows
only)"
db2 terminate
```

4. 具有“当前落实”的“游标稳定性”

我们现在介绍“当前落实”功能的效果。为此，我们将模拟一种场景，当 2 个查询并发运行时，该场景可能出现潜在的读/写封锁。然后，我们比较切换参数 `cur_commit` 后在结果及执行时间方面出现的差异。

我们将使用 DB2 的命令行处理器 (CLP) 来模拟同时访问数据库的应用。

4.1 “之前”场景：不具有“当前落实”

4.1.1 关闭“当前落实”

1. 首先，我们将检查对“当前落实”的现有设置。在终端中输入以下命令。由于我们将使用多个终端，所以我们将此终端称为终端 A。

```
db2 get db cfg for sample
```



```
db2inst1@dmsrvr:~> db2 get db cfg for sample

Database Configuration for Database sample

Database configuration release level      = 0x0d00
Database release level                  = 0x0d00

Database territory                       = US
Database code page                       = 1208
```

`cur_commit` 参数靠近该列表的底部。此时，它应当显示为 ON（打开），因为在 DB2 9.7 中，这是新数据库的默认值。


```

db2inst1@dmsrvr:~/labs
File Edit View Terminal Tabs Help
Auto-Revalidation (AUTO REVAL) = DEFERRED
Currently Committed (CUR_COMMIT) = ON
CHAR output with DECIMAL input (DEC_TO_CHAR_FMT) = NEW
Enable XML Character operations (ENABLE_XMLCHAR) = YES
WLM Collection Interval (minutes) (WLM_COLLECT_INT) = 0

Monitor Collect Settings
Request metrics (MON_REQ_METRICS) = BASE
Activity metrics (MON_ACT_METRICS) = BASE
Object metrics (MON_OBJ_METRICS) = BASE
Unit of work events (MON_UOW_DATA) = NONE
Lock timeout events (MON_LOCKTIMEOUT) = NONE
Deadlock events (MON_DEADLOCK) = WITHOUT_HIST
Lock wait events (MON_LOCKWAIT) = NONE
Lock wait event threshold (MON_LW_THRESH) = 5000000

SMTP Server (SMTP_SERVER) =

db2inst1@dmsrvr:~/labs>

```

2. 下一步是禁用“当前落实”语义。对此，利用下面的命令，把 `cur_commit` 的值修改为 `DISABLED`（禁用）：

```
db2 update db cfg for sample using cur_commit disabled
```

```

db2inst1@db2rules:~
File Edit View Terminal Tabs Help
db2inst1@db2rules:~> db2 update db cfg for sample using cur_commit disabled
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
db2inst1@db2rules:~>

```

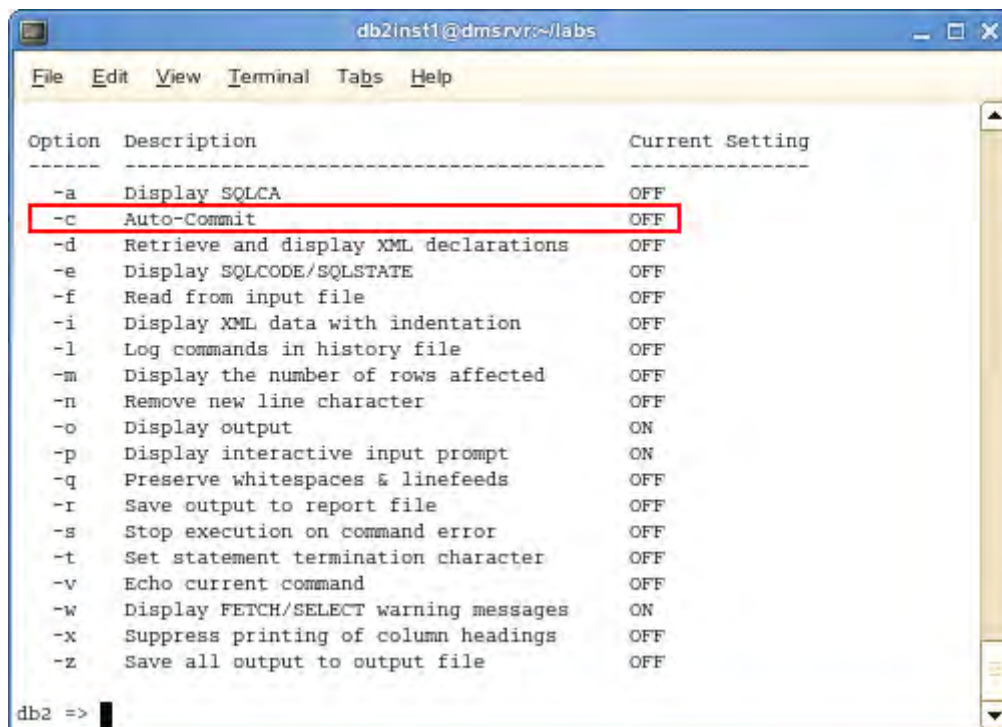
4.1.2 在终端 A 中执行一次写入查询

1. 为了模拟长时间运行的事务的行为，我们首先需要禁用“自动落实” (`auto-commit`) 功能，该功能在 `CLP` 中默认为 `ON`（打开）。当“自动落实”处于激活状态时，`CLP` 会在每一条 `SQL` 语句执行完毕后自动发出一条 `COMMIT`（落实）命令。因此，我们需要禁用它，这样我们就能够指定在什么时候落实该事务。输入下面的命令，进入 `CLP` 提示符。“+c”选项将禁用本会话的“自动落实”功能。

```
db2 +c
```

2. 执行下面的命令，可以检查“自动落实”功能是否已经关闭。由于“自动落实”功能处于 `OFF`（关闭）状态，从现在起，你执行的所有 `SQL` 语句都将成为同一个事务的一部分，直到你发出“`commit`”（落实）或“`rollback`”（回滚）命令为止。

```
list command options
```



```

db2inst1@dmsrvr:~/labs
File Edit View Terminal Tabs Help

Option  Description                               Current Setting
-----  -
-a      Display SQLCA                                 OFF
-c      Auto-Commit                                  OFF
-d      Retrieve and display XML declarations         OFF
-e      Display SQLCODE/SQLSTATE                     OFF
-f      Read from input file                         OFF
-i      Display XML data with indentation            OFF
-l      Log commands in history file                 OFF
-m      Display the number of rows affected          OFF
-n      Remove new line character                    OFF
-o      Display output                               ON
-p      Display interactive input prompt             ON
-q      Preserve whitespaces & linefeeds            OFF
-r      Save output to report file                   OFF
-s      Stop execution on command error              OFF
-t      Set statement termination character           OFF
-v      Echo current command                         OFF
-w      Display FETCH/SELECT warning messages       ON
-x      Suppress printing of column headings        OFF
-z      Save all output to output file               OFF

db2 =>

```

3. 接入数据库“sample”。

connect to sample



```

db2inst1@db2rules:~
File Edit View Terminal Tabs Help

db2 => connect to sample

Database Connection Information

Database server          = DB2/LINUX 9.7.1
SQL authorization ID    = DB2INST1
Local database alias    = SAMPLE

db2 =>

```

4. 在对该表进行任何更新之前，我们迅速进行一次查询，以观察“column1”列的当前值。

select * from tbl

```

db2inst1@db2rules:~
File Edit View Terminal Tabs Help
db2 => select * from tb1

COLUMN1
-----
          10
          10
          10
          10
          10
          10
          10
          10
          10
          10

  9 record(s) selected.

db2 =>

```

5. 然后再执行一次更新查询，它将在该事务被落实之前一直对这些行加锁。我们执行一次简单的更新查询，它将把所有的值修改为 20。

```
update tb1 set column1 = 20
```

```

db2 => update tb1 set column1 = 20
DB20000I The SQL command completed successfully.
db2 =>

```

4.1.3 在终端 B 中执行一次读查询

1. 我们将打开另一个终端窗口，并把它用作企图访问该数据表的第二个应用。在桌面上右击鼠标，选择“打开终端”项，即可打开一个终端窗口。这个新终端被成为终端 B。
2. 与第一个终端相似的是，我们将以用户名“db2inst1”以及密码“password”接入数据库“sample”。输入以下命令：

```
db2 connect to sample
```

3. 下一步，我们发起一次查询，由它读取被终端 A 锁定的数据。

```
time db2 "select * from tb1"
```

time（时间）命令能够让我们量化等待时间。我们可以看到，该查询进入等待状态，无法返回任何结果。实际上，它被终端 A 的查询封锁了。

```

db2inst1@db2rules:~
File Edit View Terminal Tabs Help
COLUMN1
-----
          10
          10
          10
          10
          10
          10
          10
          10
          10
          10
          10

  9 record(s) selected.

db2 => update tb1 set column1 = 20
DB20000I  The SQL command completed successfully.
db2 =>

db2inst1@db2rules:~
File Edit View Terminal Tabs Help
db2inst1@db2rules:~> db2 connect to sampe
SQL1013N  The database alias name or database name "SAMPE" could not be found.
SQLSTATE=42705
db2inst1@db2rules:~> db2 connect to sample

  Database Connection Information

Database server      = DB2/LINUX 9.7.1
SQL authorization ID = DB2INST1
Local database alias = SAMPLE

db2inst1@db2rules:~> time db2 "select * from tb1"

```

4.1.4 释放锁

1. 把 2 个终端并排打开，我们可以观察到在终端 A 中落实查询之后产生的效果。在终端 A 中，通过以下命令落实该事务：

```
commit
```

Terminal A

```

db2inst1@db2rules:~
File Edit View Terminal Tabs Help
10
10
10
10
10
10
10
10
10
10
9 record(s) selected.
db2 => update tb1 set column1 = 20
DB20000I The SQL command completed successfully.
db2 => commit
DB20000I The SQL command completed successfully.
db2 =>

```

Terminal B

```

db2inst1@db2rules:~
File Edit View Terminal Tabs Help
SQL authorization ID = DB2INST1
Local database alias = SAMPLE
db2inst1@db2rules:~> time db2 "select * from tb1"
COLUMN1
-----
20
20
20
20
20
20
20
20
20
9 record(s) selected.
real    2m3.613s
user    0m0.008s
sys     0m0.192s
db2inst1@db2rules:~>

```

可以看到终端 B 的查询立即返回了更新后的值。终端 A 的封锁被解除了，终端 B 上的事务可以继续进行并能够访问这些值了。

4.2 “之后”场景：具有“当前落实”

我们将重复上面的过程，但这一次将打开“当前落实”功能。目的是查看第二次查询返回结果所花费时间以及实际返回的值的差异。

4.2.1 打开“当前落实”

1. 在终端 A 中，使用下面的命令打开“当前落实”：

```
update db cfg for sample using cur_commit on
```

```
db2 => update db cfg for sample using cur_commit on
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
SQL1363W One or more of the parameters submitted for immediate modification
were not changed dynamically. For these configuration parameters, all
applications must disconnect from this database before the changes become
effective.
db2 =>
```

2. 改变该值之后，我们需要断开数据库连接，以便让这个新值生效。在终端 A 中执行：

```
connect reset
```

3. 在终端 B 在执行：

```
db2 connect reset
```

4.2.2 在终端 A 中执行一次写查询

1. 与前面一节相似，我们将把该表中的值从 20 更改为 30。

```
connect to sample
update tb1 set column1 = 30
```

你应该看到，该查询已经成功完成。



```
db2inst1@db2rules:~
File Edit View Terminal Tabs Help
db2 => connect to sample

Database Connection Information

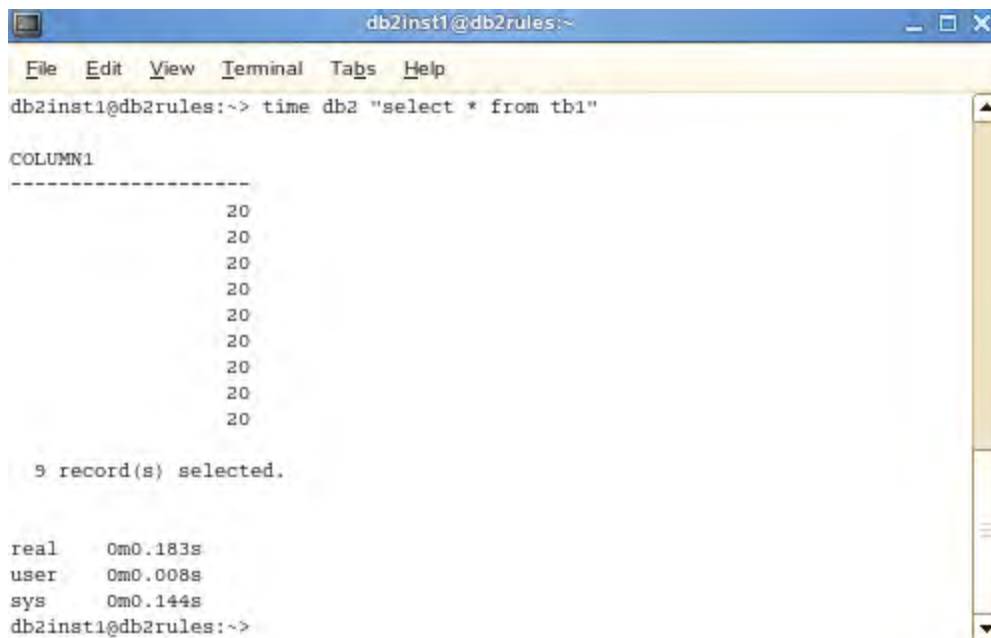
Database server          = DB2/LINUX 9.7.1
SQL authorization ID    = DB2INST1
Local database alias    = SAMPLE

db2 => update tb1 set column1 = 30
DB20000I The SQL command completed successfully.
db2 =>
```

4.2.3 在终端 B 中执行读查询

1. 在终端 B 中，重新连接数据库，并从表 tb1 中检索数据。

```
db2 connect to sample
time db2 "select * from tb1"
```



```
db2inst1@db2rules:~  
File Edit View Terminal Tabs Help  
db2inst1@db2rules:~> time db2 "select * from tb1"  
  
COLUMN1  
-----  
20  
20  
20  
20  
20  
20  
20  
20  
20  
20  
  
9 record(s) selected.  
  
real    0m0.183s  
user    0m0.008s  
sys     0m0.144s  
db2inst1@db2rules:~>
```

请注意这次查询返回结果所花费的时间。这次查询立即返回结果了，因为对该数据不存在访问封锁。同样，还要注意，所返回的值并不是来自最新的更新，因为我们还没有落实它。

2. 在终端 A 中，输入以下命令，以落实该更新。

```
commit
```

3. 把焦点切换回终端 B。我们再次执行 **select** 查询，方法再次按下向上箭头按钮以检索最后一次执行的命令，然后按 **Enter**（回车）。如果你无法找到该最后的命令，则可以键入。

```
time db2 "select * from tb1"
```

请注意这次返回的值反映了我们最近的更新，因为终端 A 中的事务已经终止，这些更新已经被落实到数据库中。

```

db2inst1@db2rules:~
File Edit View Terminal Tabs Help
db2inst1@db2rules:~> time db2 "select * from tb1"

COLUMN1
-----
30
30
30
30
30
30
30
30
30
30

9 record(s) selected.

real    0m0.188s
user    0m0.008s
sys     0m0.144s
db2inst1@db2rules:~>

```

4. 终止终端 A 中的数据库连接:

```
connect reset
```

5. 然后, 终止终端 B 中的数据库连接:

```
db2 connect reset
```

5. 可重复读

我们上面介绍了“游标稳定性”及“当前落实”功能的效果, 下面来研究“可重复读”。为此, 我们将模拟一个场景来展示“可重复读”如何把每个事务隔离开, 从而防止幻影读并发性问题的出现。

应用 A 将执行一条查询, 它将根据一些搜索标准读取若干行。应用 B 将视图插入一些能够满足应用 A 的查询条件的新数据。

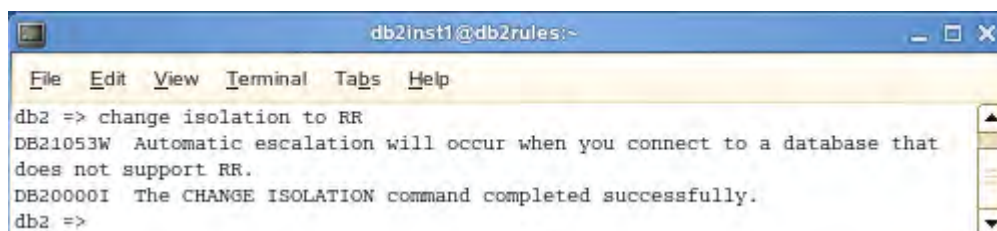
我们将利用 DB2 的命令行处理器 (CLP) 来模拟同时访问该数据库的应用。

5.1 “幻影读”场景：可重复读

5.1.1 在终端 A 中执行读查询

1. 我们需要把终端 A 当前 CLP 会话的隔离级别修改为“可重复读”。这必须在与数据库建立连接之前完成。

```
change isolation to RR
```



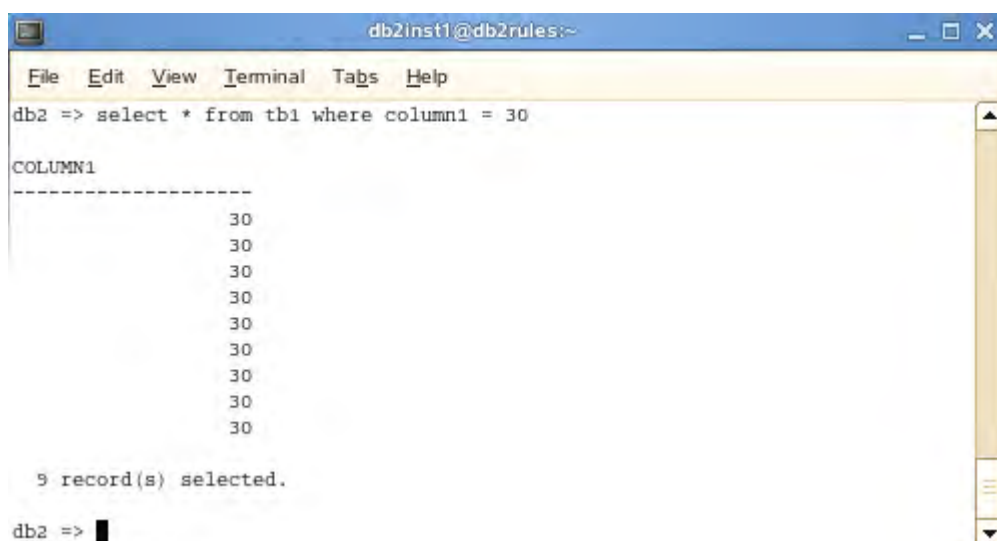
```
db2inst1@db2rules:~  
File Edit View Terminal Tabs Help  
db2 => change isolation to RR  
DB21053W Automatic escalation will occur when you connect to a database that  
does not support RR.  
DB20000I The CHANGE ISOLATION command completed successfully.  
db2 =>
```

2. 连接至数据库“sample”。

```
connect to sample
```

3. 现在我们快速进行一次查询，以便根据某些标准观察“column1”的当前值。

```
select * from tb1 where column1 = 30
```



```
db2inst1@db2rules:~  
File Edit View Terminal Tabs Help  
db2 => select * from tb1 where column1 = 30  
  
COLUMN1  
-----  
30  
30  
30  
30  
30  
30  
30  
30  
30  
30  
  
9 record(s) selected.  
db2 =>
```

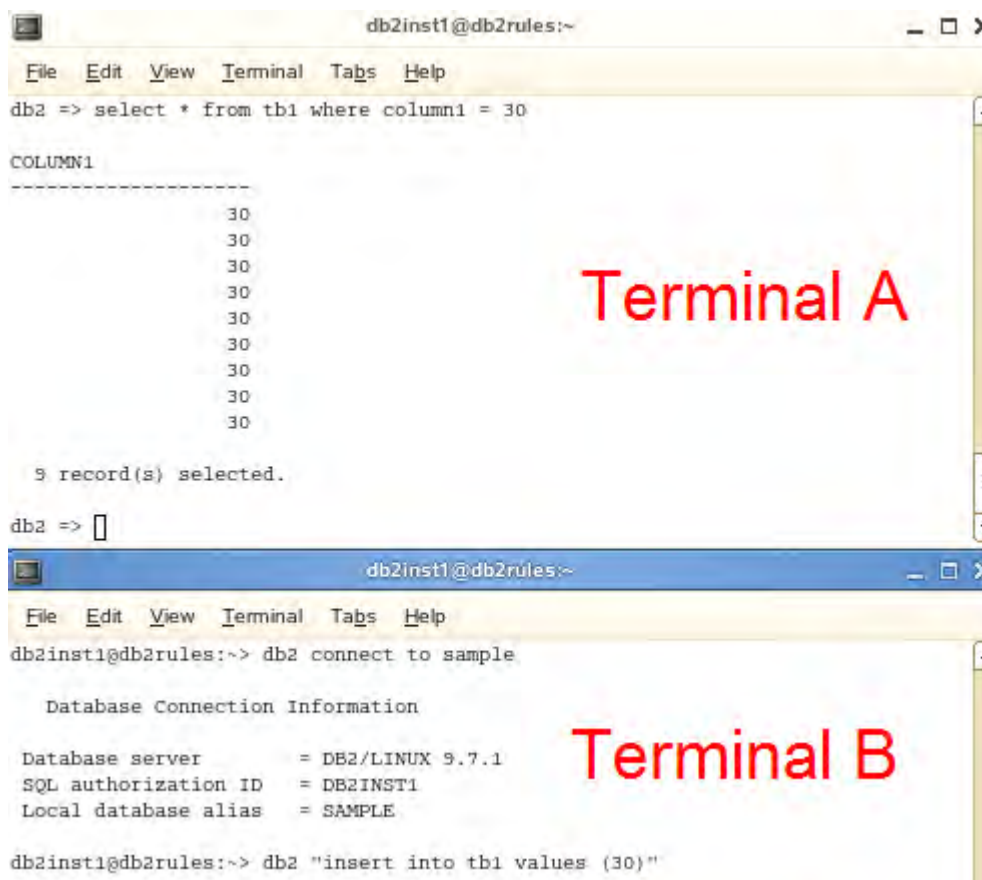
5.1.2 在终端 B 中执行写查询

1. 我们将启动一次查询，它企图向 tb1 中插入数据，而 tb1 被终端 A 锁定了。

```
db2 connect to sample
```

```
db2 "insert into tb1 values (30)"
```

我们可以看到，该操作进入等待状态，无法返回任何结果。实际上，它被终端 A 的查询封锁了。



```
db2inst1@db2rules:~  
File Edit View Terminal Tabs Help  
db2 => select * from tb1 where column1 = 30  
  
COLUMN1  
-----  
          30  
          30  
          30  
          30  
          30  
          30  
          30  
          30  
          30  
          30  
  
9 record(s) selected.  
  
db2 =>   
  
db2inst1@db2rules:~> db2 connect to sample  
  
Database Connection Information  
  
Database server          = DB2/LINUX 9.7.1  
SQL authorization ID    = DB2INST1  
Local database alias    = SAMPLE  
  
db2inst1@db2rules:~> db2 "insert into tb1 values (30)"
```

5.1.3 释放锁

1. 把 2 个终端并排打开，我们可以观察到在终端 A 中落实查询之后产生的效果。在终端 A 中，通过执行以下命令落实该事务：

```
commit
```

```

db2inst1@db2rules:~
File Edit View Terminal Tabs Help
COLUMN1
-----
          30
          30
          30
          30
          30
          30
          30
          30
          30
          30
          30
          30

  9 record(s) selected.

db2 => commit
DB20000I The SQL command completed successfully.
db2 =>

db2inst1@db2rules:~
File Edit View Terminal Tabs Help
db2inst1@db2rules:~> db2 connect to sample

Database Connection Information

Database server          = DB2/LINUX 9.7.1
SQL authorization ID    = DB2INST1
Local database alias    = SAMPLE

db2inst1@db2rules:~> db2 "insert into tb1 values (30)"
DB20000I The SQL command completed successfully.
db2inst1@db2rules:~> ||

```

可以看到终端 B 的查询立即完成了。终端 A 的封锁被解除了，终端 B 上的事务可以继续插入新的值了。

在这里我们看到，在“可重复读”隔离级别上，不会发生“幻影读”的情况，因为应用所读取的行被锁定了，无法被其他事务更新。

如果我们以“读稳定性”隔离级别来进行同样的工作，会出现什么情况呢？

2. 在终端 A 中终止数据库连接：

```
connect reset
```

3. 然后，终止终端 B 中的数据库连接：

```
db2 connect reset
```

6. 读稳定性

我们前面已经发现，在“可重复读”隔离级别上，不会发生幻影读。但是，如果采用“读稳定性”隔离级别，则有可能发生。我们将模拟一个场景来展示“读稳定性”与“可重复读”在隔离事务方面有哪些区别。

应用 A 将执行一条查询，它将根据一些搜索标准读取若干行。应用 B 将视图插入一些能够满足应用 A 的查询条件的新数据。

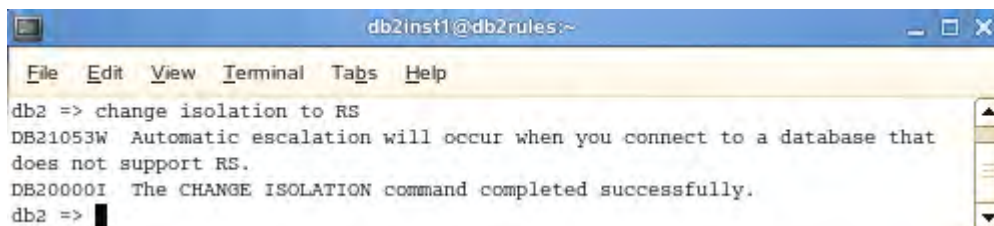
我们将利用 DB2 的命令行处理器 (CLP) 来模拟同时访问该数据库的应用。

6.1 “幻影读”场景：读稳定性

6.1.1 在终端 A 中执行读查询

1. 我们需要把终端 A 当前 CLP 会话的隔离级别修改为“读稳定性”。这必须在与数据库建立连接之前完成。

```
change isolation to RS
```



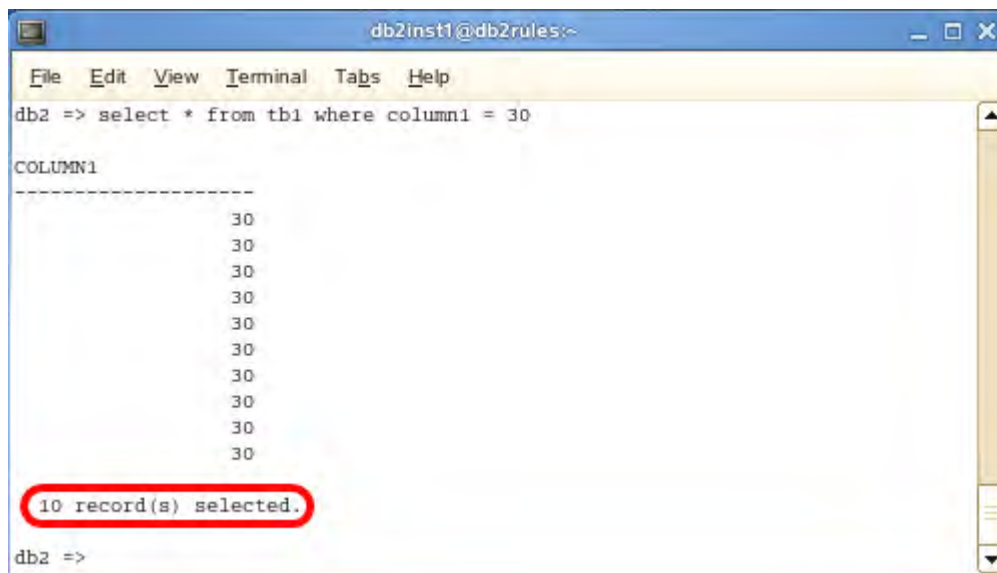
```
db2inst1@db2rules:~  
File Edit View Terminal Tabs Help  
db2 => change isolation to RS  
DB21053W Automatic escalation will occur when you connect to a database that  
does not support RS.  
DB20000I The CHANGE ISOLATION command completed successfully.  
db2 => █
```

2. 连接至数据库“sample”。

```
connect to sample
```

3. 现在我们快速进行一次查询，以便根据某些标准观察“column1”的当前值。

```
select * from tb1 where column1 = 30
```



A terminal window titled "db2inst1@db2rules:~" with a menu bar (File, Edit, View, Terminal, Tabs, Help). The prompt "db2 =>" is followed by the SQL query "select * from tb1 where column1 = 30". The output shows a column header "COLUMN1" followed by ten rows of the value "30". At the bottom, a red oval highlights the message "10 record(s) selected." followed by the prompt "db2 =>".

被查询到的记录目前是 10 个。

6.1.2 在终端 B 中执行写查询

1. 终端 B 将插入一些数据，它们符合终端 A 的查询标准。

```
db2 connect to sample
```

```
db2 "insert into tb1 values (30)"
```

我们可以看到，该查询没有等待终端 A 来落实，而是把数据插入到 tb1 了。

```
db2inst1@db2rules:~  
File Edit View Terminal Tabs Help  
db2 => select * from tb1 where column1 = 30  
  
COLUMN1  
-----  
30  
30  
30  
30  
30  
30  
30  
30  
30  
30  
30  
30  
  
10 record(s) selected.  
  
db2 =>   
  
db2inst1@db2rules:~  
File Edit View Terminal Tabs Help  
db2inst1@db2rules:~> db2 connect to sample  
  
Database Connection Information  
  
Database server          = DB2/LINUX 9.7.1  
SQL authorization ID    = DB2INST1  
Local database alias    = SAMPLE  
  
db2inst1@db2rules:~> db2 "insert into tb1 values (30)"  
DB20000I The SQL command completed successfully.  
db2inst1@db2rules:~>
```

6.1.3 在终端 A 中执行另一次读查询

1. 现在我们将快速进行一次查询，以便在落实之前观察“column1”列的当前值。

```
select * from tb1 where column1 = 30
```

```

db2inst1@db2rules:~
File Edit View Terminal Tabs Help
db2 => select * from tb1 where column1 = 30

COLUMN1
-----
          30
          30
          30
          30
          30
          30
          30
          30
          30
          30
          30
          30

11 record(s) selected.
db2 =>

```

请注意，该查询现在返回 11 行数据，而不是 10 行。多出来另外一行，虽然我们是在同一个交易中执行相同的 SQL 查询。这是因为，“读稳定性”隔离级别不会阻止幻影行的出现。

2. 在终端 A 中，输入以下命令，落实该更新：

```
commit
```

3. 终止终端 A 中的数据库连接：

```
connect reset
```

4. 然后，终止终端 B 中的数据库连接：

```
db2 connect reset
```

7. 未落实的读

既然我们知道了“可重复读”与“读稳定性”之间的差别所在，我们就能够理解最低层次的隔离如何返回作用。在使用“只读”表或者“仅限查询”语句时，“未落实的读”隔离层次会非常有用。当使用“未落实的读”时，将读取来自其他事务的未落实数据。

应用 A 将执行一次利用 RR（可重复读）更新了某一行的查询。应用 B 将尝试利用 CS（游标稳定性）和 UR（未落实的读）来读取该相同的行。

7.1 “未落实的读”场景：游标稳定性

7.1.1 在终端 A 中执行一次更新查询

1. 我们需要把终端 A 当前 CLP 会话的隔离级别修改为“可重复读”。这必须在与数据库建立连接之前完成。

```
change isolation to RR
```

2. 连接至数据库“sample”。

```
connect to sample
```

3. 现在我们快速进行一次查询，以便更新“column1”列的当前值。

```
update tb1 set column1 = 40
```



```
db2inst1@db2rules:~
File Edit View Terminal Tabs Help
db2 => change isolation to RR
DB21053W Automatic escalation will occur when you connect to a database that
does not support RR.
DB20000I The CHANGE ISOLATION command completed successfully.
db2 => connect to sample

Database Connection Information

Database server          = DB2/LINUX 9.7.1
SQL authorization ID    = DB2INST1
Local database alias    = SAMPLE

db2 => update tb1 set column1 = 40
DB20000I The SQL command completed successfully.
db2 =>
```

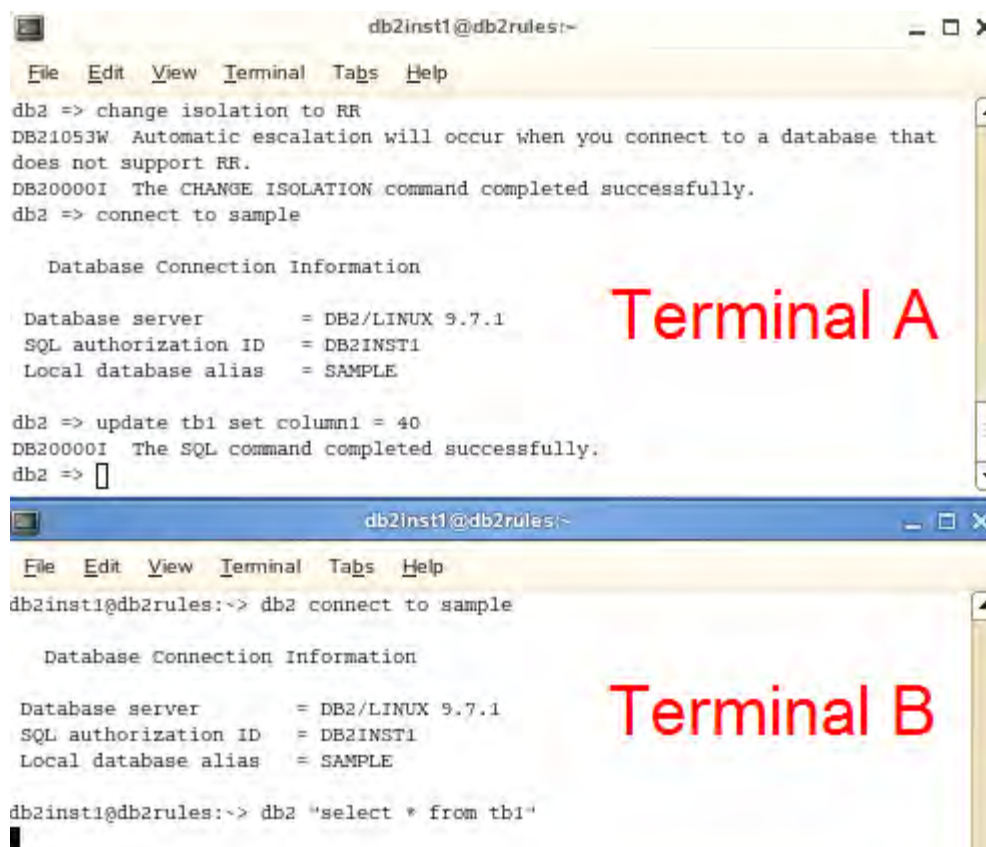
7.1.2 在终端 B 中执行读查询

1. 终端 B 将尝试利用 CS（游标稳定性）读取被终端 A 锁定的数据。

```
db2 connect to sample
```

```
db2 "select * from tb1"
```

我们可以看到，该查询语句在读取数据之前一直在等待终端 A 落实。



```
db2inst1@db2rules:~  
File Edit View Terminal Tabs Help  
db2 => change isolation to RR  
DB21053W Automatic escalation will occur when you connect to a database that  
does not support RR.  
DB20000I The CHANGE ISOLATION command completed successfully.  
db2 => connect to sample  
  
Database Connection Information  
  
Database server          = DB2/LINUX 9.7.1  
SQL authorization ID    = DB2INST1  
Local database alias    = SAMPLE  
  
db2 => update tb1 set column1 = 40  
DB20000I The SQL command completed successfully.  
db2 =>   
  
db2inst1@db2rules:~  
File Edit View Terminal Tabs Help  
db2inst1@db2rules:~> db2 connect to sample  
  
Database Connection Information  
  
Database server          = DB2/LINUX 9.7.1  
SQL authorization ID    = DB2INST1  
Local database alias    = SAMPLE  
  
db2inst1@db2rules:~> db2 "select * from tb1"
```

7.1.3 释放锁

1. 把 2 个终端并排打开，我们可以观察到在终端 A 中落实查询之后产生的效果。在终端 A 中，通过执行以下命令落实该事务：

```
commit
```


7.2 “未落实的读”场景：“未落实的读”

7.2.1 在终端 A 中执行一次更新查询

1. 我们快速进行一次查询，以便更新“column1”列的当前值。

```
update tb1 set column1 = 50
```

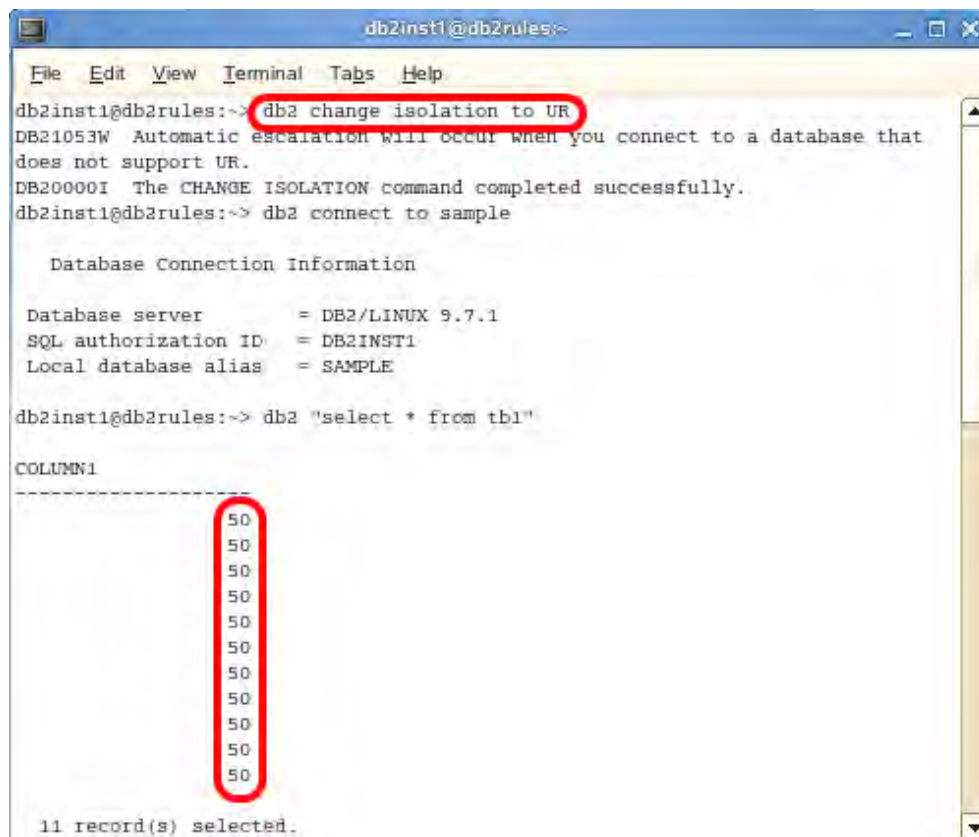
7.2.2 在终端 B 中执行读查询

1. 终端 B 将尝试利用 UR（未落实的读）读取被终端 A 锁定的数据。

```
db2 change isolation to UR  
db2 connect to sample  
db2 "select * from tb1"
```

我们可以看到，在“未落实的读”隔离级别上，该查询语句在读取数据之前并不需要等待终端 A 落实。相反，所返回的值就来自于终端 A 中未落实的事务。

如果终端 A 中的事务执行了一次回滚，终端 B 中列出的数据就不会反映 TB1 中的实际数据。这种现象被称为“脏读取”。



```
db2inst1@db2rules:~> db2 change isolation to UR
DB21053W  Automatic escalation will occur when you connect to a database that
does not support UR.
DB20000I  The CHANGE ISOLATION command completed successfully.
db2inst1@db2rules:~> db2 connect to sample

Database Connection Information

Database server      = DB2/LINUX 9.7.1
SQL authorization ID = DB2INST1
Local database alias = SAMPLE

db2inst1@db2rules:~> db2 "select * from tb1"

COLUMN1
-----
50
50
50
50
50
50
50
50
50
50
50
50
50

11 record(s) selected.
```

2. 在终端 A 中，通过执行以下命令落实该更新：

```
commit
```

3. 终止终端 A 中的数据库连接：

```
connect reset
```

4. 然后，终止终端 B 中的数据库连接：

```
db2 connect reset
```



© Copyright IBM Corporation 2011
All Rights Reserved.

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

IBM, IBM (logo), and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States, other countries, or both.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this publication is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of July 2009, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.



IBM DB2® 9.7

DB2 安全性
亲自动手实验

信息管理云计算能力中心

IBM（加拿大）研究院

目录

目录.....	2
1. 前言.....	3
2. 推荐读物.....	3
3. 基本设置.....	3
3.1 环境设置要求.....	3
3.2 准备步骤.....	4
4. 认证.....	4
4.1 在什么地方进行认证?	5
4.2 在服务器上规定认证类型.....	6
4.3 在客户端上规定认证类型.....	7
4.4 利用 Data Studio 管理认证参数.....	7
5. 授权.....	10
5.1 权限.....	10
5.1.1 实例级权限.....	10
5.1.2 数据库级权限.....	11
5.2 特权.....	11
5.3 练习 – 授予及撤销权限和特权.....	12
5.4 细粒化特权--视图.....	14
6. 角色.....	15
6.1 示例--角色.....	16

1. 前言

在本实验中，你将学习如何控制对实例的访问，如何控制数据库本身的访问，以及如何控制对数据库中的数据和数据对象的访问。

本实验结束时，你将能够：

- ▶ 对用户授予和撤销权限
- ▶ 对用户授予和撤销特权
- ▶ 创建角色
- ▶ 对用户授予和撤销角色

2. 推荐读物

Getting started with DB2 Express-C eBook (DB2 Express-C 入门电子书) (第 10 章)

<https://www.ibm.com/developerworks/wikis/display/DB2/FREE+Book+Getting+Started+with+DB2+Express-C>

这是一本免费的电子书，能够让你快速了解 DB2。

Understanding DB2 9 Security (理解 DB2 9 的安全性)

编著人：Rebecca Bond (作者)、Kevin Yeung-Kuen See (作者)、Carmen Ka Man Wong (作者)、Yuk-Kuen Henry Chan (作者)

3. 基本设置

3.1 环境设置要求

要完成本实验，你需要以下软件：

- DB2 9.7 Academic Workshop VMware® 影像 (DB2 学术研讨班 VMware® 影像)
- VMware Player 2.x 或 VMware Workstation 5.x 或者更高版本

关于如何获得这些组件的帮助信息，请遵循模块 1 中“**VMware 基础和入门**”给出的指示。

3.2 准备步骤

1. 启动 VMware® 影像。完成加载之后，当提示登录证书时，利用用户名 “db2inst1” 来提供 DBADM 权限：

用户名：**db2inst1**

密码：**password**

2. 在桌面上右击鼠标，选择“打开终端”项，即可打开终端窗口。

3. 输入以下命令，启动 DB2 Database Manager:

```
db2start
```

注：如果 database manager 已经处于活动状态，则忽略警告信息。

为执行本实验，你需要以其原始格式创建的 DB2 示例数据库。

执行下面的命令，删除（如果已经存在的话）并重建 **SAMPLE** 数据库：

```
db2 force applications all
db2 drop db sample
db2sampl
```

4. 认证

当你首次试图访问一个实例或数据库时，认证系统将会设法确定你是不是你所声称的那个人。DB2 认证与基础性操作系统的认证机制密切配合来验证你的用户 ID 和密码。DB2 也可以使用第三方认证设施（例如 Kerberos）来认证用户。

当使用 DB2 之外的外部认证系统时，DB2 不再需要存储一套冗余的密码和敏感证书。这可以把安全性缺陷及黑客攻击的机会降低到最低程度。

4.1 在什么地方进行认证？

认证类型决定了在什么地方以及如何认证。这是由服务器上的数据库管理器配置文件中的认证参数规定的，这样就可以使一个实例中的所有数据库都具有相同的认证类型。在客户端，当对远程数据库进行编目时确定认证类型。

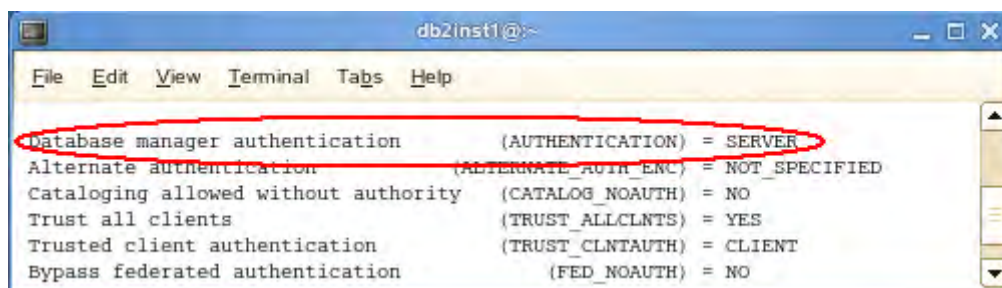
认证类型	描述
SERVER	所有的认证都在服务器上进行。当你接入数据库的时候，你必须提供你的用户 ID 和密码。随后根据服务器操作系统中的证书对这些信息进行验证。
SERVER_ENCRYPT	这与 SERVER 认证类型相似的是，认证都是在服务器进行，但密码在发送至服务器进行验证之前由 DB2 在客户端进行加密。
CLIENT	认证在客户端的操作系统上进行。
KERBEROS	认证在服务器上进行，并由 Kerberos 安全软件处理。 KERBEROS 认证类型只有当 DB2 服务器及客户端操作系统都支持 Kerberos 时才可以实现。Kerberos 安全性协议使用传统的密码来创建共享密钥，该密钥即成为用于验证用户身份的证书。这样就不必在网络中传递用户 ID 及密码了。
KRB_SERVER_ENCRYPT	本认证类型与 KEBREOS 相同，只是当客户端不支持 Kerberos 安全性系统时，它将采用 SERVER_ENCRYPT。如果这些选项都不可用，客户端将会收到一条连接错误消息，而且无法建立连接。
DATA_ENCRYPT	认证在服务器上进行，其行为类似于 SERVER_ENCRYPT。在这类认证中，不但对密码进行加密，而且所有的用户数据在客户端与服务器之间传输时也都加密。
DATA_ENCRYPT_CMP	此类认证与 DATA_ENCRYPT 是一样的。但该设置提供了对那些不支持 DATA_ENCRYPT 认证的客户端的兼容性，此时将采用 SERVER_ENCRYPT 方式进行连接，用户数据在这种情况下将不会被加密。
GSSPLUGIN	这种认证利用外部 GSS-API 插件在服务器上进行。如果客户端的认证类型未加以规定，服务器将向客户端发送一份服务器所支持的插件的列表。这些插件列于 <code>srvcon_gssplugin_list</code> 数据库管理器配置参数中。客户端然后从该列表中选择在客户端插件目录中找到的第一个插件。如果客户端不支持列表中的任何插件，则利用 KERBEROS 认证方法对客户端进行验证。
GSS_SERVER_ENCRYPT	利用 GSSPLUGIN 或 SERVER_ENCRYPT 认证方法在服务器上进行认证。认证采用 GSS-API 插件，而且，如果客

客户端不支持在服务器所支持的插件列表中所找到的任何插件的话，则利用 **KERBEROS** 对客户端进行验证。如果客户端不支持 **Kerberos** 安全性协议，则利用 **SERVER_ENCRYPT** 认证方法对客户端进行验证。

4.2 在服务器上规定认证类型

1. 要想查看当前的认证类型，可以使用下面的命令。在本例中，当前的认证方法是 **SERVER**。

```
db2 GET DATABASE MANAGER CONFIGURATION
```

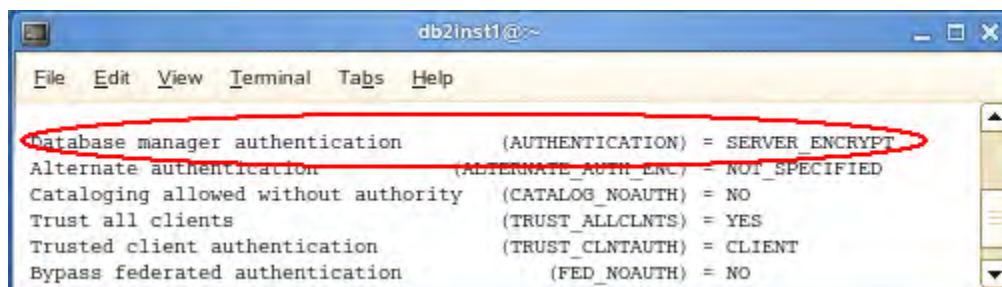


```
db2inst1@~
File Edit View Terminal Tabs Help
Database manager authentication (AUTHENTICATION) = SERVER
Alternate authentication (ALTERNATE_AUTH_ENC) = NOT_SPECIFIED
Cataloging allowed without authority (CATALOG_NOAUTH) = NO
Trust all clients (TRUST_ALLCLNTS) = YES
Trusted client authentication (TRUST_CLNTAUTH) = CLIENT
Bypass federated authentication (FED_NOAUTH) = NO
```

2. 执行下面的命令，把认证设置修改为 **SERVER_ENCRYPT**。只有 **SYSADM** 组的成员才能够对实例的安全性配置参数进行修改。

```
db2 UPDATE DBM CFG USING 认证 SERVER_ENCRYPT
```

3. 重新发出步骤 1 中的命令，以查看当前的认证设置。



```
db2inst1@~
File Edit View Terminal Tabs Help
Database manager authentication (AUTHENTICATION) = SERVER_ENCRYPT
Alternate authentication (ALTERNATE_AUTH_ENC) = NOT_SPECIFIED
Cataloging allowed without authority (CATALOG_NOAUTH) = NO
Trust all clients (TRUST_ALLCLNTS) = YES
Trusted client authentication (TRUST_CLNTAUTH) = CLIENT
Bypass federated authentication (FED_NOAUTH) = NO
```

4. 执行下面的命令，把认证设置修改回 **SERVER**。

```
db2 UPDATE DBM CFG USING AUTHENTICATION SERVER
```

4.3 在客户端上规定认证类型

客户端认证类型存储在客户端的数据库目录中。要查看系统所知悉的数据库列表，请使用下面的命令：

```
db2 LIST DATABASE DIRECTORY
```

要想修改某个连接的认证类型，需要利用新的认证类型从数据库目录中对数据库进行重新编目。

在对远程客户端进行编目时对认证类型的规定是可选的。如果已经规定了某个认证类型，它就必须与数据服务器上规定的值相匹配或者相兼容。如果不匹配，则连接失败。

为了利用 `SERVER_ENCRYPT` 认证对数据库连接进行编目，可以输入以下命令：

```
db2 CATALOG DATABASE sample AT NODE mynode AUTHENTICATION  
SERVER_ENCRYPT
```

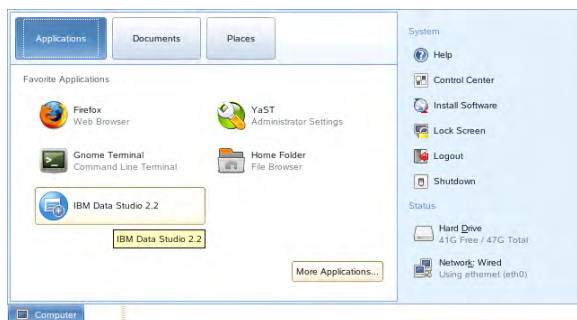
注： 由于数据库可能已经被编目，你或许会收到如下错误提示：

SQL1005N 数据库别名 "sample" 在本地数据库目录中或者系统数据库目录中已经存在。

4.4 利用 Data Studio 管理认证参数

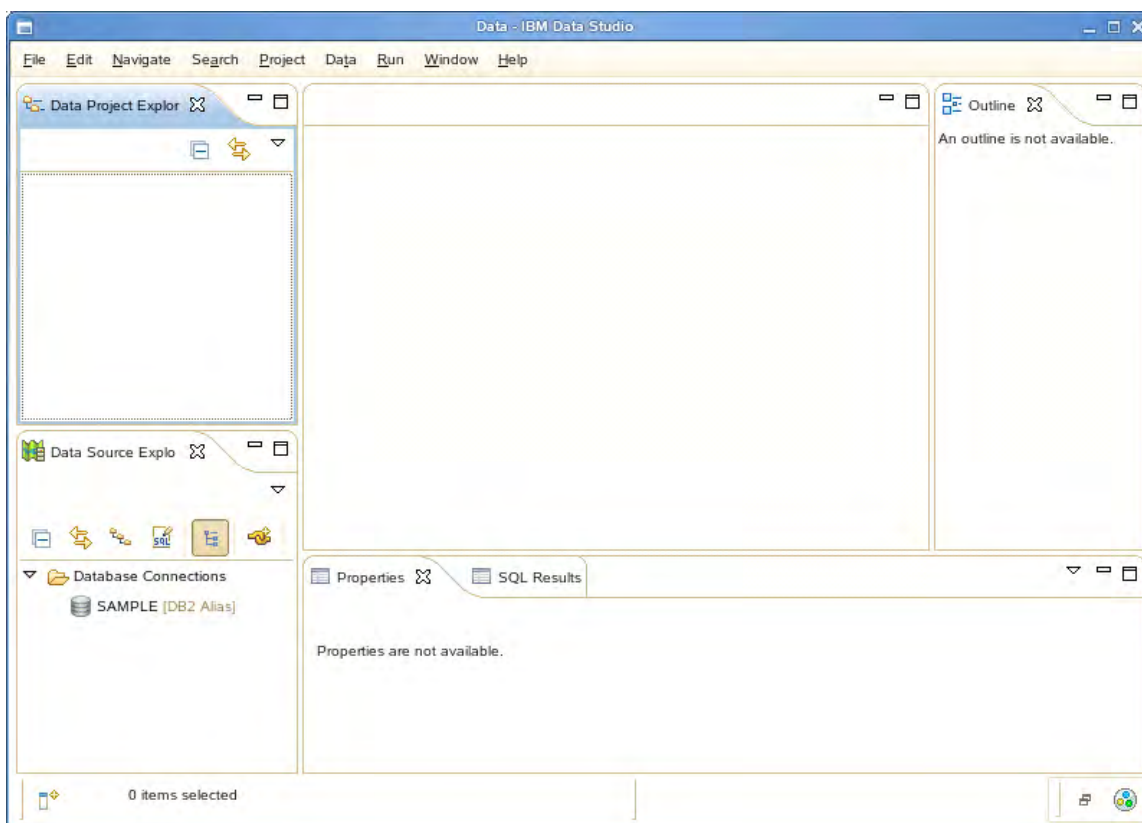
认证类型的配置也可以很方便地通过 Data Studio 来管理。

1. 在屏幕左下角点击**计算机**按钮，启动 Data Studio，然后选择 **IBM Data Studio 2.2**。



2. 在 **Select a workspace**（选择工作区）对话框中，接受默认的路径并选中 **Use this as the default**（用作默认值）及 **do not ask again**（不再询问）检查框。点击 **OK**（确定）。

3. 最小化“欢迎”窗口，你将进入如下所示的 Data 视角。



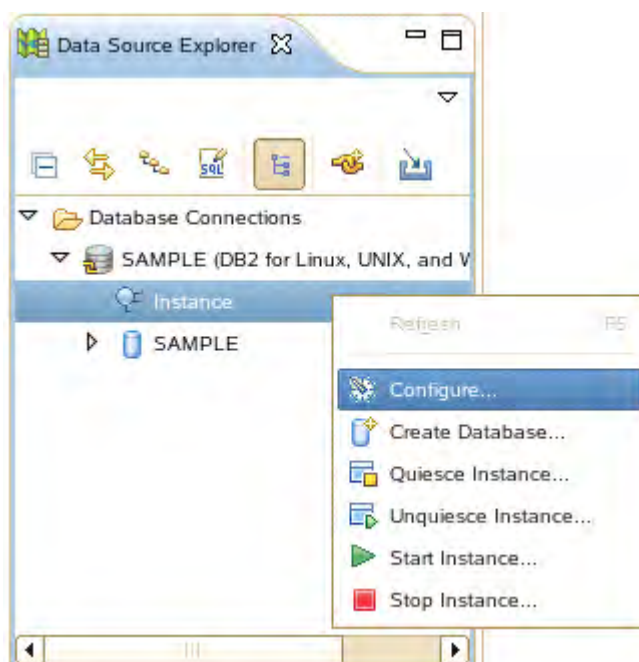
4. 接入数据库 Sample。

从 **Database Source Explorer** 面板中（左下侧面板），展开“连接”。右击 **SAMPLE** 数据库并选择 **Connect**（连接）。按照下面的证书登录：

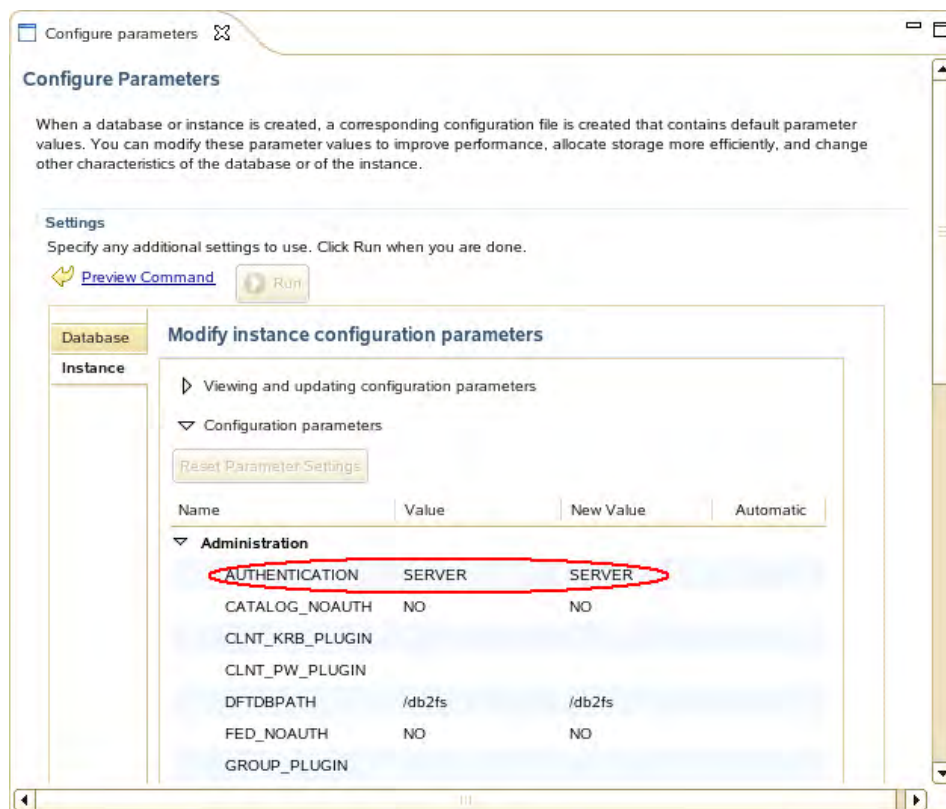
用户名：db2inst1

密码：password

5. 建立连接之后，右击 **Instance**（实例）并选择 **Configure**（配置），即可以查看实例配置参数。



认证参数显示在配置参数对话框的顶部。要想修改当前设置，只需双击该参数，然后为认证方式指定一个新值即可。



5. 授权

用户通过认证之后，授权过程成为第二道安全机制，它将决定用户能够在数据库或实例中进行哪些操作。授权由特权、权限、角色以及基于标号的访问控制 (LBAC) 凭证。

用户的权限决定了其行使数据库及实例管理操作的能力。

特权提供了比权限更加细粒化的控制。特权定义了用户可以创建及删除的对象，以及用户能够用来访问数据表、视图、索引及程序包等对象的命令。

角色是一种把用户集合到一起的方法，以便对特权进行集中管理，而不是逐一地管理。

LBAC 使用安全性标号来控制谁对表中的具体行及/或列有读访问权及写访问权。LBAC 未包含在 DB2 Express-C 中，因此 LBAC 的实施超出了本实验的范围。

5.1 权限

在管理数据库和实例时需要权限。权限可以被划分为如下两类：

- 实例级权限
- 数据库级权限

5.1.1 实例级权限

实例级权限能够让你行使全实例范围内的功能，例如创建和升级数据库、管理表空间以及监控你的实例上的活动和性能。实例级权限不提供对数据库表中数据的访问权。

实例级权限	说明
SYSADM	用于那些从总体上管理实例的用户
SYSCTRL	用于那些管理数据库管理器实例的用户
SYSMAINT	用于那些维护实例中数据库的用户
SYSMON	用于那些监控实例及其数据库的用户

实例级权限是通过数据库管理器配置来授予的，只能指定给组。组是在操作系统上定义的，然后可以把具体的用户指定到这些组中。要想在把 SYSADM、SYSCTRL、

SYSMAINT 或 **SYSMON** 权限授予某个组，可以把数据库管理器配置参数 **SYSADM_GROUP**、**SYSCTRL_GROUP**、**SYSMAINT_GROUP** 和 **SYSMON_GROUP** 设定给具体的操作系统组。

在默认情况下，在 **UNIX** 系统中，**SYSADM** 组被设定为实例所有者 **DB2GRP1** 的基组 (**primary group**)。任何属于该组的用户都具有 **SYSADM** 权限。在 **Windows** 系统中，本地管理员组的所有成员均被授予 **SYSADM** 权限。

从下面的命令中可以看到，**DB2GRP1** 被定义为 **SYSADM** 组。

```
db2 get dbm cfg | grep SYSADM_GROUP
```

```

db2inst1@db2rules:~> db2 get dbm cfg | grep SYSADM_GROUP
SYSADM group name (SYSADM_GROUP) = DB2GRP1
db2inst1@db2rules:~>

```

5.1.2 数据库级权限

数据库权限能够让用户在数据库层次上开展活动，这样就能够让用户完成诸如授予和撤销特权、插入、查询、删除及更新数据、管理负载等功能。

数据库级权限	说明
SECADM	用于管理数据库中安全性的用户
DBADM	用于管理数据库的用户
ACCESSCTRL	用于需要授予和撤销权限及特权的用户（SECADM、DBADM、ACCESSCTRL 和 DATAACCESS 权限除外。授予和撤销这些权限时需要 SECADM 权限）
DATAACCESS	用于需要访问数据的用户
SQLADM	用于监控及优化 SQL 查询的用户
WLMADM	用于管理工作负载的用户
EXPLAIN	用于需要解释查询计划的用户

5.2 特权

特权比权限更加细粒化。特权定义了用户或组能够创建、修改、删除及访问的数据库对象。特权可以通过三种不同途径获得：

显式：特权可以由对该对象具有 **ACCESSCTRL** 权限、**SECADM** 权限或 **CONTROL** 特权的用户通过 **GRANT** 或 **REVOKE** 命令加以显式授予或取消。如果某个用户在被授予特权时针对某个对象具有 **WITH GRANT OPTION**，该用户也可以显式授予特权。

隐式：当某个用户创建了一个数据库对象时，该用户将隐式获得有关该对象的全部特权。例如，当用户创建了一个数据库时，该用户就隐式获得了此数据库的 **DBADM** 权限。

间接：间接特权通常涉及到程序包。当用户执行一个程序包时，该程序包可能需要此用户所不具有的特权。此时将临时对该用户间接授予这些特权，以便执行该程序包。

5.3 练习 – 授予及撤销权限和特权

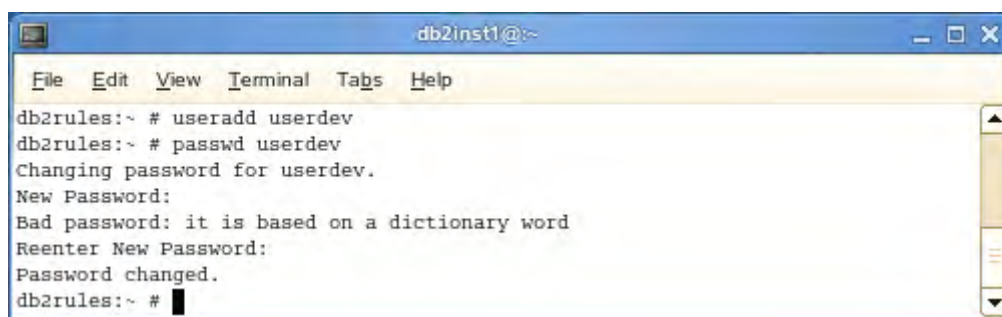
迄今为止，在本实验中，你已经以实例管理员 (**db2inst1**) 的身份发出了所有的数据库命令，实例管理员拥有访问 **DB2** 中所有实用程序、数据及数据库对象的特权。重要的是，应当仅向用户授予完成其任务所必要的特权。

在下面的场景中，假设一名新队员加入了你的团队。我们来看看如何向他授予特定的权限和特权，以保护数据库的安全。

1. 右击桌面，选择“打开终端”项，从而打开一个终端窗口。
2. **DB2** 利用基础性操作系统的安全性机制来管理用户和密码。因此，我们需要在操作系统层次上创建新用户及群组。

以根用户身份登录到操作系统中，添加新用户 **USERDEV**。将其密码修改为 'password'。

```
su -  
Password: password  
  
useradd userdev  
passwd userdev  
New password: password  
exit
```



```

db2rules:~ # useradd userdev
db2rules:~ # passwd userdev
Changing password for userdev.
New Password:
Bad password: it is based on a dictionary word
Reenter New Password:
Password changed.
db2rules:~ # █

```

3. 权限和特权如果未被授予的话，将会被隐式拒绝。在添加新用户之后，他没有 PUBLIC 组中所定义范围之外的权限或特权。

试试利用用户 USERDEV 的身份查询样例数据库的 'EMPLOYEE' 表。你会发现，该项操作被禁止了，因为 USERDEV 不具备所需要的权限或特权。

```

db2
CONNECT TO SAMPLE USER userdev USING password
SELECT * FROM DB2INST1.EMPLOYEE

```



```

db2 => CONNECT TO SAMPLE USER userdev USING password

Database Connection Information

Database server          = DB2/LINUX 9.7.1
SQL authorization ID    = USERDEV
Local database alias    = SAMPLE

db2 => SELECT * from DB2INST1.EMPLOYEE
SQL0551N "USERDEV" does not have the required authorization or privilege to
perform operation "SELECT" on object "DB2INST1.EMPLOYEE".  SQLSTATE=42501
db2 => █

```

4. USERDEV 是你的团队中的一名应用开发人员，他将开发和测试程序。他需要对数据库中的各种表具有 SELECT、INSERT、UPDATE 及 DELETE 访问权。他还需要能够向数据库中添加新的程序包并执行应用以便进行测试；因此，他需要被授予 BINDADD 权限。

为了向 USERDEV 授予这些特权，你必须是 SYSADM。以 DB2 实例所有者 (db2inst1) 的身份登录到你的机器中，然后发出 GRANT 命令。

```

CONNECT TO SAMPLE USER db2inst1 USING password
GRANT CREATETAB, BINDADD, CONNECT ON DATABASE TO USER userdev
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE employee TO USER userdev

```

5. USERDEV 现在拥有查询及修改 EMPLOYEE 表的特权了。再次试试运行第 3 步中的命令。

```

db2inst1@:~
File Edit View Terminal Tags Help
db2 => CONNECT TO SAMPLE USER userdev USING password

Database Connection Information

Database server      = DB2/LINUX 9.7.1
SQL authorization ID = USERDEV
Local database alias = SAMPLE

db2 => SELECT * FROM DB2INST1.EMPLOYEE

EMPNO  FIRSTNME  MIDINIT  LASTNAME  WORKDEPT  PHONENO  HIREDATE  JOB  EDLEVEL  SEX  BIRTHDATE  SALARY  BONUS  COMM
-----
000010 CHRISTINE  I        HAAS      A00       3978      01/01/1995  PRES  18 F      08/24/1963  152750.00  1000.00  4220.00
000020 MICHAEL   L        THOMPSON  B01       3476      10/10/2003  MANAGER  18 M      02/02/1978  94250.00  800.00  3300.00
000030 SALLY    A        KWAN      C01       4738      04/05/2005  MANAGER  20 F      05/11/1971  98250.00  800.00  3060.00
000050 JOHN     B        GEYER     E01       6789      08/17/1979  MANAGER  16 M      09/15/1955  80175.00  800.00  3214.00
000060 IRVING   F        STERN     D11       6423      09/14/2003  MANAGER  16 M      07/07/1975  72250.00  500.00  2580.00
000070 EVA      D        PULASKI  D21       7831      09/30/2005  MANAGER  16 F      05/26/2003  96170.00  700.00  2893.00
000090 EILEEN   W        HENDERSON E11       5498      08/15/2000  MANAGER  16 F      05/15/1971  89750.00  600.00  2380.00
000100 THEODORE Q        SPENSER  E21       0972      06/19/2000  MANAGER  14 M      12/18/1980  86150.00  500.00  2092.00
000110 VINCENZO G        LUCCHESI A00       3490      05/16/1988  SALESREP  19 M      11/05/1959  66500.00  900.00  3720.00
000120 SEAN     O'CONNELL A00       2167      12/05/1993  CLERK    14 M      10/18/1972  49250.00  600.00  2340.00
000130 DELORES  M        QUINTANA C01       4578      07/28/2001  ANALYST  16 F      09/15/1955  73800.00  500.00  1904.00
000140 HEATHER  A        NICHOLLS C01       1793      12/15/2006  ANALYST  18 F      01/19/1976  68420.00  600.00  2274.00
000150 BRUCE    ADAMSON  D11       4510      02/12/2002  DESIGNER  16 M      05/17/1977  55280.00  500.00  2022.00
000160 ELIZABETH R        PIANKA   D11       3782      10/11/2006  DESIGNER  17 F      04/12/1980  62250.00  400.00  1780.00
000170 MASATOSHI J        YOSHIMURA D11       2890      09/15/1999  DESIGNER  16 M      01/05/1981  44680.00  500.00  1974.00
000180 MARILYN  S        SCOTTEN  D11       1682      07/07/2003  DESIGNER  17 F      02/21/1979  51340.00  500.00  1707.00
000190 JAMES    H        WALKER   D11       2986      07/26/2004  DESIGNER  16 M      06/25/1982  50450.00  400.00  1636.00
000200 DAVID    BROWN    D11       4501      03/03/2002  DESIGNER  16 M      05/29/1971  57740.00  600.00  2217.00
000210 WILLIAM T        JONES    D11       0942      04/11/1998  DESIGNER  17 M      02/23/2003  68270.00  400.00  1462.00
000220 JENNIFER K        LUTZ     D11       0672      08/29/1998  DESIGNER  18 F      03/19/1978  49840.00  600.00  2387.00
000230 JAMES    J        JEFFERSON D21       2094      11/21/1996  CLERK    14 M      05/30/1980  42180.00  400.00  1774.00
000240 SALVATORE M        MARINO   D21       3780      12/05/2004  CLERK    17 M      03/31/2002  48760.00  600.00  2301.00
000250 DANIEL  S        SMITH    D21       0961      10/30/1999  CLERK    15 M      11/12/1969  49180.00  400.00  1534.00
000260 SYBIL    P        JOHNSON  D21       8953      09/11/2005  CLERK    16 F      10/05/1976  47250.00  300.00  1380.00
000270 MARIA    L        PEREZ    D21       9001      09/30/2006  CLERK    15 F      05/26/2003  37380.00  500.00  2190.00
000280 ETHEL    R        SCHNEIDER E11       8997      03/24/1997  OPERATOR  17 F      03/28/1976  36250.00  500.00  2100.00
000290 JOHN     R        PARKER   E11       4502      05/30/2006  OPERATOR  12 M      07/09/1985  35340.00  300.00  1227.00
000300 PHILIP  X        SMITH    E11       2095      06/19/2002  OPERATOR  14 M      10/27/1976  37750.00  400.00  1420.00
000310 MAUDE    F        SETRIGHT E11       3332      09/12/1994  OPERATOR  12 F      04/21/1961  35900.00  300.00  1272.00
000320 RAMLAL  V        MEHTA    E21       9990      07/07/1995  FIELDREP  16 M      08/11/1962  39950.00  400.00  1526.00
000330 WING     LEE      E21       2103      02/23/2006  FIELDREP  14 M      07/18/1971  45370.00  500.00  2030.00
000340 JASON    R        GOUNOT   E21       5698      05/05/1977  FIELDREP  16 M      05/17/1956  43840.00  500.00  1907.00
200010 DIAN     J        HEMMINGER A00       3978      01/01/1995  SALESREP  18 F      08/14/1973  46500.00  1000.00  4220.00
200120 GREG     ORLANDO  A00       2167      05/05/2002  CLERK    14 M      10/18/1972  39250.00  600.00  2340.00
200140 KIM      N        NATZ     C01       1793      12/15/2006  ANALYST  18 F      01/19/1976  68420.00  600.00  2274.00
200170 KIYOSHI YAMAMOTO D11       2890      09/15/2005  DESIGNER  16 M      01/05/1981  64680.00  500.00  1974.00
200220 REBA     K        JOHN     D11       0672      08/29/2005  DESIGNER  18 F      03/19/1978  69840.00  600.00  2387.00
200240 ROBERT   M        MONTEVERDE D21       3780      12/05/2004  CLERK    17 M      03/31/1984  37760.00  600.00  2301.00
200280 EILEEN   R        SCHWARTZ E11       8997      03/24/1997  OPERATOR  17 F      03/28/1966  46250.00  500.00  2100.00
200310 MICHELLE F        SPRINGER E11       3332      09/12/1994  OPERATOR  12 F      04/21/1961  35900.00  300.00  1272.00
200330 HELENA   WONG     E21       2103      02/23/2006  FIELDREP  14 F      07/18/1971  35370.00  500.00  2030.00
200340 ROY      R        ALONZO   E21       5698      07/05/1997  FIELDREP  16 M      05/17/1956  31840.00  500.00  1907.00

42 record(s) selected.

db2 =>

```

5.4 细粒化特权--视图

有两种方式来限制对表中具体部分数据的访问：视图，或者基于标号的访问控制 (LBAC)。LBAC 未包含在 DB2 Express-C 中，因此，LBAC 的实施超出了本实验的范围。我们重点讨论视图的实施。

视图是虚拟的表（动态计算出来，而且不会被显式保存），它们从一个或多个表或视图中推延出来。它们可以用来向用户提供定制的数据子集，让用户看到同一数据集合的不同展示，或者隐藏用户无权访问的数据。视图可以进行删除、插入及更新操作，也可以是只读的。该分类指明了允许对视图进行的 SQL 操作的种类。

下面我们利用 **sample** 数据库中的 **employee** 表来介绍如何实施视图。

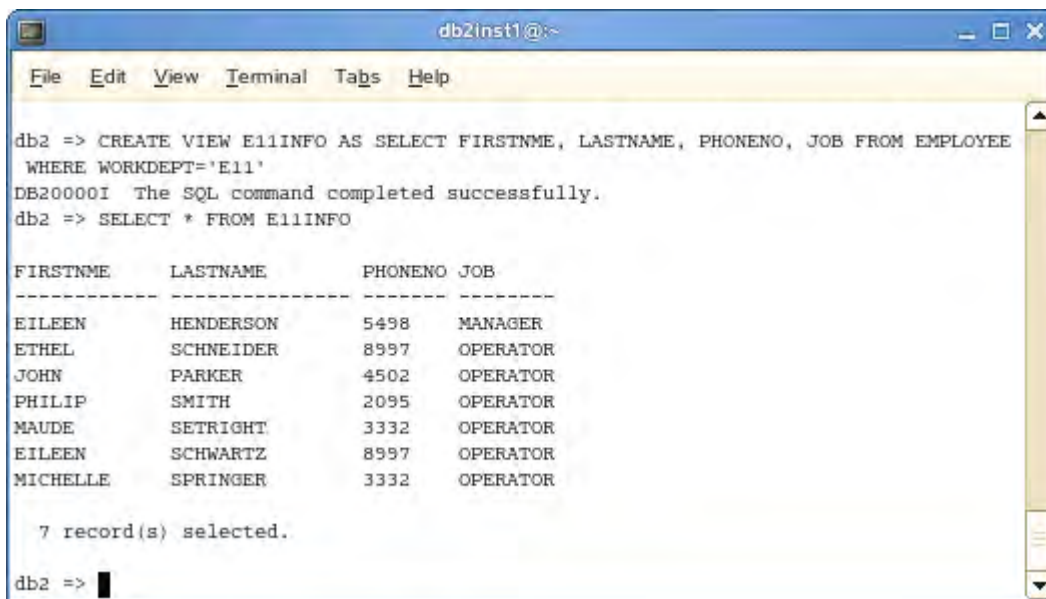
Employee 表中含有一些保密信息，例如雇员的工资和奖金。这些信息不是每个人都应当看到的。为了保护这些保密信息，可以根据 **employee** 表创建一个视图，以限制用户看到某些列。这将授予用户对视图的访问权，而不是对基本表的访问权。

1. 我们将创建一份视图，其中含有部门 **E11** 的人员构成的名录。该名录仅含有名、姓、电话号码及职位。

```
CONNECT TO SAMPLE
CREATE VIEW E11INFO AS SELECT FIRSTNME, LASTNAME, PHONENO, JOB FROM
EMPLOYEE WHERE WORKDEPT='E11'
```

2. 如果一个用户对该视图发出查询语句，他只能看到如下四列：

```
SELECT * FROM E11INFO
```



```
db2inst1@~
File Edit View Terminal Tabs Help

db2 => CREATE VIEW E11INFO AS SELECT FIRSTNME, LASTNAME, PHONENO, JOB FROM EMPLOYEE
WHERE WORKDEPT='E11'
DB20000I The SQL command completed successfully.
db2 => SELECT * FROM E11INFO

FIRSTNME      LASTNAME      PHONENO      JOB
-----
EILEEN        HENDERSON     5498         MANAGER
ETHEL         SCHNEIDER     8997         OPERATOR
JOHN          PARKER        4502         OPERATOR
PHILIP        SMITH         2095         OPERATOR
MAUDE         SETRIGHT      3332         OPERATOR
EILEEN        SCHWARTZ      8997         OPERATOR
MICHELLE      SPRINGER      3332         OPERATOR

7 record(s) selected.

db2 => █
```

3. 最后一步是撤销对基本表的访问，并授予对该视图的访问权：

```
REVOKE ALL ON employee FROM USER userdev
GRANT SELECT ON e11info TO USER userdev
```

6. 角色

角色是一种数据库对象，它把一项或多项特权组合到一起，并可以通过 **GRANT** 语句把角色指定给用户、组、**PUBLIC** 或者其他角色。角色简化了对特权的维护和管理。

角色可以依照组织结构的体系来确定。在创建角色时可以直接映射组织机构中的职责。利用角色，就不必再把相同的特权集合授予负有特定职责的每个具体用户，而是可以把这个特权集合授予某个角色，然后再向用户授予与其职责对应的角色成员资格。当用户的职责变化时，可以很容易地授予或撤销其角色成员资格。

6.1 示例--角色

继续上一节的场景，假设你的团队扩充了，有更多的应用开发人员加入你的团队。你不必管理他们每个人的特权，而是可以更容易地利用角色进行管理。

安全性管理员拥有创建、删除、授予、撤销及注释角色的权限。

1. 连接到 **sample** 数据库，并创建一个称为 'developer' 的新角色。

```
CONNECT TO SAMPLE  
CREATE ROLE DEVELOPER
```

2. 在定义了角色之后，利用 **GRANT** 语句把权限和特权授予该角色。

```
GRANT CREATETAB, BINDADD, CONNECT ON DATABASE TO ROLE developer  
GRANT SELECT, INSERT, UPDATE, DELETE ON TABLE db2inst1.employee TO ROLE  
developer
```

3. 角色 **DEVELOPER** 被授予用户 **USERDEV**:

```
GRANT ROLE DEVELOPER TO USER USERDEV
```

4. 当 **USERDEV** 改变了职务，不再担任开发人员时，可以从数据库中撤销其角色。

```
REVOKE ROLE DEVELOPER FROM USER USERDEV
```



© Copyright IBM Corporation 2011
All Rights Reserved.

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

IBM, IBM (logo), and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both

UNIX is a registered trademark of The Open Group in the United States, other countries, or both

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this publication is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of February 2009, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.

IBM DB2® 9.7

备份和恢复 亲自动手实验

信息管理云计算能力中心

IBM（加拿大）研究院

目录

目录.....	2
1. 前言.....	3
2. 基本设置.....	3
2.1 环境设置要求.....	3
2.2 准备步骤.....	3
3. DB2 日志记录.....	4
3.1 日志参数.....	4
3.1.1 LOGFILSIZ.....	5
3.1.2 LOGPRIMARY 和 LOGSECOND.....	5
3.1.3 LOGBUFSZ.....	5
3.2 日志类型.....	6
3.2.1 循环日志.....	6
3.2.2 归档日志.....	6
4. 恢复场景.....	8
4.1 场景 1 – 整个数据库被意外删除或者被损坏.....	8
4.2 场景 2 – 数据库前滚至某个时间点.....	9
4.3 场景 3 – 增量备份和恢复.....	11
4.3.1 增量备份.....	11
4.3.2 增量恢复.....	12

1. 前言

有各种各样的情况可能会影响到数据库的完整性，其中包括系统中断、硬件故障、事务故障以及灾难。DB2 的备份和恢复能够防止你丢失数据。

本实验结束时，你将能够：

- ▶ 进行完整备份和恢复
- ▶ 把数据库恢复至某个时间点
- ▶ 进行增量备份和恢复

2. 基本设置

2.1 环境设置要求

要完成本实验，你需要以下软件：

- DB2 Academic Workshop VMware® 影像（DB2 学术研讨班 VMware® 影像）
- VMware Player 2.x 或 VMware Workstation 5.x 或者更高版本

关于如何获得这些组件的帮助信息，请遵循模块 1 中“**VMware 基础和入门**”给出的指示。

2.2 准备步骤

1. 启动 VMware® 影像。完成加载之后，当提示登录证书时，利用用户名“db2inst1”来提供 DBADM 权限：

用户名：**db2inst1**

密码：**password**

2. 键入命令“**startx**”，进入图像环境。

3. 在**桌面**上右击鼠标，选择“**打开终端**”项，即可打开终端窗口。

4. 如果 Database Manager 尚未启动，发出以下命令：

```
db2start
```

为执行本实验，你需要以其原始格式创建的 **DB2** 示例数据库。

执行下面的命令，删除（如果已经存在的话）并重建 **SAMPLE** 数据库：

```
db2 force applications all
db2 drop db sample
db2saml
```

3. DB2 日志记录

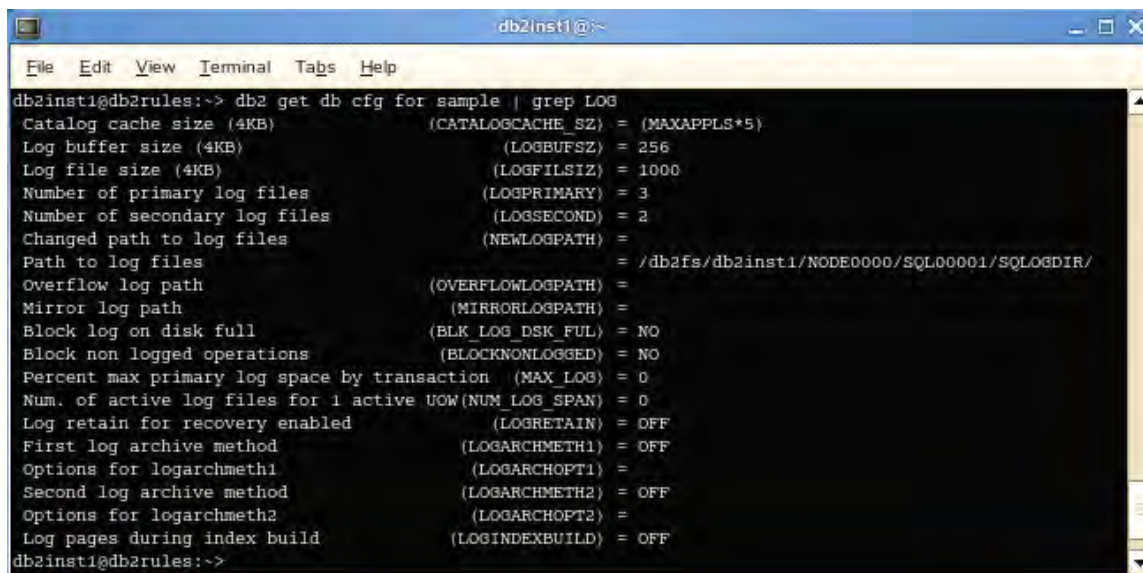
一次事务处理是一个逻辑工作单元。Every transaction performed by DB2 处理的每项事务首先写入到日志中，然后才对数据进行操作。DB2 依赖这些日志文件进行备份和恢复。

在我们讨论不同类型的 DB2 日志之前，我们首先来理解一些日志参数。

3.1 日志参数

为了查看与日志相关的数据库配置，可以运行下面的命令：

```
db2 get db cfg for sample | grep LOG
```



```
db2inst1@db2rules:~> db2 get db cfg for sample | grep LOG
Catalog cache size (4KB)          (CATALOGCACHE_SZ) = (MAXAPPLS*5)
Log buffer size (4KB)              (LOGBUFSZ) = 256
Log file size (4KB)                (LOGFILSIZ) = 1000
Number of primary log files        (LOGPRIMARY) = 3
Number of secondary log files      (LOGSECOND) = 2
Changed path to log files          (NEWLOGPATH) =
Path to log files                  = /db2fs/db2inst1/NODE0000/SQL00001/SQLLOGDIR/
Overflow log path                  (OVERFLOWLOGPATH) =
Mirror log path                    (MIRRORLOGPATH) =
Block log on disk full             (BLK_LOG_DSK_FUL) = NO
Block non logged operations        (BLOCKNONLOGGED) = NO
Percent max primary log space by transaction (MAX_LOG) = 0
Num. of active log files for 1 active UOW(NUM_LOG_SPAN) = 0
Log retain for recovery enabled     (LOGRETAIN) = OFF
First log archive method           (LOGARCHMETH1) = OFF
Options for logarchmeth1           (LOGARCHOPT1) =
Second log archive method          (LOGARCHMETH2) = OFF
Options for logarchmeth2           (LOGARCHOPT2) =
Log pages during index build       (LOGINDEXBUILD) = OFF
db2inst1@db2rules:~>
```

3.1.1 LOGFILSIZ

它是每份事务日志文件的长度，以 4KB 的页面为测量单位。默认长度为 1000 页或者 4 MB，这意味着它可以容纳最多 4 MB 的事务数据。对于大事务量 OLTP 类型的环境，可以把该长度配置得更大一些。在 OLTP 环境中，长度太小的日志文件很快就会被填满，于是就需频繁地创建新的日志文件。

3.1.2 LOGPRIMARY 和 LOGSECOND

LOGPRIMARY 是主日志文件的数量。在任何时候，数据库中总有一些未落实的事务占用着活动日志空间。活动日志空间是指由未落实的事务占用的日志空间总和。在默认情况下，其值为 3；因此，如果你有 3 个日志文件被为落实的事务所占用，则任何新的事务都将开始利用辅助日志文件。

LOGSECOND 是辅助日志文件的数量。只有当某个事务耗尽了为主日志配置的所有空间之后，才会分配辅助日志空间，其目的是为了满足不同事务活动的突然大量增长。当使用辅助日志文件的事务落实或者回滚之后，DB2 将继续使用主日志。你可以对此进行配置。在默认情况下，LOGSECOND 为 2，这意味着，如果主日志文件被未落实的事务所充满，则临时另外分配 2 个日志文件来处理峰值需求。如果所有的主日志文件和辅助日志文件都被使用完毕，则返回一条错误消息：

SQL0964C 数据库事务日志已满。

把 LOGPRIMARY 设为 5，把 LOGSECOND 设为 3；可以在终端窗口中发出以下命令：

```
db2 update database configuration for sample using LOGPRIMARY 5
db2 update database configuration for sample using LOGSECOND 3
```

可能会返回一条警告消息：

SQL1363W 所提交的一个或多个被立即修改的参数无法动态改变。对于这些配置参数，在变更生效之前，所有的应用都必须断开与本数据库的连接。

为了使配置变更生效，只需断开与数据库的连接，然后再重新连接，因为这是此时与数据库之间的唯一连接。

```
db2 terminate
db2 connect to sample
```

3.1.3 LOGBUFSZ

所有的日志记录在写入磁盘之前先写入内存。LOGBUFSZ 规定了这方面内存的大小。默认的 8 个 4KB 页面对于大多数场景而言太小了。该参数对于 OLTP 性能甚为重要。可以

把 LOGBUFSZ 设为 256，这是个不错的起始数字。在实际环境中，可以取一个 OLTP 负载，并以更高的 LOGBUFSZ 进行基准测试，以寻找最优值。

3.2 日志类型

DB2 数据库支持两种不同的日志模式：循环日志和归档日志。

3.2.1 循环日志

对于新创建的数据库，这是 DB2 默认的日志方式。它依次使用主日志文件，直至达到 LOGPRIMARY 参数所指定的日志文件数量。如果某个长时间运行的事务在结束之前耗尽了所有的主日志文件，则该事务就溢出到辅助日志文件中。当工作被落实之后，DB2 就返回至第一个日志文件并以循环方式继续进行。

这种日志方法不支持前滚恢复，因为日志文件没有保存下来，而是不断地被覆盖。只支持崩溃恢复和版本恢复。如果某数据库采用了循环日志方式，则该数据库只能通过离线备份方式进行备份。

要想使用循环日志，可以把 LOGARCHMETH1 和 LOGARCHMETH2 两个数据库配置参数设置为 OFF（关闭）。

3.2.2 归档日志

在归档日志中，所有的日志文件都被保存下来；它们从来不会被覆盖。为了实现在线备份并进行前滚恢复，数据库必须能够支持归档日志。

为了支持归档日志，你需要把 LOGARCHMETH1 的值规定为 OFF 之外的其他东西。如果 LOGARCHMETH1 和 LOGARCHMETH2 都被规定了，则归档日志将被归档两次。

无限日志是归档日志的一种变形，这时 LOGARCHMETH2 被设置为 -1。在这种日志方式中，辅助日志文件将不断被分配，直至工作单元被落实。

1. 我们现在把日志方法修改为归档日志并设置归档位置：

```
mkdir /home/db2inst1/logarch
db2 update db cfg for sample using LOGARCHMETH1
    disk:/home/db2inst1/logarch
```

2. 终止数据库连接，然后再重新接入 sample 数据库：

```
db2 terminate
db2 connect to sample
```

但是，当你试图重新接入 **sample** 数据库的时候，你会收到如下错误消息：

*SQL1116N 无法接入或激活数据库 "SAMPLE"，因为处于 BACKUP PENDING
(备份暂挂) 状态。 SQLSTATE=57019*

收到此消息是因为刚刚对此数据库启用了归档日志，因此它被置入备份暂挂状态。我们前面说过，一旦对数据库启用了归档日志，就可以进行前滚恢复了。但是，前滚恢复只能在备份影像得以恢复，而且数据库被置于“前滚暂挂”状态时才能够进行。因此，在数据库能够被使用之前，必须进行一次完整数据库备份。

3. 利用以下命令创建一个存储备份文件的目录并进行一次完整数据库备份：

```
mkdir /home/db2inst1/backups  
db2 backup database sample to /home/db2inst1/backups
```

如果不出现错误的话，你将看到类似下面的消息，但时间戳不同：

备份成功。本备份影像的时间戳是：*20100509163937*

在创建备份影像时，创建之时的时间戳以 **yyyymmddhhmmss** 的格式返回。这是有用的信息，因为 **Restore**（恢复）实用程序将利用时间戳区分多个可用的备份影像。

请记住你的备份命令返回的时间戳，在下面的练习中，它将被称为 **T1**。

T1:

4. 再次接入数据库。这次它应当成功了。

```
db2 connect to sample
```

4. 恢复场景

在本节实验中，我们来讨论几个利用 DB2 Recovery 实用程序从故障中进行恢复的场景。

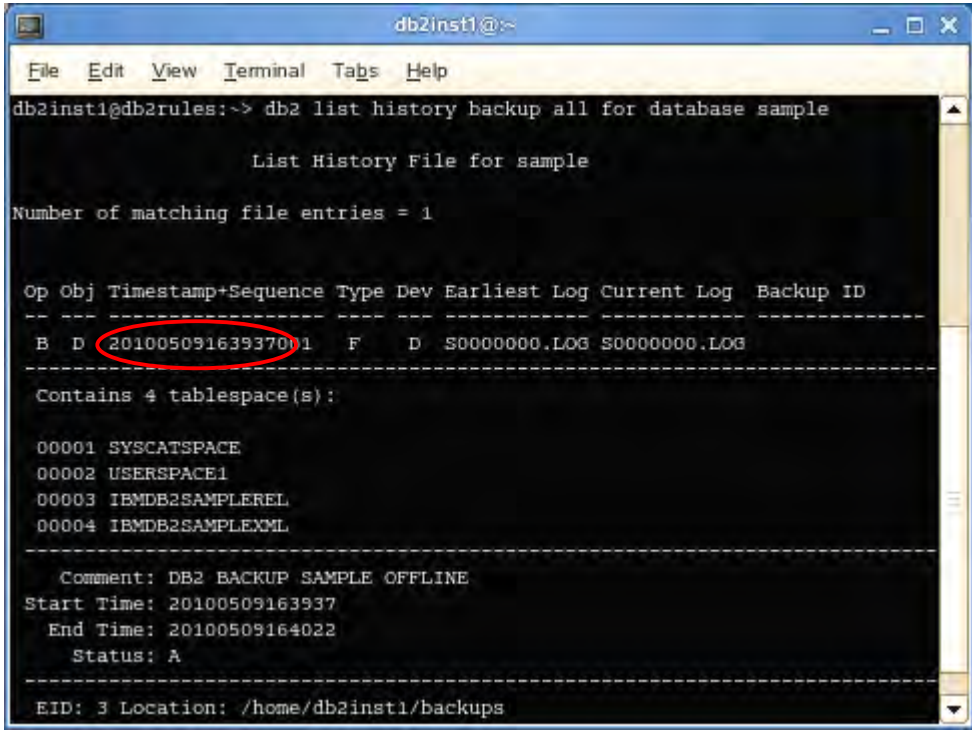
4.1 场景 1 – 整个数据库被意外删除或者被损坏

如果数据库被意外删除或者被损坏，你可以通过恢复完整备份来恢复数据库。

在本例中，我们将从练习 3.2.2.结束时获得的离线备份影像中进行恢复。如果你没有记下获得该备份时的时间戳 (T1)，你可以随时检查“恢复”历史文件以寻找该备份时间戳，方法是执行下面的命令：

```
db2 list history backup all for database sample
```

时间戳位于下面屏幕截图的圆圈中：



```
db2inst1@db2rules:~> db2 list history backup all for database sample

List History File for sample

Number of matching file entries = 1

Op Obj Timestamp+Sequence Type Dev Earliest Log Current Log Backup ID
-----
B D 20100509163937001 F D S0000000.LOG S0000000.LOG
-----

Contains 4 tablespace(s):

00001 SYSCATSPACE
00002 USERSPACE1
00003 IBMDB2SAMPLEREL
00004 IBMDB2SAMPLEXML

-----
Comment: DB2 BACKUP SAMPLE OFFLINE
Start Time: 20100509163937
End Time: 20100509164022
Status: A
-----
EID: 3 Location: /home/db2inst1/backups
```

1. 为模拟此场景，断开数据库连接并删除它：

```
db2 force applications all
db2 drop database sample
```

如果试图现在就接入 **sample** 数据库，你会收到下述错误消息：

```
db2 connect to sample
```

SQL1013N 数据库别名或数据库名称 **"sample"** 无法找到。 **SQLSTATE=43705**

2. 为了从本故障中恢复，你可以恢复原先创建的完整数据库备份。

恢复在前面的练习中创建的数据库备份影像。你需要把前面记录下的时间戳 **T1** 带入下面的命令中：

```
db2 restore database sample from /home/db2inst1/backups taken at <T1>
without rolling forward
```

请注意，在该恢复命令中，有一个 **without rolling forward**（不进行前滚）子句。由于恢复是从离线备份中进行的，在恢复之后不必进行前滚。当不需要进行前滚时，这非常有用，因为备份只需一步就能够完成了。

在恢复完成时，你应当能够接入 **sample** 数据库，而不必显式进行前滚。

4.2 场景 2 – 数据库前滚至某个时间点

前滚是在进行恢复之后应用事务日志文件的过程。例如，假设最后一次备份是在星期日进行的，而数据库在随后的星期二丢失了。在从星期日的备份恢复之后，就需要应用日志文件中的事务，以便恢复在备份之后才执行的那些事务。这是通过前滚至 **END OF LOGS**（日志结束）来实现的。

可能存在这样的情况：我们不希望应用所有的事务。例如，假设有大量的记录被用户错误地从数据库中删除了。在这种情况下，为了恢复恢复所有被删除的数据，更为适当的方式是前滚至删除发生之前的么 **POINT IN TIME**（时间点）。

1. 为模拟此场景，我们从表中删除一些行。

在开始之前，首先检查一下 **sample** 数据库中原始 **STAFF** 表的行数：

```
db2 connect to sample
db2 "select count(*) from staff"
```

STAFF 表应为 **35** 行。

现在运行下面的命令，从 **STAFF** 表中删除一些数据：

```
db2 "delete from staff where dept=10"
```

执行删除命令之后再检查 STAFF 表的行数:

```
db2 "select count(*) from staff"
```

现在 STAFF 表应当是 31 行。

2. 我们现在对 EMPLOYEE 表运行另外的删除语句。但是，假设这些行是被意外删除的。

运行 “date” 命令，并记下我们在 “意外” 发出删除语句之前的时间戳。

```
date +%F-%H.%M.%S
```

我们把该时间戳称为 T2。把它记录下来，因为需要把它用作恢复时间点：

```
T2:
```

现在检查原始 EMPLOYEE 表的行数:

```
db2 "select count(*) from employee"
```

EMPLOYEE 表的行数应当是 42。

现在我们偶然地从 EMPLOYEE 表中删除一些数据:

```
db2 "delete from employee where edlevel=12"
```

执行该删除语句之后，检查 EMPLOYEE 表的行数:

```
db2 "select count(*) from employee"
```

EMPLOYEE 表中现在有 39 行。

3. 你刚才从 EMPLOYEE 表删除的那些行原本是不应当删除的。如果我们把该数据库恢复至最后一次完整备份，对 STAFF 表各行的删除操作将将被撤销。对此，我们可以仅仅恢复至对 EMPLOYEE 发出删除语句的时间点，在本例中即 T2。
4. 把数据库恢复至我们在练习 3.2.2 中获得的最后一次备份影像，这是时间点 T1:

```
db2 restore database sample from /home/db2inst1/backups taken at <T1>
without prompting
```

5. 既然数据库已经被恢复了，就可以继续前滚至对表 EMPLOYEE 发出删除命令的之前的时间点，这是 T2。

```
db2 rollforward db sample to <T2> using local time
```


请注意，必须以如下格式提供前滚时间戳：**yyyy-mm-dd-hh.mm.ss**。

6. 最后，解除数据库的“前滚暂挂”状态。执行下面的命令：

```
db2 rollforward database sample stop
```

7. 接入 **sample** 数据库，检查 **STAFF** 表及 **EMPLOYEE** 表的行数。

```
db2 connect to sample
db2 "select count(*) from staff"
db2 "select count(*) from employee"
```

可以看到，从 **STAFF** 表返回的行数是 **31**，从 **EMPLOYEE** 表返回的行数是 **42**。

从 **EMPLOYEE** 表中“偶然”删除的行已经通过执行时间点恢复得以复原。前滚一直进行到发出删除语句之时。删除语句是在此时间点之后发出的。因此，它没有被重演。

在本例中，如果进行了 **END OF LOGS**（日志结束）前滚，将会重演对 **EMPLOYEE** 表的删除语句，从而会把这些行再次删除。当数据库被丢失之后，**END OF LOGS** 选项是很有用的，这时需要通过所有可用的日志进行恢复，以确保全部事务都能够得到复原。

4.3 场景 3 – 增量备份和恢复

4.3.1 增量备份

随着数据库长度越来越大，进行完整备份将是一项十分昂贵的工作，这既反映在备份影像的存储量上，也反映在执行备份所需要的时间上。这时候就需要采用增量备份。增量备份可以让用户仅备份自从上一次备份以来发生的变化，而无需每次都备份整个数据库。

要想使用增量备份，必须对数据库启用它。这可以通过打开 **TRACKMOD** 数据库配置参数来实现。当 **TRACKMOD** 被打开时，数据库将跟踪已经被修改的表空间。当执行增量备份命令时，它将跳过自从上一次备份以来未被修改过的表空间。

1. 把 **TRACKMOD** 数据库配置参数置为 **ON**（打开）状态：

```
db2 connect to sample
db2 update db cfg for sample using TRACKMOD ON
```

这是将出现一条警告信息：

SQL1363W 所提交的一个或多个被立即修改的参数无法动态改变。对于这些配置参数，在变更生效之前，所有的应用都必须断开与本数据库的连接。

2. 为了使配置变更生效，重新接入数据库。

```
db2 terminate
db2 connect to sample
```

- 增量备份需要一次完整备份作为增量变化的参考点。利用在线模式创建该数据库的备份：

```
db2 backup database sample online to /home/db2inst1/backups
```

记下该备份的时间戳，把它称为 T3。

T3:

- 对 STAFF 表进行一些修改，增加每个人的工资：

```
db2 connect to sample
db2 "update staff set salary=salary*0.9"
```

- 在把 TRACKMOD 修改为 ON 状态使数据库支持增量备份之后，并且在创建了数据库的完整备份之后，现在就可以进行增量备份了，即仅包含发生的变化：

```
db2 backup db sample incremental to /home/db2inst1/backups
```

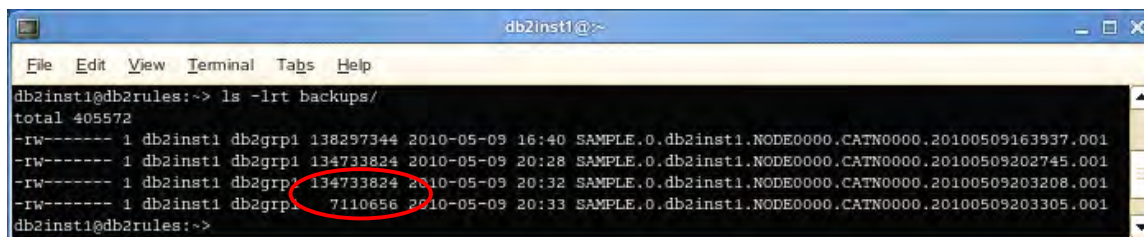
记下创建增量备份时的时间戳。把它称为 T4。

T4:

- 比较完整备份影像与增量备份影像的长度。在命令提示符下，运行下面的命令来查看长度：

```
ls -lrt /home/db2inst1/backups
```

圆圈所示为最后两次备份影像的长度。可以看到，最后一次影像（增量备份影像）的长度要比其上面的那个影像（完整备份影像）小许多。这是因为增量影像仅包含最后一次完整备份以来发生的变化。自最后一次完整备份以来未被修改过的任何表空间都不会被包含到增量数据库备份中。



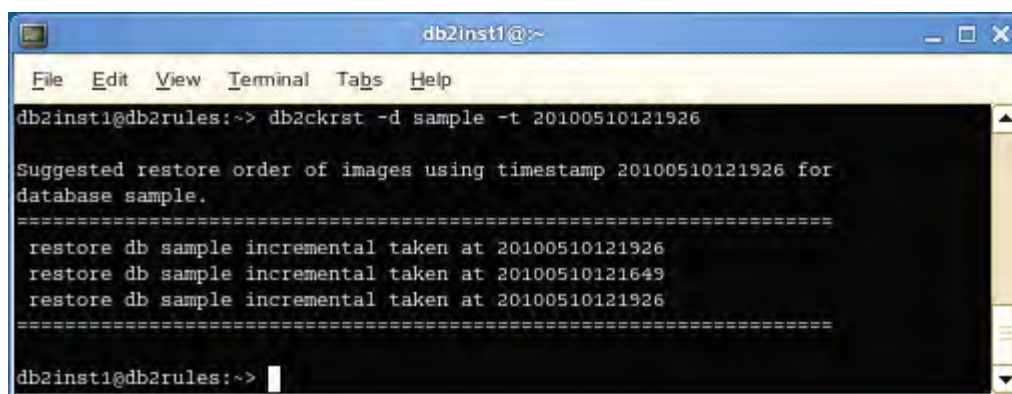
```
db2inst1@db2rules:~$ ls -lrt backups/
total 405572
-rw-r----- 1 db2inst1 db2grp1 138297344 2010-05-09 16:40 SAMPLE.0.db2inst1.NODE0000.CATN0000.20100509163937.001
-rw-r----- 1 db2inst1 db2grp1 134733824 2010-05-09 20:28 SAMPLE.0.db2inst1.NODE0000.CATN0000.20100509202745.001
-rw-r----- 1 db2inst1 db2grp1 134733824 2010-05-09 20:32 SAMPLE.0.db2inst1.NODE0000.CATN0000.20100509203208.001
-rw-r----- 1 db2inst1 db2grp1 7110656 2010-05-09 20:33 SAMPLE.0.db2inst1.NODE0000.CATN0000.20100509203305.001
db2inst1@db2rules:~$
```

4.3.2 增量恢复

在从增量备份中进行恢复时，必须使用完整备份、增量备份以及增量差异备份这样正确的顺序。在实际的环境中，这很快就会变得极其复杂。因此，在恢复命令中有一个 **AUTOMATIC**（自动）选项，它可以让 DB2 考虑并应用正确的备份顺序。还有一个 **MANUAL**（手动）选项，但强烈推荐使用 **AUTOMATIC** 选项。

db2ckrst 实用程序可以被用来查询数据库历史，并生成增量恢复所需要的备份影像时间戳列表。

```
db2ckrst -d sample -t <T4>
```



```
db2inst1@db2rules:~> db2ckrst -d sample -t 20100510121926

Suggested restore order of images using timestamp 20100510121926 for
database sample.
=====
restore db sample incremental taken at 20100510121926
restore db sample incremental taken at 20100510121649
restore db sample incremental taken at 20100510121926
=====
db2inst1@db2rules:~>
```

这些输出表明，最后一次增量影像将被首先读取，但仅仅是提取控制信息及头部信息。然后开始从完整备份影像中恢复数据库。最后，再次读取增量影像，这一次将应用影像中的数据。

从命令行上发出以下命令，可以把 **SAMPLE** 数据库恢复至最后一次增量备份影像。

```
db2 "restore db sample incremental automatic from
/home/db2inst1/backups taken at <T4>"
```



© Copyright IBM Corporation 2011
All Rights Reserved.

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

IBM, IBM (logo), and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

VMware is a trademark or VMware Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this publication is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of July 2009, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.



IBM DB2® 9.7

DB2 pureXML
亲自动手实验

信息管理云计算能力中心

IBM（加拿大）研究院

目录

1. 前言	3
2. 目标	3
3. 推荐读物	3
4. 设置	3
4.1 环境设置要求.....	3
4.2 登录到虚拟机.....	4
4.3 启动 DB2 服务器和管理服务器.....	4
5. 数据库创建	4
6. XQUERY	6
6.1 使用 XML 查询.....	6
7. SQL/XML	8
8. XMLTABLE 函数	9

1. 前言

在本实验中，你将有机会练习使用 SQL/XML 和 XQuery 抽取及操纵来自 XML 文档的数据。

2. 目标

本实验结束时，你将能够：

- ▶ 利用 SQL/XML 和 XQuery 操纵对象

3. 推荐读物

Getting started with DB2 Express-C eBook (DB2 Express-C 入门电子书) (第 15 章)

<https://www.ibm.com/developerworks/wikis/display/DB2/FREE+Book+Getting+Started+with+DB2+Express-C>

这是一本免费的电子书，能够让你快速了解 DB2。

4. 设置

4.1 环境设置要求

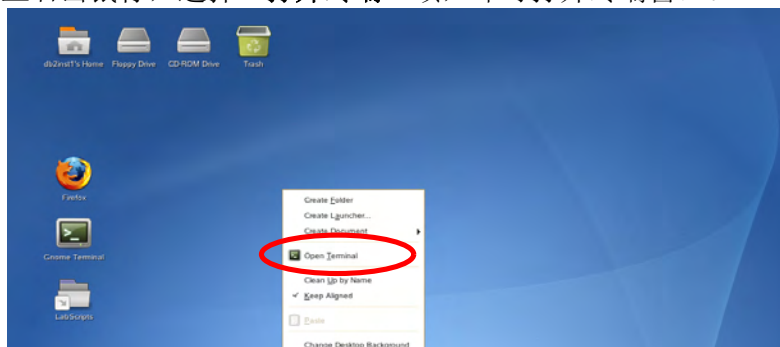
要完成本实验，你需要以下软件：

1. DB2 Academic Workshop VMware® 影像 (DB2 学术研讨班 VMware® 影像)
2. VMware Player 2.x 或 VMware Workstation 5.x 或者更高版本

关于如何获得这些组件的帮助信息，请遵循“**VMware 基础和入门**”模块中给出的指示。

4.2 登录到虚拟机

1. 利用下述信息登录到 VMware 虚拟机：
用户名：**db2inst1**
密码：**password**
2. 在命令窗口中键入命令“**startx**”，进入图像环境。
3. 在桌面上右击鼠标，选择“**打开终端**”项，即可打开终端窗口。



4.3 启动 DB2 服务器和管理服务器

1. 在终端窗口中按顺序输入以下命令，启动 **DB2** 服务器和管理服务器：

```
db2start
su - dasusr1
db2admin start
exit
```

5. 数据库创建

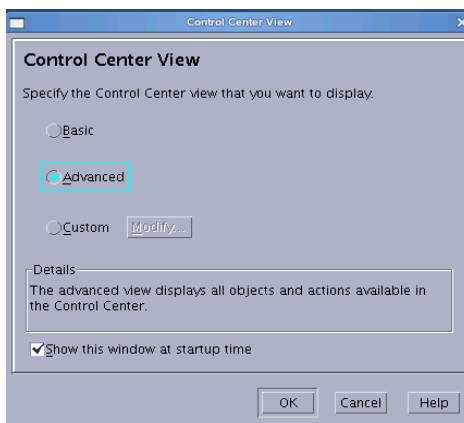
1. 在桌面上右击鼠标，选择“**打开终端**”项，即可打开终端窗口。
2. 执行下面的命令，创建名为“**purexml**”的样例数据库，下面将向该数据库中填充 XML 数据。

```
db2sampl -name purexml -xml
```

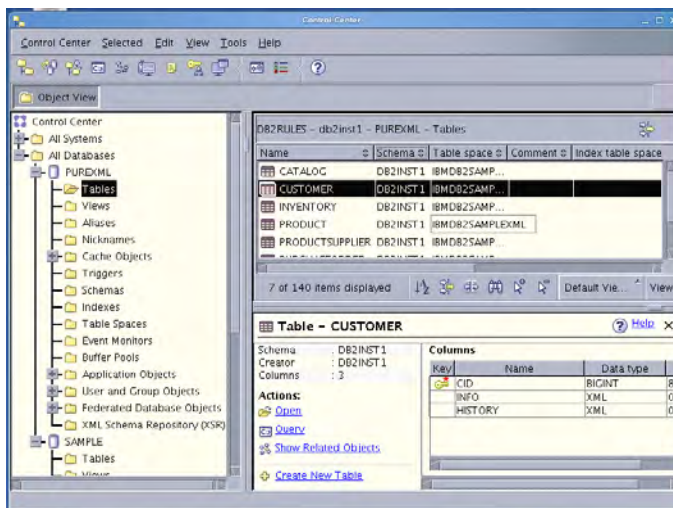

3. 我们将通过控制中心来使用 PUREXML 数据库。在命令窗口中键入以下命令，启动 **DB2 Control Center**（控制中心）：

```
db2cc
```

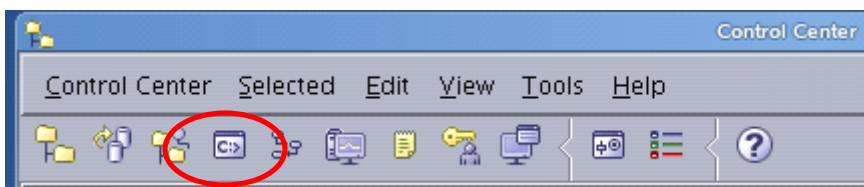
4. 在 Control Center View（控制中心视图）中，选择“**Advanced**”（高级）选择模式，以便能够使用所有的选项。然后点击“**OK**”（确定）继续。




5. 应当显示出类似下面的屏幕：



6. 点击下面圆圈中的图标，打开“命令编辑器”，以便与数据库交互。



7. 在新打开的命令编辑器中输入下面的命令，并按  按钮执行该命令，从而接入早先创建的 PUREXML 数据库：

```
connect to purexml;
```

8. 在底部面板中右击鼠标，选择“**Clear Results**”（清理结果）选项，从而清除本命令的输出结果。


6. XQuery

XQuery 用于查询 XML 数据，其方式是与使用 SQL 在数据库中查询传统关系数据一样。我们在下面的步骤中将会看到，这可以通过让 XQuery 使用 XPath 表达式语法来访问 XML 文档的特定部分来实现。

6.1 使用 XML 查询

我们将开始查询一份 XML 文档，其中含有一份客户名单以及姓名、地址、电话号码等信息。

注：下面所有的命令都应当作为一条查询放到一行上。

1. 在命令编辑器窗口中输入以下查询，然后点击  执行它，以检索结果：

```
XQuery db2-fn:xmlcolumn("CUSTOMER.INFO");
```

你或许会注意到，函数 `xmlcolumn` 返回了完整的 XML 文档。如果打算检索 XML 文档中具体的信息，我们可以使用 XPath 表达式。此外，XPath 能够让我们在方括号中规定谓词，以便过滤查询结果。

2. 在 XPath，有一个特殊的谓词，它被称为“位置谓词” (positional predicate)，能够从 XML 文档的规定位置返回节点。例如，下面的 XQuery 中有一个 XPath 表达式，其中含有位置谓词 (`[1]`)，总能够从每一个 XML 文档中返回第一个电

话号码（即每一名客户的第一个电话号码）。你可以在命令编辑器窗口中输入下面的查询，然后执行它并查看结果。

```
xquery db2-fn:xmlcolumn("CUSTOMER.INFO")
/*:customerinfo/*:phone[1]
```

3. 我们还可以查询居住在多伦多的客户的详细信息，方法是在命令编辑器窗口中输入下面的 XQuery，然后执行它并查看结果：

```
xquery db2-fn:xmlcolumn('CUSTOMER.INFO')/customerinfo[addr
/city='Toronto']
```

4. 我们还可以编写一个 XPath 表达式，用于抓取客户的助理的姓名（没有标签），这些客户的 Cid 大于 1003 并且属于加拿大。方法如下：

```
xquery db2-fn:xmlcolumn("CUSTOMER.INFO")/*:customerinfo
[@Cid > 1003]/*:addr[@country="Canada"]/../*:assistant
/*:name/text()
```

5. 现在检索具有“work”（工作）电话号码“905-555-7258”的客户的名称。方法如下：

```
xquery db2-fn:xmlcolumn('CUSTOMER.INFO')/
customerinfo/phone[@type='work' and text()='905-555-
7258']/../name
```

6. 然后我们可以利用下面的查询检索国家为加拿大的城市：

```
xquery db2-
fn:xmlcolumn('CUSTOMER.INFO')//addr[@country="Canada"]/city
```

7. 至此，我们已经看到了如何从 XML 文档中获取具体的元素/属性。XQuery 还能够在查询过程中构造 XML 文档。现在，我们编写一条 XQuery，它将返回单一的元素<ShippedItems>，其中含有已经发运的订单中所有物品的名称：

```
xquery <ShippedItems>
{db2-fn:xmlcolumn("PURCHASEORDER.PORDER")
/*:PurchaseOrder[@Status="Shipped"]/*:item/*:name}
</ShippedItems>
```

8. 除了能够临时构造 XML 片断之外，XQuery 还能够实现嵌套循环操作。下面所示的 XQuery 表达式能够返回其状态为“已发运”的采购单中所有物品的名称和数量（你可以使用第二个“for”语句对物品数量进行迭代）：

```
xquery for $po in
```

```
db2-fn:xmlcolumn("PURCHASEORDER.PORDER")/*:PurchaseOrder
for $quantity in $po/*:item/*:quantity
where $po/@Status="Shipped"
return ($po/*:item/*:name, $quantity)
```

7. SQL/XML

除了支持 XQuery 之外，DB2 还提供了众多内置的 SQL/XML 函数，它们可以把 XML 数据转变为关系数据，也可以把关系数据转变为 XML 数据。某些 SQL/XML 函数也可以用来解析、序列化 XML 数据以及把 XML 数据类型转变为关系类型。

我们现在来考察及格 SQL/XML 函数，例如 XMLQUERY、XMLEXISTS，它们可以用来抓取满足给定谓词的 XML 节点。

1. 下面的 SELECT 语句仅返回那些具有助理的客户的 ID (CID):

```
select CID from CUSTOMER where XMLEXISTS
('$d/customerinfo/assistant' passing INFO as "d")
```

这里，仅从含有助理元素的文档中返回 CID。

2. 下面的 SELECT 语句返回其地址国家为“加拿大”而且其城市为“多伦多”的所有客户:

```
select XMLQUERY( '$d/*:customerinfo/*:name' passing INFO
as "d") from CUSTOMER where XMLEXISTS
('$x/*:customerinfo/*:addr[@country ="Canada" and
*:city="Toronto"]' passing INFO as "x");
```

3. 我们现在利用一个 <PurchaseOrder> 元素标签和 4 个子元素标签 (poid, status, custid 和 orderdate) 构建一份 XML 文档。用于该文档的值可以从 POID 是 5001 的 PURCHASEORDER 表中获得。

```
select XMLELEMENT (NAME "PurchaseOrder",
XMLELEMENT (NAME "poid", POID),
XMLELEMENT (NAME "status", STATUS),
XMLELEMENT (NAME "custid", CUSTID),
XMLELEMENT (NAME "orderdate", ORDERDATE))
from PURCHASEORDER where POID = 5001
```

4. SQL/XML 函数 XMLAGG 能够把特定的值聚集到一组。下面的 SELECT 语句返回具有父元素<Orders>的 XML 片断，其中含有来自表 PURCHASEORDER（作为子元素）的全部 POID:

```
select XMLELEMENT (NAME "Orders",
XMLAGG (XMLELEMENT (NAME "poid", POID))) from PURCHASEORDER
```

5. XMLAGG 函数通常与 SELECT 语句的 GROUP BY 从句一起使用，例如:

```
select XMLELEMENT (NAME "Orders",
XMLATTRIBUTES (STATUS as "status"),
XMLAGG (XMLELEMENT (NAME "poid", POID)))
from PURCHASEORDER group by STATUS
```

上面的 SELECT 语句按照采购单的状态对结果进行分组，这能够帮助我们注意到存在重复的行。

我们还可以利用 XMLNAMESPACES 函数在 XML 文档中构造新的名称空间。

6. 例如，下面的查询能够返回含有名称空间 <http://posample.org> 以及子元素 <item> 的新元素节点，并包含来自 PRODUCT 表的名称。

```
select XMLELEMENT (NAME "allProducts",
XMLNAMESPACES (DEFAULT 'http://posample.org'),
XMLAGG (XMLELEMENT (NAME "item", NAME))) from PRODUCT
```

8. XMLTABLE 函数

XMLTABLE 函数是最常用的 SQL/XML 函数之一，因为它有助于从 XML 数据生成关系表。该函数用于帮助从 XML 数据中创建视图。当 XML 文档中的某些部分需要被显示为关系数据时，它非常有用。例如，它能够帮助报告设计人员编写对关系视图的查询，而无需担心 XML 数据模型。

1. 下面的 SELECT 语句返回一个关系表，其中含有两列（NAME 为 varchar(30)，ADDRESS 为 varchar(65))，并把地址的所有元素连接为一个单一项目:

```
select X.* from
XMLTABLE ('db2-fn:xmlcolumn ("CUSTOMER.INFO")/customerinfo'
COLUMNS name varchar(30) PATH 'name', address
varchar(65) PATH 'fn:string-join(addr/*," ")') as X;
```

XMLTABLE 函数的语法是一目了然的。它把 XQuery 或 XPath 表达式作为输入，并用 XPath 表达式和 PATH 从句的值填充命名的关系列。

注：要确保从路径表达式中产生的值总能够生成原子值，以便成功地把这些值转换为关系数据类型。对于产生多个值的 XPath 表达式而言，这些值可以被存储为关系数据库中 XML 列的一部分。

2. 我们现在利用客户电话号码检索含有存储了客户名称的列的表数据以及含有 XML 文件的 XML 列：

```
select X.* from
XMLTABLE ('db2-fn:xmlcolumn("CUSTOMER.INFO")/customerinfo'
COLUMNS name varchar(30) PATH 'name',
phone xml PATH 'for $x in phone return $x') as X;
```

请注意，如果同一个 XML 文档中有一个以上电话元素，那么，它们都将出现在相同的 XML 列值中。

XMLTABLE 函数也可以通过 SELECT 语句以及 INSERT 语句来填充另一个关系表。

3. 例如，我们可以使用下面的 SQL 语句利用给定的模式首先创建一个名为 CUSTOMERDATA 的表：

```
create table CUSTOMERDATA (CID integer, NAME varchar(30),
CITY varchar(20), COUNTRY varchar(20));
```

4. 我们然后使用 INSERT 语句利用 XMLTABLE 函数的结果集合填充改变，如下所示：

```
insert into CUSTOMERDATA
select X.* from CUSTOMER,
XMLTABLE ('$d/customerinfo' passing INFO as "d" COLUMNS
cid integer PATH '@Cid', name varchar(30) PATH 'name',
city varchar(20) PATH 'addr/city', country varchar(20)
PATH 'addr/@country') as X;
```

5. 最后，我们可以通过下面的查询来检查运行结果：

```
select * from CUSTOMERDATA
```



© Copyright IBM Corporation 2011
All Rights Reserved.

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

IBM, IBM (logo), DB2 are trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States, other countries, or both

VMware is a trademark of VMware Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this publication is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of July 2009, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.



IBM DB2® 9.7

**DB2 应用开发
亲自动手实验**

信息管理云计算能力中心

IBM（加拿大）研究院

内容

内容.....	2
1. 前言.....	4
2. 目标.....	4
3. 推荐读物.....	4
4. 设置.....	5
4.1 环境设置要求.....	5
4.2 登录到虚拟机.....	5
5. 开发存储过程、UDF 及触发器.....	6
5.1 创建 SQL PL 存储过程的程序.....	6
5.2 创建 UDF 的程序.....	9
5.3 创建触发器的程序.....	10
6. 利用客户端应用访问 DB2 数据库.....	13
6.1 创建并填充一张表.....	13
6.2 在 IBM Data Studio 中打开应用.....	14
6.3 安装 JDBC 驱动器.....	16
7. 接入 DB2.....	19
7.1 关闭连接.....	20
8. 查询数据.....	21
8.1 把 SELECT 与应用整合到一起.....	23
8.2 利用应用搜索数据库.....	23
9. 插入数据.....	26
9.1 把 INSERT 与应用整合到一起.....	27
9.2 利用应用把数据插入数据库.....	28

1. 前言

在本实验中，你将创建一个存储过程、一个用户定义的函数以及一个触发器。在本实验的后面部分，你还将利用 JDBC 应用访问 DB2 数据库。

2. 目标

完成本实验之后，你将能够：

- 开发存储过程、UDF（用户定义的函数）以及触发器
- 在 IBM Data Studio 中安装 JDBC 驱动器
- 编写具有下述功能的 Java 代码：
 - 创建与 DB2 之间的连接
 - 正确关闭与 DB2 之间的连接
 - 利用 SELECT 语句查询数据
 - 向数据库中添加新数据

3. 推荐读物

Getting started with DB2 Application Development eBook（DB2 应用开发入门电子书）（第 1、3、4 章）

<https://www.ibm.com/developerworks/wikis/display/db2oncampus/FREE+ebook+-+Getting+started+with+DB2+application+development>

这是一本免费的电子书，能够让你快速了解 DB2 应用开发。

4. 设置

4.1 环境设置要求

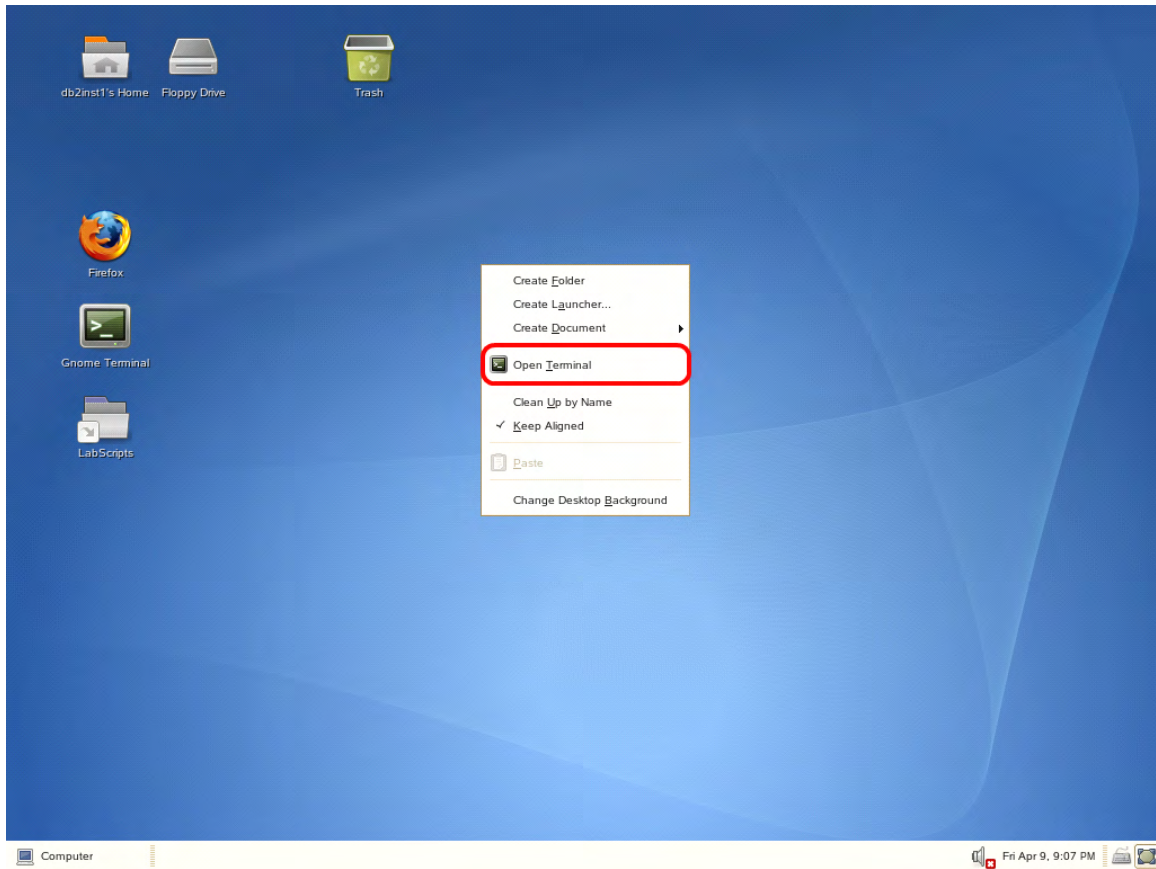
要完成本实验，你需要以下软件：

- DB2 Academic Workshop VMware® 影像（DB2 学术研讨班 VMware® 影像）
- VMware Player 2.x或 VMware Workstation 5.x 或者更高版本

关于如何获得这些组件的帮助信息，请遵循“**VMware 基础和入门**”模块中给出的指示。

4.2 登录到虚拟机

1. 利用下述信息登录到 VMware 虚拟机：
用户名：**db2inst1**
密码：**password**
2. 在命令窗口中键入命令“**startx**”，进入图像环境。
3. 在**桌面**上右击鼠标，选择“**打开终端**”项，即可打开终端窗口。



4. 在终端窗口中键入“**db2start**”，启动 DB2 服务器。

db2start

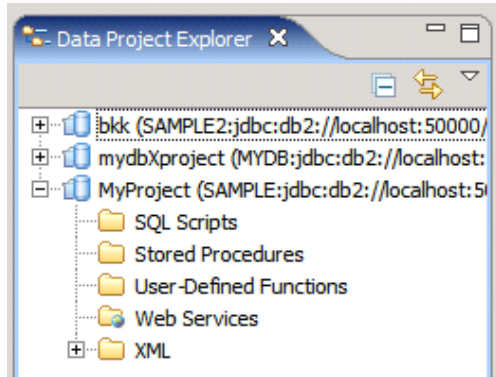
5. 为执行本实验，你需要以其原始格式创建的 DB2 示例数据库。执行下面的命令，删除（如果已经存在的话）并重建 **SAMPLE** 数据库：

```
db2 force applications all
db2 drop db sample
db2sampl
```

5. 开发存储过程、UDF 及触发器

5.1 创建 SQL PL 存储过程的程序

1. 我们来利用 IBM Data Studio 创建 SQL PL 存储过程。IBM Data Studio 在开发存储过程时并非必须的；但是，我们推荐使用此工具，因为它使开发工作更容易。
2. 打开 IBM Data Studio
3. 点击 Data Project Explorer（数据项目浏览器）窗口，然后选择：
File（文件）→ New（新建）→ Project（项目）→ Data Development Project（数据开发项目）
4. 点击 Next（下一步）之后，为你的项目起名字 *MyProject*，然后点击 *Next*（下一步）。
5. 从列表中选择你打算与该项目关联的数据库，然后点击 *Next*（下一步），再然后点击 *Finish*（完成）。如果你遇到一个对话框，它指出你的配置文件尚未完成，请点击 *Edit*（编辑）。选择 *Driver properties*（驱动器属性），确保填写你的 *User name*（用户名）和 *password*（密码）字段，并确保点击 *Test Connection*（测试连接）后连接正常。如果这些都没有问题的话，点击 *OK*（确定）及 *Finish*（完成）。
6. 该数据库名称应该是 SAMPLE。默认值的其他内容都应当是正确的。提供你的用户 ID/密码，然后点击 *Test Connection*（测试连接）。如果连接成功，点击 *Next*（下一步），然后点击 *Finish*（完成）。
7. 在 Data Project Explorer 中，你现在应当能够看到你的项目了。如下所述展开你的项目树，直至列出 SQL 脚本、存储过程等。



8. 右击存储过程文件夹，然后选择 *New* (新建) → *Stored Procedure* (存储过程)
9. 为你的存储过程提供一个名称，例如 *myprocedure*，然后点击 *Next* (下一步)。
10. 在 *SQL Statements* (SQL 语句) 窗口中，IBM Data Studio 为你提供了一个样例语句 *SELECT PROCSCHEMA, PROCNAME FROM SYSCAT.PROCEDURES*。以 *SELECT * FROM EMPLOYEE* 语句替代它，然后点击 *Finish* (完成)。你将得到下面的模板代码。

```
CREATE PROCEDURE MYPROCEDURE ( )
    DYNAMIC RESULT SETS 1
-----
-- SQL Stored Procedure
-----
P1: BEGIN
    -- Declare cursor
    DECLARE cursor1 CURSOR WITH RETURN FOR
        SELECT *
            FROM EMPLOYEE;

    -- Cursor left open for client application
    OPEN cursor1;
END P1
```

11. 你可以把 IBM Data Studio 提供的代码用作你编写自己的代码的模板。这段代码声明了一个游标，然后在结束时打开了这个游标，这意味着该

游标将被返回给调用者。在不加修改的情况下，我们使用该存储过程进行一次演示。

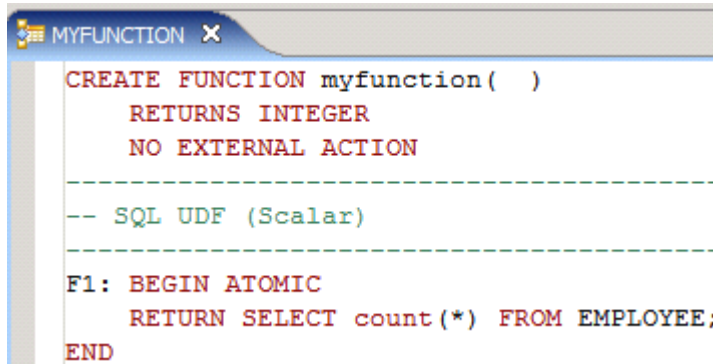
12. 在 Database Explorer 面板中，展开树结构以显示你的连接、数据库、模式及存储过程。请注意，存储过程 *myprocedure* 并未显示出来。Database Explorer 显示了数据库中的对象。此时，*myprocedure* 存储过程还未处于数据库中，因为它还没有被部署。为了部署该过程，请在 Data Project Explorer 窗口中右击它，然后选择 *Deploy* (部署)。在 Deploy (部署) 选项窗口中，选择所有的默认值，然后单击 *Finish* (完成)。
13. 如果部署不成功，请检查在部署时收到的错误代码，然后解决之。如果部署成功了，你就可以准备运行该过程了。在 Data Project Explorer 面板中右击该过程，然后选择 *Run* (运行)。在右下角，你将会看到该过程运行的结果。
14. 现在已经成功完成了该过程的代码编写、部署和运行，你就可以从 DB2 CLP 中对它进行测试运行了：

```
db2 => connect to sample
db2 => call myprocedure()
```

5.2 创建 UDF 的程序

1. 创建用户定义的函数 (UDF) 与创建存储过程非常相似。可以在 DB2 Command Window (命令窗口) 或者 Linux 外壳中发出 CREATE FUNCTION 语句，从而创建 UDF。为了简化部署工作，我们建议你使用 IBM Data Studio。
2. 为了从 IBM Data Studio 中开发 UDF，我们将使用在练习存储过程时使用的项目。右击 User-Defined Functions (用户定义的函数) 文件夹，然后单击 *New* (新建) → *User-Defined Function* (用户定义的函数) 菜单项。
3. 为你的函数起名 *myfunction*，然后单击 *Next* (下一步)。
4. 在 SQL 语句或表达式窗口中，IBM Data Studio 将提供一个 SQL 语句作为例子。把该语句修改为：
SELECT count() FROM EMPLOYEE*

5. 在此时点击 *finish* (完成)。将显示出下述模板:



```
CREATE FUNCTION myfunction( )
  RETURNS INTEGER
  NO EXTERNAL ACTION
-----
-- SQL UDF (Scalar)
-----
F1: BEGIN ATOMIC
    RETURN SELECT count(*) FROM EMPLOYEE;
END
```

6. 我们现在来部署并运行该函数。在 Data Project Explorer 面板中, 右击函数 *myfunction*, 选择 *Deploy* (部署)。在 Deploy (部署) 选项窗口中, 接受所有的默认值, 然后点击 *Finish* (完成)。
7. 成功完成部署之后, , 再次右击该函数, 选择 *Run* (运行)。在右下角, 你会看到运行结果。
8. 为了从 DB2 CLP 中运行该函数, 执行下述命令:

```
db2 => connect to sample
db2 => values myfunction()
```

9. 如果不想使用 values 从句, 你也可以发出如下的 SELECT 语句:

```
db2 => connect to sample
db2 => select myfunction() from sysibm.sysdummy1
```

SYSIBM.SYSDUMMY1 是 DB2 中提供的一个表, 它包含了一行、一列。它可以用于以下场合: 你从 SELECT 语句中调用一个函数, 但你实际上并不需要访问任何表。由于 SELECT 语句需要把某个表用作其语法的一部分, 我们就必须提供一个表。当然, 你也可以为此目的创建你自己的“哑”表。

5.3 创建触发器的程序

1. 你也可以通过在 DB2 CLP 中发出 CREATE TRIGGER 语句创建触发器。如果你打算使用 IBM Data Studio, 你就必须以脚本的方式创建触

发器。在 Data Studio 中没有为开发触发器规定特定的目录。控制中心 (Control Center) 可以被用来创建触发器，但并不鼓励这样做。利用下面的命令从命令窗口或 Linux 外壳中启动控制中心：

```
db2cc
```

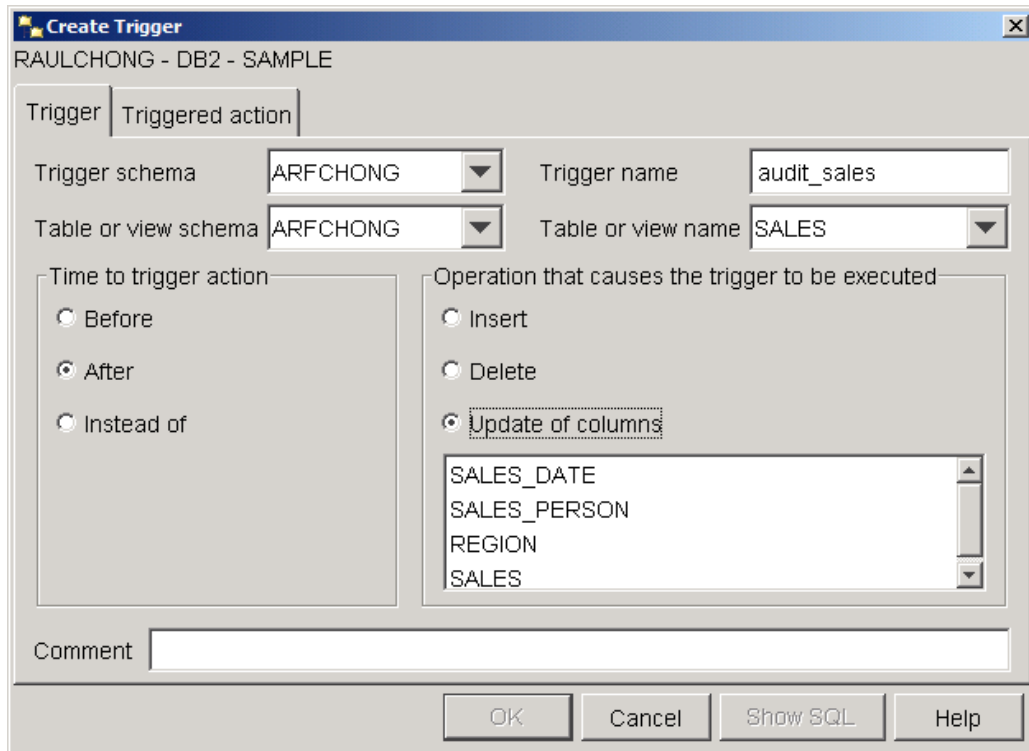
2. 在此使用触发器的例子中，你需要另外创建一张用作日志记录的表。方法如下：

```
db2 => create table saleslog (userid varchar(128) not null,  
daytime timestamp not null)
```

3. 在控制中心中，展开 SAMPLE 数据库文件夹。右击触发器文件夹并选择 *Create* (创建) 选项。将打开 Create Trigger (创建触发器) 对话框。
4. 在对话框中填写下列信息：

触发器属性	值
Trigger Schema (触发器模式)	你登录时的用户身份的用户 ID (应当是默认设置)
Trigger Name (触发器名称)	audit_sales
Table/View Schema (表/视图模式)	你登录时的用户身份的用户 ID (应当是默认设置)
Table/View Name (表/视图名称)	Sales
Time to trigger action (触发行动的时间)	After
Operation that causes the trigger to be executed (导致执行触发器的操作)	更新列时 (不要指定任何列，因为我们希望该触发器在任何列被更新时都启动)。
Comment (注释)	在 Sales 表中记录所有的更新行动。

5. 下图示出了需要设置的相应值：

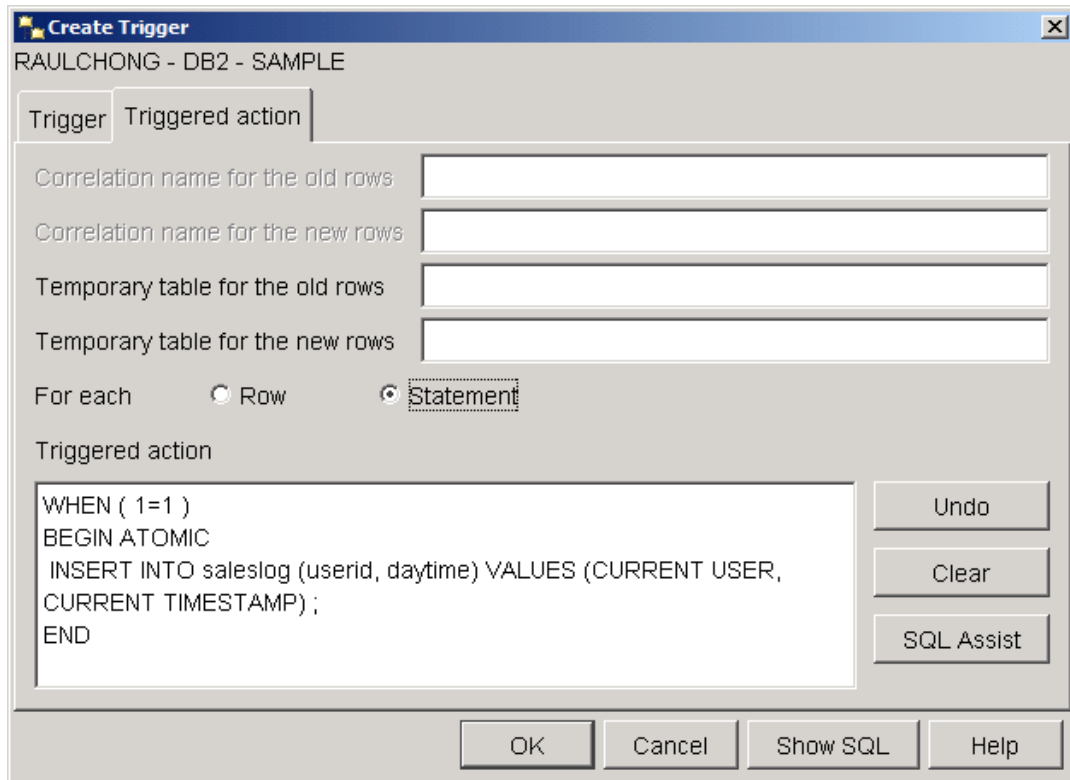


6. 在 Triggered action（触发的行动）标签中，选择 For Each STATEMENT（对每个语句）选项。对触发的行动使用下述代码：

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

注：

在激活触发器的语句执行完毕之后，语句触发器就会启动。这是 FOR EACH STATEMENT（对每个语句）选项的含义。行触发器则规定：每当触发的 SQL 语句影响到任一行，就执行被触发的行动。这是由 FOR EACH ROW（对每一行）语句规定的。



点击 OK（确定）按钮创建触发器。

7. 现在你就能够在控制中心的 Triggers（触发器）文件夹中看到此触发器了。
8. 查询 saleslog 表，确保其中没有数据。使用 DELETE FROM saleslog 命令删除其中可能存在的任何行。

```
db2 => delete from saleslog
```

9. 在 sales 表中更新一条记录，以测试触发器：

```
db2 => update sales set sales_date = current date where year (sales_date) = 1996
```

10. 检查 saleslog 表的内容。现在里面有几行？

6. 利用客户端应用访问 DB2 数据库

在本部分实验中，你将研究一个 JDBC 应用，它能够访问 DB2 数据库。首先，你需要为本实验创建并填充一张表，并在 Data Studio 中配置 JDBC。

6.1 创建并填充一张表

我们来创建一张简单的表，它将在本实验过程中被更新。该表名为“ESQLEMPLOYEE”，其中将填充 1 行数据。

1. 进入存放脚本文件的目录。

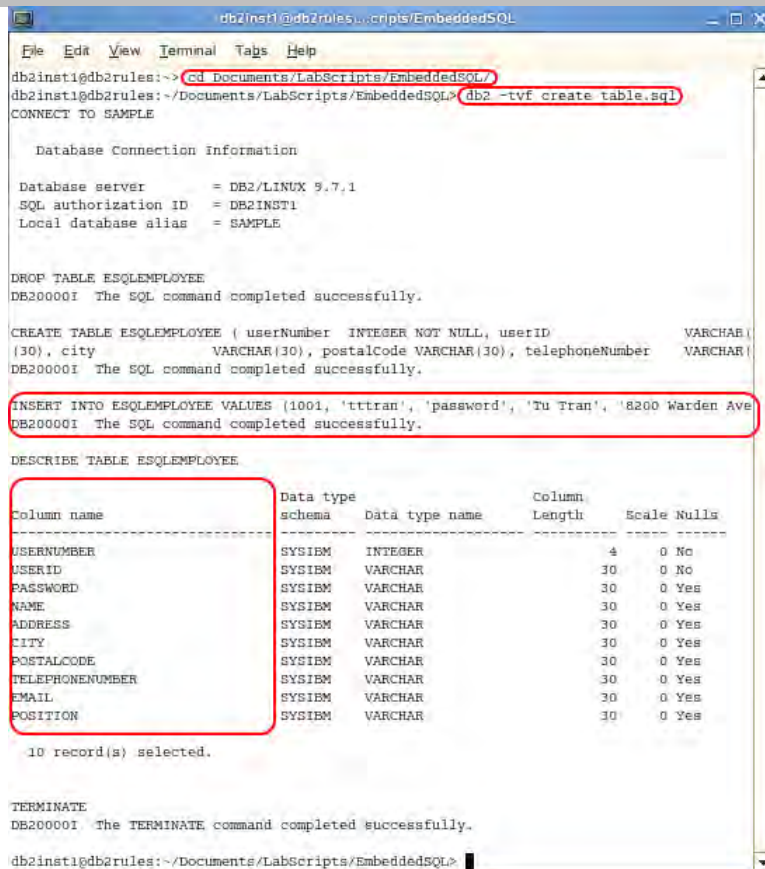
```
cd /home/db2inst1/Documents/LabScripts/EmbeddedSQL
```

2. 首先利用下面的命令查看这个简单的查询：

```
cat create_table.sql
```

3. 为了运行查询，在终端窗口中输入：

```
db2 -tvf create_table.sql
```



```
db2inst1@db2rules:~/Documents/LabScripts/EmbeddedSQL
db2inst1@db2rules:~/Documents/LabScripts/EmbeddedSQL> db2 -tvf create_table.sql
CONNECT TO SAMPLE

Database Connection Information

Database server          = DB2/LINUX 9.7.1
SQL authorization ID    = DB2INST1
Local database alias    = SAMPLE

DROP TABLE ESQLEMPLOYEE
DB20000I The SQL command completed successfully.

CREATE TABLE ESQLEMPLOYEE ( userNumber INTEGER NOT NULL, userID          VARCHAR(
(30), city          VARCHAR(30), postalCode VARCHAR(30), telephoneNumber  VARCHAR(
DB20000I The SQL command completed successfully.

INSERT INTO ESQLEMPLOYEE VALUES (1001, 'tttran', 'password', 'Tu Tran', '8200 Warden Ave
DB20000I The SQL command completed successfully.

DESCRIBE TABLE ESQLEMPLOYEE

Column name          Data type          Column
-----
schema              Data type name    Length  Scale Nulls
-----
USERNUMBER          SYSIBM            INTEGER          4      0 No
USERID              SYSIBM            VARCHAR          30     0 No
PASSWORD            SYSIBM            VARCHAR          30     0 Yes
NAME                SYSIBM            VARCHAR          30     0 Yes
ADDRESS             SYSIBM            VARCHAR          30     0 Yes
CITY                SYSIBM            VARCHAR          30     0 Yes
POSTALCODE          SYSIBM            VARCHAR          30     0 Yes
TELEPHONENUMBER    SYSIBM            VARCHAR          30     0 Yes
EMAIL               SYSIBM            VARCHAR          30     0 Yes
POSITION            SYSIBM            VARCHAR          30     0 Yes

10 record(s) selected.

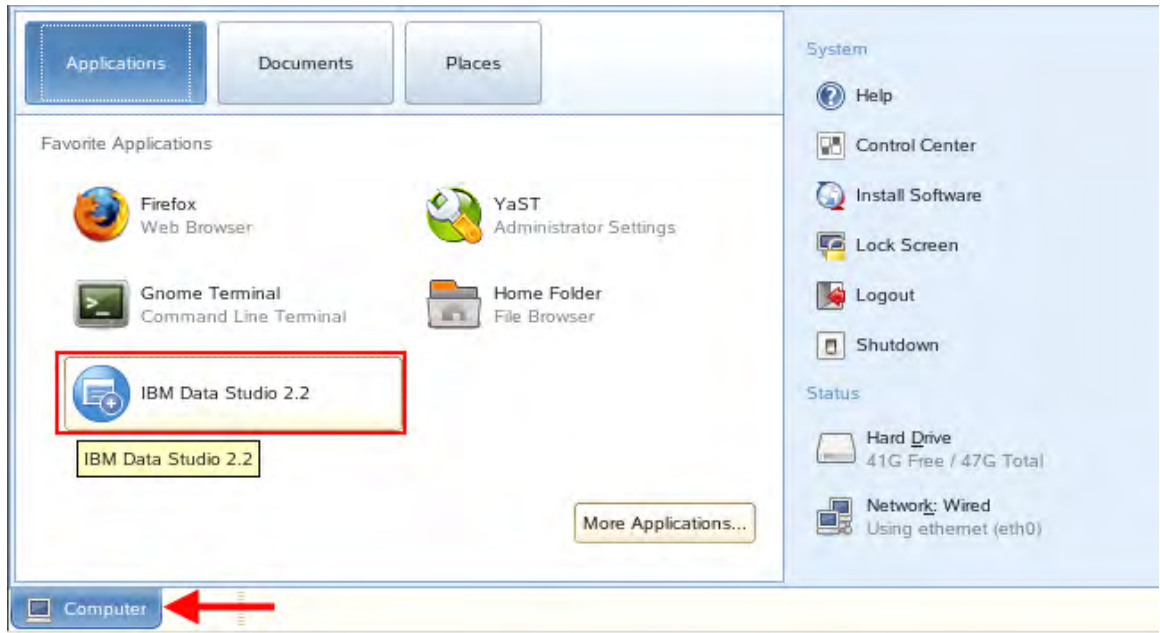
TERMINATE
DB20000I The TERMINATE command completed successfully.

db2inst1@db2rules:~/Documents/LabScripts/EmbeddedSQL>
```

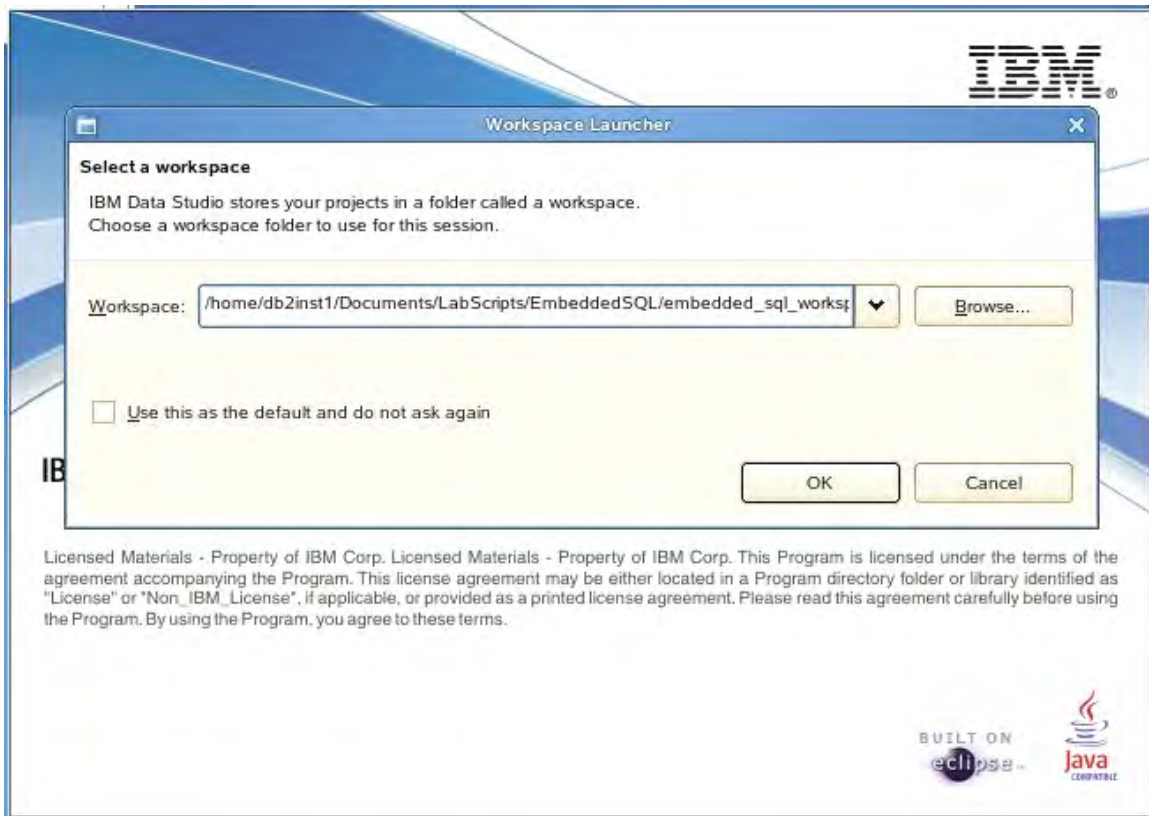
6.2 在 IBM Data Studio 中打开应用

现在数据库已经准备好了，而且也已经创建了 ESQLEMPLOYEE 表，我们需要在 IBM Data Studio 中打开将要使用的应用。打开以后，我们就可以为该应用配置 DB2 JDBC 驱动器。

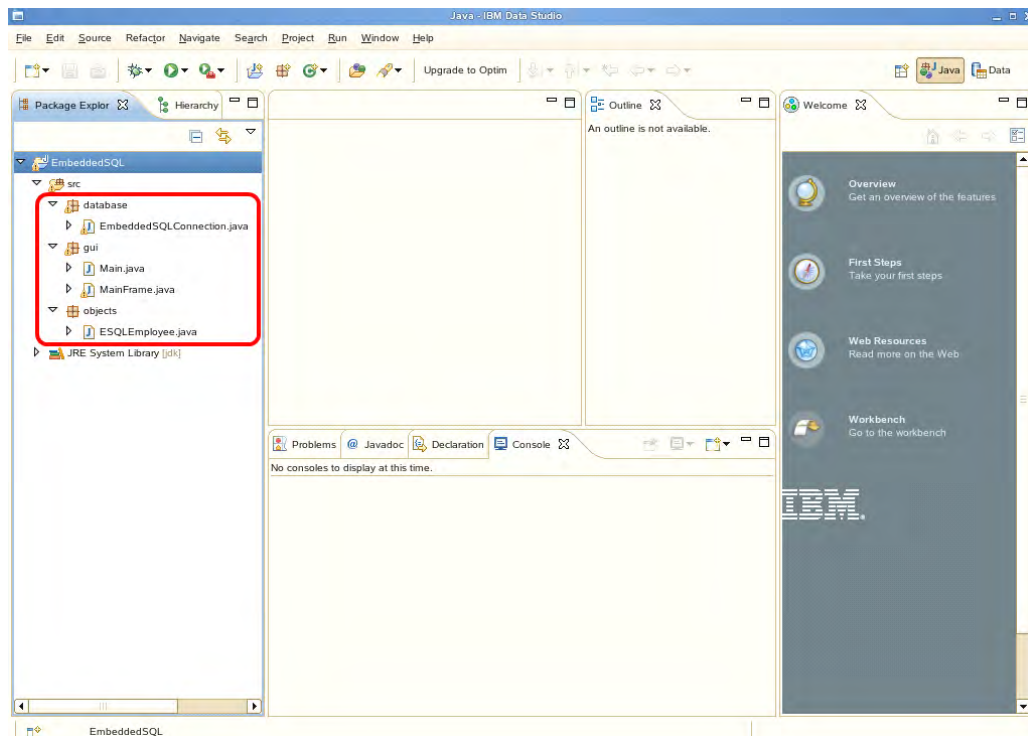
1. 点击 **Computer**（计算机），选择 **IBM Data Studio 2.2**，从而打开 IBM Data Studio:



2. 将出现选择工作站的提示。输入
“/home/db2inst1/Documents/LabScripts/EmbeddedSQL/embedded_sql_
workspace” 作为工作站并选择 **OK**（确定）。



3. 下面的屏幕将显示出来:

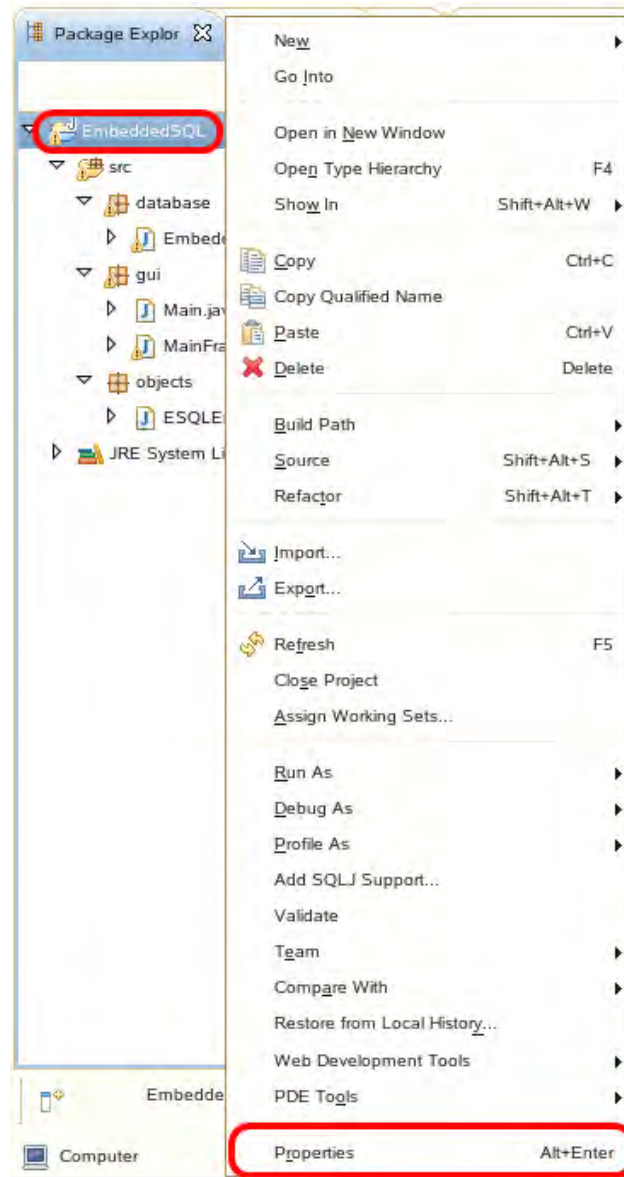


我们已经成功地在 IBM Data Studio 中打开了我们的应用；但是，它还没有准备好接入 DB2。为了连接 DB2，我们首先必须在我们的项目中安装 JDBC 驱动器。

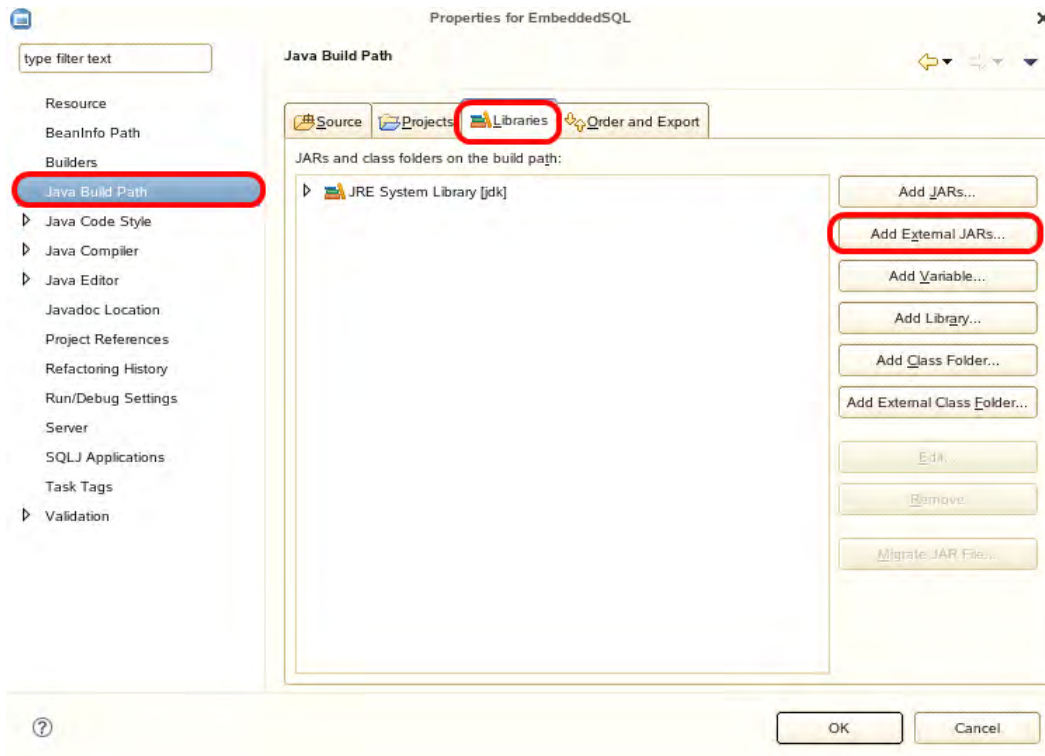
6.3 安装 JDBC 驱动器

JDBC 驱动器能够让 Java 应用接入兼容 SQL 的数据库，发送 SQL 语句，并处理返回的消息和数据。

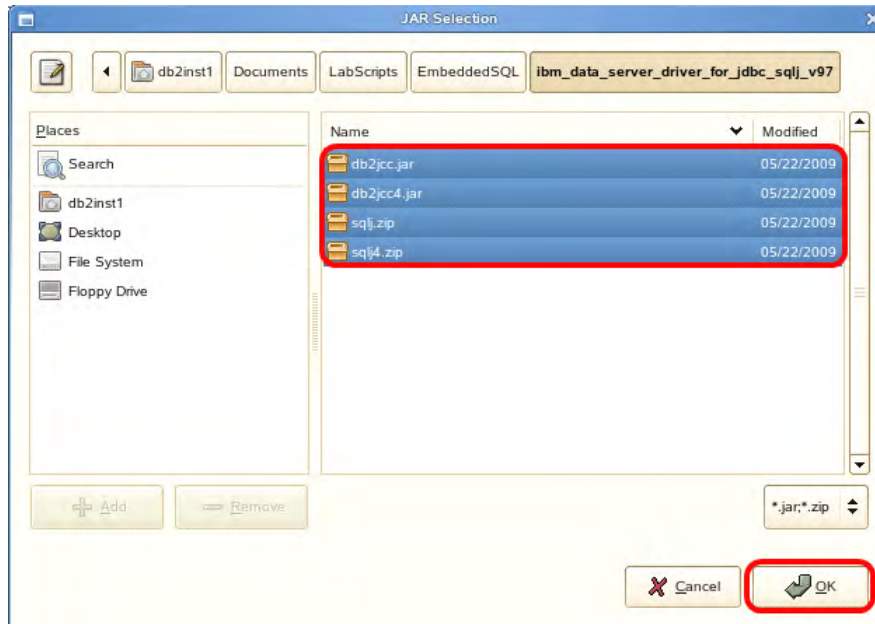
1. 在 IBM Data Studio 打开的情况下，右击 **EmbeddedSQL** 并选择 **Properties**（属性）。



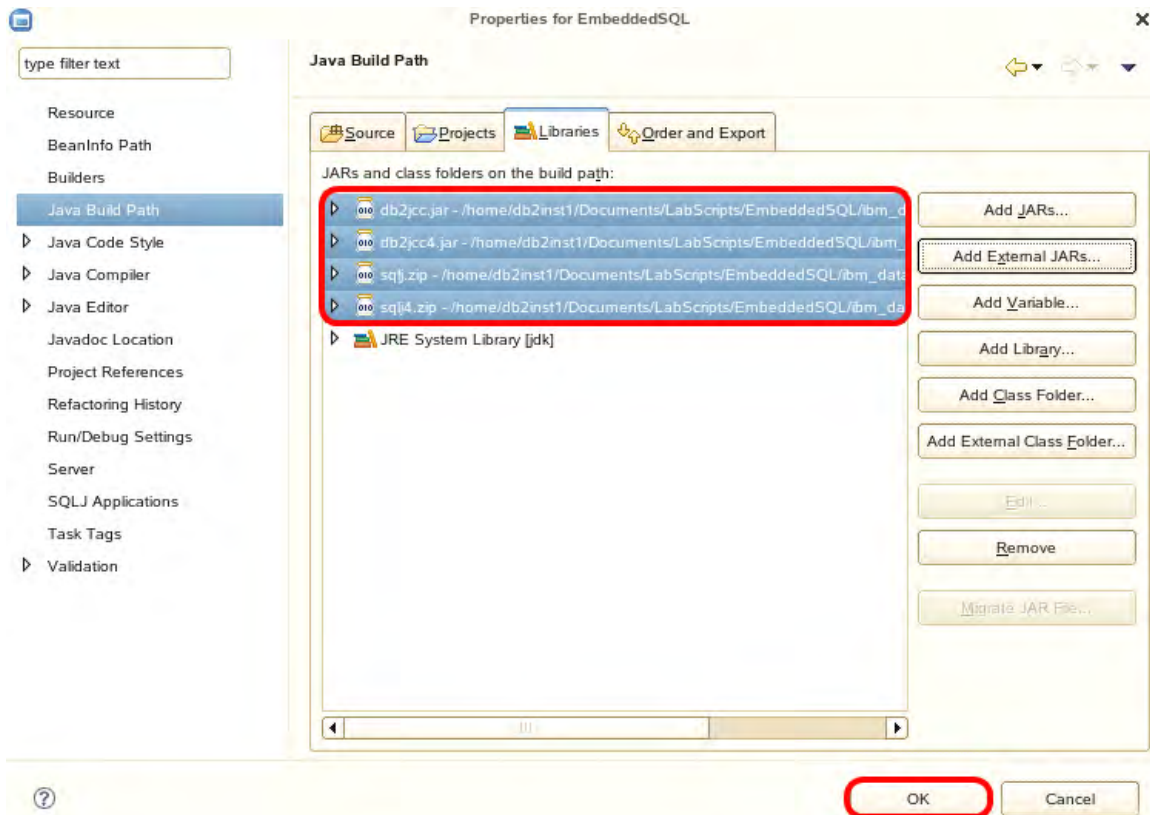
2. 从列表中选择 **Java Build Path**（**Java 构建路径**）。然后选择 **Libraries**（**库**）选项卡并点击 **Add External JARs**（**添加额外的 JAR**）按钮。



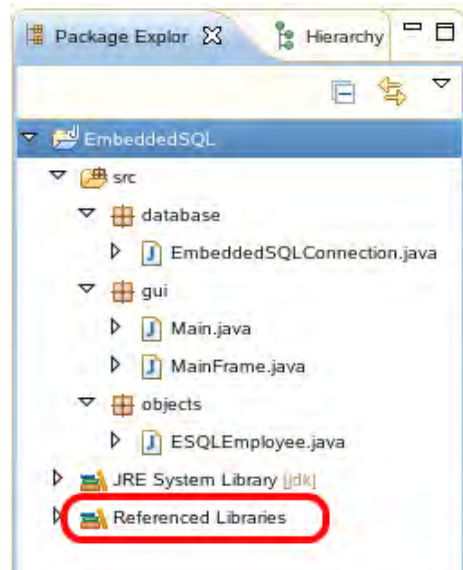
3. 浏览至
“/home/db2inst1/Documents/LabScripts/EmbeddedSQL/ibm_data_server_driver_for_jdbc_sqlj_v97” 并选择该文件夹中的所有文件。



4. 被选中的文件出现在 **Libraries** (库) 选项卡中。选择 **OK** (确定) 继续。



5. **JDBC 驱动器**现在已成功安装。你现在可以查看被添加的库，方法是在 **Package Explorer**（程序包浏览器）中选择 **Referenced Libraries**（被引用的库）。

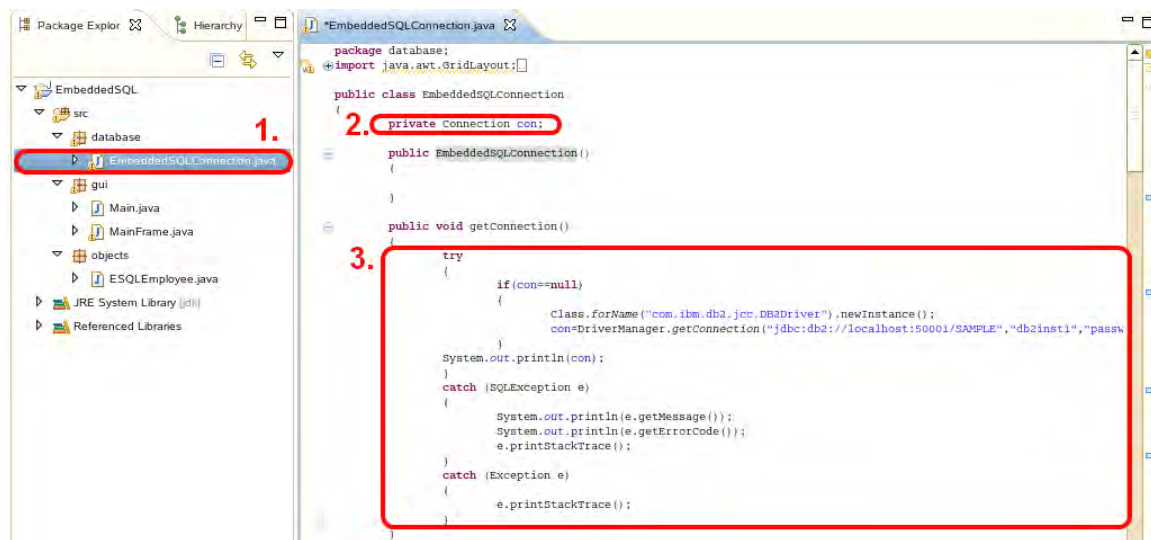


现在，JDBC 驱动器已经正确安装到我们的开发环境中，我们可以在我们的应用中添加代码以便接入 DB2。

7. 接入 DB2

任何时候当我们希望与某个数据库交互时，我们都必须首先与该数据库服务器建立连接。

1. 在数据库程序包中，打开 **EmbeddedSQLConnection** 类。选择 **EmbeddedSQL** 项目并按 **F5** 以打开该类。



EmbeddedSQLConnection 类是本练习中最重要的类。该类包含了与数据访问相关的应用中的所有函数。

2. 为了创建连接，首先必须声明一个 **Connection** 类型的变量，以用于容纳 **Connection** 对象。

```
private Connection con;
```

现在我们可以利用 **DriverManager** 对象创建与 **DB2** 服务器之间的连接，最后我们把该连接存储在“con”变量中。

3. 取消 **getConnection()** 中所提供的代码的注释，完成 **getConnection()** 函数。

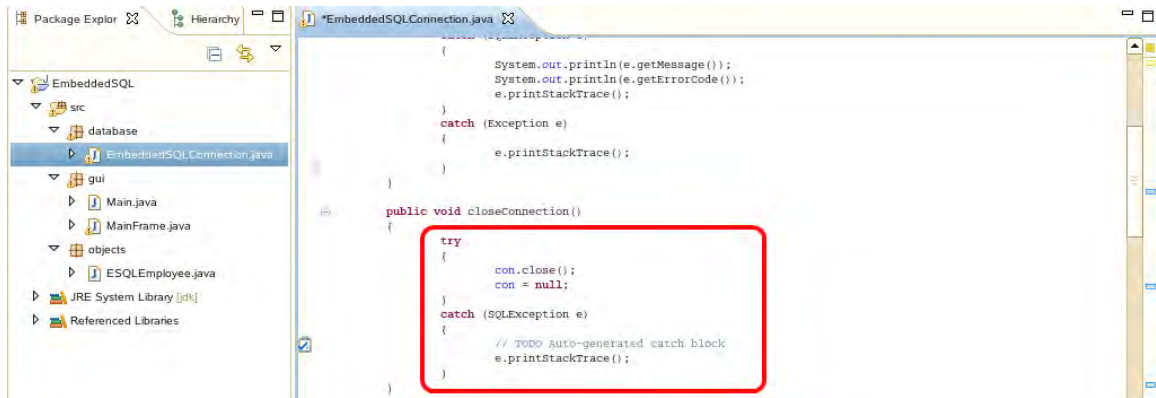
```
con=DriverManager.getConnection("jdbc:db2://localhost:50001/SAMPLE","db2inst1","password");
```

getConnection()方法试图与给定的数据库 **URL** 建立连接。我们利用用户名“db2inst1”以及密码“password”通过 **JDBC API** 在端口 **50001** 上与 **DB2** 的 **SAMPLE** 数据库建立连接。

7.1 关闭连接

我们已经知道了如何建立连接，我们还需要知道当不再需要连接时如何正确地关闭与 **DB2** 之间的连接。这是很重要的一步，因为它将能够为你的应用及数据库服务器释放系统资源。

1. 在 **EmbeddedSQLConnection** 类中，取消 **closeConnection()**中所提供的代码的注释，完成 **closeConnection()**函数。



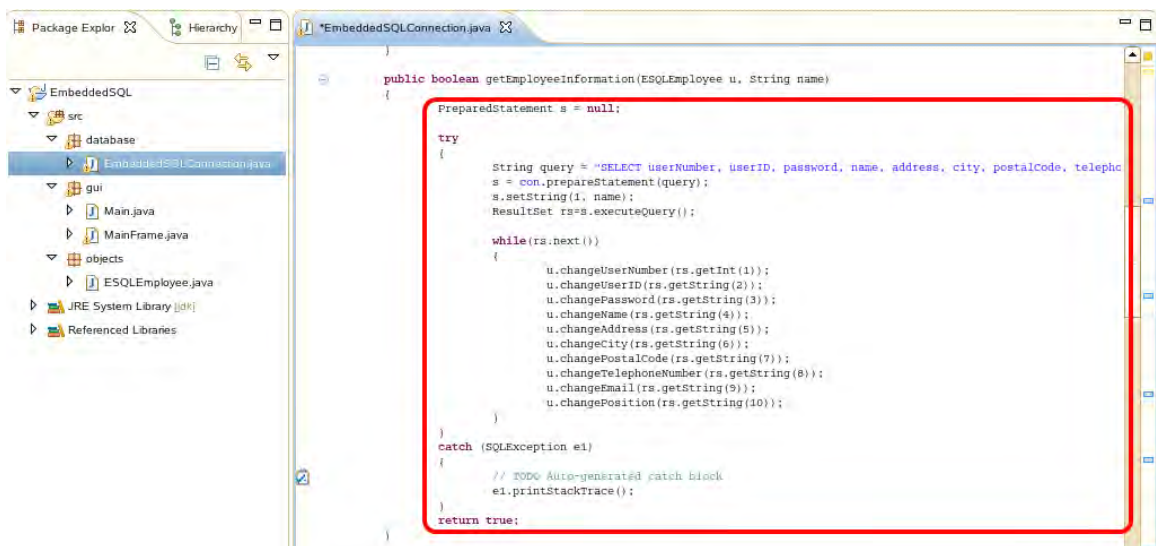
`con.close()`

`close()`方法用于终止与上述 `getConnection()`方法所规定的数据库之间的连接。连接被终止之后，我们还可以设置 `con = null`。

8. 查询数据

我们在前面学习了如何创建一些函数来建立及关闭与 DB2 之间的连接，下面我们将编写一条查询来检索及显示数据。

1. 在 `EmbeddedSQLConnection` 类中，取消 `getEmployeeInformation()` 中所提供的代码的注释，从而完成 `getEmployeeInformation()` 函数。



`getEmployeeInformation()` 函数根据所提供的姓名查询员工信息。应用利用它来检索数据库，以获得具体员工的信息。

2. 为了创建和执行查询，必须首先规定一个类型为 `PreparedStatement` 的对象。

```
PreparedStatement s = null;
```

`PreparedStatement` 对象 “s” 将被用来容纳 SQL `SELECT` 语句。

3. SQL 语句本身被编码为一条字符串。

```
String query = "SELECT userNumber, userID, password, name, address, city, postalCode, telephoneNumber, email, position FROM ESQLEMPLOYEE WHERE name = ?";
```

4. 我们现在可以利用 `Connection` 对象 `con` 以及 `prepareStatement()` 方法来创建 `PreparedStatement`。所产生的对象存储在变量 “s” 中。

```
s = con.prepareStatement(query);
```

5. 请仔细看看查询串，注意到 “?” 了吗？这被称为参数标记。它标记出在运行时刻将插入某个值的地方。在本例中，就是用户在执行该应用时所提供的检索标准。

下面的命令用于把一个值与参数标记关联起来。

```
s.setString(1, name);
```

例如，假如我们打算检索名为 `Tu Tran` 的员工，该串查询就变为 `"SELECT userNumber, userID, password, name, address, city, postalCode, telephoneNumber, email, position FROM ESQLEMPLOYEE WHERE name = Tu Tran"`;

6. 在我们执行查询时，必须把查询所返回的结果保存起来。我们把这些数据存储在类型为 `ResultSet` 的对象中。

```
ResultSet rs=s.executeQuery();
```

7. 最后，我们可以检索存储在 `ResultSet` 中的数据。

```
while(rs.next())
{
    u.changeUserNumber(rs.getInt(1));
    u.changeUserID(rs.getString(2));
    u.changePassword(rs.getString(3));
    u.changeName(rs.getString(4));
}
```

```
u.changeAddress(rs.getString(5));

u.changeCity(rs.getString(6));

u.changePostalCode(rs.getString(7));

u.changeTelephoneNumber(rs.getString(8));

u.changeEmail(rs.getString(9));

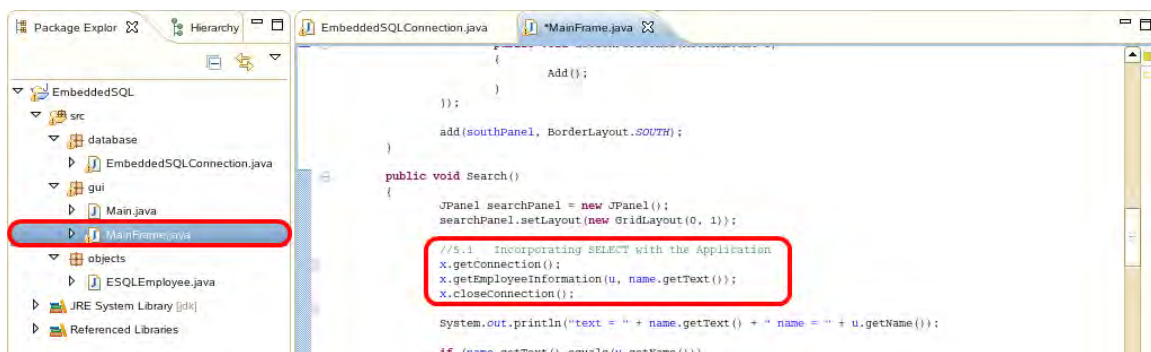
u.changePosition(rs.getString(10));

}
```

8.1 把 SELECT 与应用整合到一起

我们已经创建了一些函数用来建立及关闭与 DB2 之间的连接，并且利用 SELECT 语句返回数据。我们如何在我们的应用中使用这些函数呢？

1. 在 GUI 程序包中，打开 MainFrame 类。




MainFrame 类是我们将利用所创建的函数与 DB2 进行交互的地方。该类中包含了用于让用户与应用进行交互的所有函数。

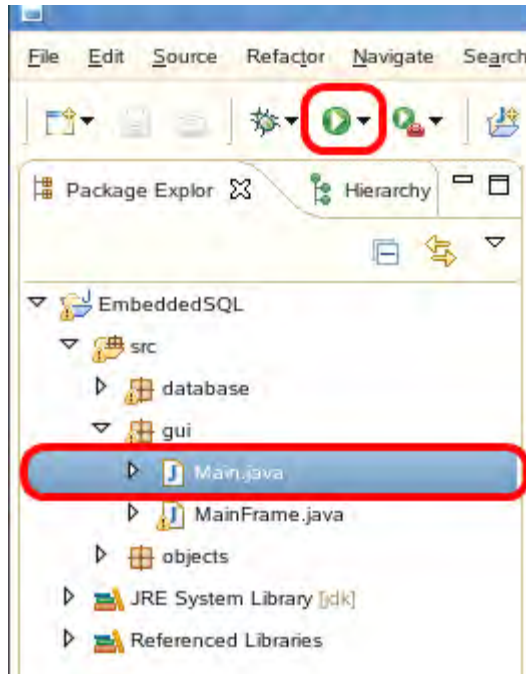
2. 进入 Search() 函数，取消所提供的代码的注释。

```
//5.1 Incorporating SELECT with the Application
x.getConnection();
x.getEmployeeInformation(u, name.getText());
x.closeConnection();
```

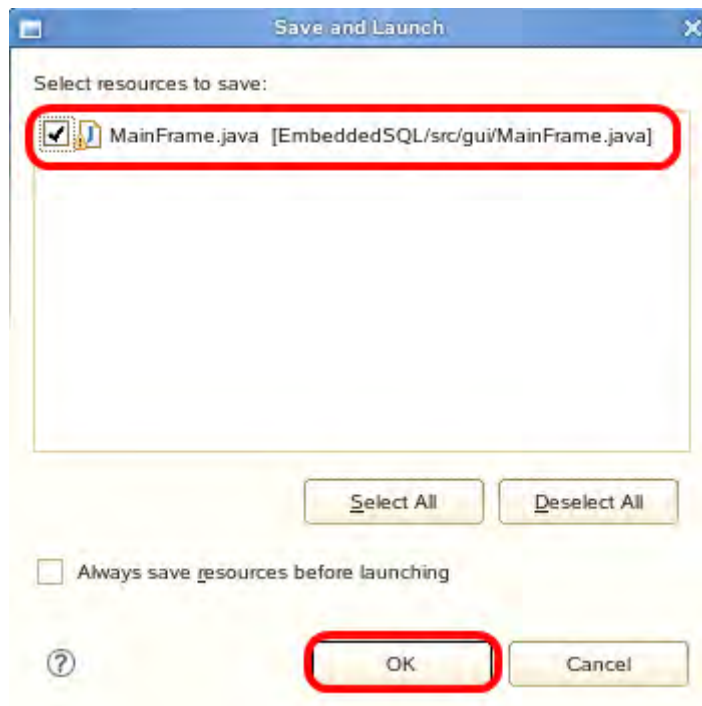
8.2 利用应用搜索数据库

该应用现在能够对数据库执行 SELECT 语句了，而且能够显示所返回的信息。

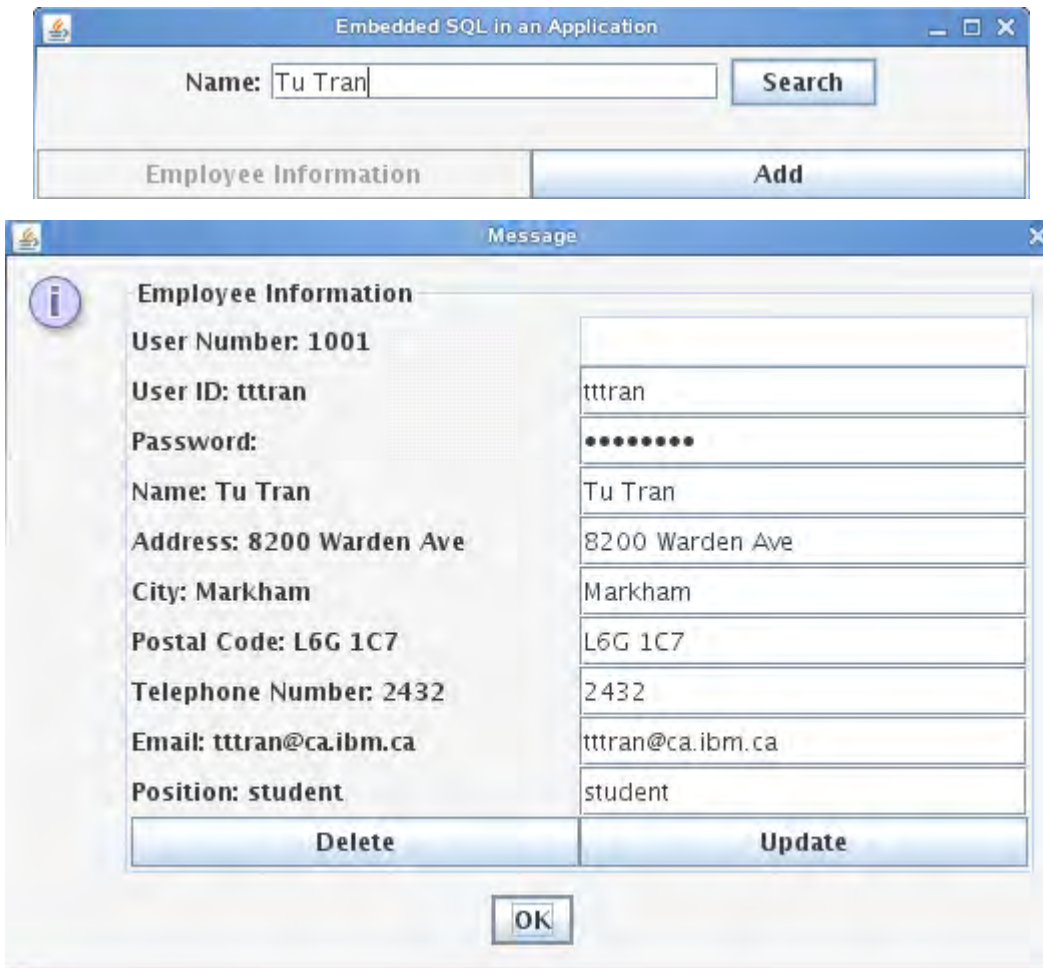
1. 在 gui 程序包中，打开 Main 类并点击 .



2. 如果你尚未保存你的修改，你将会被提示保存文件。选择所有需要保存的资源，然后点击 **OK**（确定）。



3. 下面的程序将显示出来。输入姓名“Tu Tran”并点击 **Search**（检索）。



可以看到，从数据库中返回了如下员工数据。

4. 打开一个终端并接入 **Sample** 数据库，以便查看 **ESQLEMPLOYEE** 表。

```
db2 connect to sample
db2 "SELECT * FROM ESQLEMPLOYEE"
```

```
db2inst1@db2rules:~> db2 connect to sample

Database Connection Information

Database server          = DB2/LINUX 9.7.1
SQL authorization ID    = DB2INST1
Local database alias    = SAMPLE

db2inst1@db2rules:~> db2 "SELECT * FROM ESQLEMPLOYEE"

USERNUMBER  USERID                PASSWORD                NAME
-----
1001 tttran                                     password                Tu Tran

1 record(s) selected.

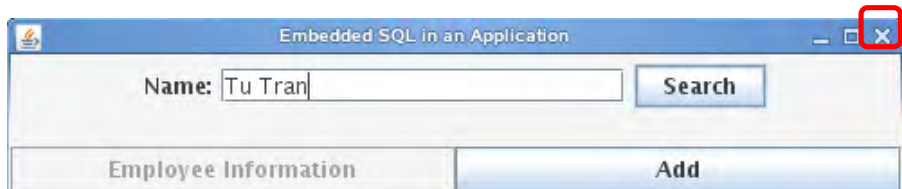
db2inst1@db2rules:~> █
```

可以看到，通过嵌入到 SQL 应用返回的数据与直接接入 Sample 数据库所获得的数据是一样的。

5. 返回 Java 程序，点击“OK”（确定），关闭窗口。



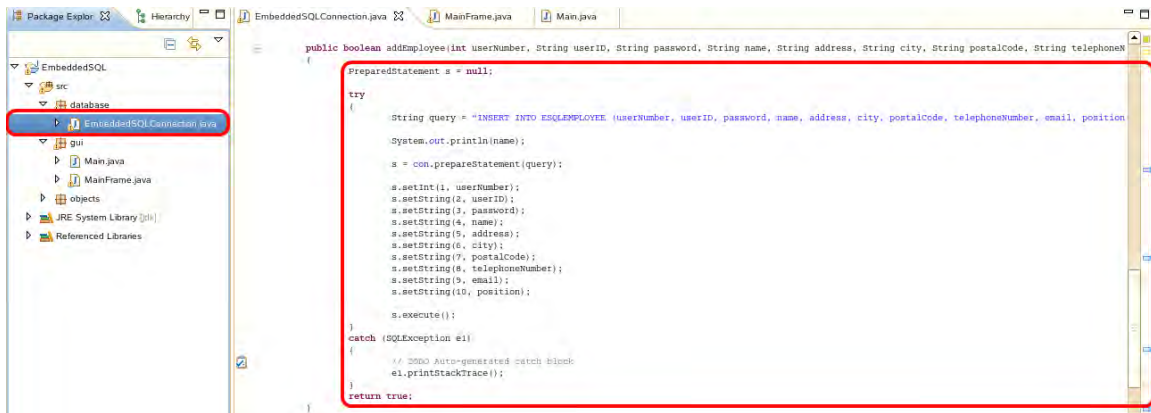
6. 点击“X”关闭应用。



9. 插入数据

INSERT 语句的使用方法与 SELECT 语句很相似，只是我们不需要在 ResultSet 中存储数据了。

1. 在 EmbeddedSQLConnection 类中，取消 addEmployee() 所提供的代码的注释，完成 addEmployee() 函数。



`addEmployee ()` 函数把新的员工信息插入到 `ESQLEMPLOYEE` 表中。

2. 像以前一样，需要规定一个类型为 `PreparedStatement` 的对象以存储 SQL 语句。

```
PreparedStatement s = null;
```

3. 该 SQL 语句被编码为下面的字符串。

```
String query = "INSERT INTO ESQLEMPLOYEE (userNumber, userID, password, name, address, city, postalCode, telephoneNumber, email, position) VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
```

4. 我们现在可以利用 `Connection` 对象 `con` 以及 `prepareStatement()` 方法来创建 `PreparedStatement`。

```
s = con.prepareStatement(query);
```

5. 请注意，`INSERT` 语句中有多个参数标记。像以前一样，它们用来把用户所提供的值与待执行的 SQL 语句关联起来。利用类似下面的命令把某个值与某个参数关联起来。

```
s.setString(2, userID);
```

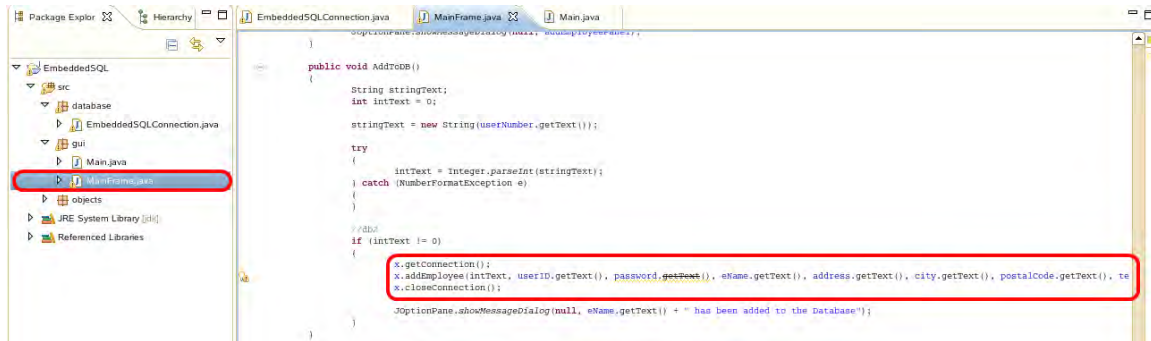
6. 现在可以执行该语句了。

```
s.execute();
```

9.1 把 `INSERT` 与应用整合到一起

我们已经创建了一些函数用来建立及关闭与 `DB2` 之间的连接，并且利用 `INSERT` 语句插入数据。我们如何在我们的应用中使用这些函数呢？

1. 在 GUI 程序包中，打开 `MainFrame` 类。




MainFrame 类是我们将利用所创建的函数与 **DB2** 进行交互的地方。该类中包含了用于让用户与应用进行交互的所有函数。

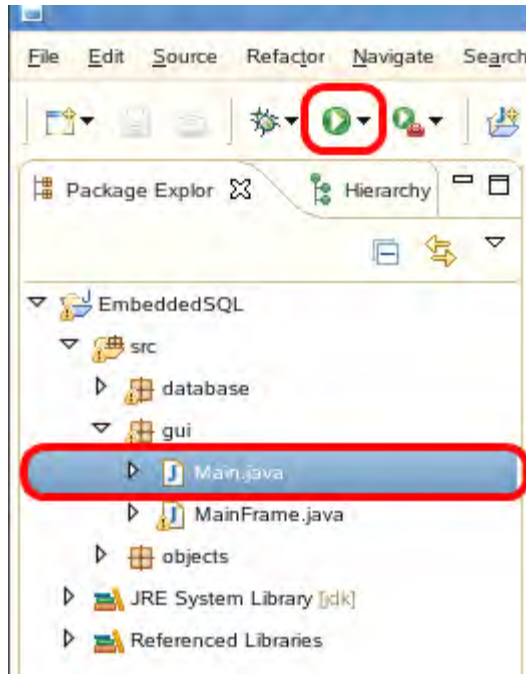
2. 进入 **AddToDB()** 函数，取消所提供的代码的注释。

```
//6.1 Incorporating INSERT with the Application
x.getConnection();
x.addEmployee(intText, userID.getText(), password.getText(),
eName.getText(), address.getText(), city.getText(),
postalCode.getText(), telephoneNumber.getText(), email.getText(),
position.getText());
x.closeConnection();
```

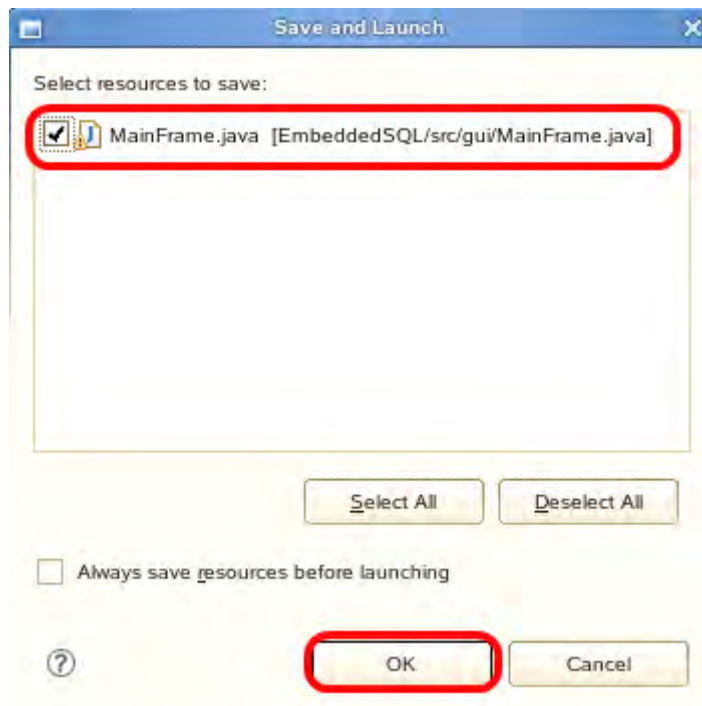
9.2 利用应用把数据插入数据库

该应用现在能够执行 **INSERT** 语句把新的员工插入到数据库中。

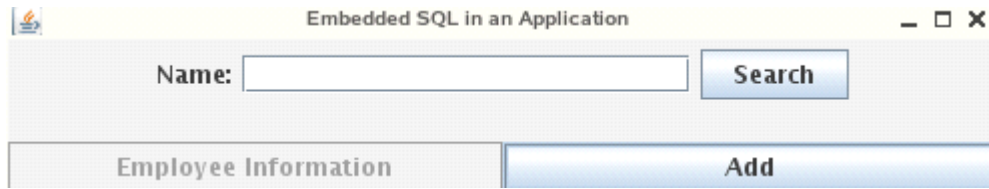
1. 在 **gui** 程序包中，打开 **Main** 类并点击 。



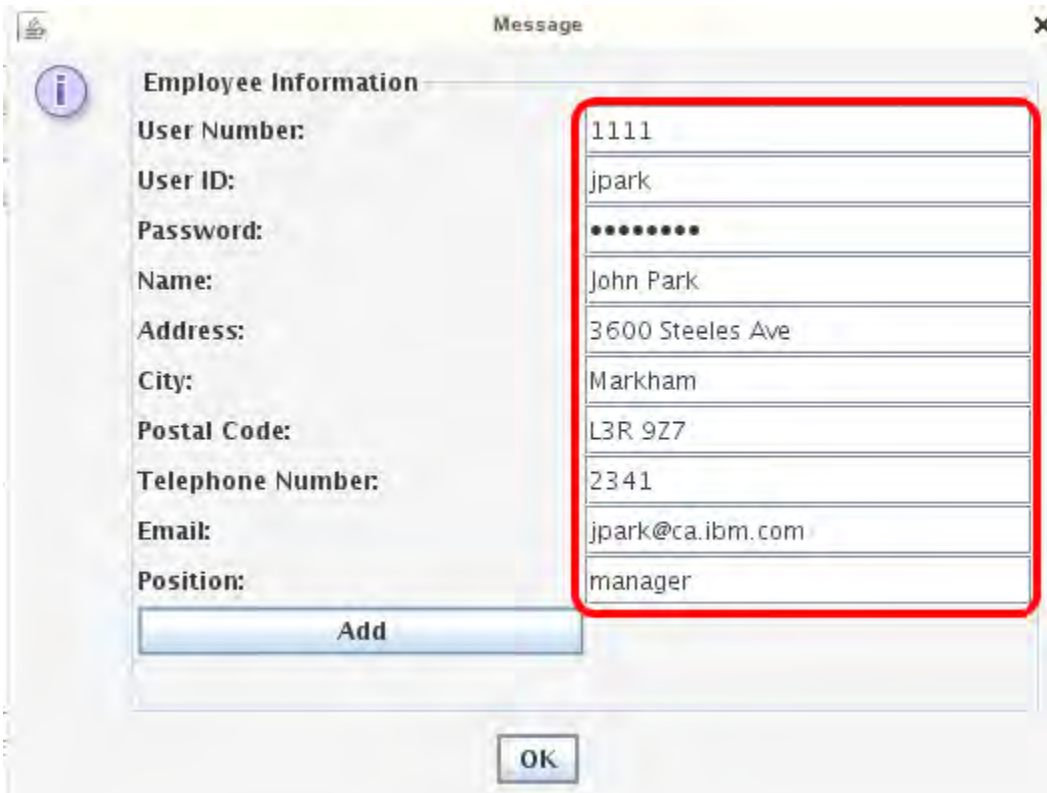
2. 如果你尚未保存你的修改，你将会被提示保存文件。选择所有需要保存的资源，然后点击 **OK**（确定）。



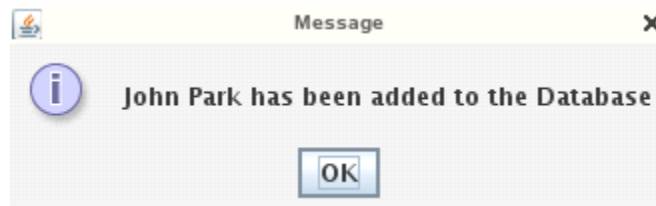
3. 下面的程序将显示出来。点击 **Add**（添加）按钮。



4. 将弹出下面的窗口。输入下面的信息，点击 **Add**（添加）。



可以看到，员工数据被成功添加到数据库中。点击 **“OK”**（确定），关闭弹出的消息。



5. 打开一个终端，接入 **SAMPLE** 数据库，查看 **ESQLEMPLOYEE** 表。

```
db2 connect to sample
db2 "SELECT * FROM ESQLEMPLOYEE"
```

```
File Edit View Terminal Tabs Help
db2inst1@db2rules:~> db2 connect to sample

Database Connection Information

Database server          = DB2/LINUX 9.7.1
SQL authorization ID    = DB2INST1
Local database alias    = SAMPLE

db2inst1@db2rules:~> db2 "SELECT * FROM ESQLEMPLOYEE"

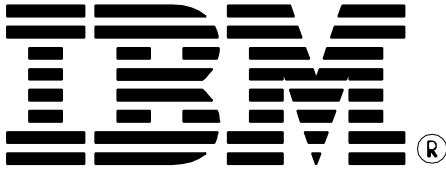
USERNUMBER  USERID          PASSWORD          NAME
-----
          1001 tttran          password          Tu Tran
          1111 jpark          password          John Park

2 record(s) selected.

db2inst1@db2rules:~> █
```

可以看到，雇员“John Park”已经被添加到 Sample 数据库中。

祝贺你！你已经创建了一个能够与 DB2 交互的简单应用。在本练习中，我们学习了如何利用 JDBC API 从数据库中检索数据以及向数据库中插入数据。在该应用中，还有用于 DELETE 和 UPDATE 语句的函数。请仔细阅读这些代码，以便更好地理解我们如何在 Java 应用中使用 JDBC。



© Copyright IBM Corporation 2011
All Rights Reserved.

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

IBM, IBM (logo), and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States, other countries, or both.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this publication is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of June 2010, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.



IBM DB2® 9.7

故障解决
亲自动手实验

信息管理云计算能力中心

IBM（加拿大）研究院

目录

1. 前言	3
2. 目标	3
3. 推荐读物	3
4. 使用 HELP (?) 命令	3
4.1 试试看：练习 HELP (?) 命令.....	4
5. 使用 DB2DIAG.LOG 文件	4
5.1 试试看：利用 DB2DIAG.LOG 文件诊断故障.....	4
6. 使用 DB2 信息中心	5
6.1 试试看：利用 DB2 信息中心获得更多信息.....	5
7. 答案	5
试试看：练习 HELP (?) 命令.....	5
试试看：利用 DB2DIAG.LOG 文件诊断故障.....	5
利用 DB2 信息中心获得更多信息.....	7

1. 前言

在本实验中，你将练习故障解决技巧。

2. 目标

完成本实验之后，你将能够：

- ▶ 使用 DB2 Help (?) 命令
- ▶ 利用 db2diag.log 文件诊断故障
- ▶ 从 DB2 信息中心查找关于某主题的更多信息

3. 推荐读物

Getting started with DB2 Express-C eBook (DB2 应用开发入门电子书) (附录 A)
<https://www.ibm.com/developerworks/wikis/display/DB2/FREE+Book+Getting+Started+with+DB2+Express-C>

这是一本免费的电子书，能够让你快速了解 DB2。

4. 使用 help (?) 命令

在使用 DB2 时，你或许会遇到问题，也或许会返回一个 SQLCODE。利用 DB2 help (?) 命令，你可以找到有关这条 SQLCODE 的更详细信息。例如，对于 SQLCODE -104 及 -204，可以分别试试下面的命令：

对于 DB2 命令窗口或 Linux 外壳，键入：

```
db2inst1@db2rules:~> db2 "? SQL0104N"  
db2inst1@db2rules:~> db2 "? SQL0204N"
```

如果你不记得某条 DB2 命令，或者不记得命令的完整语法，你也可以使用 help (?) 命令。例如，假设你只是部分记得一条命令起始于“LIST”，但你不记得后面是什么了。你可以在 DB2 命令窗口或 Linux 外壳中键入以下命令：

```
db2inst1@db2rules:~> db2 "? LIST"
```

请注意，HELP 命令不能与 SQL 语句一起使用。可以利用 DB2 信息中心了解 SQL 语句的详细信息。

4.1 试试看：练习 help (?) 命令

1. 试寻找关于这些 SQLCODE 的详细信息：-805, +100
2. 利用 help (?) 命令寻找 BACKUP 命令的语法

5. 使用 db2diag.log 文件

db2diag.log 文件是在使用 DB2 服务器过程中遇到问题时可以求助的另一个信息源。请找到该文件的位置并查看一下它的内容。文件的位置在不同的操作系统中有所不同：

- Windows Vista 以及更高版本
 - ▶ ProgramData\IBM\DB2\
- Windows XP/2003
 - ▶ C:\Documents and Settings\All Users\Application Data\IBM\DB2\DB2COPY1\
- Linux/UNIX
 - ▶ INSTHOME/sqlllib/db2dump (INSTHOME 是实例所有者的主目录)

5.1 试试看：利用 db2diag.log 文件诊断故障

1. 把 db2diag.log 改名为 db2diag.old.log。然后，从 DB2 命令窗口或 Linux 外壳中进行如下变更：

```
db2inst1@db2rules:~> db2 "update db cfg for sample using LOCKLIST 4"
db2inst1@db2rules:~> db2 "update db cfg for sample using MAXAPPLS 100"
```

检查 db2diag.log，寻找与上述变更相对应的条目。

2. 从 DB2 命令窗口发出一下命令：

```
db2inst1@db2rules:~> db2 "connect to SAMPLE"
db2inst1@db2rules:~> db2stop
```

检查 db2diag.log。得到了什么错误消息？

3. 本节到此结束。利用下面的命令重置配置参数的变化：

```
db2inst1@db2rules:~> db2 reset db cfg for sample
db2inst1@db2rules:~> db2stop force
db2inst1@db2rules:~> db2start
```

6. 使用 DB2 信息中心

DB2 信息中心包含 DB2 在线手册。要获得它，请访问以下网站：

<http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp>

6.1 试试看：利用 DB2 信息中心获得更多信息

1. 假设你想知道 CREATE TABLE 语句的语法方面的详细信息。请在 DB2 信息中心查找这些信息。

7. 答案

试试看：练习 help (?) 命令

第 4.1 节的答案

1. 试寻找关于这些 SQLCODE 的详细信息：-805, +100

在 DB2 命令窗口或 Linux 外壳中键入以下命令：

```
db2inst1@db2rules:~> db2 "? SQL0805N"
db2inst1@db2rules:~> db2 "? SQL0100W" (or db2 "? SQL0100")
```

利用 help (?) 命令寻找 BACKUP 命令的语法：

```
db2inst1@db2rules:~> db2 "? backup"
```

试试看：利用 **db2diag.log** 文件诊断故障

第 5.1 节的答案

1.

把 **db2diag.log** 改名为 **db2diag.old.log**。然后，从 DB2 命令窗口或 Linux 外壳中进行如下变更：

```
db2inst1@db2rules:~> db2 "update db cfg for sample using LOCKLIST 4"
db2inst1@db2rules:~> db2 "update db cfg for sample using MAXAPPLS 100"
```

这时应当创建了一个新的 **db2diag.log** 文件。查看一下内容。你看到了什么？

这大概是你应当看到的东西。很明显，时间戳应当是不同的。

```
2011-05-07-23.03.38.359000-240 I1097H468          LEVEL: Event
PID       : 2448                TID  : 8048                PROC  : db2syscs.exe
INSTANCE: DB2                  NODE  : 000                DB    : SAMPLE
APPHDL   : 0-588               APPID: *LOCAL.DB2.080907152844
AUTHID   : ARFCHONG
EDUID    : 8048                EDUNAME: db2agent (SAMPLE)
FUNCTION: DB2 UDB, config/install, sqlfLogUpdateCfgParam, probe:20
CHANGE   : CFG DB SAMPLE: "Locklist" From: "50" To: "4"
```

```
2011-05-07-23.04.16.468000-240 I1567H482          LEVEL: Event
PID       : 2448                TID  : 8048                PROC  : db2syscs.exe
INSTANCE: DB2                  NODE  : 000                DB    : SAMPLE
APPHDL   : 0-588               APPID: *LOCAL.DB2.080907152844
AUTHID   : ARFCHONG
EDUID    : 8048                EDUNAME: db2agent (SAMPLE)
FUNCTION: DB2 UDB, config/install, sqlfLogUpdateCfgParam, probe:20
CHANGE   : CFG DB SAMPLE: "Maxappls" From: "40" <automatic> To: "100"
```

2. 从 DB2 命令窗口发出一下命令：

```
db2inst1@db2rules:~> db2 "connect to SAMPLE"
db2inst1@db2rules:~> db2stop
```

检查 **db2diag.log**。得到了什么错误消息？

```
2011-05-04-23.05.18.515000-240 I5006H302          LEVEL: Error
PID       : 2200                TID  : 528                PROC  : db2syscs.exe
INSTANCE: DB2                  NODE  : 000
EDUID    : 528
FUNCTION: DB2 UDB, base sys utilities, DB2StopMain, probe:502
MESSAGE  : EntryId[000][Reason:GATEWAY_INUSE]
```




© Copyright IBM Corporation 2011
All Rights Reserved.

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

IBM, IBM (logo), and DB2 are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States, other countries, or both.

Windows is a trademark of Microsoft Corporation in the United States, other countries, or both.

VMware is a trademark or VMware Inc. in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.

References in this publication to IBM products or services do not imply that IBM intends to make them available in all countries in which IBM operates. The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

The information in this publication is provided AS IS without warranty. Such information was obtained from publicly available sources, is current as of July 2009, and is subject to change. Any performance data included in the paper was obtained in the specific operating environment and is provided as an illustration. Performance in other operating environments may vary. More specific information about the capabilities of products described should be obtained from the suppliers of those products.