Information Management

IBM® DB2® 9.7
302A 全球认证考试
辅导材料

IBM 加拿大实验室

信息管理
生态系统合作伙伴

V20110520

目录

欢迎参加 DB2 9.7 学术培训！

在本培训中，你将利用 IBM DB2 （主流的 IBM 数据库服务器）和 IBM Data Studio （一套用于数据库管理和开发的、基于 Eclipse 的工具）学习数据库基础。

Disclaimer

课程目标

向学员提供担任以下工作所需要的技能：

数据库设计人员

数据库管理员 **(DBA)**

数据库开发人员

数据库咨询人员

数据库技术支持

学习数据库的基本概念，包括 **SQL** 和 **XML**

学习 **DB2** 和 **IBM Data Studio** 的基础知识

通过实际动手实验获得实际知识

本培训的**课程目标**是让学员：

1. 了解下述职业的工作内容：数据库设计人员，数据库管理员 (DBA)，数据库开发人员，数据库咨询人员 / 技术支持人员
2. 学习数据库的基本概念，包括 SQL 和 XML
3. 学习 DB2 和 IBM Data Studio 的基础知识，而且
4. 通过实际动手实验获得实际知识

课程前提条件（建议，但非必须）

Windows 或 **Linux** 操作系统的基础知识

数据库概念和 **SQL** 的基础知识

XML 的基础知识

如果在课堂环境中参加本课程，需完成 **db2university.com** 课程 **AA001EN “DB2** 学术培训**”**

前提条件：

建议的课程前提条件（并非必须）是：

1) Windows 或 Linux 操作系统的基础知识

2) 数据库概念（包括 SQL ）的基础知识

3) XML 的基础知识

4) 如果在课堂环境中参加本课程，在参加本课程之前完成 db2university.com 课程“ AA001EN” DB2 学术培训”将非常有益

本课程所需要的材料是：

1) DB2 学术培训 DVD

如果你正在课堂环境中参加本培训，本材料或许已经提供给你了。如果尚未提供的话，你可以在注册 AA001EN “DB2 学术培训”课程之后从 db2university.com 下载本 DVD 内容。该 DVD 中包含用于亲自动手实验的 PDF 格式的演示、实验指导、视频以及 VMWare 影像。

2) 硬拷贝的书籍“ Getting started with DB2 Express-C”（ DB2 Express-C 入门电子书）。

如果你正在课堂环境中参加本培训，本材料或许也已经提供给你了。如果尚未提供的话， DVD 中也包含本书的一份软拷贝以及其他支持性材料。下一张幻灯片中有这方面的更多信息。

对于实验设置，请确保在你的系统中已经安装了免费的 VMWare Player （最新版本）。你可以从显示的链接下载免费的 VMware Player 。本课程的资料包括实验“ 1.0 - VMware Basics and Introduction.pdf”以及关于使用 VMware 的指示以及所提供的影像。

目前，我们建议您在你的计算机 /DVD 中找到有关的演示和实验。并验证 VMWare 影像在你的系统中得到正确的安装。

支持本课程的免费 DVD

www.db2university.com 中的课程 **AA001EN** 有一个链接，你可以在此下载本 **DVD** 的软拷贝 **(~4.5GB)**

这是 DB2 学术培训 DVD （与本研讨班一同提供）的样子。如前所述，如果你没有收到本 DVD 的一份物理拷贝，你可以在注册 AA001EN 课程之后从 db2university.com 上下载其内容。

支持本课程的免费 DVD

免费的 **“校园 DB2”** 书籍在以下地址提供
www.ibm.com/db2/books

Database Fundamentals （数据库原理）

Getting started with DB2 Express-C （ **DB2 Express-C** 入门）

Getting started with IBM Data Studio for DB2 （面向 **DB2** 的 **IBM Data Studio** 入门）

Getting started with DB2 application development （ **DB2** 应用开发入门）

“ 校园 DB2” 系列丛书由 20 多本书构成，它们由世界各地的社区人员（专业人员、教师、学生、 IBM 人员）。这些书籍的在线版本是免费的。作为校园 DB2 项目的一部分，这些书籍中也有一些有印刷版本，在本培训中免费发放（并非在所有的情况下）。

所有这些书籍都是入门级的读物。其主题可能并不一定是关于 DB2 的；但是，如果书籍中的某一种是讨论数据库的，它将使用 DB2 。

对于本培训，本幻灯片所列出的书籍可以被用作支持性材料。这些书籍中的内容曾经被用来开发这些培训材料：大约 60% 来自于《 DB2 Express-C 入门》 。 25% 来自于《数据库原理》， 10% 来自于《 Data Studio 》， 5% 来自于《 DB2 应用开发》 。

如果你有兴趣参加本项目来协助编写更多的书籍，请通过校园 DB2 的 Facebook 群组联系 Raul Chong 。

本课程的免费支持在线课程

提供地址： **www.db2university.com**

DB2 9.7 学术培训

Code AA001EN

与本课程类似的内容

利用本课程的论坛解决问题

SQL Fundamentals I （**SQL** 基础 **I**）

DB2 Essential Training I （**DB2** 精要培训 **I**）

DB2 Essential Training II （**DB2** 精要培训 **II**）

DB2 Essential Training III （**DB2** 精要培训 **III**）

Data Studio Essential Training I （**Data Studio** 精要培训 **I**）

db2university.com 是一个在线教育网站，提供免费的及收费的在线课程。你可以注册本网站或者利用你现有的 Facebook/Gmail/Yahoo/ChannelDB2 用户 ID 来登录（使用 Open ID 验证）。

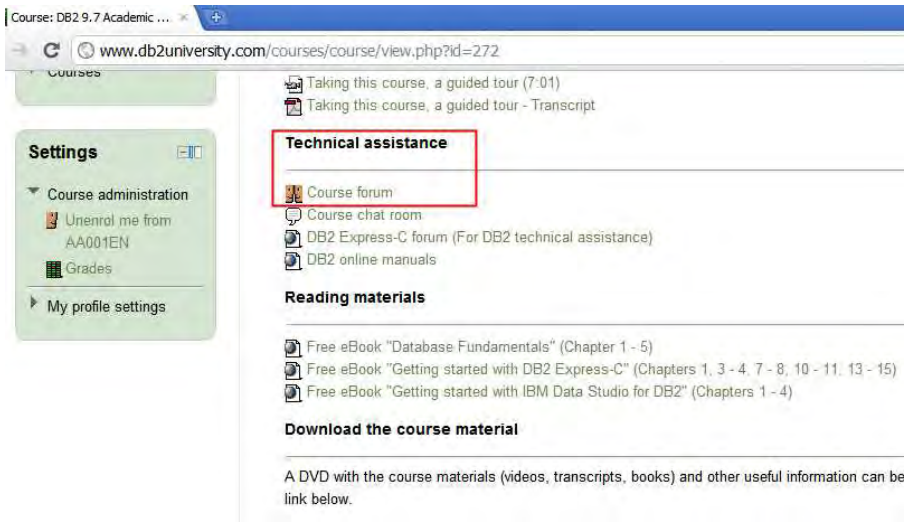
在准备参加本培训课程时，最好事先（或者在培训过程中）参加列于校园 DB2 目录中的这些免费的课程：

DB2 精要培训 I, II, III； Data Studio 精要培训 I； DB2 应用开发 I。

多数课程有 5 次课，其中包含视频、脚本、练习及考试。这些课程的内容都涉及到校园 DB2 书籍。

此外，你可以仅参加这一门课程：“**AA001EN: DB2 9.7 学术认证培训**”，它包含来自上述所有课程的视频，并使用了与本培训课程所使用的资料类似的演示文稿。另外，还有一次模拟考试，它能够很好地说明你在真实测验 302 中将表现如何。

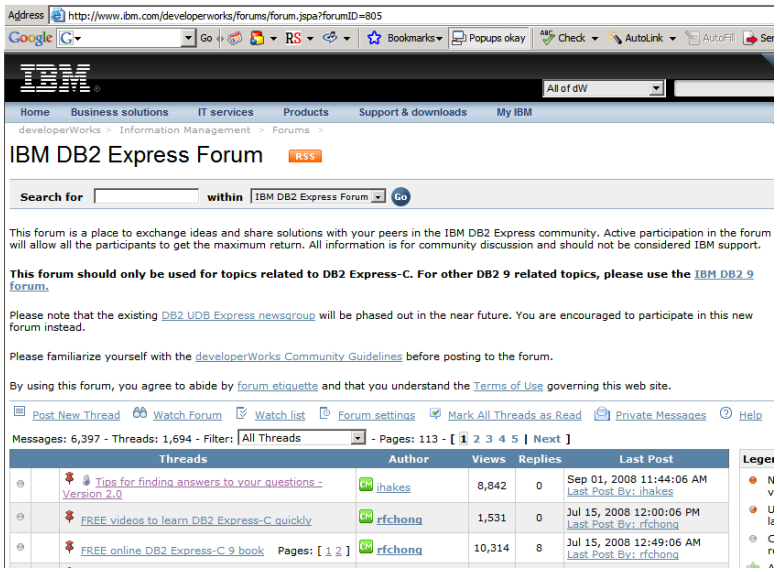
课程辅导： db2university.com 课程论坛



www.db2university.com/courses/mod/forum/view.php?id=7021

如果在参加本培训过程中遇到问题，你可以利用 db2university.com 中 AA001EN 课程的论坛。如你所看到，它在“技术辅导”板块中。

DB2 在线辅导： DB2 Express-C 论坛



www.ibm.com/db2/express

如果你遇到 DB2 技术问题，可以随时把问题贴到 DB2 论坛上。 DB2 Express 论坛由 IBM DB2 Express-C 团队成员监控，该论坛很受欢迎。

加入该论坛是免费、自愿的。要想张贴问题，你需要注册。

访问本论坛时，请先进入 DB2 Express-C 主页 **ibm.com/db2/express** ，然后点击“ DB2 online forum”（ DB2 在线论坛）按钮。

此外，对于与本培训相关的技术问题，你可以在注册参加 AA001EN 课程之后在使用 db2university.com 中的课程论坛。

课程主题

欢迎

关系数据库的概念

亲自动手实验

DB2 入门

亲自动手实验

Data Studio 入门

亲自动手实验

SQL 和数据库对象入门

亲自动手实验

这些是本培训将讨论的课程主题。它们应当按顺序讨论。这些主题可以在一个大学学期内教授，或者利用几个整天传授。

每个主题都有其相应的亲自动手实验。

这些主题包含对关系数据库的概念、 DB2 、 Data Studio 及 SQL 的介绍。

.

课程主题（续）

数据并发和锁
- 亲自动手实验

数据库安全性
- 亲自动手实验

备份和恢复
- 亲自动手实验

DB2 pureXML
- 亲自动手实验

然后是更高级的主题，包括并发性和锁定、数据库安全性、备份和恢复、 DB2 pureXML 。

课程主题（续）

DB2 应用开发

- 亲自动手实验

故障解决

- 亲自动手实验

测验 **302** 交付（可选）

** 参加本测验的一个前提条件是完成本课程的在线调查

最后，我们重点讨论数据库应用开发以及故障解决。

免费安排测验在本课堂中可能是可选的。你的课堂管理员或教师将通知你们在培训结束时是否安排测验。在本培训结束之后，测试将在稍候的日期进行，以便让学员有时间复习。如果你通过了本测验，你将成为 IBM 认证学术伙伴。

课程结束时进行的调查可以在下述地址找到：

http://www.surveymonkey.com/s/db2_academic

完成该调查能够帮助我们改进课程材料！

如果你仅仅通过在线方式参加本课程，你将无法得到免费参加本测验的机会。对此，你可以付费参加测验，或者查看你所在地的课堂安排。如果你无法找到这些信息，可以联系我们，电子邮件地址为： imschool@us.ibm.com

下一张幻灯片将详细讨论该测验。

测验 302 -- 详细信息

测验 **302 -- DB2 9** 数据库和应用基础

通过本测验可以成为 ***IBM Certified Academic Associate*** （ ***IBM*** 认证学术伙伴）

问题的数量（多项选择题）： 60 个

答题时间（分钟）： 90

及格分数： 60%

测试语言：

- 英语
- 汉语（简体）
- 葡萄牙语
- 西班牙语（拉美）

** 测验在许多学术场合是免费的；但是，请向你的课堂导师确认

本幻灯片提供了关于测验 302 的更多信息。该测验共设 60 道多项选择题，时间为 90 分钟。为了通过本测验，你需要至少答对 60%

该测验采用英语、简体汉语、葡萄牙语以及西班牙语。

如前所述，如果你参加了我们的现场培训，你就可以免费参加本测验。

我们建议你首先通过在线方式参加本课程，然后再参加现场培训，这样你就能够免费参加本测验，也有利于提高你通过本测验的机会。

关于本测验的详细信息可以在以下地址找到：

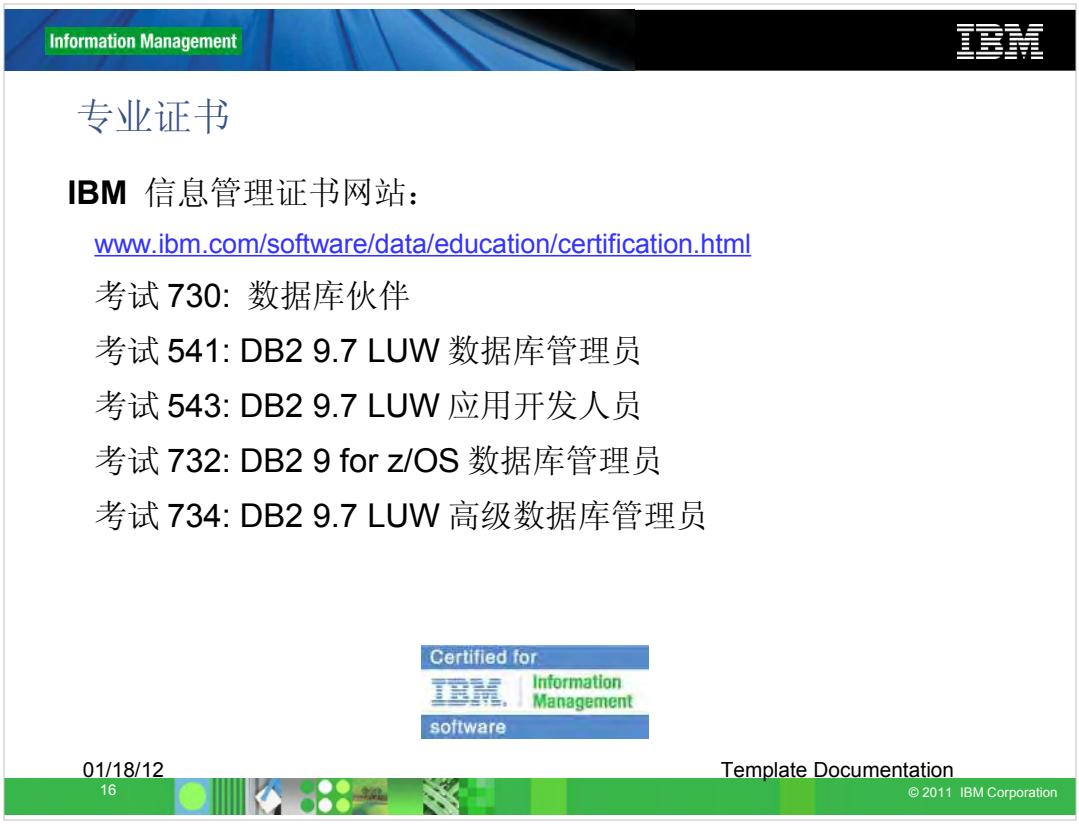
http://www-03.ibm.com/certify/tests/ovr000-302.shtml

测验 302 -- 详细信息（续）

测验 **302**： **IBM** 认证学术伙伴 **-- DB2 9** 数据库和应用基础

测验分解：

这张幻灯片是按主题分解的测验项目。



如果你打算进一步学习 DB2 并获得专业的 IBM 证书，你可以遵循以下步骤：

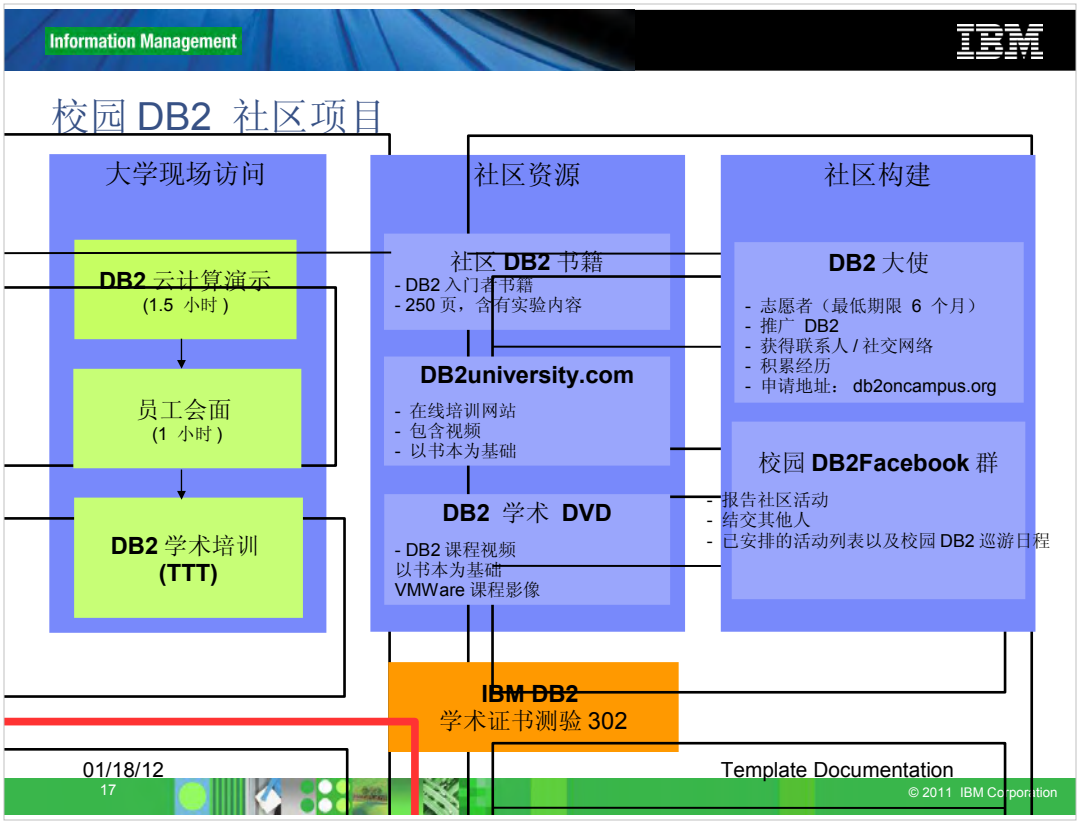
步骤 1: **IBM 认证数据库伙伴，面向 DB2 9 基础，考试 730**
考试信息： http://www-03.ibm.com/certify/tests/obj730.shtml
免费教程： http://www.ibm.com/developerworks/offers/lp/db2cert/db2-cert730.html

步骤 2（面向 DBA）： **IBM 认证数据库管理员，面向 DB2 9.7 DBA for LUW，考试 541**
考试信息： http://www-03.ibm.com/certify/tests/obj541.shtml
免费教程 (DB2 9)： http://www.ibm.com/developerworks/offers/lp/db2cert/db2-cert731.html

步骤 2（面向开发人员）： **IBM 认证应用开发人员，面向 DB2 9.7，考试 543**
考试信息： http://www-03.ibm.com/certify/tests/obj543.shtml

步骤 2（面向 z/OS）： **IBM 认证数据库管理员，面向 DB2 9 DBA for z/OS，考试 732**
考试信息： http://www-03.ibm.com/certify/tests/obj732.shtml

步骤 3： **IBM 认证高级数据库管理员，面向 DB2 9 DBA for LUW，考试 734**
考试信息： http://www-03.ibm.com/certify/tests/obj734.shtml



从该图中可以看到， DB2 学术培训是一项称为“校园 DB2” 的更大范围计划的一部分。校园 DB2 是 IBM 赞助的一个社区项目。本图中列出的所有东西均免费提供。
本项目有三个主要组成部分：
大学访问：
校园 DB2 社区的成员或者 IBM 人员将访问各个大学，以期建立关系，形成现场存在，并提升 IBM 信息管理产品的知名度。通常情况下，我们通过向学生及教师播放一些讨论“热点”技术的演示文件来扩大知名度。例如，今年的演示被称为“云计算中的 DB2” 。然后，我们会见教员，讨论一些具体的关心问题或需求。
此外，我们还提供（需要向教员提供） DB2 学术培训。 TTT 是指“培训教师”（“ Teach the teachers” ），因此，主要想法是向教师培训本材料，然后再有他们在课堂上向其学生传授。无论是参加本课程的教师，还是从教师那里接受培训的学生，都有资格免费参加认证测验 302 。

在社区资源方面，我们可以看到许多在前面已经讨论过的免费资源：免费的电子书， db2university.com 上免费的电子学习课程，免费的 DB2 学术培训 DVD ，等等。这些材料中有些已经被社区人员翻译为不同的语言，它们都是相互关联的，因此，你可以更容易地学习，而且可以用不同的方式！

在社区构建方面，校园 DB2 项目能够促进参与和社会交往。我们邀请专业人员和学生担任“ DB2 大使”。在下一张幻灯片中将更详细地讨论这个问题。

在虚线之下，是所有这些活动的最终目标：我们需要更多的人获得 DB2 认证，以便应付不断增长的需求。幻灯片中提到的所有这些资源都能够帮助你准备参加测验 302 。如果你通过了本测验，你就可以获得认证，成为 “ DB2 学术伙伴”。

DB2 （学生）大使项目

学生或专业志愿者

最低期限： 6 个月

无需事先具备 DB2 知识

完成的工作越多，越能够丰富你的简历

扩大交往范围，但不保证就业

申请地址： www.db2oncampus.org

如前所述， DB2 大使即可以是专业人员，也可以是学生，他们愿意积极在其组织机构中推广 DB2 以及其他 IBM 信息管理产品，例如 Data Studio 。最短参与期限为 6 个月。要想成为一名 DB2 大使，可以在 db2oncampus.org 上申请。在申请时你无需了解 DB2 ，但我们希望你在担任大使期间能够获得 DB2 认证。

你担任大使期间完成的工作越多，你就越能够丰富你的阅历。成为一名大使的主要好处是你能够得到与世界各地的 IBM 人员交往的机会。这或许可以帮助你（虽然不能保证）找到更好的工作。

DB2 （学生）大使项目

任务示例：

建立（大学） DB2 用户群

推广 DB2 Express-C 下载

创建 / 翻译 db2university.com 上的课程

编写 / 翻译文章或书籍

把应用迁移至 DB2

参加应用开发项目

选择优秀文章张贴于博客

与 DB2 学生大使候选人会面

这些是 DB2 大使可以完成的工作的例子。

为了在大学中或者公司中建立一个 DB2 用户组，只需要建立一个 Facebook 群组就足够了，然后可以在你的组织机构推广它，吸引更多的人参加。

在 IBM 多伦多实验室开展工作的学生举例

Henrique Copelli Zambon

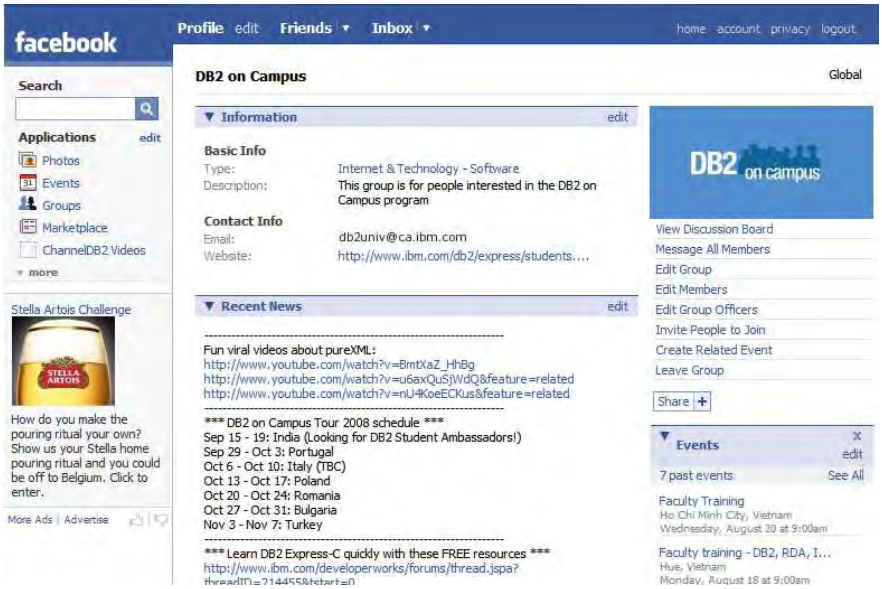
巴西圣保罗大学计算机工程专业学生

在 IBM 多伦多实验室参加为期 16 个月的实习，自

他的项目中所使用的技术：
Ruby on Rails
PHP
Web 服务
云计算
Android

这是一个巴西学生的例子，他参加了 IBM 多伦多实验室为期 16 个月的实习，这得益于他与校园 DB2 项目相关的交往活动。本幻灯片列出了他目前正在开展的项目 / 技术。

校园 DB2 在 Facebook 上的群组



加入 Facebook 中的校园 DB2 群组，随时了解该项目的最新消息。结交新朋友，与从事 DB2 的同事和专业人员建立联系。校园 DB2 巡回日程也公布于本群组中。该群组的直接链接是：

http://www.facebook.com/group.php?gid=3000790461

IDUG: 国际 DB2 用户组

全球超过 10,000 名 DB2 用户构成的社区

IDUG 优势和特点：

- 世界各地亲自参加的活动
- 动态的网站： www.IDUG.org
- 在线教育、培训以及提示和技巧
- 在线社区讨论论坛
- *Solutions Journal*（*解决方案杂志*）， IDUG 同行审阅的出版物
- 向供应商提交未来产品方向及功能的交互式的、协作性的流程

观看 http://youtube.com/watch?v=S2kQR21gHMg ， 体验各种 IDUG 服务。

或者，在以下网址了解 IDUG 的详细信息： www.IDUG.org 。

IDUG 是世界各地 DB2 专业人员的社区。它有 10,000 多名成员。欢迎加入结交新朋友！

ChannelDB2 是 DB2 迷的社交网络。它提供了与 DB2 及相关产品有关的短视频。在 ChannelDB2 中观看视频，可以快速学习信息管理产品最新的功能。建立群组，邀请朋友，结交网络！

IBM 信息倡导者项目

IBM 对以下社区成员的认可：
在论坛中做出贡献
利用 IBM Data Management 软件运行网站
编写文章和技术指南
视频视频和播客（ podcast ）
在技术会议上演讲
传授技术技能。

奖励：
受邀参加特殊活动
在 IBM 网站上给予特别介绍
免费获得软件供个人使用，等等

www.ibm.com/software/data/champion

IBM 信息倡导者项目是 IBM 提供的一个单独项目，它用于弘扬社区产品对推广 IBM 信息管理产品所做出的贡献。许多 DB2 大使都获得提名，有些已经获得了此声誉。

其他读物

红皮书（免费） (www.redbooks.ibm.com/)

理解 **DB2 9** 安全性 0-13-1345907

面向开发人员的 **DB2 9** 978-158347-071-9

印刷书籍（收费的）

DB2 9 基础 978-1-58-347072-5

面向 **Linux** 、 **UNIX** 及 **Windows** 数据库管理的 **DB2 9** 158347-077-8

面向 **z/OS** 数据库管理的 **DB2 9** 978-158347-074-9

面向 **Linux** 、 **UNIX** 及 **Windows** 数据库管理升级的 **DB2 9** 158347-078-6

面向 **Linux** 、 **UNIX** 及 **Windows** 的 **DB2 9 –** 第六版 0-13-185514-X

理解 **DB2** ： 通过示例以可视化方式学习 0-13-158018-3

红皮书是免费资料，可以从以下地址下载： www.redbooks.ibm.com 。它们涵盖了广泛的主题，能够帮助你免费获得 IBM 软件的技能！

如果你打算进一步学习 DB2 并获得认证的话，你或许还需要购买一些收费的印刷书籍。

面向 IBM 业务合作伙伴的其他研讨班（训练营）

DB2 9.7 LUW 及迁移诊所

DB2 pureXML

Informix 11.5

IBM InfoSphere Warehouse v9.7

IBM Optim 解决方案

Guardium

SolidDB

InfoSphere 变化数据采集

InfoSphere 信息服务器

InfoSphere 主数据管理

Guardium

www.ibm.com/developerworks/data/bootcamps/

IBM 还为 IBM 业务合作伙伴提供了免费的现场（面对面）训练营。这些研讨班涵盖了与 IBM 信息管理产品相关的不同主题。虽然其目标受众是 IBM 业务合作伙伴，但在某些情况下，如果地方富裕的话，教员们也可以参加。与该训练营相关的某些免费在线资料可以在以下地址找到：

https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=4ac531ae-6ed1-43b7-91e9-ab9b1142b24b

其他资源

DB2 信息中心（在线手册）：

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/

教程 **/** 自学

www.ibm.com/software/data/education/selfstudy.html

数据管理杂志

www.ibmdmmagazinedigital.com/

性能角度 **–** 关于随需应变信息的洞察和思想

www-01.ibm.com/software/data/performance-perspectiv

该幻灯片介绍了你可以利用的更多资源。 DB2 信息中心提供了在线的 DB2 手册。关于 DB2 故障 / 问题的多数答案都可以从 DB2 信息中心找到。我们将在本课程结束时的“故障解决”演示中详细讨论 DB2 信息中心。

其他资源（续）

IBM 学术计划（面向员工）

www.ibm.com/developerworks/university/academicinitiative/

IBM 学术计划学生门户

www.ibm.com/developerworks/university/students/index.html

IBM 学生机会系统

www.ibm.com/developerworks/university/students/sos/

IBM developerWorks

www.ibm.com/developerworks/

如果你正在参加本培训课程，你很可能已经知道了 **IBM Academic Initiative （学术计划）** (AI) 。如果你尚未参加，可以访问 AI 门户并注册！ 加入 IBM 学术计划能够让你免费获得可以在你的研究项目及课堂上使用的软件和教育资料！ 这些资源只能用于研究 / 教育目的。

对于学生，也有一个 AI 门户。此外，获得 IBM 认证的学生可以把他们的简历张贴于 **IBM SOS （学生机会系统）。**

DeveloperWorks 是一个面向开发人员和管理员的门户，它提供了大量免费资料。你可以从该网站获得关于 IBM 产品的大量文章、教程和演示。此外，你还可以加入“ **MyDeveloperworks**”，这是一个由对 IBM 产品有兴趣的人员组成的专业社交网络。

下载免费软件

DB2 Express-C 和 Data Studio

www.db2express.com

或

www.ibm.com/db2/express/download.html

使用本培训课程所提供的 VMWare 影响实际上是可选的。如果你喜欢的话，你可以下载本课程所使用的软件并把它们安装到你自己的计算机上：

DB2 Express-C

IBM Data Studio

两者都可以从所提供的任一链接中下载。它们是免费产品。

如果你对本课程所涵盖的资料有什么技术问题的话，可以使用 db2university.com 中关于课程 AA001EN 的论坛。那里的同学、老师和 IBM 人员可以帮助你！

在本演示中，你将学习关系数据库的基本概念

议题

- 概述
- 信息和数据模型
- 关系模型
  - 实体 - 关系图
  - 关系的类型
  - 把实体映射为表
  - 关系模型的概念
  - 关系模型约束
  - 规范化

支持性阅读材料和视频

阅读材料

数据库原理电子书

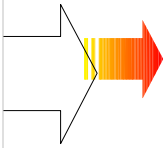
- 第 1 章：数据库和信息模型
- 第 2 章：关系数据模型
- 第 3 章：概念数据模型（可选）
- 第 4 章：关系数据库设计（可选）

视频

db2university.com 课程 AA001EN

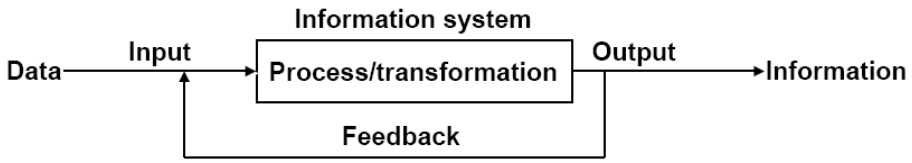
- 第 1 课：关系数据库的概念

议题



- 概述
- 信息和数据模型
- 关系模型
  - 实体 - 关系图
  - 关系的类型
  - 把实体映射为表
  - 关系模型的概念
  - 关系模型约束
  - 规范化

数据与信息



数据：文字、数据或事实的集合

信息：经过处理，能够提供价值的数据

我们首先来了解“数据”与“信息”之间的区别。如图所示，数据 是文字、数据或事实的集合，它本身不会带来什么价值，或者根本没有价值。当数据被作为输入传递到信息系统进行处理和转换之后，它就成为信息。信息能够带来价值。在处理数据的过程中，可以不断给以反馈，以便优化待收集的数据类型以及收集方式。

数据库和 DBMS

数据库

- 数据的储存库

DBMS（数据库管理系统）

- 用来管理数据库的软件系统
- “数据库”、“DBMS”、“数据服务器”、“数据库服务器”经常互换地指代 DBMS

为什么要使用 **DBMS?**

- 安全性
- 可以以很高的性能处理许多用户
- 允许并发处理，同时保持数据的一致性
- 防止灾难

数据可以用多种方式存储。例如，可以采用文本文件，电子表格，等等。但是，当访问或者操纵数据时，这些方法会带来一些问题。例如，如果 100 名用户需要访问并修改一份文本文件，那么，打开该文件就会成为一个问题。这就是需要数据库的原因。数据库并不一定比其他方法便宜，但它更易于管理数据。

数据库是数据的储存库，**DBMS**（数据库管理系统）是管理数据库的软件系统。通常，“数据库”、“DBMS”、“数据服务器”、“数据库服务器”等术语可以互换地指代 DBMS 。

为什么要使用 DBMS?

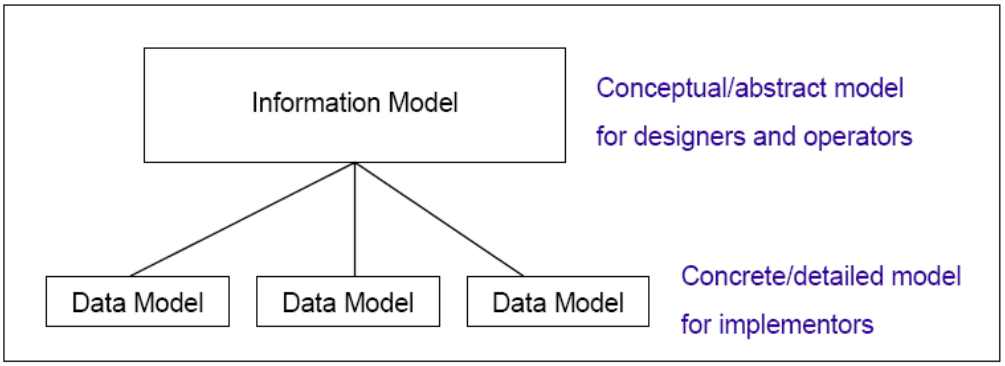
DBMS 提供了安全性以及更好的性能，即使是有数千个用户使用它。它能够处理并发性，同时又能够保持数据的一致性，还可以防止灾难的发生。

议题

- 概述
- 信息和数据模型
- 关系模型
  - 实体 - 关系图
  - 关系的类型
  - 把实体映射为表
  - 关系模型的概念
  - 关系模型约束
  - 规范化

信息和数据模型

信息模型与数据模型之间的关系



信息模型是对实体的一种抽象、正规的表达，其中包含实体的属性、关系以及可以对实体进行的操作。

信息模型的主要目的是在概念层次上对被管理的对象进行建模，这独立于用来传输数据的任何具体的实施或协议。

而数据模型是在更具体的层次定义的，它包含更多的细节。数据模型是面向软件开发人员的，其中包含了许多与具体协议相关的构造。数据模型是任何数据库系统的蓝图。

该图示出了信息模型与数据模型之间的关系。

数据模型

网络

层次

关系

实体 - 关系

扩展的关系

语义

面向对象

对象 - 关系

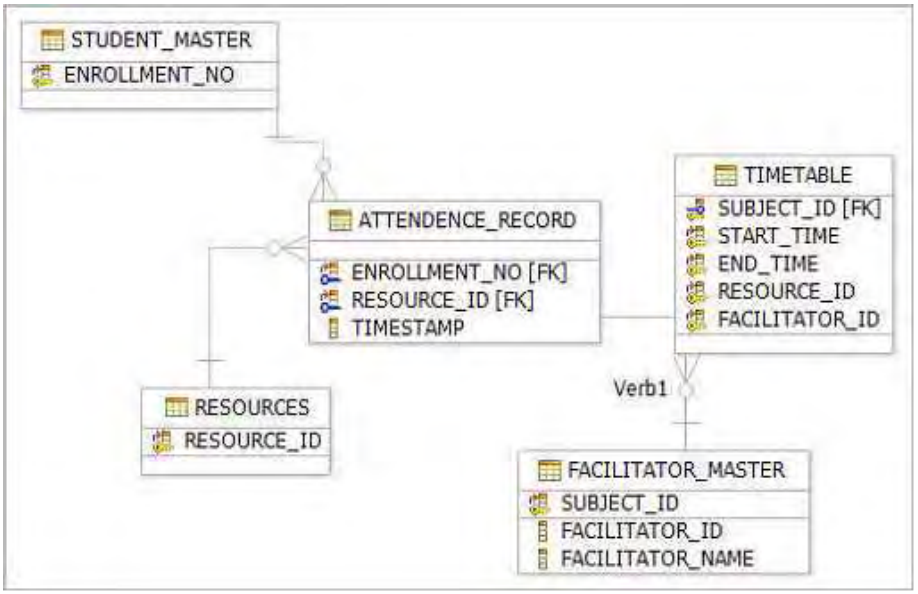
半结构化

数据模型的发展可以份为九个历史阶段：
网络、层次、关系、实体 - 关系、扩展的关系、语义、面向对象、对象 - 关系、半结构化
在本培训中，我们仅集中讨论“关系数据模型”。

议题

- 概述
- 信息和数据模型
- 关系模型
  - 实体 - 关系图
  - 关系的类型
  - 把实体映射为表
  - 关系模型的概念
  - 关系模型约束
  - 规范化

关系模型



关系数据模型简单而优雅。它具有基于集合理论及谓词演算的坚实的数学基础，是当今数据库中使用最广泛的数据模型。

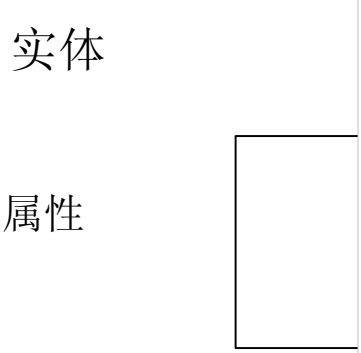
本图示出了一个实体 - 关系（即 E-R 图）的例子，表达出一个简单关系模型的若干实体或表以及它们之间的关系。

实体 - 关系图

构建块

实体

属性



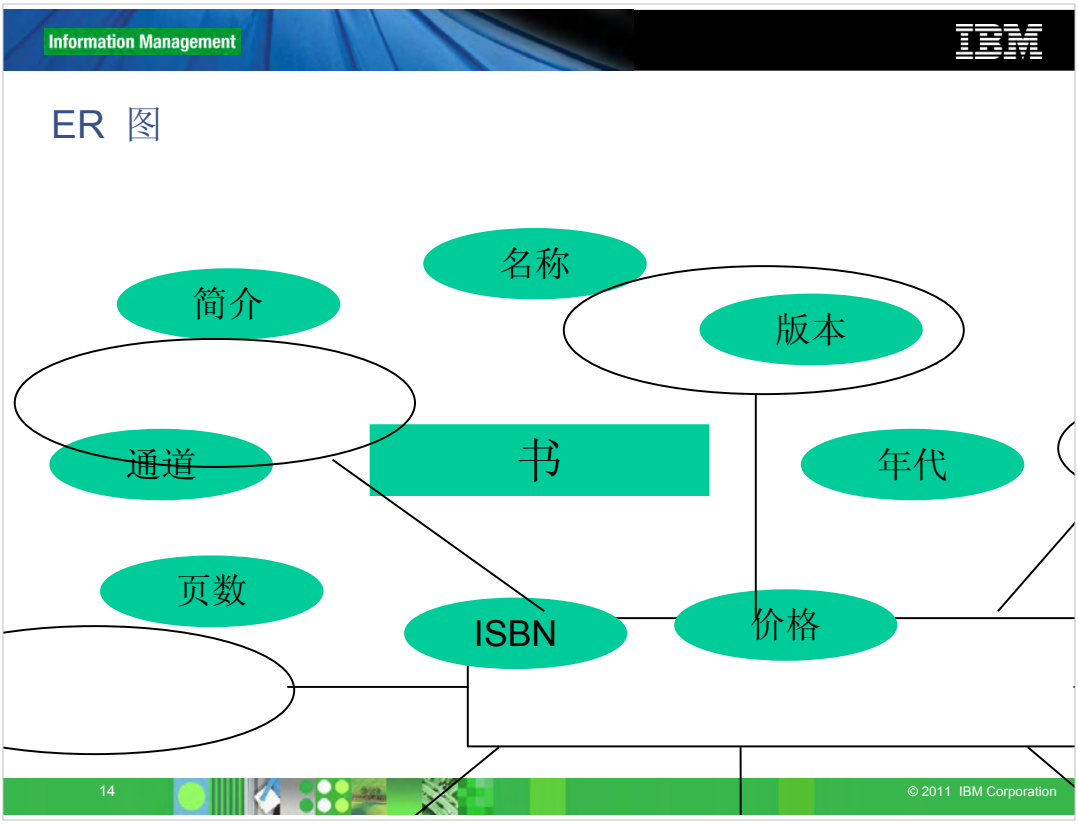
实体 - 关系图 (ERD) 在支持关系模型方面很受欢迎。
ERD 的构建块是实体和属性。
实体被画为长方形，而属性被画为椭圆形，如图所示。
属性通过一条直线仅仅与一个实体相连。
实体是具有某些属性的对象。

实体和属性

实体可以是一个名词。例如，汽车。

属性是该实体的特定属性。

例如，汽车有不同的颜色，有不同的燃油类型（例如汽油或柴油）及构造，还有制造日期。



我们来看看另一个例子 -- 书。书在这里是一个实体。

书有特定的特征，例如名词、版本、出版日期、价格、 ISBN 、页数，等等。这些特征就是属性。

现在，你可以看到，我们有一个实体，即由长方形代表的一本书。该长方形以及画在椭圆中的多项属性，与这一实体相连接。

这一图示就是一幅抽象的 ER 图，它只是实际设计关系数据库模型众多工作中的第一步。

练习：寻找实体和属性

房屋

电话号码

社会保障号码

计算机

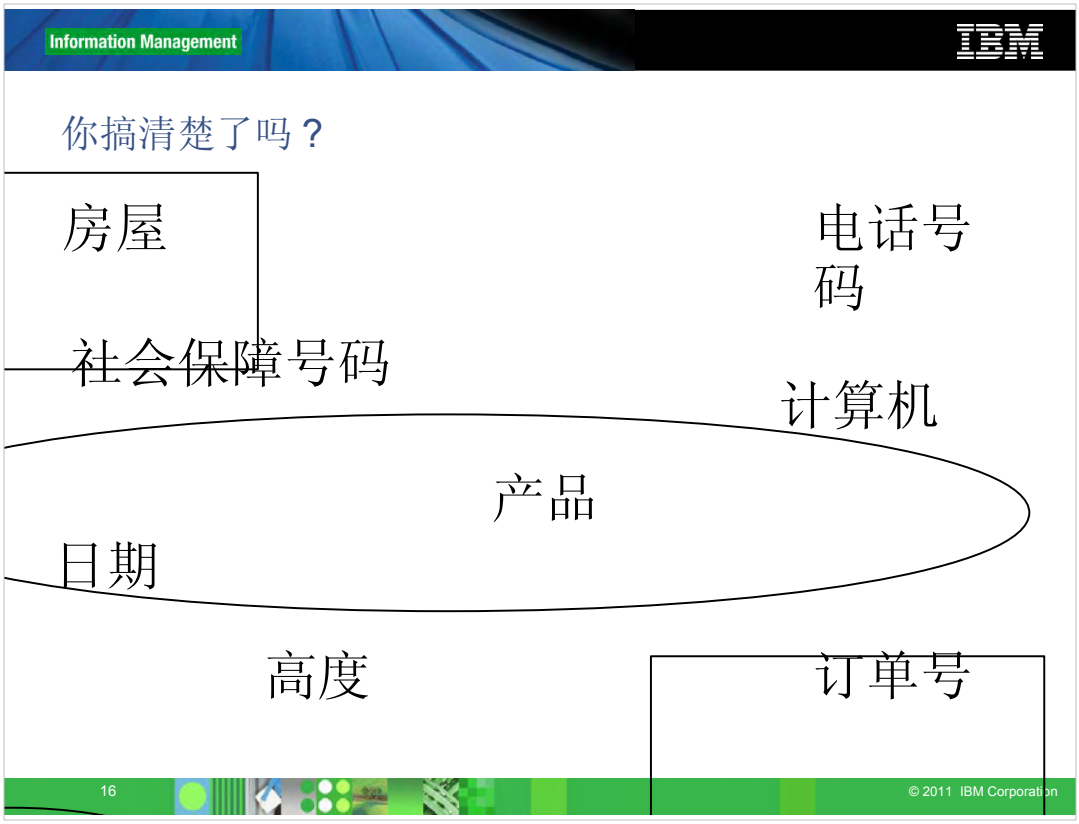
产品

日期

高度

订单号

你能从下面的列表中指出实体和属性吗？



房屋、日期、计算机、社会保障号码、高度、订单号、产品、电话号码。

房屋、计算机 、产品是实体。

能够比较容易地识别出实体，因为它们是物体。

其余的都是某种形式的属性，它们只能与某个特定的物体相关联。我们来回忆一下：实体以长方形来表示，而属性以椭圆形来表示。

议题

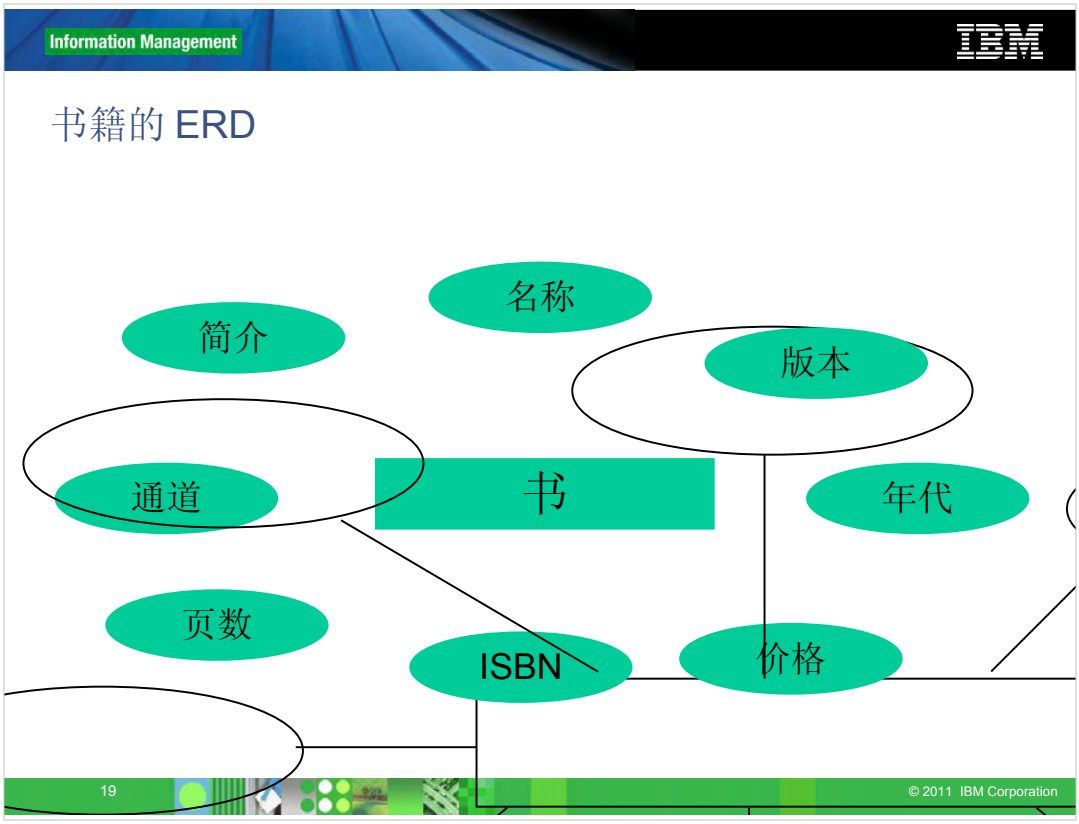
- 概述
- 实体 - 关系图
- 关系模型
  - 实体 - 关系图
  - 关系的类型
  - 把实体映射为表
  - 关系模型的概念
  - 关系模型约束
  - 规范化

关系

构建块

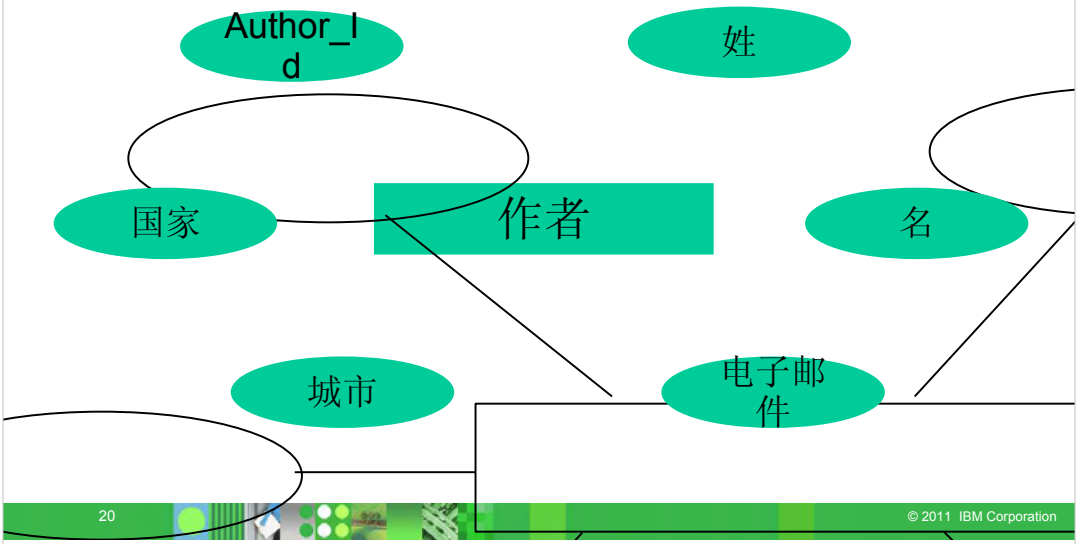
- 实体集
- 关系集
- 鸡爪符号

关系的构建块包括实体集、关系集以及鸡爪符号 (crows foot notation)。
你已经熟悉了用长方形来表达实体集。
关系集由菱形来表示，它有多条线段与相关的实体相连。
在表达关系时可以使用不同的技巧。
我们使用鸡爪符号以便于理解。
这些符号有的是大于号、小于号和一条垂直线。



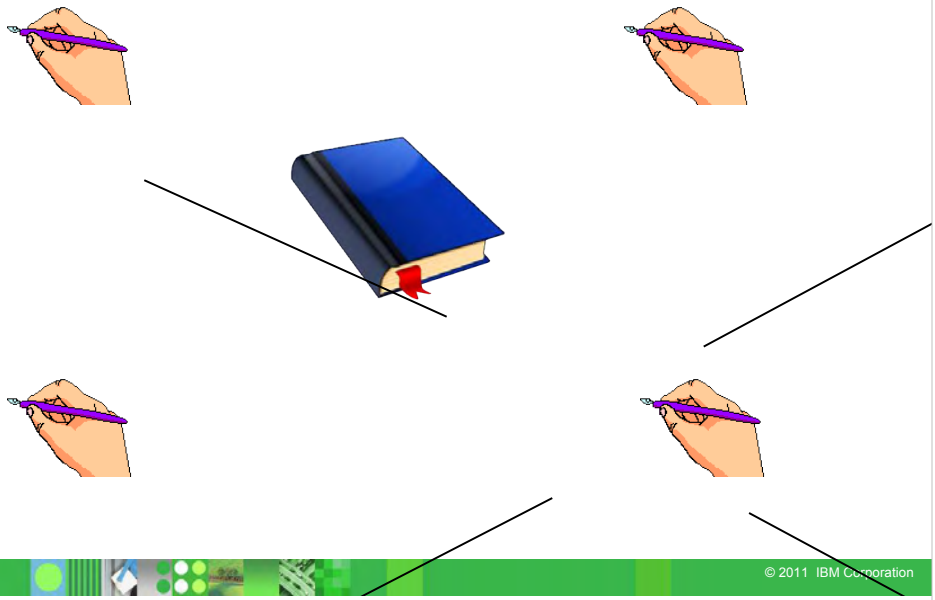
你或许还记得，我们在前面为实体“书”画过类似这样的 ERD 。

作者的 ERD



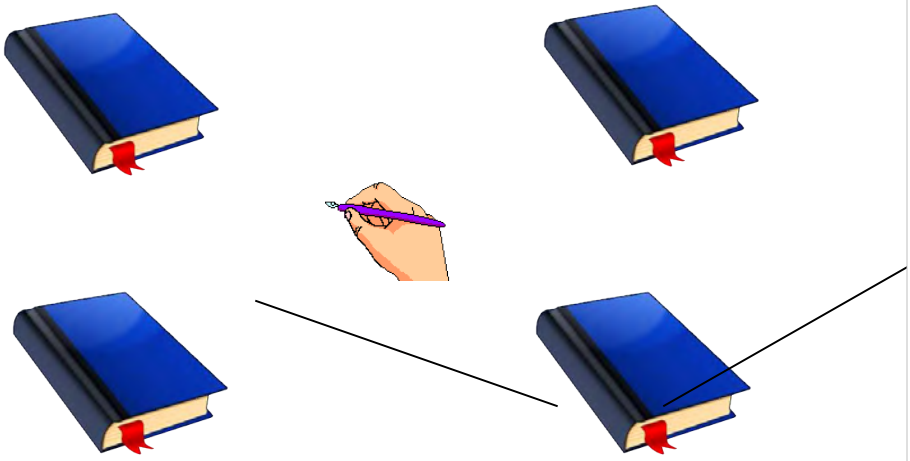
我们再来看一个实体 --“ 作者”，其 ERD 看起来应当是在这样。
实体“作者”具有姓、名、电子邮件地址、城市、国家以及作者 ID 等属性。
我们下面来看看“书”和“作者”这些实体之间是如何关联的。

示例 1



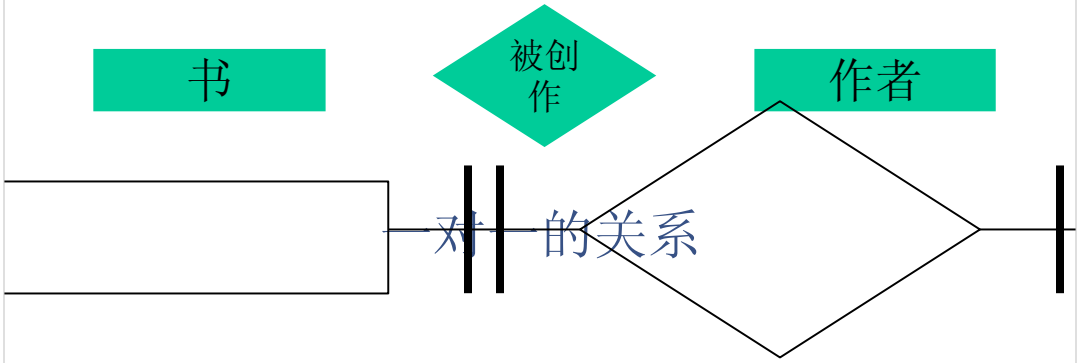
我们来看一个例子。我们都知道，一本书起码需要一名作者来编写。
两名作者合写一本书也很常见。
有时，我们也会看到许多作者共编一本书。

示例 2



我们来看看另一个例子。一名作者可能仅有一本著作，也可能有两本或者更多本。在所有这些情况下，书与作者之间都存在某种关系。

关系的类型



我们现在来讨论实体之间存在的关系类型。

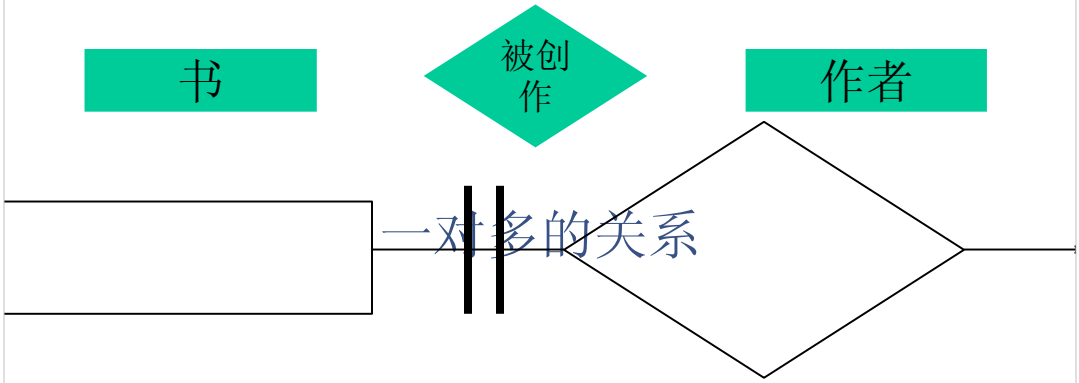
我们看到，一本书至少有一名作者。

这里的关系集是“被编写”。我们可以说，一本书必须由一名作者编著。

粗线表示实体集中的每个实体至少涉及到一个且仅一个关系。这被称为“一对一关系”。

你或许注意到，我们在这里仅使用了实体。属性经常被省略掉，因为它们可能使得关系图显得很凌乱。

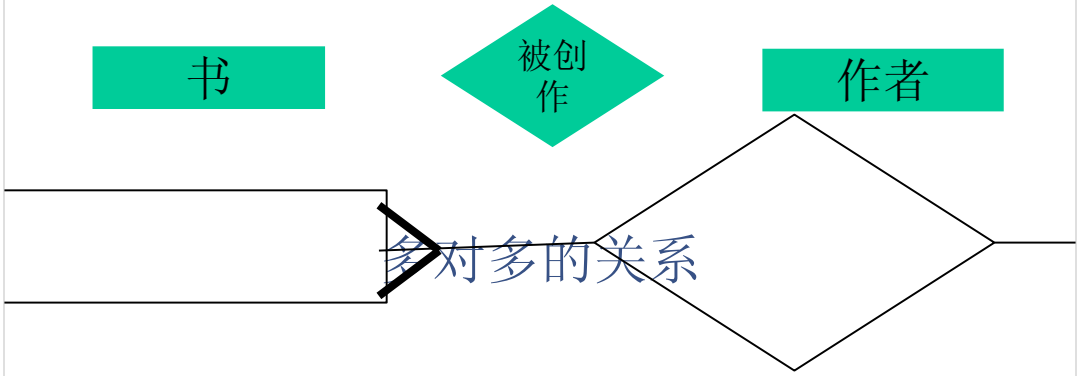
关系的类型（续）



我们前面看到，多名作者可能共同编写一本书。这可以用一个不同的鸡爪符号来表示。在本例中，使用一个小于号。

这表明，一个“书”实体参加到关系集的多个关系中。这是一对多的关系。如果说多名作者创作一本书的话，我们也可以把这称为“多对一”的关系。

关系的类型 （续）

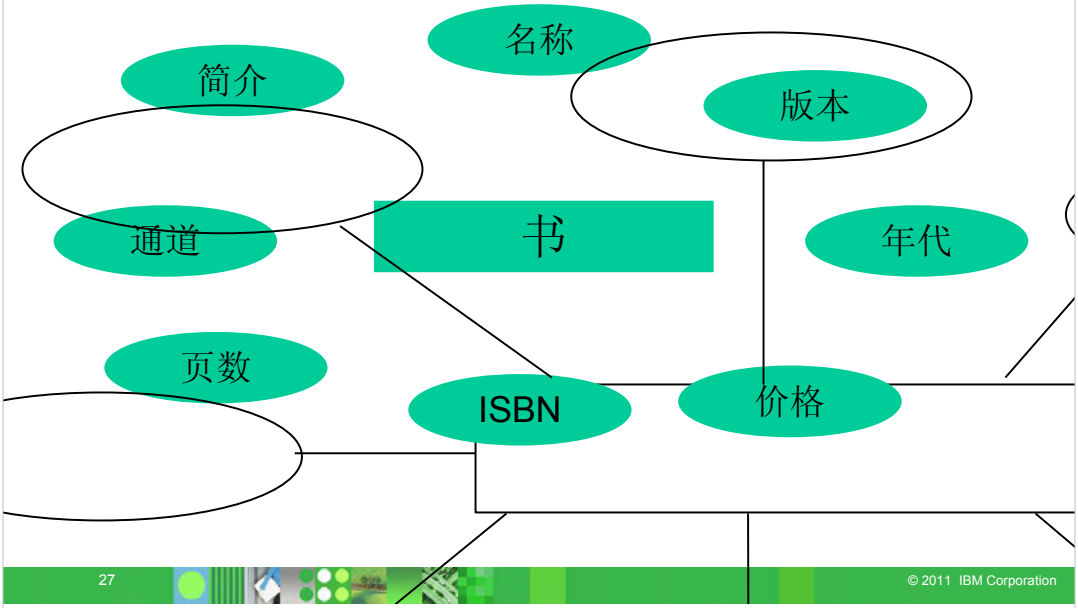


我们来看看当有多名作者创作多本书时该如何表达。
我们在这个关系集的两侧使用了大于号和小于号。
这是一个多对多的关系，其实体集中的每个实体均参加一个以上的关系。
这是许多本书由许多作者编著，或者许多作者编著许多本书。

议题

- 概述
- 实体 - 关系图
- 关系模型
  - 实体 - 关系图
  - 关系的类型
  - 把实体映射为表
  - 关系模型的概念
  - 关系模型约束
  - 规范化

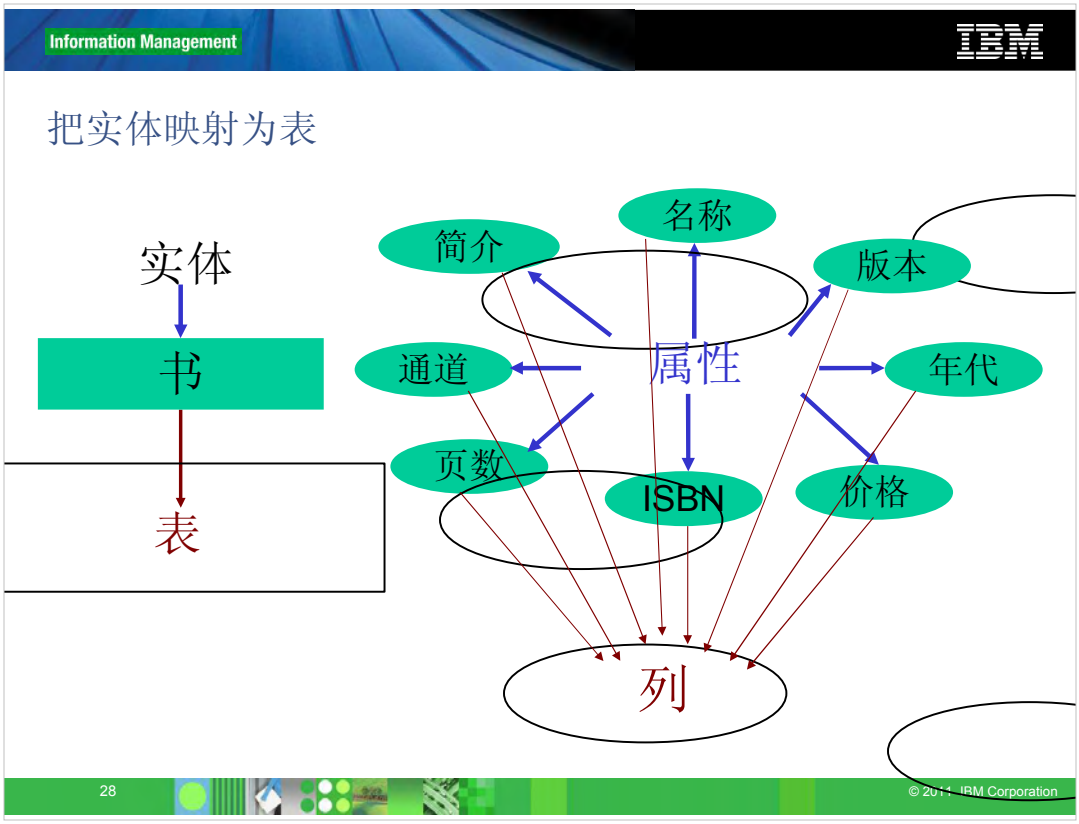
修改的 ERD



我们现在看看如何把实体映射为表。你们已经知道， ERD 是设计数据库的基本基础。

在关系数据库设计中，我们首先从 ERD 入手，然后把它们映射为数据库中的表。

你们对这个 ERD 已经很熟悉了，是吧？ 实体“书”有多个属性。我们来看看如何把这些实体和属性映射为表。



为了便于理解，我把实体与属性分开。

书是实体，有一些与该实体相关联的属性。

在映射过程中，实体“书”成为一张表，该表具有同样的名称“书”（ Book ）。

所有的属性均转化为表中的列。我们现在看看如何在关系数据库模型中表达一张表。

把实体映射为表 （续）

表： Book

Title	Edition	Year	Price	ISBN	Pages	Aisle	Description
数据库原理	1	2010	24.99	978-0-98662 83-1-1	300	DB-A02	向你教授数据库原理
DB2 Express-C 入门	1	2010	24.99	978-0-98662 83-5-1	280	DB-A01	利用 DB2 Express-C （ DB2 的免费版本），向你教授 DB2 的核心内容

你们都知道，表是行和列的结合。在映射时，实体变为表。虽然这样说，它还没有呈现表和列的形式。
在表中被转化为列的属性提供了实际的表的格式。
我们随后可以向每一列添加一些值，从而实现表的形式。

把实体映射为表（续）
表： Author

Author_ID	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

再来看看另一个实体“作者” (author)，它被从我们在前面看到的 ERD 转化为表。
所有的属性构成了各列，一些数据填入各列中形成了此表。

议题

- 概述
- 实体 - 关系图
- 关系模型
  - 实体 - 关系图
  - 关系的类型
  - 把实体映射为表
  - 关系模型的概念
  - 关系模型约束
  - 规范化

关系模型的概念

IBM 公司的 E.F. Codd 博士于 1970 年发表论文：
“ 一种面向大型共享数据资料库的关系模型”

构建块

关系

集合

我们来谈谈关系模型的概念。此模型最早由 IBM 公司的 E.F. Codd 博士于 1970 年在其论文“一种面向大型共享数据资料库的关系模型”中提出。

关系模型的构建块是关系和集合。

数据的关系模型是以“关系”的概念为基础的，而关系是一个以“集合”的思想为基础的数学概念。

集合是由互不相同的元素构成的无顺序的集体。它是由相同类型的项目构成的集体。

它们一般没有顺序，没有重复。

关系数据模型旨在提供更好的数据独立性。

关系数据库

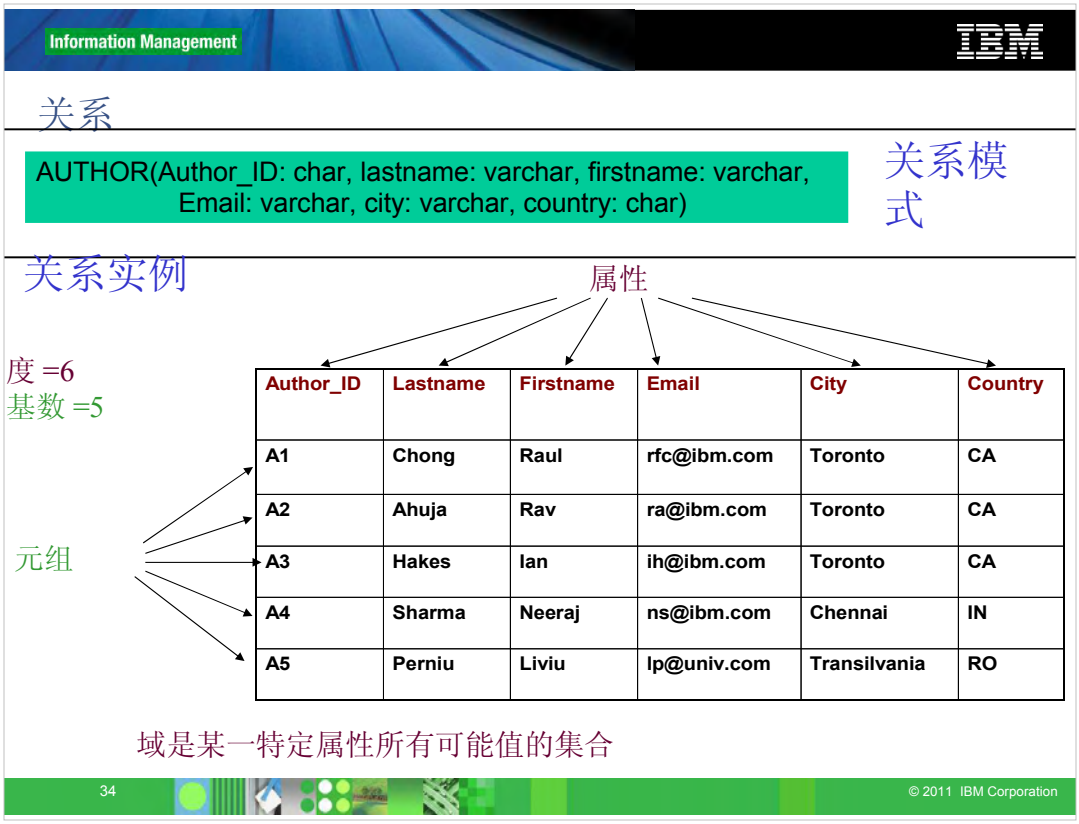
关系数据库

关系

关系模式

关系实例

关系数据库是关系的集合。
关系是表的数学名称。
表进而由行和列构成。
关系由 2 部分构成 – 即关系模式和关系实例。
关系模式规定了关系的名称以及各个列的姓名和类型。



我们来看看我们在前几课中看到过的实体“ Author”（作者）。

AUTHOR 是关系的名字。 Author_ID 是一个属性，可以容纳类型为 char 的数据。同样， lastname 、 firstname 、 email 和 city 都是类型 varchar 。如果你不清楚这些是什么，它们是数据类型。最后一个属性 Country 是 char 数据类型。

这构成了关系模式。

另一方面，关系实例是由行和列构成的表。

列在关系中被成为属性或者字段，而行被称为元组。

度是指关系中列的数量。

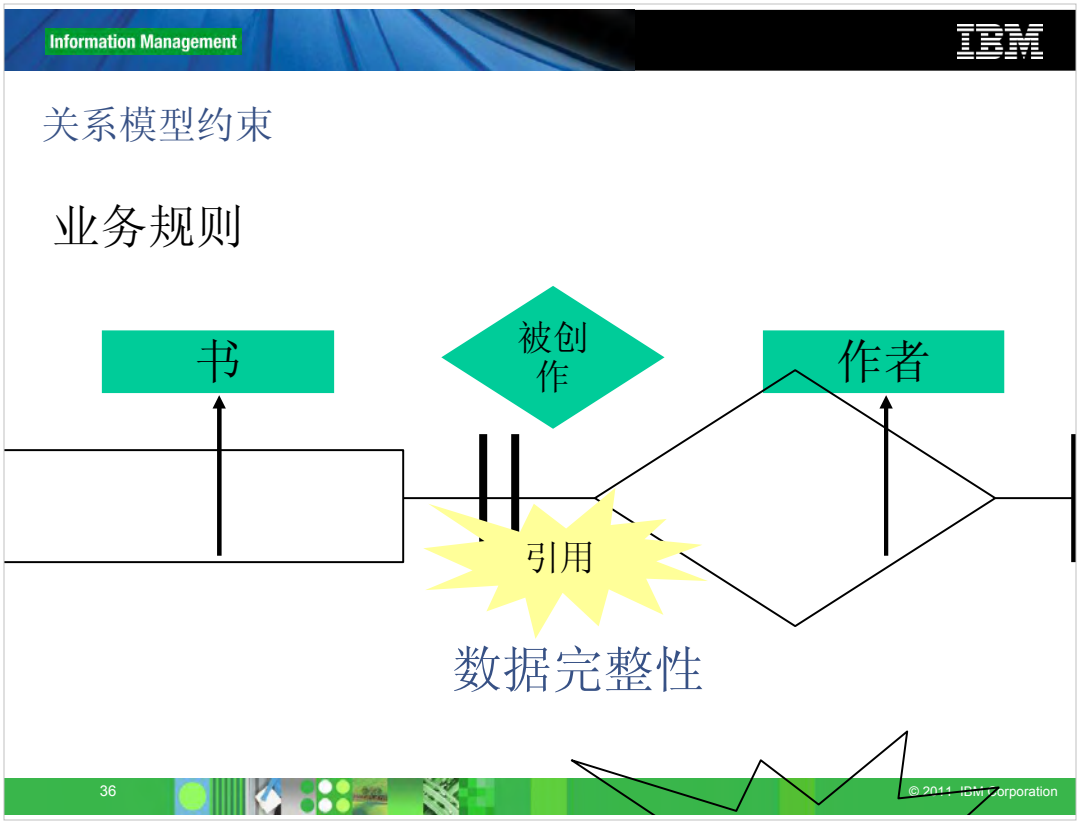
基数是元组的数量。

该图显示，读数为 6 ，因为我们有 6 列，而基数为 5 ，因为我们有 5 个数据行。

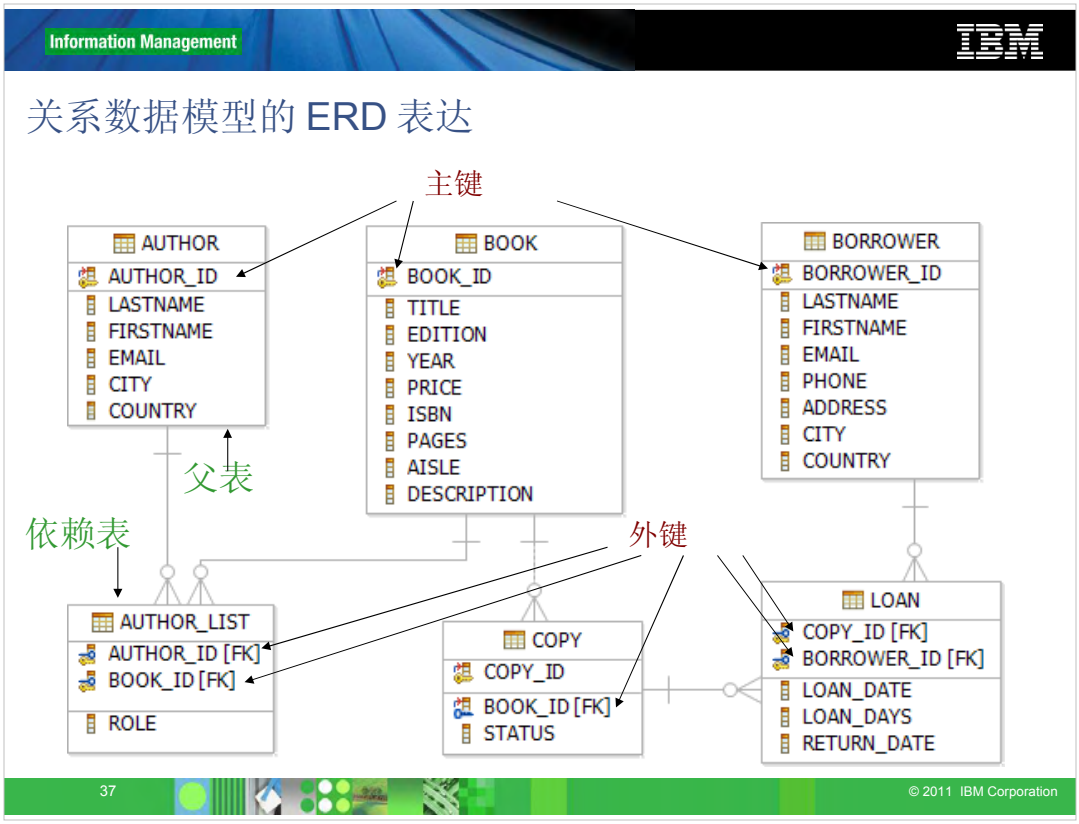
域是某个特定属性所有可能值的集合。例如，对于属性 “ Country” 而言，域就是所有可能的国家代码的集合。

议题

- 概述
- 实体 - 关系图
- 关系模型
  - 实体 - 关系图
  - 关系的类型
  - 把实体映射为表
  - 关系模型的概念
  - 关系模型约束
  - 规范化



我们来简单谈一谈关系模型约束。在任何业务中，数据通常都必须符合某些约束和规则。
以“书”和“作者”为例。
除非书由作者编写，否则，书实际上是不会存在的。
至少“一对一”的关系是必须的。
如你在例子中所看到，或者实体“书”指向实体“作者”以查找作者信息，或者相反。
这在关系数据模型中被称为“引用”。
这对于在关系数据库的两个关系之间建立数据完整性是非常重要的。
我们可以有多个引用，而不是仅限于“一对一”的关系。



我们来看看如何使用 ERD 方式来表达关系模型。

我们有多个实体，例如 Author 、 Book 以及 Borrower ，等等。还要注意， Author_ID 、 BOOK_ID 和 BORROWER_ID 带有一个特殊的图标，其中有一把钥匙。

该类属性被称为主键。观察一下本 ERD 下面部分的实体。某些属性在其旁边的方括号中有 FK 字样。这被称为外键。

要注意，这些实体是它们上面那些实体之间的关系集的一部分。它们之间都存在“一对多” D 关系。

关系表的主键是能够唯一地识别表中各个记录的列的集合。推荐采用主键，但它是可选的。

外键是指向其他表的主键的列的集合。它是可选的，但可以用来建立引用 / 数据完整性。

在开始讨论关系模型约束之前，我们来学习另外两个在关系模型经常使用的术语。

如果一个含有主键的表与至少一个外键相关，则该表被称为父表。在我们的例子中， Author 实体就是一个父表。 Book 实体也是父表。含有一个或多个外键的表被称为依赖表。在本图中， author_list 有两个外键，它们指向两个不同的父表。

约束

- 实体完整性约束
- 引用完整性约束
- 语义完整性约束
- 域约束
- Null （空值）约束
- 检查约束

回到约束问题。约束有助于实施业务规则，我们在关系数据库模型中定义了如下约束。

实体完整性约束、引用完整性约束、语义完整性约束、域约束、 null （空值）约束和检查约束。我们在下面几张幻灯片中详细学习它们。

实体完整性约束

作者

Author_ID [PK]	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

要识别出关系中的每个元组，该关系中必须有一个主键。主键是能够识别各个行的唯一值。这就是实体完整性约束。一般使用主键约束和唯一约束这样的说法。为了实施这些约束，通常要采用索引，这将在后面的演示中详细介绍。

实体完整性约束是指，参加关系中主键的属性不允许出现空值。

为了进行解释，我以前面演示中已经看到的关系 Author 为例加以说明。

实体完整性约束

作者

Author_ID [PK]	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

你们已经知道， Author_ID 是关系 Author 中的主键。

现在，该键可以识别出此关系中的各个元组。

例如，我说 Author_ID A1，它指的就是来自 Toronto 的作者 Raul Chong 。

实体完整性约束

AUTHOR

Author_ID [PK]	Lastname	Firstname	Email	City	Country
NULL	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

如果我以 MULL 取代值 A1，会怎么样呢？ 我们仍然可以识别出作者 Raul Chong 。

实体完整性约束

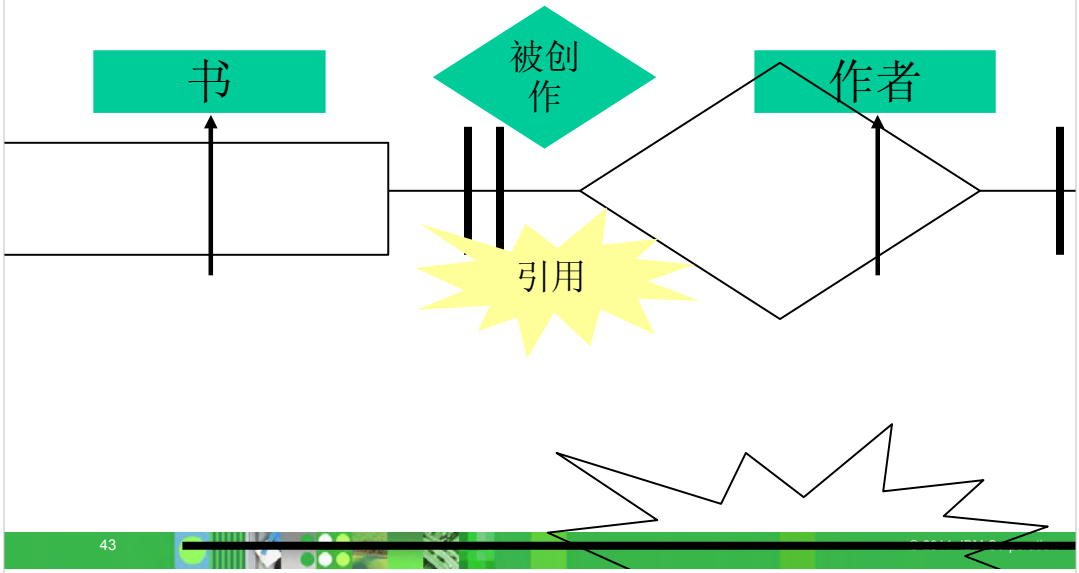
AUTHOR

Author_ID [PK]	Lastname	Firstname	Email	City	Country
NULL	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
NULL	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

如果我也以 NULL 取代 A4，又会怎么样呢？现在，哪个 NULL 将识别出哪一行呢？是不是有些困难？

因此，被约束，即实体完整性约束，重点强调在发挥主键作用的属性中不得存在空值。

引用完整性约束



引用完整性约束定义了表之间的关系，并确保这些关系是有效的。

如在本演示最开始所指出的那样，一本书要想存在，它就必须至少由一名作者编写。否则，一本书凭借自己的力量就存在，这种说法毫无意义。

关系数据库中的外键或者触发器有助于实施这种引用完整性约束。

语义完整性约束

AUTHOR

Author_ID [PK]	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

语义完整性约束

关于这一点，我们来看看 author 关系中的另一个例子。

语义完整性约束

AUTHOR

Author_ID [PK]	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	*12(*)&^23*	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

如果 city 一列含有一些垃圾值，例如该值，而不是 Toronto，它对列名称有什么意义吗？
语义完整性约束是指数据意义上的正确性。

域约束

AUTHOR

Author_ID	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

域约束规定了给定属性的允许值。

我们以关系 author 中的 country 属性来说明。

我们知道，这个特定的属性必须含有由两个字母组成的国家代码，例如 CA 代表加拿大， IN 代表印度，等等。

.

域约束

AUTHOR

Author_ID	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	*34*
A2	Ahuja	Rav	ra@ibm.com	Toronto	*34*
A3	Hakes	Ian	ih@ibm.com	Toronto	*34*
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

假设把 34 这样的值输入进来代表国家，有意义吗？没有意义，对吧？

当然，除非有另外一张表把以字母表示的 Country 代码映射为这些数字。但我们不想惹这样的麻烦了。

NULL 约束

AUTHOR

Author_ID	Lastname	Firstname	Email	City	Country
A1	Chong	Raul	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	Hakes	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

NULL 约束规定，属性值不得为 null 。

NULL 约束

AUTHOR

Author_ID	Lastname	Firstname	Email	City	Country
A1	Chong	*NULL*	rfc@ibm.com	Toronto	CA
A2	Ahuja	Rav	ra@ibm.com	Toronto	CA
A3	*NULL*	Ian	ih@ibm.com	Toronto	CA
A4	Sharma	Neeraj	ns@ibm.com	Chennai	IN
A5	Perniu	Liviu	lp@univ.com	Transilvania	RO

我们来看一看，如果在 lastname （姓）或 firstname （名）属性输入了空值会出现什么结果。

如果 lastname 或 firstname 是一样的话，这就为识别正确的作者带来困难。这就是说，作者必须有一个 lastname 和一个 firstname ，不得为 NULL 。

检查约束

BOOK

Title	Edition	Year	Price	ISBN	Pages	Aisle	Description
数据库原理	1	2010	24.99	978-0-98662 83-1-1	300	DB-A02	向你教授数据库原理
DB2 Express-C 入门	1	2010	24.99	978-0-98662 83-5-1	280	DB-A01	利用 DB2 Express-C （DB2 的免费版本），向你教授 DB2 的核心内容

检查约束通过限制能够为某个列所接受的值来保证域完整性。

Author 关系中没有一个合适的属性用来方便地解释检查约束。因此，我以 Book 关系为例来说明。

检查约束

BOOK

Title	Edition	Year	Price	ISBN	Pages	Aisle	简介
数据库原理	1	2010	24.99	978-0-98662 83-1-1	300	DB-A02	向你教授数据库原理
DB2 Express-C 入门	1	*2015*	24.99	978-0-98662 83-5-1	280	DB-A01	利用 DB2 Express-C （DB2 的免费版本），向你教授 DB2 的核心内容

year （年代）属性是特定的一本书出版时的年代。

如果某个 year 值比当前的 2011 年还大，那有意义吗？

议题

- 概述
- 实体 - 关系图
- 关系模型
  - 实体 - 关系图
  - 关系的类型
  - 把实体映射为表
  - 关系模型的概念
  - 关系模型约束
  - 规范化

规范化

在数据库设计中消除冗余的过程 redundancies

示例：

考虑下表，它列出了每名员工的所有任务：

ID	Name	Office City	Extension	Task
1	John S	Toronto	54213	Planning
1	John S	Toronto	54213	Marketing
1	John S	Toronto	54213	Testing
2	Susan P	New York	59867	Marketing
3	Jennifer L	Chicago	59415	Planning
3	Jennifer L	Chicago	59415	Testing

问题：

如果 John 转移到另一座城市，与他有关的所有项目都必须更新。

规范化用来减少冗余。有多种形式的规范化，例如 3NF （第三范式）、 BCNF (Boyce-Codd 范式)，等等。每种范式代表了数据必须受约束的程度。这也能够让你确定你是否没有足够的关系来充分地把模式中的所有键连接起来。

有这样的可能性：一个数据集无法达到某种范式，因为没有足够的能力为该数据集创建约束。

在本例中，你可以看到，如果你需要对该表稍微进行一些修改，相关的所有记录也都需要修改。例如，如果 John 不在 Toronto 办事处工作了，你就需要进行三次修改。

在下一个幻灯片中，将向你介绍规范化如何帮助你减少表中的冗余。

规范化（续）

无冗余，无异常，无信息丢失

Employee Table

ID	Name	Office City	Extension
1	John S	Toronto	54213
2	Susan P	New York	59867
3	Jennifer L	Chicago	59415

Task Table

ID	Task
1	Planning
2	Marketing
3	Testing

Employee Tasks Table

Employee ID	Task ID
1	1
1	2
1	3
2	2
3	1
3	3

在这个例子中可以看到，一张大表被分解为多个小表，这些小表中的字段也较少；但是，每个小表中都有大表中的外键。该外键与小表中的主键联系到一起，然后，再把这些主键联系到一起，从而重建出与大表一样的全部选项。

你可以看到，通过这种方式，所有的值仍然能够适当地共存，而且如果你在什么地方进行了修改，你不再需要进行前面的那么多修改，因为减少了冗余。例如，如果 John 不在 Toronto 办事处工作了，现在只需要做一处更新就可以了。

如果你对本课程所讨论的内容有什么技术方面的问题的话，可以利用 db2university.com 课程 AA001EN 中的论坛。 同学、教员以及 IBM 人员可以帮助你！

Information Management

IBM

DB2 入门

IBM 信息管理云计算能力中心
IBM （加拿大）研究院

议题

- **DB2** 服务器版本、客户端和驱动程序
- **DB2 Express-C** 概述
- **DB2** 命令行处理器（演示）
- **DB2** 环境
- 配置 **DB2**
- 使用脚本
- 接入 **DB2** 服务器
- 数据移动实用工具

支持性阅读材料和视频

阅读材料

DB2 Express-C 入门电子书

- 第 1 章：什么是 DB2 Express-C?
- 第 3 章：DB2 安装
- 第 4 章：DB2 环境
- 第 5 章，第 5.10 节：脚本编写
- 第 6 章，第 6.3 节：DB2 存储模式
- 第 7 章：DB2 客户端连接
- 第 9 章：数据移动实用工具

视频

db2university.com 课程 AA001EN

- 第 2 课：DB2 入门
- 第 4 课：使用脚本
- 第 5 课：接入 DB2 服务器

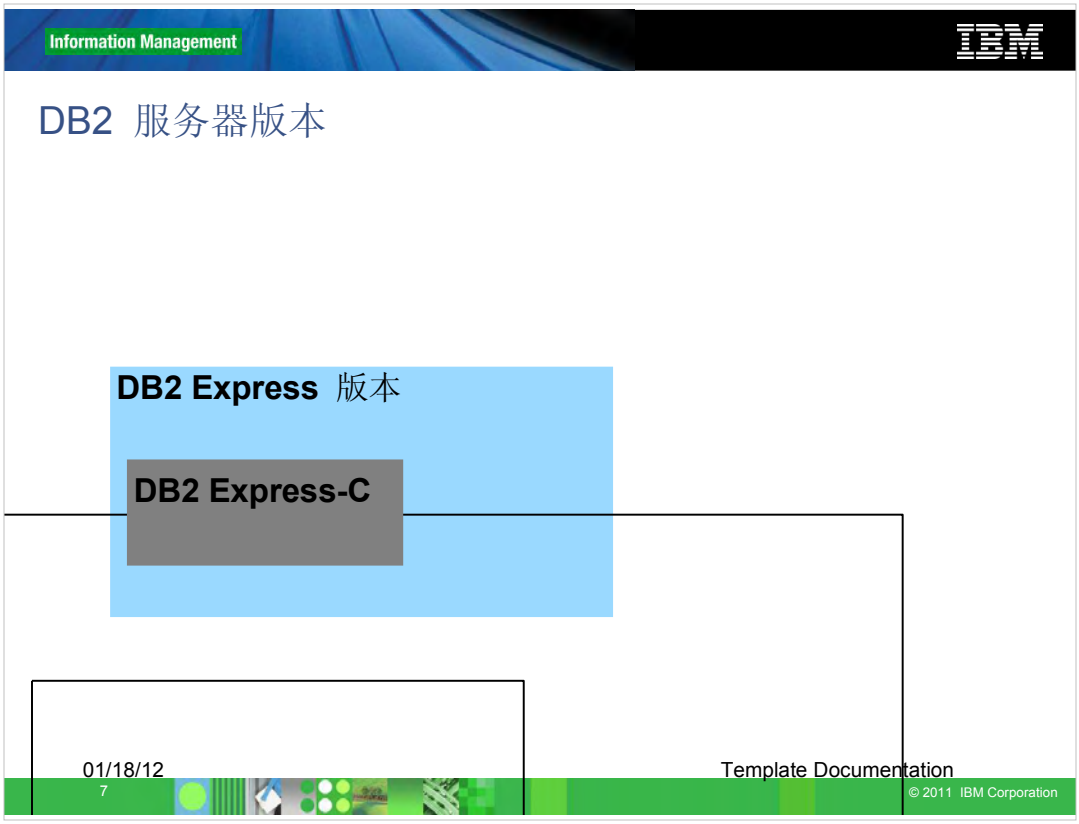
议题

- **DB2 服务器版本、客户端和驱动程序**
- DB2 Express-C 概述
- DB2 命令行处理器（演示）
- DB2 环境
- 配置 DB2
- 使用脚本
- 接入 DB2 服务器
- 数据移动实用工具

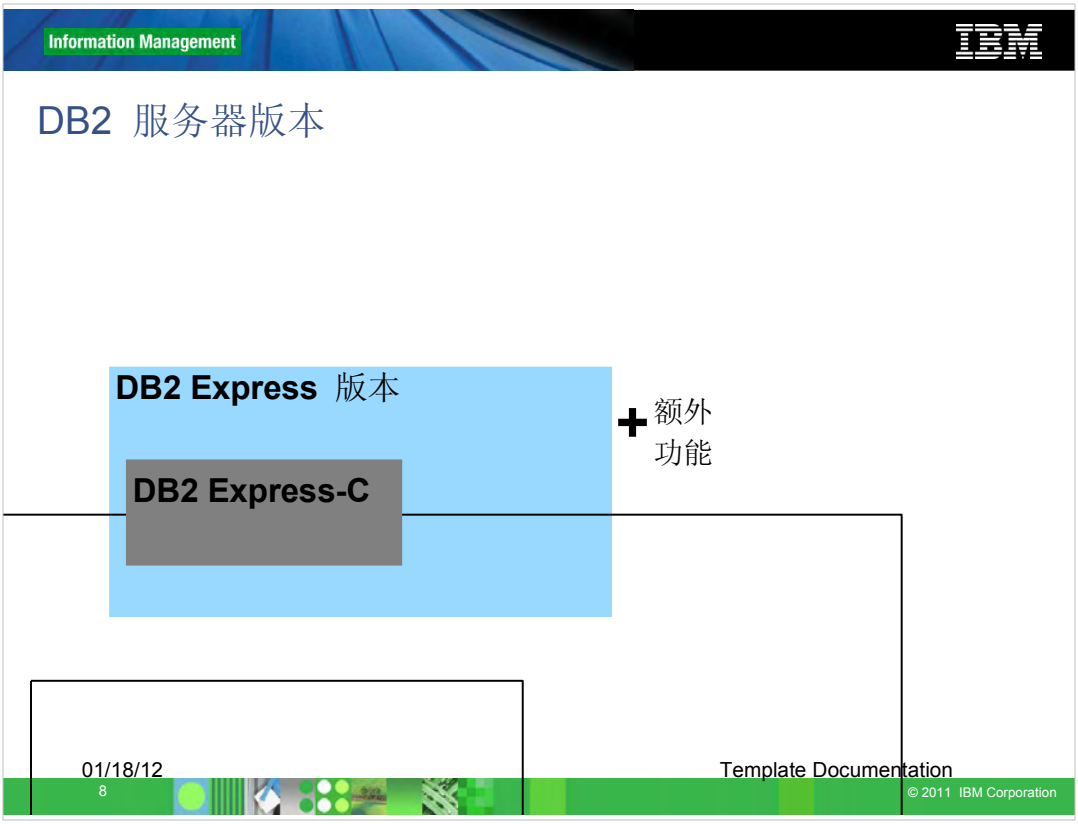
DB2 服务器（也被称为 DB2 数据服务器或者 DB2 数据库服务器）是供你用来安装某种 DB2 服务器版本的地方。 DB2 服务器接受来自本地及远程 DB2 客户端的外来连接。 DB2 是采用 C/C++ 语言开发的，其工具是采用 Java 开发的。这适用于 Linux 、 UNIX 、 Windows 及 Mac 上所有的 DB2 版本。有多个不同的 DB2 服务器版本，它们互为基础：

DB2 Express-C 是 DB2 的核心。

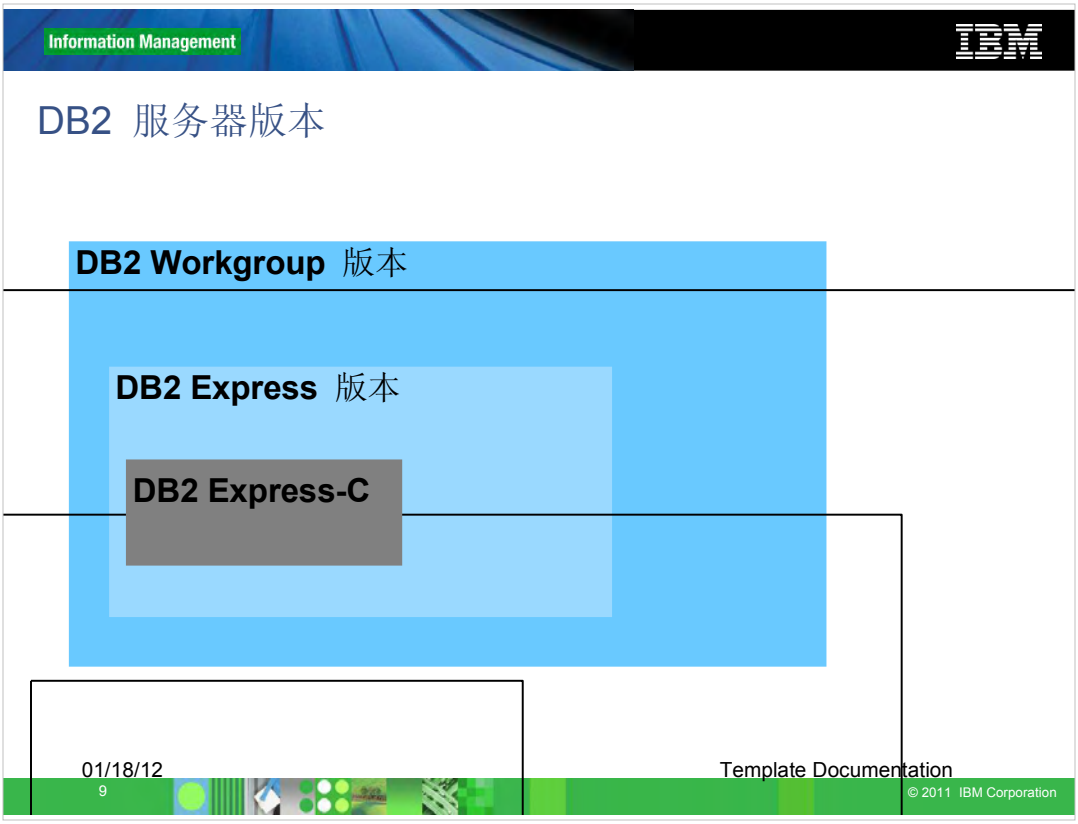
然后我们向 DB2 Express-C 添加更多功能，



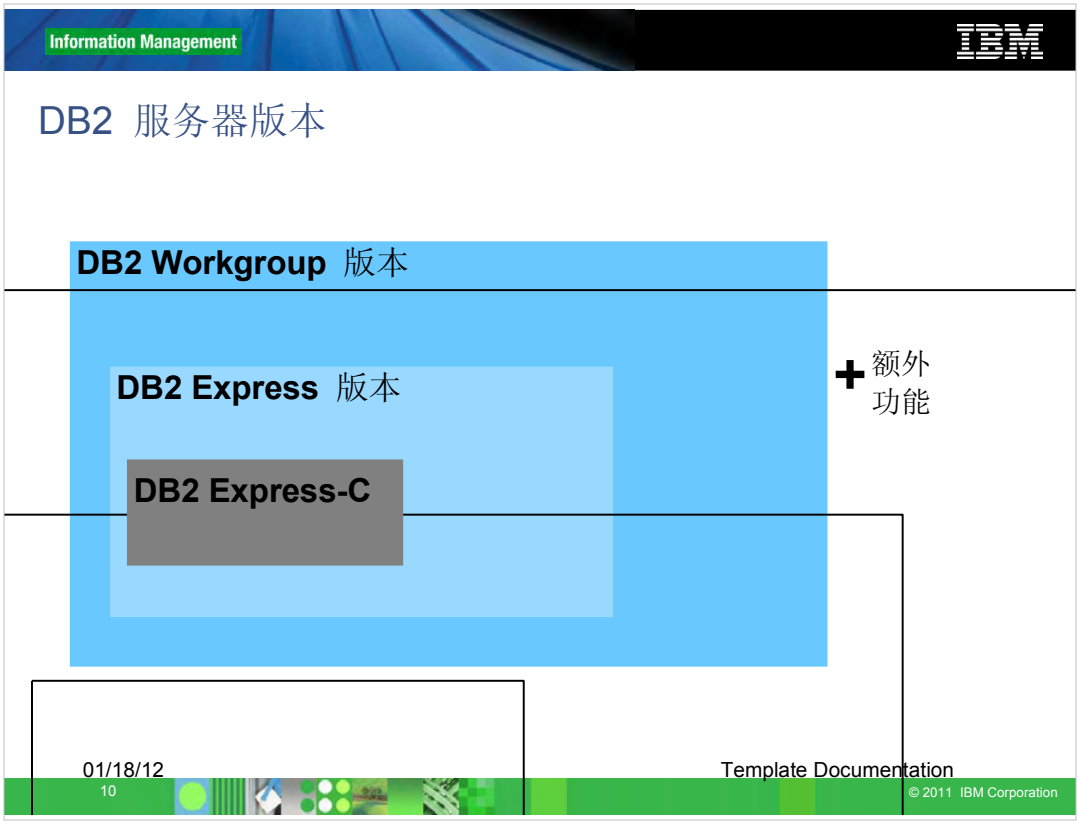
这就形成了 DB2 Express 。



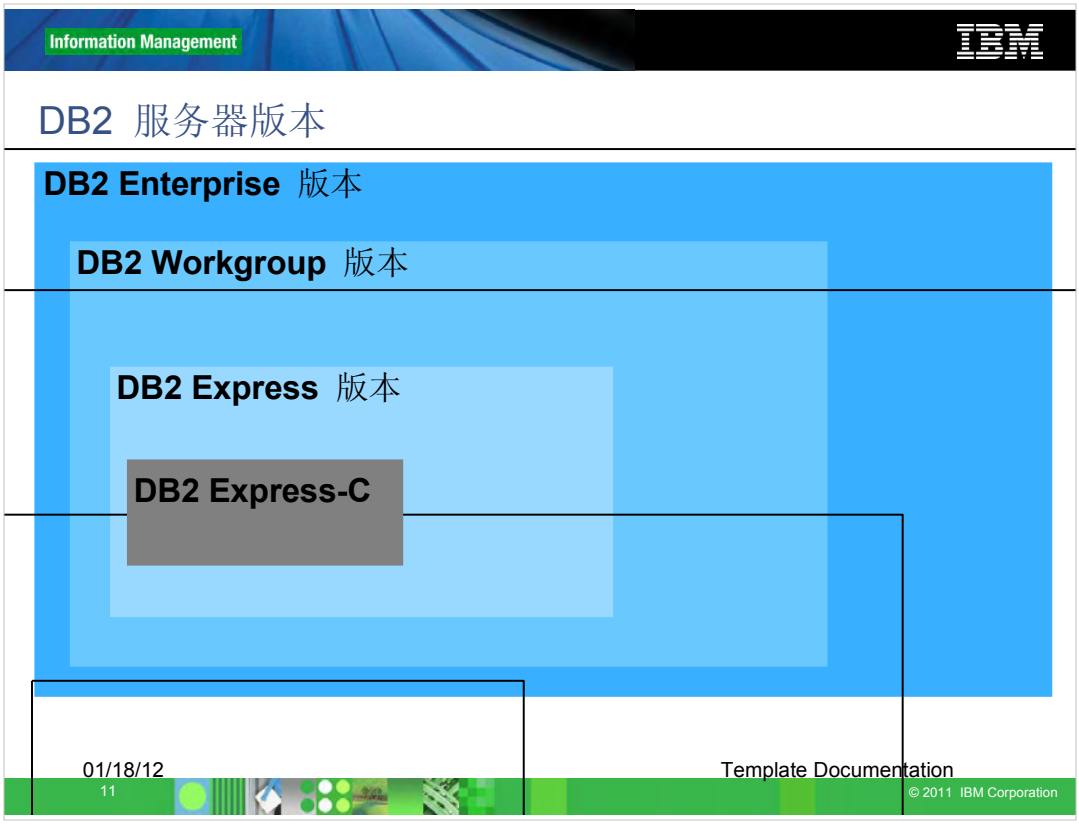
当我们再向 DB2 Express 添加更多功能，



就形成了 DB2 Workgroup，

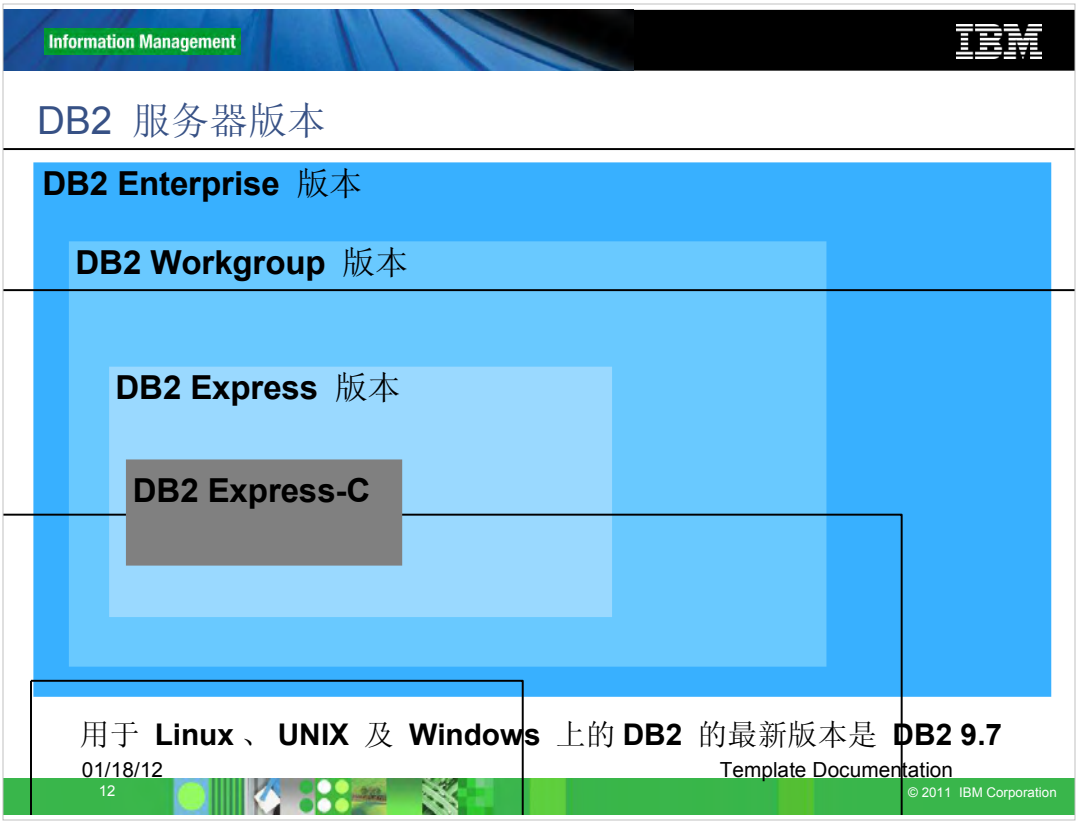


然后我们再向 DB2 Workgroup 添加更多功能，

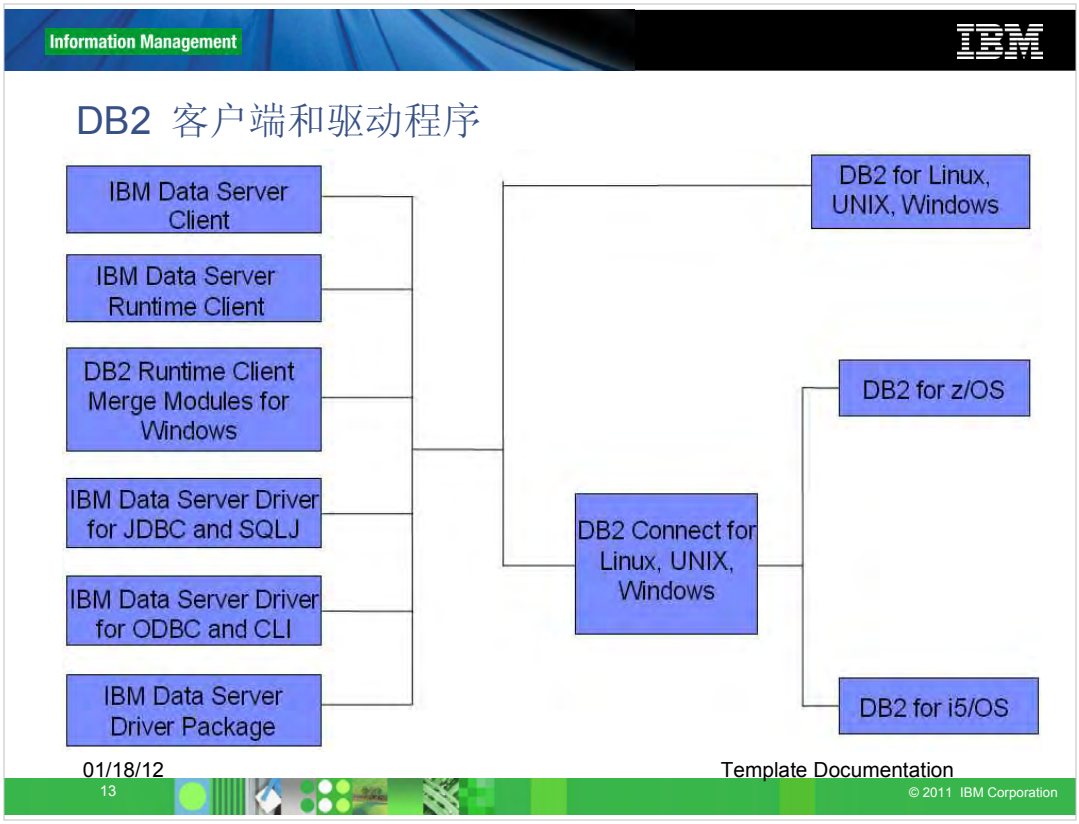


就形成了 DB2 Enterprise 版。选择何种服务器版本取决于你的需求。版本只是我们对产品打包的一种方式；它能够让用户根据他们的需求选择他们所需要的软件包。例如，对于小型创业公司，最佳选择或许是 DB2 Express-C，因为它是免费的。另一方面，对于地位稳固的大型企业，如果他们寻求诸如压缩、高可用性、数据库分区等功能，他们的最佳选择或许是 DB2 Enterprise。如前面所示，DB2 各个版本是互为基础构建的，均采用了相同的 C/C++ 代码。因此，用来创建 DB2 Express-C 的代码与其他版本是相同的。这意味着，如果你掌握了 DB2 Express-C，你也就学会了所有其他的 DB2 版本，因为 DB2 Express-C 是核心。而且，如果你为 DB2 Express-C 开发了一套应用，该应用无需修改就可以用于所有其他的 DB2 版本；原因一样，因为 DB2 Express-C 是核心。

所有的 DB2 服务器版本均包含一套客户端组件以及众多 GUI 工具。一台 DB2 服务器可以用作另一台 DB2 服务器的客户端系统。例如，如果从 DB2 Express-C 服务器接入 DB2 Enterprise 版本的服务器，DB2 Express-C 服务器基本上用作客户端。



面向 LUW 的 DB2 最新版本是 DB2 9.7



DB2 客户端和驱动程序提供了连接 DB2 服务器并与之协同工作的必要功能。驱动程序可能随同DB2客户端一同提供，也可能单独提供。根据编程接口的不同，你可能仅需要安装一个DB2驱动程序，也可能需要安装整个DB2客户端。例如，JDBC Type 4 应用只需要安装 JDBC驱动程序就可以接入DB2 服务器，你不需要安装整个客户端。所有的DB2服务器都包含DB2客户端组件；因此，一旦你安装了一台服务器，它既可以接入本地数据库（在同一台服务器上创建的数据库），也可以接入远程数据库（在其他服务器上创建的数据库）。DB2 客户端和驱动程序都是免费的，可以从DB2 Express-C网站上下载。

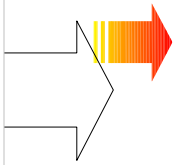
在这张图的左上角，列出了3种客户端。名称以“IBM Data Server”（IBM数据服务器）起始的客户端不但可以与DB2一同工作，而且也可以与其他IBM数据服务器一同工作，例如 Informix。IBM Data Server Client（IBM数据服务器客户端）是最完整的，它含有众多图像化工具，以及不同编程语言需要的所有驱动程序。IBM Data Server Runtime Client（IBM数据服务器运行时间客户端）是一套包含基本功能的轻量级客户端，也包含了驱动程序。DB2 Runtime Client Merge Modules for Windows（面向Windows的DB2运行时间客户端合并模块）主要用于把DB2运行时间客户端嵌入为Windows应用装置的一部分。在这张图的左下角，列出了3种驱动程序：

- IBM Data Server Driver for JDBC and SQLJ（面向JDBC及SQLJ的IBM数据服务器驱动程序）：能够让Java应用接入 DB2 服务器，而无需安装客户端

- IBM Data Server Driver for ODBC and CLI（面向ODBC及CLI的IBM数据服务器驱动程序）能够让ODBC及CLI应用接入 DB2 服务器，而无需安装客户端

- IBM Data Server Driver Package（IBM数据服务器驱动程序包）包含了一套针对 Windows的驱动程序，除了支持 ODBC、CLI 和开源之外，还能够支持 .NET 环境。在右侧，我们列出了DB2服务器。本视频系列的焦点是面向Linux、UNIX和Windows的DB2。为了接入大型机 (DB2 for z/OS) 及中等服务器 (DB2 for i5/OS) 上的DB2服务器，你需要购买一套称为 DB2 Connect 的软件。

议题



- DB2 服务器版本、客户端和驱动程序
- **DB2 Express-C 概述**
- DB2 命令行处理器（演示）
- DB2 环境
- 配置 DB2
- 使用脚本
- 接入 DB2 服务器
- 数据移动实用工具

什么是 DB2 Express-C?

DB2 数据库服务器的免费版本

该名称中的 **“ C”** 代表 **“ Community”** （社区）

免费开发、部署和发布 **...** 无任何限制！

- 无用户数限制
- 无 DB2 实例限制
- 无数据库规模限制

DB2 Express-C 是 IBM 的 DB2 数据库服务器的免费版本。此名称中的 'C' 代表 “Community”（社区），因此，DB2 Express-C 是为开发人员、学生、教师、专业人员及其他对数据库感兴趣的人员组成的社区而创建的。它可以免费开发、部署和发布，无任何限制。免费开发意味着，如果你是一名应用开发人员，并需要为你的应用寻找一部数据库，你可以免费使用DB2 Express-C。免费部署意味着，你可以在生产中免费使用DB2 Express-C。例如，假设你有一个销售玩具的电子商务网站。你所销售的玩具的图片以及价格、描述等内容都可以存储在DB2 Express-C数据库中。免费发布意味着，你可以把 DB2 Express-C 嵌入到你的应用中。当你销售你的应用时，你可以随着你的应用一同提供及发布 DB2 Express-C；你无需向 IBM 支付任何东西。因此，DB2 Express-C 可以帮助你赚钱！“无限制” 是指如下事实：对数据库用户的数量没有限制，对你创建的DB2实例的数量没有限制，对数据库规模没有限制，就是说，你的数据库可以增长至任何规模（其他供应商提供的其产品的免费版本通常会设定 4GB 的数据库规模限制）。

DB2 Express-C 能够运行于哪些系统中？

Windows

Windows 工作站平台 (XP, Vista, Windows 7)
Windows 服务器平台 (2003, 2008)

Linux

Red Hat 、 Suse 、 Ubuntu 等等

Mac OS X (beta)

Solaris

支持 **32** 位和 **64** 位系统

DB2 Express-C 可以运行在 Windows（XP、Vista 和 Windows 7）上作为工作站平台，也可以运行在 Windows Server 2003 和 2008 上。它可以运行在 Linux 上（Red Hat、Suse、Ubuntu，等等）。还有关于 Mac OS X 和 Solaris 的 beta 版本。它在 32 位及 64 位系统上都得到支持。

DB2 Express-C 有哪些要求？

最低要求

内存：

256MB， 512MB (DB2 GUI 工具)， 1GB （推荐）

磁盘：

取决于创建的数据库的数量和规模

在 Linux 上，建议在 /tmp 中有 2GB 空间

最高要求：

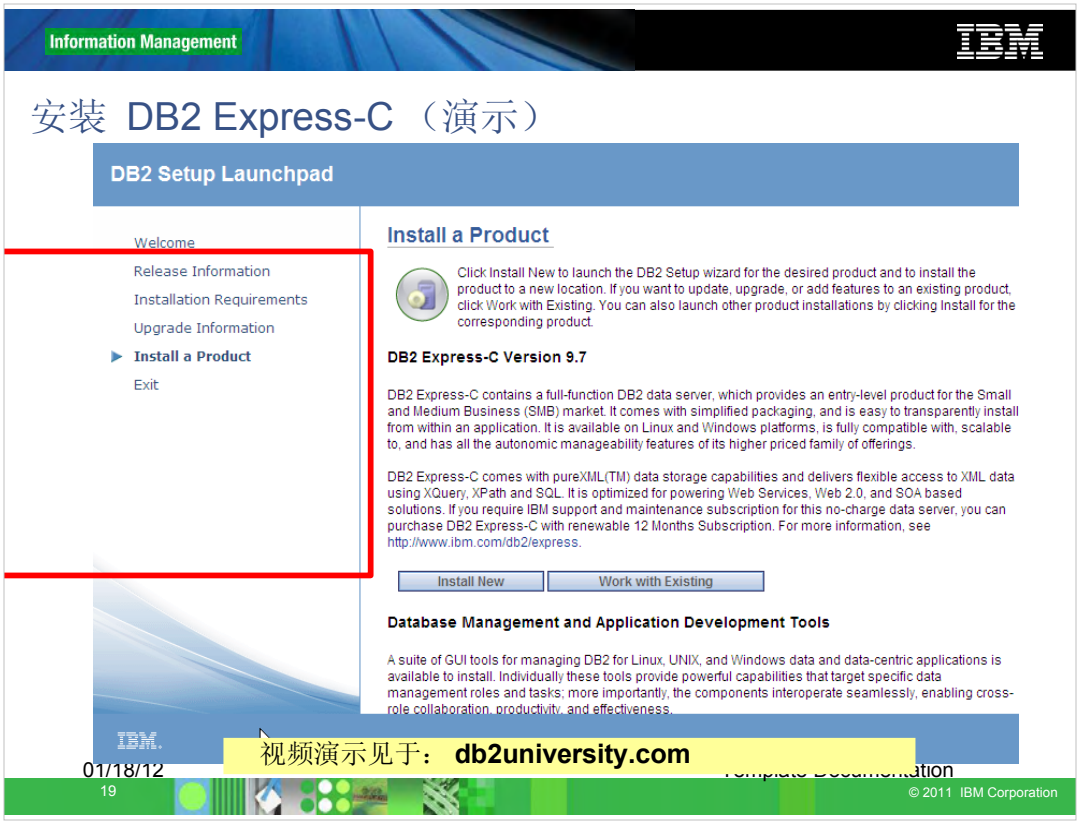
任何规模的系统

最多使用 2 个内核及 2 GB 内存

DB2 可以安装在当今的大多数系统中。就最低内存要求而言，DB2 Express-C 可以在低至 256Mb 内存的系统中运行。如果要使用 DB2 图形化工具，你至少需要 512Mb。推荐至少配备 1GB 内存。就最低磁盘要求而言，这取决于你打算创建的数据库的数量和长度。在 Linux 中，推荐 /tmp 中有 2GB 空间。

就最高要求而言，DB2 Express-C 可以安装到任何规模的系统中。这意味着系统可以非常大（例如，16 个 CPU 内核及 32GB 内存）；但 DB2 Express-C 将自动限制其自己使用 2 个核心和 2GB 内存。其他的 DB2 版本允许使用更多的资源。

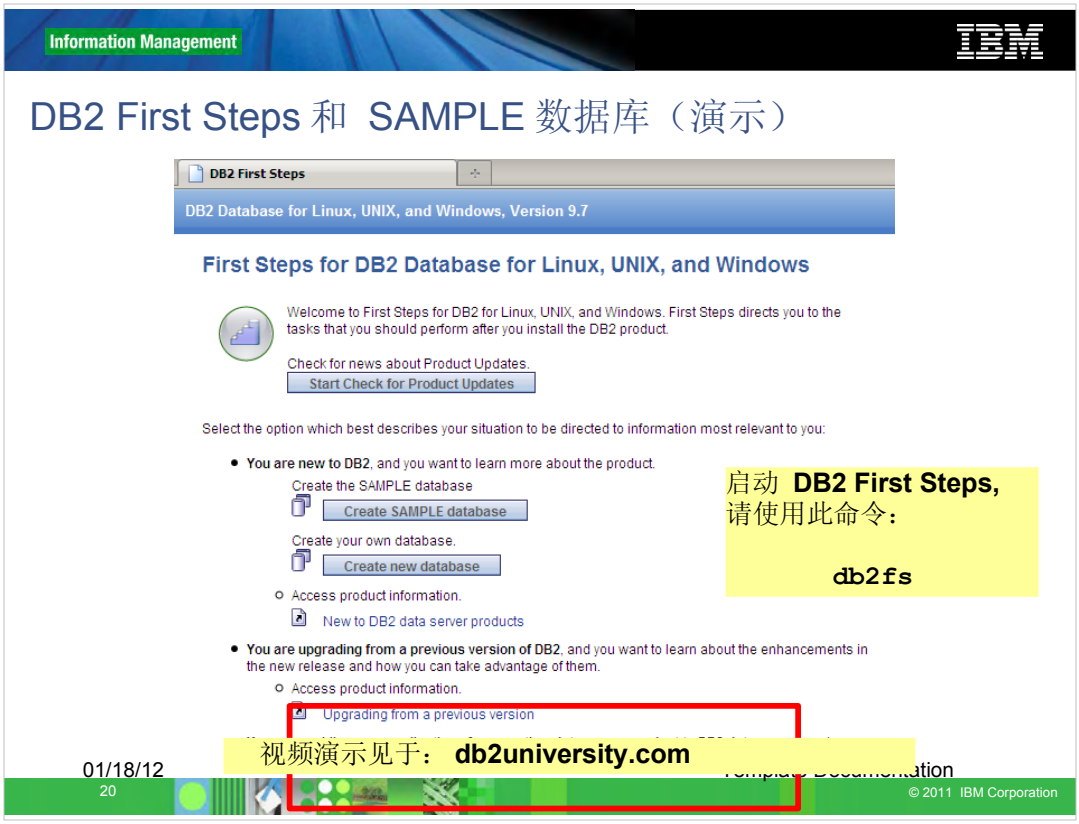
观看 db2university.com 上的视频，了解如何下载和安装 DB2 Express-C



观看 db2university.com 上的视频，了解如何下载和安装 DB2 Express-C

注意，这个屏幕截图上显示了两个按钮：“安装新产品”（ Install New ）和“使用现有产品”（ Work with Existing ）。如果这是你首次安装 DB2 Express-C ，你只会看到一个按钮：“安装新产品”。但是，如果你已经在你的计算机中安装了 DB2 ，那么，第二个按钮“使用现有产品”也会出现。这基本上意味着，你可以安装多个版本的 DB2 （或者把同一版本安装多次），或者在不同的修补包（ fixpack ）级别上安装 DB2 。例如：

DB2 9.5, DB2 9.7, DB2 9.7 FP2, DB2 9.7 FP4，等等



安装DB2之后，你的浏览器将自动启动 “DB2 First Steps”（DB2初始步骤）。如果未启动的话，你可以使用命令 “db2fs” 从 DB2 命令窗口或 Linux 外壳中手动启动。

如果你在使用 Firefox，你可能会被提示创建 DB2_FIRSTSTEPS 配置文件。我们建议你创建该配置文件，这样你就能够使用 DB2 First Steps 的全部功能了。如果你不打算创建，可以点击“Do not create profile”（不创建配置文件），或者简单地关闭该窗口就可以了，就像我这里所做的那样。First Steps 工具非常易于理解。许多人第一次都使用它来创建 SAMPLE 数据库。

SAMPLE 数据库是随同 DB2 一起提供的数据库，因此，你可以“摆弄”它。SAMPLE 数据库可能在安装过程中已经为你创建好了。你可以从 DB2 命令窗口中利用以下命令验证它是否已经创建：

db2 connect to sample

如果你得到一条错误消息，你就需要手工创建它。这时，请返回 DB2 First Steps 工具，然后点击 “Create SAMPLE database”（创建SAMPLE数据库）按钮。将出现一个窗口。选择第二个选项“XML and SQL objects and data”（XML和SQL对象和数据）。这 将创建 SAMPLE 数据库，并包含 XML 数据，以后将使用这些数据测试 DB2 如何与 XML 协调工作。点击 OK（确认）即开始创建 SAMPLE 数据库。这需要几分钟的时间。在创建出 SAMPLE 数据库之后，你可以检验能否接入它并从中检索信息。在 DB2 命令窗口或 Linux 外壳中，键入：

db2 connect to sample

db2 “select * from employee”

如果两个语句都得到成功执行（SELECT应当返回若干行），你就已经顺利创建了SAMPLE 数据库。你也可以使用命令行命令“db2sampl”来创建 SAMPLE 数据库。

议题

- DB2 服务器版本、客户端和驱动程序
- DB2 Express-C 概述
- **DB2 命令行处理器（演示）**
- DB2 环境
- 配置 DB2
- 使用脚本
- 接入 DB2 服务器
- 数据移动实用工具

在 Windows 中， DB2 命令窗口看起来类似常规的 MS-DOS 窗口，但窗口标题为 “ DB2 CLP”。你可以从该窗口中执行 DB2 命令。在 Linux 中，没有 DB2 命令窗口，但可以使用 Linux 外壳（只要你在登录时使用的用户名已经将其环境设置为能够使用 DB2 即可。例如，使用 DB2 实例所有者的名称，一般为 db2inst1 ）。在 Linux 和 Windows 环境中，也都有命令行处理器 (CLP) 。在 DB2 命令窗口 /Linux 外壳与 CLP 之间有什么区别？可以看看提示符。如果提示符是 “ db2 =>”，你就处于 CLP 之中，这是你不需要为任何 DB2 命令增加 “ db2” 的前缀。

例如：

db2 => connect to sample

另一方面，如果提示符来自操作系统，你就处于 DB2 命令窗口 /Linux 外壳中，这时你就需要为 DB2 命令增加 “ db2” 的前缀。

例如：

C:\> db2 connect to sample

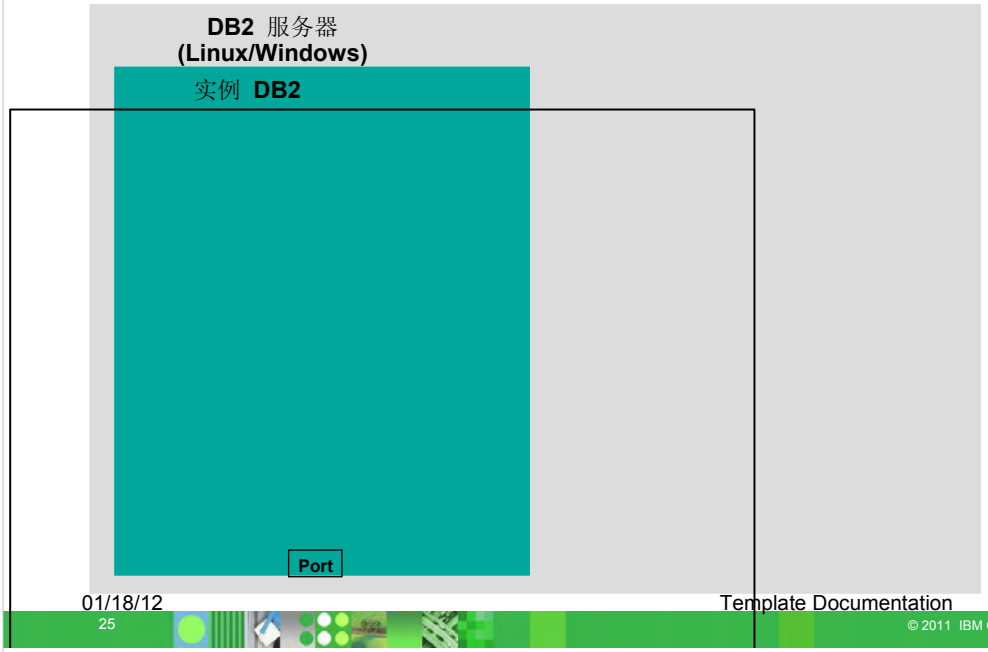
议题

- DB2 服务器版本、客户端和驱动程序
- DB2 Express-C 概述
- DB2 命令行处理器（演示）
- **DB2 环境**
- 配置 DB2
- 使用脚本
- 接入 DB2 服务器
- 数据移动实用工具

DB2 环境

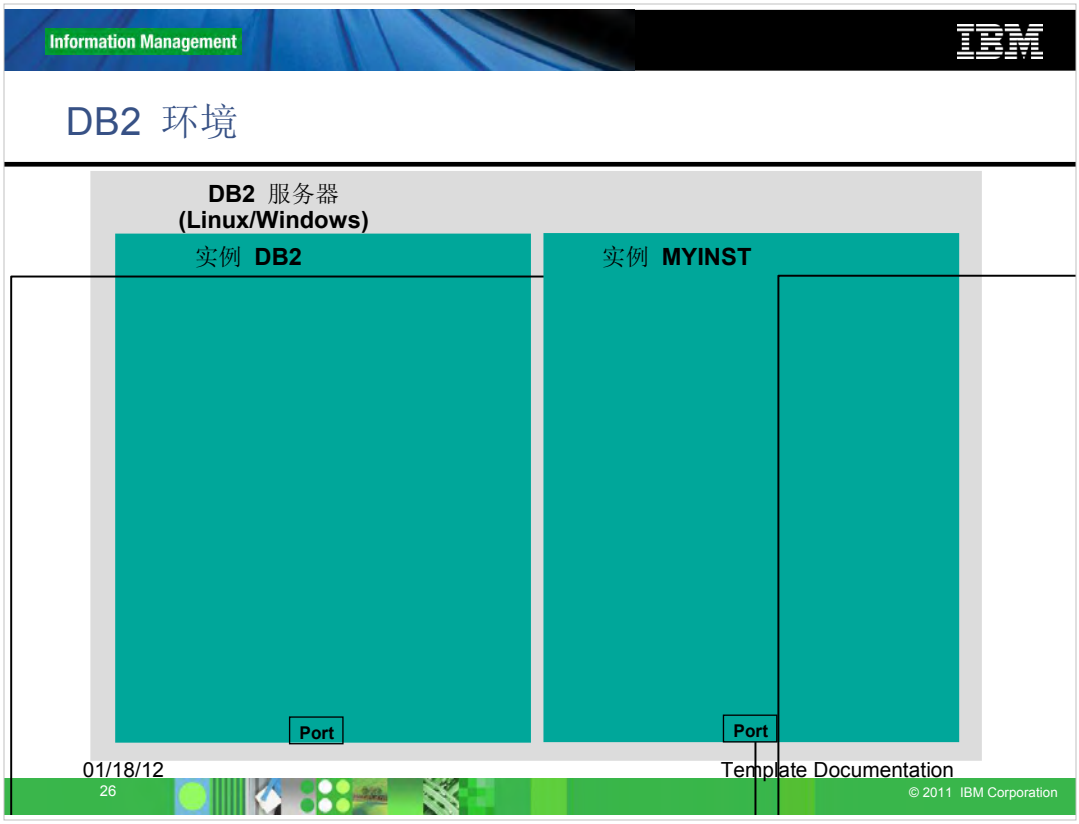
图中的方框代表了安装 DB2 Express-C 9.7 之后的 DB2 数据服务器。

DB2 环境



作为在 Windows 中安装的一部分，一个名为 DB2 的默认实例（在 Linux 中为 db2inst1 ）被创建。这由图中左侧的绿色方框来代表。一个实例就是一个用来运行应用及创建数据库的独立环境。你可以在数据库服务器上创建多个实例，并可以把它们用于不同的目的。例如，可以把一个实例用于容纳生产目的的数据库，而把另一个实例用作测试环境的数据库，再把另一个实例用作开发环境。所有这些实例都是独立的，就是说，在一个实例中进行操作不会对其他实例造成影响。

要想创建新的 DB2 实例，可以使用命令 db2icrt <instance name>，其中 <instance name> 应当由任何 8 个字母构成的名称来代替。例如，为了创建实例 MYINST，我们应当使用这个命令： db2icrt myinst.



该图在右侧示出了一个新的实例，名为 myinst，用单独的绿色方框表示。

注意，每个实例都有一个独特的端口号码。这有助于在你利用 TCP/IP 从远程客户端接入给定实例中的数据库时对不同实例加以区分。如果你使用 DB2 命令窗口，你可以通过在 Windows 执行下面的操作系统命令使任何 DB2 实例成为活动的实例：

```
set db2instance=myinst
```

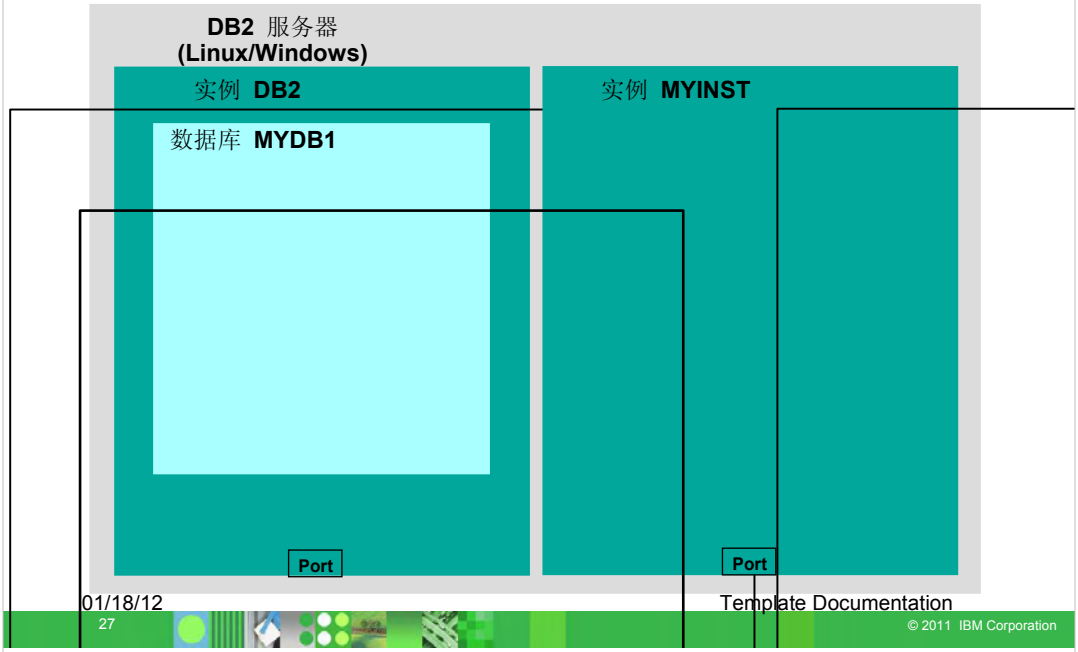
注意，等号 (=) 前后没有任何空格。在本例中，如果你现在从命令窗口创建数据库，该数据库将在实例 MYINST 中创建。

要列出你的系统中的实例，可以运行下面的命令：

```
db2ilist
```

在 linux 中，一个实例必须与一个 Linux 操作系统用户相匹配；因此，要想在实例之间切换，你只需切换用户即可。该用户被称为实例所有者。你可以利用另一个实例所有者用户的身份退出（ logoff ）及登录（ logon ），或者使用 su 命令。

DB2 环境



为了在活动的实例中创建数据库，可以从 DB2 命令窗口中发出以下命令：

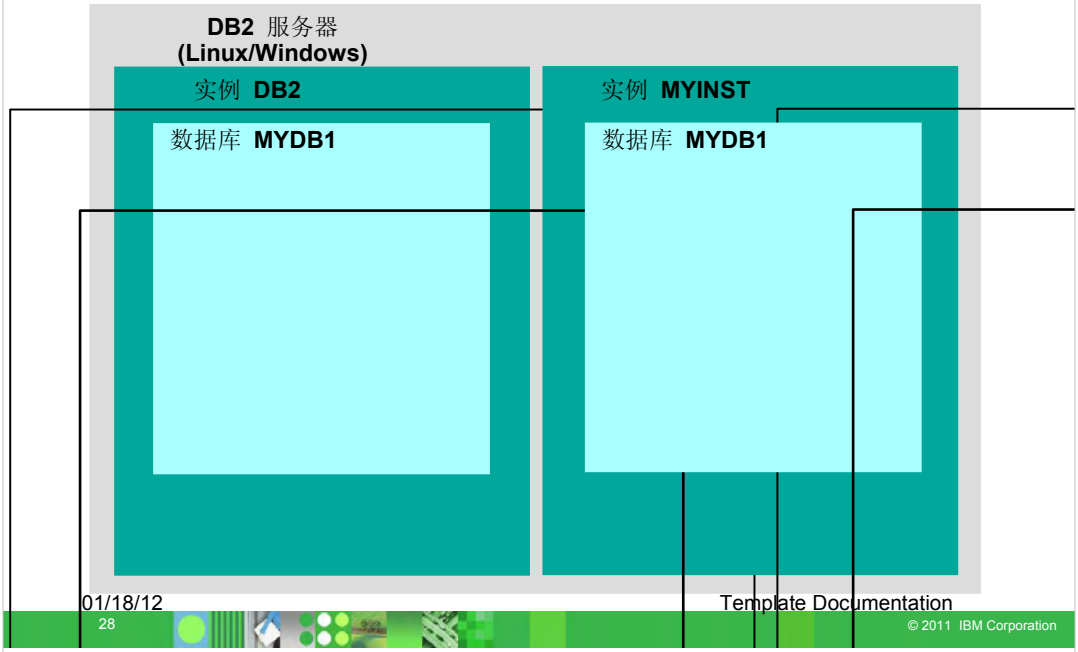
db2 create database MYDB1

为了列出所有已创建的数据库，可以运行下面的命令：

db2 list db directory

在任何一个实例中，你可以创建多个数据库。数据库是表、视图、索引等对象的集合。数据库是独立的单元，因此，不能与其他数据库共享对象。该图显示出在实例 DB2 内创建的数据库 MYDB1 。

DB2 环境



如果我们打算创建另一个同名的 (MYDB1) 数据库，但在实例 MYINST 中创建，可以从 DB2 命令窗口中发出以下命令：

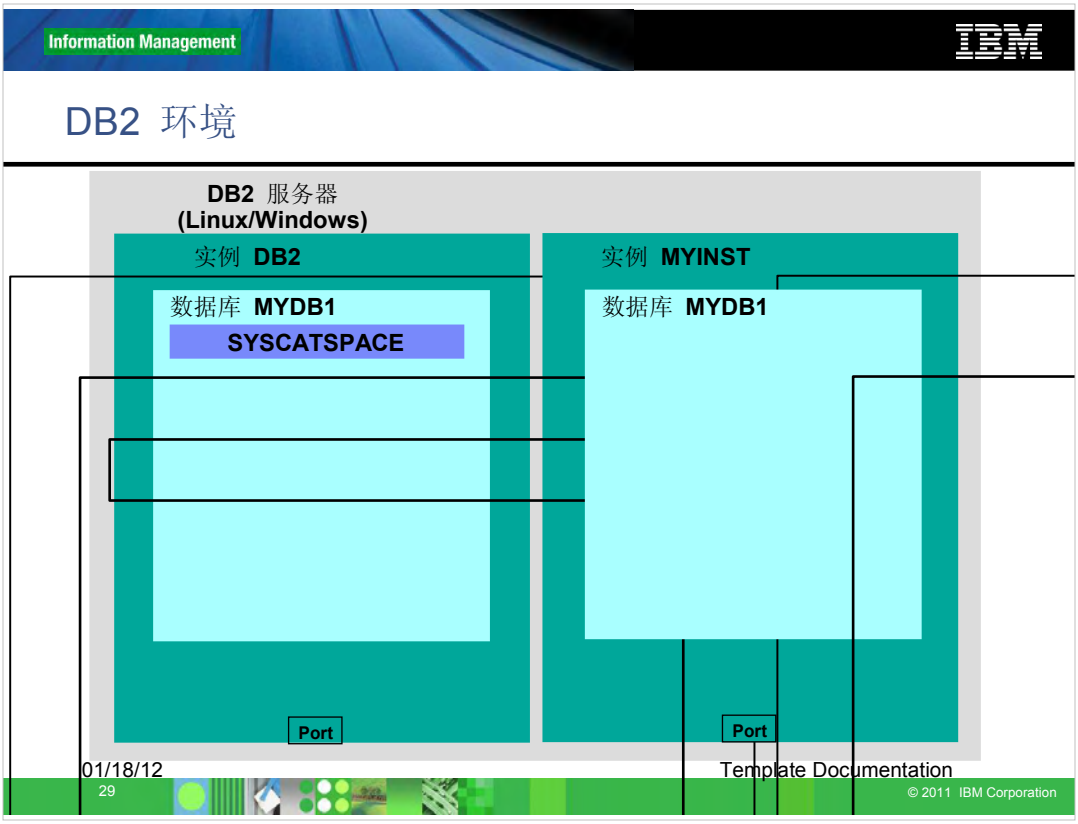
db2 list db directory

set db2instance=myinst （在 Linux 中： su - myinst ）

db2 create database MYDB1

set db2instance=db2 （在 Linux 中： su - db2inst1 ）

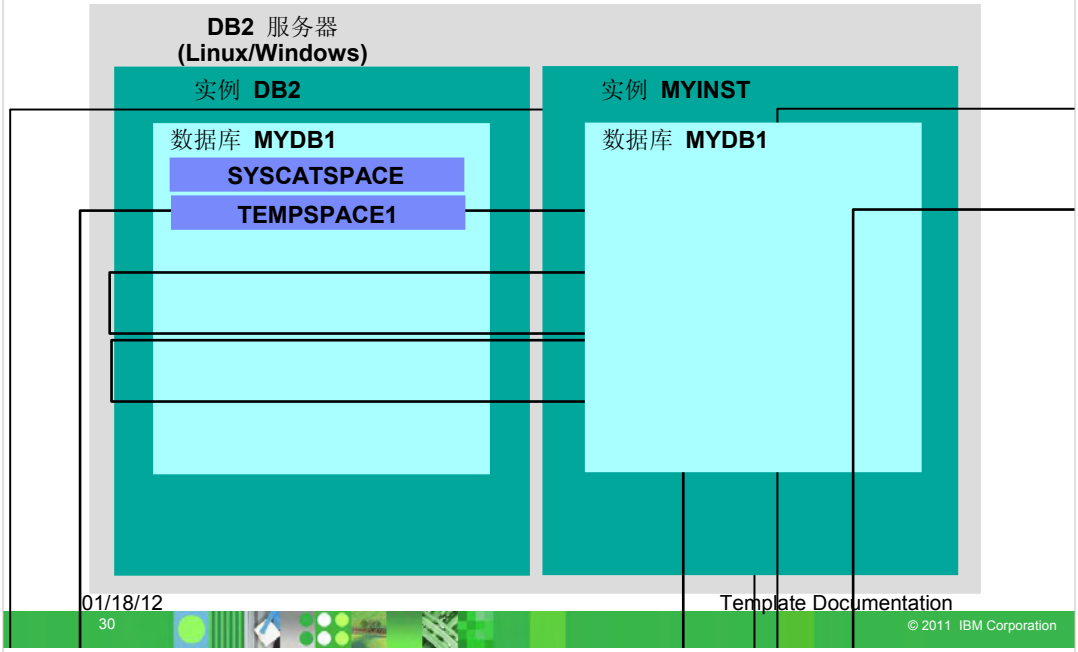
该图示出在实例 MYINST 中创建了新的数据库 MYDB1 。



在创建一个数据库时，同时默认创建多个对象：表空间、表、缓冲池以及日志文件。创建这些对象需要花费一些时间，这就是创建数据库的命令需要几分钟时间来处理的原因。该图在左侧示出了默认创建的此类表空间之一。表空间将在今后进行详细讨论。

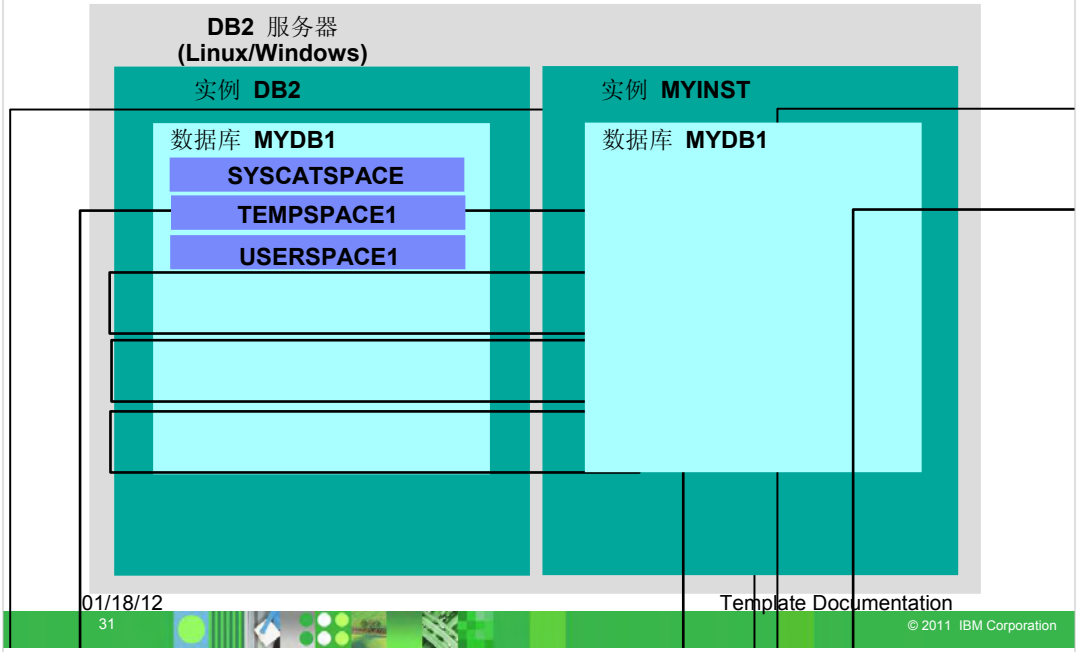
表空间 SYSCATSPACE 中含有 System Catalog （系统编目）表。System Catalog 在其他关系数据库管理系统中被称为数据字典。它主要包含一些系统信息，这些信息不能修改和删除；否则的话，数据库将无法正常工作。

DB2 环境



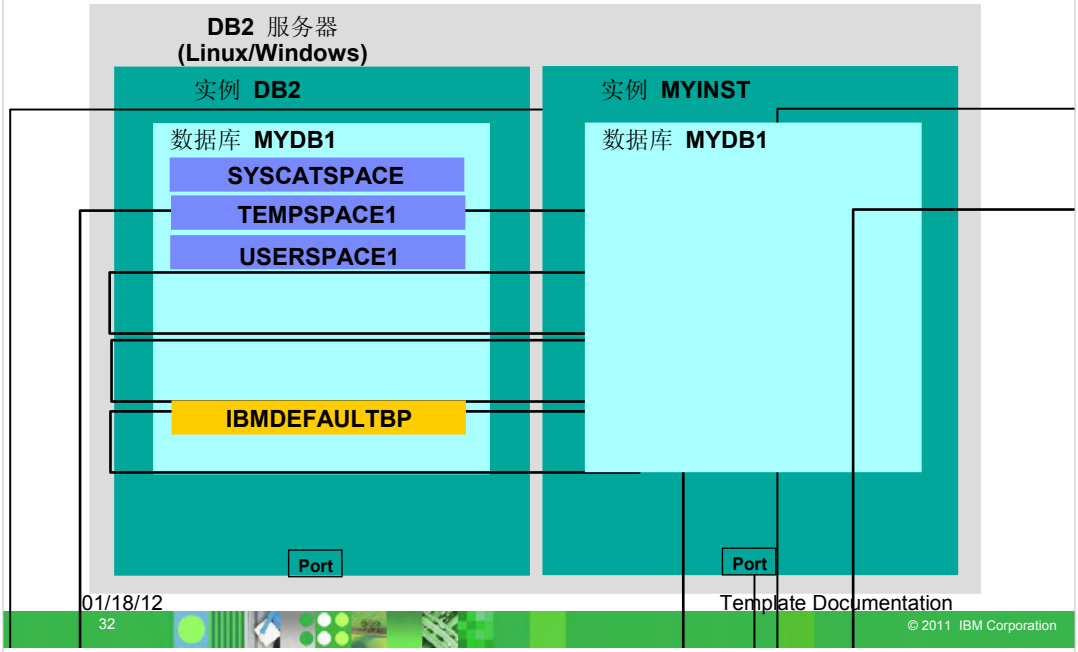
表空间 TEMPSPACE1 由 DB2 用于其需要额外空间进行某些操作之时，例如进行排序操作。这是一个 SYSTEM （系统）临时表空间。在任何时候，都必须至少有一个系统临时表空间。

DB2 环境



表空间 USERSPACE1 通常用于在创建表时未指定表空间的情况下存储用户数据库表。

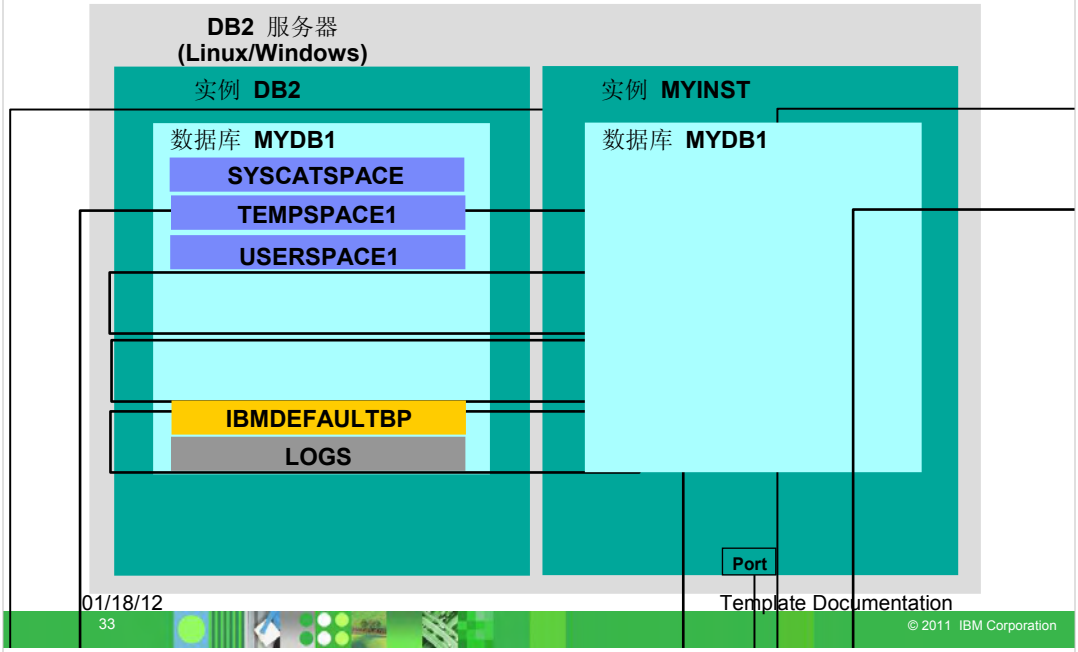
DB2 环境



下面我们来看看缓存池，它也是在创建数据库时默认创建的。该默认缓存池的名称是 IBMDEFAULTBP 。

缓冲池基本上是数据库使用的内存缓存。你可以创建一个或多个缓冲池，但总应当有一个缓冲池的页面大小与现有表空间的页面大小相一致。页面是 DB2 用于处理你的数据的最小单元。页面大小可以是 4k 、 8k 、 16k 、 32k 。

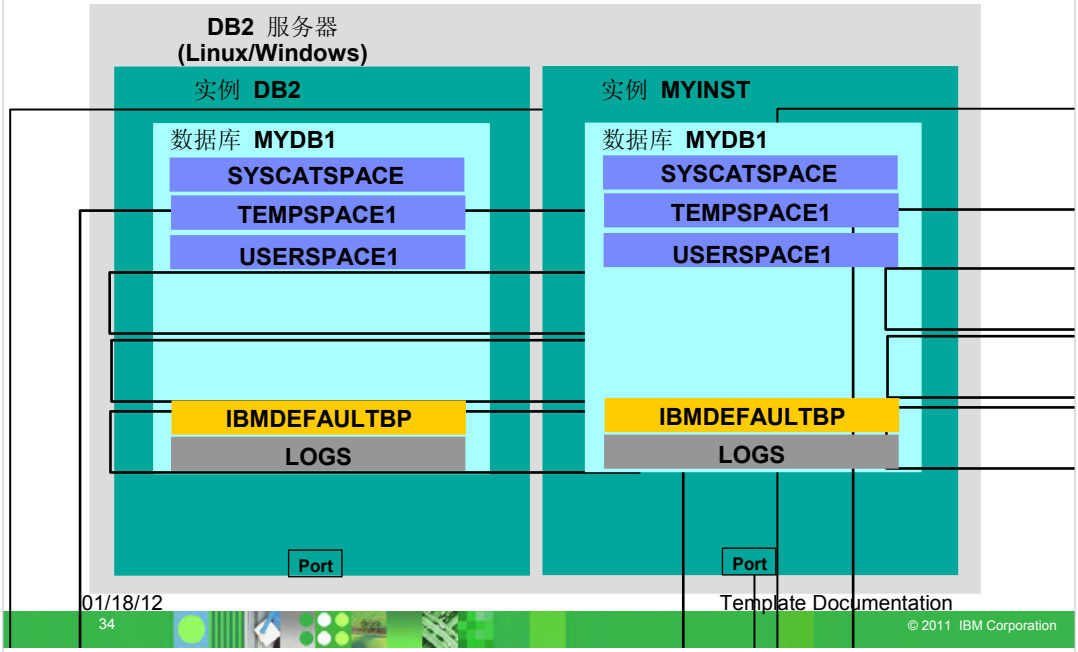
DB2 环境



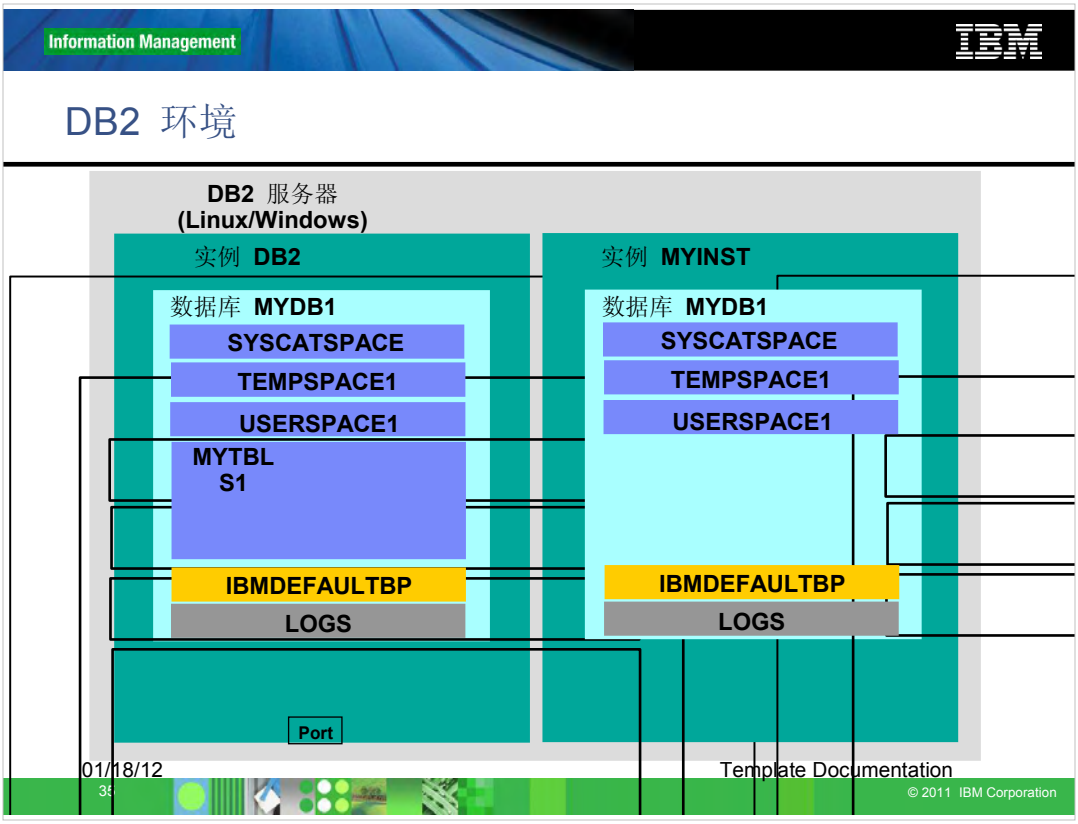
在创建数据库时还默认创建日志文件，它将用于数据库的恢复。

当你处理一个数据库时，不但把关于数据库的信息存储在磁盘上，而且在处理数据库时，日志文件中还存储对数据执行的所有操作。可以把日志看做进行“自动保存”操作的临时文件。日志将在备份和恢复部分进行详细讨论。

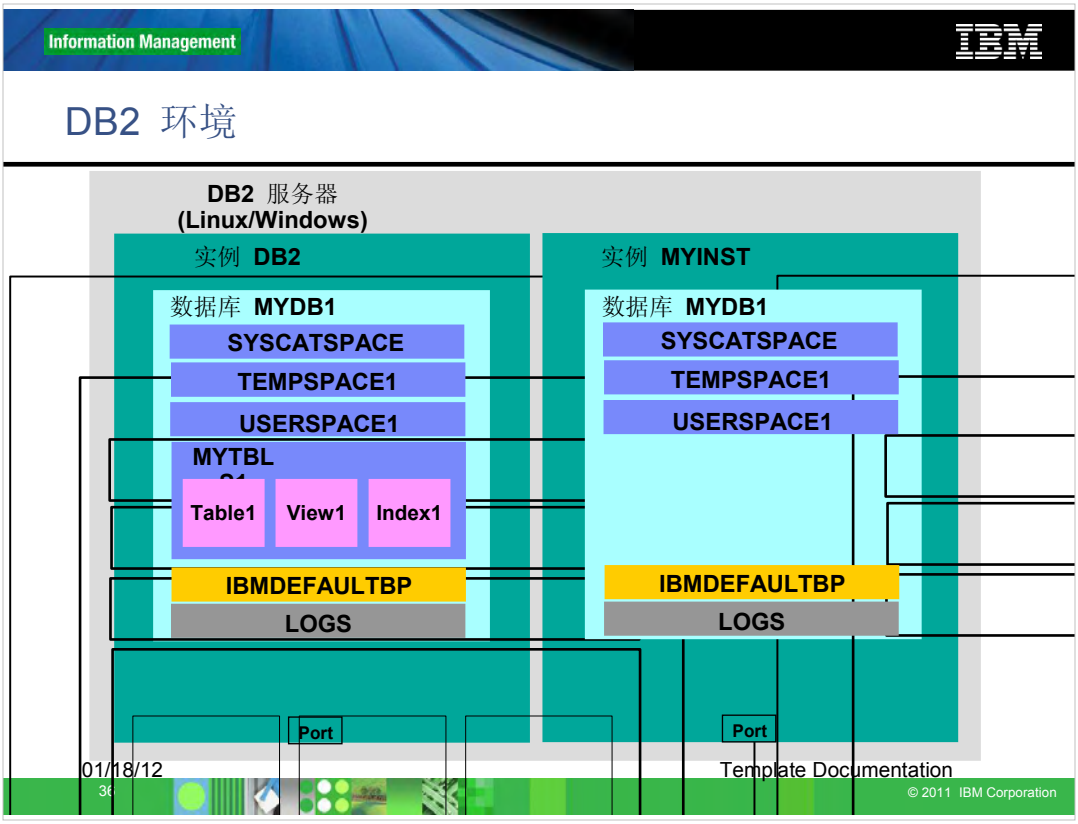
DB2 环境



请注意，在实例 MYINST 中的另一个数据库 MYDB1 中，也创建这些默认的数据库对象。

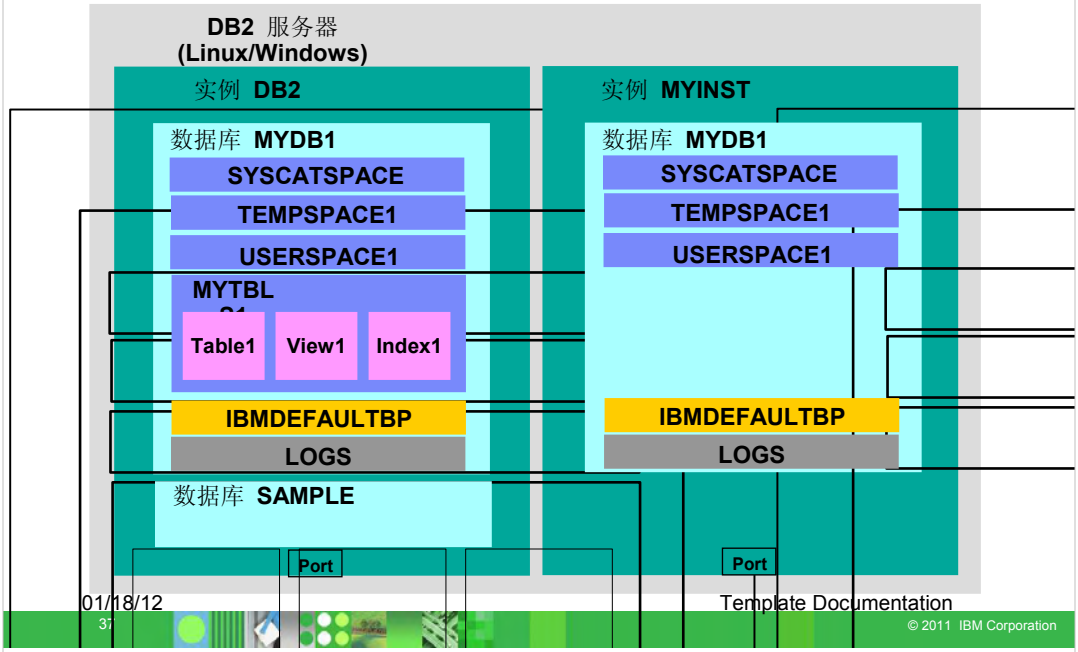


你也可以利用 CREATE TABLESPACE 语句创建你自己的表空间。该图示出了在实例 DB2 中的数据库 MYDB1 内创建的表空间 MYTBLS1 。当你创建表空间时，你同时规定待使用的磁盘以及待使用的内存（缓冲池）。因此，如果你有一份“热”表，就是说，一份使用非常频繁的表，你可以为其分配最快速的磁盘以及最多的内存，方法是为其指定一个具有这些特征的表空间。

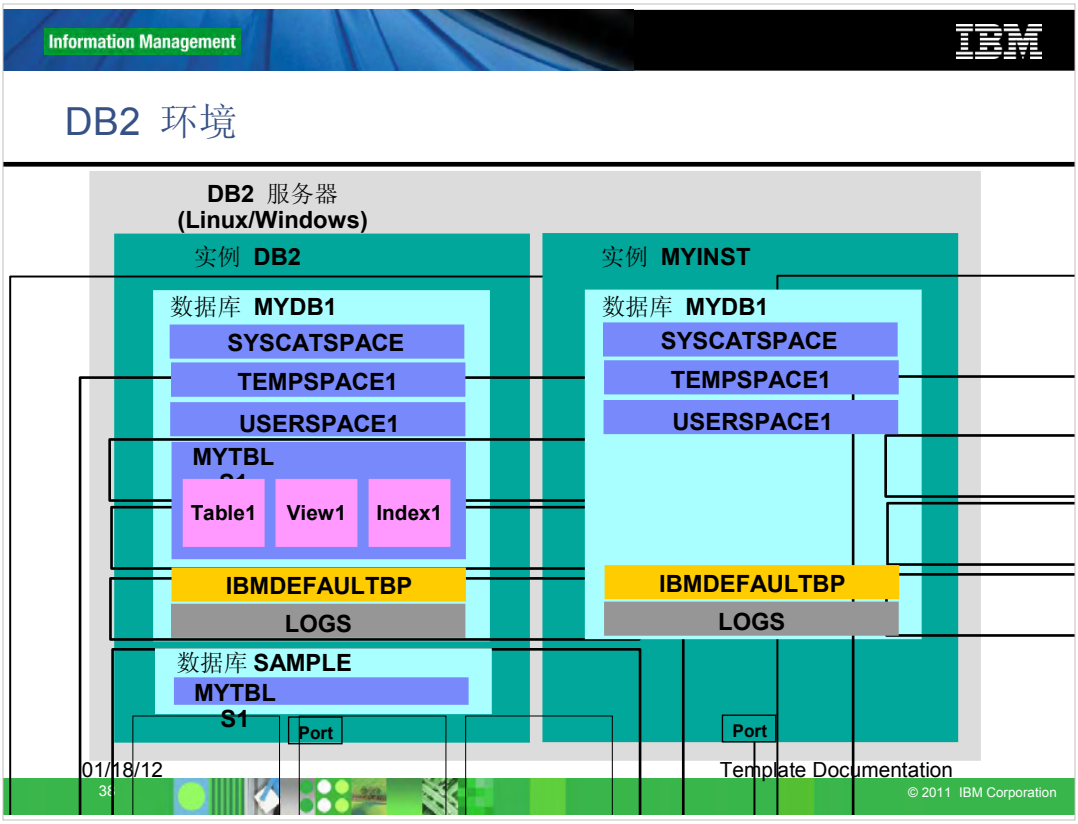


在该表空间中，你可以创建表、视图、索引等对象。

DB2 环境



我们前面说过，实例是独立的环境，因此，可以在不同的实例中创建同名的数据库。就像实例一样，数据库也是独立的单元；因此，一个数据库中的对象与另一个数据库中的对象没有关系。



因此，在图中，我们看到在实例 DB2 的数据库 MYDB1 和数据库 SAMPLE 中都有表空间 mytbls1 。这是合法的，因为数据库是独立的单元。请注意，由于图中空间所限，该图中没有显示出数据库 SAMPLE 的其他默认对象。

实例层次的命令概述

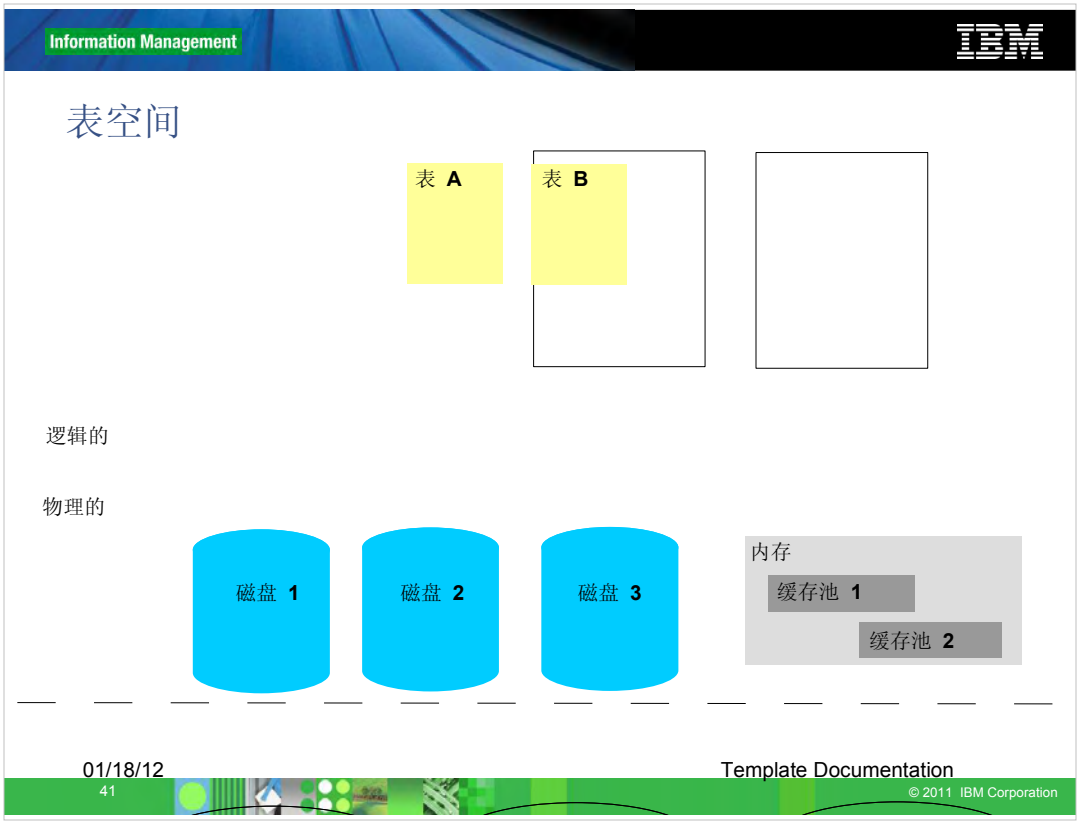
Command	Description
`db2start`	Starts the current instance
`db2stop`	Stops the current instance
`db2icrt`	Creates a new instance
`db2idrop`	Drops an instance
`db2ilist`	Lists the instances you have on your system
`db2 get instance`	Lists the current active instance

这归纳了你在实例层次上最常用的命令。

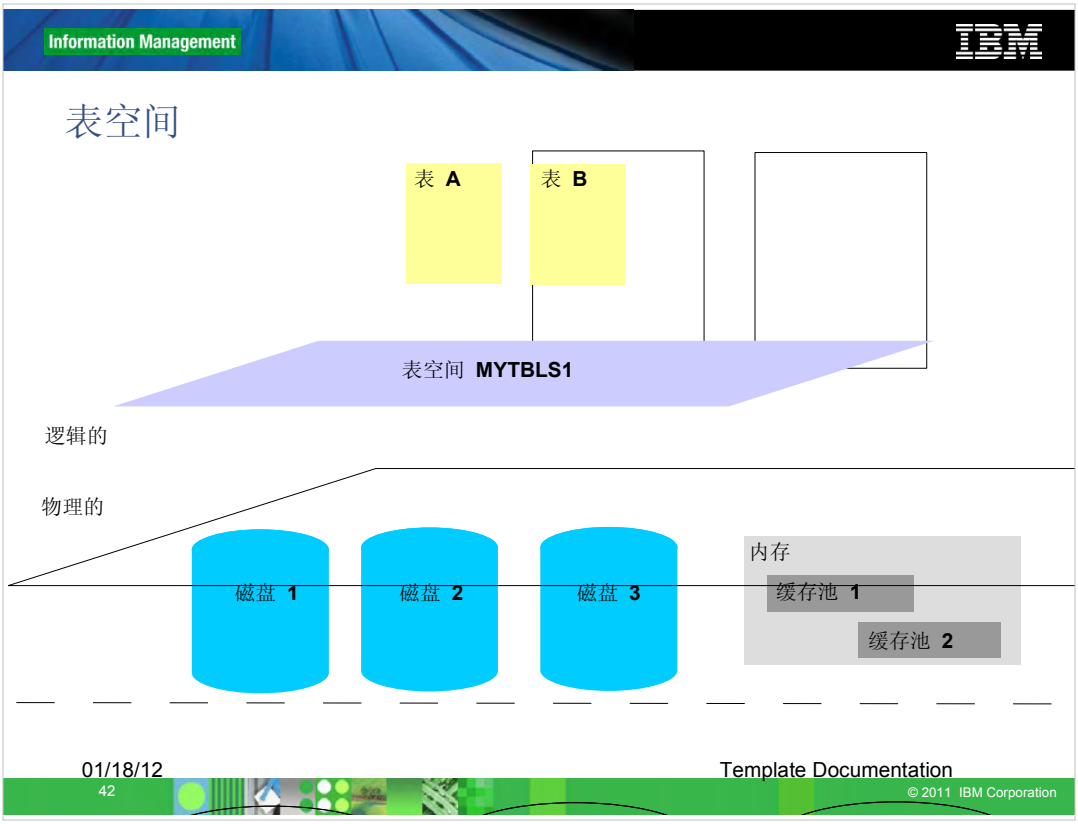
数据库层次的命令概述

Command/SQL statement	Description
`db2 create database`	Creates a new database
`db2 drop database`	Drops a database
`db2 connect to <database_name>`	Connects to a database
`db2 create table/create view/create index`	SQL statements to create table, views, and indexes respectively

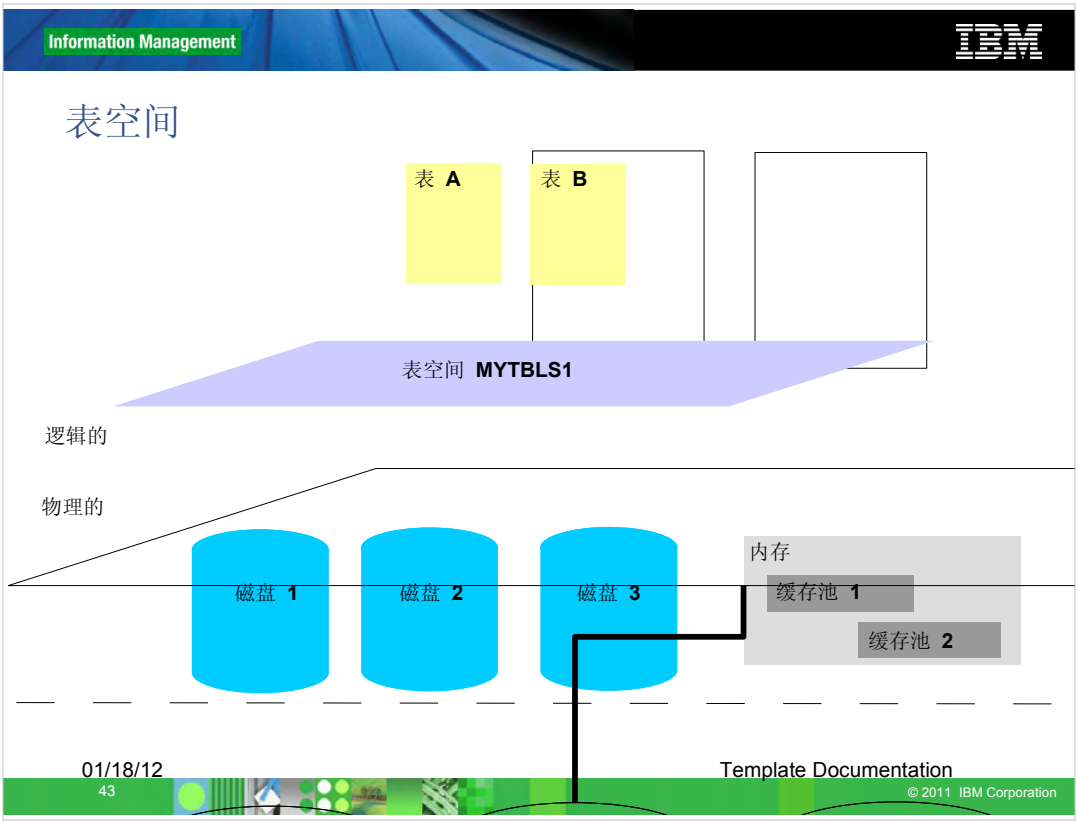
这归纳了你在数据库层次上最常用的命令。



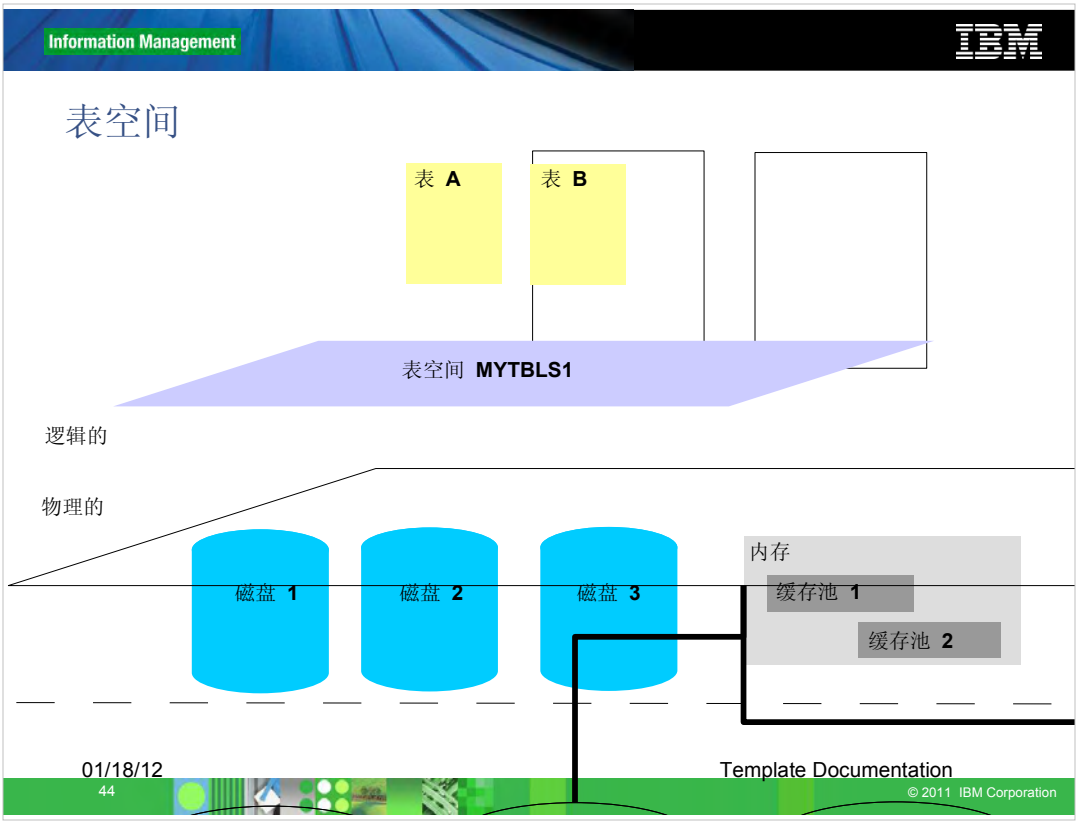
在该图中，在虚线之下有一些物理资源，例如磁盘和内存。在虚线之上，有一些逻辑对象，例如表。现在，我们假如说，“表 B” 是一份“热”表，就是说，它被众多用户频繁使用，你希望在访问该表时有最短的响应时间。怎么实现呢？



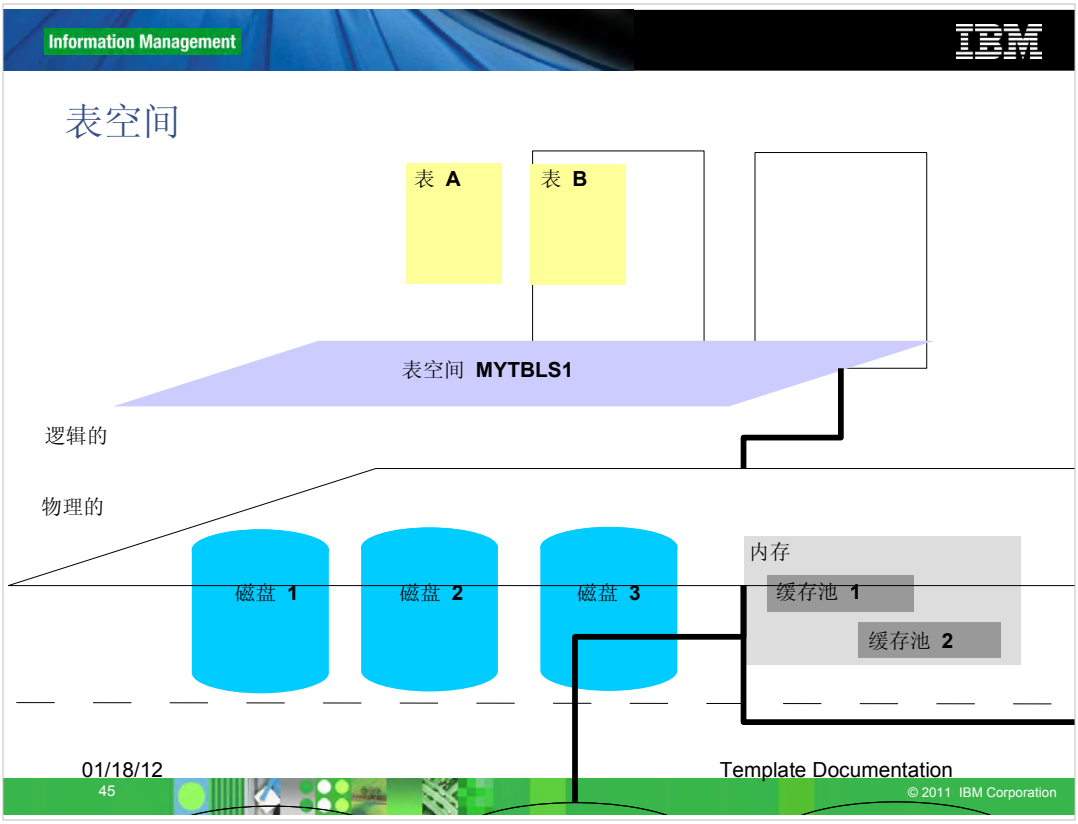
你可以把一个表空间（假设我们将其称为 “表空间 MYTBLS1”）用作你最快速的物理资源与该逻辑表之间的逻辑层。



假如 “磁盘 2” 是最快速的磁盘，而 “缓存池 1”是最大的缓存池（就是数据库缓存），那么，你可以把磁盘 2 与表空间 MYTBLS1 关联起来，



然后把缓存池 1 与表空间 MYTBLS1 关联起来。这实际上是在你创建表空间时完成的。



现在，你只需把表 B 与表空间 MYTBLS1 关联起来。

表空间管理

由系统管理 (SMS)

由操作系统进行管理

容器是目录

由数据库管理 (DMS)

由 DB2 管理。

容器是预先分配的文件或者裸设备

由自动存储器管理

旨在作为表空间的“单点存储器管理”

创建一个数据库并把一系列存储路径与它关联起来

未提供明确的容器定义

在存储路径上自动创建容器

扩大现有容器以及添加新容器完全由 DB2 管理

由系统管理

此类表空间被称为 System Managed Storage (SMS)（系统管理的存储器）。这意味着，操作系统来管理存储器。它们很易于管理，容器是文件系统目录。空间不是预先分配的，但文件是动态增长的。一旦你规定了容器，它们就在创建时固定下来了，以后也不能再添加其他容器了，除非使用了重定向的存储。当使用 SMS 表空间时，表数据、索引及 LOB 数据将无法在不同的表空间之间传播。

由数据库管理

此类表空间被称为 Database Managed Storage (DMS)（数据库管理的存储器）。这意味着， DB2 管理存储器。对空间的管理需要 DBA 进行更多的手工干预。容器可以是预先分配的文件或者裸设备。对于裸设备而言，数据将直接写入，不进行操作系统缓存。

可以利用 ALTER TABLESPACE 语句添加、删除及调整容器的大小。 DMS 表空间最利于提高性能，而且表数据、索引及 LOB 数据可以分拆到独立的表空间中，这有利于提高性能。

由自动存储器管理

由自动存储器管理的方式企图把上面两种方式的优点结合起来：

SMS 表空间的管理便利性，以及

DMS 表空间的良好性能 / 灵活性。

你首先需要利用自动存储器（在执行“ create database” 命令时它是默认开启的）创建数据库。然后你可以创建使用自动存储器的表空间。这既适用于 SMS ，也适用于 DMS 。在默认情况下，容器是跨越存储路径创建的。扩展数据块 (extent) 写入一个容器，然后再写入另一个容器，等等。由于容器是跨越存储路径创建的，这有利于提高性能，因为，假设存储路径驻留在不同的磁盘上的话，就不会出现仅有一个磁盘时的瓶颈。

缓存池基础

为表 / 索引数据提供实际的内存缓存

减少直接顺序 **I/O**

每部数据库至少需要一个缓冲池

以 **4K** 、 **8K** 、 **16K** 及 **32K pages** 页面为单位分配内存

根据页面大小，至少为表空间提供一个匹配的缓存池

STMM （自优化内存管理器）能够自动根据需要重新调整缓存池

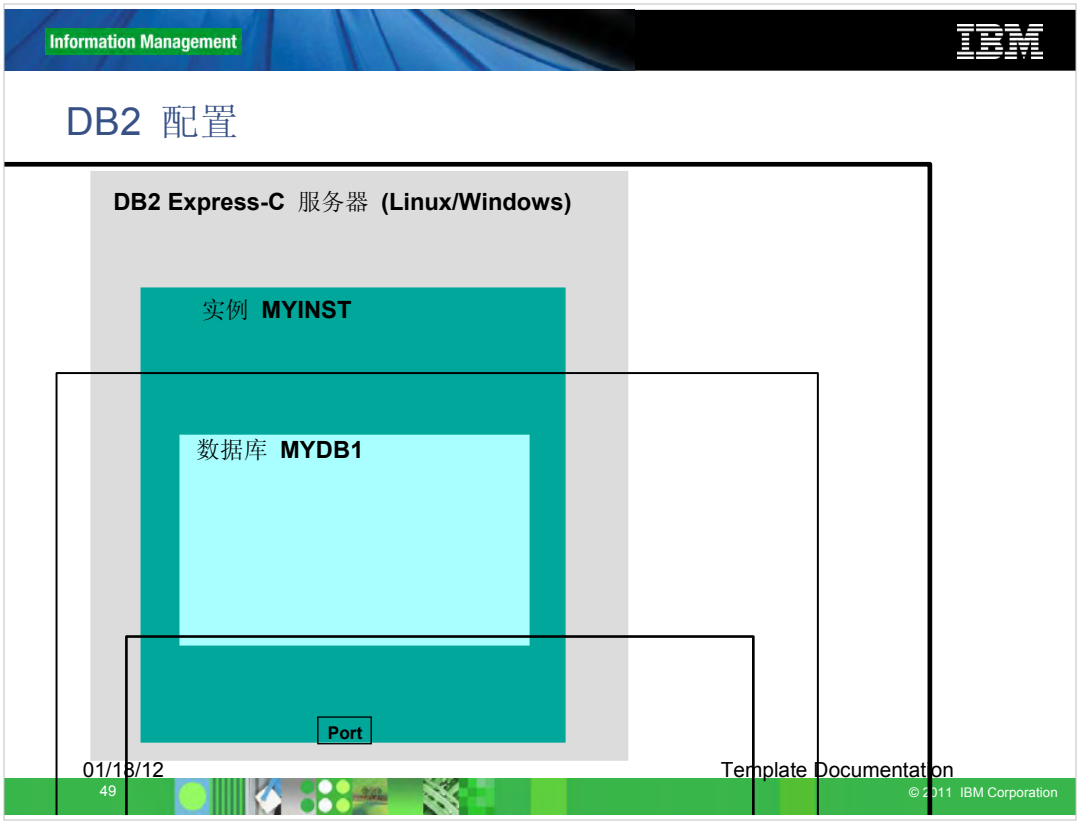
缓存池基本上是用于数据库的缓存（内存）。它主要为表 / 视图 / 索引缓存数据。缓存池总能够提高性能，因为它缩短了 DB2 访问磁盘（直接顺序 I/O ）所需要的时间。利用缓存池， DB2 还能够预取页面。这意味着，根据你打算进行的工作， DB2 可能能够提前猜测出你将需要从磁盘中获得哪些页面，而且能够把它们预取到内存中（即缓存池中）。这是异步进行的，就是说， DB2 无需按顺序进行此工作，它可以等待其他事情。

缓存池应当按照 4k 、 8k 、 16k 及 32k 的页面大小进行分配。页面是 DB2 开展工作的最小单元。页面越大，行就可以越大，但是，这也会耗费更多的内存和空间。例如：你可以这样来创建一种表： **create table mytable (col1 varchar(1024), col2 varchar(1024), col3 varchar(1024), col4 varchar(1024))…** 这个表就不适合 4K 页面，因此，你需要使用 8K 页面；但你也可以使用 16K 或 32K 的页面。

如前所述，有一个 8k 缓存池称为 IBMDEFAULTBP ，它与表空间 SYSCATSPACE 、 TEMPSPACE1 和 USERSPACE1 相关联。在 DB2 9 中，每个页面长度中还有一些隐藏的缓存池。对于大对象 (LOB) ， DB2 通常不使用缓存。 LOB 太大，如果把它们装入内存的话，它们就会覆盖所需要的“正常”页面，或者导致此类页面的交换。

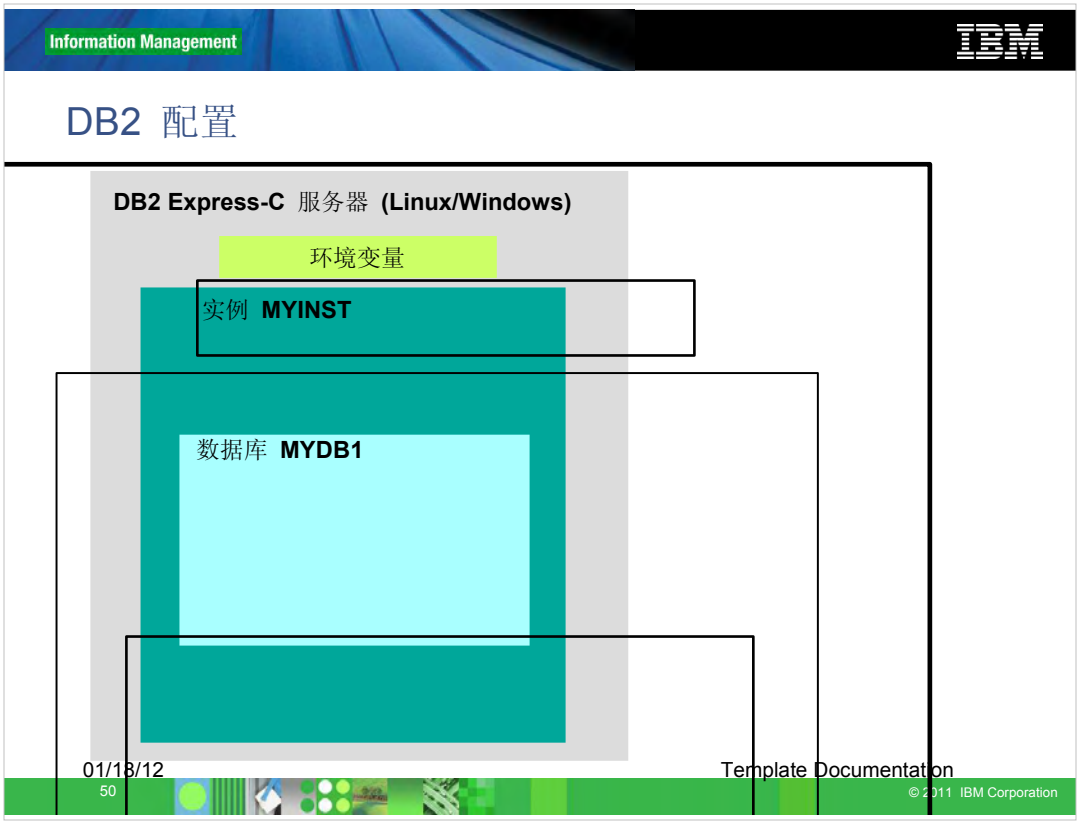
议题

- DB2 服务器版本、客户端和驱动程序
- DB2 Express-C 概述
- DB2 命令行处理器（演示）
- DB2 环境
- **配置 DB2**
- 使用脚本
- 接入 DB2 服务器
- 数据移动实用工具



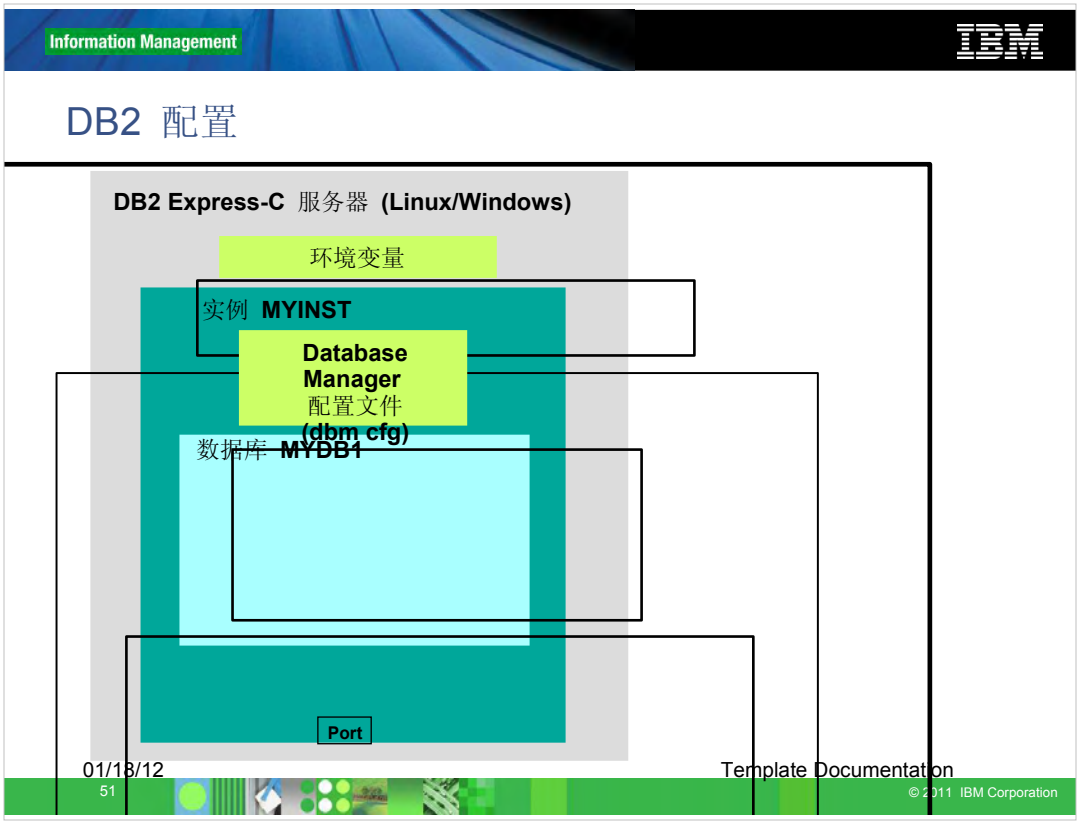
DB2 服务器可以在四种不同的层次上进行配置：

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



第一个层次是操作系统层次，通过“环境变量”进行。一个关键的环境变量是DB2INSTANCE。该变量指出了你当前正在使用的实例以及你的DB2命令将适用的实例。例如，为了在 Windows中把活动实例设置为myinst，你可以运行这个操作系统命令：

```
set db2instance=myinst
```



第二个层次是使用数据库管理器配置文件 (dbm cfg)。dbm cfg文件中含有许多参数，它们能够影响实例以及实例中的所有数据库。为了查看 dbm cfg 配置参数，使用以下命令：

db2 get dbm cfg

如果要修改某个参数，例如，“INTRA_PARALLEL”参数，使用以下命令：

db2 update dbm cfg using intra_parallel yes

许多参数都是动态的，就是说，修改能够立即生效；但是，某些参数的修改可能需要停止和启动实例。在命令行上，这可以使用 DB2stop 和 db2start 命令来完成。

在终止一个实例之前，必须断开所有的应用。如果你打算强行终止一个实例，你可以使用 DB2stop force 命令。

如果要查看“ db2 get dbm cfg”命令的输出的详细信息，或者要查看某个参数值是否生效或者被推迟，你可以在命令的末尾加上“ show detail”，例如“ db2 get dbm cfg show detail”。要让这个子句发挥作用，你首先需要使用下面的命令连接到实例中：

db2 attach to <instance_name>

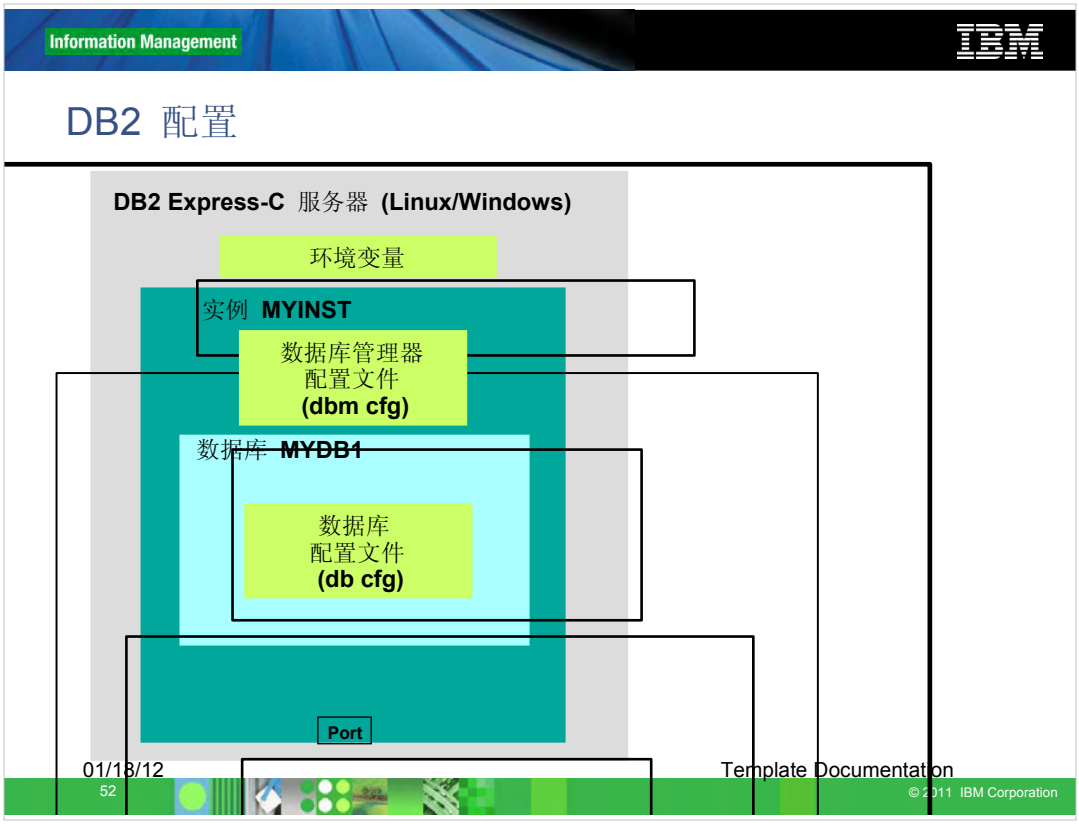
例如，为了连接实例“ DB2”，你应当：

db2 attach to db2

然后再执行：

db2 get dbm cfg show detail

你很少需要显式连接一个实例来使用它。一般情况下，连接都是隐式完成的。



第三个层次是使用数据库配置文件 (db cfg)。这将在数据库层次上配置参数。为了查看 db cfg 配置参数，使用以下命令：

db2 get db cfg for SAMPLE

要修改参数，可以使用：

db2 update db cfg for SAMPLE using LOGFILSIZ 1234

如果你已经接入数据库中，你可以省略上述例句中的 “for SAMPLE” 子句，例如：

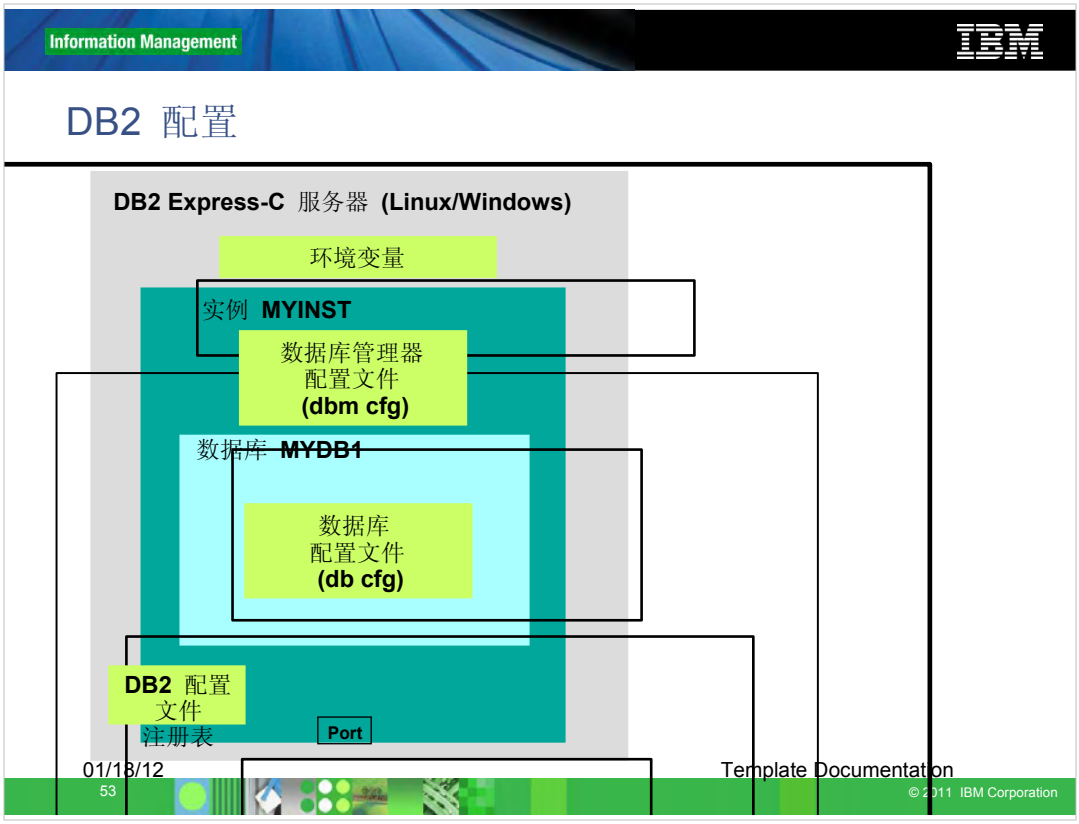
db2 connect to sample

db2 get db cfg

db2 update db cfg using LOGFILSIZ 1234

有些 DB CFG 参数是动态的，因此变化能够立即生效。而有些参数则需要在终止所有连接之后才能修改。在进行第一次连接之后，参数的新值将会生效。

如果要查看 “ db2 get db cfg” 命令的输出的详细信息，或者要查看某个参数值是否生效或者被推迟，你可以在命令的末尾加上 “ show detail”，例如 “ db2 get dbm cf show detail” 。要让这个子句发挥作用，你首先需要接入数据库。



最后，配置DB2的最后一个层次是使用 DB2 配置文件注册表，它可以在操作系统层次上或者在实例层次上进行设置。要注意，这个DB2注册表与Windows registry注册表没有任何关系。DB2 注册表变量可以用来做许多工作，例如，针对某种操作系统为DB2设定一些特殊的行为，以便充分利用某项操作系统功能。

要修改 DB2 配置文件注册表变量，可以使用 “db2set” 命令。进行修改之后，你需要执行 db2stop/db2start 命令，以确保变更能够生效。

这 4 个层次（操作系统、dbm、db、DB2 配置文件注册表）上的参数是完全不同的。换言之，并不是说，你可以在4个不同层次的任何层次上设置同一个参数；相反，在4个配置层次的每一个层次上，参数都是完全不同的。

DB2 配置命令概述

操纵 **dbm cfg** 的命令（实例层次）

Command	Description
`db2 get dbm cfg`	Retrieves information about the dbm cfg
`db2 update dbm cfg using <parameter_name> <value>`	Updates the value of a dbm cfg parameter

操纵 **db cfg** 的命令（数据库层次）

Command	Description
`get db cfg for <database_name>`	Retrieves information about the db cfg for a given database
`update db cfg for <database_name> using <parameter_name> <value>`	Updates the value of a db cfg parameter

如前所述，对于 db cfg，如果你已经接入了数据库，你在命令语法中无需在包含 “ for <database_name>”。

DB2 配置命令概述

操纵 **DB2** 配置文件注册表的命令

Command	Description
`db2set -all`	Lists all the DB2 profile registry variables that are currently set
`db2set -lr`	Lists all the DB2 profile registry variables
`db2set <parameter>=<value>`	Assigns a parameter with a given value

注：
在 db2set 命令中，在等号与参数名之间以及等号与值之间不能存在空格。例如：

```
db2set db2comm = tcpip    （错误）
db2set db2comm=tcpip      （正确！）
```


要想查看已经设置的 DB2 配置文件注册表变量，可以使用下面的命令：

db2set -all

要想查看所有能够设置的 DB2 配置文件注册表变量，可以使用下面的命令：

db2set -lr

要想设置某个变量，例如把 DB2COMM变量设置为值 “tcpip”，可以使用下面的命令：

db2set db2comm=tcpip

请注意，等号前后不能有空格。

议题

- DB2 服务器版本、客户端和驱动程序
- DB2 Express-C 概述
- DB2 命令行处理器（演示）
- DB2 环境
- 配置 DB2
- **使用脚本**
- 接入 DB2 服务器
- 数据移动实用工具

SQL 脚本

优点

- 易于理解
- 平台独立性

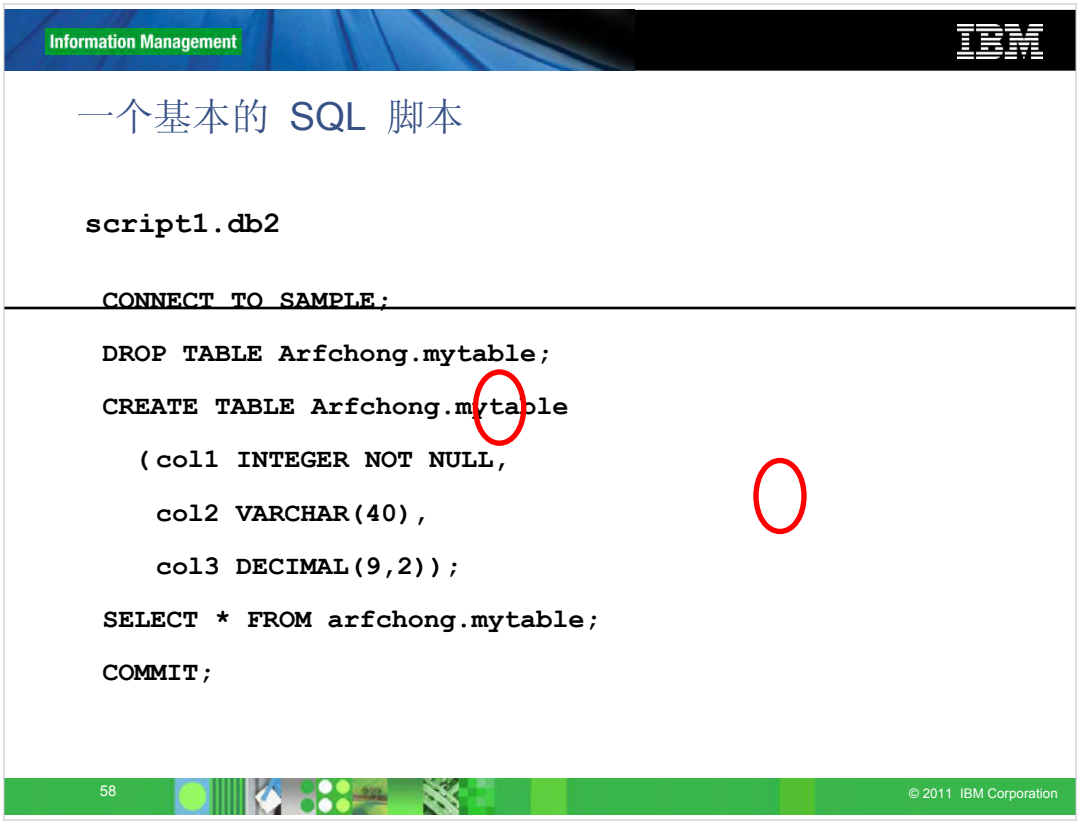
缺点

- 不支持脚本参数

SQL 脚本将很快在后面讲解，它们非常易于理解。

它们是独立于平台的，就是说，它们无需修改就能够在不同的平台（例如 Windows 和 Linux ）之间进行移植。

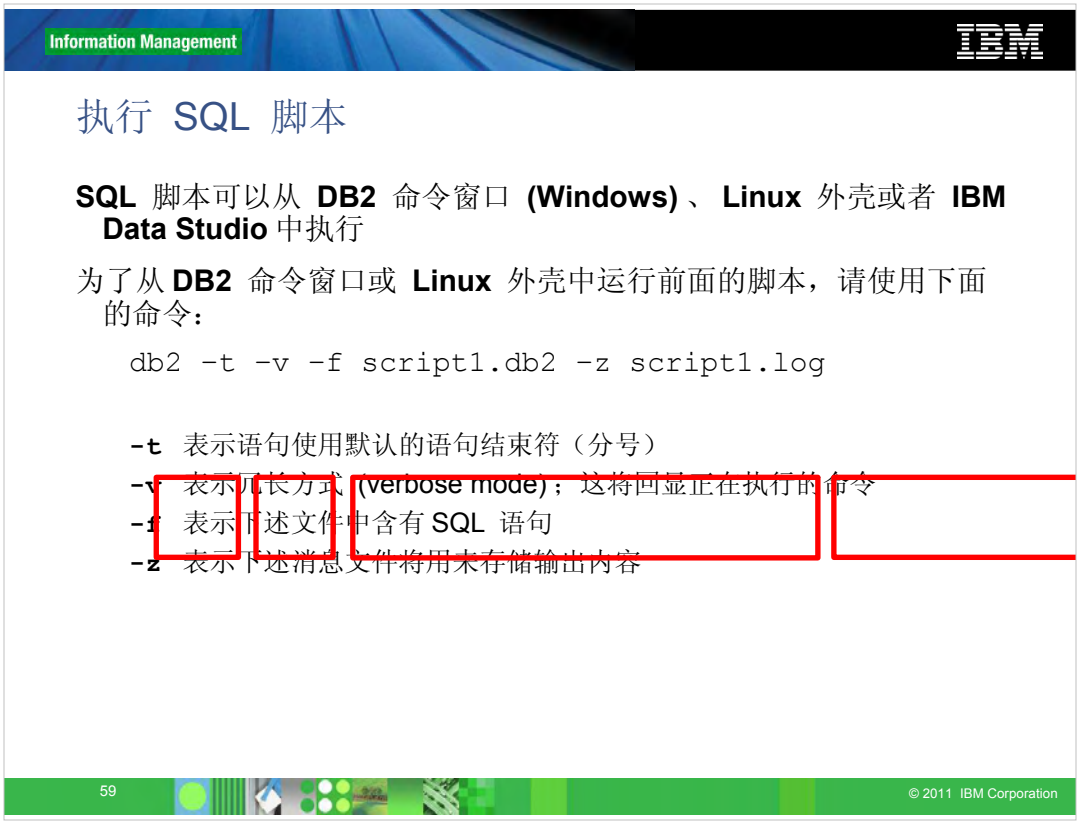
但是， SQL 脚本不许使用参数。因此，要想真正发挥 SQL 脚本的全部优势，你应当把它们与操作系统脚本结合起来。



我们来看看一段非常基本的 SQL 脚本。假设你已经创建了文件 script1.db2 。其名称和扩展名无关紧要。你可以使用任何你喜欢的扩展名。我们来看看此文件的内容。

如您所见，它由多个 SQL 语句构成。请注意，每个语句都以分号结束。这是 DB2 默认的语句结束符。它让 DB2 知道一个给定的语句在哪里结束。在本例中，我们接入 SAMPLE 数据库，然后我们删除表 Arfchong.mytable ，随后再创建它。我们从表中查询所有的记录，最后我们发出 COMMIT 语句。

请注意，使用 DROP TABLE 语句是为了“把事情打扫干净”。在我们首次运行该脚本时， DROP TABLE 语句只会给我们提示一条错误消息，指出表 Arfchong.mytable 不存在。以后再运行该脚本时就正常了，它先删除表 Arfchong.mytable ，然后再创建它。包含 DROP 语句是 DBA 和开发人员的常见做法。把它放到脚本的起始很常见。为了执行该脚本，你可以在 Windows 中或 Linux 外壳中使用使用 IBM Data Studio 或者 DB2 命令窗口。



为了从命令窗口或 Linux 外壳中执行它，可以发出以下的命令：

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

我们首先键入“ db2”来调用 DB2 命令行处理器，它将使用我们所给出的那些选项。 -t 表示将使用语句结束符，而且是默认的语句结束符（分号）。

-v 代表“ verbose”（冗长）。它意味着我们希望在提供输出之前首先回显正在执行的语句。

-f 后跟一个文件名（在本例中为“ script1.db2”），这意味着待执行的语句来自于文件 script1.db2 。

-z 后跟一个文件名（在本例中为“ script1.log”），这意味着所有的输出都将记录到该文件中。在执行 DB2 脚本之前先删除这些消息文件是个不错的主意，这样可以避免先前的脚本执行所产生的输出不会与当前的脚本执行所产生的输出相混淆。

使用不同的语句结束符

```
CREATE TABLE Arfchong.mytable
  (col1 INTEGER NOT NULL,
   col2 VARCHAR(40),
   col3 DECIMAL(9,2)
  );
```


这里说明了你或许需要改变语句结束符的情况。

例如，在本例中的表与我们在前面的脚本中创建的是一样的。

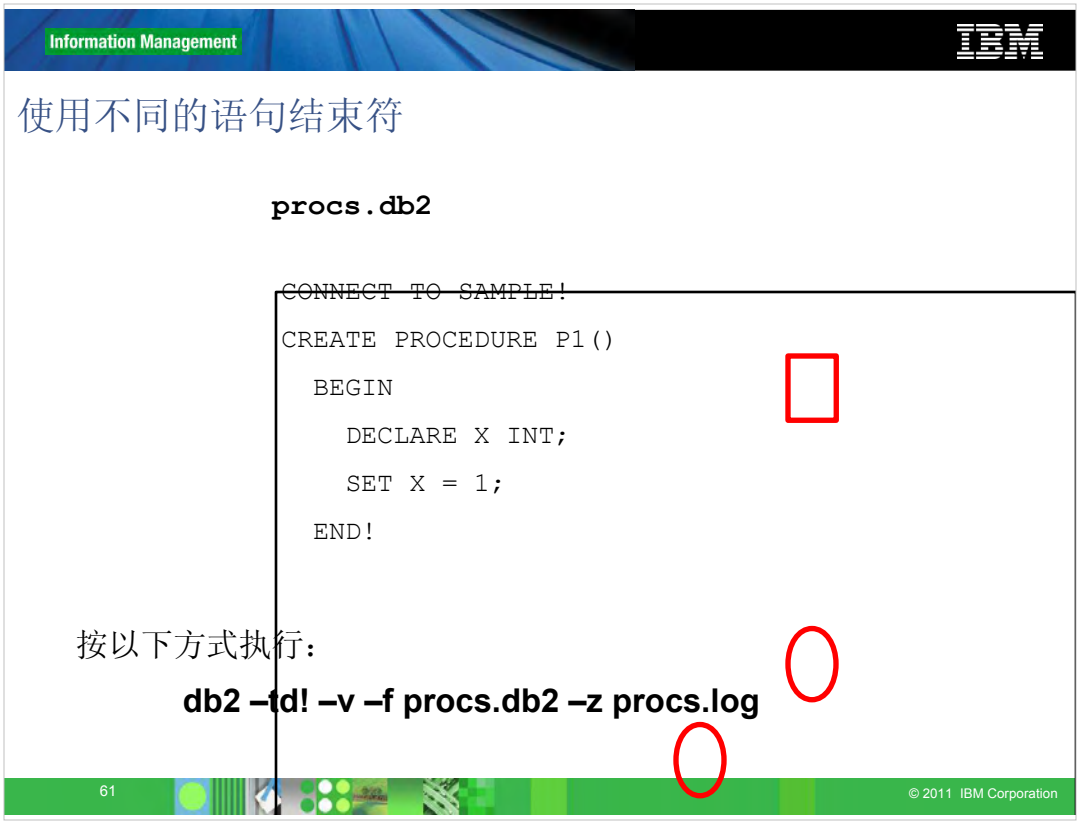
小红方框里是语句结束符，即分号。

小椭圆里是逗号，它是“ create table”语法的一部分。

如果我规定（我将在后面讲解）把逗号用作语句结束符，

那么， DB2 就会感到混乱，它会认为在它第一次遇到逗号时，语句就结束了，这里就是 null 之后。

因此，在本例中使用逗号作为语句结束符是个坏主意。



这个幻灯片是另外一个例子，这一次使用了 create procedure 语句。根据语法，每个语句或者每一行结束时都使用分号，但这会给我带来一个问题，如果我使用默认的语句结束符的话（在本例中也是分号）。

因此，在本例中，我必须改变结束符。我决定使用感叹号。

现在执行本脚本，我必须告诉 DB2，我正在使用一个不同的结束符。

实现方法是采用 -t 标志，后面跟 -d 标志（定界符），然后再跟我正在使用的结束符，如图所示。因此我需要键入 db2 -td! … 其他部分与先前的例子都一样了。

操作系统脚本

优点：

- 更大的灵活性
- 额外的逻辑可能性
- 支持参数 / 变量

缺点：

- 平台依赖性

如其名字所表示的，这些脚本来自操作系统，因此他们与 DB2 无关，我不打算详细介绍如何创建操作系统脚本了。

我想向你们介绍如何把 SQL 脚本与操作系统脚本结合起来。

操作系统脚本非常灵活，可以让你使用参数。

操作系统脚本是依赖于平台的。这意味着， Windows 和 Linux 有不同的脚本语法。

例如，我们来向你介绍这些称为 'create database.bat' 的 Windows 操作系统脚本。由于是 Windows，你需要使用 .bat 扩展名把它标示为操作系统脚本。对于 Linux 或 Unix，你只需要 chmod + x 以便执行本文件；因此，基本而言，修改期限以便使其可以执行。你还需要修改识别参数的方法，以便它们使用 “ $”，而不是 “ %”。

在本文件中，我们使用我们在重点介绍的几个参数。

因此，我们有 DBPATH，我们后面在本语句中使用它。我们还有另一个参数，名为 DBNAME，它被指定给 MYDB。

如您所见，这里任何带有 db2 的句子都调用 DB2 命令或者 DB2 脚本。因此，'db2 CREATE DATABASE' 这一句调用一条 DB2 命令并传递其参数。因此，在这个操作系统脚本中，我们也调用 SQL 基本。

我们还有这些称为 memory 的其他参数，我们在 CONNECT 语句中再次使用它。我们把用户 ID 作为参数 %1 来传递

因此，当你执行本脚本时，你将通过 'create database.bat' 来调用它，然后你给出你的用户 ID，它将代替这个 '%1'；你还使用了 %2，它将匹配你对 password （密码）所输入的任何东西。

议题

- DB2 服务器版本、客户端和驱动程序
- DB2 Express-C 概述
- DB2 命令行处理器（演示）
- DB2 环境
- 配置 DB2
- 使用脚本
- **接入 DB2 服务器**
- 数据移动实用工具

目录

系统数据库目录

列出数据库目录

本地数据库目录

列出 < 磁盘 / 路径 > 上的数据库目录

节点目录

列出节点目录

DCS 目录

列出 dcs 目录

DB2 目录是二进制文件，我们在其中存储有关我们可以连接的数据库的信息。有四种目录：

1) 系统数据库目录

可以把它想象为一本书的目录。它将列出你可以连接的所有数据库，无论是本地的还是远程的。

2) 本地数据库目录

列出在本地的数据库，也就是在你发出“ list db”命令的服务器上的数据库。

3) 节点目录

这是你存储连接信息的目录，例如，你使用 TCP/IP 协议时的端口号码和 IP 地址。

4) DCS 目录

这不在本课程范围内，但如果安装了“ DB2 Connect”软件，它就会显示出来，从而能够让你接入大型机 (DB2 for z/OS) 或中等系统（即 AS/400 ） (DB2 for i5/OS) 上的服务器。

接入 DB2 服务器的要求

I) 本地连接

在发出 CREATE DATABASE 命令时，系统数据库目录自动填充连接信息

II) 远程连接

服务器上需要的设置：

1) 打开 TCPIP 监听器
2) 指定 DB2 实例端口

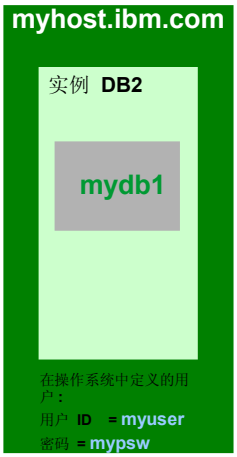
在客户端需要的设置：

1) 在客户端的节点目录中创建条目
2) 在客户端的系统数据库目录中创建条目

如果连接是本地的，也就是 DB2 客户端和 DB2 服务器在同一个系统上，那么，通常情况下，无需进行设置，连接所需要的信息已经在执行 Create Database 命令时添加到了系统本地的 DB2 目录中。

如果连接是远程的，你就需要在服务器上和客户端上进行一些设置。在服务器上，你需要打开 TCP/IP 监听器，并指定它们所监听的 DB2 实例端口。在客户端上，你需要向节点及系统目录中添加条目。

远程连接 - 服务器上需要的设置

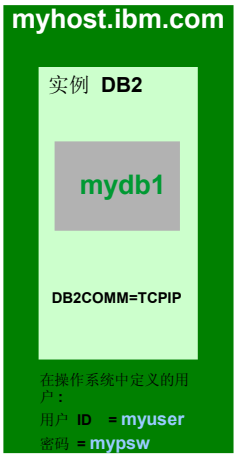


服务器上需要设置两样东西：

远程连接 - 服务器上需要的设置

1) 打开 **TCP/IP** 监听器

db2set DB2COMM=TCPIP



1) DB2COMM

这个 DB2 注册表变量决定了哪些通信协议监听器应当监控来自客户端的请求。一般情况下， TCP/IP 是最经常使用的通信协议。修改此参数需要重启实例。

db2set DB2COMM=TCPIP

db2stop

db2start

远程连接 - 服务器上需要的设置

1) 打开 **TCP/IP** 监听器

db2set DB2COMM=TCPIP

2) 指定 **DB2** 实例端口

利用 svcename 50000 更新 dbm cfg

或者

利用 svcename db2c_DB2 更新 dbm cfg

如果你使用服务名称，一定要更新客户端上的这些文件：

Linux: /etc/services

Windows: c:\winnt\system32\drivers\etc\services

服务文件中的示例条目：

db2c_DB2 50000/tcp

2) SVCENAME

该数据库管理器配置参数应当设置为服务名（在 TCP/IP 服务文件中定义）或者你在访问本实例的数据库时所使用的端口号。使用以下命令：

update dbm cfg using svcename <port # or service name>

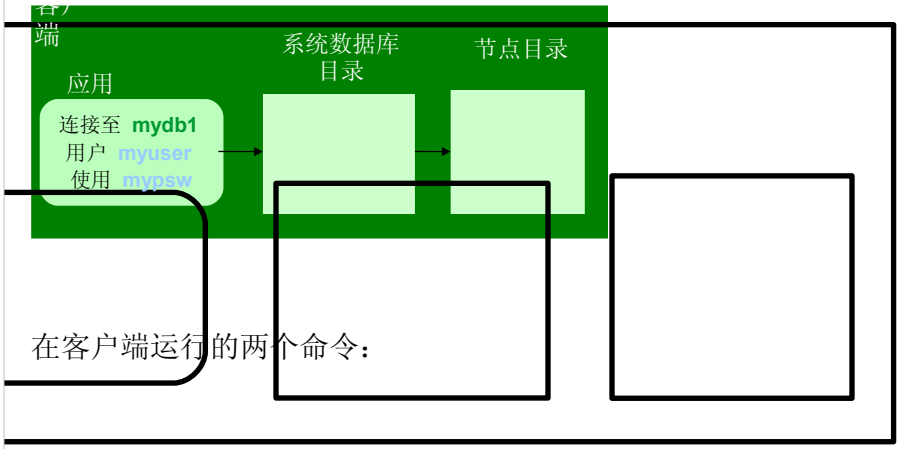
如果你使用服务名，一定要更新这些文件：

Linux: /etc/services

Windows: c:\winnt\system32\drivers\etc\services

示例 : db2cdb2inst1 50000/tcp

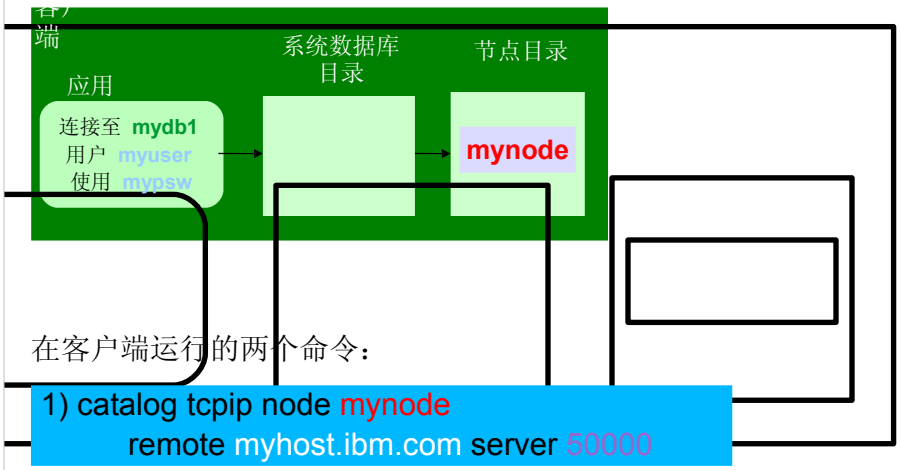
远程连接 - 客户端需要的设置



在客户端，你需要事先知道这些信息：

- 你希望接入的数据库的信息
- DB2 实例在数据库所在的服务器上的端口号码。你也可以使用服务名称，只要在 TCP/IP 服务文件中有匹配的项目即可。
- 操作系统使用用户 ID 和密码接入数据库。该用户 ID 必须事先在服务器上定义过。

远程连接 - 客户端需要的设置

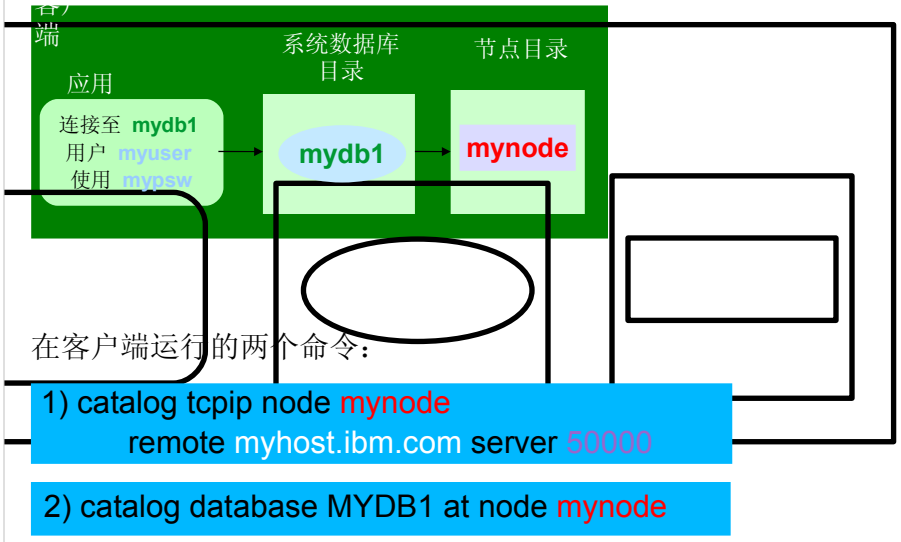


在客户端运行的两个命令：

1) catalog tcpip node mynode
remote myhost.ibm.com server 50000

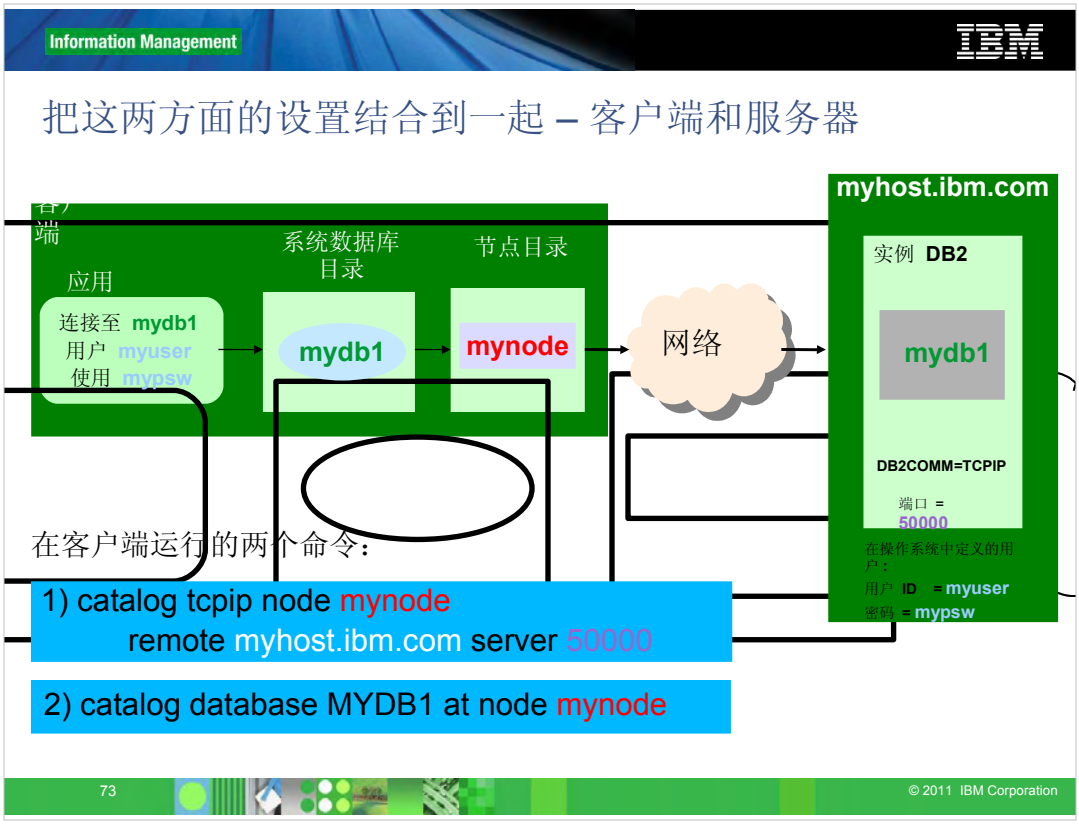
第一条命令 (catalog tcpip) 将把一个条目置入节点命令中

远程连接 - 客户端需要的设置



第二条命令 (catalog db) 将把一个条目置入系统数据库目录中，而且将根据节点名称与节点目录联系起来。

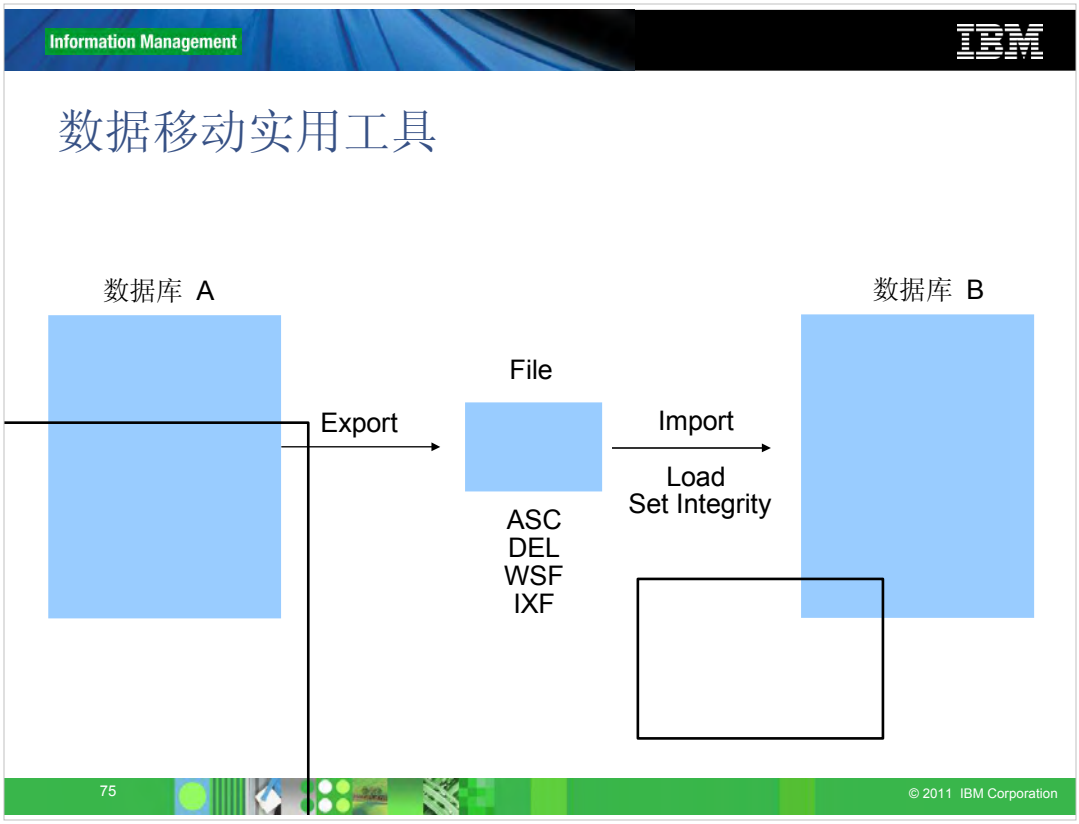
完成设置之后，从一个应用中进行连接，在你打算接入的服务器上提供用户 ID 和密码。



这张图示出了把客户端及服务器上需要进行的设置组合在一起之后的场景。

议题

- DB2 服务器版本、客户端和驱动程序
- DB2 Express-C 概述
- DB2 命令行处理器（演示）
- DB2 环境
- 配置 DB2
- 使用脚本
- 接入 DB2 服务器
- **数据移动实用工具**



这张图概述了一些 DB2 数据移动实用工具。左侧是数据库 A，右侧是数据库 B 。利用 EXPORT （导出）工具可以把数据库 A 中某个表内的各行导出到一个文件中。该文件可以是以下任何一种格式：

ASC 代表 ASCII 码， DEL 代表带定界符的 ASCII 码（基本上是 ASCII 码格式，但使用了一个定界符）； WSF 代表工作表 (Work-Sheet) 格式（主要用于 Excel 或 Lotus-1-2-3 软件）； IXF 代表 IBM eXchange 格式。

该格式是 IBM 特有的，它很有用，因为它包含 DDL 或者表的结构作为文件的一部分。其他格式并不包含 DDL，但它们更为标准化。

现在，一旦你利用上面的任何一种格式存储了表中的行，你就可以使用 IMPORT （导入）或者 LOAD （加载）工具把这些行加载到同一个数据库或另一个数据库的表中，在本例中，即数据库 B 中。如果你在使用 IXF 格式，则无需事先创建表。因为采用 IXF 格式时，表将创建出来，因为 DDL 被导出到输出文件中。但是，如果你在使用另一种格式，例如 ASCII 、有定界符的 ASCII 或者 WSF ，那么，你就需要事先在目标位置手工创建出该表。我们已经讨论了 IMPORT ，但我们还有了一套工具，即 LOAD 。 LOAD 与 IMPORT 一样，但速度更快。 IMPORT 在后台使用 SQL INSERT 语句，因此，这些语句由 DB2 引擎进行处理，它将激活触发器、检查约束条件以及把变更记录到日志文件中。完成这些工作之后，它将把数据插入到磁盘上的页面中。

但 LOAD 不经过 DB2 引擎，而是直接把数据存储到数据页面中。这速度会快许多，但它意味着触发器未被激活，也未检查约束条件。因此，加载的数据可能会存在完整性问题。当你已经知道数据是完整的，因而无需让数据库检查这些条件时，则使用 LOAD 是非常理想的。你也可以在执行完 LOAD 之后运行 SET INTEGRITY 命令，以便检查数据的完整性。

数据移动实用工具

EXPORT, IMPORT, LOAD

db2move

方便地导出 / 导入 / 加载 / 复制表及数据 (IXF)

db2look

抽取：

DDL

权限

数据库统计

表空间特征

Export 、 import 和 LOAD 都是表级工具，因此，它们针对表进行操作。
如果你打算对许多表运行这些工具，你可以创建一个脚本，或者你可以使用一个称为 db2move 的工具。通过 db2move ，你可以规定一个给定的模式，这样你就能够根据该模式移动或者导出给定数量的表，然后再把它们导入到另一个数据库中。你可以把 db2move 与 export 、 import 和 load 一起使用，但它只能使用 IXF 格式。

我们还有另一套工具，名为 db2look 。 db2look 用于抽取诸如 DDL 、期限、数据库统计等信息。它基本上能够让你对数据库结构进行克隆。

如果你对本课程所讨论的内容有什么技术方面的问题的话，可以利用 db2university.com 课程 AA001EN 中的论坛。 同学、教员以及 IBM 人员可以帮助你！

Information Management

IBM

Data Studio 入门

IBM 信息管理云计算能力中心
IBM （加拿大）研究院

议题

DB2 工具纵览

Data Studio 概述

工作台浏览（演示）

管理连接（演示）

使用 **SQL** 和 **SQL** 脚本（演示）

支持性阅读材料和视频

阅读材料

IBM Data Studio for DB2 入门电子书

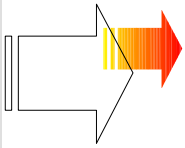
第 1 章：概述和安装
第 2 章： 管理你的数据库环境
第 3 章，第 3.2.1 节： 创建表空间

视频

db2university.com 课程 AA001EN

第 3 课 : Data Studio 入门

议题



DB2 工具纵览

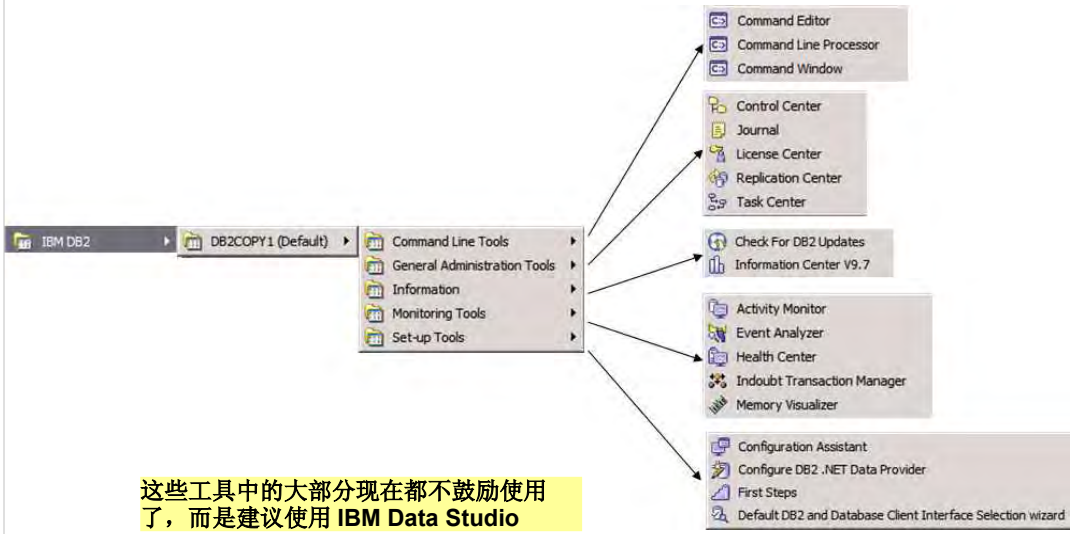
Data Studio 概述

工作台浏览（演示）

管理连接（演示）

使用 SQL 和 SQL 脚本（演示）

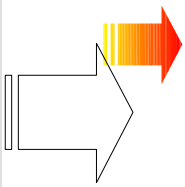
DB2 工具



这些工具中的大部分现在都不鼓励使用了，而是建议使用 IBM Data Studio

本图示出了随同 DB2 一起提供的工具。在 Linux 和 Windows 中提供相同的工具（ Command Window[命令窗口除外] ）。如图所示，这些工具大部分今后都不会再增强了，最终会消逝，转而采用 IBM Data Studio 。 Command Line Processor （命令行处理器）和 First Steps （初始步骤）工具很可能会保留下来。

议题



DB2 工具纵览

Data Studio 概述

工作台浏览（演示）

管理连接（演示）

使用 SQL 和 SQL 脚本（演示）

什么是 IBM Data Studio?

用于数据库管理及数据库开发的免费工具

以 **Eclipse** 为基础

可以用于：

面向 Linux 、 UNIX 或 Windows 的 DB2

面向 iSeries 的 DB2

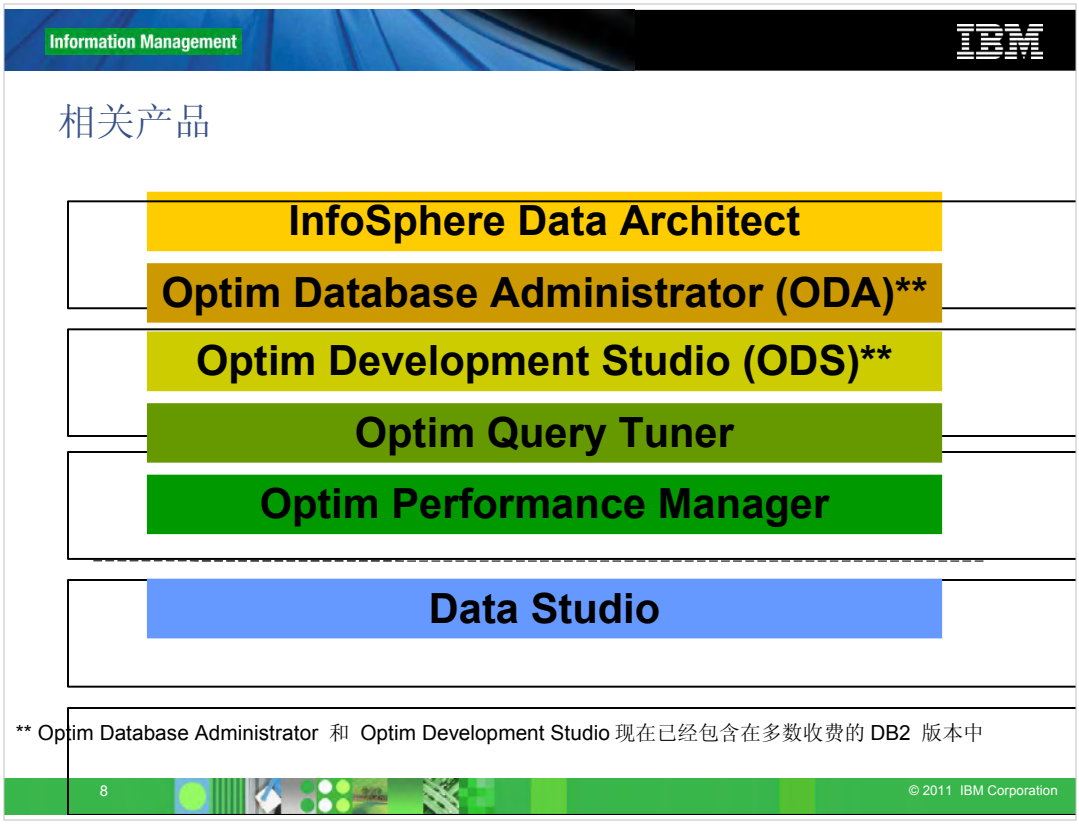
面向 z/OS 的 DB2

Informix Dynamic Server

IBM Data Studio 是一套免费的数据库管理和开发工具。
最新版本是 2.2.1 。它基于 Eclipse 3.4.2 并能够用于：

- 面向 Linux 、 UNIX 或 Windows 的 DB2 (DB2 LUW)
- 面向 iSeries 的 DB2
- 面向 z/OS 的 DB2
- Informix Dynamic Server

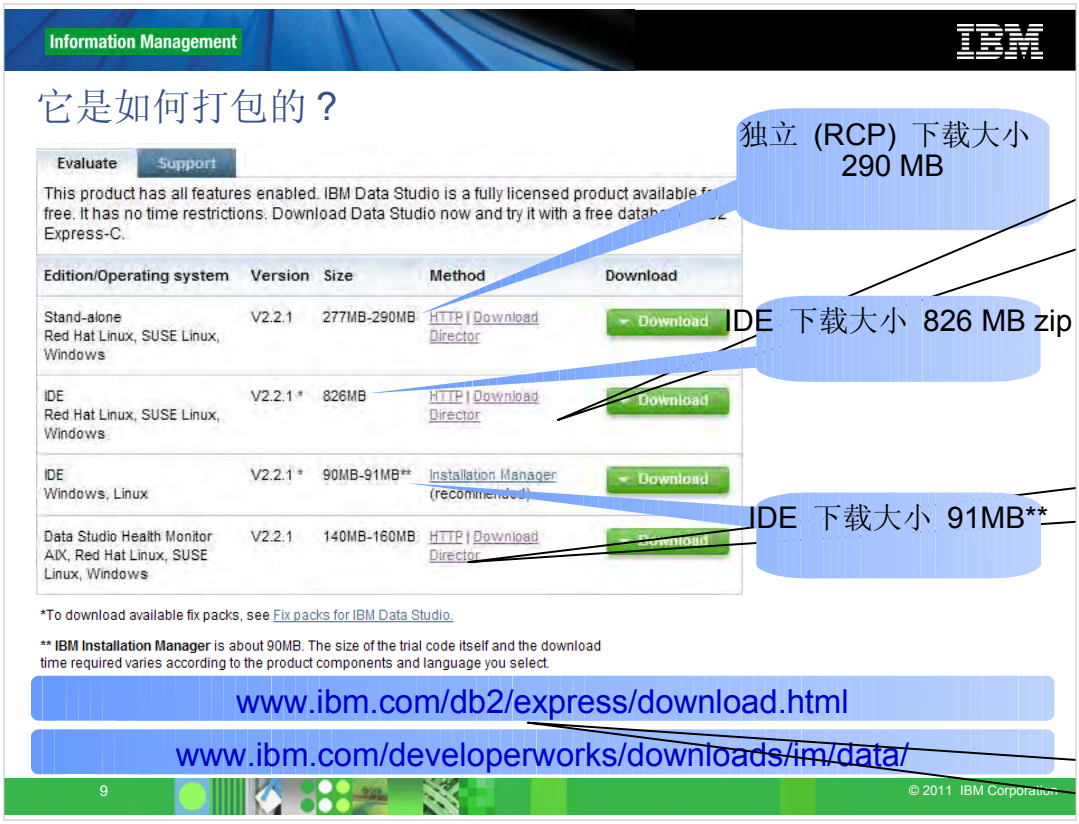
请注意，虽然 Data Studio 以 Eclipse 为基础而且是免费的，但它并不是开源的。



有多款与 IBM Data Studio 相关的收费产品，它们中多数是基于 Eclipse 的。例如，我们有 InfoSphere Data Architect (IDA) （ InfoSphere 数据体系结构），它为数据库设计提供了完整的解决方案。 Data Studio 借用了 IDA 不多的功能，表之间的图形化关系是一个例子。

我们还有 Optim Database Administrator (ODA) （ Optim 数据库管理员）。 ODA 为数据库管理提供了完整的解决方案。 Data Studio 具有 ODA 中的多数功能，但“变更管理”除外。当需要对数据库机构进行修改，而这会影响相互之间关联的多个表时，变更管理会很有用。

利用此功能， ODA 可以处理变更传递问题并保持数据一致性。我们还有 Optim Development Studio (ODS) （ Optim 开发工作台）。它为数据库开发人员提供了一套环境，能够让他们使用函数、开发存储过程、 SQL 脚本、 Web 服务，等等。 Data Studio 也提供了这些功能，但不具有 pureQuery 支持。 pureQuery 是 IBM 公司提供的一套新 Java API，它能够让 Java 开发人员以简单的方式编码。它也能够帮助 DBA 提高应用的性能，而无需修改任何代码。请注意，下面的两个星号指出， Optim Database Administrator 和 Optim Development Studio 现在已经包含在多数收费的 DB2 版本中。因此，比方说，如果你已经购买了 DB2 Workgroup 版本，你就可以免费使用 ODS 和 ODA 了。要注意，为了运行 pureQuery 应用，你需要购买 Optim pureQuery Runtime 。我们还拥有 Optim Query Tuner （查询优化器），它是一套用来帮助你优化查询的工具。 Data Studio 中包含了此工具的部分功能。最后， Optim Performance Manager （ Optim 性能管理器）可以帮助你监控和优化数据库。 Data Studio 也包含 Optim Performance Manager 的部分功能。概括地说， Data Studio 提供的功能可以被视为这些其他产品中某些功能的基础。



Data Studio 以两种软件包来提供：大量方式的，以及 IDE 方式的。
IDE 意味着集成开发环境。在这个软件包中，你可以得到 Data Studio 提供的所有东西，其中包括一套安装管理器。它也能够让你很好地与 IBM 的其他基于 Eclipse 的工具相集成。例如，如果你在系统中安装了 IDA，那么，在安装并启动 Data Studio IDE 软件包之后，你就可以在同一个控制台上看到来自 IDA 的所有菜单，就像是一个工具一样。当你安装更多能够像这样集成的产品时，你会看到更多的菜单出现在相同的控制台中。你可以 "shell-share"（外壳共享），意即你可以在所有这些产品之间共享相同的 Eclipse 。目前， IDE 包非常大。其下载大小达到了 826MB 。 IDE 包可以利用图中下一行所示的不同方法进行下载。因此，可以利用这一不同的软件包，或者另一种方法来下载 IDE 。你只需下载 91 mb 即可。这个软件包将首先下载安装管理组件，然后由安装管理器下载它所需要的组件，而不是整个 Data Studio 软件。如果你已经安装了其他基于 Eclipse 的 IBM 产品，而它们能够与 Data Studio 共享组件，则这种方式将非常有用。另一种包是独立软件包，也称为 RCP ，意即富客户端平台（一个 Eclipse 组件）。这个包具有与 IDE 包完全相同的外观和功能，但它不具有其他基于 Eclipse 的 IBM 产品所具有的可扩展性。对一些功能，它也没有提供支持，例如 Data Web 服务和 XML 编辑器。但是，其下载长度要小许多 (290Mb) 。 Data Studio 独立版本或许就能够满足你的全部需要，它是使用 DB2 Express-C 数据服务器的人的首选。

Data Studio 可以运行于哪些系统？

Windows

Windows 工作站平台 (XP, Vista, Windows 7)
Windows 服务器平台 (2003, 2008)

Linux

Red Hat, SUSE, 并可以工作在其他发布版本中

可以运行在 **32** 位及 **64** 位系统中

对于 64 位系统， Data Studio 以 32 位模式运行

Data Studio 可以运行在 Windows 工作站平台上，例如 (XP, Vista, Windows 7) 和 Windows 服务器平台 (2003, 2008) 。
它也可以运行在 Linux 上。 Redhat 和 SUSE 一般都进行了测试，但其他版本可能也可以使用。
它也可以运行 32 及 64 位系统，虽然在 64 位系统上运行时， Data Studio 实际上采用 32 位模式。

Data Studio 有哪些要求？

最低要求

内存 : 1GB

磁盘空间： 500MB （独立）， 2GB (IDE)

具体要求可以在以下地址找到：

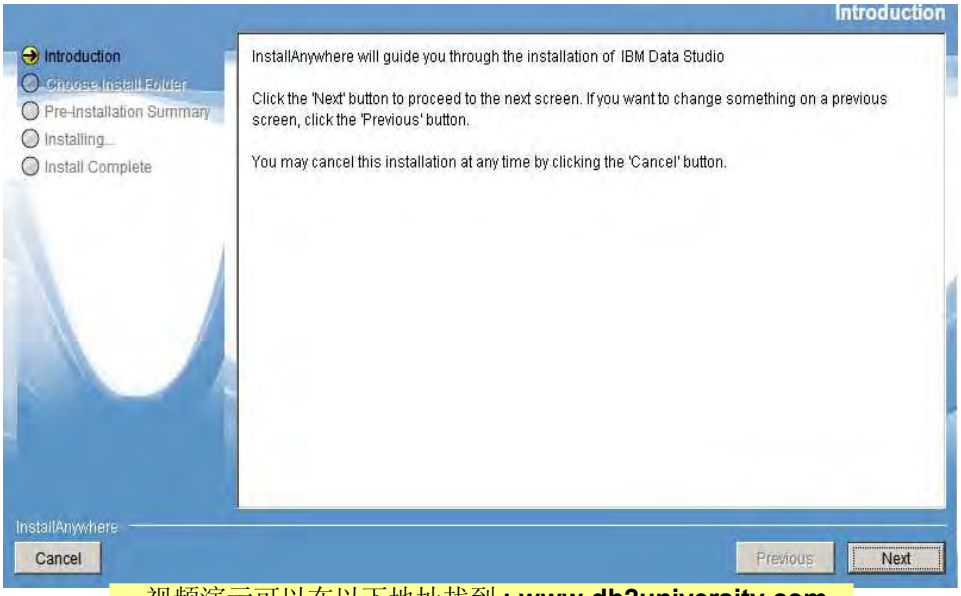
http://publib.boulder.ibm.com/infocenter/idmhelp/ds-v2r2/index.jsp

关于 Data Studio 需求：
最低需要 1GB 内存；对于磁盘空间，独立软件包需要 500 MB， IDE 软件包需要 2GB 。

关于详细信息，请访问所提供的连接。

第一个链接实际上把你带到 DB2 Express-C 网站，不过，该网站上有下载 Data Studio 的链接。而且，你也可以下载 DB2 Express-C，在下载 DB2 Express-C 的过程中，你也可以选择下载 Data Studio。

安装 Data Studio （演示）



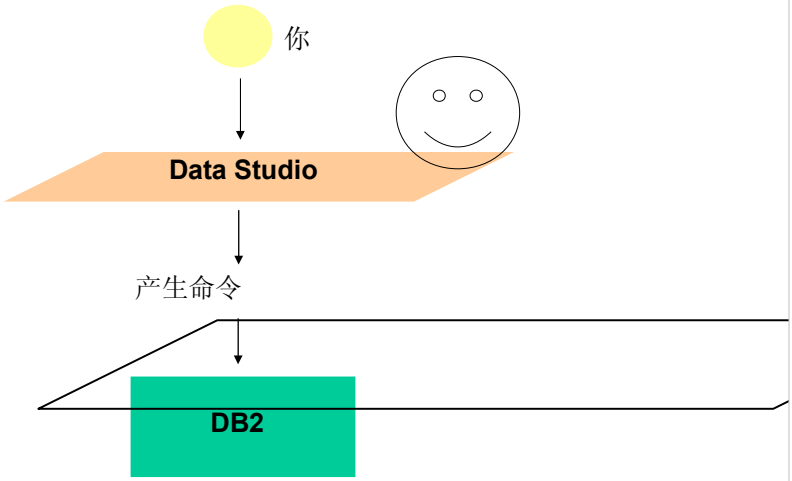
视频演示可以在以下地址找到：**www.db2university.com**

安装过程很直白。选择所有的默认值即可。可以观看 db2university.com 上的演示视频。

议题

- DB2 工具纵览
- Data Studio 概述
- **工作台浏览（演示）**
- 管理连接（演示）
- 使用 SQL 和 SQL 脚本（演示）

有时人们忘记了 GUI 是 ... GUI

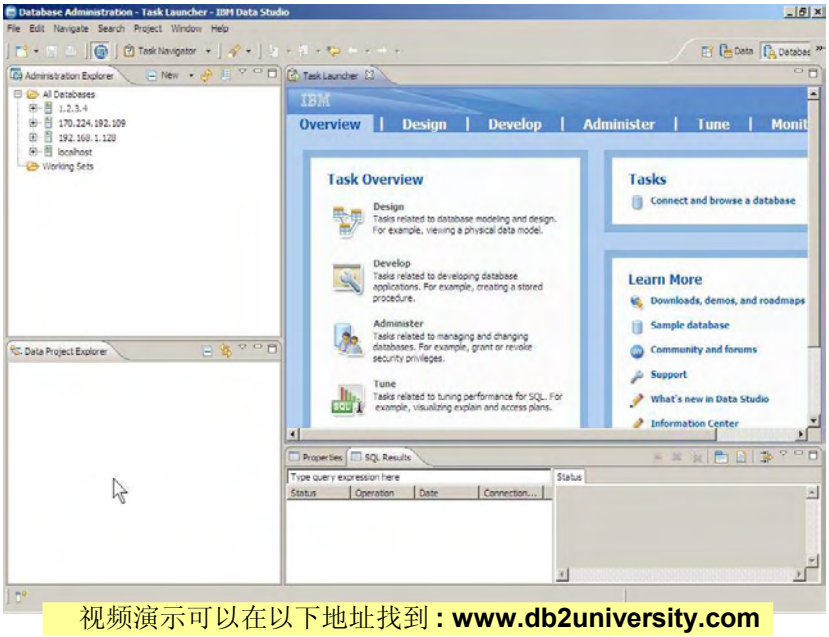


有时人们会忘记 GUI（图形用户接口）就是 GUI。Data Studio 也是一个 GUI，一套图形界面工具，你可以在其中输入你的命令，而 Data Studio 在幕后生成其他命令，这些命令将用于 DB2。如果你的 DB2 基础很扎实，你使用 Data Studio 就绝对无问题。

另一方面，如果你是 DB2 的新手，刚开始使用 Data Studio，或许在某些情况下，你会发现有些东西不是那么直观。例如，在 Data Studio 中，你有一个面向组和用户的文件夹。当你点击一个用户时，有一个选项是“新用户或新组”，这或许会让你觉得，你能够通过 Data Studio 来创建用户和组，而且该用户和组将实际在 DB2 中创建。但实际上在 DB2 中，我们并没有创建用户和组。在一般是由第三方应用来进行的，例如由操作系统（通常的默认方式）或 LDAP 来进行。

因此，再说一遍，请记住，Data Studio 是一个图形界面。我们建议你对 DB2 的工作方式有一定的了解。

Data Studio 工作台浏览



视频演示可以在以下地址找到：**www.db2university.com**

这只能通过演示来观看。 db2university.com 上有视频演示，你也可以在你本地的计算机上自行演示。

议题

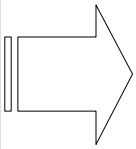
DB2 工具纵览

Data Studio 概述

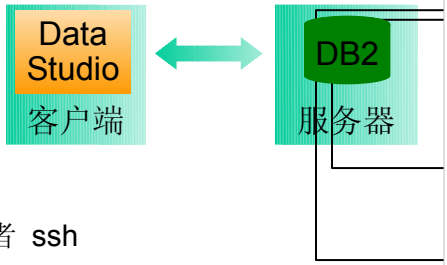
工作台浏览（演示）

管理连接（演示）

使用 SQL 和 SQL 脚本（演示）



Data Studio 连接需求



客户端

- Data Studio 是一个客户端
- Data Studio 通过以下方式接入数据库：
  - JDBC 、数据库管理服务器 (DAS) 或者 ssh

服务器

- 需要安装 DB2
- 有些操作需要 DAS 或 Secure Shell (ssh) 才能进行
- 服务器上不需要其他 Data Studio 组件

视频演示可以在以下地址找到 : **www.db2university.com**

首先是如何在 Data Studio 管理连接的演示（在 db2university.com 上有可以播放的视频）。

这张幻灯片是不言自明的。请注意，在使用 Data Studio 时，某些操作仍然需要 DAS 。它还没有被淘汰。

议题

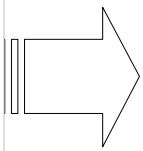
DB2 工具纵览

Data Studio 概述

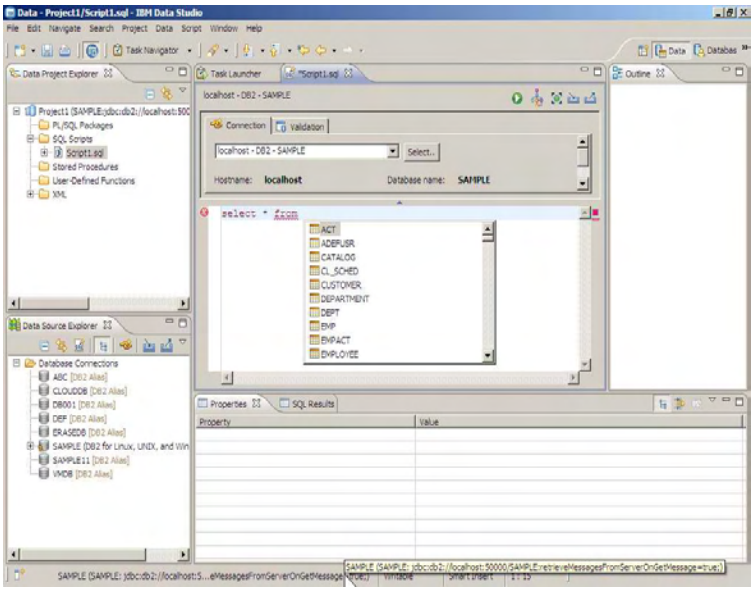
工作台浏览（演示）

管理连接（演示）

使用 SQL 和 SQL 脚本（演示）



使用 SQL 和 SQL 脚本



视频演示可以在以下地址找到 : **www.db2university.com**

示出了一段 db2university.com 视频，演示了我们如何使用 Data Studio 和 SQL

如果你对本课程所讨论的内容有什么技术方面的问题的话，可以利用 db2university.com 课程 AA001EN 中的论坛。 同学、教员以及 IBM 人员可以帮助你！

Information Management

IBM

SQL 和数据库对象简介

IBM 信息管理云计算能力中心
IBM （加拿大）研究院

议题

概述

数据库对象

SQL 简介

SELECT 语句

UPDATE 、 INSERT 和 DELETE 语句

使用 JOIN

支持性阅读材料和视频

阅读材料

DB2 Express-C 入门电子书

第 8 章：使用数据库对象

数据库原理电子书

第 5 章：简介 SQL

IBM Data Studio for DB2 入门电子书

第 4 章：创建 SQL 和 XQuery 脚本

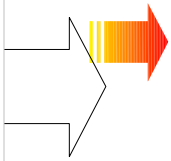
视频

db2university.com 课程 AA001EN

第 6 课：使用数据库对象

第 7 课：SQL 简介

议题



概述

数据库对象

SQL 简介

SELECT 语句

UPDATE、INSERT 和 DELETE 语句

使用 JOIN

本演示中，我们将讨论数据库对象。为了创建和操纵这些对象，我们使用 SQL 语言。

议题

- 概述
- **数据库对象**
- SQL 简介
- SELECT 语句
- UPDATE、INSERT 和 DELETE 语句
- 使用 JOIN

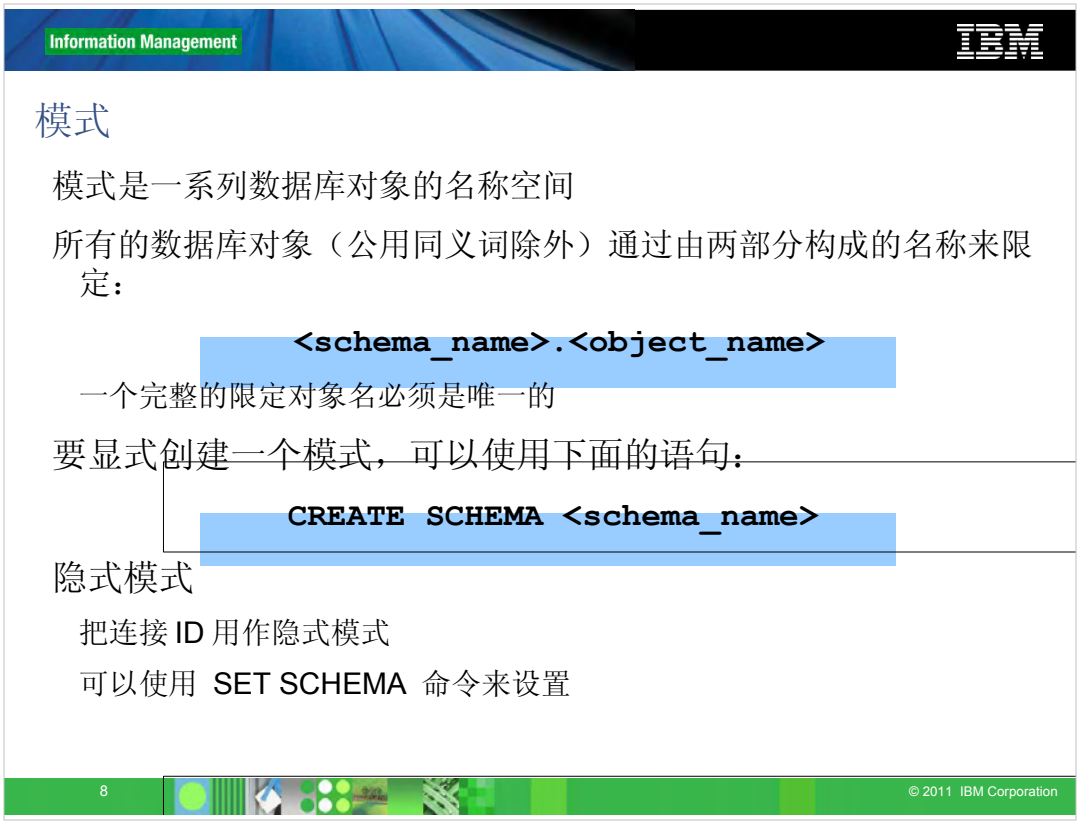
数据库对象

- 模式
- 表
- 视图
- 索引
- 同义词 / 别名
- 数据库应用对象
  - 序列
  - 触发器
  - 用户定义的函数 (UDF)
  - 存储过程

这些对象将在后面关于应用开发的课程中讲述

每种对象都将在后面的演示中进行详细介绍。

对于数据库应用对象，我们在本单元中仅介绍“序列”。剩下的（以红色标识的）将在今后讨论。



在其他的产品中，或者当讨论关系数据库的概念时，“模式” (schema) 通常是指由表、视图等构成的数据库的整个结构。在 DB2 中，模式一般是指对象的前缀或标识符。DB2 中的每个对象名称都有两部分构成：**<schema_name>（模式名称）和 <object_name>（对象名称）。**

在你使用 DB2 中的数据库对象时， schema_name 并非任何时候都需要提供，因为可能使用隐含的模式。

例如： ***db2 connect to sample user db2admin using mypassword --> OK***

db2 create table abc.t1 (col1 int, col2 int) → OK

请注意，“ abc” 并无需映射至操作系统中的任何用户。它只是一个任意的名称。此模式被隐含地创建出来，当然前提是该用户具有 implicit_schema （隐式模式）的特权。

db2 select * from t1 → 错误， 表 “ db2admin.t1” 未找到。

注意， DB2 使用连接的用户 ID 作为隐含模式。

db2 select * from abc.t1 --> OK

db2 create table t1 (col1 int) --> OK, 表 db2admin.t1 被创建

db2 select * from db2admin.t1 --> OK

db2 set schema raul --> OK, 改变当前窗口中的默认模式

db2 select * from t1 --> 错误，表 raul.t1 未找到

DB2 新手常见的错误先使用一个模式创建了对象，后来他们却无法找到这些对象，因为他们是使用另外一个用户 ID 接入的。

模式可以用来把一组表捆绑到一起。或许它们是有关系的。一个模式可能用于测试，另一个用于开发，还有一个用于生产。 DB2 中的某些实用工具，例如 db2move ，可能把模式用作确定应当移动哪个表的选项。

你可能会使用不同模式的场景：

- 比如说，你可能开发了一套应用（例如在 Java 中），而且你嵌入了 SQL 语句，这些语句指向 DB2 对象，但你没有包含模式。 - 在运行时向该程序传递了用户 ID/ 密码连接参数。 - 你有两套表，它们定义了完全一样的东西，但一套在创建时使用的用户 ID 为 “ TEST”，而另一套使用的用户 ID 是 “ DEV”（用于开发）。- 由于该程序指向没有模式的 DB2 对象，则使用的隐含模式就是该连接的 ID 。因此，如果你以用户 TEST 的身份连接，你就会指向 TEST 模式中的表；如果你使用用户 DEV ，你将指向 DEV 模式中的表。

表

EMPLOYEE 表

Lastname	Firstname	Age	Country
Pan	Peter	15	Singapore
Austin	Susan	32	Australia
...			

表存储数据

在逻辑上，表是由按照列和行排列的数据构成

- 每个列含有相同数据类型的值
- 每个行包含各个可用列构成的一组值

关系数据库把数据展示为表的集合。

表

EMPLOYEE 表

Lastname	Firstname	Age	Country
Pan	Peter	15	Singapore
Austin	Susan	32	Australia
...			

字段
（列）

表存储数据

在逻辑上，表是由按照列和行排列的数据构成

每个列含有相同数据类型的值

每个行包含各个可用列构成的一组值

在逻辑上，表是由按照列和行排列的数据构成。

每个列含有相同数据类型的值。

列的存储展示被称为字段 (field) 。**因此，一般情况下，“列”、“字段”或者“属性”是指同一种东西。**

表

EMPLOYEE 表

Lastname	Firstname	Age	Country
Pan	Peter	15	Singapore
Austin	Susan	32	Australia
...			

字段
（列）

表存储数据

在逻辑上，表是由按照列和行排列的数据构成

每个列含有相同数据类型的值

每个行包含各个可用列构成的一组值

表

EMPLOYEE 表

Lastname	Firstname	Age	Country
Pan	Peter	15	Singapore
Austin	Susan	32	Australia
...			

字段
（列）

表存储数据

在逻辑上，表是由按照列和行排列的数据构成

- 每个列含有相同数据类型的值
- 每个行包含各个可用列构成的一组值

表

EMPLOYEE 表

Lastname	Firstname	Age	Country
Pan	Peter	15	Singapore
Austin	Susan	32	Australia
...			

字段
（列）

表存储数据

在逻辑上，表是由按照列和行排列的数据构成

- 每个列含有相同数据类型的值
- 每个行包含各个可用列构成的一组值

表

EMPLOYEE 表

记录（行）

Lastname	Firstname	Age	Country
Pan	Peter	15	Singapore
Austin	Susan	32	Australia
...			

表存储数据

在逻辑上，表是由按照列和行排列的数据构成

- 每个列含有相同数据类型的值
- 每个行包含各个可用列构成的一组值

每个行包含各个可用列构成的一组值。

行的存储展示被称为记录 (record) 。**因此，一般情况下，“行”、“记录”或者“元组”是指同一种东西。**

表

EMPLOYEE Table

记录（行）→

Lastname	Firstname	Age	Country
Pan	Peter	15	Singapore
Austin	Susan	32	Australia
...			

表存储数据

在逻辑上，表是由按照列和行排列的数据构成

每个列含有相同数据类型的值

每个行包含各个可用列构成的一组值

表

EMPLOYEE Table

记录（行）→

Lastname	Firstname	Age	Country
Pan	Peter	15	Singapore
Austin	Susan	32	Australia
...			

表存储数据

在逻辑上，表是由按照列和行排列的数据构成

每个列含有相同数据类型的值

每个行包含各个可用列构成的一组值

表

EMPLOYEE Table

记录（行）→

Lastname	Firstname	Age	Country
Pan	Peter	15	Singapore
Austin	Susan	32	Australia
...			

表存储数据

在逻辑上，表是由按照列和行排列的数据构成

- 每个列含有相同数据类型的值
- 每个行包含各个可用列构成的一组值

表

EMPLOYEE Table

Lastname	Firstname	Age	Country
Pan	Peter	15	Singapore
Austin	Susan	32	Australia
...			

字段
（列）

表存储数据

在逻辑上，表是由按照列和行排列的数据构成

- 每个列含有相同数据类型的值
- 每个行包含各个可用列构成的一组值

行和列的交叉点被称为值。

表

EMPLOYEE Table

记录（行）

Lastname	Firstname	Age	Country
Pan	Peter	15	Singapore
Austin	Susan	32	Australia
...			

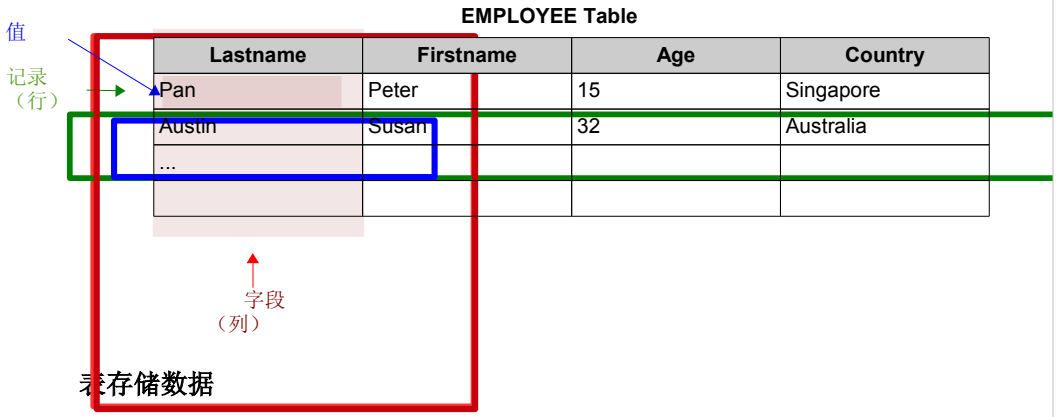
字段（列）

表存储数据

在逻辑上，表是由按照列和行排列的数据构成

- 每个列含有相同数据类型的值
- 每个行包含各个可用列构成的一组值

表



EMPLOYEE Table

Lastname	Firstname	Age	Country
Pan	Peter	15	Singapore
Austin	Susan	32	Australia
...			

表存储数据

在逻辑上，表是由按照列和行排列的数据构成

每个列含有相同数据类型的值

每个行包含各个可用列构成的一组值

同一个列中能够存储不同类型的数据吗？不能。

同一个表中的行会具有不同数量的列吗？不会。

创建表

CREATE TABLE myTable (col1 integer)

myTable

col1
120

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


创建表的方式在几乎所有的数据库产品中都是一样的。需要之处的是，在 DB2 中，如果我们打算把这个表与 表空间 mytbls1 关联起来，我们需要使用子句 ***“ in mytbls1”*** 。如果未指定该子句，表通常会在表空间 USERSPACE1 中创建。

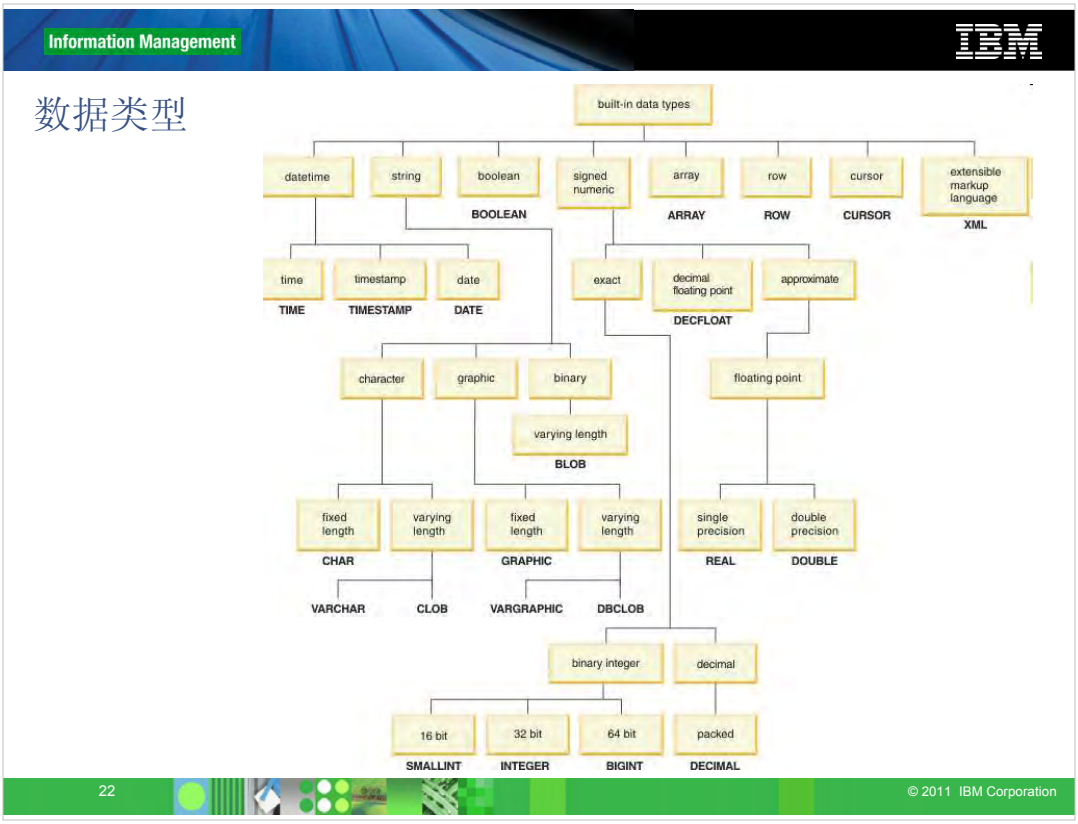
小提示：

1) 为了列出你的当前模式下的所有表 / 视图，使用：
db2 list tables

2) 对于另一个模式，例如 “ arfchong”，则使用：
db2 list tables for schema arfchong

3) 为了描述一个表或视图（查看列和数据类型），你可以：
db2 describe table employee

4) 为了描述 SELECT 结果表的结构，使用：
db2 "describe select * from employee"



这幅图示出了 DB2 的内置数据类型。 XML 数据类型将在 pureXML 演示中讨论。“ Getting stated with DB2 Express-C”（ DB2 Express-C 入门）一书更详细地解释了这些数据类型。

选择适当的数据类型

选择适当的数据类型非常重要，因为它会影响到性能和磁盘空间。为了选择正确的数据类型，你需要了解你的数据的使用方式以及它们的可能取值：

你的数据长度是可变的吗，而且最大长度是 10 个字符吗？使用 CHAR

你的数据长度是可变的吗，而且最小长度是 10 个字符吗？使用 VARCHAR

你的数据长度是固定的吗？使用 CHAR

你的数据将用于排序操作吗？使用 CHAR, VARCHAR,DECIMAL, INTEGER

你的数据将用于算术运算吗？使用 DECIMAL, REAL, DOUBLE, BIGINT, INTEGER, SMALLINT

你的数据需要小数吗？使用 DECIMAL, REAL, DOUBLE, FLOAT

你需要存储非常少量的非传统数据，例如音频或视频吗？使用 CHAR FOR BIT DATA, VARCHAR FOR BIT DATA, LONG VARCHAR FOR BIT DATA

你需要存储非传统数据，例如音频或视频吗？或者数据长度大于字符串的存储能力吗？使用 CLOB, BLOB, DBCLOB

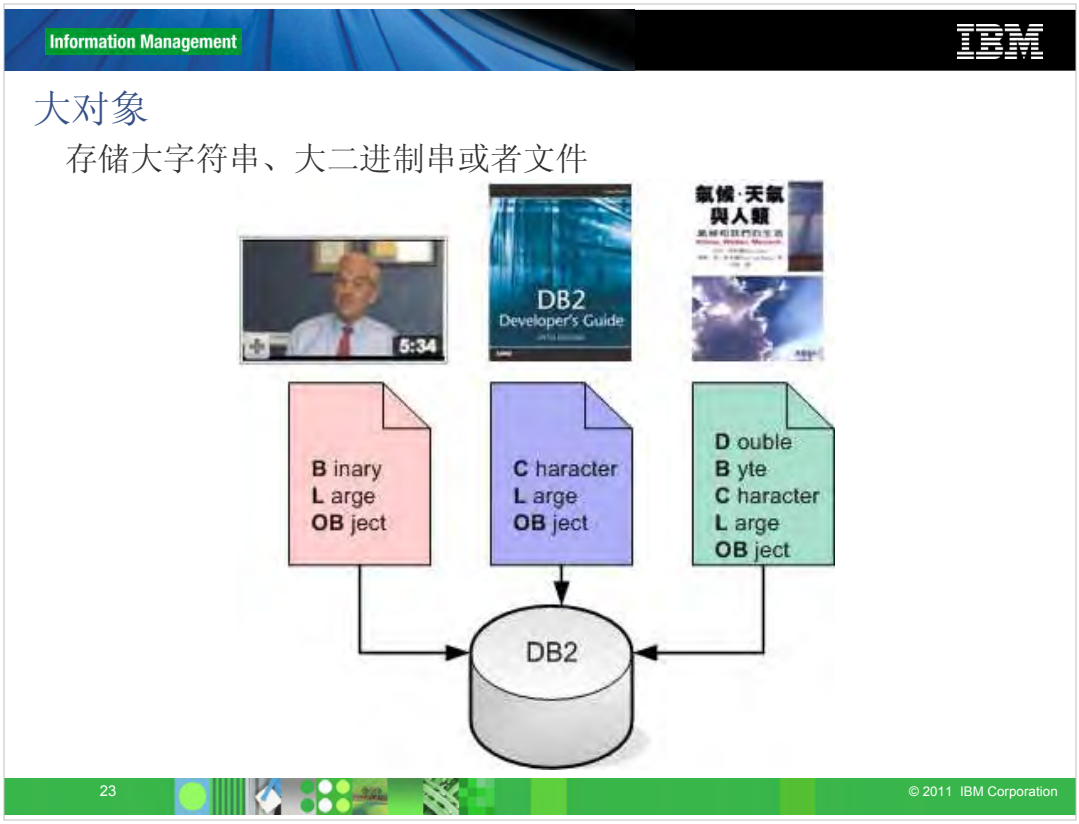
数据包含时间戳信息吗？使用 TIMESTAMP

数据包含时间信息吗？使用 TIME

数据包含日期信息吗？使用 DATE

你需要某种数据类型来实施你的具有特定含义（超出了 DB2 内置的数据类型）的业务规则吗？使用用户定义的类型

你将在数据库中使用 XML 文档吗？使用 XML



我们有支持大对象的数据类型，例如大规模二进制文件（用于视频 / 音频）。数据类型是 BLOB 、 CLOB 和 DBCLOB 。

BLOB 以二进制格式存储可变长度的数据，非常适合于在数据库中存储音频或视频信息。该数据类型有一些限制，例如，你不能对此类型的列进行排序。

CLOB 存储大量的可变长度单字节字符集 (SBCS) 或者多字节字符集 (MBCS) 字符串，例如，大量的文本，诸如白皮书或者长篇文档。

DBCLOB 存储大量的可变长度双字节字符集 (DBCS) 字符串，例如，大量的中文文本。

与 LONG VARCHAR 和 LONG VARGRAPHIC 数据类型类似的是， LOB 可以直接从磁盘上访问，不经由缓冲池，因此，使用 LOB 比使用其他数据类型慢一些。此外，由于对数据库的修改被记录在事务日志文件中，因此，在修改 LOB 列时，日志文件很可能很快就会装满。为防止这种情况出现， **CREATE TABLE** 语句有一个针对 LOB 列的 **NOT LOGGED** 选项。对于长度定义大于 1GB 的 LOB 列， **NOT LOGGED** 是必须的。

CREATE TABLE 语句还有一个用于 LOB 的 **COMPACT** 选项，它用于仅分配所必要的磁盘空间。但是，如果你对 LOB 进行了更新，而这会增加 LOB 的长度，那么， DB2 在此时就需要分配更多的空间，这会导致牺牲性能。请注意，该选项并不会压缩 LOB 。

注

不用使用 LOB 来存储长度低于 32K 的数据。相反，应当使用 VARCHAR 或 VARCHAR FOR BIT DATA ，它们能够容纳最大 32,672 字节。这有助于提高数据库性能。

Null （空值）

NULL 代表一种未知的状态

当某个列中必须包含某种已知的数据值时，请在该列定义中使用 NOT NULL 。

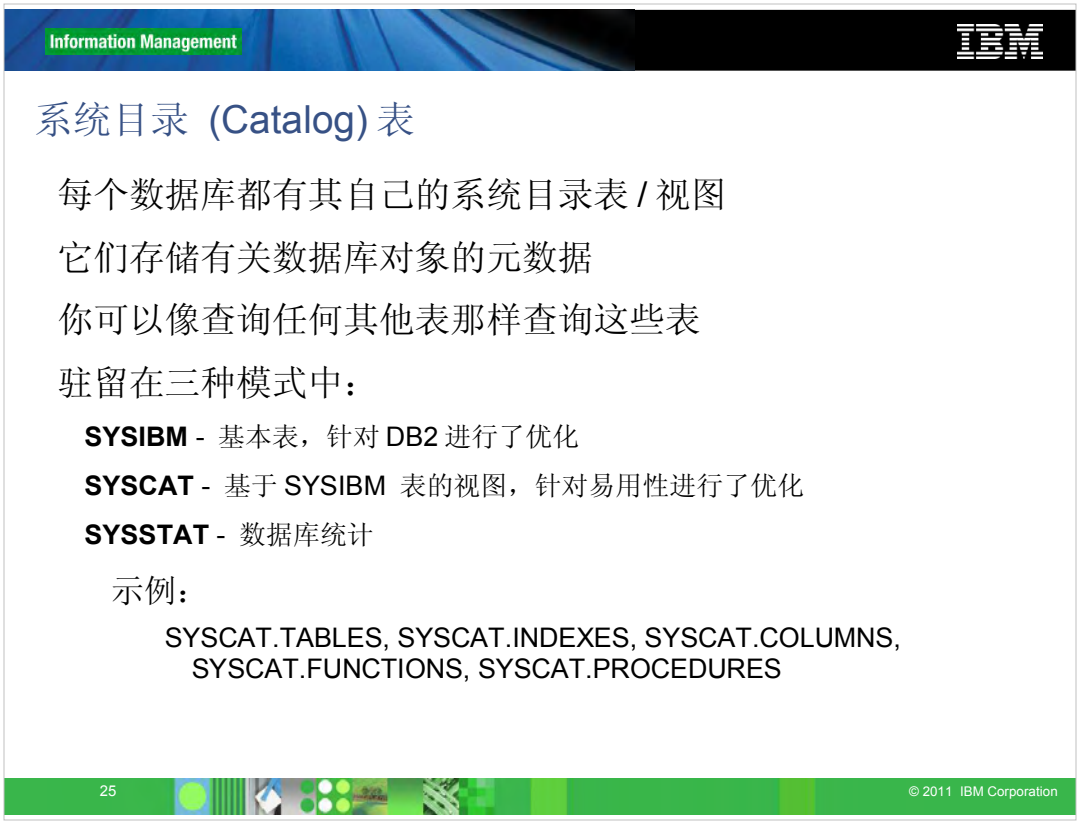
为 NOT NULL 列规定默认值是很有用的

```
CREATE TABLE Staff (
    ID        SMALLINT NOT NULL,
    NAME      VARCHAR(9) ,
    DEPT      SMALLINT NOT NULL with default 10,
    JOB       CHAR(5) ,
    YEARS     SMALLINT ,
    SALARY    DECIMAL(7,2) ,
    COMM      DECIMAL(7,2) with default 15
    )
```


与 Oracle 数据库的行为进行比较：

假如 col1 具有 NULL 值
'xyz' || 'abc' || col1 ==> 'xyzabc'

在 DB2 中
'xyz' || 'abc' || col1 ==> NULL （ANSI 标准)



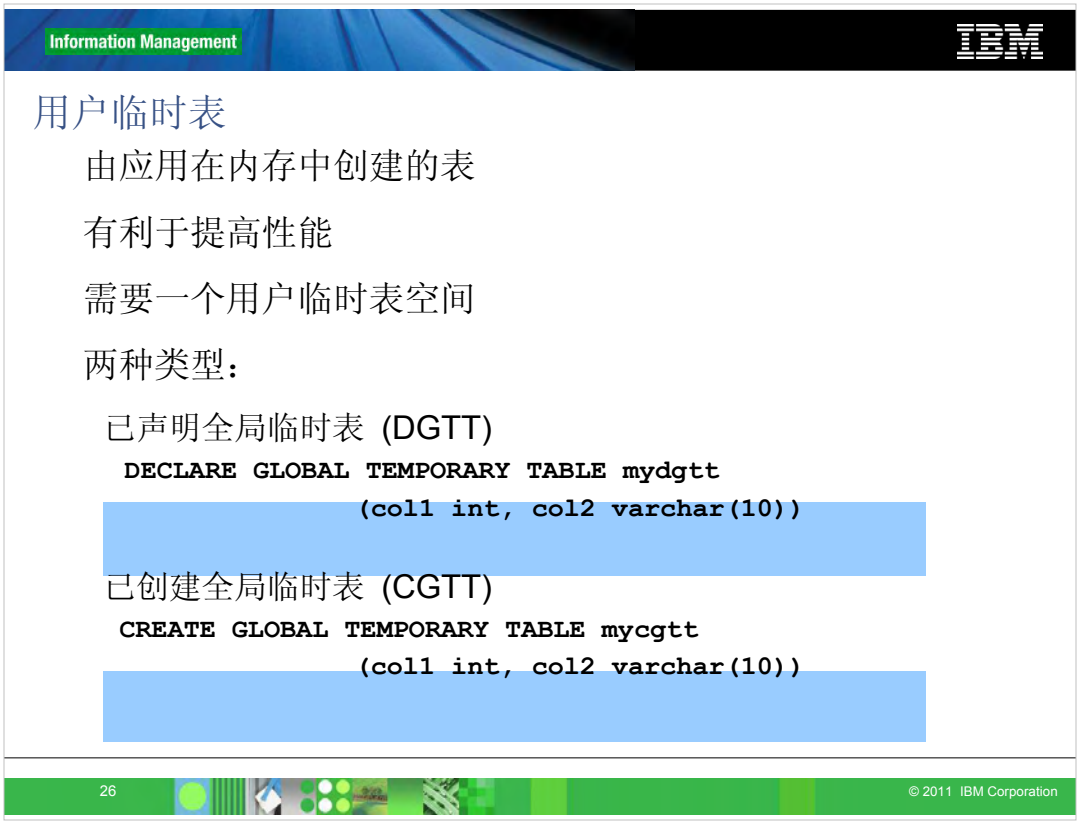
DB2 在创建数据库时将自动创建系统类目表。它们总是驻留在 SYSCATSPACE 表空间中。系统目录表中含有关于数据库中所有对象的信息。例如，当你创建表空间时，其信息将存储在一个或多个系统目录表中。在以后的操作中引用此表空间时， DB2 将检查相应的系统目录表，以查看该表空间是否存在以及是否允许此操作。如果没有系统目录表， DB2 将无法工作。

系统目录表的中包含的某些信息含有以下内容：
- 所有数据库对象的定义
- 表和视图的列数据类型
- 已定义的约束
- 对象特权
- 对象依赖性

系统目录表或视图使用 SYSIBM 、 SYSCAT 或 SYSSTAT 模式。

SYSIBM 模式用于基本系统目录表。
SYSCAT 模式用于系统目录表中定义的视图。 DB2 用户一般应当查询 SYSCAT 视图来获取信息，而不应当查询 SYSIBM 表。
SYSSTAT 模式用于含有数据库统计信息的视图，它也是以系统目录表为基础。

虽然你无法更新驻留在 SYSIBM 和 SYSCAT 模式下的表和视图，但你可以更新 SYSSTAT 模式下的视图。更新这些视图有时可能会影响 DB2 优化器选择不同的访问路径。



临时 (Temporary) 表可以被划分为系统表或用户表。 DB2 在系统临时表空间中管理系统临时表。 DB2 自动创建和删除这些表。由于用户无法控制系统临时表，我们不再讨论它们。

你可以在用户临时表空间中创建用户临时表。例如，以下语句创建了一个称为 *usrtmp4k* 的用户临时表空间。

CREATE USER TEMPORARY TABLESPACE usrtmp4k

MANAGED BY SYSTEM USING ('C:\usrtmp')

用户临时表（以后称为临时表），存储临时数据，也就是在会话或连接结束时将销毁的数据。临时表一般用作需要从一次运算中计算一个大结果集的场合，也用于需要临时存储结果集以便进行进一步处理的场合。虽然事务日志记录也允许处理临时表，但多数用户并不需要记录临时数据。实际上，不对此类表记录事务日志有利于提高性能。临时表仅在一次连接中存在；因此，不存在并发性或锁定方面的问题。为了创建临时表，可以使用 **DECLARE** 语句或 **CREATE GLOBAL TEMPORARY** 语句。这两者之间的差异在于，在第一个语句中，当临时表被删除之后，表 DDL 也将消失；而对于后者 (CGTT)，表结构会继续存在，只是各行被删除。

DB2 为所有的临时表使用模式 *session*，无论接入该数据库的用户 ID 如何。

视图

从其他表或视图中派生出来的虚拟表

在调用时填充内容，以 SELECT 语句为基础

有些视图是可以更新的，另一些是只读的

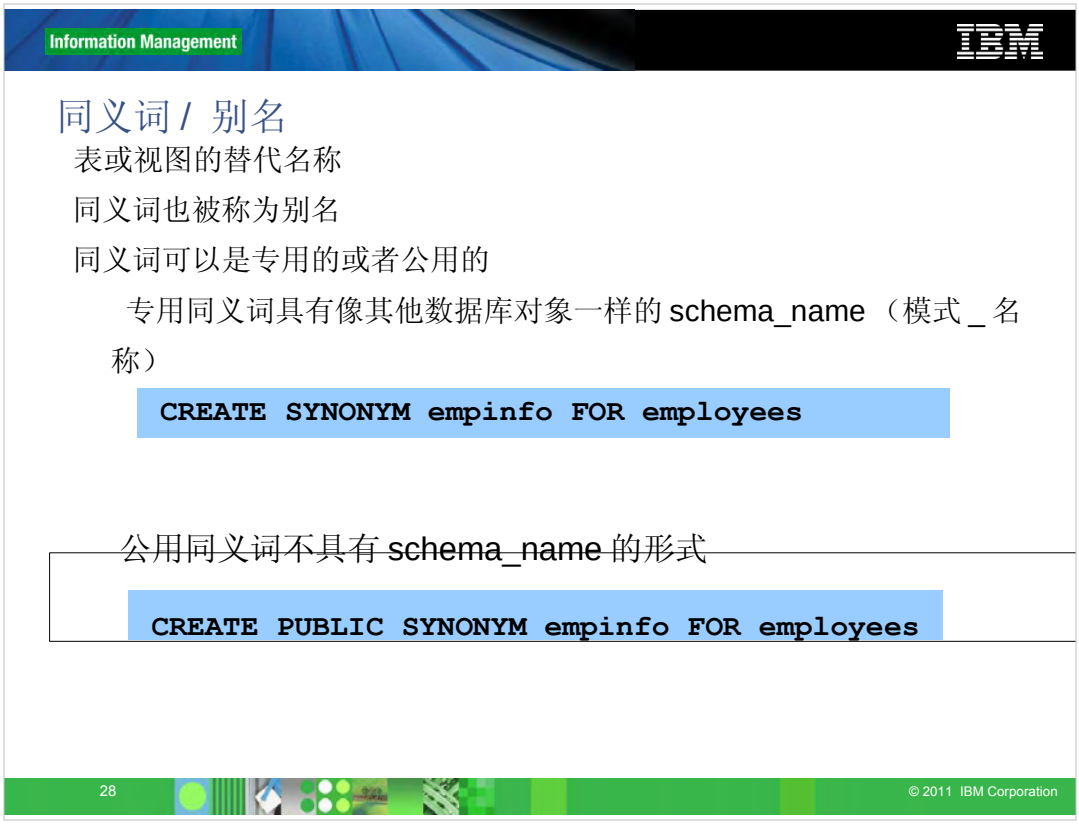
```
CONNECT TO MYDB1

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

```
SELECT * FROM MYVIEW1

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


视图从其他一个或多个表或视图中推导出来的虚拟表。它之所以是虚拟的，是因为它不包含任何数据，只包含根据 **SELECT** 语句的结果所形成的表的定义。

视图无需包含基本表中的全部列。它的列也不必与基本表的列名称相同。你可以使用视图对用户隐藏保密信息。

视图可以是可更新的、可删除的、可插入的或者只读的。在创建视图时，要考虑视图属于哪一类，并相应地需要遵守多项规则。对于可更新的、可插入的、可删除的视图，幕后发生的事情是，基本表是真正被修改的（更新的 / 插入到 / 删除的）



别名，也称为同义词，是为某个数据库对象提供的另一个名称。对于 **PUBLIC** SYNONYMS （公共同义词），你可以创建本同义词而不需要任何模式（甚至不需要隐式的）。它是 DB2 对象中唯一的一个名字不需要由两部分 (模式名 . 对象名) 构成的对象。它可以是“ Object.name”（对象 . 名称）。

例如：

connect to sample user arfchong using mypsw

create public synonym raul for table arfchong.staff

select * from raul

select * from arfchong.raul ## 错误

connect to sample user db2admin using psw

select * from raul

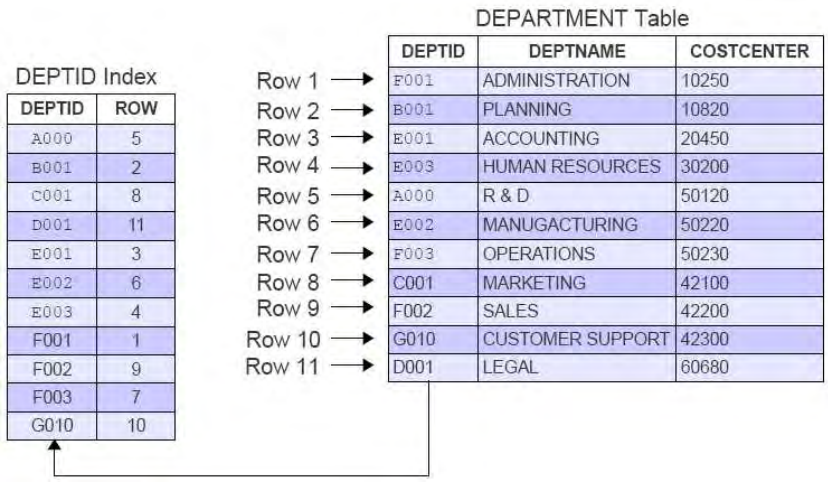
在该例中，你首先以用户 arfchong 的身份连接，并创建公用同义词 raul ，它指向表 arfchong.staff 。该同义词本身并不使用模式。如果你企图使用模式，就会收到错误消息。其他用户，例如 db2admin ，也可以使用同义词 raul ，它是公共的。

如果未使用关键字 PUBLIC ，则所创建的同义词就是专用同义词。

索引

指向基本表中各行的指针的有序集合。

它们以一个或多个列为基础，但是被存储为一个单独的实体。



索引是根据表中的一列或多列构建的数据库对象。之所以使用索引，有两方面的原因：

- 提高查询性能。索引能够用来更迅速地访问数据，因为它能够根据索引键值直接访问各行。
- 当索引被定义为唯一索引时，可以保证唯一性。

创建**集群索引**是为了把索引页面物理地映射到数据页面。就是说，具有相同索引键的所有记录都物理地集中在一起。

索引

有利于提高性能，能够保证唯一性

索引的特性：

- 升序或降序
- 唯一的或者非唯一的
- 复合的
- 集群的
- 双向的（默认的行为）

示例：

```
create unique index artno_ix on artists (artno)
```


索引可以是升序的或者降序的（想想书的索引，通常是按照字母升序来整理书名的）。

唯一索引用来唯一地确定表中的某一行。非唯一索引不能唯一地确定表中的某一行，但仍然有助于提高性能。

复合索引由构成该索引的多个列构成。

集群索引是指索引数据在物理上存储在靠近其相应数据的地方。这有助于提高性能。

双向索引是能够以两种方向中任何一种方向遍历的索引（因此，它是升序的或者降序的）。

序列 (Sequence) 对象

产生一个唯一的数字值
用于数据库层次，独立于表
示例：

```
CREATE SEQUENCE myseq
  START WITH 1
  INCREMENT BY 1
  CACHE 5
```

```
INSERT INTO t1 VALUES (nextval for myseq, ...)
```

```
SELECT prevval for myseq FROM sysibm.sysdummy1
```


序列是一种能够自动产生值的数据库对象。与标识列不同的是，该对象并不依赖于任何表—同一个序列对象可以被整个数据库使用。要创建一个序列，可以使用 **CREATE SEQUENCE** 语句，如这里所示。

```
CREATE SEQUENCE myseq AS INTEGER
    START WITH 1 INCREMENT BY 1
    NO MAXVALUE
    NO CYCLE
    CACHE 5
```

本语句创建了序列 *myseq*，其类型为 INTEGER。该序列起始于值 1，然后，当它为下一个值被调用时，就增加 1。

NO MAXVALUE 子句表示不存在该序列将终止的显式的最大值；因此，它只受数据类型极限的限制，在本例中即为 INTEGER 型值的极限。

NO CYCLE 子句表示该序列在达到极限值以后不会从头再来。

CACHE 5 表示将在内存缓存五个序列数字，而序列中的第六个数字将存储在一个目录表中。把序列值缓存在内存中是为了提高性能；否则的话，DB2 需要不断地访问目录表来顺序检索下一个值。

如果你的计算机崩溃了，而以下几个值（5, 6, 7, 8, 9）还在缓存中，会发生什么情况呢？这些数字就丢失了，下一次当 DB2 需要检索一个数值时，它将从目录表中获取。在这种情况下，10 就是下一个生成的数值。如果你在使用序列值生成唯一的标识符（它必须是连续的，不能有间隔），这种情况就不适合你的要求了。这时，需要使用 **NO CACHE 子句**来保证无间隔地顺序生成数值，但你需要为此付出性能代价。

对于序列值，你可以使用任何小数部分为零的数值型数据类型，包括 SMALLINT、INTEGER、BIGINT 以及 DECIMAL。此外，用户定义的任何以这些数据类型为基础的独特类型也可以容纳序列值。

约束

约束条件能够让你为表中的数据定义规则。
有不同类型的约束：（建议在命名时使用前缀）：

PRIMARY KEY: _pk
UNIQUE: _uq
DEFAULT: _df
CHECK: _ck
FOREIGN KEY: _fk

约束能够让你为表中的数据定义规则。有不同类型的约束：

UNIQUE （唯一性）约束能够防止在表中出现重复值。这时通过唯一索引来实施的，它规定在 CREATE TABLE 语句中，使用关键字 UNIQUE 。

NULL （空值）是 UNIQUE 数据值域的一部分。 PRIMARY KEY 约束类似于 UNIQUE 约束，但它排除了 NULL 作为有效数据。主键总是带有与它相关的约束。

REFERENTIAL （引用）约束用来支持引用完整性，它能够让你管理表之间的关系。

CHECK （检查）约束能够保证你输入到列中的值符合约束中规定的规则。

约束 - 示例

```
CREATE TABLE EMPLOYEE
(
  ID integer NOT NULL CONSTRAINT ID_pk PRIMARY KEY,
  NAME varchar(9),
  DEPT smallint CONSTRAINT dept_ck1
          CHECK (DEPT BETWEEN 10 AND 100),
  JOB char(5) CONSTRAINT dept_ck2
        CHECK (JOB IN ('Sales','Mgr','Clerk')),
  HIREDATE date,
  SALARY decimal(7,2),
  CONSTRAINT yearsal_ck
      CHECK (YEAR(HIREDATE) > 1986 OR SALARY > 40500)
)
```


下面的例子是带有多项 CHECK 约束的表定义，而且它定义了一个 PRIMARY KEY （主键）。对于该表，任何数据在插入之前，必须满足四项约束条件。

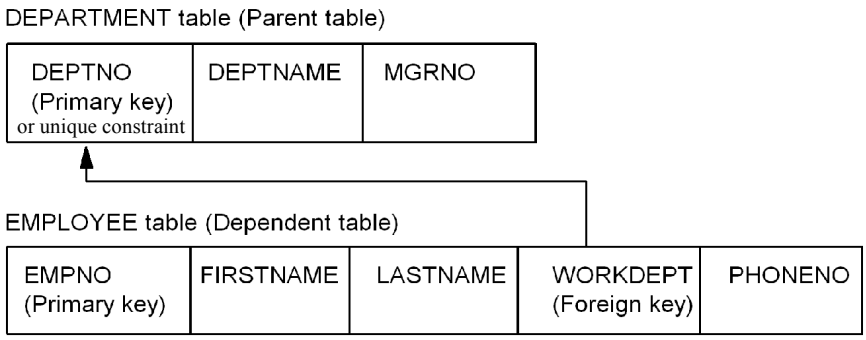
这些约束是：
PRIMARY KEY 约束用于列 ID 。这意味着，不得插入重复值或 null 值。

CHECK 约束用于 DEPT 列。只允许插入介于 10 和 100 之间的值。

CHECK 约束用于 JOB 列。只允许插入 ‘ Sales’ 、 ‘ Mgr’ 或 ‘ Clerk' 这几个值。

CHECK 约束用于 HIREDATE 和 SALARY 列的组合。只允许插入聘用年份大于 1986 而且 SALARY 大于 40500 的数据。

引用完整性




```
create table employee (empno .............
    primary key (empno)
    foreign key (workdept)
    references department on delete restrict)
in DMS01
```


引用完整性 (RI) 在表之间建立关系。通过主键与外键的组合，可以实施数据的有效性。由于无需在应用层次上进行数据级的引用验证 ，引用完整性降低了应用代码复杂性。如果一个表的列值依赖于另一个表中的值，则该表被称为***依赖表***或者***子表；***而被引用的表被称为***基本表***或者***父表。***只有其列被定义为 UNIQUE 或 PRIMARY KEY 的表才可以在其他表中作为外键来引用，以实现引用完整性。

父表：

控制性数据表，父键存在于其中。

依赖表：

依赖于父表中的数据。它也包含一个外键。一个行要想在依赖表中存在，父表中必须首先存在一个匹配的行。

主键：

不能包含 NULL 值，而且所有的值必须是唯一的。一个主键包含表中一个或多个列。

外键：

依赖表中的一个或多个列，用于引用父表中的列，这些列在父表中必须具有主键或唯一性约束。如例所示，在语法上，你无需指出父表的列名称。 DB2 会自动知道的。

请注意，在使用的语法中，在 “ department” 之后，无需指定用来建立 RI 的列 (deptno) 。 DB2 可以自动探测到这一点。但是，我们后面将会看到，你可以使用其他语法来指定父表中的列名称。

引用完整性 – 关于删除或更新的规则

```
create table employee (empno ...
   ... references department on delete restrict)
```

当父表中的某一行被删除或更新时，依赖表中的某一行会发生什么情况呢？

在定义 RI （引用完整性）时可以使用子句 “ *on delete <behavior>*” 或者 “ *on update <behavior>*” 来设定行为：

Behavior	Description
NO ACTION	Do not allow the delete or update to happen on parent row (enforce *after* other referential constraints). This is the default.
RESTRICT	Do not allow the delete or update to happen on parent row (enforce *before* other referential constraints)
CASCADE	Cascade delete of update of parent row to dependant rows
SET NULL	Set all dependant rows to NULL

把 NO ACTION 或 RESTRICT 用作引用约束的删除规则或更新规则能够决定在什么时候实施约束。 RESTRICT 的删除规则和更新规则在所有其他约束之前实施，其中包括那些具有 CASCADE 或 SET NULL 等修改规则的引用约束。 NO ACTION 的删除规则和更新规则在其他引用约束之后实施。明显产生不同行为的一个例子涉及到从一个视图中删除行，而该视图被定义为 UNION ALL 相关表。

表 T1 是表 T3 的父表；删除规则如下所示。

表 T2 是表 T3 的父表；删除规则为 CASCADE 。

```
CREATE VIEW V1 AS SELECT * FROM T1 UNION ALL SELECT * FROM T2
DELETE FROM V1
```

如果表 T1 是表 T3 的父表，具有 RESTRICT 的删除规则，那么，如果在 T3 中有 T1 父键的任何子行，就会发生限制违规 (SQLSTATE 23001) 。

如果表 T1 是表 T3 的父表，具有 NO ACTION 的删除规则，那么，当 NO ACTION 删除规则针对 T1 中的删除实施之前先删除 T2 中的行时，子行可能被 CASCADE 删除规则所删除。如果在 T2 中进行的删除没有导致在 T3 中删除 T1 父键的所有子行时，就会发生约束违规 (SQLSTATE 23504) 。

请注意，根据删除规则或更新规则是 RESTRICT 或 NO ACTION 之不同，返回的 SQLSTATE 也不同。

引用完整性 – 语法 1

```
CREATE TABLE DEPENDANT_TABLE
(
  ID integer CONSTRAINT id_fk
   REFERENCES BASE_TABLE(UNIQUE_OR_PRIMARY_KEY),
  NAME varchar(9),
  ...
);
```


使用此语法，通过用于约束的列（本例中即 “ ID” 列）来定义约束。

引用完整性 – 语法 2

```
CREATE TABLE DEPENDANT_TABLE
(
   ID integer,
   NAME varchar(9),
   ...,
   CONSTRAINT constraint_name FOREIGN KEY (ID)
   REFERENCES BASE_TABLE(UNIQUE_OR_PRIMARY_KEY)
);
```


利用该语法，在所有的列被定义之后才定义约束。

引用完整性 – 语法 3

```
CREATE TABLE DEPENDANT_TABLE
(
   ID INTEGER,
   NAME VARCHAR(9),
    ...
);
```

```
ALTER TABLE DEPENDANT_TABLE
 ADD CONSTRAINT constraint_name FOREIGN KEY (ID)
 REFERENCES BASE_TABLE(UNIQUE_OR_PRIMARY_KEY);
```


利用本语法，约束将在表被创建之后才定义。

在上面的代码样例中，如果没有指定约束名称， DB2 系统将自动生成该名称。所生成的串长度为 15 个字符，例如 ‘ CC1288717696656’ 。

议题

概述

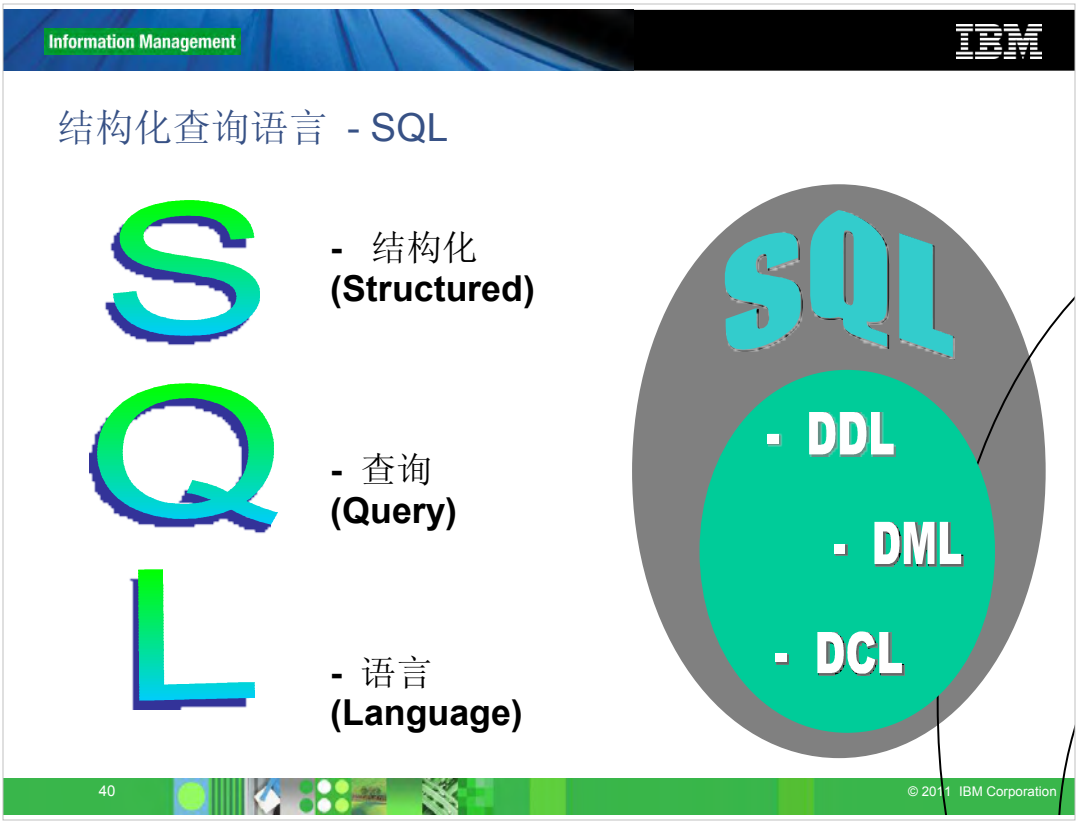
数据库对象

SQL 简介

SELECT 语句

UPDATE 、 INSERT 和 DELETE 语句

使用 JOIN



SQL - 结构化查询语言 - 是标准的关系数据库语言。结构化查询语言是一种高级语言，它能够让用户操纵关系数据。 SQL 的优点之一是，用户只需要指定所需要的信息即可，而无需知道如何检索该信息。数据库管理系统负责确定检索数据所需要的访问路径。 SQL 在集合层次上开展工作，就是说，它在设计上是检索一个或多个表中的行。

根据所涉及的功能， SQL 可以划分为三类：

DDL – 数据定义语言，用于定义、修改或删除数据库对象。

DML – 数据操纵语言，用于读取和修改数据。

DCL – 数据控制语言，用于授予及撤销权限。

有些人也提及 TCL （事务控制语言），它涉及 COMMIT 、 ROLLBACK 、 SAVEPOINT 等语句，它们把 DML 语句组合在一起作为一个单元来执行。

数据定义语言 (DDL)

创建、修改和删除数据库对象

语句：**CREATE, ALTER, DROP, DECLARE**

示例：

```
CREATE TABLE <table name> ...
CREATE SCHEMA <schema name>...
ALTER TABLE <table name>...
ALTER INDEX <index name>...
DROP TABLE <table name>...
DROP INDEX <index name>...
DECLARE GLOBAL TEMPORARY TABLE <table name>...
```

数据定义语言 (DDL)

其他示例：

```
ALTER TABLE myTable
      ALTER COLUMN col1 set not null
```

```
RENAME <object type> <object name> to <new name>
```

```
ALTER TABLE <table name>
   RENAME COLUMN <column name> TO <new name>
```


在数据库对象被创建之后，有可能需要修改其属性，以适应不断变化的业务需求。删除和重建对象是进行这种修改的方法之一；但是，删除对象会产生严重的副作用。修改数据库对象的一个较好方法是使用 ALTER SQL 语句。例如，假设你打算修改一个表定义，以便在某个给定的列中不出现 NULL 值，你可以使用这个 SQL 语句：

ALTER TABLE myTable ALTER COLUMN col1 SET NOT NULL

同样，也可以对表进行其他修改，例如添加或删除列，定义或删除 PRIMARY KEY，等等，这都能够通过使用适当的 alter table（修改表）语法来实现。 ALTER 语句也可以用来处理其他数据库对象。

数据操纵语言 (DML)

检索、插入、更新及删除数据库对象

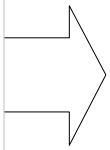
语句： **SELECT, INSERT, UPDATE, DELETE**

示例：

```
SELECT * FROM employee
INSERT INTO EMPLOYEE (name) values ('Peter')
UPDATE EMPLOYEE set name = 'Paul'
DELETE FROM employee
```

议题

- 概述
- 数据库对象
- SQL 简介
- **SELECT 语句**
- UPDATE 、 INSERT 和 DELETE 语句
- 使用 JOIN



SELECT 语句

select * from <tablename>

Select 语句： 查询
查询的结果： 结果集 / 表

数据库管理系统的主要目的不仅仅是存储数据，而是还要便于数据的检索。

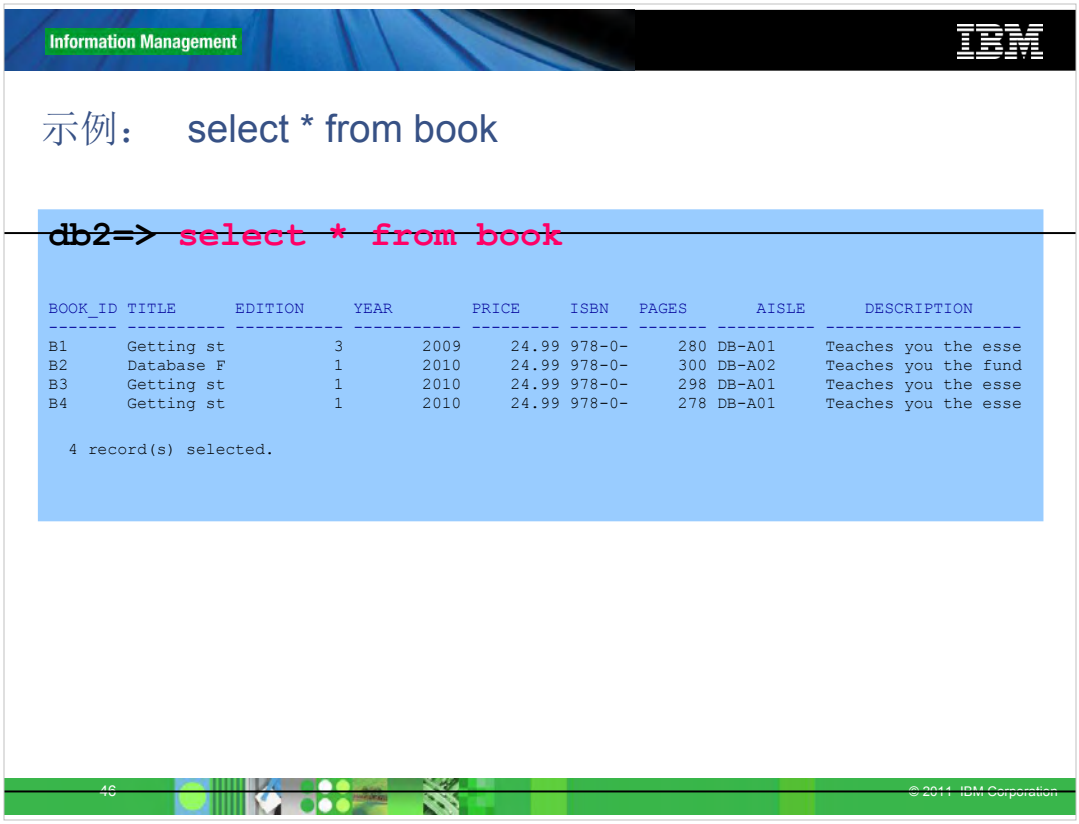
我们如何能够看到存储在表中的数据呢？

这就是 select 语句的用途。

select 语句最简单的形式是

select * from <tablename>”

在这里， SELECT 语句被称为“查询”，我们执行本查询得到的输出被称为结果集或者结果表。



我们以表 book 为例。

select * from book”

从命令提示符下执行时，将会产生类似这样的结果集。

也可以使用 IBM Data Studio 来执行这些 select 语句。

为了看得更清楚一些，我们使用命令行输出。

请注意，这并不是本具体查询的准确输出，因为我们为了显示之目的对结果集做了裁剪。

“*” 在这里是指表中的所有列。

如您所见，表 book 中各个列的各个数据行都显示出来了。

示例：

select <column 1>, … , <column n> from <tablename>

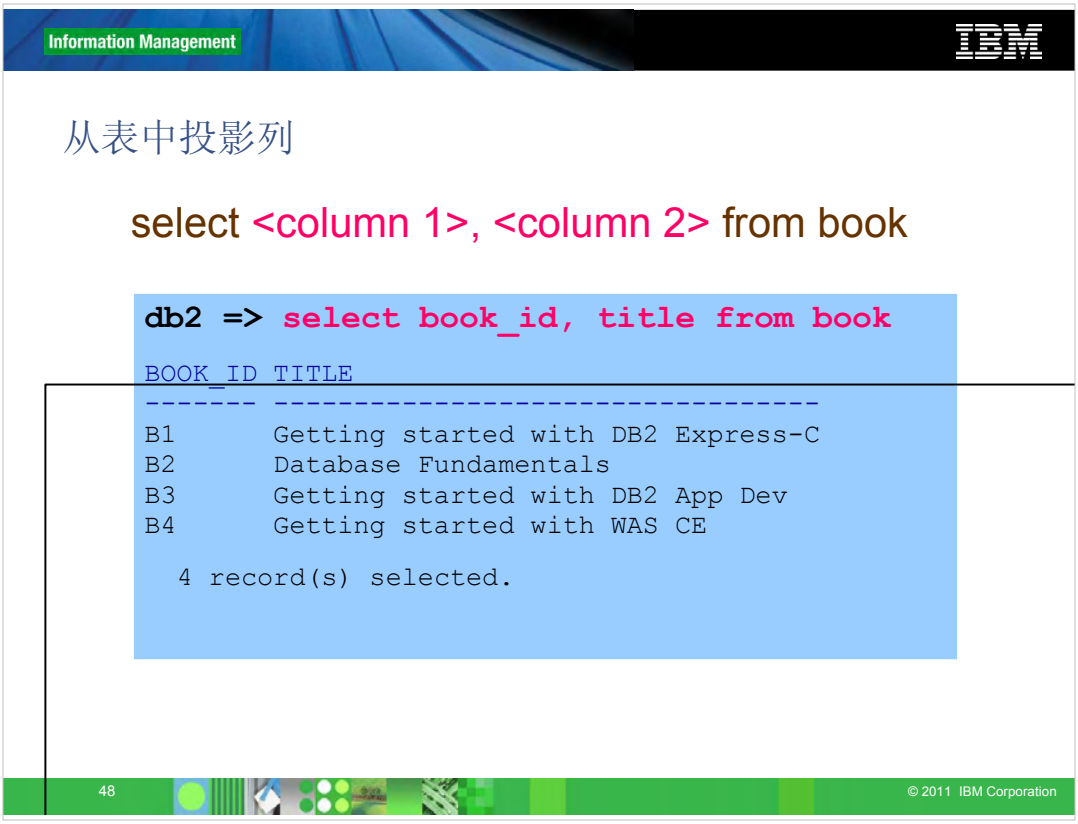
```
db2=> select book_id, title, edition, year, price
```

```
BOOK_ID TITLE      EDITION     YEAR        PRICE     ISBN   PAGES       AISLE      DESCRIPTION
------- ---------- ----------- ----------- --------- ------ ------- ---------- --------------------
B1      Getting st           3        2009     24.99 978-0-     280 DB-A01     Teaches you the esse
B2      Database F           1        2010     24.99 978-0-     300 DB-A02     Teaches you the fund
B3      Getting st           1        2010     24.99 978-0-     298 DB-A01     Teaches you the esse
B4      Getting st           1        2010     24.99 978-0-     278 DB-A01     Teaches you the esse

  4 record(s) selected.
```


通过在 select 语句中具体指出列名称，我们也可以从所有列中检索数据。

这是一个例子。

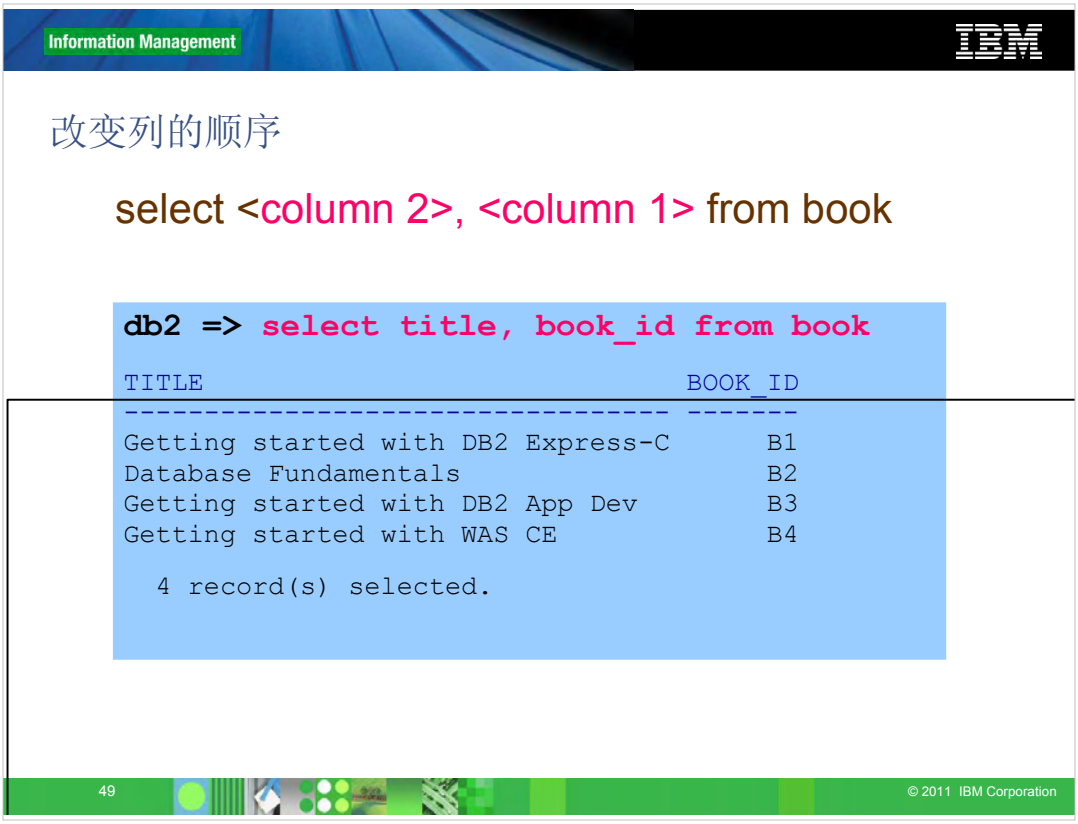
结果集与前面的例子是一样的，但我们在这里使用了列名称。



关系数据库也允许从表中检索列的子集。

在本例中，我们仅从表 book 的 book_id 和 title 两个列中检索数据。

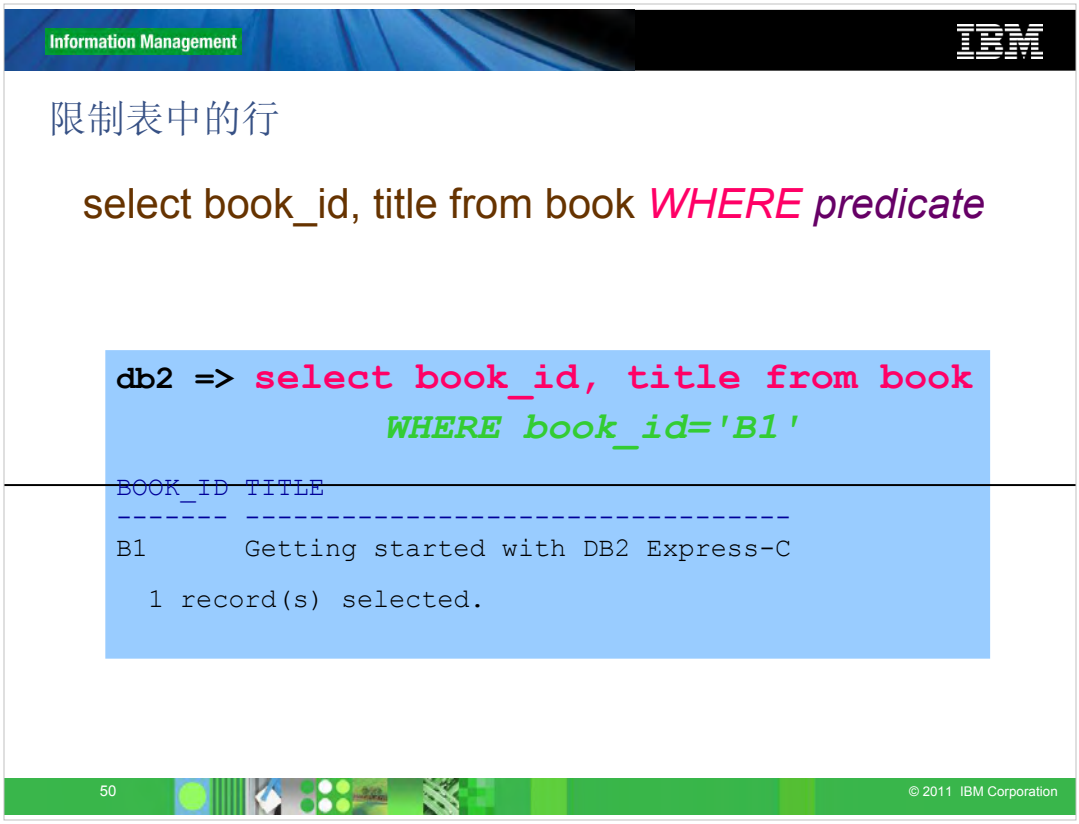
select book_id, title from book



我们来看看如果交换列的排列顺序，会出现什么结果。

Select title, book_id from book.

如你在结果集中所看到的那样，列的顺序总是与传递给 SELECT 语句的顺序相同。
在 create table 语句中定义的顺序被忽略。



我们对这个查询已经很熟悉了，是吧？

select book_id, title from book

假设我们需要对结果集做出一些限制。

例如，我们需要知道 book_id 为 B1 的书籍的名称。

关系运算能够让我们限制结果集，其途经是使用子句“ where”。

Where”总需要一个谓词。

谓词是判断“真”、“假”或“未知”的条件。

就我们的需要而言，我们应当使用带有谓词 book_id='B1' 的 where 子句。

来看看我们刚使用过的查询。

结果集被限制到仅一行，其条件判断为“真”。

限制表中的行（续）

比较运算符

等于	=
大于	>
小于	<
大于等于	>=
小于等于	<=
不等于	<>

前面的例子使用了比较运算符“等于”。
关系数据库管理系统还支持其他的比较运算符。
这是一份比较运算符的列表。等于，大于，小于
大于等于，小于等于，不等于
你可以自己试试这些不同的运算符，看看它们如何工作。

限制结果集合的大小

select book_id, title from book
FETCH FIRST <n> ROWS ONLY

```
db2 => select book_id, title from book
            FETCH FIRST 2 ROWS ONLY

BOOK_ID TITLE
------- ---------------------------------
B1      Getting started with DB2 Express-C
B2      Database Fundamentals

  2 record(s) selected.
```


关系数据库也能够让我们限制待查看的行的数量，虽然结果集里可能会返回许多条记录。

什么使用 “ fetch first <n> rows only”

这是一个限制结果集的子句。

这个例子，

select book_id, title from book FETCH FIRST 2 ROWS ONLY

将把结果集设置为只有 2 行。

利用多重条件限制行数

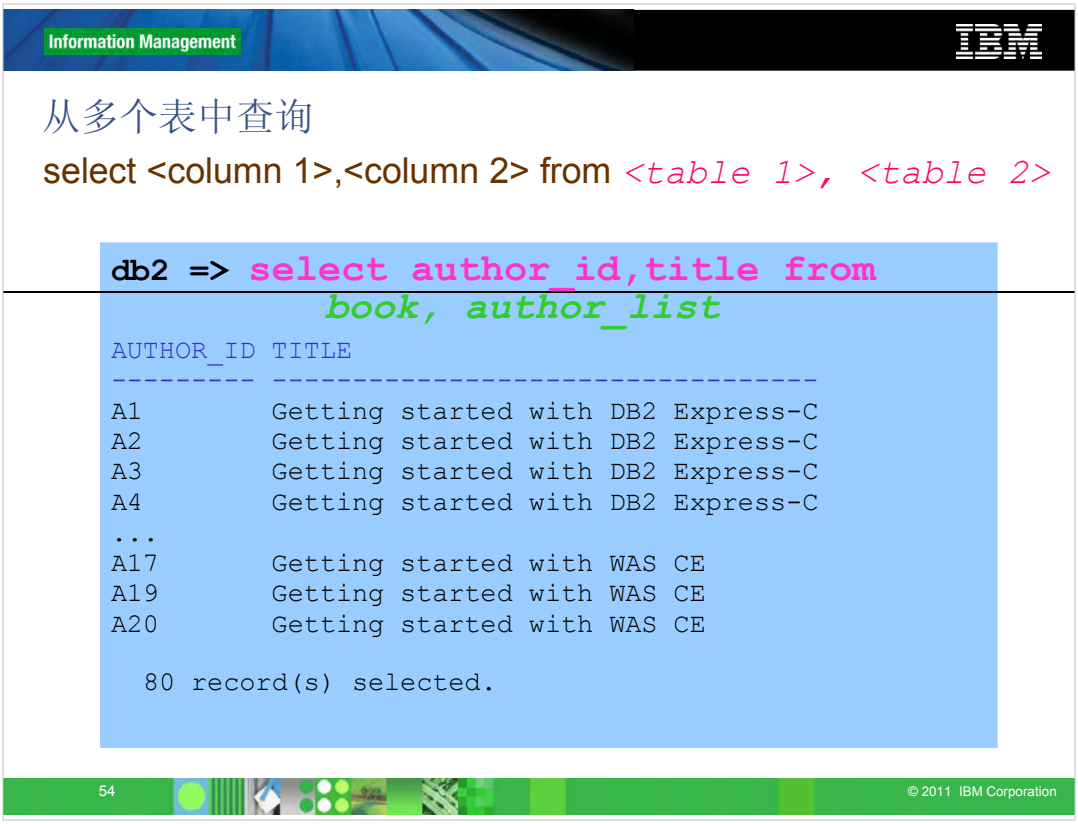
select book_id, title from book where
predicate_1 AND predicate_2

```
db2 => select book_id, title from book wher

BOOK_ID TITLE
------- ----------------------------------
B1      Getting started with DB2 Express-C
B3      Getting started with DB2 App Dev
B4      Getting started with WAS CE

  3 record(s) selected.
```


我们可以使用 Boolean 运算符 AND 来提供更多的条件，从而进一步限制行数。
AND 运算符需要两个谓词，而能够同时满足两个谓词条件的行被称为合格行。
请注意，谓词的顺序不影响结果集。

select book_id, title from book where book_id like
'B%' and title like 'Getting%'.

你或许注意到了这里的字符 % 。
我们将在后面的几个幻灯片中重新讨论这个特殊的字符。



假设我们需要作者的姓和名以及书的题目。你能想出什么简单的办法检索到我们所需要的数据吗？没有一个简单 select 查询能够实现这个要求。背后的原因是，表 book 中仅含有书的详细信息，而表 author 中仅含有作者的详细信息。为了获得书的题目以及该书的作者，我们需要在一次查询中同时调用两张表。

我们来看一个利用两张表来查询数据的简单例子。我们使用表 author_list，它含有 20 行；还有表 book，它含有 4 行。请注意结果集合中的最终结果。

select author_id, title from book, author_list

它显示查询到 80 条记录。

怎么会投射出这么多行呢？

这是两张表的直接乘积。来自 author_list 的 20 行乘以来自表 book 的 4 行，结果总共 80 行。

这种方式称为笛卡尔乘积。再次提请注意，这个输出因为显示的目的被裁剪了。但这并不是我们从多个表中检索时打算看到的东西，对吧？我们可以使用连接 (join) 条件来缩小搜索范围并得到有意义的结果。

从多个表中查询（续）

```
db2 => select title, lastname, firstname from book, aut
         book.book_id=author_list.book_id and author_li
         book.book_id='B1'

TITLE                     LASTNAME     FIRSTNAME
--------------------      ------       -----
Getting started with      CHONG        RAUL
Getting started with      AHUJA        RAV
Getting started with      HAKES        IAN

  3 record(s) selected.
```


在本例中，结果集非常有意义，它显示出书的标题以及作者。但是，看一下这个查询。

是不是有点复杂？

当你真正了解了在什么时候以及如何使用多个表之后，你就不会觉得复杂了。

在使用多个表检索数据时，连接条件发挥着重要作用。

你可以看到，在本例中的查询中，我们告诉数据库搜索这样的行：其在表 book 中的 book_id 与表 author_list 中的 book_id 相匹配，而且表 author_list 中的 author_id 与表 author 中的 author_id 相匹配。

我们还传递了另一个条件： book_id 只能是 B1 。

为了显示的目的，这里的结果集数据值被裁剪了。

谈论表 book 中的 book_id 、表 author table 中的 author_id 等等是不是让人觉得云山雾绕？

在编写查询时也可能会使人一头雾水。因此，我们采用了相关名。它们只是在 SQL 语句中被引用的表的别名。

这些相关名紧跟在 FROM 子句中的表名称后面。

相关名

```
db2 => select title,lastname,firstname from book B, a

TITLE                     LASTNAME   FIRSTNAME
--------------------      --------   ---------
Getting started with      CHONG      RAUL
Getting started with      AHUJA      RAV
Getting started with      HAKES      IAN

  3 record(s) selected.
```


我们用相关名重写我们前面的查询。

只用一个字母 B 提供给表 book， AL 提供给表 author_list， A 提供给表 author 。

对于连接条件，使用所提供的相关名对表中的列进行限定。

再说一遍，为了显示的目的，结果集中的数据值被裁剪了。

对结果集合进行排序

```
db2 => select title from book

TITLE
---------------------------------
Getting started with DB2 Express-C
Database Fundamentals
Getting started with DB2 App Dev
Getting started with WAS CE

  4 record(s) selected.
```


这是以下查询形成的简单的结果集。

“select title from book”.

顺序看起来不太好，是吧？关系数据库提供了一个选项对输出进行排序。 Order by 子句在查询中被用来依据指定的列对结果集进行排序。

Order By 子句

```
db2 => select title from book
          order by title

TITLE
---------------------------------
Database Fundamentals
Getting started with DB2 App Dev
Getting started with DB2 Express-C
Getting started with WAS CE

  4 record(s) selected.
```


看看这一句： select title from book order by title

我们曾经使用 order by 对列 title 进行排序

默认情况下，它以升序排序。

为了按降序排序，我们只需使用关键字 desc 。

结果集现在将按照指定的列以降序排序。

Order By 子句（续）

```
db2 => select title from book
           order by title desc

TITLE
---------------------------------
Getting started with WAS CE
Getting started with DB2 Express-C
Getting started with DB2 App Dev
Database Fundamentals

  4 record(s) selected.
```


在这个例子中，对 title 列按降序排序。

注意前三条记录的顺序。

题目中的前 3 个字都是一样的。

排序从字符不同之处开始。

还有另一种方式来指定需要对哪些列排序。

Order By 子句（续）

```
db2 => select title, pages from book
                order by 2

TITLE                                PAGES
-------------------------------      -------
Getting started with WAS CE          278
Getting started with DB2 Express-C   280
Getting started with DB2 App Dev     298
Database Fundamentals                300

  4 record(s) selected.
```


本例： select title, pages from book order by 2 指出了查询中用于排序的列顺序号码。
我们没有指出列名称 pages ， 而是使用了 # 2 。
列 “ pages” 在查询的列名单中处于第二的位置，因此使用 # 2 来排序。

消除重复

```
db2 => select country from author order by 1
COUNTRY
-------
AU
BR
...
CN
CN
...
IN
IN
IN
...
RO
RO

  20 record(s) selected.
```


这个查询列出了作者所属的国家。

select country from author order by 1

有大约 20 名作者来自不同的国家，其中有些作者来自与另一些作者相同的国家。

在我们罗列出来的内容中，存在一些重复。此时我们只是想知道作者们都来自哪些国家。

这些重复值的确毫无意义。

对此，消除重复是最可能的途经。

要想消除重复，可以使用关键字“ distinct”（不同的）。

Distinct 子句

```
db2 => select distinct(country) from author

COUNTRY
-------
AU
BR
CA
CN
IN
RO

  6 record(s) selected.
```


看看这个例子，它把结果集缩减到仅仅 6 行，而原来的例子中有 20 行。

select distinct(country) from author

我们现在想知道来自同一个国家的作者的数量。

你觉得我们现在应该怎么办？

是的，对他们计数能够让我们获得来自每个国家的作者的准确人数。

为此，我们使用子句“ group by”。

Group by 子句

```
db2 => select country, count(country) from author g

COUNTRY 2
------- -----------
AU                1
BR                1
CA                3
CN                6
IN                6
RO                3

  6 record(s) selected.
```


“group by”子句把结果分组为多个子集，每个子集对一个或多个列具有匹配的值。

在我们的例子中，对国家进行了分组，然后进行计数。

请注意在显示的结果集中第 2 列的列标题。

它显示出一个数字值 2，因为它是结果集中的第 2 列，它是推导出来的列，不是表中直接存在的东西。

Group by 子句（续）

```
db2 => select country, count(country)
       as count from author group by country

COUNTRY COUNT
------- -----------
AU                1
BR                1
CA                3
CN                6
IN                6
RO                3

  6 record(s) selected.
```


我们可以在该查询中为所推导出来的这些列传递一个有意义的名字，以用于在结果集中显示出来。

现在我们已经有了来自不同国家的作者的计数，我们可以通过传递一些条件来进一步限制列的数量。

例如，我们可以查看是不是存在超过 4 名作者来自同一个国家的情况。

要想向 group by 子句传递条件，我们使用关键字 HAVING 。

Having 子句

```
db2 => select country, count(country) as co

COUNTRY COUNT
------- -----------
CN                6
IN                6

  2 record(s) selected.
```


Having 子句与 Group By 子句一起使用。

很重要的一点是，请注意，“ where”子句是用于整个结果集的，而“ having”子句仅与“ group by”从句相配合。

我们可以使用以下语句来进一步限制结果集。

select country, count(country) as count from author

group by country having count(country) > 4

字符串模式

```
db2 => select firstname from author
where firstname like 'R%'

FIRSTNAME
---------------
RAUL
RAV

  2 record(s) selected.
```


如果我们只记得数据行中的某些字母而不是整个值，我们如何检索数据呢？

你是否觉得这种检索有些困难？

在关系数据库中并不困难。

它提供了字符串模式，可以用来搜索匹配此类条件的数据行。

我们来看一些例子。

select firstname from author where firstname like'R%'

在字母 R 后使用 % 意味着，该数据起始于 R，后跟任何字母或数字。

此外， % 也意味着 R 之后的字符的数量没有限制。

串模式（续）

```
db2 => select firstname, lastname from autl

FIRSTNAME       LASTNAME
--------------- ---------------
Jiang Lin       Quan
Dai             Xuan
Chi Run         Hua

  3 record(s) selected.
```


这个串模式的另一个用法是采用字符‘_’。

如你在本例中所见，这仅仅取代一个字符。

select firstname, lastname from author

where lastname like '_ua%'

我们搜索具有以下特征的作者的姓：它起始于任何字符，后跟字母 ua，在之后则可以是任何东西。

可以看到，本查询发现了 3 名作者。

示例：

```
db2 => select title, pages from book where pa

TITLE                              PAGES
---------------------------------- -----------

Database Fundamentals                      300

Getting started with DB2 App Dev           298

  2 record(s) selected.
```


截至目前，我们已经根据数学运算符和串模式检索过数据记录。
我们来假设需要寻找其页数大于 290 但低于 300 的书籍。
你是否觉得我们应当编写这样的查询？

```
select title, pages from book
where pages >= 290 AND pages <= 300
```

当然可以。我们能够检索到所需要的数据。

在一定范围内检索

select <column 1>, <column 2> from <tablename>
where <column n> ***BETWEEN <starting range> AND <ending range>***

```
db2 => select title, pages from book where pa

TITLE                              PAGES
---------------------------------- --------
Database Fundamentals                   300
Getting started with DB2 App Dev        298

  2 record(s) selected.
```


我们来利用一个新的子句“ between-and” 重写这个查询。

select title, pages from book where pages between 290 and 300

结果集与我们前面的查询是一样的。

只是这是进行范围搜索的更好方法，因为范围值是包含在内的。

示例：

```
db2 => insert into book values ('B5','For Example',5,
DB20000I  The SQL command completed successfully.
```


为了介绍下面两个运算符， IS 和 IS NOT，
我们利用下面的值向表 Book 中插入一行。
如你所看到的那样，我们为 year 列插入了一个 NULL 值。

检索 NULL

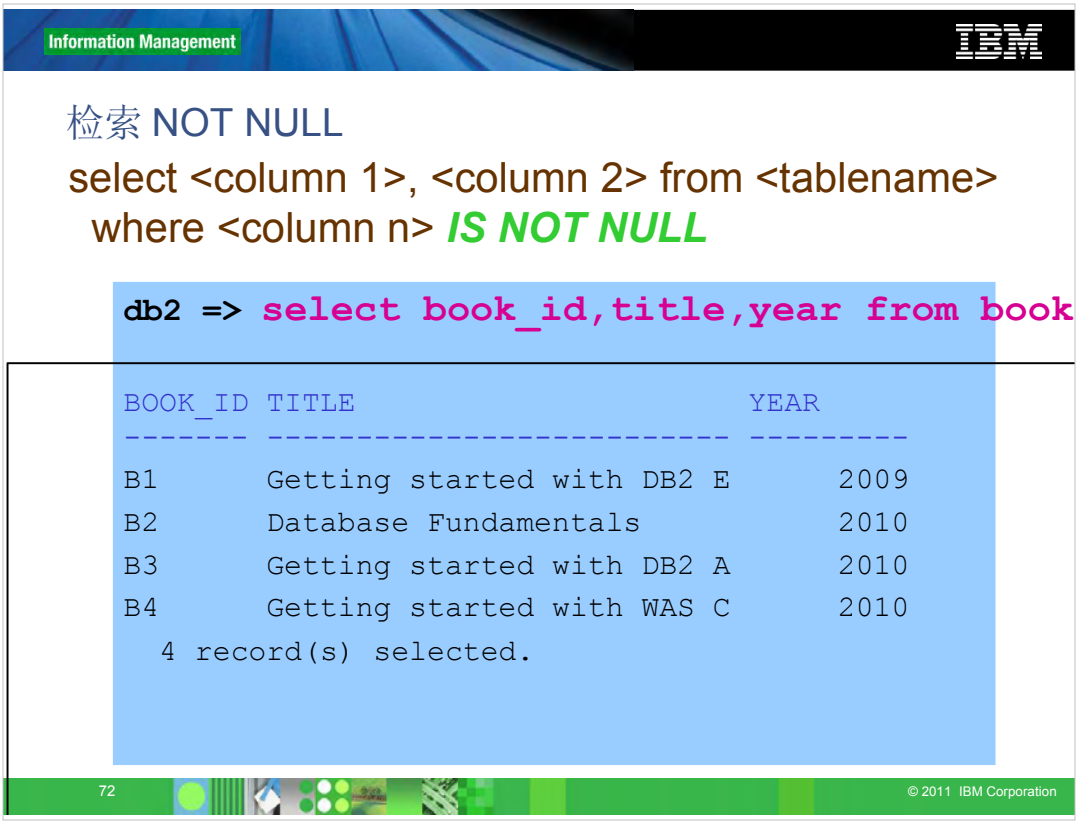
select <column 1>, <column 2> from <tablename>
where <column n> ***IS NULL***

```
db2 => select book_id,title,year from book

BOOK_ID TITLE                     YEAR
------- ------------------------- -----------
B5      For Example                         -

  1 record(s) selected.
```


我们来看看当一个特定值为 NULL 时应该如何检索数据。

Is 运算符用于获取 NULL 。

select book_id, title year from book where year IS NULL



同样， IS NOT 运算符用于获取属于 NOT NULL 的值。

select book_id, title, year from book where year is NOT NULL

示例：

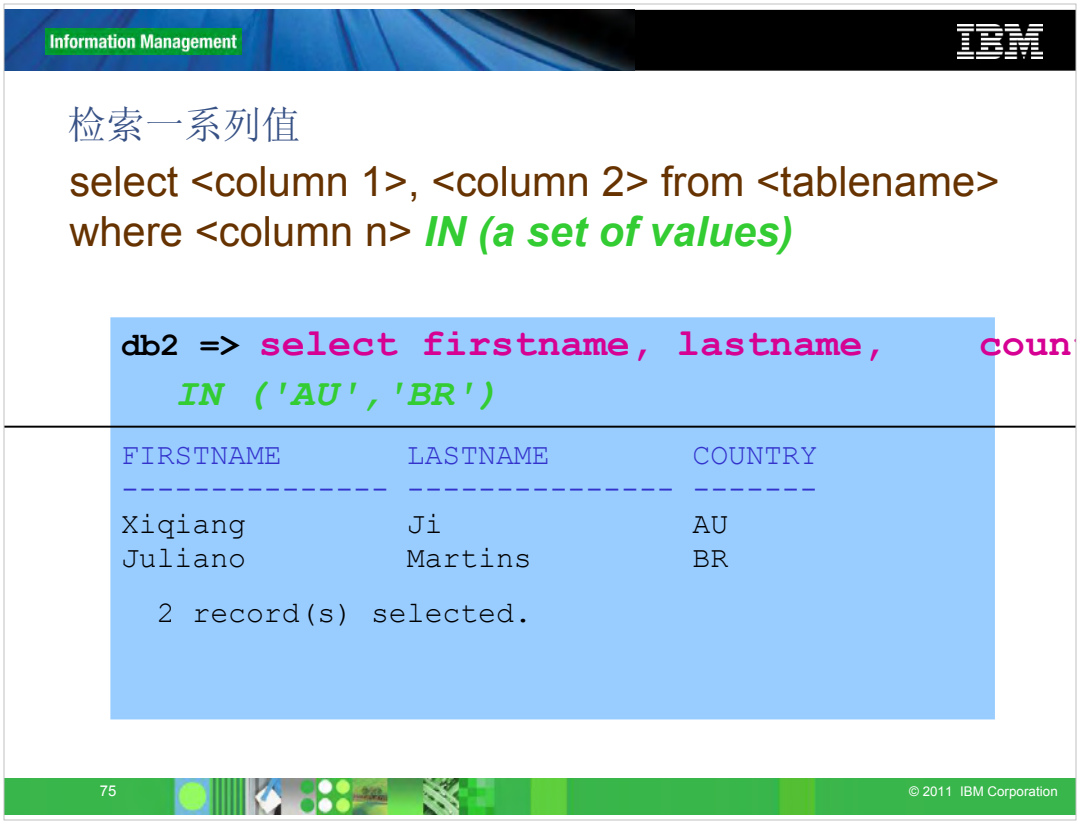
```
db2 => select firstname, lastname, country fr

FIRSTNAME       LASTNAME        COUNTRY
--------------- --------------- -------
Xiqiang         Ji              AU
Juliano         Martins         BR

  2 record(s) selected.
```


有些情况下，数据值无法按照范围分组，例如 country （国家）。
如果我们需要查找来自 AU 和 BR 两国的作者，我们必须使用这样的查询：
select firstname, lastname, country from author where
country='AU' OR country'='BR'
这是一个可以使用的简单语句。

续 ...

```
db2 => select firstname, lastname, coun
country='CA' or country='IN' or country
```


如果我们需要列出来自 CA 、 IN 、
RO 和 CN 这些国家的作者，该怎么办？
select firstname, lastname, country from author where
country='CA' or country='IN' or country='RO' or country='CN'
列出这么长的条件时，需要十分小心。
如果不这样，使用 IN 运算符如何？



IN 运算符能够让我们在 where 子句中规定多个值。

这个运算符将使用待比较的表达式列表。

下面的语句将返回属于 AU 和 BR 国家的作者的信息。

select firstname, lastname, country from author where country IN ('AU','BR')

议题

概述

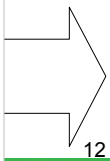
数据库对象

SQL 简介

SELECT 语句

UPDATE、INSERT 和 DELETE 语句

使用 JOIN



INSERT 语句

添加新行到表中或者可更新的视图中

向可更新的试图中添加数据时，数据将被添加到基础表中

语法：

```
INSERT INTO [TableName | Viewname]
<([ColumnName],...)>
VALUES ([Value],...)
```


在创建了一个表之后，需要向其中填充数据。

INSERT SQL 语句可以直接用于待填充的表或者指向待填充表的可更新视图。我们可以使用该语句每次插入一行。

INSERT 语句的语法是这样的：

INSERT INTO [TableName | ViewName]

ColumnName, VALUES

其中 INSERT 和 INTO 是关键字。

ColumnName 表示表或视图中的一列或多列。

Values 可以指定一个或多个数据值，用于插入到所规定的表中或可更新视图中各列。

在 *values* 子句中所提供的值的数量必须等于在 column name 列表中所指定的列名称的数量。

INSERT 语句（续）

示例：

```
INSERT INTO AUTHOR
  (AUTHOR_ID, LASTNAME, FIRSTNAME, EMAIL, CITY, COUNTRY)
VALUES
   ('A1','CHONG','RAUL','RFC@IBM.COM','TORONTO','CA')
```

VALUES 子句规定了待添加到指定的表或指定的可更新视图的列中的数据值。

假设你希望向名为 AUTHOR 的基本表添加一条记录，它有以下列： AUTHOR_ID LASTNAME,FIRSTNAME,EMAIL,CITY,COUNTRY 。

INSERT 语句看起来是这样的：

INSERT INTO AUTHOR 后跟列名称以及需要插入表中的值。

INSERT 语句（续）

语法：

```
INSERT INTO [TableName | ViewName]
   <([ColumnName],...)>
   [SELECT 语句]
```

示例：

```
INSERT INTO AUTHOR_1
 SELECT AUTHOR_ID,LASTNAME,FIRSTNAME,EMAIL,CITY,COUNTRY
   FROM AUTHOR
  WHERE FIRSTNAME = 'RAUL'
```


这是 INSERT SQL 语句的另一种语法，其中 SELECT 语句替代了 VALUES 语句，这个子查询的结果被赋给适当的列。

这是一个例子。 AUTHOR_1 表被填充以对表 AUTHOR 执行 SELECT 语句后所返回的值。

INSERT 语句（续）

示例 – 简单的 INSERT

```
INSERT INTO AUTHOR VALUES
  ('A1','CHONG','RAUL','RFC@IBM.COM.','TORONTO','CA')
```

示例 – 插入多行

```
INSERT INTO AUTHOR
 (AUTHOR_ID, LASTNAME, FIRSTNAME, EMAIL, CITY, COUNTRY)
VALUES
('A1','CHONG','RAUL','RFC@IBM.COM.','TORONTO','CA'),
('A2','AHUJA','RAV','RA@IBM.COM.','TORONTO','CA')
```


这是一些 INSERT 语句的例子。

通过在 values 子句中指定以逗号分隔的各行，可以一次插入多行。

UPDATE 语句

修改表或视图中的数据

语法：

```
UPDATE [TableName | ViewName]
   SET [[ColumnName]=[Value] | NULL | DEFAULT, ...]
<WHERE [Condition]>
```

类型

- 选择性更新 (Searched update)
  - 更新表中的一行或多行
- 被定位的更新 (Positioned update)
  - 总是嵌入到一个程序中 – 使用游标

存在于表中的特定值可以通过执行 UPDATE SQL 语句来修改。

UPDATE 语句的语法是：

UPDATE [TableName | ViewName]

SET [[ColumnName]=[Value]

WHERE [Condition],

其中， update 是关键字， Column 指出了含有待修改数据值一个或多个列的名称。

Value 指出了用于替换所指定列中现有值的一个或多个数据值。

与 INSERT 语句相同的是， update 语句也可以使用查询或子查询的结果。

有两类 UPDATE 语句：

1) 被检索的更新：用于更新表中的一行或多行。

2) 被定位的更新：这些更新总是嵌入到一个程序中，而且需要在待更新的行中创建、打开及定位一个游标。在本演示中，我们主要讨论“被检索的更新”。

UPDATE 语句（续）

示例

```
UPDATE AUTHOR
   SET  LASTNAME  = 'KATTA'
        FIRSTNAME = 'LAKSHMI'
   WHERE AUTHOR_ID = 'A2'
```

或者

```
UPDATE AUTHOR
  SET  (LASTNAME,FIRSTNAME) = ('KATTA','LAKSHMI')
  WHERE  AUTHOR_ID = 'A2'
```

警告：如果没有 **WHERE** 子句，所有的行都会被更新！

在本例中可以看到，对于 AUTHOR_ID 为 A2 的行， lastname 和 firstname 被更新为 'KATTA' 和 'LAKSHMI' 。

这是更新值的另一种方法。

如果未规定 WHERE 从句，那么，所有的行都将被更新为 KATTA 和 LAKSHMI 作为姓和名。

UPDATE 语句（续）

删除值的示例

```
UPDATE AUTHOR
   SET COUNTRY = NULL
```


update 语句还可以用来从列中删除值。

这是通过把列的当前值修改为 NULL 来实现的。

请注意，这仅对于可以存在空值的列才有效。在本例中，所有行中的 COUNTRY 都通过被设置为 NULL 而删除。

DELETE 语句

用于从表中或视图中删除行

语法

```
DELETE FROM [TABLEName | ViewName]
    <WHERE [Condition]>
```

类型

- 选择性删除 (Searched Delete)
  - 用于删除表中的一行或多行
- 被定位的删除 (Positioned Delete)
  - 总是嵌入到一个程序中 – 使用游标

DELETE 语句用于一个或多个整行数据都需要删除的场合。

DELETE 语句也有两种类型：

被检索的删除和被定位的删除。

这里是 delete 语句的语法：

DELETE FROM [TABLEname | ViewName]

后跟 WHERE 条件。

DELETE 语句（续）

示例

```
DELETE FROM AUTHOR
    WHERE AUTHOR_ID IN ('A2','A3')
```

警告：

如果不使用 **WHERE** 子句，所有的行都会被删除！

这是一个简单的例子，它把 AUTHOR_ID 为 A2 和 A3 的行从表中删除了。

如果未规定 WHERE 子句，则表中所有的行都会被删除。

议题

概述

数据库对象

SQL 简介

SELECT 语句

UPDATE、INSERT 和 DELETE 语句

使用 **JOIN**

Join （连接）的类型

INNER JOIN （内连接）

LEFT OUTER JOIN （左外连接）（或者简称 **LEFT JOIN** （左连接））

RIGHT OUTER JOIN （右外连接）（或者简称 **RIGHT JOIN** （右连接））

FULL OUTER JOIN （全部外连接）（或者简称 **FULL JOIN** （全连接））

CROSS JOIN （交叉连接）

JOIN 运算符能够让你把两个表中的数据结合到一起。当涉及到操纵数据时， JOIN 运算符非常重要。其原因非常易于理解。如果你观察关系模型，你会注意到，信息被“分裂”到不同的表中。这是规范化的自然结果，而规范化是与关系建模相关的一个非常重要的概念。因此，当你试图把信息组合到一起来创建报告时，你通常需要从多个表中收集数据。这时，你就需要使用 JOIN 运算符。

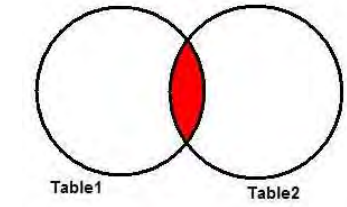
JOIN 运算符是标准 SQL 的一部分，就是说：它在 DB2 中或任何其他 RDMBS 中的工作方法基本相同。当你使用 JOIN 运算符时，你总是把两个表中的数据结合到一起。你需要做的第一件事情是确定两张表之间的关系。请记住，我们是在讨论关系数据库。换言之，你需要识别出每个表中应当用作这些表之间“链接”的一列或多列。

其次，确定你打算使用的连接类型。我们很快将详细讨论本幻灯片中所列出的各个不同类型的 JOIN 。

SQL 为你提供了不同类型的连接。它们完全取决于你打算实现什么目的。例如，你可能抽取对应于两个表交集的数据集。或者，你可能打算选择一个更大的数据集。你也可能打算对两个表中全部数据的组合进行查询。

INNER JOIN

结果是两个表的交集




```
SELECT *
FROM TABLE T1 INNER JOIN T2
ON T1.ID = T2.ID
```

T1

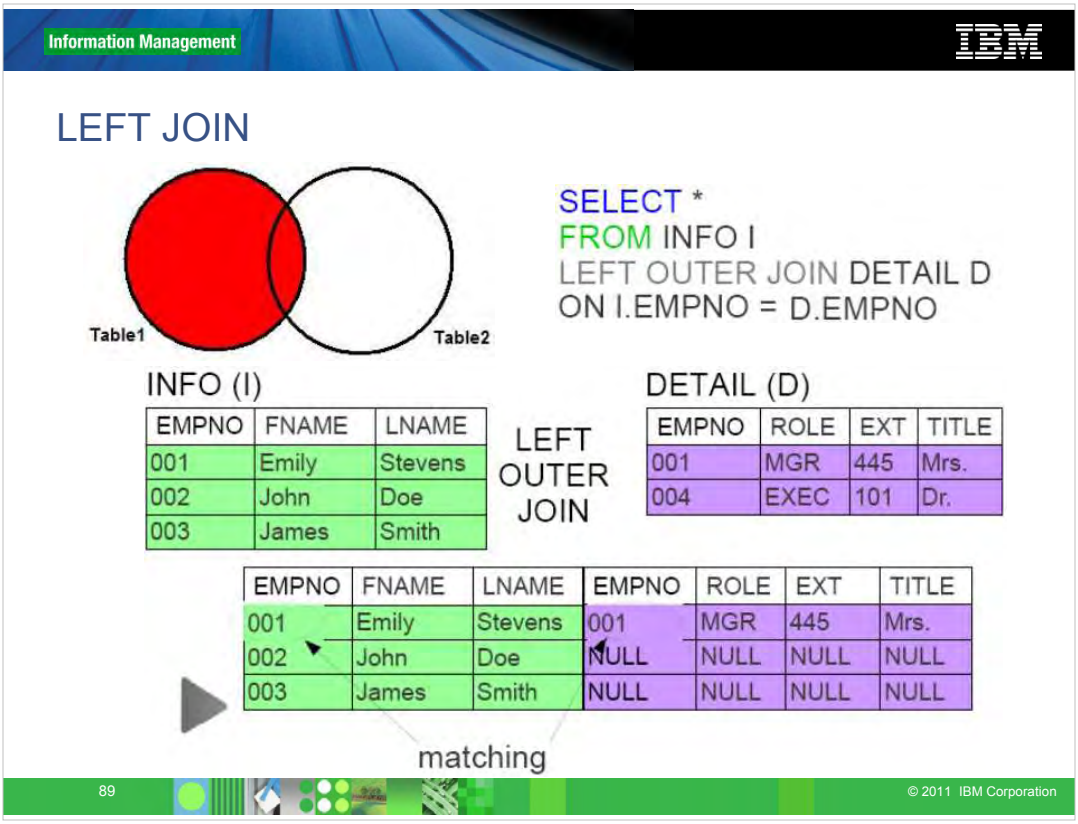
ID	C2
A	11
B	22

T2

ID	C4
A	33
Z	44

T1.ID	C2	T2.ID	C4
A	11	A	33

最常见的连接类型是 inner join （内连接），它返回代表两个表交集的数据集。
看一下表示这种连接的维恩图 (venn diagram)，以加深连接。
然后你可以查看你需要使用的语法来完成此连接。



我们现在看看 left join （左连接）运算符。左连接运算符的维恩图是这样的。你留下左侧表中的全部内容，以及第二个表中仅与另一个表相匹配的信息。当我们说，左侧表时，我们是指 join 运算符左侧的表，在本例中，就是 borrower 表。如你所见，此语法与前面的很相似。我们只是改变了运算符本身。

请注意，当我们运行 left join 时，我们将从两个表中选取列，但右侧表中的某些记录显示为 NULL 值。这发生在右侧表中不存在相应的键值时。

当你使用 right join （右连接）运算符时，你会看到恰恰相反的效果。你此时可能也没有获得值，但此时是左侧表的记录受到影响。

为了“处理”此连接的结果集，遵循以下步骤可以使问题简单一些：

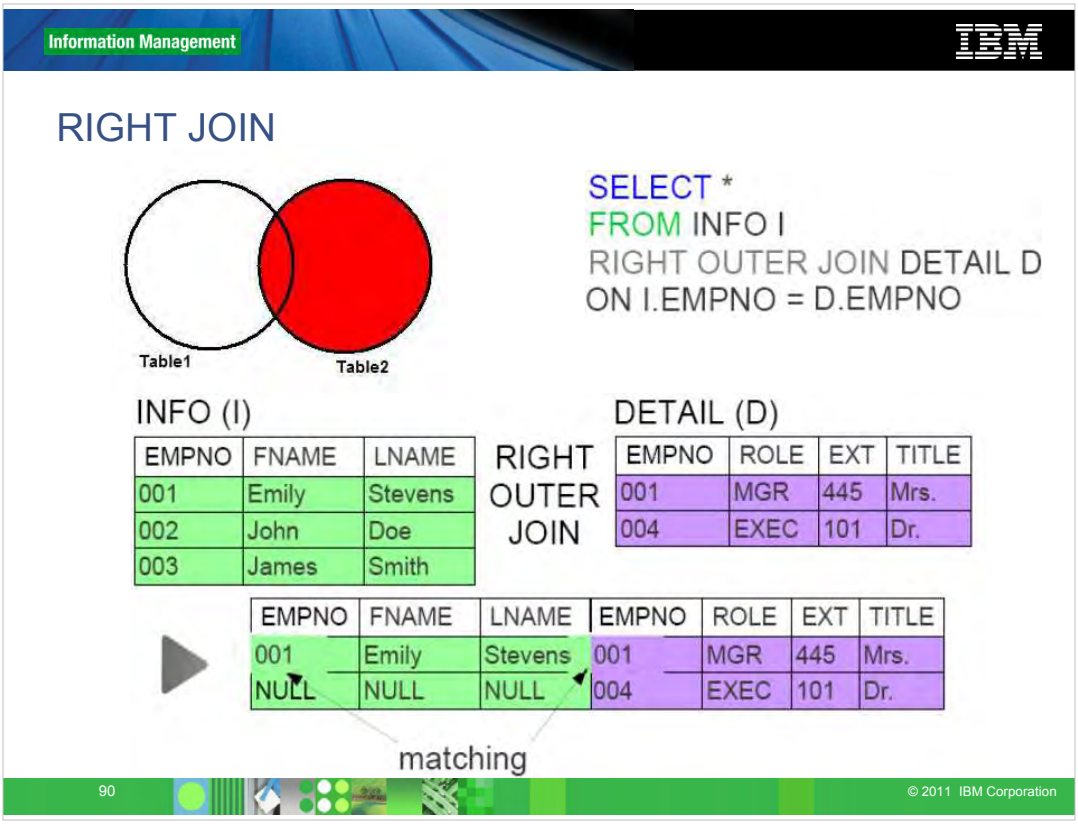
1) 获得整个左侧表；

2) 从右侧表中获得一行，然后根据 JOIN 中使用的列（在本例中即 EMPNO ）查看是否存在匹配的行；

3) 如果有匹配，就把它放在左侧表中相应列的旁边；

4) 如果无匹配，则置入 NULL ，并以 null 填写所有的行；

5) 如果问题是你最终会得到多少行，你知道，你将从左侧表中选取所有的行，因此，如果没有 WHERE 子句，你至少会得到与左侧表相同的行数。



RIGHT JOIN 是 LEFT JOIN 镜像：它选取对应于两表交集的数据集，另加 JOIN 运算符右侧的表中的剩余记录。我们再看看相应的维恩图，以及 SQL 语法。如你所见，语法完全相同。你只需要把运算符从“ left join” 改为“ right join” 就可以了。

为了“处理”此连接的结果集，遵循以下步骤可以使问题简单一些：

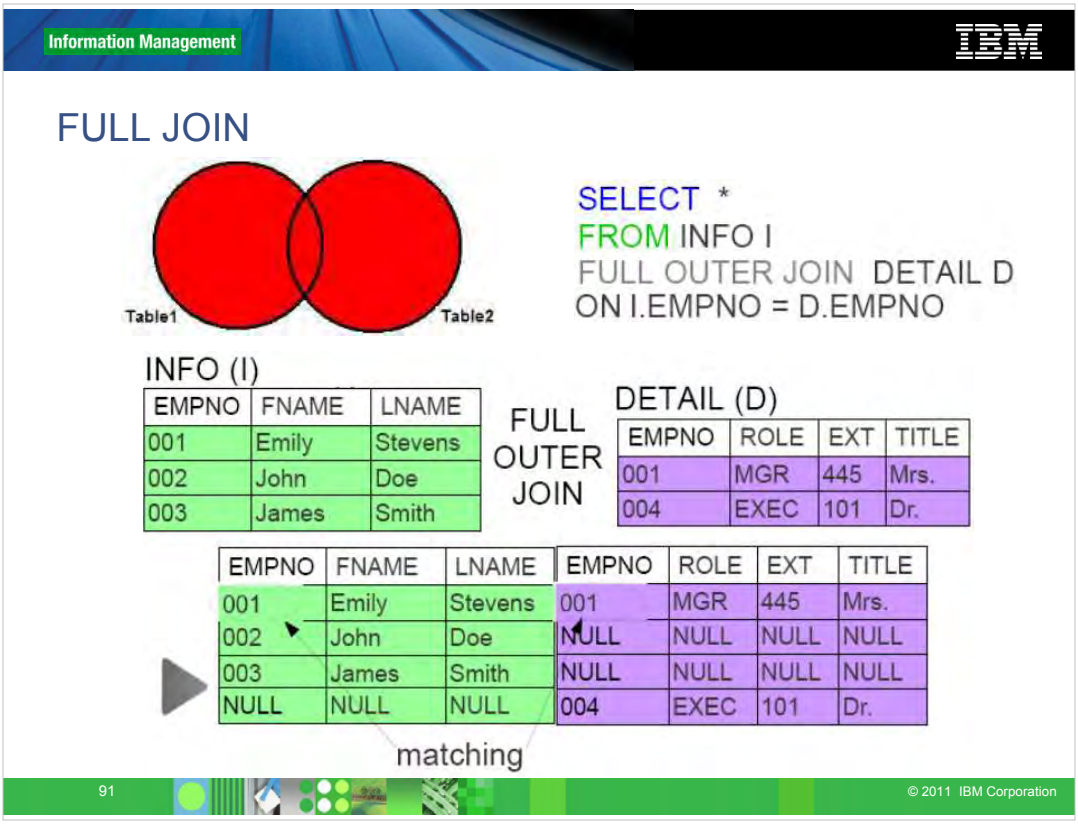
1) 获得整个右侧表；

2) 从左侧表中获得一行，然后根据 JOIN 中使用的列（在本例中即 EMPNO ）查看是否存在匹配的行；

3) 如果有匹配，就把它放在右侧表中相应列的旁边（放在左侧）；

4) 如果无匹配，则置入 NULL ，并以 null 填写所有的行；

5) 如果问题是你最终会得到多少行，你知道，你将从右侧表中选取所有的行，因此，如果没有 WHERE 子句，你至少会得到与右侧表相同的行数。



FULL JOIN （全连接）从两个表中获得所有的数据。它更类似于抽取两个数据库的联合。 NULL 值可能出现在任何一侧表的列中，也可能出现在两个表中，这取决于两个表中键值的对应情况。

语法几乎完全相同。我们只需要改变运算符即可。这里是把运算符从 RIGHT JOIN 改为 FULL JOIN 。

为了“处理”此连接的结果集，遵循以下步骤可以使问题简单一些：

1) 选取行数多的那张表，本例中为左侧表（这些步骤均以此假设为基础）；

2) 从右侧表中获得一行，然后根据 JOIN 中使用的列（在本例中即 EMPNO ）查看是否存在匹配的行；

3) 如果有匹配，就把它放在左侧表中相应列的旁边（放在右侧）；

4) 如果无匹配，则创建一个新行，并在改行的其余列中填写 NULL ；

5) 如果问题是你最终会得到多少行，你知道，你将从左侧表中选取所有的行，并从右侧表中选取所有的行，然后再减去具有相同 EMPNO 的行数。在本例中，左侧表有 3 行，右侧表有 2 行。有一行具有相同的 EMPNO ，因此，最终结果是 3+2-1= 4 行。

最后，我们来看看 CROSS JOIN （交叉连接）运算符。 CROSS JOIN 与前面的例子有很大的不同。它计算两个表的笛卡尔乘积，即：把第一个表中的每条记录与第二个表中的每条记录相结合。

这一次，语法稍有不同。对于 CROSS JOIN ，没有结合键。 CROSS JOIN 极少被使用，但当你需要生成测试数据时，它非常有用。

为了“处理”此连接的结果集，遵循以下步骤可以使问题简单一些：

1) 获取左侧表；

2) 根本没有 JOIN 条件，因此，对于左侧表中的每一行，置入右侧表中的每一行。

例如：

001 Emily Stevens 001 MGR 445 Mrs

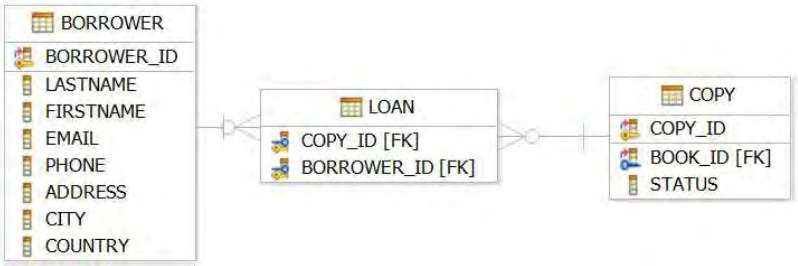
001 Emily Stevens 004 EXEC 101 Dr.

对左侧表的每一行重复该过程。

3) 如果问题是你最终会得到多少行，只需把左侧表的行数与右侧表的行数相乘就可以了。
在本例中： 3 x 2 = 6

连接多个表

你每次只能合并 2 张表。




```
SELECT B.LASTNAME, L.COPY_ID, C.STATUS
FROM BORROWER B
    INNER JOIN LOAN L ON B.BORROWER_ID = L.BORROWER_ID
    INNER JOIN COPY C ON L.COPY_ID = C.COPY_ID
```


截至目前，你只看到如何结合两张表。但是，如果你需要结合三张或者更多张不同表中的数据，应当怎么办？

只需要在 SQL 语句中添加新的 join 即可。实际上，你可以使用不同类型的 join 把这些表结合到一起。

我们来看看这份简单的图示。

我们这里有 3 张表： BORROWER 、 LOAN 和 COPY

如果我需要编写一个从它们之中获取信息的 SQL 语句，

我只需要进行简单的连接，把它们两两结合到一起。

首先，我连接来自 BORROWER 和 LOAN 的信息，

其次，我连接来自 LOAN 和 COPY 的信息。非常简单。

如果你对本课程所讨论的内容有什么技术方面的问题的话，可以利用 db2university.com 课程 AA001EN 中的论坛。 同学、教员以及 IBM 人员可以帮助你！

Information Management

IBM

数据并发性和锁定

IBM 信息管理云计算能力中心
IBM （加拿大）研究院

议题

事务

并发性和锁定

锁等待

死锁

支持性阅读材料和视频

阅读材料

DB2 Express-C 入门电子书

第 13 章：并发性和锁定

视频

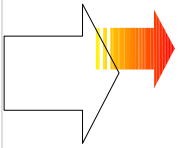
db2university.com 课程 AA001EN

第 8 课：Data 并发性和锁定

如果你需要关于本主题的详细内容，请参阅本支持材料：

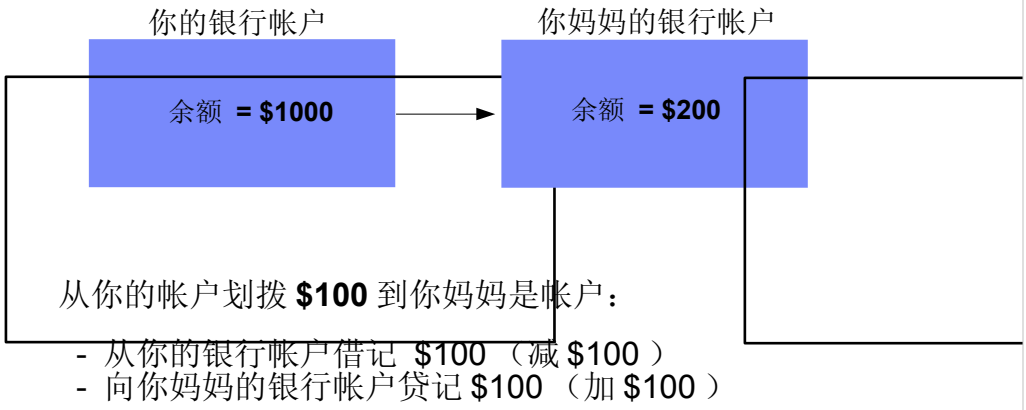
- “DB2 Express-C 入门（第三版）”，第 13 章：并发性和锁定；以及
- db2university.com 课程 AA001EN DB2 学术培训，第 8 课

议题



- 事务
- 并发性和锁定
- 锁等待
- 死锁

什么是事务（事务）?



这个简单的例子可以更好地解释什么是事务。假设你可以访问银行中的两个帐户；一个是你的银行帐户，另一个是你妈妈的银行帐户。你的银行帐户有余额 1000 美元，你妈妈的银行帐户有余额 200 美元。现在，假如你打算从你的帐户向你妈妈的帐户转账 100 美元。幕后的操作是，首先从你的帐户借记 100 美元，然后再向你妈妈的银行帐户贷记 100 美元。但是，如果在从你的帐户中借记 100 美元之后，突然停电了，操作无法完成，这时会发生什么情况？你会丢失 100 美元！正如你所说的那样，在任何现实应用中都不会允许这种事情发生。为解决此问题，借记和贷记这两项操作应当作为一个单元来处理；也就是一项事务。

什么是事务？（续）

一个或多个 SQL 语句，它们在一起被视为一个单元

也被称为工作单元 (UOW)

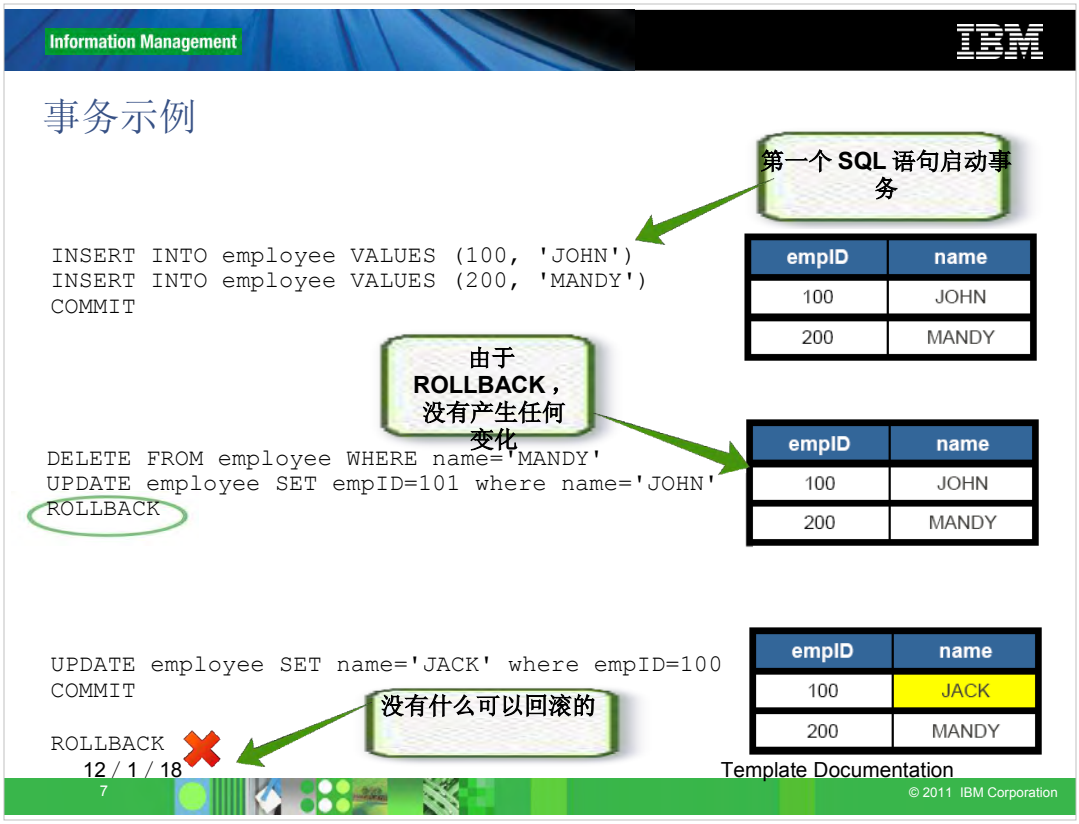
一次事务 A 起始于任何 SQL 语句，结束于 COMMIT （落实）或 ROLLBACK （回滚）

COMMIT 语句使得修改对数据库永久生效

ROLLBACK 语句逆转所做的修改

COMMIT 语句和 ROLLBACK 语句释放所有的锁

所以，事务基本上就是把一个或多个 SQL 语句当做一个单元。如果事务中的一个语句不成功，整个事务就不成功。事务也被称为工作单元 (UOW)。如我们将在下一张幻灯片中所看到的那样，一项事务起始于任何 SQL 语句，并结束于 COMMIT 或 ROLLBACK 语句。 COMMIT 语句使得修改对数据库永久生效，而 ROLLBACK 语句则逆转所做的修改。两个语句都释放在各个行上获得的全部锁。



例如，在本幻灯片中，你可以看到多项事务。在第一项事务中，向 employee 表插入了两行。该事务以 COMMIT 语句结束，这可以保证这两行（对于 'JOHN' 和 'MANDY'）都存储到数据库中。下一项事务以 DELETE 语句起始，并以 ROLLBACK 结束。由于它以 ROLLBACK 结束，所做的任何修改都没有实际生效，该表仍然是事务启动之前的样子。然后我们有另一项事务，它把 empID=100 的行 name 值更新为 'JACK'。由于该事务以 COMMIT 结束，此改变就生效了，如幻灯片中绿色所示。最后，还有一个简短的事务，它仅仅由一句 ROLLBACK 构成。这项仅一句的事务不会带来任何效果，因为此时没有什么可以回滚的。

事务 – ACID 规则

Atomicity （原子性）

事务中的所有语句被视为一个单元。

如果事务成功完成，所有的东西都被落实。

如果事务失败，截至失败点的所有东西都回滚。

Consistency （一致性）

任何事务都将把数据从一个一致的状态带入另一个状态，因此，只有有效的一致数据被存储在数据库中。

Isolation （隔离）

并发事务不能相互干扰。

Durability （持久性）

已经落实的事务应当使它们的修改在数据库中持久存在。

ACID 规则保证数据库事务以可靠的方式得到处理。

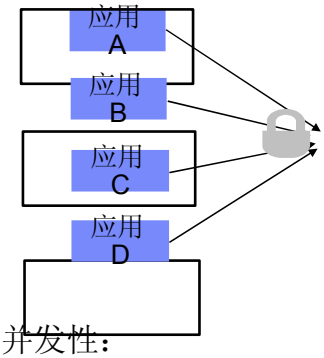
“ 隔离”规则将在后面的幻灯片中详细讨论。

一致性和持久性将在关于“备份和恢复”的演示中详细讨论。

议题

- 事务
- 并发性和锁定
- 锁等待
- 死锁

并发性和锁定



ID	Name	Age
3	Peter	33
5	John	23
22	Mary	22
35	Ann	55

并发性：

多个用户同时访问相同的资源

锁定：

用于确保数据完整性和一致性的机制

我们继续，这是并发性和锁定的概述。假如有多个应用；应用 A 、 B 、 C 和 D ，它们都试图访问同一个表中的同一列。假如没有并发控制的话，你或许会看到，所有这些应用可能会同时对这一行进行更新。那么，最终的结果是什么呢？不得而知。并发性总体上是指多个应用可以在保持数据一致性的前提下同时对同一个数据库进行操作，或者同时对同一个数据库中相同的表进行操作。锁定是数据库管理系统用于保证数据完整性和一致性的途经。因此，在本例中，如果应用 B 是第一个对第 2 行进行更新的应用，它将立即获得该行的锁，而其他应用则必须等待，直至该锁被释放。

锁定

根据需要自动获得锁，以便根据“隔离级别”支持事务

COMMIT 和 ROLLBACK 语句释放所有的锁

两个基本类型的锁：

- 共享锁（ S 锁） – 当一个应用希望读取某一行并阻止其他应用更新该行时获得的锁
- 排他锁（ X 锁） – 当一个应用更新、插入或删除某行时的锁

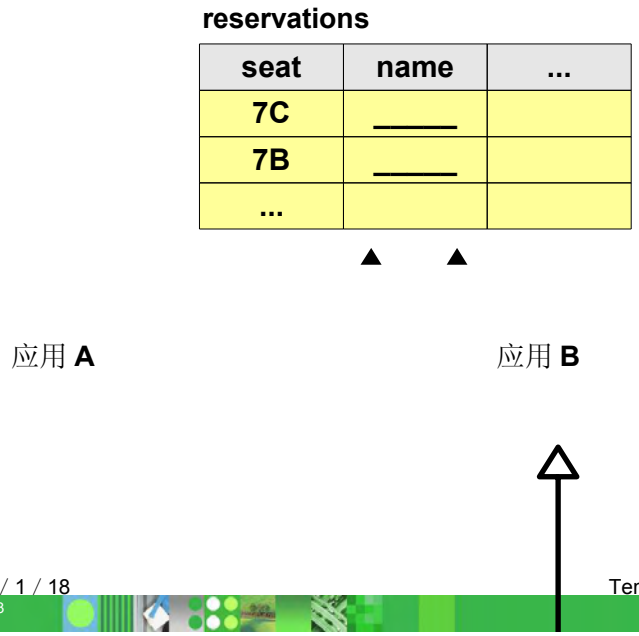
锁通常可以根据你设定的隔离级别自动为你获得。隔离级别就好像是关于如何获得锁的“策略”，这将在后面详细讨论。如前所述， COMMIT 或 ROLLBACK 语句将释放行上的锁。 DB2 中有多种类型的锁。为简单起见，我们仅介绍主要的两类：当你进行“读”操作（例如使用 SELECT 语句时），将获得用于“共享”的 'S' 锁。当你进行 DELETE 、 UPDATE 或者 INSERT 操作时，将获得“排他性”的 'X' 锁。

如果没有并发控制，将会出现的问题

- 丢失更新
- 未落实的读 (Uncommitted Read)
- 不可重复读 (Non-Repeatable Read)
- 幻影读 (Phantom read)

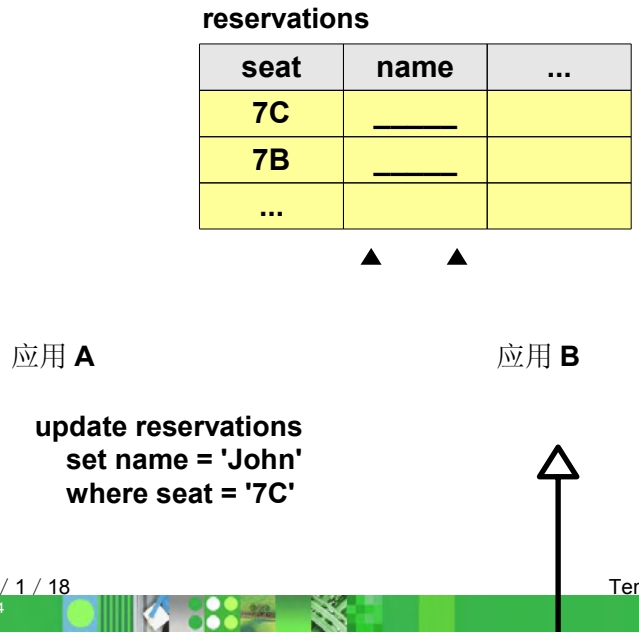
并发能够让用户同时访问一个数据库，这样，多个应用就可以同时使用同一个数据库，而数据仍然能够保持一致。如果没有并发控制，将可能发生多种问题。可能发生的四个问题是：丢失更新、未落实的读、不可重复读以及幻影读。我们下面将详细解释这些问题。

丢失更新



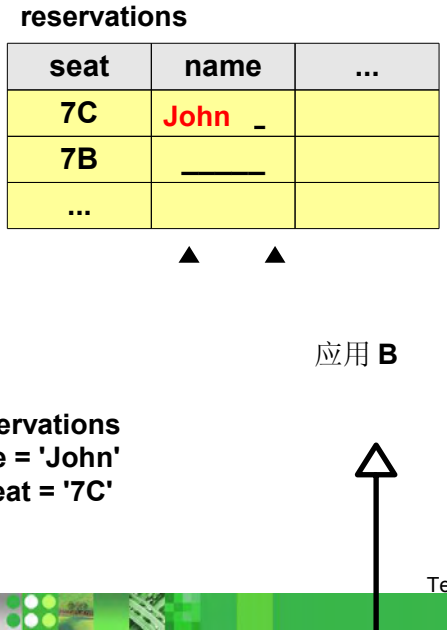
我们利用下面的例子来解释“丢失更新”问题。在本例中，假设你有一个称为“ reservations”的表，它有“ seat”列以及“ name”列，等等。

丢失更新



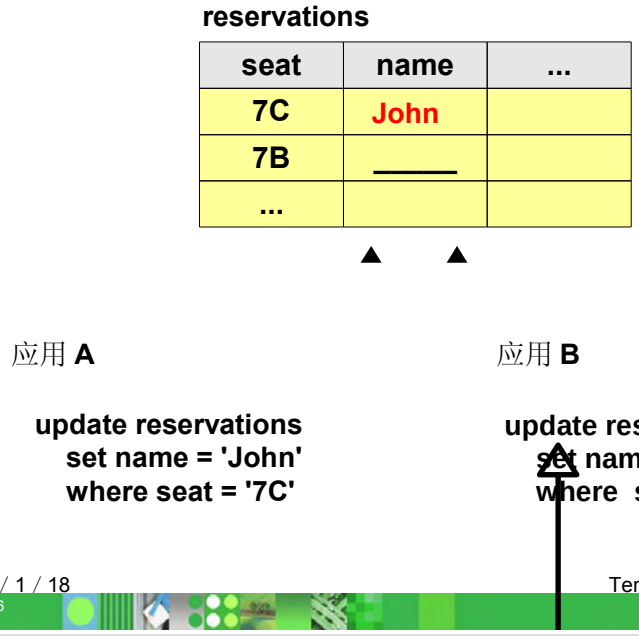
假如应用“ A”更新了表“ reservation”，于是 seat 7C 的名称

丢失更新



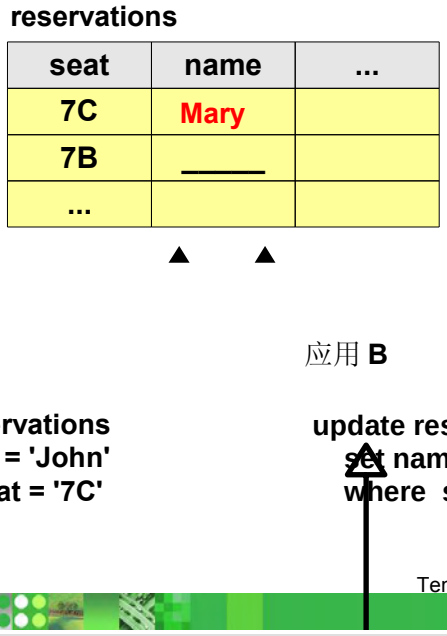
修改为“ John”。

丢失更新



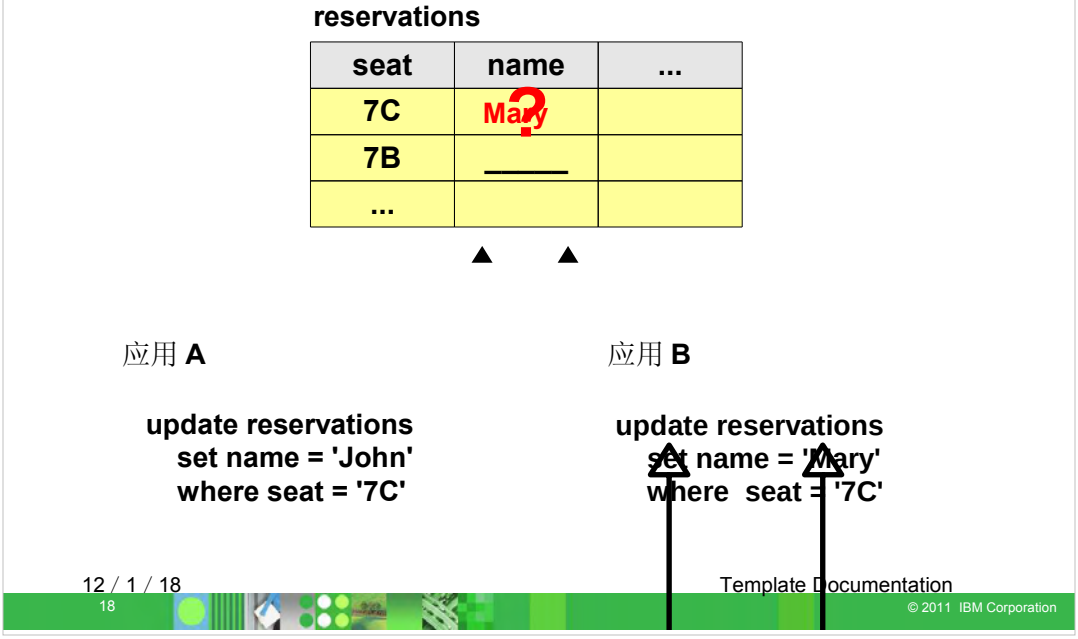
现在假设有另一个应用“ B” 同时也试图做同样的事情，但是

丢失更新



使用 “ Mary” 的名字。

丢失更新



假设应用 B 在比应用 A 稍微晚一些的时候发出 update 语句，那么， name 的最后值就是“ Mary”，这意味着，应用 A 的第一次更新是一次“丢失的更新”。

如果我们再次重复该过程，但这一次是应用 B 先进行更新操作，然后应用 A 进行操作，在这种情况下， name 的最后值就是“ John”，而来自应用 B 的第一次更新将丢失。

因此，大致而言，如果没有某种形式的控制，我们就会丢失第一个应用的更新。

未落实的读（也称为“脏读”）

reservations

seat	name	...
7C	_____	
7B	_____	
...		

应用 **A**

应用 **B**

为了解释第二个问题 --“未落实的读”，我们继续使用前面的例子。

未落实的读（也称为“脏读”）

reservations

seat	name	...
7C	_____	
7B	_____	
...		

应用 **A**

应用 **B**

```
update reservations
   set name = 'John'
   where seat = '7C'
```


应用 A 进行一次更新，使得 “ John” 成为对应于 seat '7C' 的名称。

未落实的读（也称为“脏读”）

reservations

seat	name	...
7C	John	
7B	_____	
...		

应用 **A**

应用 **B**

```
update reservations
   set name = 'John'
   where seat = '7C'
```

未落实的读（也称为“脏读”）

reservations

seat	name	...
7C	John	
7B	_____	
...		

应用 **A**

```
update reservations
   set name = 'John'
   where seat = '7C'
```

应用 **B**

```
Select name
   from reservations
   where seat is '7C'
```


然后，应用 B 发出 SELECT 语句，检索相同记录中的 name，

未落实的读（也称为“脏读”）

reservations

seat	name	...
7C	John	
7B	_____	
...		

应用 **A**

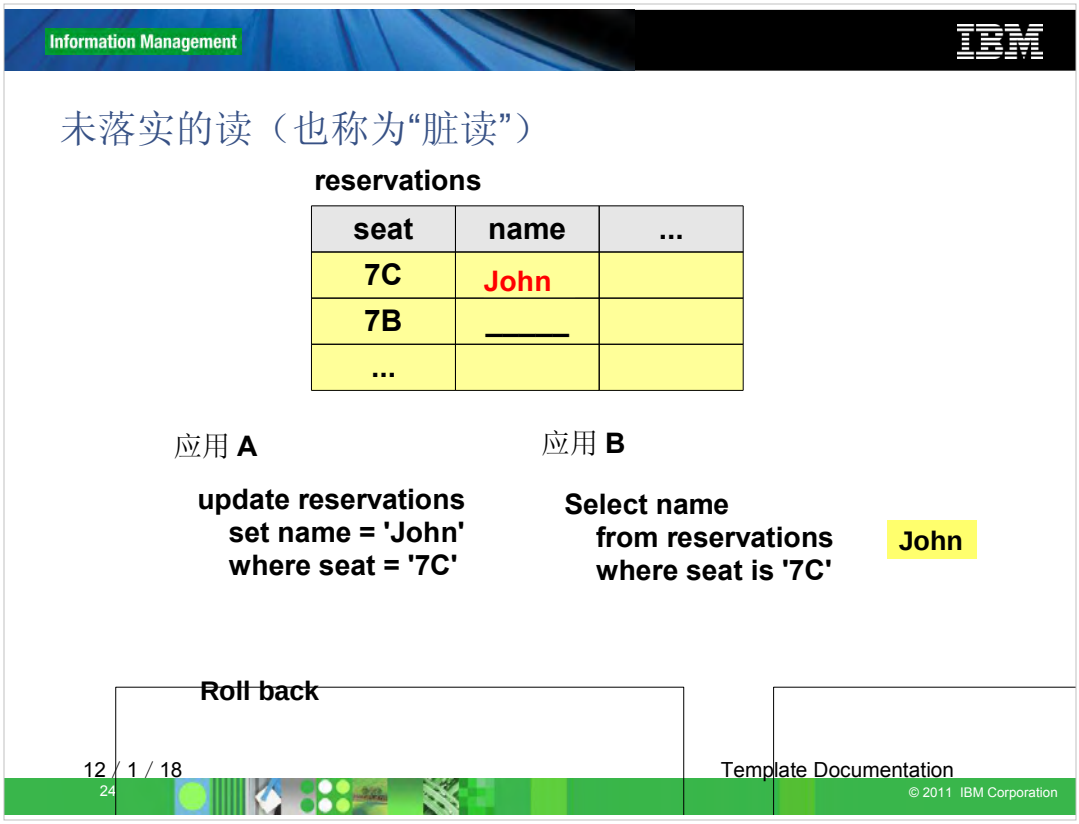
```
update reservations
   set name = 'John'
   where seat = '7C'
```

应用 **B**

```
Select name
   from reservations
   where seat is '7C'
```

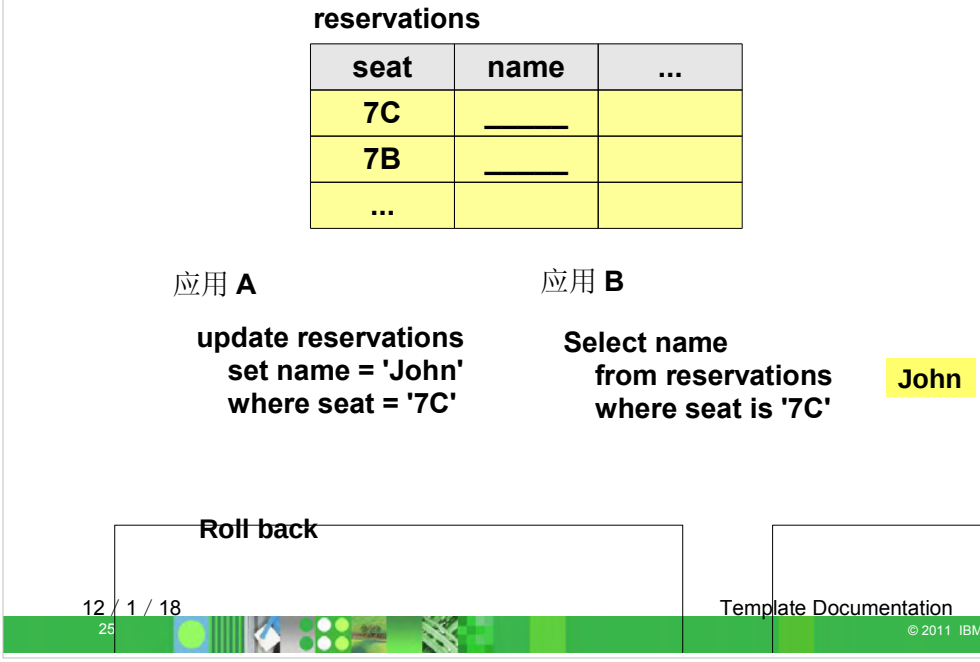
John

结果，输出为“ John”。



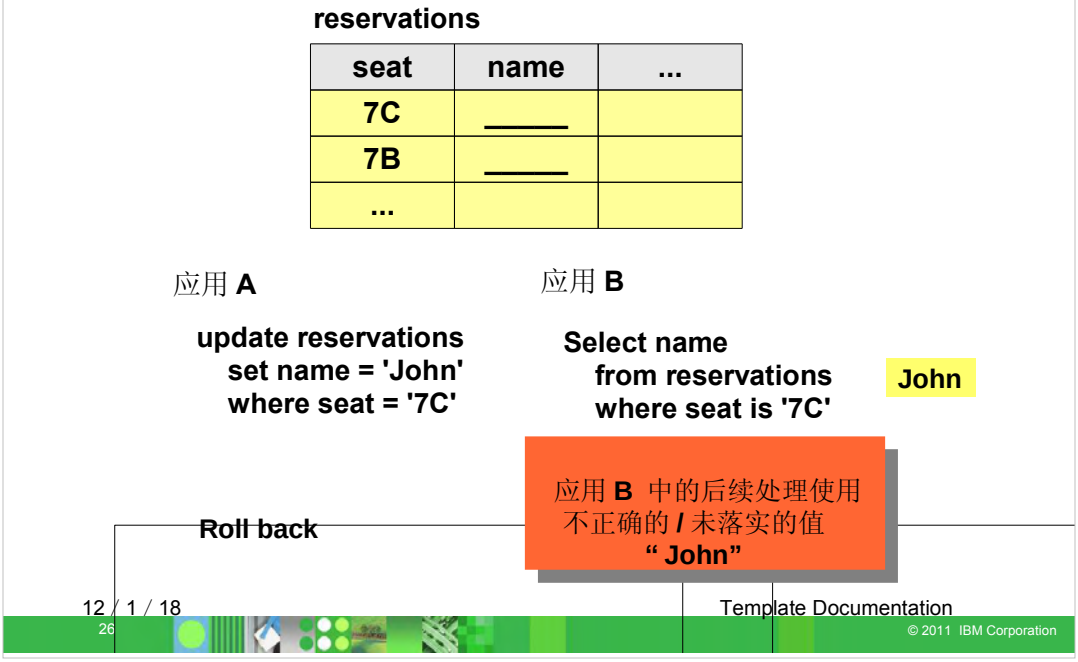
随后，应用 A 发出了 ROLLBACK 语句。

未落实的读（也称为“脏读”）



这意味着，所做的任何修改均被放弃，因此， seat '7C' 对应的 name 又变回 NULL 。

未落实的读（也称为“脏读”）



如果应用 B 继续利用“ John”进行处理，它实际上就是在使用未落实的数据，因此是不正确的。

这个问题被称为“未落实的读”。

不可重复读

reservations

seat	name	...
7C	_____	
7B	_____	
...		

应用 **A**　　应用 **B**

我们来继续看下一个问题，‘不可重复读’。
在本例中，我们有相同的 reservation 表以及相同的应用。

不可重复读

reservations

seat	name	...
7C	_____	
7B	_____	
...		

应用 **A**

应用 **B**

```
select seat
  from  reservations
  where name is NULL
```


应用 A 这次发出 SELECT 语句。

不可重复读

reservations

seat	name	...
7C	_____	
7B	_____	
...		

应用 **A**

应用 **B**

7C

7B

```
select seat
  from  reservations
  where name is NULL
```


它检索到两个 seat： '7C' 和 '7B' 。

不可重复读

reservations

seat	name	...
7C	_____	
7B	_____	
...		

应用 **A**

7C
7B

```
select seat
  from  reservations
  where name is NULL
```

应用 **B**

```
update reservations
   set name = 'John'
   where seat = '7C'
```


随后应用 B 发出 update 语句，

不可重复读

reservations

seat	name	...
7C	John	
7B	_____	
...		

应用 **A**

7C
7B

```
select seat
  from  reservations
  where name is NULL
```

应用 **B**

```
update reservations
   set name = 'John'
   where seat = '7C'
```


这使得座位 '7C' 被赋给了名字为 'John' 的乘客。

不可重复读

reservations

seat	name	...
7C	John	
7B	_____	
...		

应用 **A**

7C
7B

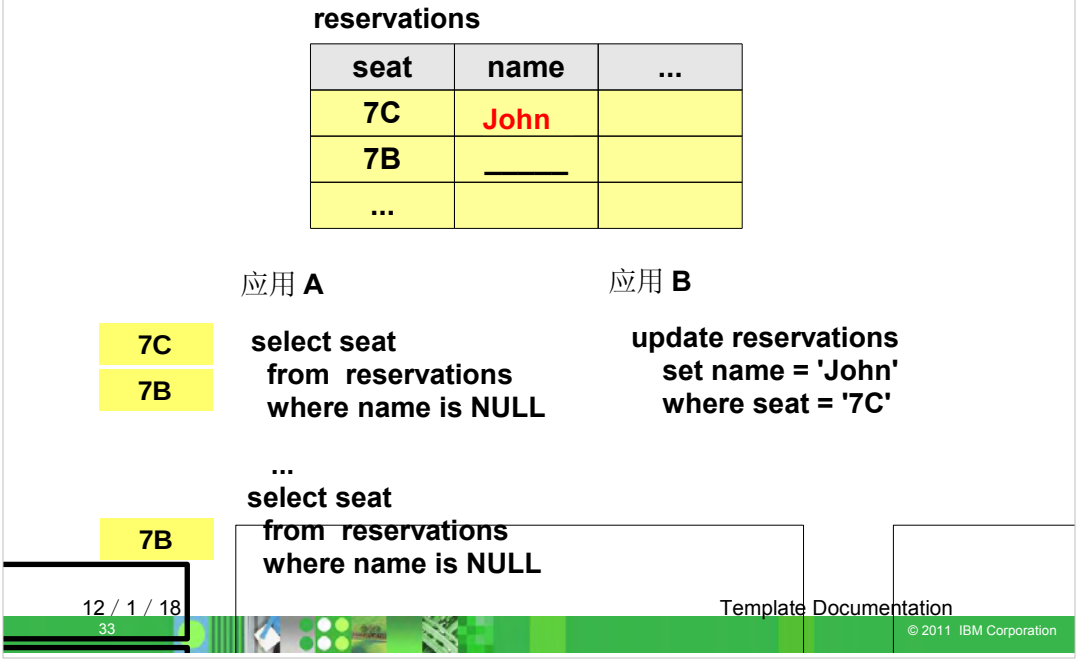
```
select seat
  from  reservations
  where name is NULL

...
select seat
  from  reservations
  where name is NULL
```

应用 **B**

```
update reservations
   set name = 'John'
   where seat = '7C'
```

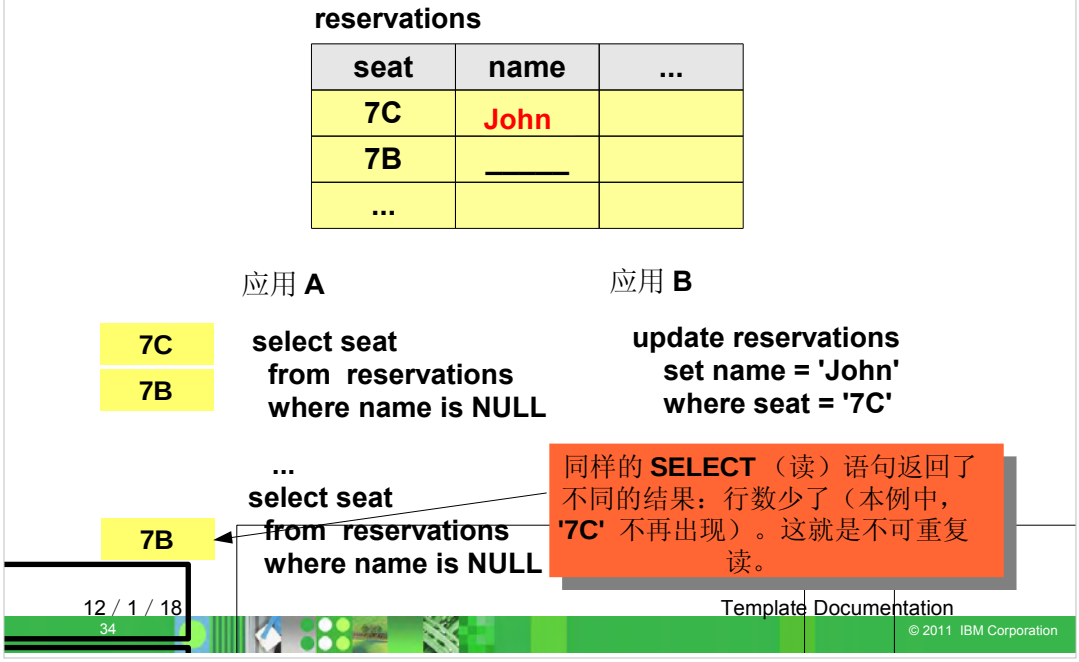

当应用 A 在同一个事务中再一次发出与第一个 SELECT 完全相同的语句时，

不可重复读



该应用现在只得到了一个 seat： '7B' 。

不可重复读



因此，它不是可重复读：在同一个事务中发出完全相同的 SELECT 语句却得到了不同的结果。

幻影读

reservations

seat	name	...
7C	Susan	
7B	_____	
...		

应用 **A**　　应用 **B**

现在，我们讨论最后一个问题，‘幻影读’。

假设 reservations 表中有一个乘客 'Susan' 被赋给了 seat '7C'。

幻影读

reservations

seat	name	...
7C	Susan	
7B	_____	
...		

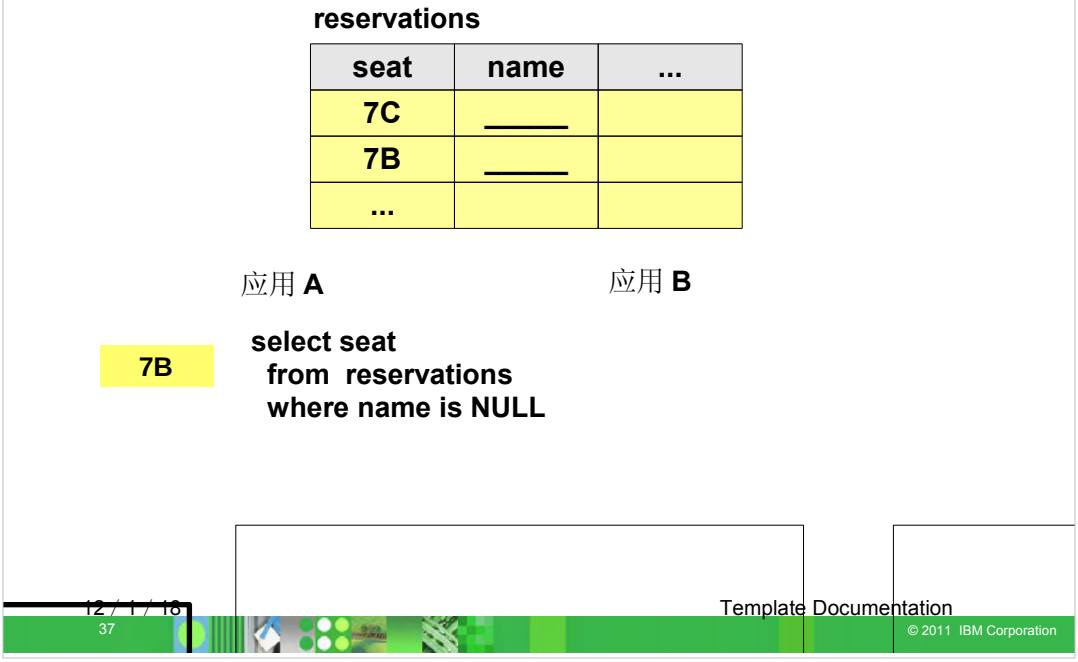
应用 **A**

应用 **B**

```
select seat
  from  reservations
  where name is NULL
```

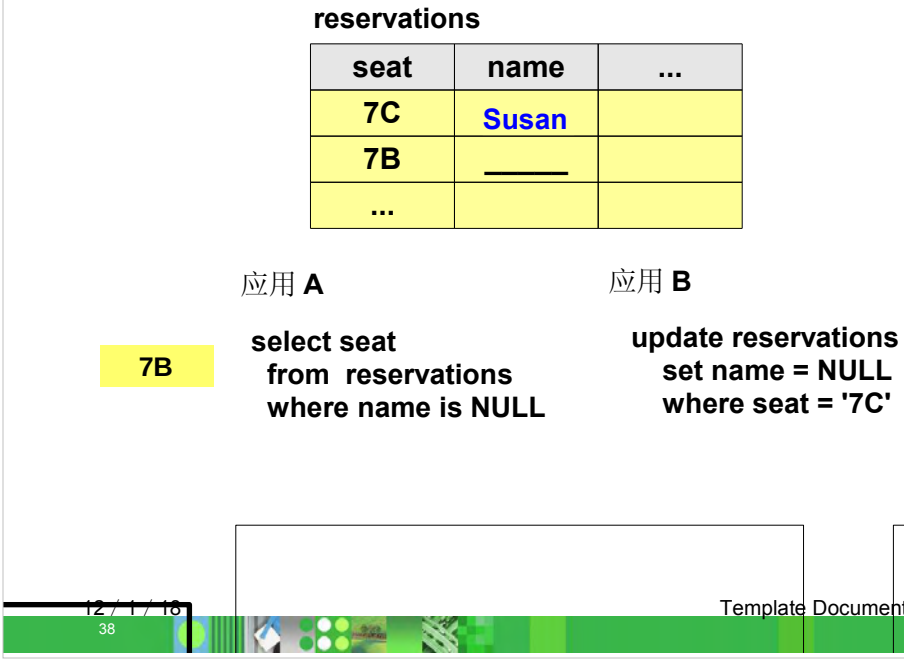

当应用 A 执行 SELECT 语句来检索尚未分配的所有 seat 时（对应的 name 为 NULL ），

幻影读



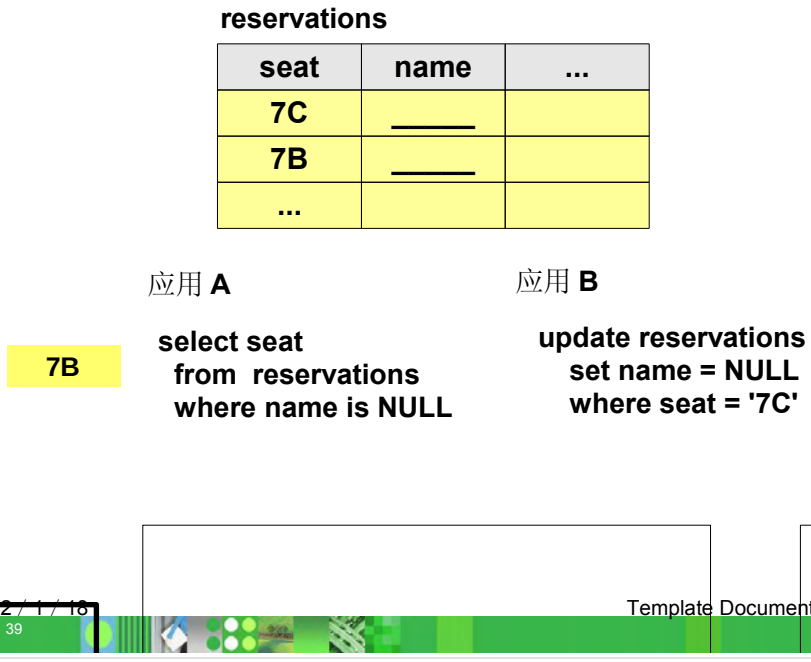
它将仅返回一个值： '7B' 。

幻影读



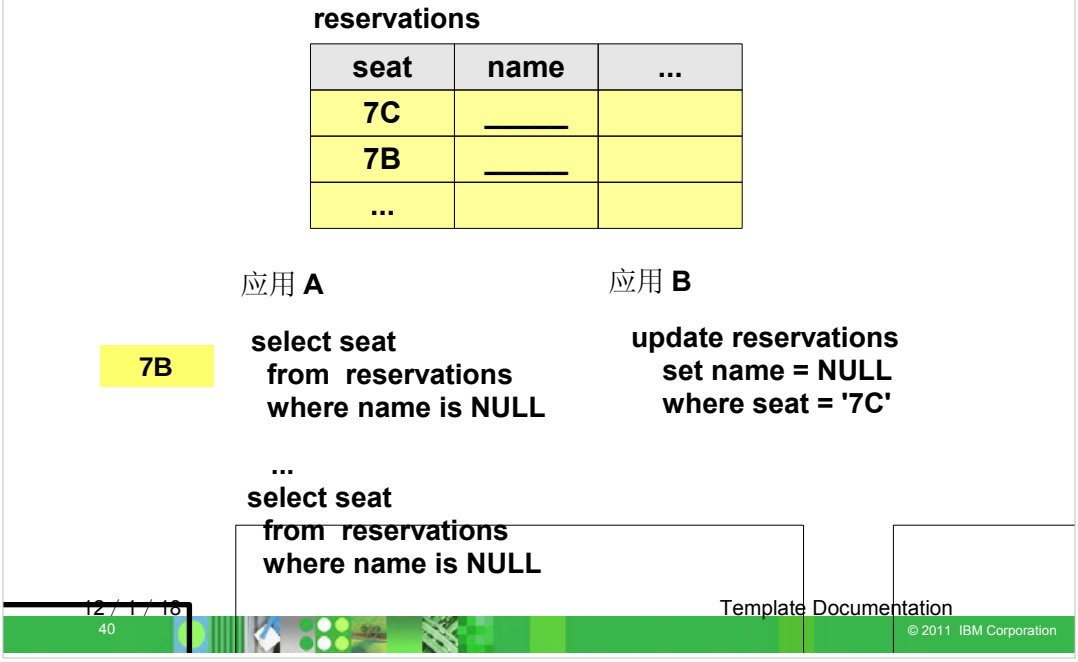
下面，应用 B 更新 seat '7C'

幻影读



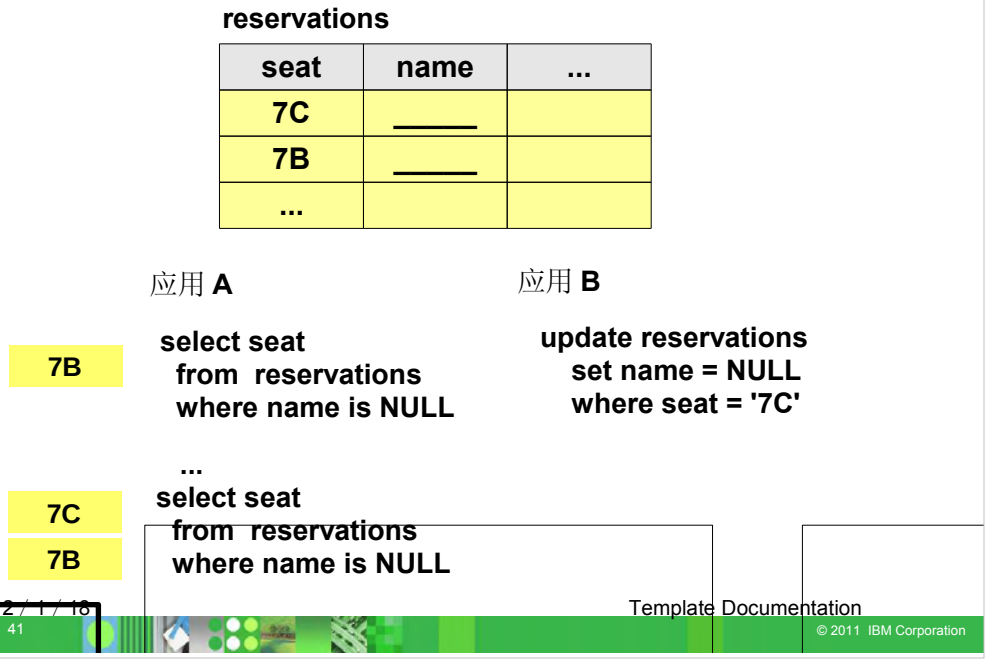
把 name 设置为 NULL 。

幻影读



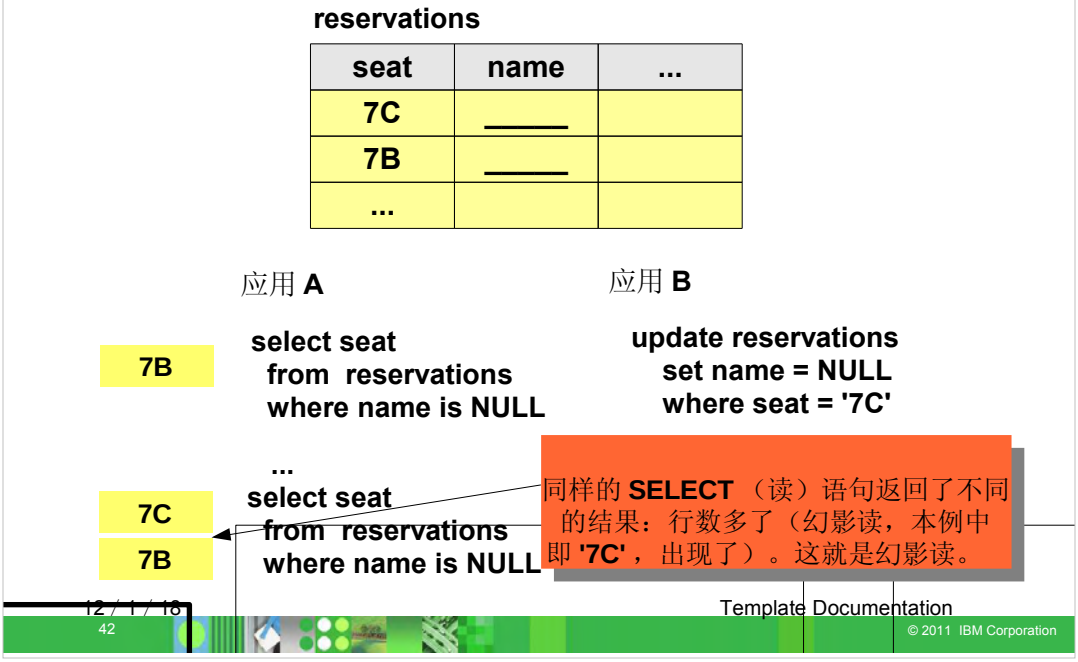
如果应用 A 在同一个事务中再次执行完全相同的 SELECT 语句，

幻影读



它现在会检索到两行： '7C' 和 '7B' 。

幻影读



因此，在这种情况下，同一个事务中的同一个 SELECT 语句多检索出一行，即“幻影”行。这就是这种情况被称为“幻影读”的原因。它与“不可重复读”的情况很相似。在“不可重复读 ”中，你得到的行数变少了；而对于“幻影读”，你得到的行数变多了。

隔离级别

控制在什么时候加锁的“策略”

DB2 为隔离数据提供了不同层次的保护

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

你可以把隔离级别想象为你如何让 DB2 处理锁的策略。

有多种不同的隔离级别，它们的名称类似于我们前面讨论的问题。我们有：

- 未落实的读，或 UR，也称为脏读，
- 游标稳定性，或 CS，
- 读稳定性，或 RS，
- 可重复读，或 RR。

设置隔离级别

可以在许多层次上规定隔离级别

会话（应用），

连接，

语句

对于语句层次，请使用 WITH {RR， RS, CS, UR} 子句：

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

对于嵌入式 **SQL**，在绑定时设置级别

对于动态 **SQL**，在运行时设置级别

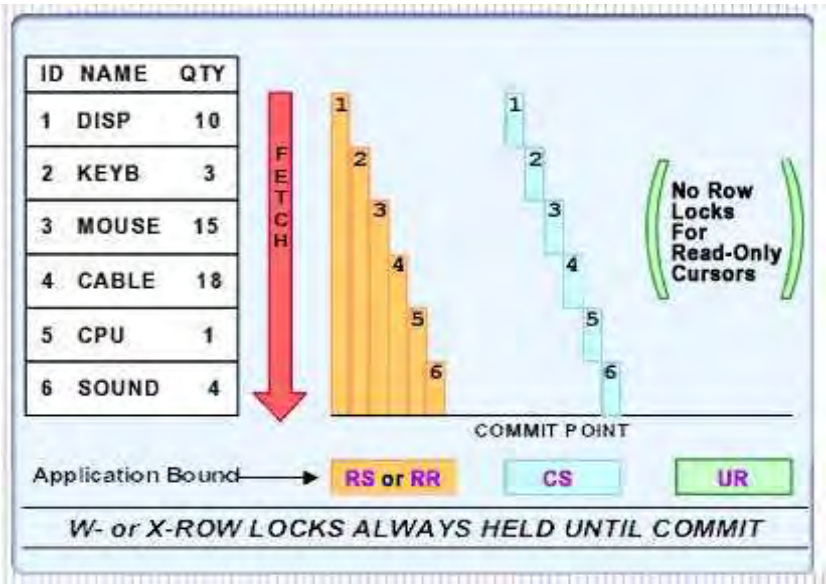
你可以在不同的层次上设置这些隔离级别：
- 你可以在应用层次或者会话层次上设置，这时的默认隔离级别是游标稳定性。
- 你可以在连接层次上设置；
- 你可以在语句层次上设置；
- 如果你在使用嵌入式 SQL，你可以在绑定时设置。
- 对于动态 SQL，你可以在运行时设置。

示例场景：

某个应用需要“大概”了解一个表中有多少行。性能具有极端重要性。需要游标稳定性隔离级，但下面一个 SQL 语句除外：

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

隔离级别比较



我们来比较一下各个隔离级别如何在保持和释放锁方面发挥作用。

该图示出一张表，其左侧有六列。我们还看看当你的应用使用不同的隔离级别从该表中获取行时会发生什么情况。

我们从 UR 开始。如你所见，在这种情况下，当你获取行时，没有获得锁。

现在看看使用 CS 隔离级别（这是默认设置）时会发生什么情况：你获取第 1 行，并获得一个锁。当你继续获取第 2 行时，第 1 行的锁被释放，但获得第 2 行的锁。当你继续获取第 3 行时，第 2 行的锁被释放，但获得第 3 行的锁。如此等等。

现在看看使用 RS 和 RR 隔离级别时会发生什么情况：你获取第 1 行，并获得一个锁。当你继续获取第 2 行时，第 1 行的锁仍然保持住，而且现在还获得了第 2 行的锁。当你继续获取第 3 行时，第 1 行和第 2 行的锁仍然保持住，而且现在还获得了第 3 行的锁。如此等等。

因此，如你所见，隔离级别 UR 能够实现最大的并发性。而 RR 或 RS 只能容许最小的并发性。但另一方面， UR 在检索结果方面只能提供最低的准确性，而 RS 或 RR 则可以提供最大的准确性。

带有当前落实 (CC) 语义的游标稳定性

带有当前落实语义的游标稳定性是默认的隔离级别

使用 *cur_commit* db cfg 参数来启用 / 禁用

避免超时和死锁

游标稳定性

Situation	Result
Reader blocks Reader	No
Reader blocks Writer	Maybe
Writer blocks Reader	Yes
Writer blocks Writer	Yes

具有当前落实的游标稳定性

Situation	Result
Reader blocks Reader	No
Reader blocks Writer	No
Writer blocks Reader	No
Writer blocks Writer	Yes

当使用 CS 时，有两种不同的行为（它们将在下一张幻灯片中详细讲解）。

更能够防止超时 / 死锁的行为是启用了“当前落实”时情形。

主要差别在于（见两个表中的第 3 行）：

如果不带有当前落实，写入方将会封锁读取方（即： UPDATE 将封锁 SELECT ），因此，基本上， SELECT 只能等待。

如果带有当前落实，写入方将不会封锁读取方。因此，即使一项应用正在对某行进行 UPDATE 操作，另一项应用也可以对该行进行 SELECT 操作。

带有当前落实 (CC) 语义的游标稳定性

游标稳定性，不带有 当前落实

应用 **A**

reservations

seat	name	...
7C	Susan	
7B	_____	
...		

应用 **B**

游标稳定性，带有当前落实 （默认行为）

应用 **A**

reservations

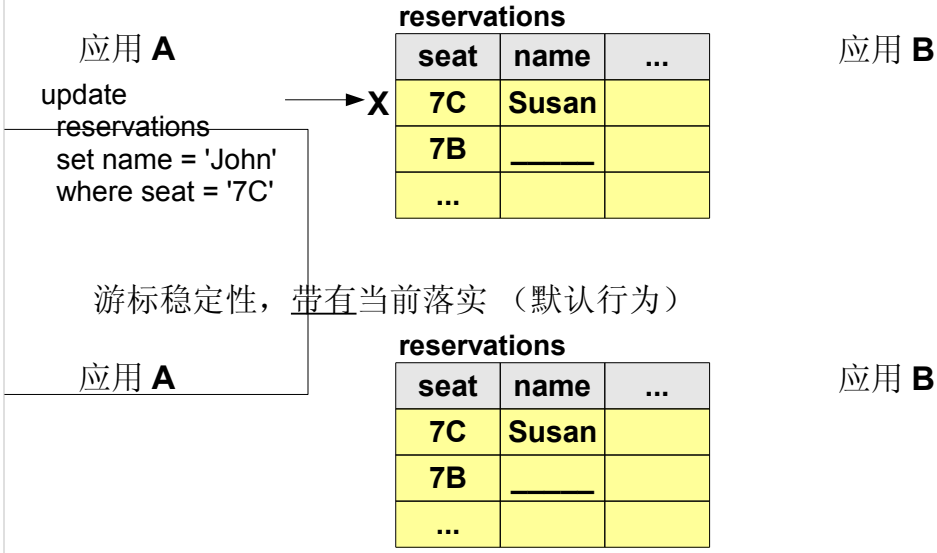
seat	name	...
7C	Susan	
7B	_____	
...		

应用 **B**

这张幻灯片示出了使用 CS 时的不同行为。上半部分，我们将解释使用不带有当前落实的 CS 时的行为。下半部分，我们将解释使用带有当前落实的 CS 时的行为。

带有当前落实 (CC) 语义的游标稳定性

游标稳定性，不带有 当前落实



游标稳定性，带有当前落实 （默认行为）

我们从上半部分开始。假设一个应用 A 进行 UPDATE 操作，它必须获得获得对该行的 X （排他）锁。

带有当前落实 (CC) 语义的游标稳定性

游标稳定性，不带有 当前落实

应用 **A**

```
update
  reservations
  set name = 'John'
  where seat = '7C'
```

→ **X**

reservations

seat	name	...
7C	John	
7B	_____	
...		

应用 **B**

游标稳定性，带有当前落实 （默认行为）

应用 **A**

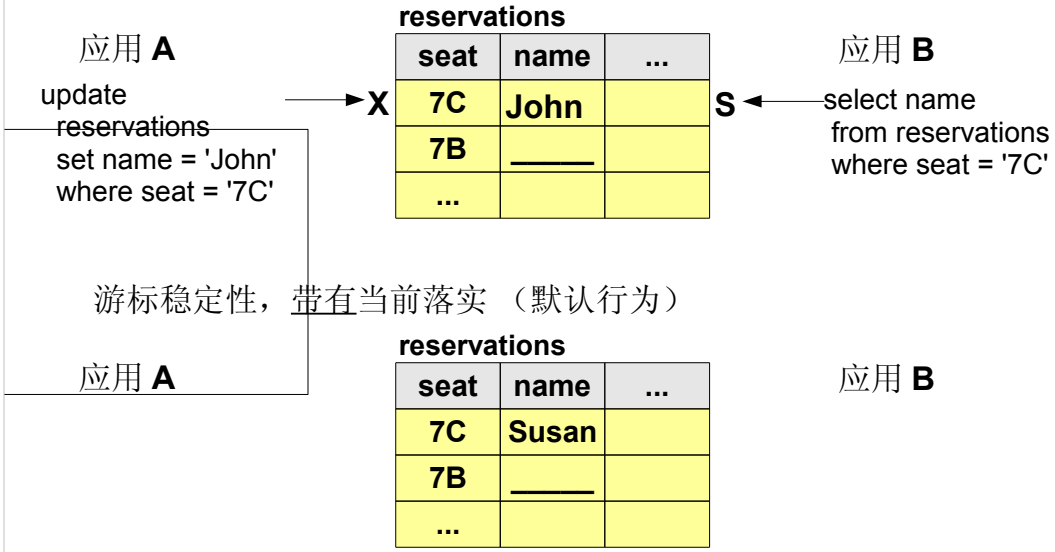
reservations

seat	name	...
7C	Susan	
7B	_____	
...		

应用 **B**

带有当前落实 (CC) 语义的游标稳定性

游标稳定性，不带有 当前落实

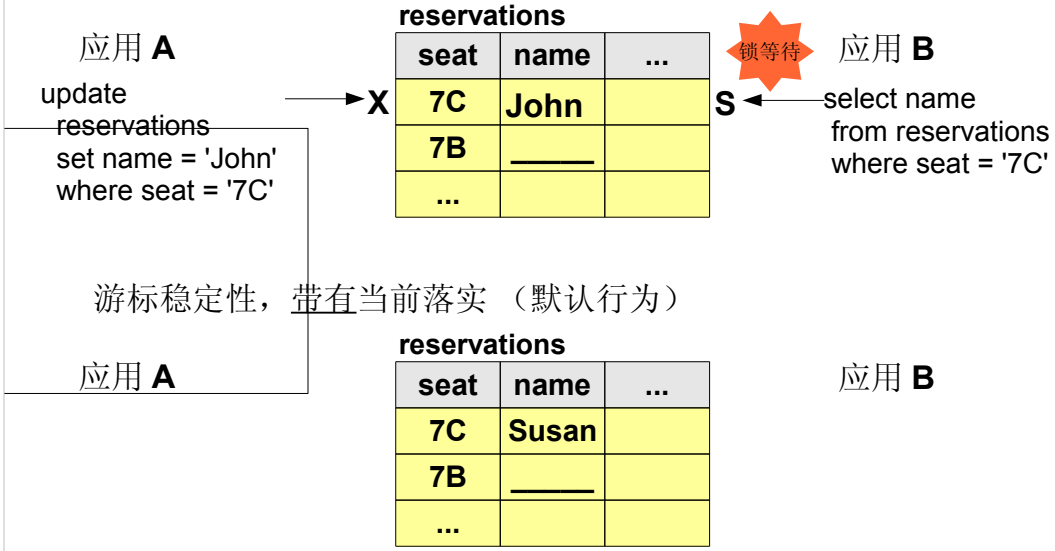


游标稳定性，带有当前落实 （默认行为）

应用 B 以不带有 CC (当前落实) 的 CS 方式运行，它也试图对该行进行 SELECT 操作。它将请求获得 'S'（共享）锁；而这与 X 锁不兼容，

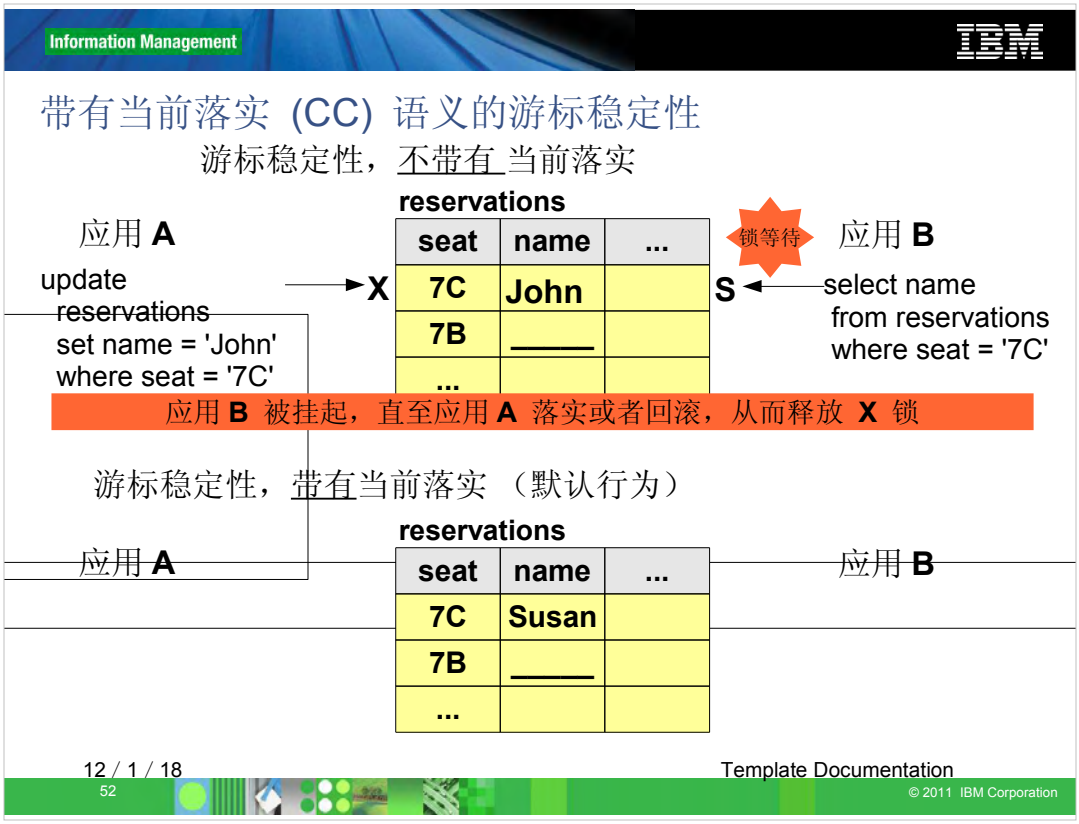
带有当前落实 (CC) 语义的游标稳定性

游标稳定性，不带有 当前落实

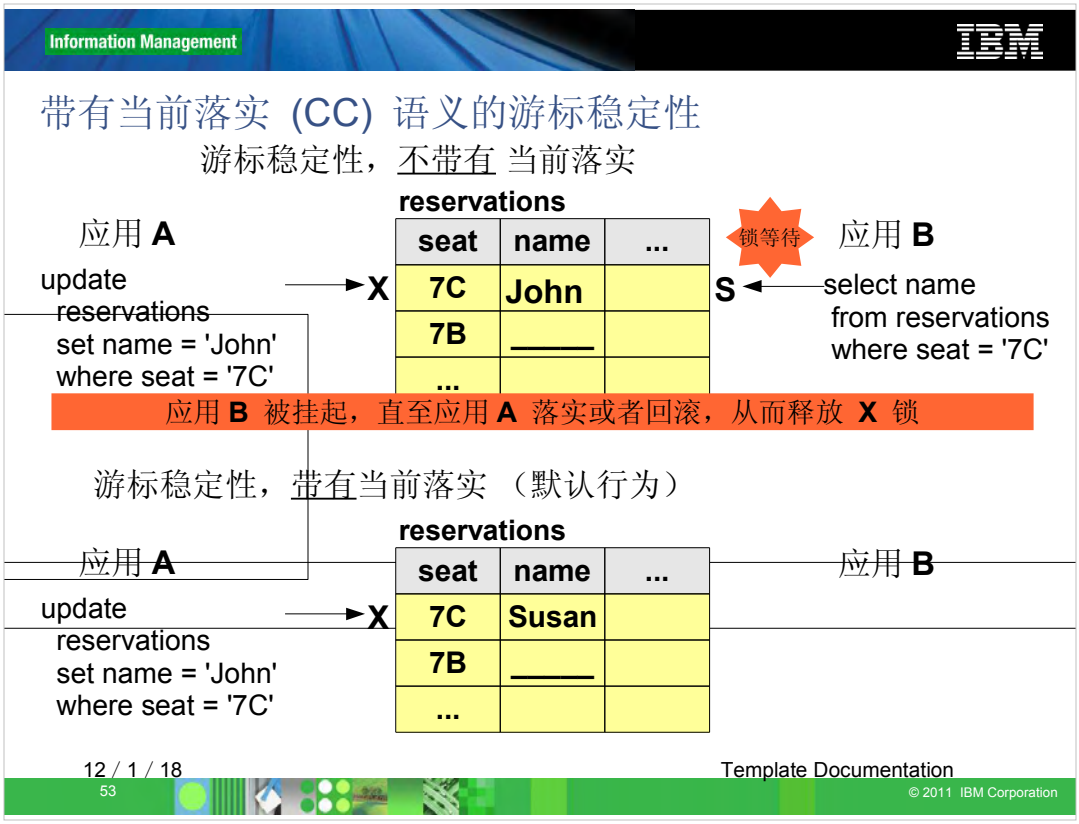


游标稳定性，带有当前落实 （默认行为）

于是，它只能等待。



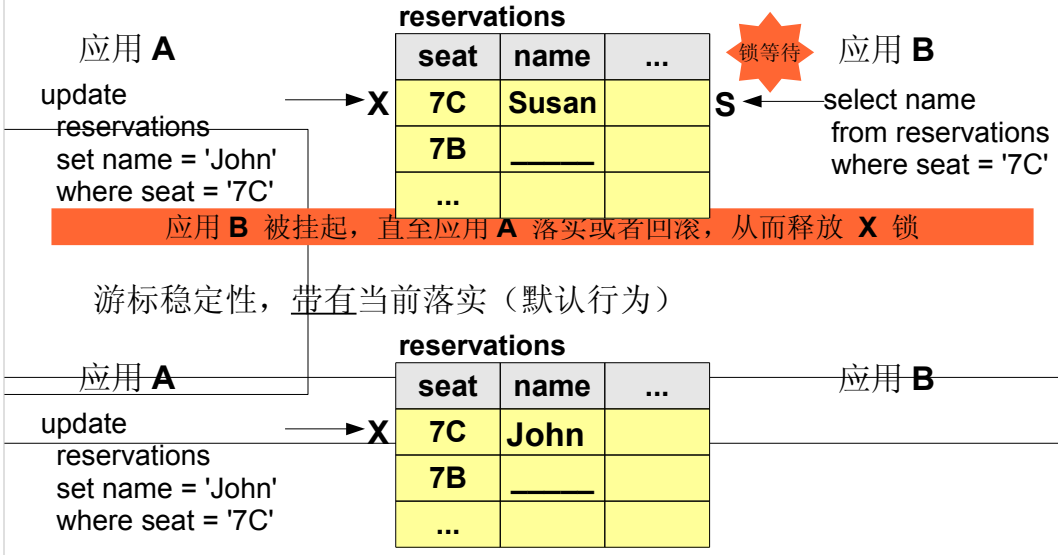
在用户看来，应用 B 似乎被挂起了。只有当应用 A 落实之后，或者回滚之后，应用 B 才能够继续处理 SELECT 。



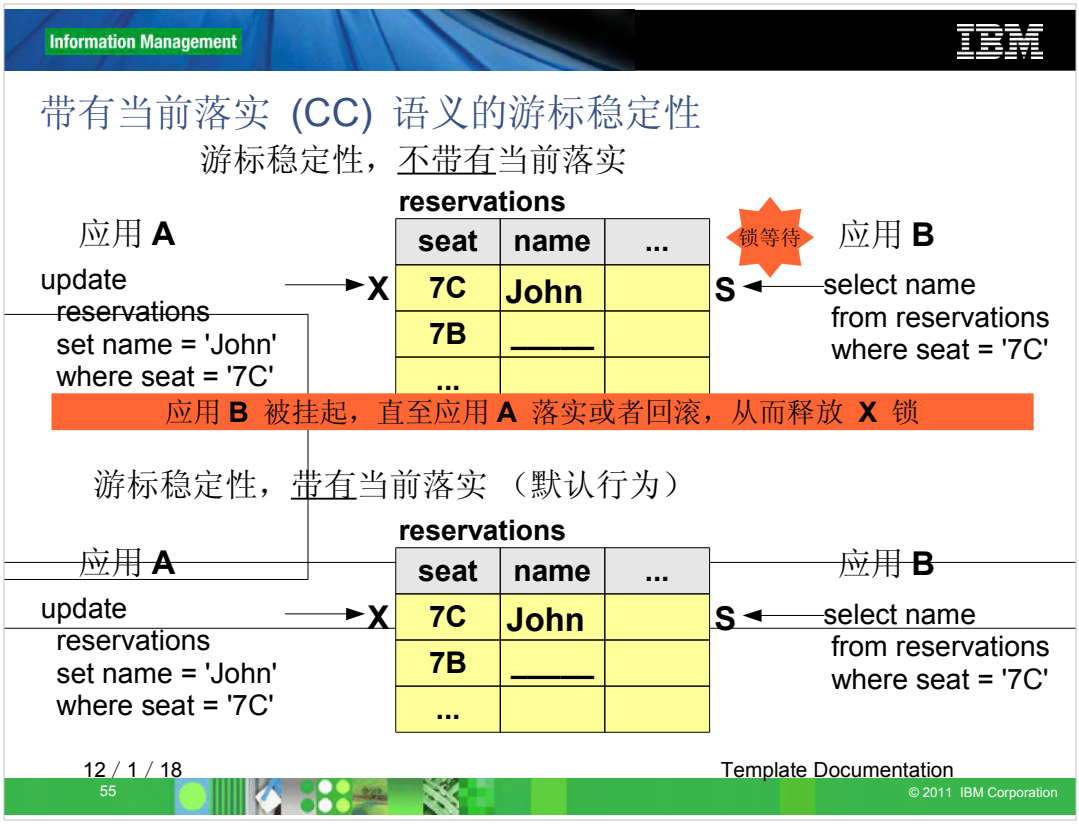
现在我们来看看下半部分带有 CC 的 CS 的行为：
在相同的情况下，应用 A 进行了一次更新，并获得了一个排他锁（ X 锁）。

带有当前落实 (CC) 语义的游标稳定性

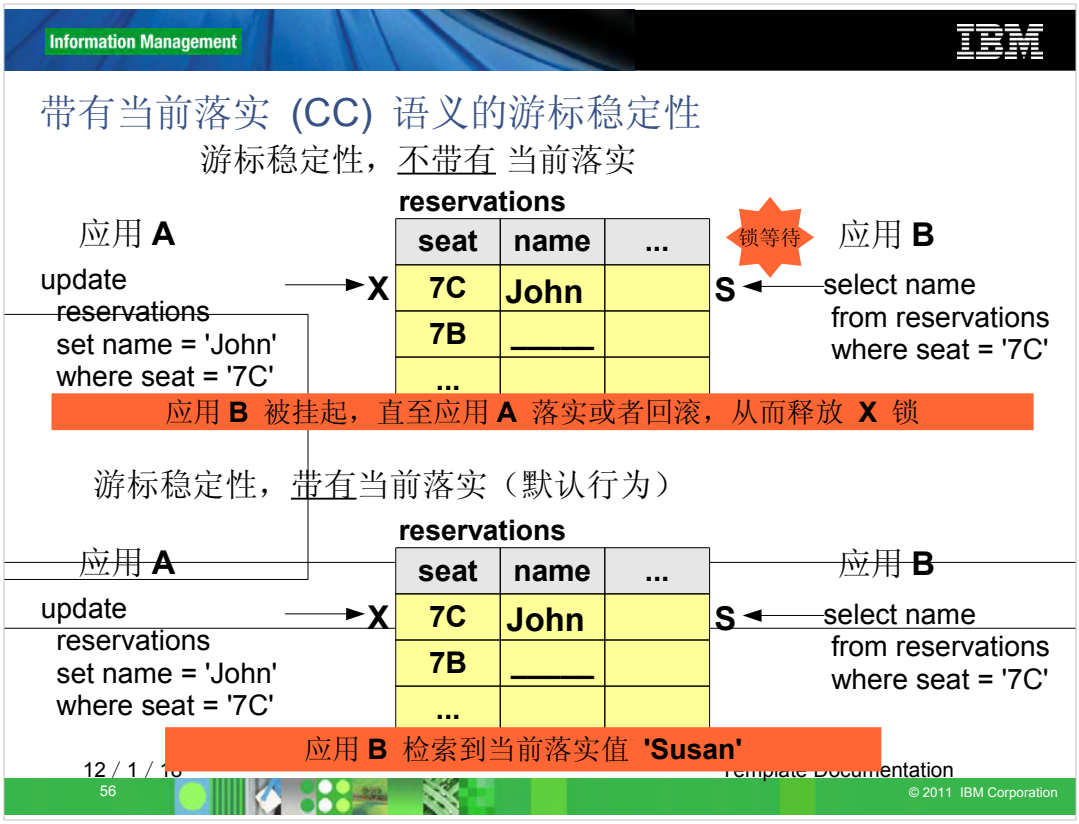
游标稳定性，不带有 当前落实



游标稳定性，带有当前落实（默认行为）



当应用 B 对同一行发出 SELECT 时，它被允许执行，



虽然它将检索到***当前落实***的值，在本例中即为“ Susan”。因此在本例中，写入方 (update) 没有封锁读取方 (SELECT)

注：如果应用 B 在使用 UR，所检索到的值将会是 'John'，这是一个未落实的值；但在 CS 之下，你只能检索到已落实的值。

隔离级别的比较和选择

Isolation Level	Lost update	Dirty Read	Non-repeatable Read	Phantom Read
Repeatable Read (RR)	-	-	-	-
ReadStability (RS)	-	-	-	Possible
Cursor Stability (CS)	-	-	Possible	Possible
Uncommitted Read (UR)	-	Possible	Possible	Possible

Application Type	High data stability required	High data stability not required
Read-write transactions	RS	CS
Read-only transactions	RS or RR	UR

本图总结了哪些隔离级别解决了哪些并发性问题。“丢失更新”问题可以由所有的隔离级别解决。

下端的表向你示出了在什么时间使用那种类型的隔离级别，这取决于你的应用类型以及你的数据的“稳定性”，就是说，你打算让这些数据有什么样的准确程度。

议题

事务

并发性和锁定

锁等待

死锁

锁等待

默认情况下，应用会 infinite 以获得任何需要的锁

LOCKTIMEOUT (db cfg):

规定了等待获得锁的秒数

默认值为 -1 或者*无限期*等待

示例：与使用不带有 CC 的隔离 CS 时相同）：

应用 **A**

reservations

seat	name	...
7C	Susan	
7B	_____	
...		

应用 **B**

在我们讨论不带当前落实的游标稳定性时，我们已经看到了锁等待的工作方式。我们来重新看一下这个例子：

锁等待

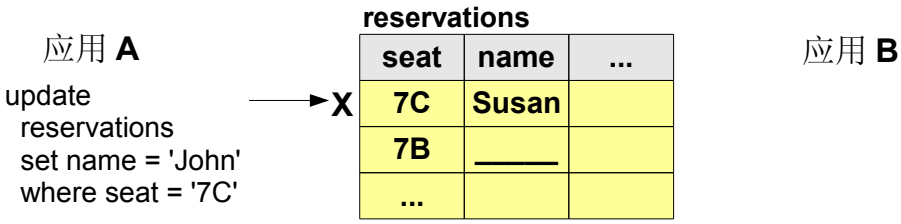
默认情况下，应用会无限期等待以获得任何需要的锁

LOCKTIMEOUT (db cfg):

规定了等待获得锁的秒数

默认值为 -1 或者*无限期*等待

示例：与使用不带有 CC 的隔离 CS 时相同）：



应用 A 进行一次更新，获得了 X 锁。

锁等待

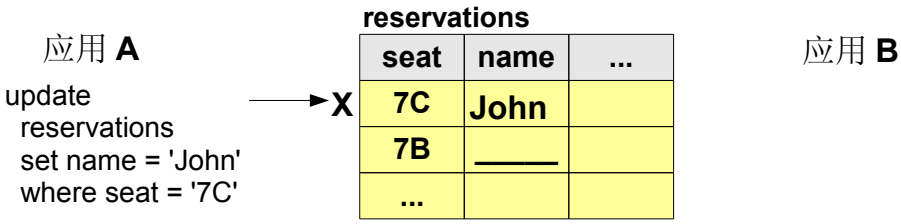
默认情况下，应用会无限期等待以获得任何需要的锁

LOCKTIMEOUT (db cfg):

规定了等待获得锁的秒数

默认值为 -1 或者*无限期*等待

示例：与使用不带有 CC 的隔离 CS 时相同）：



锁等待

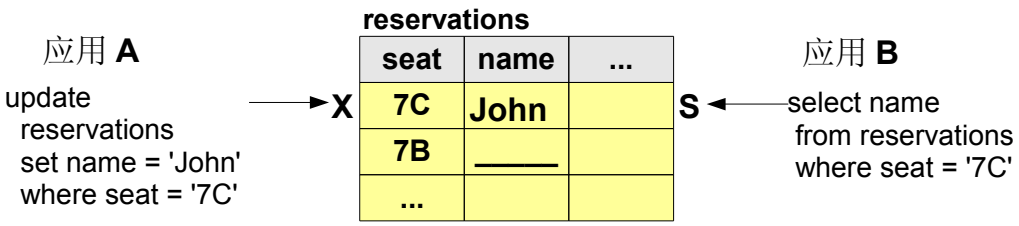
默认情况下，应用会无限期等待以获得任何需要的锁

LOCKTIMEOUT (db cfg):

规定了等待获得锁的秒数

默认值为 -1 或者*无限期*等待

示例：与使用不带有 CC 的隔离 CS 时相同）：



应用 B 希望读取该行（它将获得 S 锁），但无法进行，因为在不带 CC 的 CS 中， X 锁与 S 锁不兼容。

锁等待

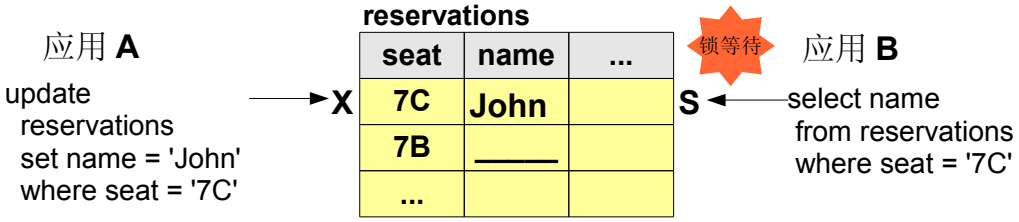
默认情况下，应用会无限期等待以获得任何需要的锁

LOCKTIMEOUT (db cfg):

规定了等待获得锁的秒数

默认值为 -1 或者*无限期*等待

示例：与使用不带有 CC 的隔离 CS 时相同）：



因此，应用 B 必须等待。

它将等待多久呢？

锁等待

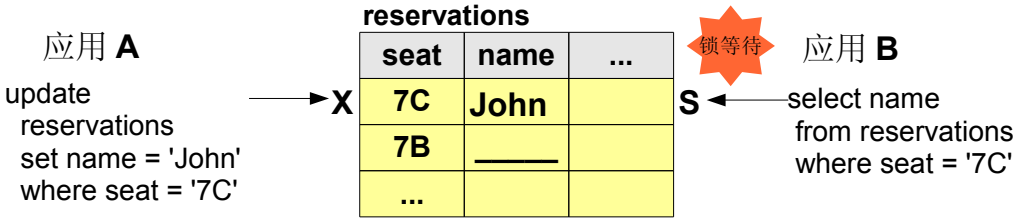
默认情况下，应用会无限期等待以获得任何需要的锁

LOCKTIMEOUT (db cfg):

规定了等待获得锁的秒数

默认值为 -1 或者*无限期*等待

示例：与使用不带有 CC 的隔离 CS 时相同）：

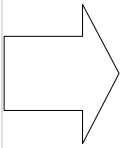


应用 **B** 等待 **“ LOCKTIMEOUT”** 秒数以获得第一行的 **'S'** 锁

这由参数 LOCKTIMEOUT 设定。默认情况下，该参数被设置为 -1，这意味着无限期等待。

议题

- 事务
- 并发性和锁定
- 锁等待
- 死锁



死锁

两个或多个应用无限期等待某项资源时

每个应用都掌握着别的应用所需要的资源

等待无法得到解决

在本例中，假设我们正在使用不带有 CC 的 CS 隔离

应用 **A**

reservations

seat	name	...
7C	Susan	
7B	_____	
...		
8E	Raul	
9F	Jin	

应用 **B**

本示例说明死锁是如何发生的。请注意，应用 A 中的操作发生在一个事务中。应用 B 也一样。注意到这一点很重要，因为，假如应用 A 进行了 UPDATE，然后它又执行了 COMMIT，它将会释放全部的锁，那样的话，我们就面对完全不同的情况了。

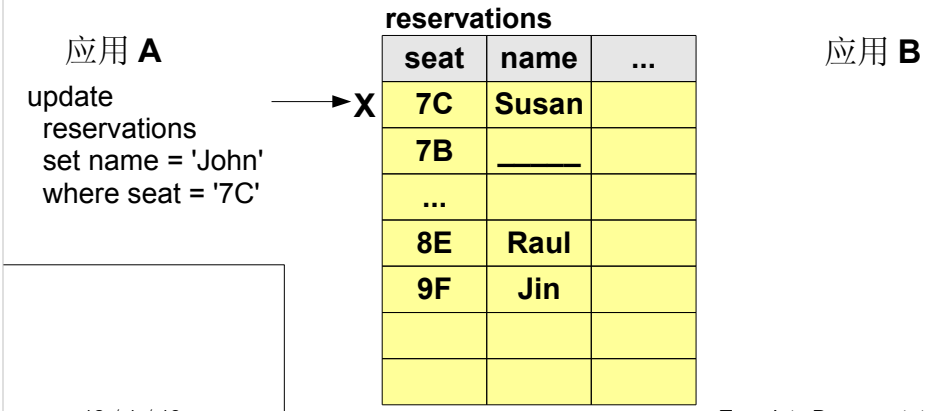
死锁

两个或多个应用无限期等待某项资源时

每个应用都掌握着别的应用所需要的资源

等待无法得到解决

在本例中，假设我们正在使用不带有 CC 的 CS 隔离



应用 A 进行了一次更新，因此在 seat 7C 所对应的行上获得了 X 锁。

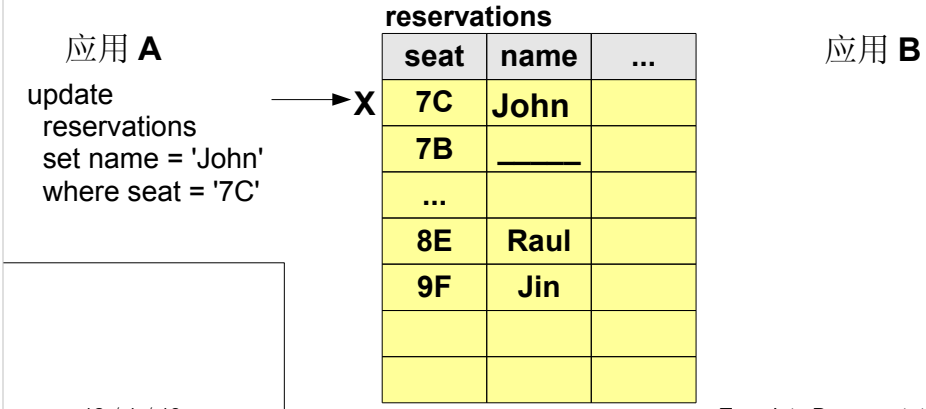
死锁

两个或多个应用无限期等待某项资源时

每个应用都掌握着别的应用所需要的资源

等待无法得到解决

在本例中，假设我们正在使用不带有 CC 的 CS 隔离



该行被更新为“ John”，虽然此修改还没有落实。

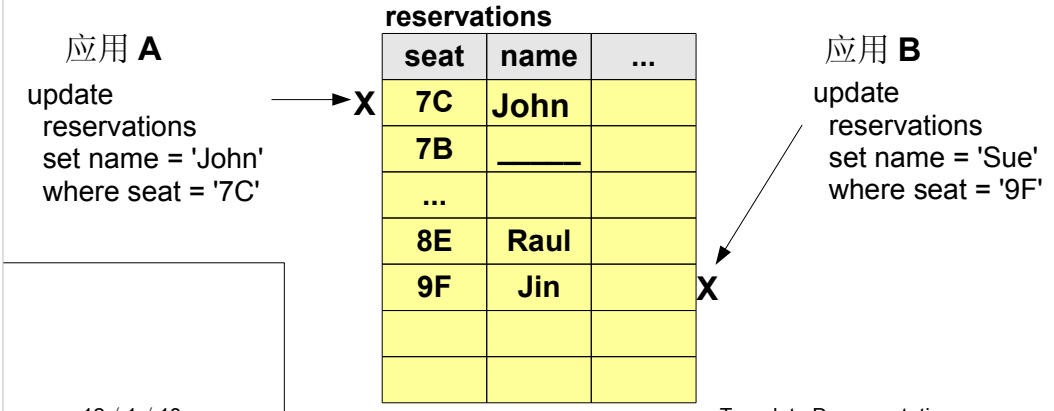
死锁

两个或多个应用无限期等待某项资源时

每个应用都掌握着别的应用所需要的资源

等待无法得到解决

在本例中，假设我们正在使用不带有 CC 的 CS 隔离



应用 B 对另一行进行更新，即 seat '9F' 对应的行。它也在此行上获得 X 锁。

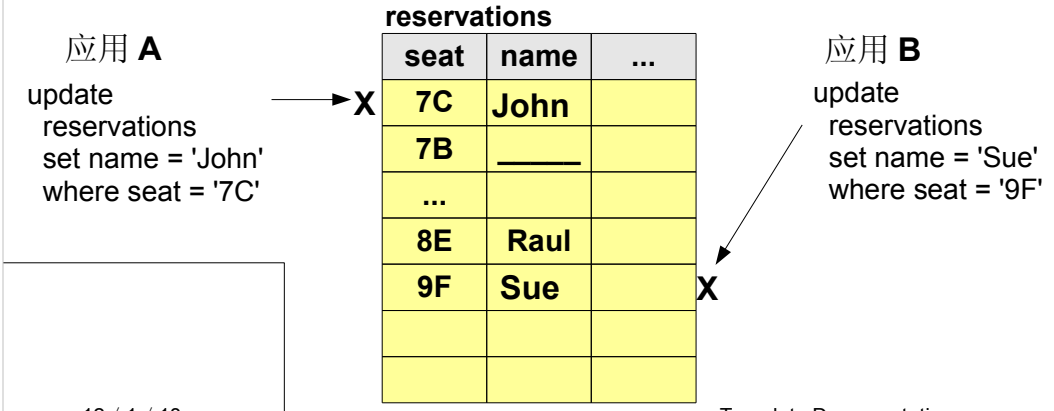
死锁

两个或多个应用无限期等待某项资源时

每个应用都掌握着别的应用所需要的资源

等待无法得到解决

在本例中，假设我们正在使用不带有 CC 的 CS 隔离



该行被更新为“ Sue”，虽然此修改还没有落实。

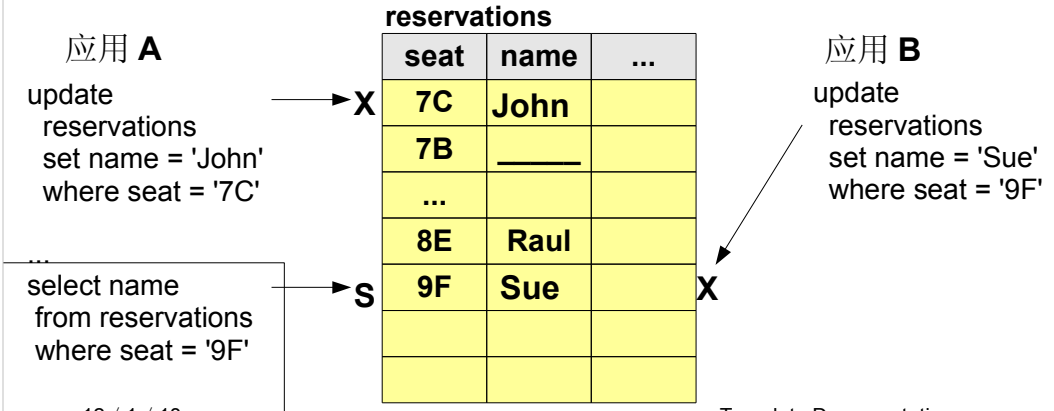
死锁

两个或多个应用无限期等待某项资源时

每个应用都掌握着别的应用所需要的资源

等待无法得到解决

在本例中，假设我们正在使用不带有 CC 的 CS 隔离



应用 A 对 seat 9F 对应的行发出 SELECT 语句，因此，它希望对该行获得 S 锁。

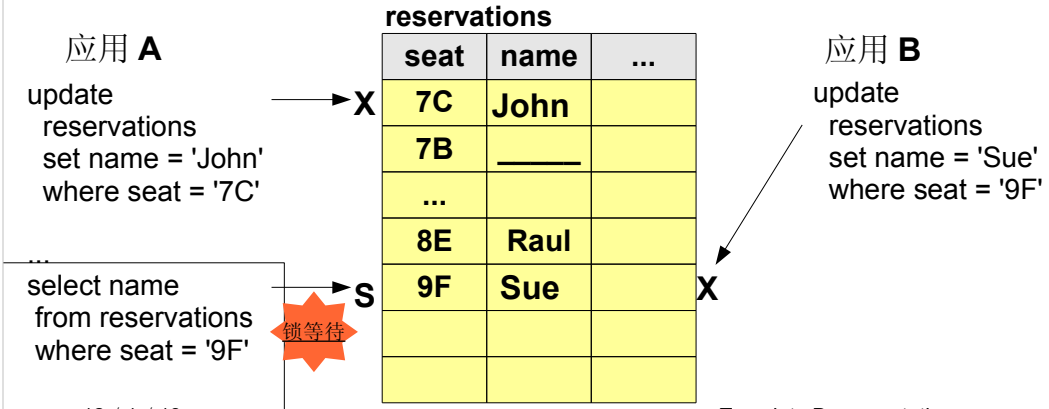
死锁

两个或多个应用无限期等待某项资源时

每个应用都掌握着别的应用所需要的资源

等待无法得到解决

在本例中，假设我们正在使用<u>不带有</u> CC 的 CS 隔离



应用 A 挂起。它只能等待，因为应用 B 在 seat 9F 对应的行上有一个 X 锁。

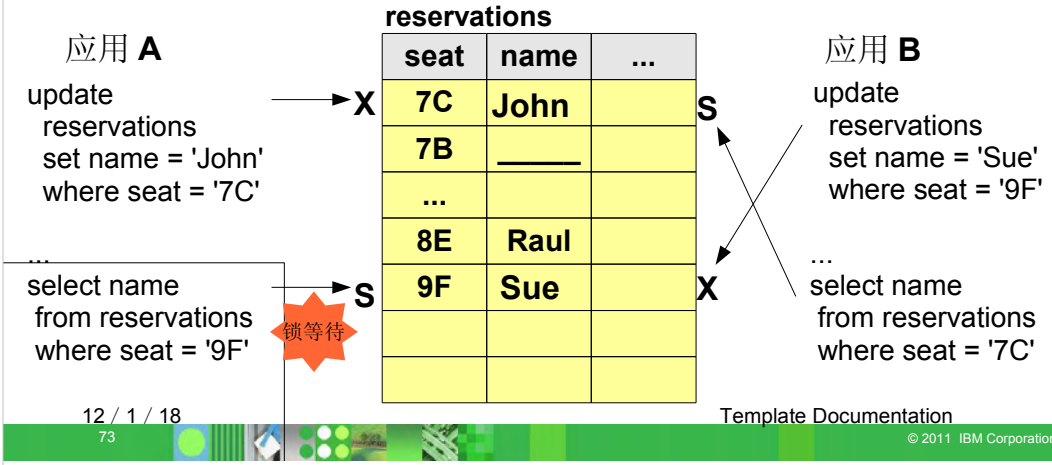
死锁

两个或多个应用无限期等待某项资源时

每个应用都掌握着别的应用所需要的资源

等待无法得到解决

在本例中，假设我们正在使用不带有 CC 的 CS 隔离



应用 B 对 seat 7C 对应的行发出 SELECT 语句，因此，它希望对该行获得 S 锁。

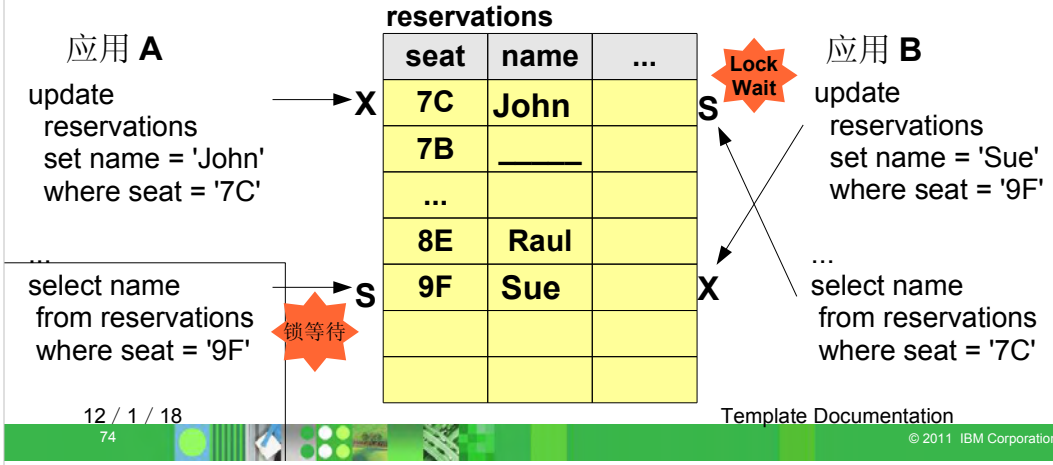
死锁

两个或多个应用无限期等待某项资源时

每个应用都掌握着别的应用所需要的资源

等待无法得到解决

在本例中，假设我们正在使用不带有 CC 的 CS 隔离



应用 B 挂起。它只能等待，因为应用 A 在 seat 7C 对应的行上有一个 X 锁。

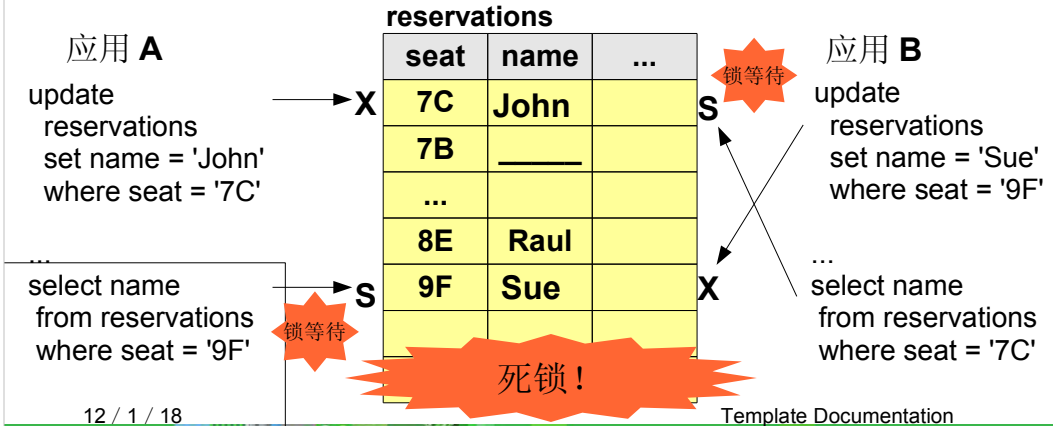
死锁

两个或多个应用无限期等待某项资源时

每个应用都掌握着别的应用所需要的资源

等待无法得到解决

在本例中，假设我们正在使用不带有 CC 的 CS 隔离



于是，我们遇到死锁了！应用 A 在等待应用 B 释放它的锁，然后应用 A 才能继续进行；而应用 B 也在等待应用 A 释放它的锁，然后应用 B 才能继续进行。

死锁

死锁通常是由于不良的应用设计引起的

DB2 提供了死锁探测器

使用 **DLCHKTIME** (db cfg) 来设置检查死锁的时间间隔

但发现死锁时， DB2 使用一种间隔算法来决定哪项事务应当回滚，哪项事务应当继续进行

被强制回滚的事务得到一条 SQL 错误消息。回滚导致其全部的锁被释放

死锁通常是由于不良的应用设计引起的。
DB2 具有死锁探测器。如果存在死锁的话，它会检查每一个‘ x’ 的秒数。 ‘ x’ 由死锁检查时间 (DLCHKTIME) db cfg 参数来设定。当死锁探测器探测到死锁时， DB2 将使用一种间隔算法来挑选导致死锁的两个应用中的某一个，而且对该应用进行回滚操作。被回滚的应用收到的准确 SQL 错误消息是：
SQLCODE -911 (SQLSTATE 40001)，原因代码 2（这里的原因代码‘ 2’ 是指，“该事务由于死锁而被回滚”。另一个应用则被准许继续执行。

如果你对本课程所讨论的内容有什么技术方面的问题的话，可以利用 db2university.com 课程 AA001EN 中的论坛。 同学、教员以及 IBM 人员可以帮助你！

Information Management

IBM

DB2 数据库安全性

IBM 信息管理云计算能力中心
IBM （加拿大）研究院

议题

- DB2 安全性概述
- 认证
- 授权
- 角色
- PUBLIC 组
- 基于标签的细粒度访问控制
- 基于标号的访问控制
- 扩展的安全性（仅限 Windows ）
- 可信上下文

支持性阅读材料和视频

阅读材料

DB2 Express-C 入门电子书

第 10 章： 数据库安全性

视频

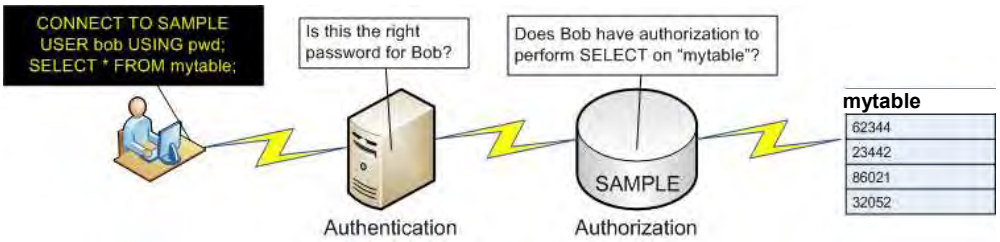
db2university.com 课程 AA001EN

第 9 课： 数据库安全性

议题

- DB2 安全性概述
- 认证
- 授权
- 角色
- PUBLIC 组
- 通过视图进行细粒化访问
- 基于标号的访问控制
- 扩展的安全性（仅限 Windows）
- 可信上下文

DB2 安全性概述



DB2 安全性包含两个步骤：

认证（在 **DB2** 外部进行）

检查用户名和密码

由安全性插件完成，它由调用 DB2 之外的设施（默认为操作系统）

授权（由 **DB2** 进行）

检查通过认证的用户是否有权进行所请求的操作

由 DB2 通过使用存储在 DB2 目录及 DBM 配置文件中的消息来完成

本张幻灯片表明， DB2 中的安全性是通过两个步骤来实施的：第一步被称为 "**认证**"，通常由操作系统来进行，或者由 DB2 之外的外部安全性设施来进行。第二步被称为 "**授权**"，由 DB2 进行。例如，在上半部分，我们看到用户 Bob 发起对 SAMPLE 数据库的连接，他传递了 "*USER bob using pwd*"，其中 " pwd" 是密码。当它发出这个 connect （连接）语句时，该语句将首先进入 DB2，由 DB2 在服务器上检查 dbm cfg 参数 *authentication* 的值，然后根据该参数值，请求客户端或服务器的操作系统（或外部安全性设施）进行认证。

因此，我们再次看到，认证并不是由 DB2 进行的，而通常是由操作系统或外部设施进行的，甚至你可以自己编写代码来进行认证。这与其他关系数据库管理系统不同，因为其他系统通常可以在其数据库系统中定义用户；但在 DB2 中，你确实不能定义用户。用户是在 DB2 之外定义的；组也是这样。他们都在 DB2 之外定义。通常这也是操作系统用户或操作系统组。这种方法可以避免在不同的层次上定义太多的用户 ID/ 密码。

一旦操作系统确定有一个密码为 "pwd" 的用户 "bob"，它就通知 DB2，好的，这个人可以连接。此后， DB2 就接管 "*授权*" 部分（第 2 步），由 DB2 检查该用户是否可以执行 : **select * from mytable**，等等。在本例中，当用户开始在数据库执行操作时， DB2 将检查用户 bob 是否具有 "select" 特权以及他是否能够对表 "mytable" 执行该操作。

议题

- DB2 安全性概述
- 认证
- 授权
- 角色
- PUBLIC 组
- 通过视图进行细粒化访问
- 基于标号的访问控制
- 扩展的安全性（仅限 Windows）
- 可信上下文

认证

认证在什么地方进行（用户 ID/ 密码检查）？

DBM CFG 参数 AUTHENTICATION （在 DB2 服务器上）决定了在什么地方 / 如何进行认证

AUTHENTICATION 参数的一些有效值是：

ID	NAME	AGE	PHONE	SALARY
12	Peter	30	99999	10000
3	Mary	23	88888	15000

如前所述， DB2 不进行认证；但它确实有一个称为 “ AUTHENTICATION” 的 dbm cfg 参数来规定在在什么地方 / 如何进行认证。

该表示出了本参数的不同值。

SERVER 是默认值

SERVER_ENCRYPT 与 SERVER 类似，但密码通过网络时也被加密

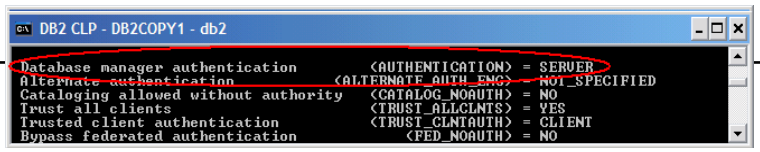
DATA_ENCRYPT 与 SERVER_ENCRYPT 类似，但通过网络传递的数据也被加密

如果你打算在你的代码指定插件，由它进行认证的话，你就应当选择 GSSPLUGIN 。

在 DB2 服务器上配置认证

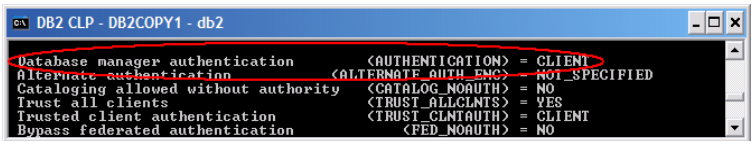
示例：

```
db2 "GET DBM CFG"
```



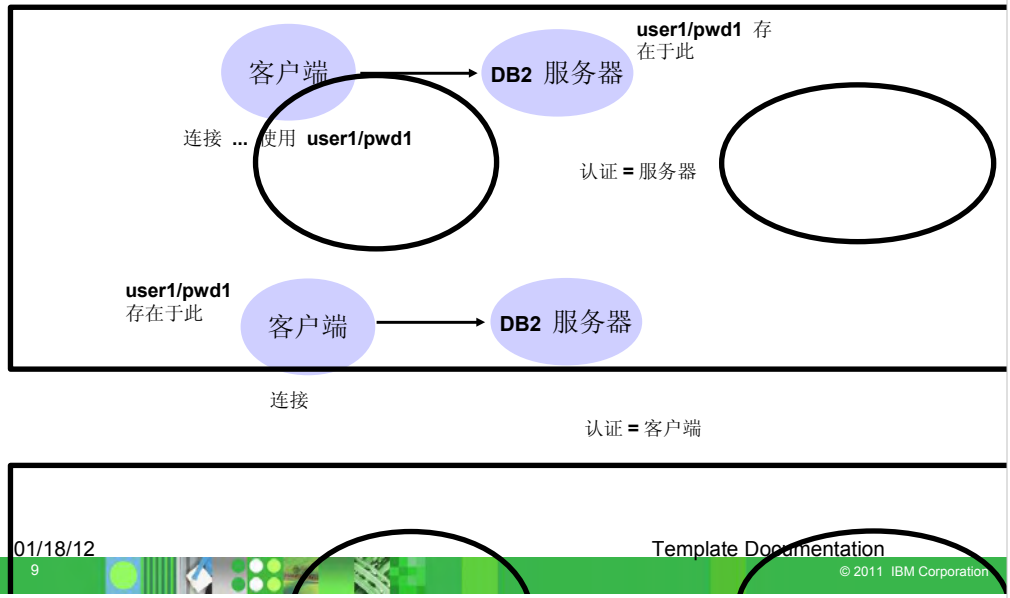
修改使认证在客户端进行：

```
db2 "UPDATE DBM CFG USING AUTHENTICATION CLIENT"
```



这解释了如何查看 AUTHENTICATION 参数的内容以及如何更新它。

认证（续）



该图再次说明了认证实际上根据 dbm cfg AUTHENTICATION 参数的值在什么地方进行。

如果 AUTHENTICATION 被设置为服务器，具有密码“ pwd1”的用户“ user1”就需要在服务器的操作系统（或者外部安全性设施）中定义。

如果 AUTHENTICATION 被设置为客户端，具有密码“ pwd1”的用户“ user1”就需要在客户端的操作系统（或者外部安全性设施）中定义。当把 AUTHENTICATION 设置为客户端时，有多项参数需要设置，但本演示中不讨论这些内容了。

议题

- DB2 安全性概述
- 认证
- 授权
- 角色
- PUBLIC 组
- 通过视图进行细粒化访问
- 基于标号的访问控制
- 扩展的安全性（仅限 Windows）
- 可信上下文

授权

验证某个授权 ID 是否有足够的特权来进行所期望的数据库操作

例如： Bob 有对表 MYTABLE 进行 SELECT 操作的权限吗？

可以利用以下方法来赋予权利：

特权：
细粒化

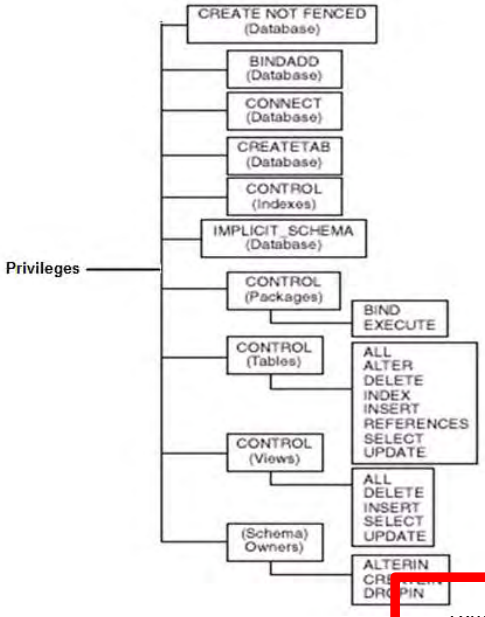
权限：
内置的。例如： SYSADM, DBADM, SECADM, 等等

角色：
例如用户定义的权限

这是“安全性概述”幻灯片所述内容的第二部分。在本部分中， DB2 进行安全性检查，例如，“用户 Bob 有权对表 mytable 进行 SELECT 操作吗 ?"

在下面几个幻灯片的“授权”主题中，将详细讨论特权、权限和角色。

特权



我们从“授权”主题的第一项开始：特权。本图示出了你能够向用户或组授予的不同特权的列表。特权提供了细粒化的安全性控制，你可以藉此仅允许用户获得在特定操作中针对特定数据库对象的访问权。有许多种特权，长方形中的这些特权将在下面几张幻灯片中进行详细讨论。

特权 - 示例

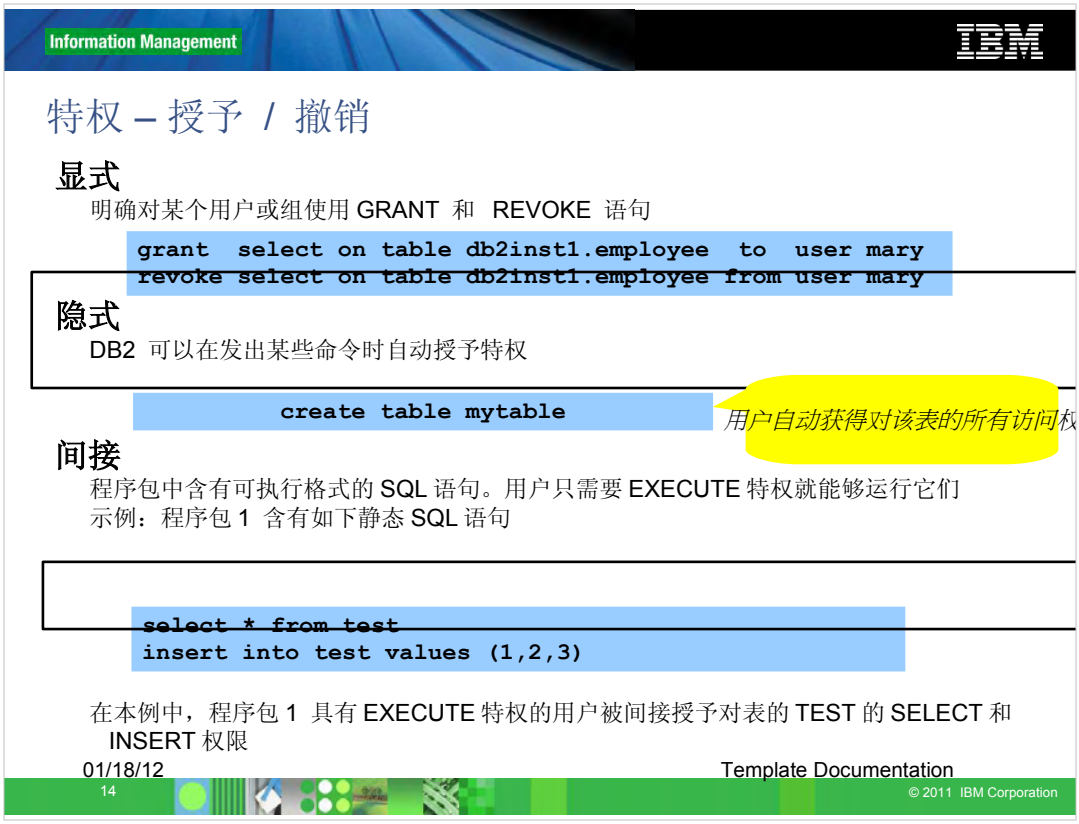
模式特权

CREATEIN: 可以在模式中创建对象
ALTERIN: 可以改变模式中的对象
DROPIN: 可以从模式中删除对象

表和视图特权

CONTROL: 对表或视图具有完全控制权，包括删除、授予 / 撤销
- DELETE: 可以删除行
- INSERT: 可以插入行并运行 IMPORT 工具
- SELECT: 可以检索行、创建视图、运行 EXPORT 工具
- UPDATE: 可以修改列、表或视图中的项目
- ALTER: 可以修改表
- INDEX: 可以对表创建索引
- REFERENCES 可以创建和删除外键

本幻灯片示出了在模式层次以及在表和视图层次的不同特权，并对它们进行了简要描述。注意，如果你把 CONTROL 特权授予某个用户或组，你不仅授予了删除的权利，或者授予了再向其他人授予该特权的权利，而且你还隐式授予了所列出的所有其他特权，例如 DELETE 、 INSERT 、 SELECT ， 等等。



有三种途经向用户授予特权：

- 显式
- 隐式
- 间接

如果一项特权隐式提供了其他特权，那么，当你撤销这项特权时，有些事情可能不那么直观。当你撤销该特权时，隐式获得的那些特权将不会被撤销。你需要显式撤销它们。

例如：

db2 grant control on table employee to user raul

==> 授予 control 以及所有特权，例如 ALTER 、 DELETE 、 INDEX ，等等（参见第 12 张幻灯片）

db2 revoke control on table employee from user raul

==> 仅撤销 CONTROL 权。其他的特权，例如 ALTER 、 DELETE 等，仍然存在。

在撤销 CONTROL 之后，你必须显式发出：

db2 revoke all privileges on table employee from user raul

以确保所有隐式授予的特权都被撤销。

（演示）

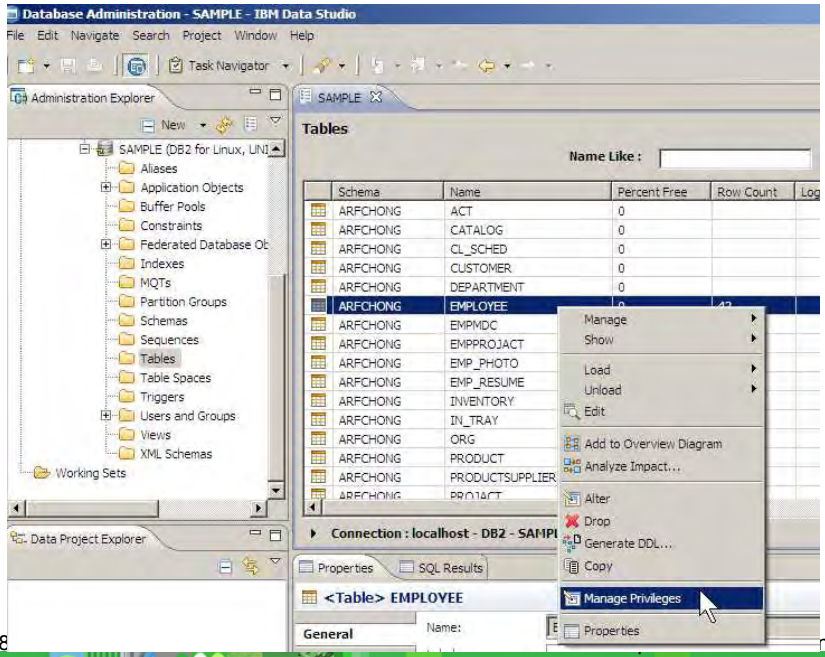
特权 – 授予 / 撤销（示例）

```
GRANT   SELECT   ON    TABLE     T1    TO   USER   user1
GRANT    ALL     ON    TABLE     T1    TO   GROUP  group1
REVOKE   ALL     ON    TABLE     T1   FROM  GROUP  group1
GRANT   EXECUTE  ON  PROCEDURE   p1    TO   USER   user1
REVOKE  EXECUTE  ON  PROCEDURE   p1   FROM  USER   user1
REVOKE  CONNECT  ON   DATABASE         FROM  USER   user2
```


这是向用户或组显式授予特权的一项例子。

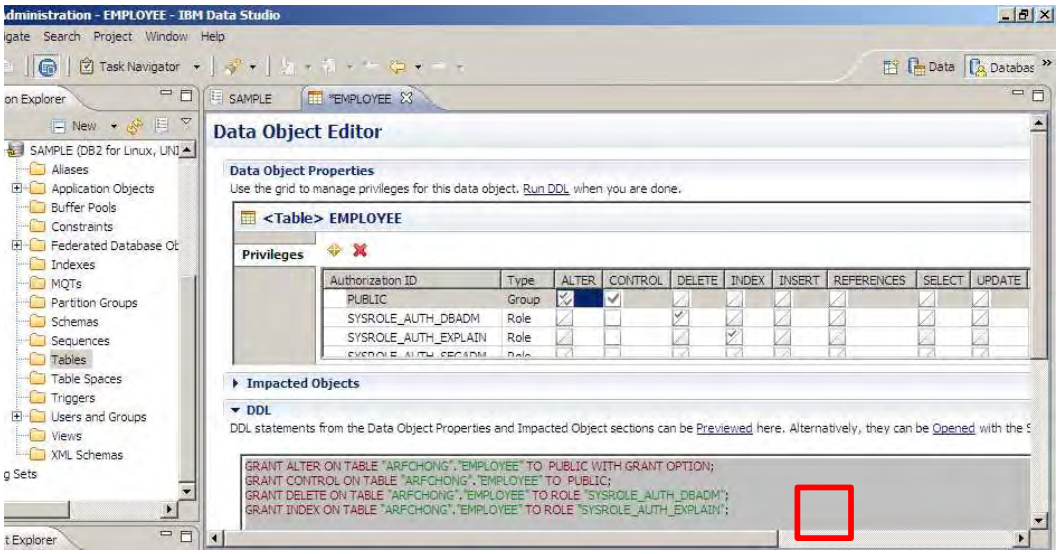
请注意，最后一个例子未提供数据库的名称。在运行 GRANT/REVOKE 语句时，假设你已经接入数据库；因此，如果你打算撤销对数据库的 CONNECT 特权，你不需要指定该数据库的名称，因为当你发出此语句时你已经接入该数据库了。

在 Data Studio 中管理特权



在 Data Studio 中，右击此表，选择“管理特权”。

在 Data Studio 中管理特权



在 Data Studio 中，当你看到图中突出显示的两个箭头时，这意味着拥有该特权的用户也能够该特权授予其他人。

幕后的过程是， Data Studio 生成 GRANT/REVOKE 语句，而两个箭头意味着，在语句中将包含“ WITH GRANT OPTION 选项。

（演示）

权限

实例级权限

SYSADM, SYSCTRL, SYSMAINT, SYSMON

数据库级权限

DBADM, SECADM, SQLADM, WLMADM, EXPLAIN, ACCESSCTRL, DATAACCESS, 等等

我们上面讨论了特权，现在来看看“授权”主题的第二项：权限。

“内置”了许多不同的权限（它们随同 DB2 一同提供），它们可能是实例级的权限，也可能是数据库级的权限。

你可以把权限视为一种把特权组合到一起的方法。

因此，对于实例级权限，我们有 SYSADM, SYSCTRL, SYSMAINT, SYSMON 。

对于数据库级权限，我们有 DBADM, SECADM, SQLADM, WLMADM, EXPLAIN, ACCESSCTRL, DATAACCESS ，等等。

实例级权限

Authority	Description
SYSADM	Manages the instance as a whole
SYSCTRL	Administers a database manager instance
SYSMAINT	Maintains databases within an instance
SYSMON	Monitors the instance and its databases

此表简要说明了各项权限能够完成的工作。如我们在后面将会看到的，拥有任何此类权限并不意味着对存储在数据库中的数据拥有访问权。例如， SYSADM 可能无法看到你的数据库中的内容。在过去的 DB2 版本中， SYSADM 拥有全部权力，可以做任何事情，但从 DB2 9.7 开始，事情不再是这样了。

实例级权限

SYSADM 、 SYSCTRL 、 SYSMAINT 和 SYSMON 根据操作系统组在 DBM CFG 中定义：

```
update dbm cfg using SYSCTRL_GROUP   <group_name>
update dbm cfg using SYSMAINT_GROUP  <group_name>
update dbm cfg using SYSADM_GROUP    <group_name>
update dbm cfg using SYSMON_GROUP    <group_name>
```

各个实例有其自己的权限组定义

在 Windows 中，如果这些值是空白的话，本地 Windows Administrators 组、 LocalSystem 帐户和 DBADMNS 组（如果启用了 Extended Security 的话）将是 SYSADM

在 Linux/UNIX 中，实例所有者的组是 SYSADM

为了向某个组授予 SYSADM 或者任何其他实例级权限，你可以如本幻灯片所示在 dbm cfg 中进行。该组应当在操作系统层次上（或者外部安全性设施中）定义。

如果本幻灯片中所示参数是空白的话（即取默认值），那么， Windows 中的 LOCAL ADMINISTRATOR 组，或者 Linux 中 DB2 实例所有者（例如： db2inst1 ）所属的组，都将是 SYSADM 。

实例级权限 - SYSADM

实例级上最高层次的管理权限

只有具备 SYSADM 权限的用户才能够：

升级和恢复数据库

修改数据库管理器配置文件，其中包括规定具有 SYSADM 、 SYSCTRL 、 SYSMAINT 或 SYSMON 权限的组

不能隐式获得 DBADM 权限，也不能自动获得对数据库的访问权

通过 `sysadm_group` 参数在 DBM CFG 中规定

示例：把 SYSADM 权限授予组 'mygrp':

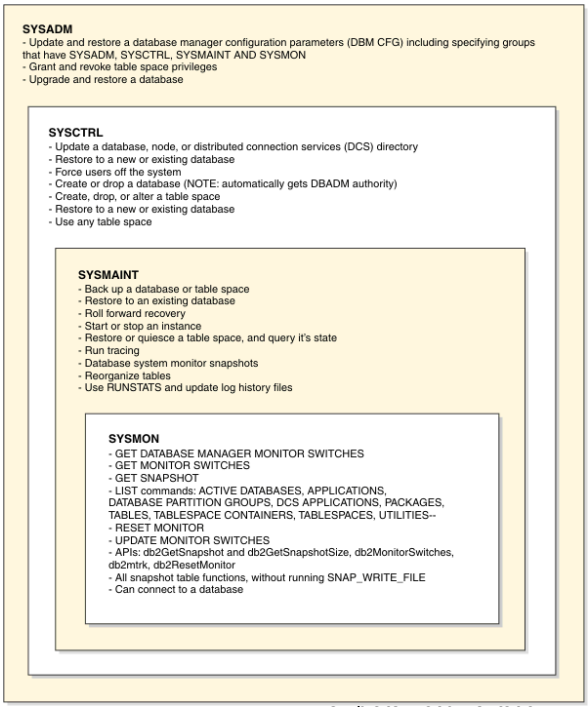
```
UPDATE DBM CFG USING SYSADM_GROUP mygrp
```


本幻灯片提供了关于 SYSADM 的详细信息。

获得 SYSADM 权限并不意味着用户也将成为 DBADM 。前面说过， SYSADM 也不必然拥有对数据库的访问权。我们将看到，要授予这种访问权，还需要 SECADM 。

为了授予 SYSADM ，你需要更新 DBM CFG ，必须在其中指定操作系统组。组中的每名用户都将成为 SYSADM 。

实例级权限



本图直接取自 DB2 手册，它概述了 DB2 实例级权限的层次，也介绍了它们能完成的工作。

数据库级权限

Authority	Description
SECADM	Manages security within a database
DBADM	Administers a database
new in V9.7 ACCESSCTRL	Grants and revokes authorities and privileges (other than SECADM, DBADM, ACCESSCTRL, and DATAACCESS authority. Note that SECADM authority is required to grant and revoke these authorities)
new in V9.7 DATAACCESS	Provides the ability to access data in a database.
SQLADM	Monitors and tunes SQL queries
WLMADM	Manages workloads
EXPLAIN	Users who need to explain query plans (EXPLAIN authority does not give access to the data itself)

我们现在来讨论“数据库级”权限。

该表取自“ DB2 Express-C 入门”一书，它列出了不同的数据库级权限并做了描述。

SECADM 权限

SECADM 是给定数据库的 Security Administrator （安全管理员）

授予 / 撤销数据库级别上的所有安全性权限。没有实例级的权限。

默认情况下，创建数据库的用户即为 SECADM

不能访问存储在用户表中的数据

只能由具有 SECADM 权限的用户授予

可以向 SYSADM 授予 SECADM 权限

SECADM （安全性管理员）是用于管理数据库安全性的权限。它在实例级层次上什么事情也做不了。在默认情况下， SECADM 是创建数据库之人。 SECADM 可以向其他人授予 SECADM 权限以及数据访问权。

DBADM 权限

给定数据库的超级用户

- 没有实例级的权限
- 可能不具有对数据库的访问权
- 可能无法授予 / 撤销权限 / 特权

需要 SECADM 权限才能够授予 / 撤销对用户、组或角色的 DBADM 。例如：

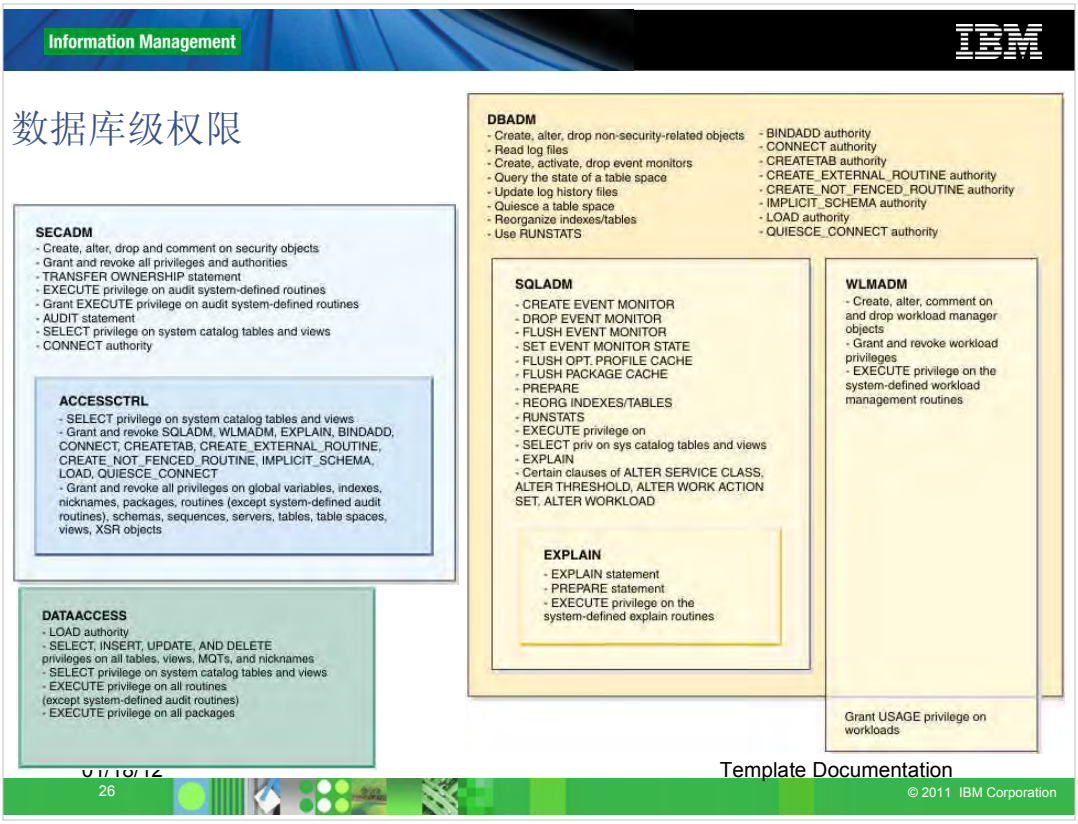
```
GRANT DBADM on database to user <userID>
```

DBADM 在默认情况下还将获得下述权限：

- DATAACCESS
- ACCESSCTRL

DBADM 是数据库管理员。他仅具有数据库级的特权，而没有实例级的特权。 DBADM 并不必然需要拥有对数据的访问权；但是，在默认情况下，他获得 DATAACCESS 和 ACCESSCTRL 权限；因此，他可以访问数据并把权限授予其他人。

许多人会感到混乱，认为也可以使用 dbm cfg 来授予 DBADM ；但如你所见，它是使用 GRANT 语句来授予的。



这些图直接取自于 DB2 手册，示出了不同的数据库级权限以及它们的层次，并描述了它们能完成的工作。

选择我们来看看“授权”主题中的第三个项目：角色

角色

角色类似于“用户定义的”数据库级权限

不能向角色授予实例级权限（例如 SYSADM ）

示例：

```
CREATE ROLE TESTER
GRANT SELECT ON TABLE STAFF TO ROLE TESTER
GRANT SELECT ON TABLE DEPT  TO ROLE TESTER
GRANT  ROLE TESTER TO USER RAUL, USER JIN
REVOKE ROLE TESTER FROM USER JIN
```


你可以把角色视为“用户定义的”数据库级权限。
你首先创建一个角色并授予其不同的特权。
然后你可以对用户或组授予 / 撤销角色，其方式与对用户 / 组授予 / 撤销数据库级权限相同。

议题

- DB2 安全性概述
- 认证
- 授权
- 角色
- PUBLIC 组
- 通过视图进行细粒化访问
- 基于标号的访问控制
- 扩展的安全性（仅限 Windows）
- 可信上下文

PUBLIC 组

在 DB2 中自动定义的一个特殊组
- 任何能够由操作系统 / 网络认证服务识别的用户 id 均属于 PUBLIC 组

在默认情况下，下述权限 （及其他权限）被授予 PUBLIC 组：
- CONNECT
- CREATE TAB
- IMPLICIT_SCHEMA
- BINDADD

为了“锁定”你的系统，你可以：
- 执行 CREATE DATABASE 时使用 RESTRICTIVE 选项，或者撤销 PUBLIC 组的这些特权

PUBLIC 组是 DB2 中的一个特殊组，其每名成员都是来自操作系统的用户。

比如，假如我现在正在使用 Windows 或 Linux，而且我创建了一个新的操作系统用户，则该新的操作系统用户将立即成为 PUBLIC 组的一部分。

PUBLIC 组在默认情况下具有多项特权：

CONNECT, CREATE TABLE, IMPLICIT_SCHEMA, "BINDADD", 等等。

这意味着什么？如果我现在进入 Windows 或 Linux 创建一个用户，该用户就可以接入数据库。

PUBLIC 组

在 DB2 中自动定义的一个特殊组
任何能够由操作系统 / 网络认证服务识别的用户 id 均属于 PUBLIC 组

在默认情况下，下述权限 （及其他权限）被授予 PUBLIC 组：
CONNECT
CREATE TAB
IMPLICIT_SCHEMA
BINDADD

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

为了“锁定”你的系统，你可以：
执行 CREATE DATABASE 时使用 RESTRICTIVE 选项，或者
撤销 PUBLIC 组的这些特权

如果你不打算向 PUBLIC 组授予任何此类特权，你可以如图所示利用这些 revoke 语句撤销这些特权，或者利用 RESTRICTIVE 选项创建数据库。

（演示）

议题

- DB2 安全性概述
- 认证
- 授权
- 角色
- PUBLIC 组
- 通过视图进行细粒化访问
- 基于标号的访问控制
- 扩展的安全性（仅限 Windows）
- 可信上下文

我们现在来看看视图，看看它们如何帮助实现安全性。

通过视图进行细粒化访问

能够让多个用户看到相同数据的不同展示

很适合于简单的安全性策略，但在大型环境中管理起来很复杂

程序：

- 创建一个视图（基本表中数据的子集）
- 授权用户访问该视图
- 撤销用户对基本表的访问权

EMPLOYEE

ID	NAME	AGE	PHONE	SALARY
12	Peter	30	99999	10000
3	Mary	23	88888	15000

EMP_VIEW

ID	NAME	AGE	PHONE
12	Peter	30	99999
3	Mary	23	88888

```
CREATE VIEW EMP_VIEW AS (
  SELECT ID, NAME, AGE,PHONE
    FROM EMPLOYEE);
```


利用视图，你可以提供对数据的细粒化访问权；如本幻灯片所示，视图非常易于建立。

在本例中，有一个表，其中有五列，最后一列含有 SALARY（工资）信息。由于这是敏感信息，你不希望让所有的用户都看到它，因此，你创建了视图 EMP_VIEW，其中隐去了该列。

然后，你向用户授予对该视图的访问权，并撤销用户对基本表的访问权。

议题

- DB2 安全性概述
- 认证
- 授权
- 角色
- PUBLIC 组
- 基于标签的细粒度访问控制
- 基于标号的访问控制
- 扩展的安全性（仅限 Windows）
- 可信上下文

基于标号的访问控制 (LBAC)

```
SELECT * FROM EMP
  WHERE
    SALARY >= 50000
```

No LBAC	SEC=254	SEC=100	SEC=50	ID	SALARY
				255	60000
				100	50000
				50	70000
				50	45000
				60	30000
				250	56000
				102	82000
				100	54000
				75	33000
				253	46000
				90	83000
				200	78000

基于标号的访问控制在行及列的层次上提供了细粒化的安全性。它使用了一个既与用户会话相关联，也与数据行或列相关联的标号，用于授予对表中数据的访问权。该图示出了LBAC的工作方式。

在本图中，表 ***EMP*** 有一列，即 ***SALARY***，还有一个内部列***ID***，它含有给定行的标号。图中的其他列仅仅是为了解释之用。如果执行图中所示的查询，根据该用户所拥有的标号，它能够看到不同的行。标题为 'No LBAC' 的列代表了不实施LBAC时将会被查询到的列。如你所见，所有大于等于 50,000的salary行都会被查询到。

现在假设发出该查询的用户拥有安全标号100。你会看到，在本例中，左数第三列标识的行被查询到。在此例中，DB2将寻找salary大于等于50,000的行，然后再检查该行的安全标号。例如，第一行的salary为 60000，标号 ID为 255。由于该用户的标号 ID为 100，它小于255，于是该用户就无法看到这行，因此，查询输出中就不会返回它。

LBAC安全性需要具有SECADM安全性的安全性管理员来实施。

LBAC功能未在DB2 Express-C中提供。

议题

- DB2 安全性概述
- 认证
- 授权
- 角色
- PUBLIC 组
- 基于标签的细粒度访问控制
- 基于标号的访问控制
- 扩展的安全性（仅限 Windows）
- 可信上下文

扩展的安全性（仅限 Windows ）

（通过操作系统）控制对 DB2 系统文件的访问权

DB2ADMNS Windows 组

本组以及本地管理员可以通过操作系统获得对所有 DB2 对象的全部访问权。

DB2USERS Windows 组

本组可以通过操作系统获得对所有 DB2 对象的“读”及“执行”访问权。

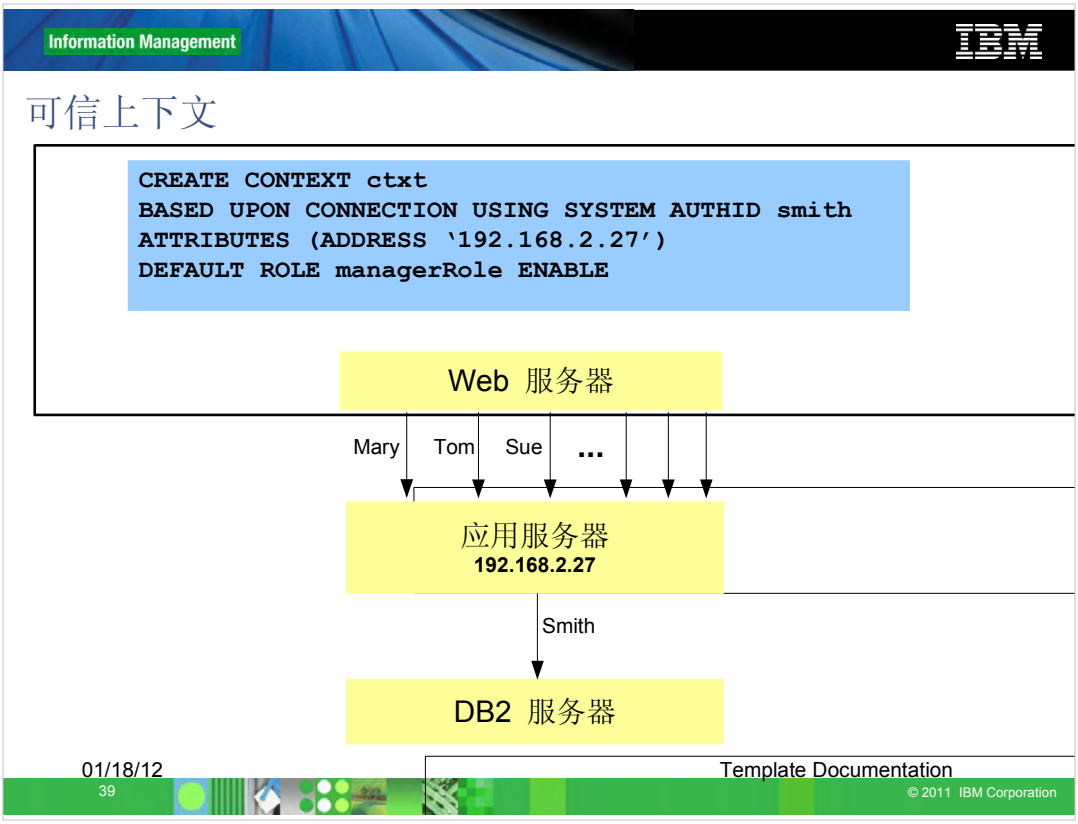
扩展的安全性 (Extended security) 仅适用于 Windows，它可以阻止不属于 DB2USERS 或 DB2ADMNS windows 组的用户访问 DB2 文件。这样一来，就不是所有的 Windows 用户都能够从操作系统中访问 DB2 系统文件了。

你可以把所有的 DB2 管理员放到 DB2ADMNS 组中，他们将能够对 DB2 进行几乎所有的操作，而且他们甚至可以提供操作系统来访问 DB2 系统文件。

在 DB2USERS 组中，你可以放入那些可以通过操作系统使用 DB2 对象的用户。

议题

- DB2 安全性概述
- 认证
- 授权
- 角色
- PUBLIC 组
- 通过视图进行细粒化访问
- 基于标号的访问控制
- 扩展的安全性（仅限 Windows）
- 可信上下文



可信上下文适用于 3 层次环境，这里你拥有一台 Web 服务器、一台应用服务器以及一台数据库服务器。

它试图解决确定谁正在数据库服务器中干什么的问题。这里考虑的问题是，虽然可能有许多用户利用他们自己的用户 ID （例如 Mary, Tom, Sue, 等等）从 Web 服务器进入应用服务器，但从应用服务器到数据库服务器的连接通常只有一个或者为数不多的几个具有固定用户 ID （例如 Smith ）的连接来进行。这么做有性能方面的原因，因为频繁地与数据库之间连接 / 断开可能代价不菲。但是，这会产生一个安全性问题：我们无法“一对一”地映射出哪个 Web 用户在数据库服务器进行了哪项操作。例如，是用户“ Mary” 删除了某项给定的记录吗？在数据库服务器中，我们无法确定这一点，因为所有的审计都将指向用户“ Smith” 。

利用可信上下文，你无法把应用服务器与数据库服务器之间的连接分解至每名用户，但是，你可以通过这些上下文（根据 IP 地址、域名等）来确定进行了该操作的用户。你也可以把角色指定给可信上下文。

在本例中，在位于 IP 地址为 192.168.2.27 的应用服务器之间创建了一个可信上下文，利用授权“ Smith” 从应用服务器接入 DB2 服务器。因此，当有一个连接来自用户 Smith 、来自此 IP 地址时，可信上下文就会产生，此连接就具有“ managerrole” 的角色。既然本上下文是“可信的”，在应用服务器上，你就可以编码（根据具体语言）出一种方式在应用服务器层次上切换用户。比如，从 Web 服务器上，你以“ mary” 的身份访问应用服务器，然后在应用服务器上，你再置入一些逻辑，使得如果它是“ mary” 的话（她恰好是一名经理），就针对可信上下文 ctxt 把“ smith” 切换至该用户 ID ；因此，在数据库服务器上，“ mary” 现在就成为进行操作的用户。

可信上下文

有用的 3 层环境

“ 可信上下文”：

DB 与应用之间可信的关系
- 切换当前用户 ID
- 通过角色继承获得额外特权

根据所使用的连接属性确定的关系
- IP 地址，
- 域名，
- 授权 ID ，
- 数据加密

因此，可信上下文适用于 3 层环境。

在应用服务器与数据库服务器建立了上下文。不涉及 Web 服务器。

有多种属性可以用于识别可信上下文，例如 IP 地址、域名、授权 ID 以及数据加密。

例如，如果同一个授权 ID 从某个不同的机器利用不同于可信上下文的某个 IP 地址接入，它可能无法访问相同的数据。

如果你对本课程所讨论的内容有什么技术方面的问题的话，可以利用 db2university.com 课程 AA001EN 中的论坛。 同学、教员以及 IBM 人员可以帮助你！

备份数据对于企业至关重要。信息丢失可能导致重大危机，或者更糟糕的是，导致企业倒闭。

可能遇到的常见问题是：

- 系统中断（电源故障，硬件故障）
- 事务故障（用户可能偶然地损坏数据库）
- 介质故障（磁盘驱动器变得不稳定）
- 灾难（数据库设施由于火灾、洪水或其他灾难而受到破坏）

DB2 备份和恢复方法旨在帮助你保证信息的安全！

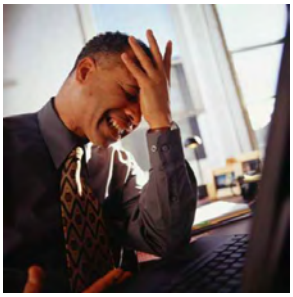
议题

备份和恢复概述

数据库日志记录

备份

恢复



支持性阅读材料和视频

阅读材料

DB2 Express-C 入门电子书

第 11 章：备份和恢复

视频

db2university.com 课程 AA001EN

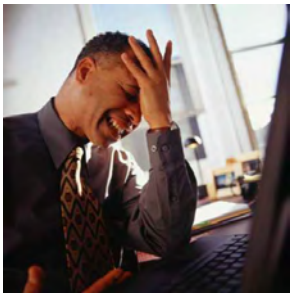
第 10 课：备份和恢复

如果你需要关于本主题的详细信息，请参阅这份支持材料：

- “DB2 Express-C 入门第 3 版”第 11 章：备份和恢复，以及
- db2university.com 课程 AA001EN DB2 学术培训第 10 课

议题

- 备份和恢复概述
- 数据库日志记录
- 备份
- 恢复

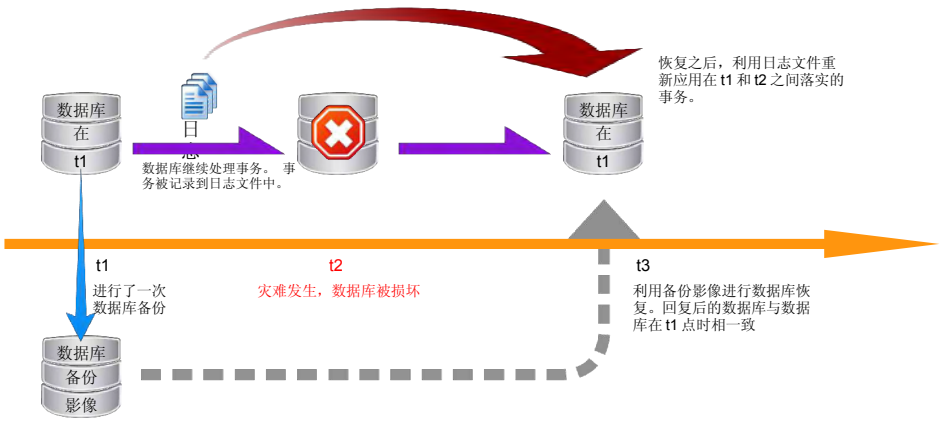


备份和恢复概述

在 t1，进行了数据库备份操作

在 t2，发生了一次破坏数据库的问题

在 t3，所有已提交数据均被恢复



在本图中，我们来简要讨论备份、恢复及前滚命令以及事务日志的概念。

在所示的时间线上，我们假设在时间 t1 上，你发出了备份命令来制作你的数据库的复制品。然后用户继续工作，而且关于其活动的信息随时都被记录到事务日志中。在时间 t2，出现一次故障，它破坏了数据库，因此在时间 t3，你发出恢复命令，这将使用你在时间 t1 获得的备份影像。这将能够让你找回你的数据库，但备份影像中并不包含时间 t1 与 t2 之间的信息，因此，你需要就这个期间应用事务日志。利用前滚命令，你可以对已经恢复的数据库应用一落实的事务，并 * 几乎 * 复原至崩溃发生前的状态。我们在下面的几张幻灯片中将详细讨论这里的每一个概念。

议题

- 备份和恢复概述
- 数据库日志记录
- 备份
- 恢复

数据库日志记录

日志跟踪对数据库对象和数据进行的修改。

它们被用于恢复：如果发生崩溃，日志就用于回放 / 重做已提交的事务，并撤销未提交的事务。

可以存储在文件中或者裸设备中

对于 DB2 中的常规表，日志记录总处于 ON 状态

- 可以把某些表或列标记为 NOT LOGGED （不记录日志）
- 可以声明和使用 USER 临时表，这些表可以不记录到日志中

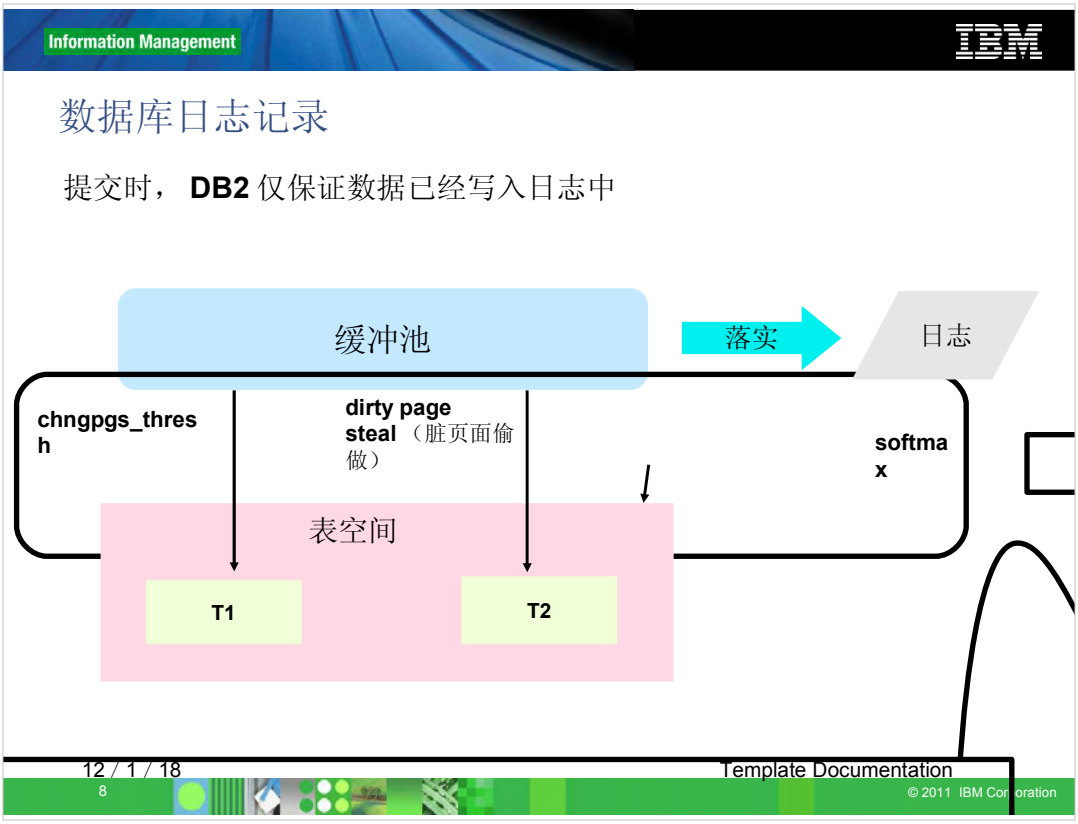
如果你在使用文本编辑器，每次当你打算确保你的文档得到保存时，你就点击保存按钮。在数据库中，COMMIT（落实）语句就是起这个作用的。每次执行COMMIT语句时，你都可以保证，无论对数据做了什么修改， 它们都将保存在某个地方。

类似地，当你使用某个文本文档时，有时你会在右下角看到一条简短消息提示：“正在自动保存”。在数据库中，也会发生这种情况，因为你的数据进行的任何操作，例如UPDATE、INSERT或DELETE，都将在你操作之时保存在某个地方。

前面说的“某个地方”指的是数据库日志。数据库日志被保存在磁盘上，被用来记录事务行动。如果发生系统或数据库崩溃的情况，日志就可以用来在恢复过程中重放并重做已落实的事务。

数据库事务日志跟踪用户对数据库进行的所有操作。例如，日志中将包含有关UPDATE语句的信息，其中包括旧值和新值。日志中将含有INSERT语句的信息以及所插入的值；日志中还含有关于COMMIT或ROLLBACK等语句的信息。

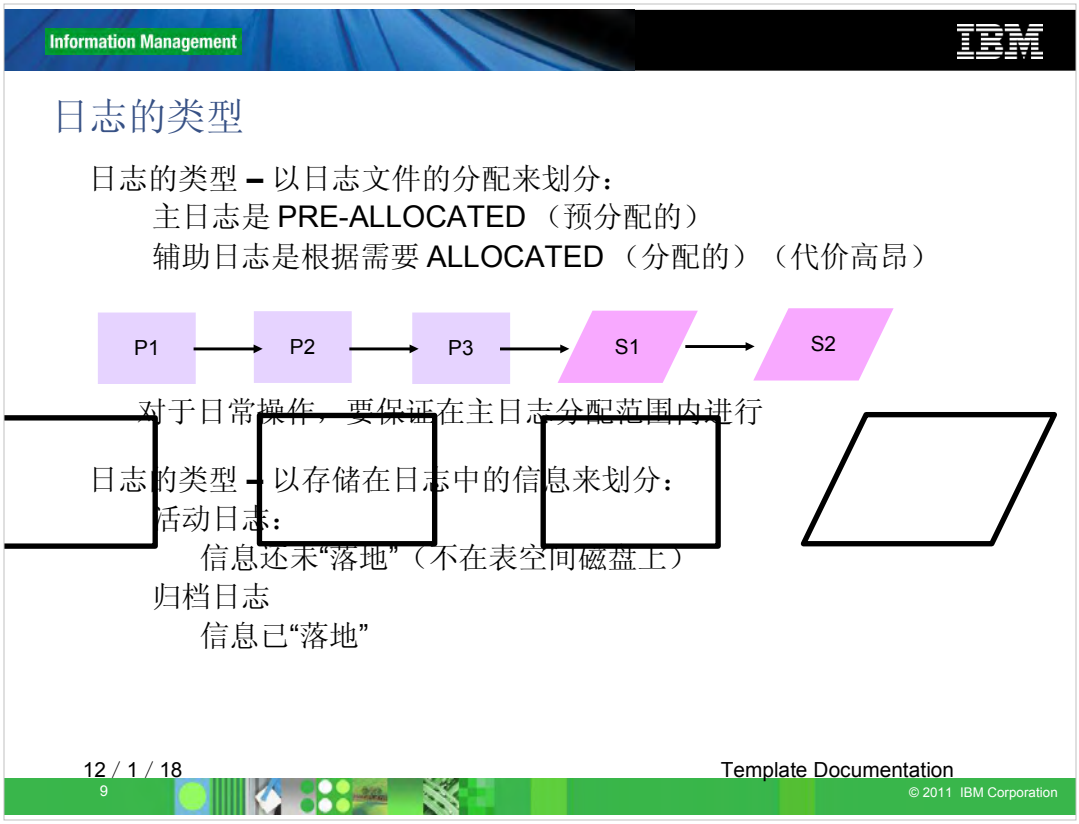
在DB2中，日志在默认时处于ON状态，它不能被关闭。你可以决定不对有些对象或操作进行日志记录。例如，你可以决定不对大对象的修改做日志记录。你也可能决定不对临时表的修改做日志记录。



在本图中，我们看到一张表和一些日志。它们都驻留在磁盘上，虽然我们建议不要把它们存放到同一个磁盘上，因为日志是用来恢复你的数据的，因此，如果把你的数据以及用来恢复你的数据的数据都放到同一个磁盘上，那么，如果磁盘发生故障，你就失去了一切。

在进行UPDATE操作时，待处理的行的页面将被装入缓冲池（内存）中。升级变更在缓冲池中完成，而旧值和新值将被存储在日志文件中，有时是立即存储，有时是在日志缓冲区被充满之后在存储。如果在UPDATE之后发出了COMMIT，则旧值和新值都将立即被存储到日志文件中。这个过程不断地为在数据库进行的众多其他SQL操作所重复。只有在某些条件得到满足之后，例如达到了CHNGPGS_THRES参数中所规定的变更页面阈值，缓冲池中的页面才会被“外在化”及写入表空间磁盘中。CHNGPGS_THRES参数规定了含有“脏”页面（即包含变更的页面）的缓冲池的比例。

从性能的角度看，对每个COMMIT操作进行两次写操作并没有什么意义：一次写入日志，另一次写入表空间磁盘；这就是为什么数据“外在化”到表空间磁盘只会发生在达到了CHNGPGS_THRES等参数的阈值之后。



根据文件位置，有两类日志：

主日志：这是预先分配的日志，其数量由 LOGPRIMARY数据库配置参数来决定。

辅助日志：它们是根据需要由 DB2动态分配的。辅助日志的最大数量由数据库配置参数 LOGSECOND来设置。动态分配日志需要付出很高的代价；因此，在日常操作中，应当保持在主日志分配范围内。当与数据库之间的所有连接被终止之后，辅助日志文件将会被删除。

根据日志中存储的信息划分的日志类型：

活动日志

尚未被落实或回滚的事务

在线归档日志

已经在活动日志目录中落实的和外在化的日志

离线归档日志

已经在独立的资料库中落实的和外在化的日志

日志记录方式的类型

循环日志

用于非生产系统

日志可以归档，也可以被覆盖

如果外在化到表空间的信息是错误的，该怎么办？（人为错误）。没有日志来重做什么事情！

归档日志

用于生产系统

日志不会被删除。

有些是在线存储的（与活动日志存储在一起），另一些以离线方式存储在外在介质中。

有两种日志记录方式：

循环日志（对于非生产系统）是其信息已经被落实的和外在化的信息，这种日志可以被覆盖；归档日志（用于生产系统）是永远不会被覆盖的日志。下面将详细讨论每一种日志记录方式。

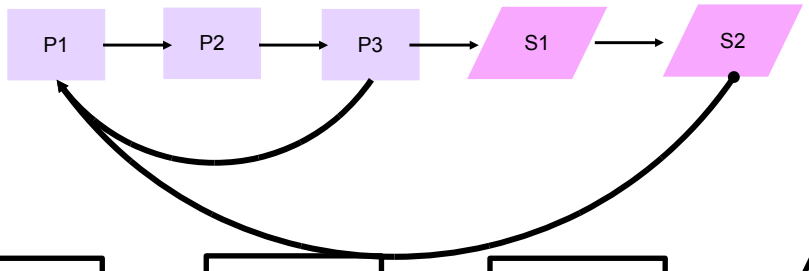
循环日志

日志的默认类型

当这些日志的内容被外在化之后，日志就被覆盖；进行崩溃恢复时不再需要它们

如果某项长事务用尽了主日志和辅助日志，就会产生日志满条件，并会返回 SQL0964C 错误消息

无法进行前滚恢复



循环日志是默认的方式，它在LOGARCHMETH1和LOGARCHMETH2两个数据库配置参数都被设置为OFF时启用。这些参数指明了用来归档日志的方法，但如果你把它们都关闭，则意味着你不打算归档这些日志，这就是循环日志的工作原理。此图示出了循环日志方式：

这里有三个主日志，因此我们可以假定 LOGPRIMARY 参数的值是3。为简化起见，假设本例中只处理一项事务。在处理该事务时，日志文件P1首先被填满，然后是P2。如果发生了落实而且该信息随后被外在化至表空间磁盘，则P1和P2都可能被覆盖，因为这些信息不会再用于崩溃恢复了（崩溃恢复将在后面详细讨论）。另一方面，如果事务太长，它用尽了 P1、P2 和 P3，而且仍然需要更多的日志空间，因为该事务还没有被落实或外在化，那么，就会动态分配一个辅助日志（图中的 S1）。如果事务继续进行，则继续分配更多的辅助日志，直至分配了最大数量的LOGSECOND日志。如果仍然需要更多的日志，则会向用户返回一条错误消息，指出达到了日志满条件，而且该事务会被回滚。作为另一种方式，你可以利用BLK_LOG_DSK_FUL配置参数来配置 DB2，以便每 5 分钟持续写入日志一次，同时让某些事务挂起。这将为DBA提供某些时间来寻找新空间，以便让事务继续进行。

循环日志能够让你从崩溃中恢复，但是，如果打算把你的数据恢复至某个给定的时间点，则最近的可用时间点将是你进行最后一次离线备份的时间。

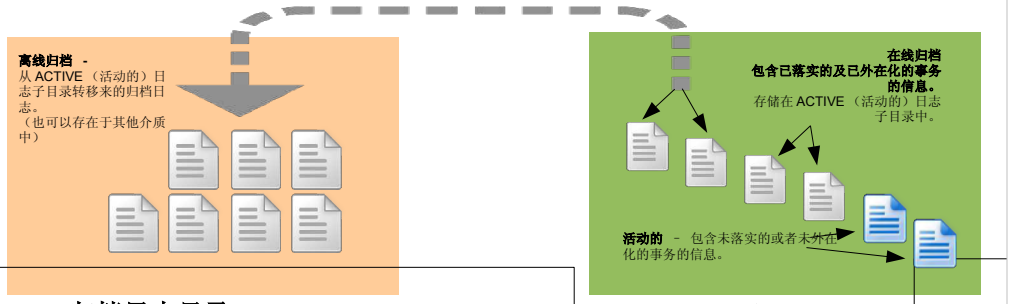
归档日志

启用 LOGARCHMETH1 db cfg 参数

一旦启用，你就会被要求进行一次离线恢复

日志文件不会被删除 – 保持在线或离线

可以进行前滚恢复和在线备份



在归档日志（也称为日志保持记录方式）中，日志文件不会被覆盖，而是以在线或离线方式得到保持。在线归档日志与活动日志保持到一起，它们仍然需要用于崩溃恢复。离线归档日志则被转移到其他介质上，例如磁带，这可用利用USEREXIT例程、 Tivoli Storage Manager或者其他第三方归档产品来完成。

为了启用归档日志，可用把数据库配置参数LOGARCHMETH1或 LOGARCHMETH2（或两者）设置为OFF以外的值。启用它的另一种方法是把LOGRETAIN配置参数设置为RECOVERY。这将自动导致 LOGARCHMETH1被设置为 LOGRETAIN。但是，LOGRETAIN参数已经被淘汰了，它主要是为了与旧版本的DB2相兼容。

归档日志通常用于生产系统中；由于日志得到保持，这能够让数据库恢复至最早的日志文件那个时间点。利用归档日志，DBA可用从人为导致的错误中恢复。例如，如果某个系统用户偶然开始执行一项持续数天的错误事务，那么，当发现该问题时，DBA可以把系统恢复至该问题被引入之前的时间点。但是，这可能需要一些手工操纵才能使事务得以正确返回。

在前滚恢复及在线恢复中，需要归档日志。该图示出了归档日志的流程。

无限日志

提供了无限的活动日志空间

- 通过开启归档日志并把 LOGSECOND 设置为 -1 来启用

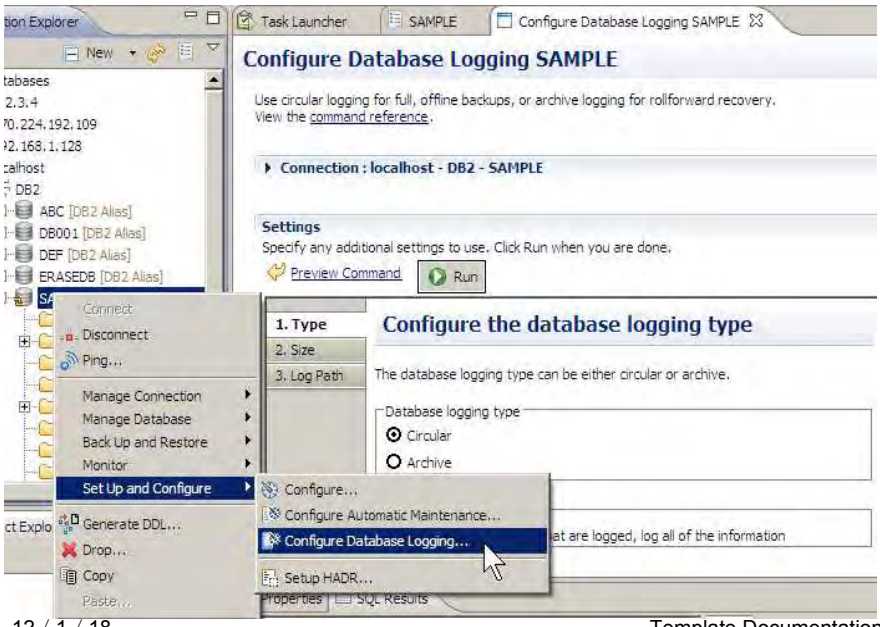
LOGSECOND 指出有多少辅助日志文件可以被分配。如果设置为 **-1**，则表明允许无限日志

不利于提高性能

- 辅助日志被不停地分配
- 不利于回滚和崩溃恢复

如果你把 LOGSECOND 设置为 -1，可以实现无限日志；但是，不建议你这样做，因为你可能会耗尽文件系统空间，而且这对于崩溃时的性能也不利。

在 Data Studio 中配置数据库日志



要想在Data Studio中设置和配置日志记录方式，只需右击数据库名称，并选择 “设置和配置 -> 配置数据库日志记录”

议题

- 备份和恢复概述
- 数据库日志记录
- 备份
- 恢复

数据库备份

数据库或表空间的复制品

用户数据、 DB2 目录、所有的控制文件（例如缓冲池文件、表空间文件、数据库配置文件）

备份模式：

离线备份

不允许其他应用或流程来访问数据库

在使用循环日志时的唯一选择

在线备份

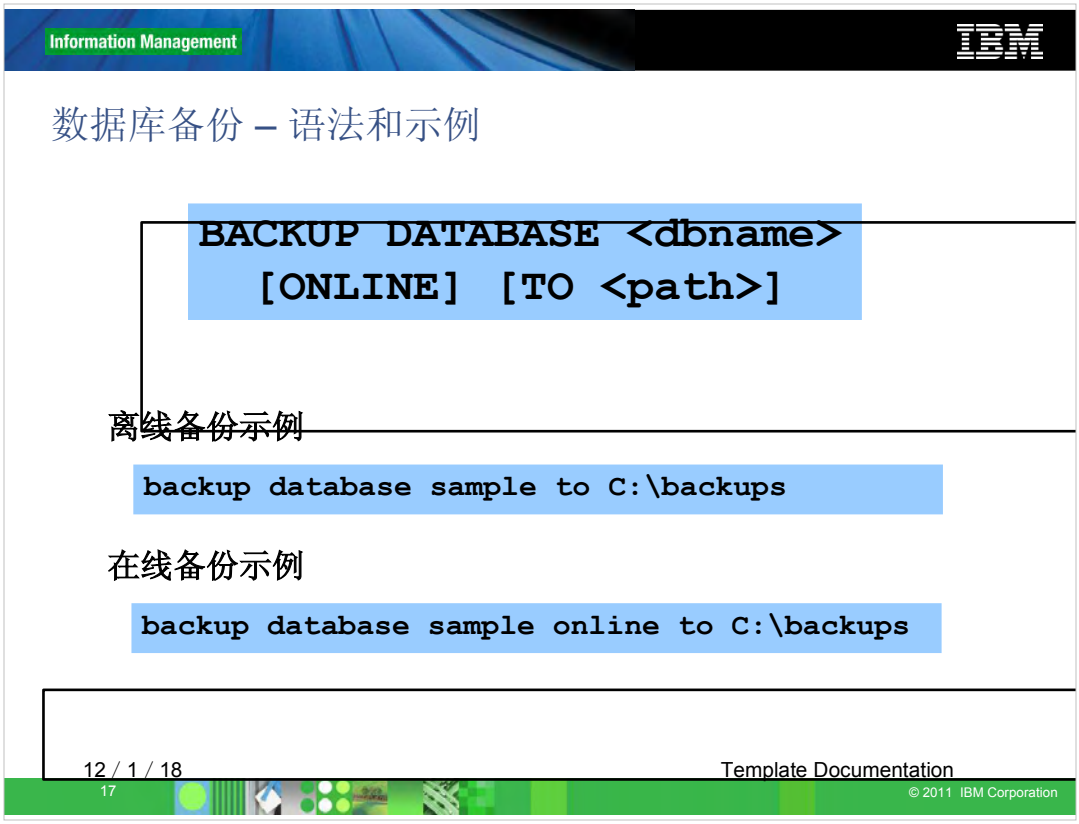
在进行备份的时候，允许其他应用或流程来访问数据库

DB2备份命令能够让你在执行此命令时获得数据库的快照。

多数命令和实用工具都可以在线或离线进行。

在线意味着，当你执行你的命令时，其他用户可以接入并对数据库进行操作。

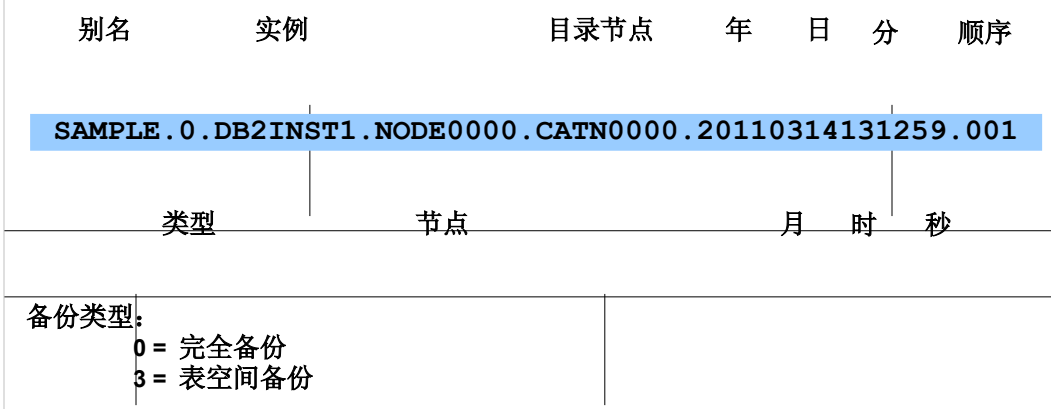
离线意味着，当你执行你的操作时，其他用户不能接入数据库。为了允许在线操作，可以向命令行语法中条件关键字ONLINE，否则的话，在默认情况下，命令会认为你是在离线执行。



如前所述，向命令行添加关键字 ONLINE 能够使备份在线进行，否则的话，在默认情况下，命令会认为你是在离线执行。

如果你进行ONLINE（在线）备份，这意味着，在用户对数据库修改的同时，你在执行备份操作。因此，备份影像可能不包含截至最后时刻的全部信息。所以，一个比较好的做法是使用子句 “INCLUDE LOGS”（包含日志）来保证在线备份之时所使用的日志被包含到备份影像中，这样的话，如果你需要从该影像中恢复，你会得到所有的东西。

数据库备份 – 文件命名规约



在本幻灯片中，我们有一个为数据库备份影像生成的文件名。换言之，在你执行备份命令之后，这就是所创建的文件的名称。

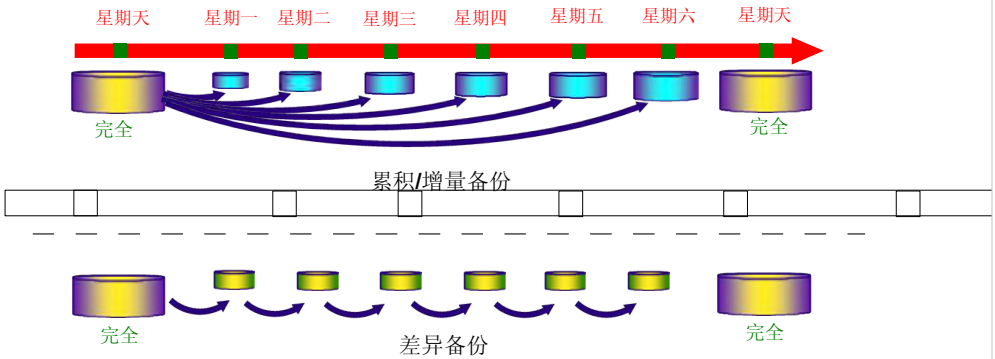
请注意，可以对整个数据库进行备份，或者仅仅对表空间进行备份。在备份影像的命名规约中，“类型”指出了这是一个数据库备份还是一个表空间备份。如果该值为零，它是一次（完整）数据库备份；如果该值为3，它是一次表空间备份。

“节点”固定为NODE0000，“目录节点”固定为CATN0000。只有当使用DPF（数据分区功能）时，这些值才会改变。

从“年”到“秒”的时间戳很重要，当你打算从备份影像中恢复时，应当记住该时间戳，因为它们唯一地区分出备份。

增量备份

适合于大型数据库；由于仅备份累积的 / 差异的变化，可以实现大量节省。



我们现在来谈谈“增量备份”。增量备份有两种类型：增量备份（也称为累积备份） - 对最近一次成功的完整备份操作以来改变的所有数据库数据进行备份。

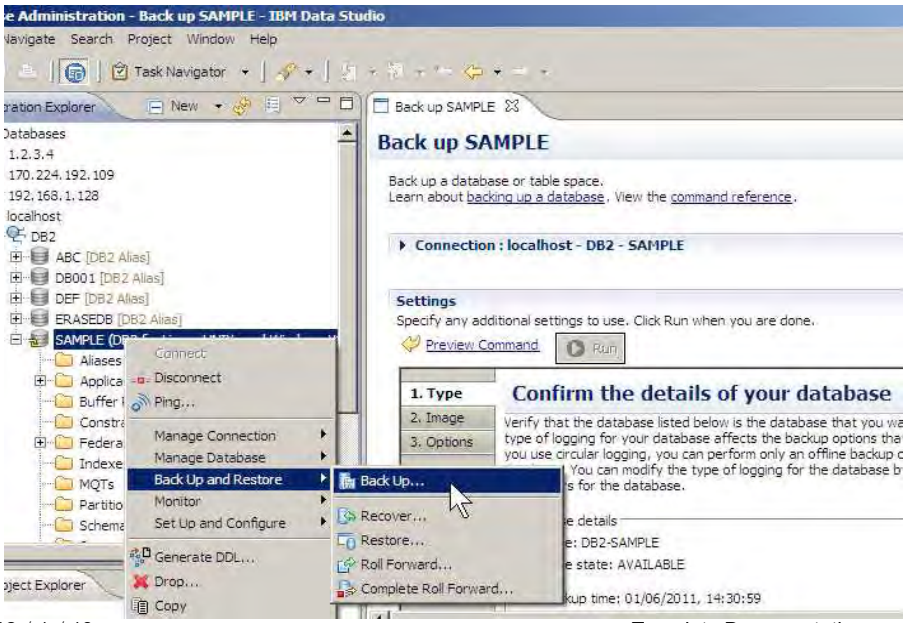
差异备份（也称为增量差异备份） - 对最后一次成功的备份以来改变的所有数据库数据进行备份，无论最后一次备份是完整备份、增量备份或者差异备份。

在本图上半部分，你在星期天进行了一次完整备份，然后在星期一，你进行了一次累积备份，它包含从星期天到星期一之间的所有改变。然后在星期二，你进行了另一次累积备份。该备份影像包含从星期天到星期二之间的所有改变（它是累积备份，因此，它也包含从星期天到星期一之间的改变），等等。在本例中，如果在星期五进行累积备份之前发生了一次崩溃，然后需要恢复到最近的时间点，那么，你就必须首先从星期天进行的完整备份中恢复，然后再恢复星期四进行的累积备份。

在图中的下半部分，是差异备份：假设你在星期天进行了一次完整备份。然后在星期一，你进行了一次差异备份，它包含从星期天到星期一之间的改变。然后在星期二，你进行了另一次差异备份，它将恢复仅仅从星期一到星期二之间的改变，等等。因此，在本例中，如果在星期五进行差异备份之前发生了一次崩溃，然后需要恢复到最近的时间点，那么，你就必须首先从星期天进行的完整备份中恢复，然后再恢复星期一、星期二、星期三及星期四进行的每一次差异备份。

幸运的是， DB2 能够跟踪你已经进行的不同类型的备份历史，并可以智能地决定应当使用哪些影像来恢复你的数据库。要想顺利地进行增量备份，你需要把 TRACKMOD 数据库配置参数设置为 ON 。

从 Data Studio 中进行数据库备份



在 Data Studio 中，可以很容易进行备份。只需右击数据库名称，选择“备份和恢复”，然后选择“备份”。

议题

- 备份和恢复概述
- 数据库日志记录
- 备份
- 恢复

数据库恢复

数据库恢复将从备份和日志中重建数据库或表空间

使用 restore （恢复）和 rollforward （前滚）命令

数据库恢复是指从备份及/或日志中恢复你的数据库。如果你只是从备份中恢复，你将重建数据库，使其达到在备份进行时的状态。

如果在备份之前启用了归档日志方式的话，你不但可以使用备份影像进行恢复，还可以从日志中恢复。我们在下一节将会看到，前滚恢复能够让你从备份中恢复，然后再应用（前滚）日志至日志末尾，或者至某一指定的时间点。

请注意，本节经常使用“recovery”（恢复）一词，恢复所使用的命令是 RESTORE。

数据库恢复的类型

崩溃恢复

防止数据库发生不一致的情况（电源失效）

版本恢复

从备份中恢复数据库
数据库将返回至备份中保存的状态
在备份之后进行的任何改变都将丢失

前滚恢复

需要启用归档日志
遍历日志，以便重新应用在备份之后所做的改变
可以前滚至日志的末尾，或者前滚至某个特定的时间点
数据损失最少

崩溃恢复或重启恢复

假设你在使用一部台式机，它正在DB2数据库中运行一些重要的事务。突然，发生了断电，或者有些人意外地拔掉了电源线：这时数据库会出现什么情况呢？

下一次你重启计算机并启动DB2时，崩溃恢复将自动进行。在崩溃恢复中，DB2将自动运行命令RESTART DATABASE，并根据活动日志读取以及重做/撤销相关的事务。当这个命令完成时，你可以确保你的数据库处于一致的状态，就是说，已经落实的东西都将得以挽回，未落实的东西都被回滚。

版本或影像恢复

此类恢复意味着，你只从备份影像中恢复；因此，你的数据库将被置于其被备份时的状态。备份之后在数据库中进行的任何事务都将丢失。

前滚恢复

利用此类恢复，你不仅可以使用RESTORE命令从备份影像中恢复，而且你还可以运行ROLLFORWARD命令在备份的基础上应用日志，这样你就能够恢复至某个指定的时间点。此类恢复能够把数据损失降低到最低程度。

数据库恢复 – Restore 命令语法

```
RESTORE DATABASE <dbname>
  [ONLINE][FROM <path>]
  [TAKEN AT <timestamp>]
  [WITHOUT PROMPTING]
```

离线恢复示例

SAMPLE.0.DB2INST.NODE0000.CATN0000.*20110718131210*.001

```
restore database sample
   from C:\backups
   taken at 20110718131210
```


这是RESTORE命令的语法。

利用RESTORE命令可以从备份影像中恢复数据库。在该语法中，“Online”用于指定该操作将在线进行，这意味着用户也可以同时接入数据库。在线恢复只适用于表空间恢复。

“Taken At” 用于指定你从备份影像中获得的时间戳。

“Without prompting”用于指定在恢复操作过程中不必发出提示。例如，假设你在一个现有的数据库基础上进行恢复。RESTORE命令将提示你是否打算覆盖此现有的数据库。如果你使用“Without prompting”，该恢复命令将会径直进行而覆盖它。

此幻灯片的底部是一个恢复数据库的例子。请注意，TAKEN AT子句用于规定从哪个备份影像中进行恢复。如例所示以及早前的解释，备份影像的名称中包含了备份进行时的时间戳。

表空间备份和恢复

允许用户备份数据库的子集

只有在启用了归档日志时才能够使用

可以规定多个表空间

可以从数据库备份中或者表空间备份中恢复表空间

使用关键字 **TABLESPACE** 来指定表空间

示例：在线备份表空间 **'tblsp1'**

```
backup database sample
   tablespace(tblsp1)online to C:\backups

restore database sample
   tablespace(tblsp1)online from C:\backups
   without prompting
```


本幻灯片表明，备份操作和恢复操作都可以在表空间层次上进行。

请注意，当你恢复和前滚表空间时，你必须进行至某个能够保证数据一致性的最低时间点(PIT)。为了找到最低 PIT，可以使用命令：GET SNAPSHOT FOR TABLESPACES ON <database>

并查找 “最低恢复时间”项。

你也可以使用 “列示表空间显示详情”(list tablespaces show detail)，但如果未启用归档日志方式的话，将不会显示最低时间点恢复的选项。

在进行备份/恢复时可以指定多个表空间。在命令中一定要使用关键字TABLESPACE，后跟表空间名称。

还要注意，表空间既可以从数据库备份中恢复，也可以从表空间备份中恢复。

本幻灯片的底部是一个示例，先备份表空间，然后恢复表空间，两者都在线进行。

利用备份和恢复还能够 ...

- 备份压缩
- 从备份镜像克隆数据库并改变容器（重定向的恢复）
- 在现有的数据库上进行恢复
- 恢复被删除的表
- 等等 ...

最后一张幻灯片示出了备份和恢复中可能出现的其他事情。

备份影像可以被压缩，这可以接受空间。

可以通过备份/恢复操作来克隆数据库；但是，如果在 Linux中进行备份，则不能在Windows中恢复，反之亦然。某些情况下，在UNIX中，可以在一种UNIX中备份数据库，然后在另一种 UNIX中恢复它。如果你需要把一个数据库从Windows系统中克隆到Linux系统中（或者相反），你需要使用 db2look 及 db2move工具。

此外，如果你打算在同样的操作系统上备份和恢复数据库，但磁盘布局不同，会发生什么情况？例如，如果你在一套具有5个磁盘的系统上进行备份，但打算在另一套具有3个磁盘的系统上进行恢复，会出现什么情况？备份影像中包含了表空间存储在什么地方的信息。如果你的恢复目的地系统中有不同的磁盘布局，你需要运行重定向恢复。利用重定向恢复，你可以手工（或者使用脚本）为你的表空间指定容器。

你也可以在现有的数据库上进行恢复，虽然你会收到一条警告消息。

也可以恢复被删除的表，如果你开启了该支持的话，但这会导致付出更多的日志记录及性能代价。

请注意，此类恢复和前滚命令之外，还有另一个称为 “Recover” 的命令，它在一条命令中进行 “Restore” 加 “Roll forward”。

如果你对本课程所讨论的内容有什么技术方面的问题的话，可以利用 db2university.com 课程 AA001EN 中的论坛。 同学、教员以及 IBM 人员可以帮助你！

Information Management

IBM

DB2 pureXML

IBM 信息管理云计算能力中心
IBM （加拿大）研究院

议题

- 概述
- 插入 XML 数据
- XPath
- XQuery
- 利用 SQL/XML 查询 XML 数据
- 利用 XQuery 查询 XML 数据
- 对 XML 进行更新和删除操作
- XML 索引
- XML 模式验证
- 其他 XML 支持

支持性阅读材料和视频

阅读材料

- DB2 Express-C 入门电子书
  - 第 15 章 : DB2 pureXML
- IBM Data Studio for DB2 入门电子书
  - 第 4 章 ： 创建 SQL 和 XQuery 脚本

视频

- db2university.com 课程 AA001EN
  - 第 11 课 ：DB2 pureXML

议题

- 概述
- 插入 XML 数据
- XPath
- XQuery
- 利用 SQL/XML 查询 XML 数据
- 利用 XQuery 查询 XML 数据
- 对 XML 进行更新和删除操作
- XML 索引
- XML 模式验证
- 其他 XML 支持

什么是 XML?

e*X*tensible *M*arkup *L*anguage（可扩展的标记语言）

XML 是一种用于描述数据的语言

一种层次数据模型

```
<book>
   <authors>
      <author  id="47">John Doe</author>
      <author  id="58">Peter Pan</author>
   </authors>
   <title>Database systems</title>
</book>
```

XML 的意思是 eXtensible Markup Language （可扩展的标记语言）。过去几年， XML 得到了快速流行和普及。为什么 XML 很重要呢？

- XML 构成了 **web 服务的基础，** 而 web 服务又是信息按需供应的基础。**信息按需供应，** 正像其名字所暗示的那样，就是在信息被请求之时能够提供出来。这可以通过按照“信息即服务”的方式提供出来。
- XML 也处于 **Web 2.0 技术的核心**。

XML 是这些概念的基础，没有 XML ，这些概念就很难实现。

XML 数据采用了一种**层次模型**，它非常适合于存储非结构化类型的信息。

XML 具有一系列具体的特征：

- **灵活的**：易于修改或改编
- **易于扩展**：你可以创建你自己的标签
- **自我描述**：XML 模式（它本身也是一个 XML 文档）描述了文档中所使用的规则和标签
- **可以转变为其他格式**：例如，转变为 HTML, XSLT
- **独立于平台或供应商**
- **易于共享**：易于与其他应用共享，因为它可以被存储为文本文档

哪些人使用 XML?

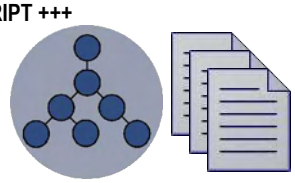
银行业
IFX, OFX, SWIFT, SPARCS, MISMO +++

生命科学
MIAME, MAGE, LSID, HL7, DICOM, CDIS, LAB, ADaM +++

零售业
IXRetail, UCCNET, EAN-UCC ePC Network +++

医疗保健行业
HL7, DICOM, SNOMED, LOINC, SCRIPT +++

电子行业
PIPs, RNIF, Business Directory, Open Access Standards +++



保险业
ACORD XML for P&C, Life +++

电信业
eTOM, NGOSS, 等等 Parlay Specification +++

金融市场 **FIX Protocol, FIXML, MDDL, RIXML, FpML +++**

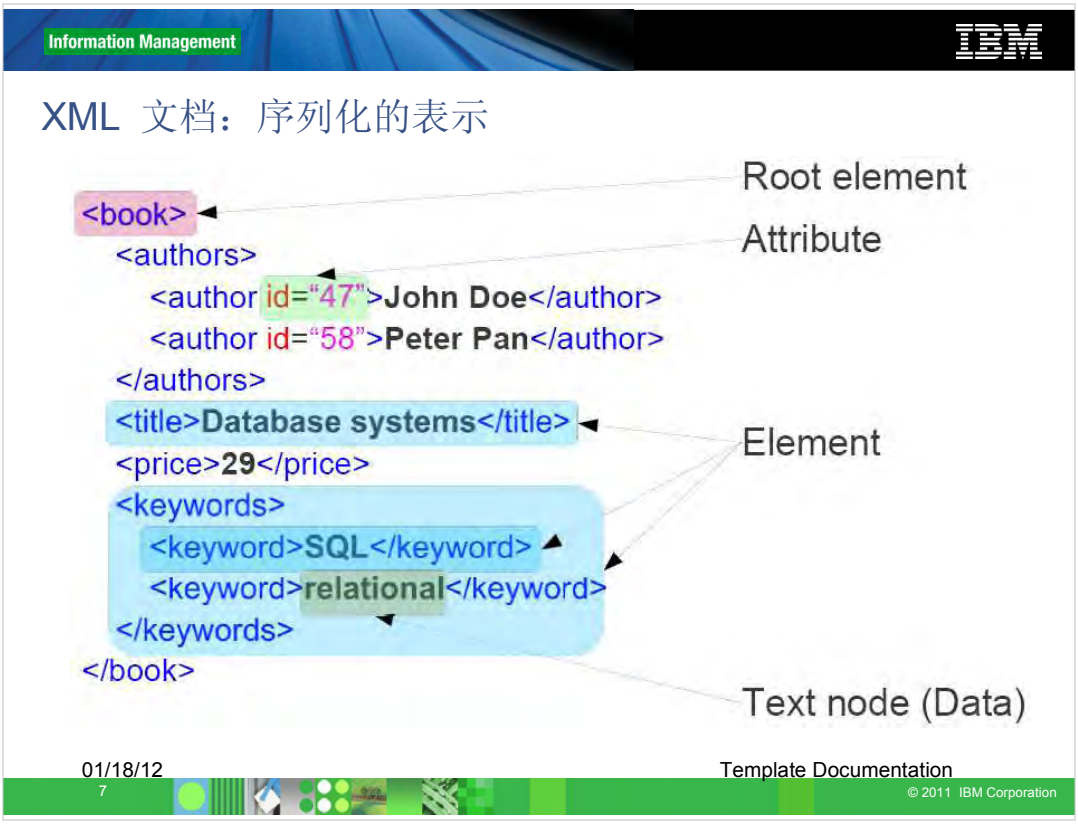
汽车行业
ebXML, 其他 B2B 标准

能源和公用事业
IEC Working Group 14 Multiple Standards CIM, Multispeak

交叉行业
PDES/STEPml SMPI Standards RFID, DOD XML+++

石化行业
Chemical eStandards CyberSecurity PDX Standard+++

XML 被许多不同的行业使用，例如银行业、医疗保健业，等等



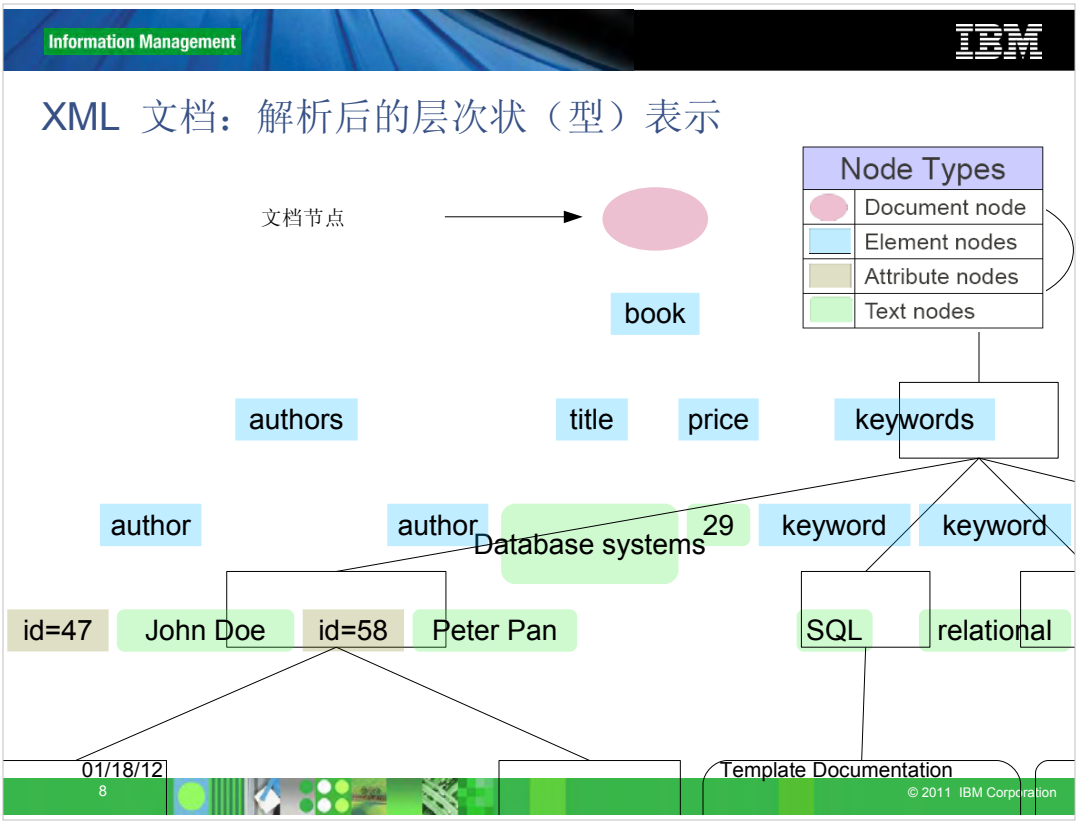
XML 是一种用于描述数据的语言。它由元素和属性等节点构成。

其成分包括：

- 元素
- 根元素
- 属性
- 文本节点

在本例中：

- 根元素 : <book>
- 元素 : <authors>, <author>, <title>, <price>, <keywords>, <keyword.
- 属性 : id="47", id="58"
- 文本节点 : John Doe, Peter Pan, Database systems, 29, SQL, relational



这只是另一个例子，它示出了 XML 可以具有的不同类型的节点：
- 根
- 元素
- 属性
- 文本

格式良好的与有效的 XML 文档

格式良好的 XML 文档是符合以下基本规则的文档：

它必须有一个，而且只有一个根元素
各个元素起始于一个起始标签，并结束于一个结束标签
一个元素可以包含其他元素、属性或文本节点
属性值必须包含在双引号中。而文本节点则无此必要。

有效的 XML 文档具备以下两项特征：

格式良好的 XML 文档
符合 XML 架构文档或者文档类型定义 (DTD) 文档中定义之规则的文档。

关于 XML 文档，人们经常会混淆两个概念。
- 格式良好的 XML 文档
- 有效的 XML 文档

格式良好的 XML 文档是符合以下基本规则的文档。
- 它必须有一个，而且只有一个根元素。
- 各个元素起始于一个起始标签，并结束于一个结束标签。
- 一个元素可以拥有其他元素、属性或文本节点。
- 属性值必须包含在双引号中。而文本节点则不应当包含在双引号中。

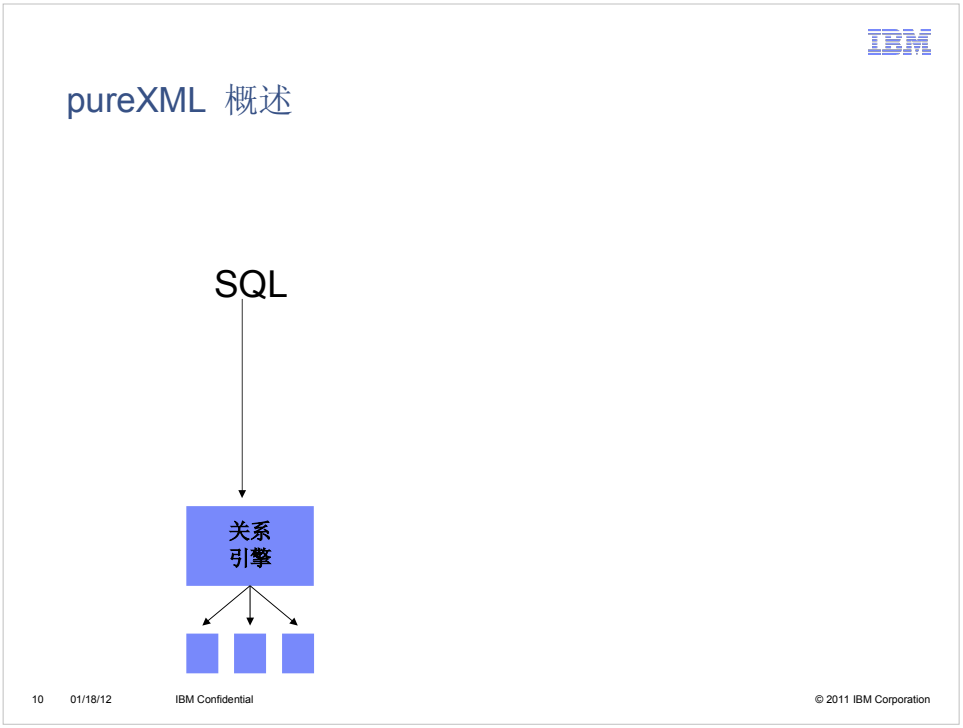
例如：
- 丢失了结束标签：

```
<employee>
  <name>John
</employee>
```

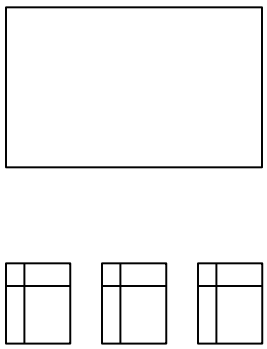
- 没有根节点

```
<department></department>
<name>John</name>
```

有效的 XML 文档是
- 格式良好的 XML 文档。
- 符合 XML 架构文档或者文档类型定义 (DTD) 文档中定义之规则的文档。

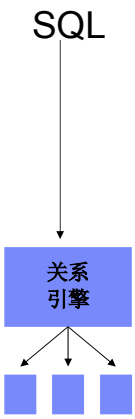


首先，我们在左侧简单地示出了一个关系管理系统，你在这里可以发出 SQL 语句，该语句由关系引擎处理，而该引擎可以访问存储在表中的数据。



pureXML 概述

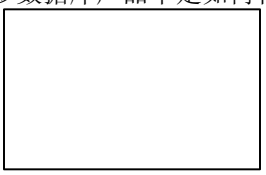
```
<employee id="9">
    <name>John</name>
    <phone> 555 1212</phone>
</employee>
```

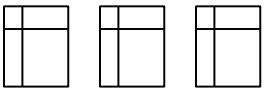


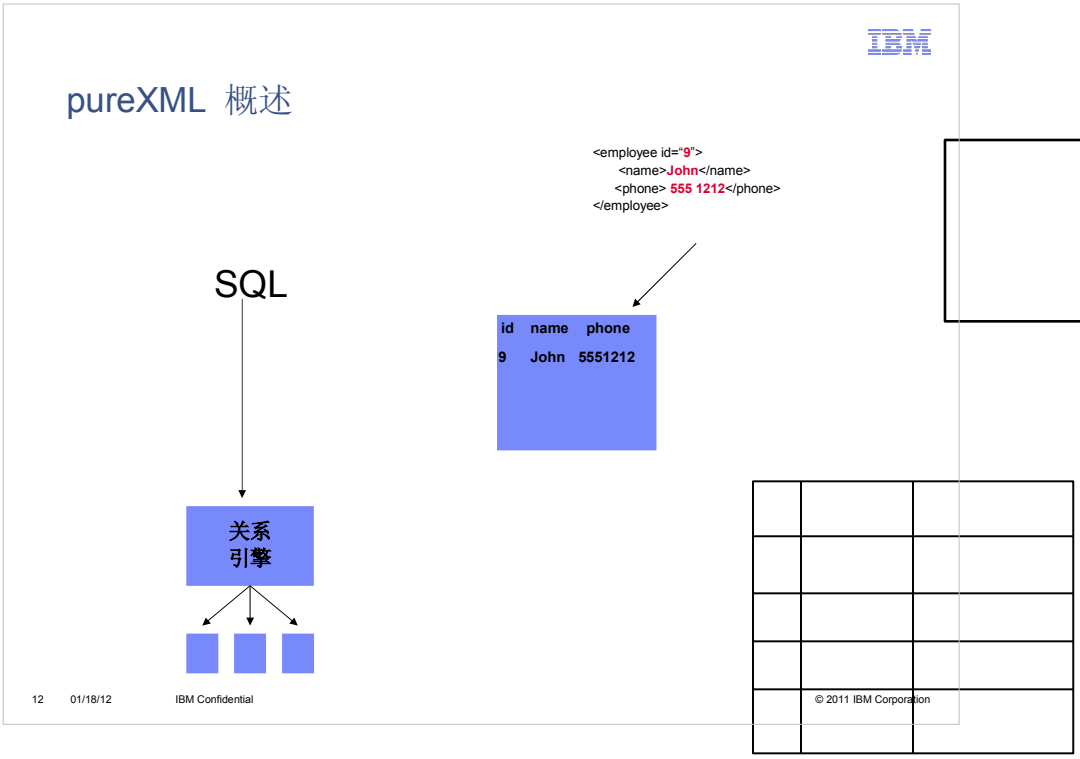
11 01/18/12 IBM Confidential

现在，由于出现了 Web 2.0 和 SOA， XML 的使用量得到飞速增长。

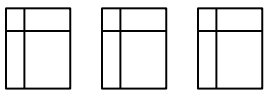
XML 在许多数据库产品中是如何存储的呢？

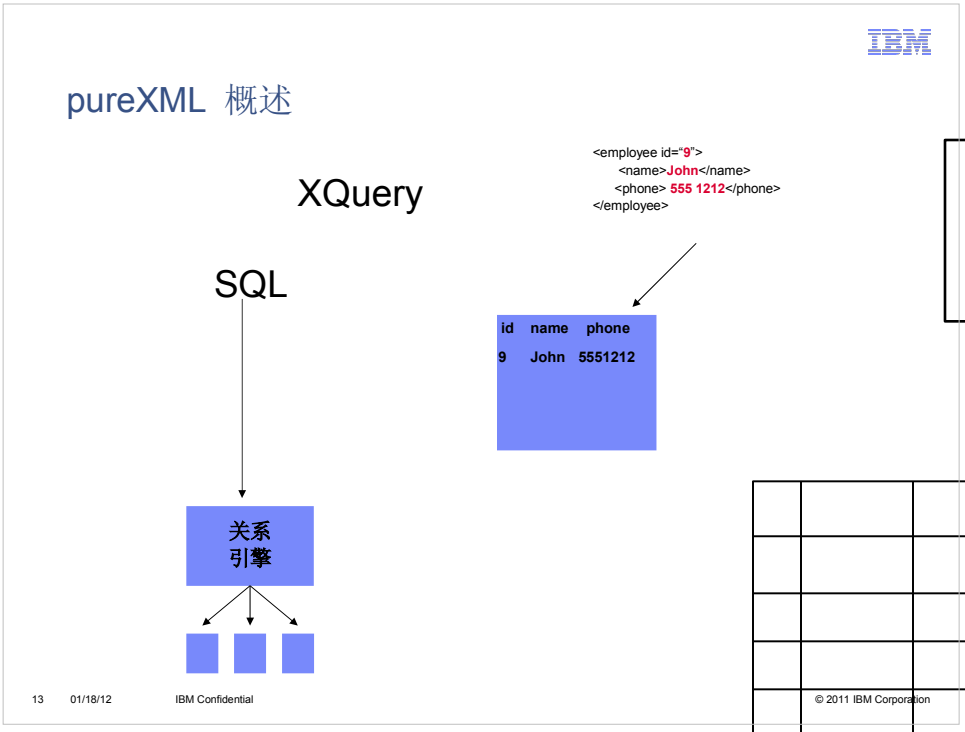




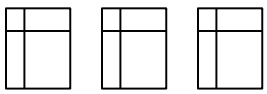


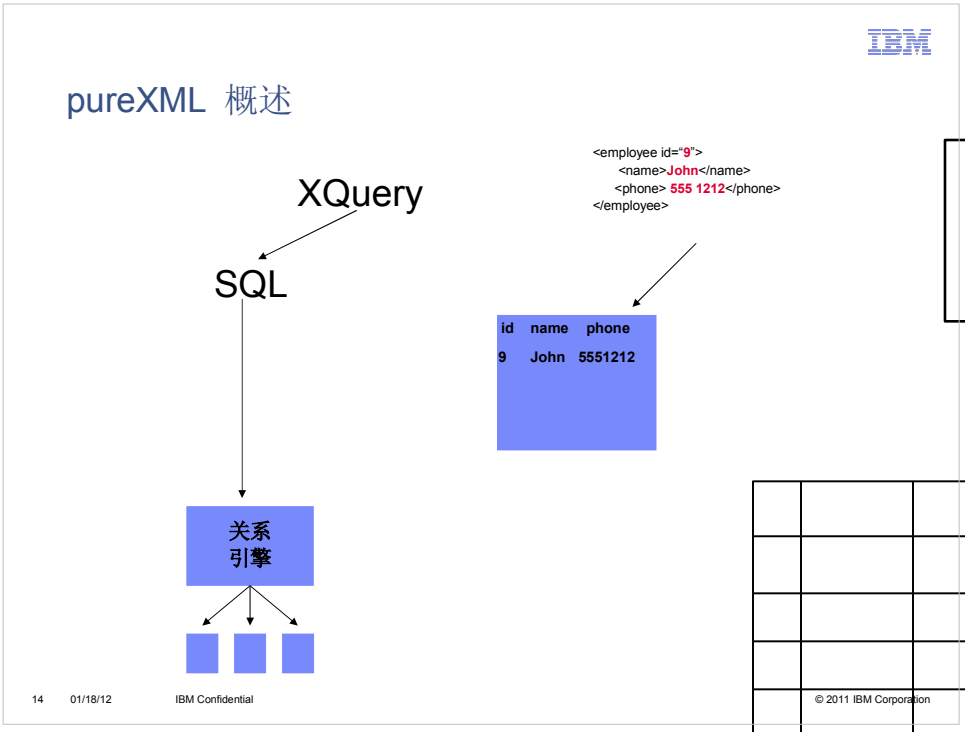
好了，现在把 XML 文档映射为表的形式（就是把它从层次格式转化为关系格式）。



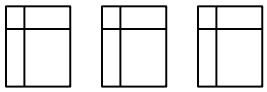


下一步，当你发出 XQuery 命令时，



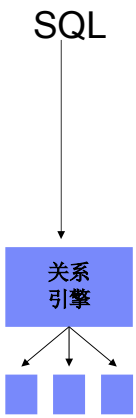


由于 Xquery 不能被关系引擎所理解，它必须首先映射为 SQL，这在幕后进行。



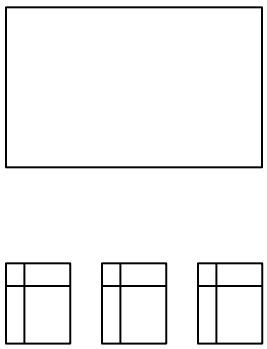
pureXML 概述

```
<employee id="9">
    <name>John</name>
    <phone> 555 1212</phone>
</employee>
```



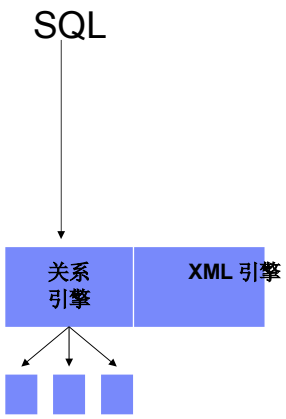
15 01/18/12 IBM Confidential

利用 DB2 pureXML 技术，我们不用这样做。



pureXML 概述

```
<employee id="9">
    <name>John</name>
    <phone> 555 1212</phone>
</employee>
```

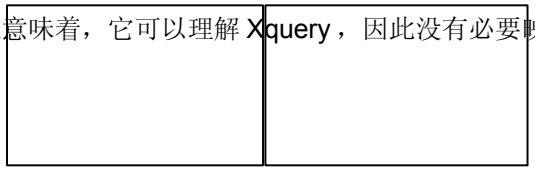


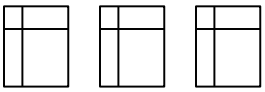
16 01/18/12 IBM Confidential © 2011 IBM Corporation

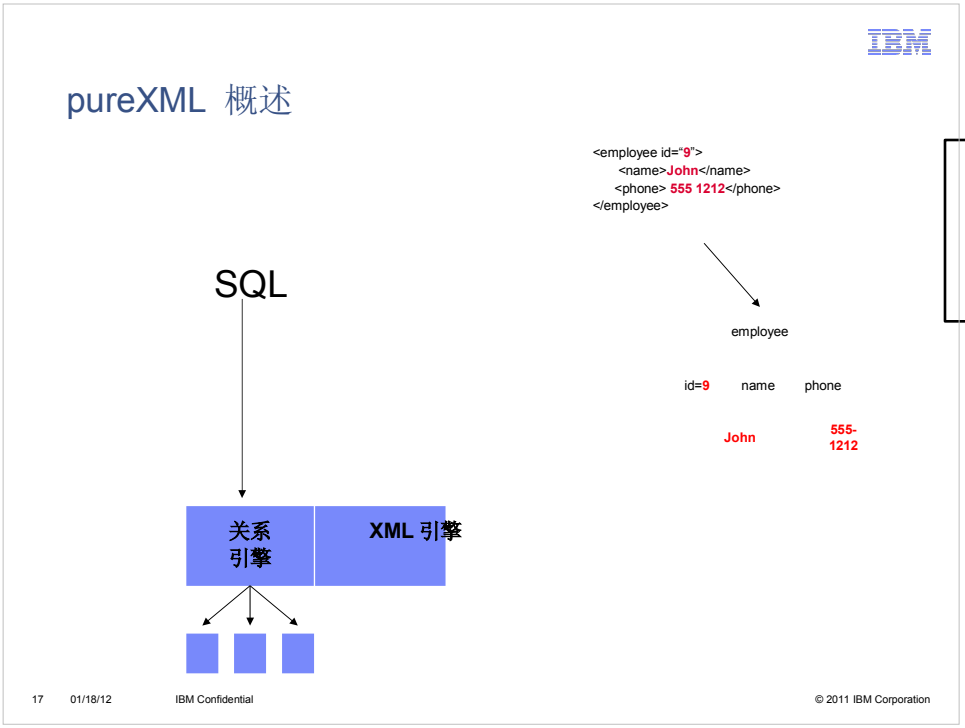
PureXML 有两个主要特征：

1) DB2 引擎的第二部分能够理解原生态 XML

这意味着，它可以理解 Xquery，因此没有必要映射至 SQL

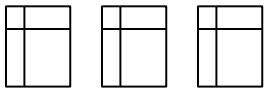


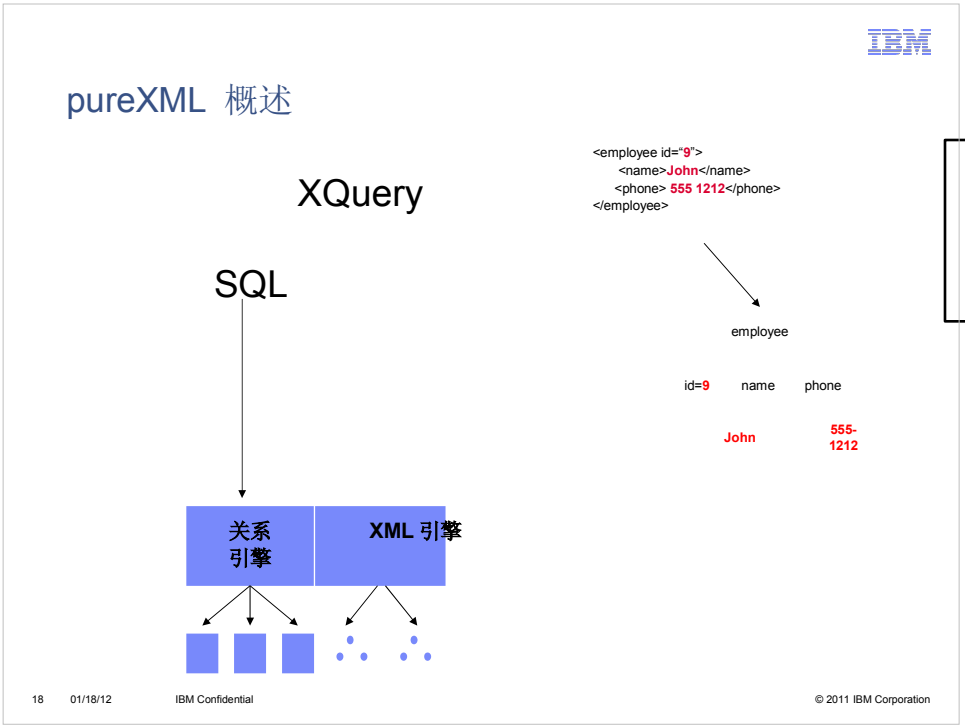




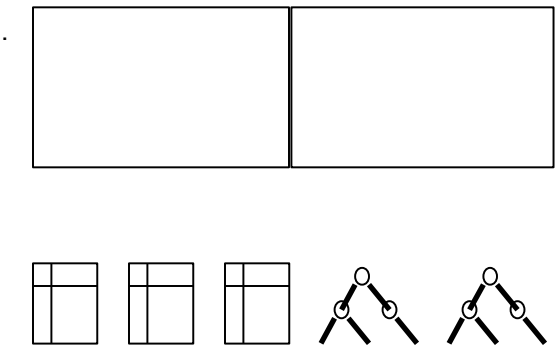
2) XML 在数据库中以解析的层次格式存储。

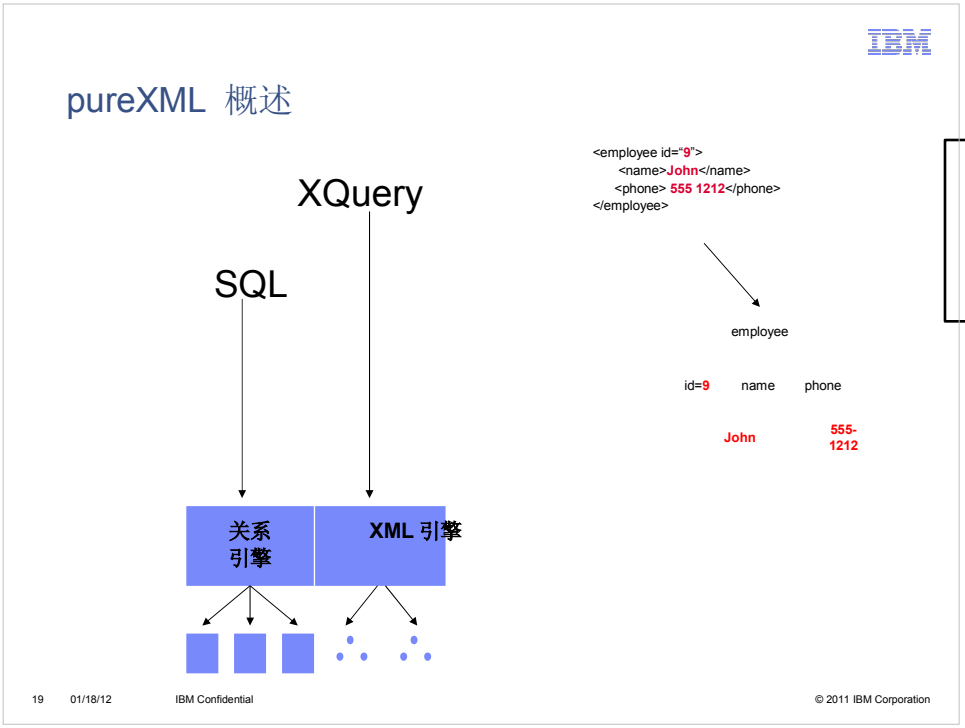
因此，不需要进行映射。没有从层次到关系的转化。 XML 被存储为树型结构，它在性质上是层次的。





因此，现在，如果我们发出一个 Xquery 语句，



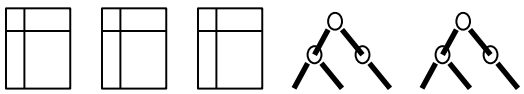


它可以由该引擎的 XML 部分所理解，而不需要进行任何映射。

简言之，存储格式 = 处理格式

因此，性能会更好，代码会更少，维护会更容易。

还要注意，引擎的这两部分（关系部分和 XML 部分）可以相互对话，因此我可以把 SQL 与 Xquery 结合起来。下面将解释。



XML 与关系数据的集成

PATIENTS 表

NAME	SEX	AGE	COFFEE	SMOKE	DNA
Elizabeth	F	65	35	15	
Raul	M	47	10	0	
…					
Tina	F	58	4	10	

把关系数据和 XML 数据集成到同一个查询中也可以产生有用的信息。例如，假设我是一名医学研究人员，我在寻找癌症的治疗方法。假设我在过去 25 年一直在从事该工作，我把患者的信息存储在一个表中，表中有许多关系数据，例如患者姓名、性别、年龄、饮用咖啡历史、抽烟史，还可以把每名患者的 DNA 存储在最后一列。我把 DNA 信息存储为 XML 文档，因为 XML 很适用于存储像 DNA 这样的非结构化信息或半结构化信息。我把 XML 文档表达为树型结构，因为 XML 文档和树在特性上都是分层次的。现在我们来假设所有的癌症患者或者我的大部分癌症患者都具有值为 987 的基因 X 。

因此，利用 DB2 ，我可以把关系数据与该基因 X 数据集成到一起。诸如咖啡和抽烟等构成了关系数据，它们属于可能导致，也可能不导致癌症的因素；而值为 987 的基因 X ，似乎在大多数癌症患者身上出现。我们在下一张幻灯片中看看如何实现这种集成。

含有 XQuery/XPath 的 SQL

```
SELECT name from PATIENTS
WHERE
xmlexists('$p/Patient/MedicalDetail[geneX="987"]'
          passing PATIENTS.DNA as "p")
and sex = 'F'
and age > 55
and coffee > 15
and smoke > 10
;
```


运行这个查询可以进行这种集成。

头两行和最后四行只是普通的 SQL 语句，我在这里从患者表中查询姓名，并根据性别、年龄、咖啡和抽烟进行某些过滤。第三行和第四行 DB2 中新出现的。首先，我们调用 XMLEXISTS 函数，该函数是 SQL/XML 标准的一部分。此函数可以让我进行一些测试，以便查看某个条件是否在 XML 文档中得到满足，我把这用作我查询的另一个过滤器。

$p 是一个变量（带有 '$' 的任何东西意味着它是一个变量），此变量被定义在第四行，我在这一行把 PATIENTS.DNA 赋给 'p'。 PATIENTS.DNA 是 XML 列，因此，我基本上是把 XML 文档传递给 'p'。回头看看第三行，在 $p/Patient/MedicalDetail... 中，我利用 Xpath 来遍历此 XML 文档，直至检验是否 geneX = '987'。

因此，我把 SQL 与 XPATH 或 XQUERY 结合到一起，或者把关系模型与 XML 的层次模型混合到一起。

最终结果是，我或许能够找到多项因素之间的某些相关关系，例如饮用咖啡或抽烟与该基因的出现之间的关系，此类查询或许有助于寻找癌症的资料。

现在 XML 文档被插入到数据库中， DB2 对 XML 文档进行分析，并在内部利用 XQuery 数据模型 (XDM) 把它存储为分析过的树结构。此树是持久的。

如果你进行相反的程序（从分析过的树格式转变为序列化的格式），则被称为序列化。

本地 XML 存储

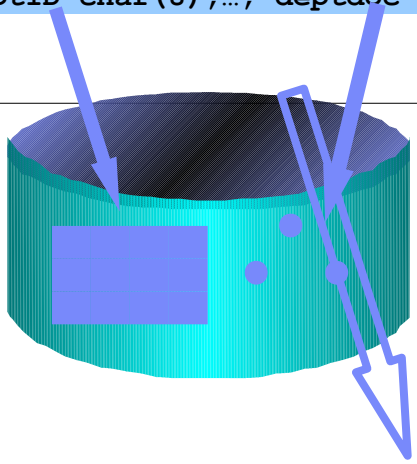
以解析的层次格式存储的 XML

```
create table dept (deptID char(8),…, deptdoc xml);
```

关系列存储在关系格式（表）中

XML 列以原生态存储

以 UTF8 格式存储的 XML



为了存储 XML 文档，你可以创建一张普通的表，但对于 XML 文档，你需要创建一个定义为 XML 数据类型的列。在本例中，有一个表“ dept”，它有一个关系列“ deptID”，定义为 char(8)；还有一个列“ deptdoc”，定义为 XML 。

议题

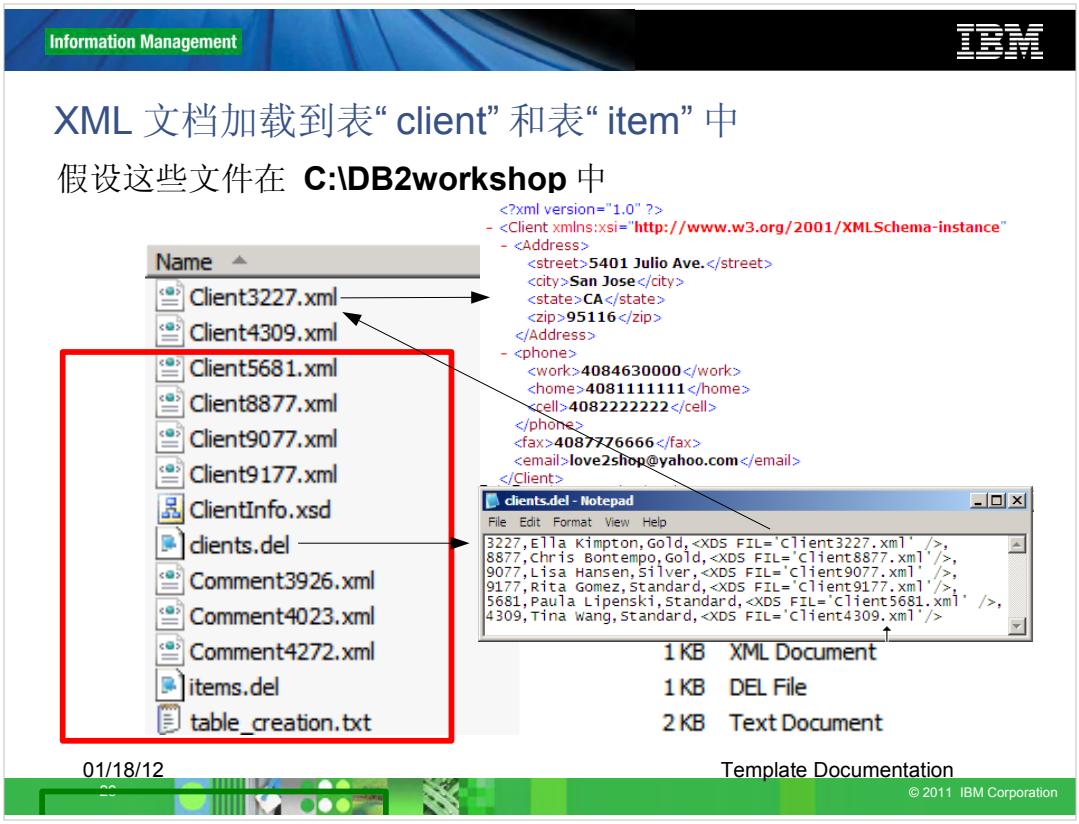
- 概述
- 插入 XML 数据
- XPath
- XQuery
- 利用 SQL/XML 查询 XML 数据
- 利用 XQuery 查询 XML 数据
- 对 XML 进行更新和删除操作
- XML 索引
- XML 模式验证
- 其他 XML 支持

具有 XML 列的表定义

```
create table items (
  id          int primary key not null,
  brandname   varchar(30),
  itemname    varchar(30),
  sku         int,
  srp         decimal(7,2),
  comments    xml
);
```

```
create table clients(
  id          int primary key not null,
  name        varchar(50),
  status      varchar(10),
  contact     xml
);
```


假设我们要创建两张表， ***items*** 表和 ***clients*** 表，两个表中的最后一列都被定义为 XML 数据类型。 XML 列并不需要在表的最后一列定义，我们只是在这一个特定的例子把它们放到这里。



本图显示在“ C:\DB2workshop” 目录中有多个文件。

文件“ CLIENT. DEL” 是一个有定界符的 ASCII 文件，它的数据以逗号（定界符）相分隔。如果我们提前这份文件的第一行以及这一行的最后一列，我们会看到类似“ <XDS FIL='Client3227.xml'>“ 这样的内容。这是一个指向 XML 文件的指针，此文件的内容也示于图中。

同样，还有一个 items.del 文件以及相应的 XML 文件。

我们大致在本图中解释了我们准备向我们前面创建的表中插入什么内容。

插入 XML 文档

第 **1** 种方法：一个简单的 **SQL INSERT**

可以是隐式的或者显式的

隐式 **XML** 解析：以字符串形式输入的 **XML** 文档

```
INSERT INTO clients VALUES (77, 'John Smith', 'Gold',
  '<addr>111 Main St., Dallas, TX, 00112</addr>') ;
```

利用 **XMLPARSE** 函数进行显式的 **XML** 解析

告诉系统如何处理空格（ strip （除去） /preserve （保持））

默认为 'Strip WHITESPACE'

```
INSERT INTO clients VALUES (77, 'John Smith', 'Gold',
  xmlparse (document '<addr>111 Main St., Dallas, TX,
  00112</addr>' preserve whitespace) );
```


有两种方法把 XML 文件插入到表中：

1) 使用简单的 SQL INSERT 语句。

你可以隐式或显式地这样做。

隐式意味着你只需把 XML 文档传递为一个串，而且由于它被插入到 XML 列中， DB2 将知道是否必须对它进行解析。

显式意味着你必须告诉 DB2 利用 XMLPARSE 函数对 XML 文档进行解析。你也可以说明如何处理空格。

插入 XML 文档

第 **2** 种方法：使用 **DB2 IMPORT** 实用程序

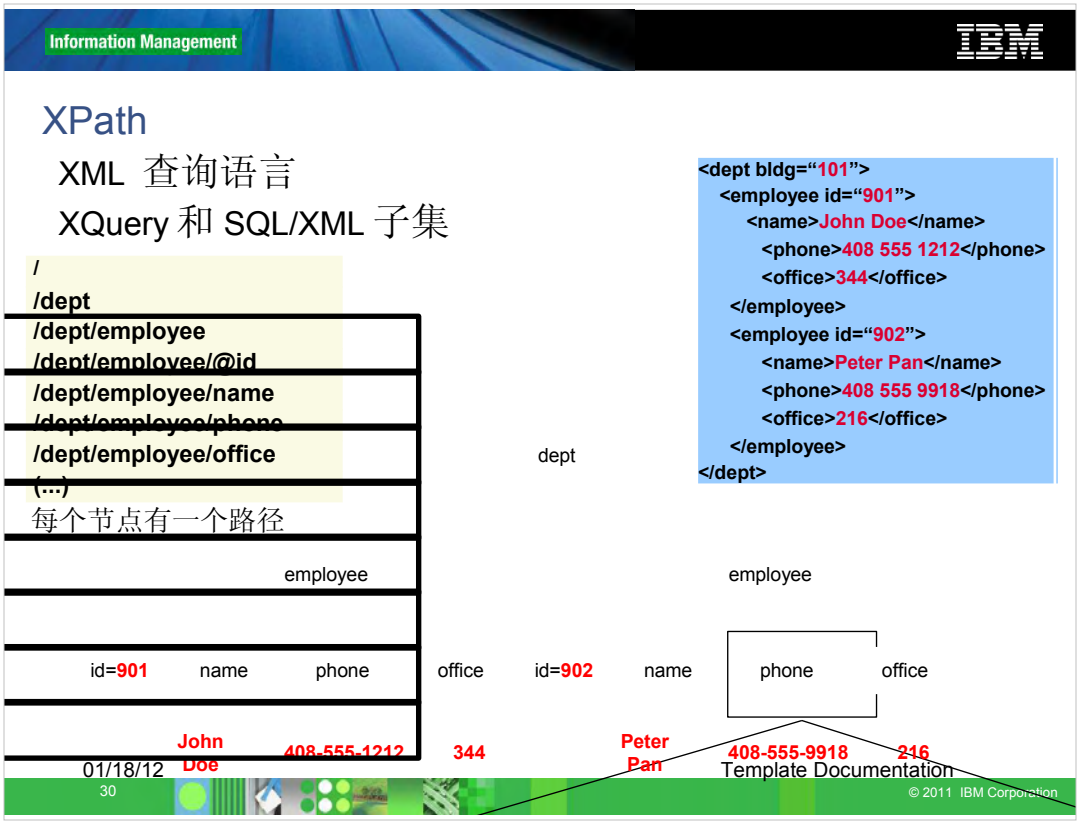
```
IMPORT from "C:\DB2workshop\clients.del" of del
   xml from "C:\DB2workshop" INSERT INTO
   CLIENTS (ID, NAME, STATUS, CONTACT);
```

```
IMPORT from "C:\DB2workshop\items.del" of del
   xml from "C:\DB2workshop" INSERT INTO
   ITEMS (ID, BRANDNAME, ITEMNAME, SKU, SRP,COMMENTS);
```


第二种方法是使用 IMPORT 实用程序。在前面两张幻灯片中，我们示出了 C:\DB2workshop 目录的内容。现在把该目录文件用于 IMPORT 命令。

议题

- 概述
- 插入 XML 数据
- XPath
- XQuery
- 利用 SQL/XML 查询 XML 数据
- 利用 XQuery 查询 XML 数据
- 对 XML 进行更新和删除操作
- XML 索引
- XML 模式验证
- 其他 XML 支持



XPath 是一套可以用来浏览 XML 文档的语言。它并不是 IBM 专有的东西，而是一套标准。它基本上是一种查询语言。这里右侧是一份以序列化方式表达的文档。

下面是以数结构表达的同一份文档，这也被称为解析的层次表达。因此，这两种表达方式实际上是一回事。在本例中，你可以把解析的层次表达视为 DB2 在数据库中持久地存储 XML 文档的方式。

某些 XPath 表达式示于左上角。 XPath 很容易学习；它非常类似于你在 MS-DOS 、 Windows 、 Linux 、 Unix 中使用的 cd 命令（变更目录命令）。例如，你可以使用：

`CD /directory/subdirectory/subdirectory`，因此，基本而言，你可以使用 **cd** 命令来浏览一个树，树结构在性质上是分层次的。

对于 XPath 也是如此。例如，你可以使用 **/dept/employee/name** ；这意味着，你首先进入 dept 元素（或者称为根），然后再进入 employee 元素，然后再进入 name 元素，这时你就得到信息 “ John Doe” 。在本例中你可以看到有两个 employee ，因此你将得到两个，你可以一直得到 John Doe ，也一直得到 Peter Pan 。

XPath：简单的表达式

使用完全限定路径来规定元素 / 属性

“@” 被用于指定属性

使用“ text()” 来规定元素之下的文本节点

```
<dept bldg="101">
  <employee id="901">
    <name>John Doe</name>
      <phone>408 555 1212</phone>
      <office>344</office>
  </employee>
  <employee id="902">
      <name>Peter Pan</name>
      <phone>408 555 9918</phone>
      <office>216</office>
  </employee>
</dept>
```

XPath	Result
/dept/@bldg	101
/dept/employee/@id	901 902
/dept/employee/name	<name>John Doe</name> <name>Peter Pan</name>
/dept/employee/name/text()	Peter Pan John Doe

还有其他的使用 XPath 的方式。有多种 XPath 表达式可以用来选择你所需要的东西，我们来快速看看一些表达式。

假设右上侧的 XML 文档是我需要处理的。如果我使用：

/dept/@bldg （ **@** symbol 符号表示我希望得到属性），这就是检索 101，因为 101 是属性。

对于 **/dept/employee**，属性 id，也是一样的。在本例中，属性将是 901 。因为有两个 employee，还有一个 902，因此我得到两个属性， 901 和 902 。

`/dept/employee/name` 与我在前面的幻灯片中给出的例子相同。要注意，我在这里也得到了标签名称，以及结束标签，它就是这样规定的，标签也包含在内。

如果你不想把标签包含在内，你可以在 XPath 的末尾添加一个称为 **text** 的函数，这样就会只获得值，就像最后一个表达式所示。

XPath：通配符

* 匹配任何标签名称

// 是“后代或自身” (descendent-or-self) 通配符

```
<dept bldg="101">
  <employee id="901">
    <name>John Doe</name>
      <phone>408 555 1212</phone>
      <office>344</office>
  </employee>
  <employee id="902">
      <name>Peter Pan</name>
      <phone>408 555 9918</phone>
      <office>216</office>
  </employee>
</dept>
```

XPath	Result
/dept/employee/*/text()	John Doe 408 555 1212 344 Peter Pan 408 555 9918 216
/dept/*/@id	901 902
//name/text()	John Doe Peter Pan
/dept//phone	<phone>408 555 1212</phone> <phone>408 555 9918</phone>

在本幻灯片中，使用与前面相同的 XML 文档示例。如果你使用：

`/dept/employee/*`（星号是通配符，它代表任何一个元素），它将进入 dept 、 employee ，并将获得所有这些： John Doe, 408, 344 ；对于另一个 employee 也是一样。

例子中的第二行也类似：进入 dept 以及后面的任何元素，并获得属性 id ， 901 和 902 。

对于第三行，两个斜线意味着，它不只是一个元素，而是在你到达 name 之前的许多元素。例如，这里是 name ，这些是其之上的任何元素，我不管它们是什么，任何元素。然后我得到文本： Peter Pan, John Doe 。

最后，例子中的最后一行：`/dept//phone` ，它意味着 dept 与 phone 之间的任何元素。

XPath： 谓词

谓词被包含在方括号 **[…]** 中
一个 **XPath** 中可以有多个谓词
位置谓词 **: [n]** 选择第 **n** 项子元素

```
<dept bldg="101">
  <employee id="901">
    <name>John Doe</name>
      <phone>408 555 1212</phone>
      <office>344</office>
  </employee>
  <employee id="902">
      <name>Peter Pan</name>
      <phone>408 555 9918</phone>
      <office>216</office>
  </employee>
</dept>
```

XPath	Result
/dept/employee[@id="902"]/name	<name>Peter Pan</name>
/dept[@bldg="101"]/employee[office >"300"]/name	<name>John Doe</name>
//employee[office="344" OR office="216"]/@id	901 902
/dept/employee[2]/@id	902

这里的方括号与 SQL 中的 WHERE 子句很相似。例如，对于第一行，就是 `/dept/employee`，属性 id 为 902，在这个地方取得 name 。于是就给出了 Peter Pan 。

在第二行的表达式中，给出了两个条件。

在第三行的表达式中，有两个斜线，因此，无论在 employee 之前有什么元素，只要这里的 office 等于 344 或者 office 等于 216，就从这里取得属性 id 。

第四行，最后一个例子，其中有一个 [2] 。这意味着，我希望得到第二个 employee 。如果它是“ 1”，那就是“我希望得到第一个 employee”。

XPath: 父节点轴

当前上下文：“.”

父节点上下文：“..”

```
<dept bldg="101">
  <employee id="901">
    <name>John Doe</name>
      <phone>408 555 1212</phone>
      <office>344</office>
  </employee>
  <employee id="902">
      <name>Peter Pan</name>
      <phone>408 555 9918</phone>
      <office>216</office>
  </employee>
</dept>
```

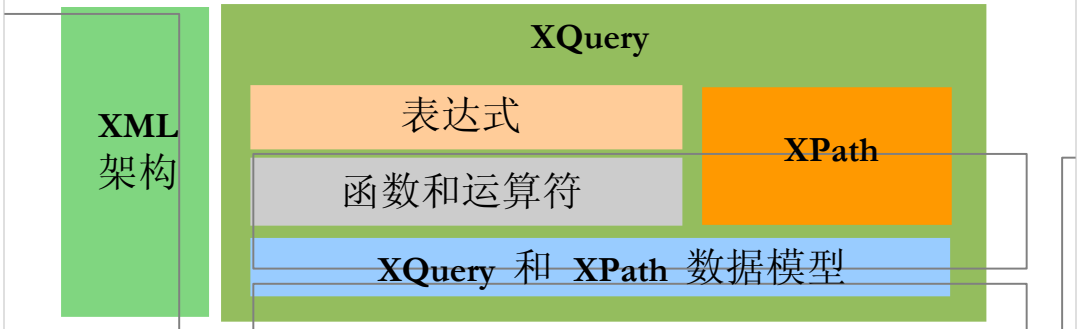
XPath	Result
/dept/employee/name[../@id="902"]	<name>Peter Pan</name>
/dept/employee/office[.>"300"]	<office>344</office>
/dept/employee[office > "300"]/office	<office>344</office>
/dept/employee[name="John Doe"]/../@bldg	101
/dept/employee/name[.="John Doe"]/../../@bldg	101

这些类似于你使用变更目录命令的情况，你可以使用“.”或“..”，它们分别表示当前上下文和父上下文。

议题

- 概述
- 插入 XML 数据
- XPath
- XQuery
- 利用 SQL/XML 查询 XML 数据
- 利用 XQuery 查询 XML 数据
- 对 XML 进行更新和删除操作
- XML 索引
- XML 模式验证
- 其他 XML 支持

什么是 XQuery?



XQuery 支持使用路径表达式来浏览 **XML**

XQuery 支持有类型的及无类型的数据

XQuery 没有空值，因为 **XML** 文档会省略失去的或未知的数据

XQuery 返回 **XML** 数据的序列

XQuery 是 XPath 的一个超级集合。 XQuery 用来浏览 XML 文档。本图中还示出了其他一些组件，它们也是 Xquery 的一部分。有些将在后面详细讨论。

Xquery 返回 XML 数据的一个序列。在 Xquery 中，不存在 NULL 值。相反，将使用空白。

XQuery: FLWOR 表达式

FOR: 在序列中迭代，将变量绑定为序列中的各项
LET: 将变量绑定为一个序列
WHERE: 消除迭代中的项目
ORDER: 对迭代的项目重新排序
RETURN: 构造查询结果

在 XQuery 中，我们有一种称为 FLWOR 表达式的东西，即 F-L-W-O-R ，利用该表达式，你可以进行更多的操纵。 FLWOR 代表： For, Let, Where, Order, Return 。

FLWOR 对于 Xquery 而言就相当于 SQL 中的 SELECT-FROM-WHERE ORDER BY 子句。

XQuery: FLWOR 表达式

```
create table dept
  (deptID char(8),deptdoc xml);


xquery
 for $d in
     db2-fn:xmlcolumn('DEPT.DEPTDOC')/dept
 let $emp := $d//employee/name
 where $d/@bldg > 95
 order by $d/@bldg
 return
    <EmpList>
    {$d/@bldg, $emp}
    </EmpList>
```

输入：

```
<dept bldg="101">
  <employee id="901">
      <name>John Doe</name>
      <phone>408 555 1212</phone>
      <office>344</office>
  </employee>
  <employee id="902">
      <name>Peter Pan</name>
      <phone>408 555 9918</phone>
      <office>216</office>
  </employee>
</dept>
```


我们来看看这个例子。右上角是一份 XML 文档，我们将把它用作输入。顶部有 **dept** 表的定义，其中有一个关系列，还有一个 XML 列，在该列中假设存储了将用作 XQuery 输入的 XML 文档。

你也可以使用 FLWOR 表达式。我们必须用 XQuery 作为起始，因为如果我们不把 XQuery 放到开头的话， DB2 将默认假设我们在使用 SQL，或者认为你将运行 SQL 语句。

我们来看看这个 FLWOR 表达式： **for $d in**，等等。这里，我把 XML 文档装入变量“ d”，该文档是通过函数 db2-fn:xmlcolumn 提供的；

“let $emp...”，我把变量 $d 的值（就是整个 XML 文档）， `//employee/name`，赋给一个新的称为 **emp** 的变量；

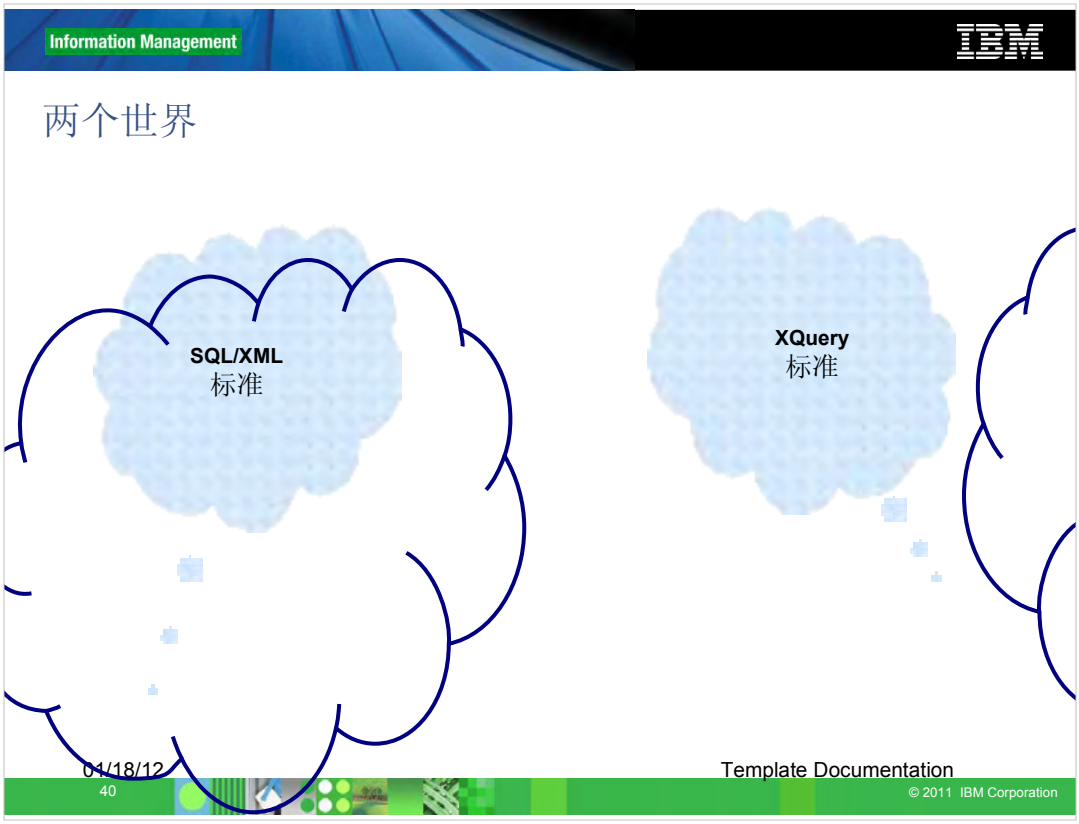
“where $d...”，这里是一些条件，它们随后再执行 **order by**

然后，我打算返回的是 **Emplist**，所以显示出来的将是 $d@bldg （它将使用 101 ），然后是 $emp （无论涉及到什么 name，因此是 John Doe 和 Peter Pan ），还有这一部分，最后以 Emplist 结束。

你可以使用 FLWOR 表达式来改变你的 XML 文档的格式。例如，你可以使它遵守 RSS 或 Atom 的规则或语法。利用这种方式，你还可以从本 XML 中创建馈入。

议题

- 概述
- 插入 XML 数据
- XPath
- XQuery
- 利用 SQL/XML 查询 XML 数据
- 利用 XQuery 查询 XML 数据
- 对 XML 进行更新和删除操作
- XML 索引
- XML 模式验证
- 其他 XML 支持

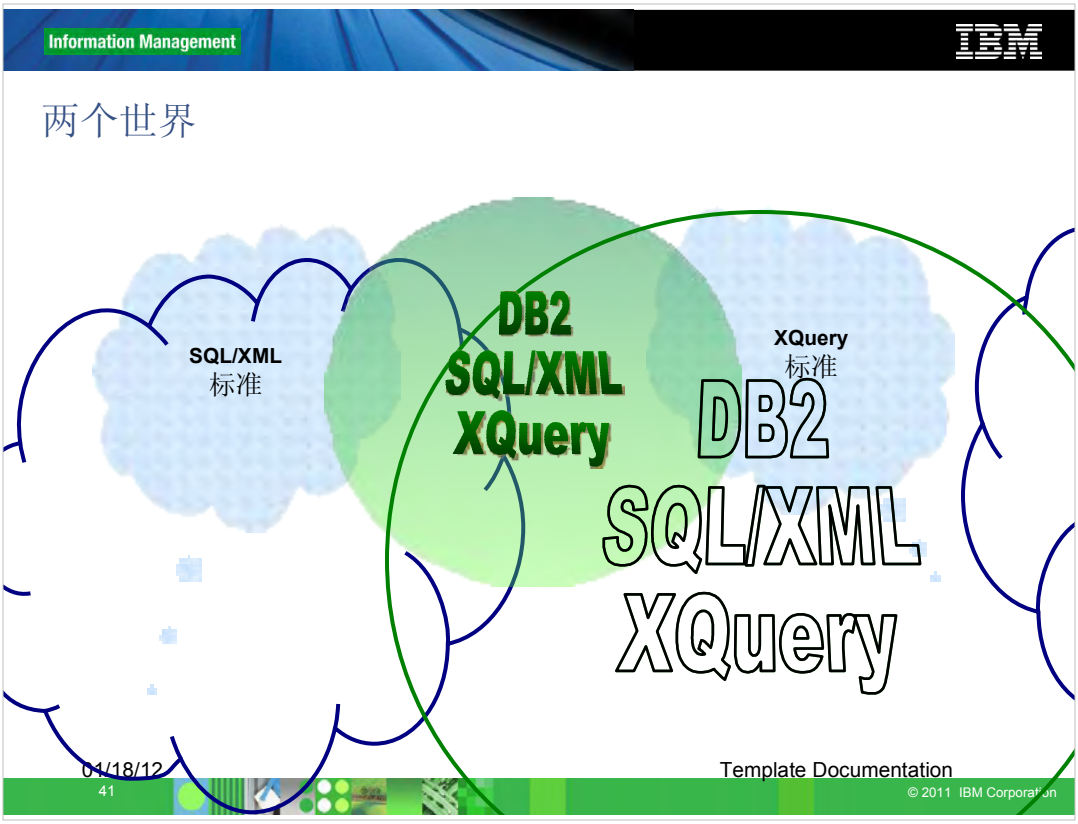


简单的 SQL 语句能够让你检索完整的 XML 文档，但你无法规定基于 XML 的查询谓词，而且你也无法检索部分 XML 文档。例如，如果我只有 SQL 的话，我将只能够获得整个 XML 文档。

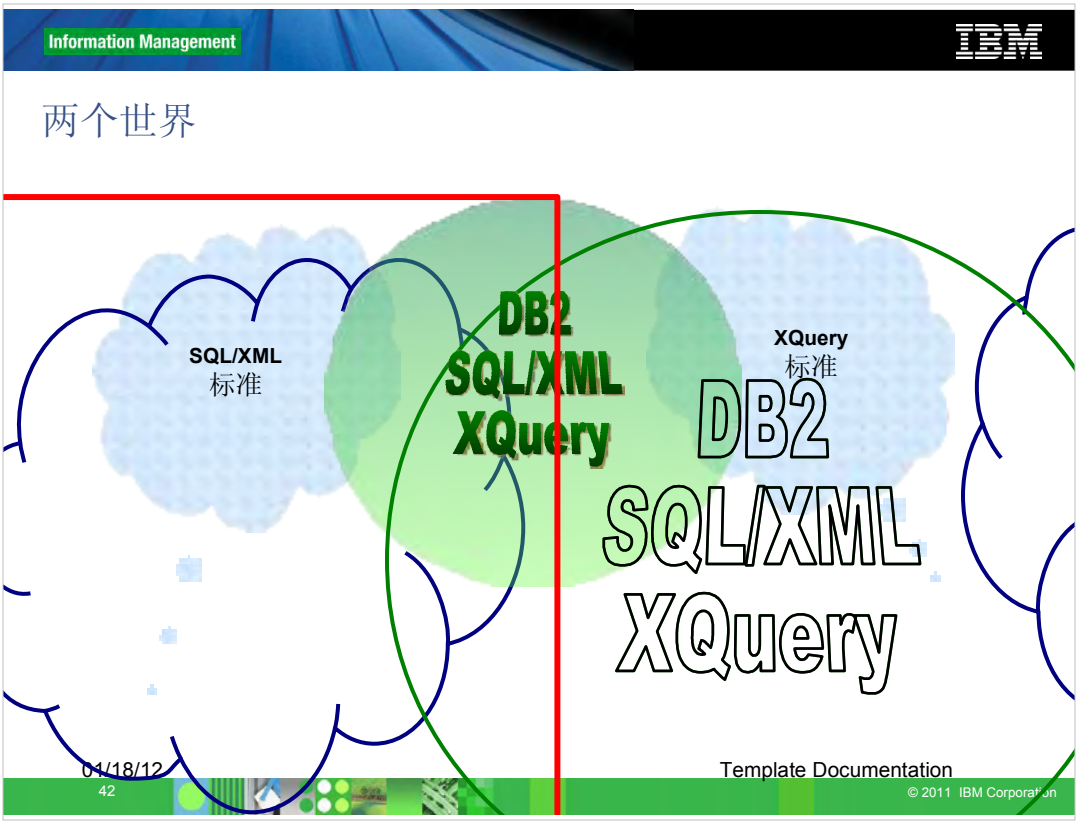
如果你执行 “ **select deptdoc from dept**”，其中的 deptdoc 是一个 XML 列，你将会检索到表中的整个 XML 文档，因此，这样一个简单的 SQL 语句可以处理 XML 。但是，如果你只想检索 XML 文档的一部分，该怎么办呢？或者，如果你打算使用 XML 文档的一部分作为 WHERE 从句中的条件，又该怎么办呢？在这些情况中，你不能只使用 SQL ，你还需要使用其他标准：

1. SQL/XML （它是 SQL 的扩展，其中包括 XML 函数，因此它可以用作 SQL 世界与 XML 世界之间的桥梁），或者

2. Xquery （一个 XPath 的超级集合）



DB2 即支持 SQL/XML 标准，也支持 Xquery 标准



我们先来描述 SQL/XML 标准

SQL/XML 函数

Function	Description
XMLPARSE	Parses character/BLOB data, produces XML value
XMLSERIALIZE	Converts an XML value into character/BLOB data
XMLVALIDATE	Validates XML value against an XML schema and type-annotates the XML value
XMLEXISTS	Determines if an XQuery returns results (i.e. a sequence of one or more items)
XMLQUERY	Executes an XQuery and returns the result sequence
XMLTABLE	Executes an XQuery, returns the result sequence as a relational table (if possible)
XMLCAST	Cast to or from an XML type

SQL/XML 是新的 SQL 2006 标准的一部分，它包含一些 XML 函数，我们可以使用。这里列出了一些流行的 SQL/XML 函数和它们的描述。

XMLPARSE，我们在介绍插入时已经讨论过 XMLPARSE；

XMLSERIALIZE 正好与 XMLPARSE 相反；

XMLVALIDATE 用于验证 XML 文档和 XML 模式。我们将在本演示结束时稍微详细讨论一下这个函数。

XMLEXISTS, XMLQUERY, XMLTABLE 等等。我们将在下面几个幻灯片中介绍这些函数的若干例子。

XMLEXISTS 函数

在 **WHERE** 从句中使用该函数根据某个 **XML** 元素值过滤各行

语法 **1:** 显式创建一个变量，用于容纳 **XML** 文档

```
select name from clients
  where
  xmlexists('$c/Client/Address[zip="95116"]'
                passing CLIENTS.CONTACT as "c")
```

语法 **2:** 利用 **XML** 列的名称自动创建一个变量

```
select name from clients
  where
xmlexists('$CONTACT/Client/Address[zip="95116"]')
```


XMLEXISTS 函数能够让我根据 XML 列进行测试，而且通过这种方法我可以限制某些数量的行，例如，这个语句： **select name from clients, where, XMLexists, $ c, Client, address, zip, 95116**, passing CLIENTS.CONTACT as c. 前两行基本上是普通的 SQL ；我希望从 clients 中获得 name 。

然后第三行使用了 **XMLexists** 。 **$ c** 是一个变量。任何带有 $ 的东西都代表一个变量。该变量是在什么地方定义的？该变量在第四行定义，也就是 **passing CLIENTS.CONTACT as c** 这一部分。

CONTACT 是含有 XML 文档的列，而 **CLIENT** 是表的名称。

因此，现在，变量 c 就含有 XML 文档。然后 `/Client/Address` 是 XPath ，而方括号的作用与 SQL 中的 where 子句类似，用于测试 zip 是不是 95116 。

基本上， XMLexists 函数能够让我根据某个元素值过滤某些行，在本例中，就是根据 zip 值进行过滤。

因此，我需要在 XML 文档中 zip 元素为 95116 的所有的行。这就是我们利用该示例完成的工作。

在 DB2 9.5 中，语法被简化了。在下面的例子中，我们执行 **select name from client where XMLexist**, 我们这里有 **$ CONTACT** 。这是一个自动创建的变量，其名称与含有 XML 的列的名称相同。在语法上无需再向它赋值了。

大小写敏感 -- 小心！

SQL 不是大小写敏感的

XPath 和 **Xquery** 是大小写敏感的！

DB2 对象（表 / 列）需要以大写字母输入。

示例：

错误：

```
select name from clients
  where
xmlexists('$contact/Client/Address[zip="95116"]')
```

正确：

```
select name from clients
  where
xmlexists('$CONTACT/Client/Address[zip="95116"]')
```

SQL 是大小写不敏感的，因此，当你使用 SQL 时，你无需在乎大小写；但是，在使用 XPath 和 XQuery 时，你确实需要当心大小写。如果你把 SQL 与 XQuery/XPath 结合到一起， SQL 部分仍然可以任意采用大小写，但对于 XPath 和 XQuery 部分，你必须考虑大小写。

例如，这个 **select name from clients** 部分可以用任意大小写来编写， XMLexists 也可以采用任何大小写，但该函数的内部是大小写敏感的。例如， ***Client***, 如果你把 ***C*** 改为小写，你很可能会得到不正确的结果。或许不会有错误提示，因为语法是正确的，但 DB2 可能无法找到该路径，因此你可能得到不正确的结果。

那么，为什么 **contact** (in lowercase)（小写）不正确，而 **CONTACT** (in uppercase)（大写）就正确呢？因为 DB2 对象必须必须采用大写，无论是表、列，等等。因为当你在 DB2 中创建表之类的对象时，它通常会以大写方式存储起来，无论在 **create table** 语句中使用了大写还是小写。当然，你可以使用双引号强迫 DB2 表以小写方式创建，但一般不推荐这样做，因为这只会为处理表的工作带来更多复杂性。

这就是为什么每次当你在查询语句的 Xpath/XQuery 部分调用列或表的名称时，需要采用大写方式。

注：如果你使用 Data Studio， Data Studio 将严格按照你的输入来创建对象。我们建议你在 Data Studio 中创建对象时使用大写。

另一件要考虑的事情是引号的使用。一定要保证它们是直引号 (straight quote)，而不是弯引号 (curly quote)。 MS-Word/Powerpoint 等软件往往自动把直引号改为弯引号，因此，如果你从这些程序中复制 / 粘贴语句，你可能会遇到引号方面的麻烦。

XMLQUERY 函数

从 **XML** 文档中检索一个或多个元素值

```
select
xmlquery('$CONTACT/Client/email')
  from clients
  where status = 'Gold'
```


XMLQUERY 函数能够让我把某个 XML 元素（例如 email ）检索为一个列。在本例中， email 被处理为一个列，但实际上，它是 XML 文档的一个元素。

然后是 **from client where status = 'Gold'** 。在本例中，我使用一个关系列来过滤我所需要的各行，我找回了一个 XML 元素。

在前面的例子中，我们做的是完全相反的事情。我们使用元素 zip 对各行进行过滤，我们找回了一个关系列。

如果你尝试这个查询时，当你获得结果时，会提示检索到 3 条记录，但你只能看到一个。这是因为前两个是空白。我们前面曾经说过，在 XML 中，没有 NULL 这个东西。在 XML 中，你得到的是空白。

利用 FLWOR 表达式检索 XML 数据

```
SELECT name,
   xmlquery ('for $e in $CONTACT/Client/email[1]
              return $e')
   FROM clients
   WHERE status = 'Gold'
```


这是另一个使用 XMLQUERY 实例，但在该函数中，我们使用了 FLWOR 表达式。我们使用了 For 。任何时候你看到 For ，都代表 FLWOR 表达式。它并不一定是完整的。在本例中，你实际上不必要使用 FLWOR 表达式；不使用 **For $e** 和 return **$e**，也一样能够返回该 email 。我们这里只是给你一个示例，以便让你看到，你可以把一个 FLWOR 表达式放到 XMLQUERY 中。

检索并把 XML 转化为 HTML

```
SELECT
   xmlquery('for $e in
$CONTACT/Client/email[1]/text()
             return <p>{$e}</p>')
FROM clients
WHERE status = 'Gold'
```


在本例中，要注意，我们使用了 和 。这是一个 HTML 标签，而 $e 将返回 XML 。因此，这个例子表明，你可以内嵌 / 结合 HTML 和 XML 。

XMLTABLE: 从 XML 到关系

```
select t.comment#,i.itemname,t.customerID,Message
  from items i,
  xmltable ('$COMMENTS/Comments/Comment'
    columns Comment#    integer      path 'CommentID',
            CustomerID integer       path 'CustomerID',
            Message    varchar(100) path 'Message') as t
```


这个函数称为 XMLtable，用于把 XML 转换为关系。举个例子，假如你们现有的系统用于处理关系模型，然后你们公司从另一家公司购买了一套软件产品，它使用 XML 来运行。现在你需要这两套系统进行交流。潜在而言，你可以把来自另一套系统的 XML 转变为关系，也就是把 XML 处理为表。因此，我们在这里调用 XMLtable 函数，我们一直查询到 Comment 元素，然后，再从这里开始，我们打算在本表中定义的列是 Comment#，被定义为 integer，路径中的最后一个元素是 CommentID 。

Comments/Comment/CommentID 。然后，对于 CustomerID，也是一样：integer， path，然后是 CustomerID 。对于 message，也是这样，虽然它是一个 Message 。

最后，我给它一个别名 t 。因此，基本上，我们进行 **select ... from items ... from t**，从两个表中检索。

XMLELEMENT: 从关系到 XML

```
select
  xmlelement (name "item",itemname),
  xmlelement (name "id", id),
  xmlelement (name "brand",brandname),
  xmlelement (name "sku",sku)
from items
where srp < 100
```

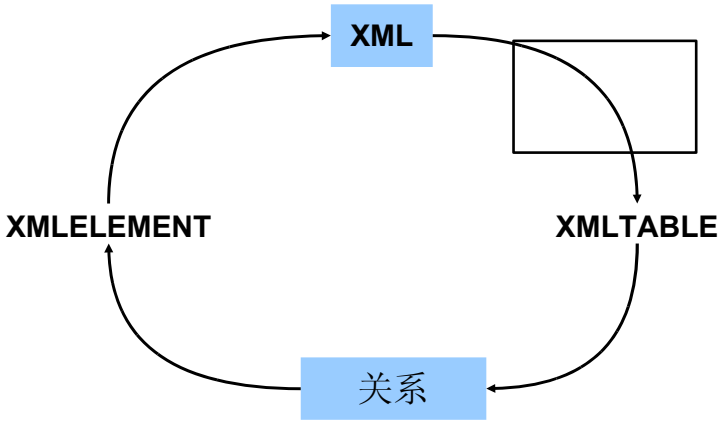
某些其他使用 XMLELEMENT 的函数有：

XMLNAMESPACES: 为所创建的元素指定一个名称空间
XMLATTRIBUTES: 为元素规定属性
XMLCONCAT: 把属性串联起来

下面我们来看看 XMLelement 函数，它类似于，或者说基本上与 XMLtable 相反。这个函数把关系转换为 XML，就是说，你把表中的全部信息转换到 XML 文档中。因此， XMLelement 是一个可以用来创建 XML 的函数。

还列出来其他一些函数，诸如 XMLNAMESPACES 、 XMLATTRIBUTES，等等，它们可以用来构建 XML 文档。

XMLTABLE 与 XMLELEMENT 的比较

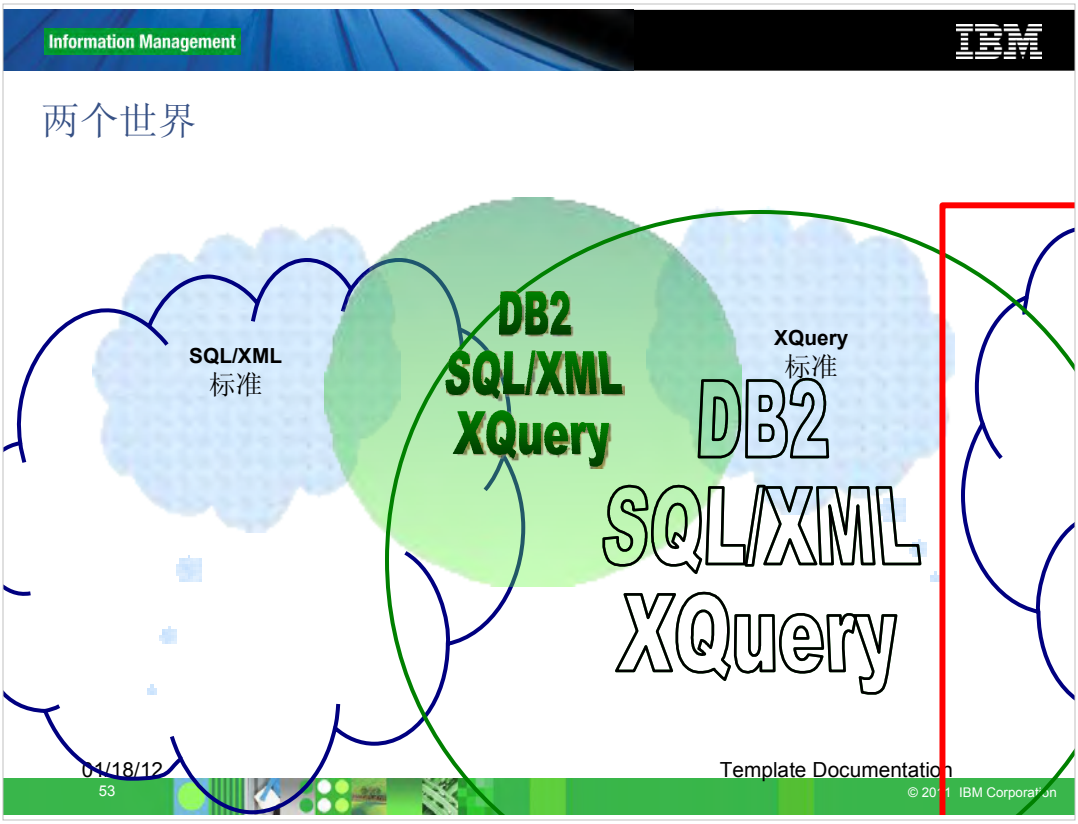


本图总结了前面两个函数的目的：

XMLtable 可以用来把 XML 转换为关系，为 XMLelement 用于把关系转换为 XML 。

议题

- 概述
- 插入 XML 数据
- XPath
- XQuery
- 利用 SQL/XML 查询 XML 数据
- 利用 XQuery 查询 XML 数据
- 对 XML 进行更新和删除操作
- XML 索引
- XML 模式验证
- 其他 XML 支持



我们现在谈谈第二部分，就是如何利用 XQuery 标准查询 XML 数据

XMLCOLUMN

xmlcolumn 函数带有一个参数，它指出了 XML 列的表名称和列名称。

一个用于返回客户联系数据的简单的 XQuery：

```
xquery
db2-fn:xmlcolumn('CLIENTS.CONTACT')
```

添加一个额外的过滤谓词

```
xquery
db2-fn:xmlcolumn
      ('CLIENTS.CONTACT')/Client/Address[zip="95116"]
```


在 XQuery 中，所有的代码都必须以关键字 XQuery 开头。如前所述，如果你加上这个关键字， DB2 将认为它是 SQL 。

XMLCOLUMN 函数用来检索 XML 列的所有内容，因此这实际上等价于在仅使用 SQL 时的 **select CONTACT from clients** 语句。

但是，在 SQL 中，这是我可以对 XML 做的全部事情，就是说， SQL 能够让我检索整个 XML 文档，但不能仅检索某一部分。利用 XMLCOLUMN 函数，你可以检索 XML 文档的某一部分。

XMLCOLUMN - 更多示例

带有 XMLCOLUMN 的 FLWOR 表达式用来检索客户传真数据：

```
xquery
  for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/fax
  return $y
```

输出样例：

```
<fax>4081112222</fax>
<fax>5559998888</fax>
```

查询 DB2 XML 数据并以 HTML 方式返回结果

```
xquery
<ul> {
for $y in db2-fn:xmlcolumn('CLIENTS.CONTACT')/Client/Address
order by $y/zip
return <li>{$y}</li>
}</ul>
```


这是在 FLWOR 表达式中使用 XMLCOLUMN 函数的另一个例子（上半部分）

下半部分是另一个 FLWOR 表达式，其中 return 使用了 HTML (
) 与 XML 相结合

XMLCOLUMN - 更多示例

```
<ul>
<li>
<address>
       <street>9407 Los Gatos Blvd.</street>
       <city>Los Gatos</city>
       <state>ca</state>
       <zip>95302</zip>
</address>
</li>
<Address>
      <street>4209 El Camino Real</street>
       <city>Mountain View</city>
       <state>CA</state>
       <zip>95302</zip>
</address>
</li>
...
</ul>
```


这是前面例子的输出样例

SQLQUERY: 在 XQuery 中嵌入 SQL

执行 SQL 查询并仅仅返回所查询数据的函数

从传递给 db2-fn:sqlquery 的查询中获得的结果集必须返回 XML 数据

该 XML 数据随后可以进一步由 XQuery 进行处理

```
xquery
for $y in
db2-fn:sqlquery('select comments from items where srp > 100')/Comments/Comment
where $y/ResponseRequested='Yes'
return (
<action>
      {$y//ProductID}
      {$y//CustomerID}
      {$y//Message}
</action>
)
```


SQLQUERY 能够让你把 SQL 嵌入 XQuery 中。我们在前面看到，当你使用 SQL/XML 时，你可以从 SELECT 语句入手，并在该语句中调用 XMLexists 、 XMLquery ，等等，这样， XML 函数就被嵌入 SQL 中。

现在，我在顶部使用 XQuery ，然后嵌入 SQL 。

这个查询是 select comments from items where srp>100 ，通过此语句，比如说，查询到 200 行。 comments 列必须是一个 XML 列。由此，我们进入 `/Comments/Comment` ，继续浏览各项元素。当然，这些名称对于解释而言可能不太好，但这些是不同的元素，由此开始我继续使用 XQuery 。

议题

- 概述
- 插入 XML 数据
- XPath
- XQuery
- 利用 SQL/XML 查询 XML 数据
- 利用 XQuery 查询 XML 数据
- 对 XML 进行更新和删除操作
- XML 索引
- XML 模式验证
- 其他 XML 支持

XML 的 DELETE 操作

根据某个使用 **XML** 元素的条件删除一行

使用 **SQL DELETE** 语句

DELETE 首先搜索文档，然后再删除它。搜索部分可以使用在查询数据时采用的 **SQL/XML** 函数来完成

```
delete from clients
  where
  xmlexists('$c/Client/Address[zip="95116"]'
                  passing CLIENTS.CONTACT as "c")
```

为了删除 **XML** 数据，请参阅下面的 **SQL UPDATE** 语句

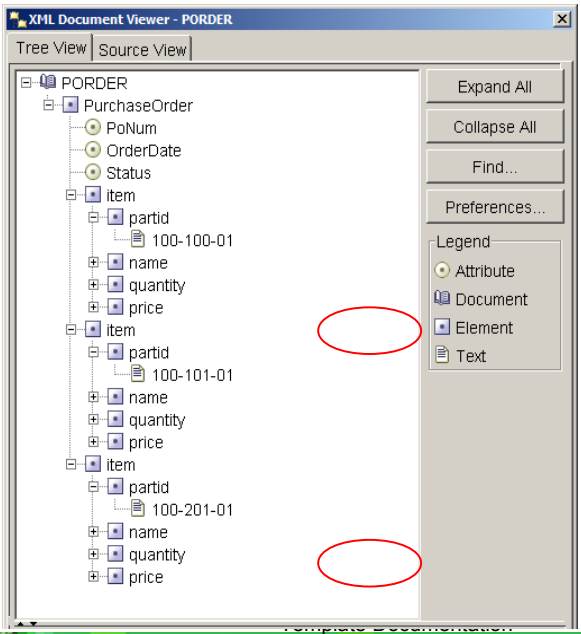
为了删除 XML 数据，你只需使用 SQL 即可，即利用 update 语句把 XML 列设置为 NULL 。

如果你打算根据 XML 文档的某种条件删除表中的列，你可以使用前面讨论的 SQL/XML 函数。这样的例子在前面已经介绍过。

如果你打算删除 XML 文档的某一部分，你实际上要做的是“更新” XML 文档，这将在“ Update” 部分绩效讨论，它使用 TRANSFORM 表达式。

XML 的 UPDATE 操作

- 使用 **TRANSFORM** 表达式
- 假如你有这样一份 **XML** 文档（来自 **SAMPLE** 数据库，**purchaseorder** 表）



这里说明了 UPDATE 操作如何在 XML 中进行。可以看到，本 XML 文档中有三个 item 元素。

UPDATE 示例

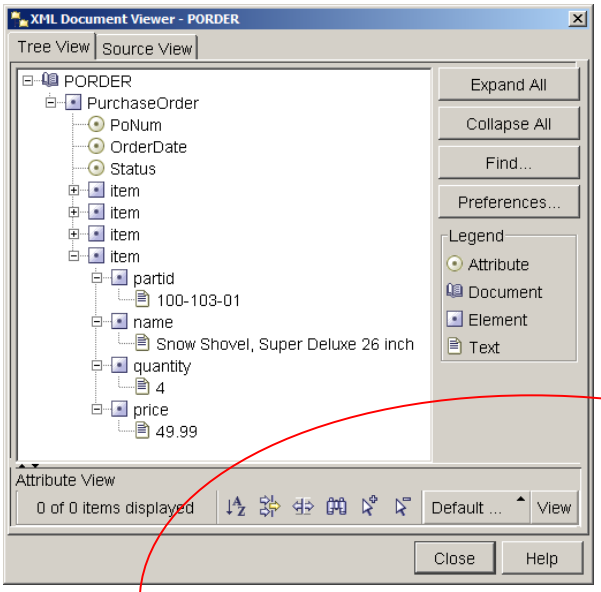
在文档末尾添加一个元素

```
UPDATE purchaseorder SET porder =
              xmlquery('transform
                        copy $po := $order
                        modify do insert
   document { <item>
               <partid>100-103-01</partid>
               <name>Snow Shovel, Super Deluxe 26 inch</name>
               <quantity>4</quantity>
               <price>49.99</price>
              </item>
            }
   into $po/PurchaseOrder
   return  $po'
   passing purchaseorder.porder as "order")
WHERE poid=5012;
```


在本例中，我们更新了 XML 文档，向其中添加了另一个 item 。

UPDATE 示例

在末尾添加元素之后：



本图示出了只需 update 之后的 4 个 item，其中最后添加的 item 已经展开了，而其他的则被收缩。

UPDATE 示例

在更新时删除某个 **XML** 元素

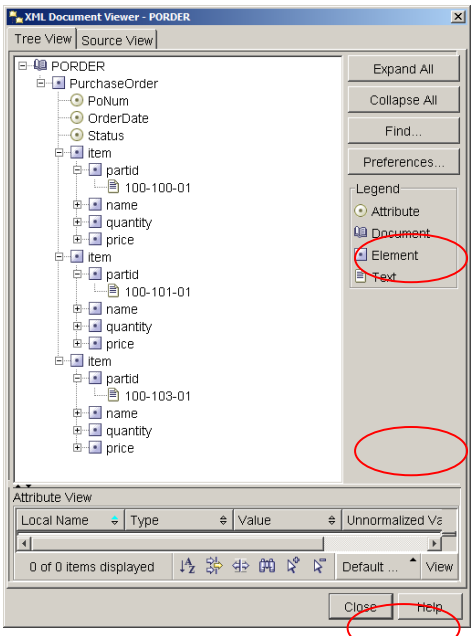
例如，删除 **partid** 为 **100-201-01** 的元素

```
UPDATE purchaseorder SET porder =
   xmlquery('transform
             copy $po := $order
             modify do delete
             $po/PurchaseOrder/item[partid = ''100-201-01'']
             return  $po'
             passing porder as "order")
WHERE poid=5012;
```


这是另一个例子，但在本例中，我们删除了 XML 文档中的一个元素。

UPDATE 示例

删除 **partid** 为 **100-201-01** 的元素之后



这显示元素被删除了。

XML 的 UPDATE 操作

你可以利用 **TRANSFORM** 做许多其他事情：

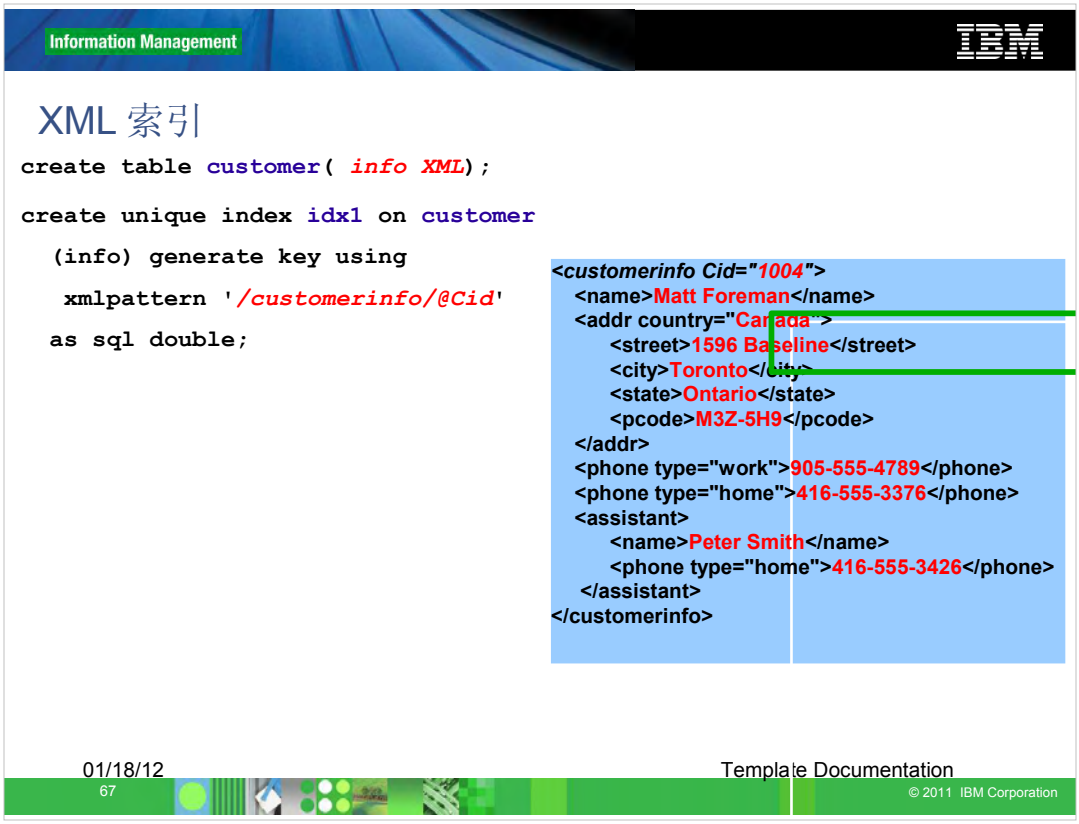
- 替换一个元素的值
- 替换一个属性的值
- 替换一个元素和属性
- 重命名一个元素和属性

例子来自于：

C:\Program Files\IBM\SQLLIB\samples\xml\xquery\clp\xupdate.db2

议题

- 概述
- 插入 XML 数据
- XPath
- XQuery
- 利用 SQL/XML 查询 XML 数据
- 利用 XQuery 查询 XML 数据
- 对 XML 进行更新和删除操作
- XML 索引
- XML 模式验证
- 其他 XML 支持



你可以根据 Xpath 表达式定义 XML 索引。

例如，如果你经常访问 Cid 属性，你就可以只针对该特定的属性创建一个 XML 索引。你可以使用该语法来实现： **create unique index ... generate key using XMLpattern** ，然后再使用一个 **Xpath** 语句指出该属性。

其他例子都相似，只是改变了 Xpath 。

XML 索引

```
create table customer( info XML);
```

```
create index idx2 on customer(info)
   generate key using
   xmlpattern '/customerinfo/name[1]'
   as sql varchar(40);
```

```
<customerinfo Cid="1004">
   <name>Matt Foreman</name>
   <addr country="Canada">
      <street>1596 Baseline</street>
      <city>Toronto</city>
      <state>Ontario</state>
      <pcode>M3Z-5H9</pcode>
   </addr>
   <phone type="work">905-555-4789</phone>
   <phone type="home">416-555-3376</phone>
   <assistant>
      <name>Peter Smith</name>
      <phone type="home">416-555-3426</phone>
   </assistant>
</customerinfo>
```

XML 索引

```
create table customer(info XML);
```

```
<customerinfo Cid="1004">
   <name>Matt Foreman</name>
   <addr country="Canada">
        <street>1596 Baseline</street>
        <city>Toronto</city>
        <state>Ontario</state>
        <pcode>M3Z-5H9</pcode>
   </addr>
   <phone type="work">905-555-4789</phone>
   <phone type="home">416-555-3376</phone>
   <assistant>
        <name>Peter Smith</name>
        <phone type="home">416-555-3426</phone>
    </assistant>
</customerinfo>
```

```
create index idx3 on customer(info)
generate key using
xmlpattern '//name'
as sql varchar(40);
```

这里的 **//text** 意味着你打算对所有的值创建索引，但不建议这样做，因为你创建出的索引会非常大。好比这样：如果你在使用一个表并创建关系索引，然后你对表中所有的列创建索引。不建议这样做。这个索引会非常大，而当你进行插入、更新或删除操作时，索引也会被更新，这会导致更多的性能问题。

议题

- 概述
- 插入 XML 数据
- XPath
- XQuery
- 利用 SQL/XML 查询 XML 数据
- 利用 XQuery 查询 XML 数据
- 对 XML 进行更新和删除操作
- XML 索引
- XML 模式验证
- 其他 XML 支持

XML 模式验证

利用**“ XML** 模式知识库**”**支持 **XML** 模式

为了根据 **XML** 模式进行验证，你可以：

- 在 **INSERT** 期间使用 **XMLVALIDATE** 函数
- 使用 **BEFORE** 触发器

为了测试某个 **XML** 文档是否得到了验证，你可以在 **CHECK** 约束中使用**“ IS VALIDATED”** 谓词

你可以利用 XML 模式对你的 XML 文档进行验证，这些模式可以存储在数据库的 XML 模式库中。

你可以在 insert 语句或者 BEFORE 触发器中使用 XMLVALIDATE 函数，以便根据你在 XML 模式库中注册的及存储的 XML 模式进行验证，这种验证是按行进行的。

可以使用具有 IS VALIDATED 谓词的 CHECK 约束，以便利用 XMLvalidate 函数检测某给定的列是否得到验证。

利用 XML 模式进行验证（示例）

```
INSERT INTO t1 VALUES(xmlvalidate(xmlparse(document('<?xml
  version="1.0" encoding="UTF-8"?>
        <po:PurchaseOrder xmlns:po="http://www.test.com/po">
          <Header>
               <Id>1</Id>
                  ...
          </Header>
          <Items>
            ...
              <Item>
                       ...
              </Item>
          </Items>
          <Customer type="regular">
                  ...
          </Customer>
     </po:PurchaseOrder>')) ACCORDING TO XMLSCHEMA ID order));
```

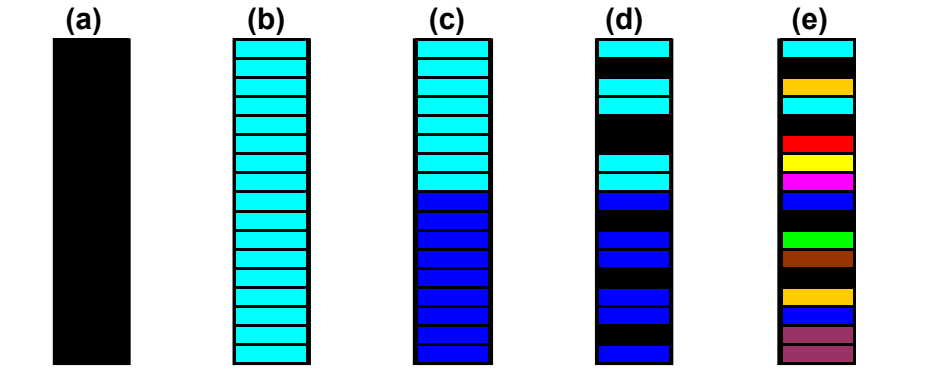


```
DROP TABLE t1;
CREATE TABLE t1 (po xml);

INSERT INTO t1 VALUES(xmlvalidate(xmlparse(document('<?xml version="1.0" encoding="UTF-8"?>
        <po:PurchaseOrder xmlns:po="http://www.test.com/po">
          <Header>
            <Id>1</Id>
            <date>2004-01-29</date>
            <description>purchase order</description>
            <value>20</value>
            <status>shipped</status>
          </Header>
          <Items>
            <Item>
              <ItemDescription color="red" weight="5">
                  <Name>Widget C</Name>
                  <SKU>1</SKU>
                  <Price>30</Price>
                  <Comment>no comment</Comment>
              </ItemDescription>
              <NumberOrdered>1</NumberOrdered>
            </Item>
          </Items>
          <Customer type="regular">
            <Name>Manoj K Sardana</Name>
            <Address>ring road, bangalore</Address>
            <Phone>918051055109</Phone>
            <email>msardana@in.ibm.com</email>
          </Customer>
              </po:PurchaseOrder>')) ACCORDING TO XMLSCHEMA ID order));
```

pureXML 模式的灵活性

按照 XML 列对零个、一个或者多个模式进行文档验证：

总是 具有良好格式的 **XML**



多数数据库仅支持 **(a)** 和 **(b)**。 **DB2** 支持 **(a)** 到 **(e)**。

这张幻灯片说明你可以利用不同的 XML 模式根据 XML 列验证不同的行

议题

- 概述
- 插入 XML 数据
- XPath
- XQuery
- 利用 SQL/XML 查询 XML 数据
- 利用 XQuery 查询 XML 数据
- 对 XML 进行更新和删除操作
- XML 索引
- XML 模式验证
- 其他 XML 支持

其他 XML 支持

对小型 **XML** 文档进行基于表的内联和压缩

可以使用 **XSLT** 功能转换 **XML** 文档

利用 **UPDATE XMLSCHEMA** 命令实现兼容的 **XML** 模式演化

pureXML 支持 **UNICODE** 或非 **UNICODE** 数据库

有注释的 **XML** 模式分解

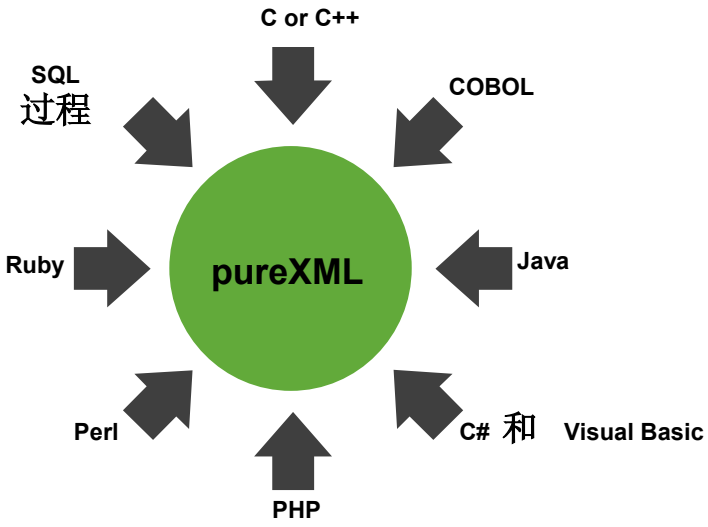
正常情况下， XML 是以不同的对象来存储的，我们在表中有一个指针指向该对象。对小型 XML 文档进行基于表的内联和压缩意味着，如果 XML 文档不是很大，而是可以装入相同的表中，那么， DB2 将把它存储在该基本表中，并对 XML 文档进行行压缩。

还提供了对 XSLT 功能的支持，这样你就可以转换你的 XML 文档，以便采用不同的格式，例如 HTML 格式。你也可以使用 CSS （级联样式表），这样，数据可以非常优雅地展示出来。

利用 UPDATE XMLSCHEMA 命令实现兼容的 XML 模式演化意味着，你的模式可以随时间推移不断演化。或许某个字符串在过去有给定的长度，但现在它已经变化了；现在它变大了，因此如果运行这个 update XML schema 语句，只要它不是一个完全不同的架构，该架构就可以演化。

我们依然支持 XML 模式分解，如果你打算不使用 pureXML 来存储 XML ，而是实际上把 XML 分解为小碎片并把各个碎片存储在表中。

对的 XML 数据的开发支持



pureXML 能够支持所有这些语言。已经为实现此种支持修改了 / 更新了所有的驱动器

如果你对本课程所讨论的内容有什么技术方面的问题的话，可以利用 db2university.com 课程 AA001EN 中的论坛。 同学、教员以及 IBM 人员可以帮助你！

Information Management

IBM

DB2 应用开发概述

IBM 信息管理云计算能力中心
IBM （加拿大）研究院

议题

DB2 应用开发概述

服务器端开发

- 存储过程
- 用户定义的函数
- 触发器

客户端开发

- 嵌入式 **SQL**
- 静态与动态 **SQL** 对比
- **CLI/ODBC**
- **JDBC / SQLJ / pureQuery**

支持性阅读材料和视频

阅读材料

DB2 Express-C 入门电子书

第 14 章 : DB2 应用开发简介

IBM Data Studio for DB2 入门电子书

第 5 章 : 开发 SQL 存储过程

第 7 章 : 开发用户定义的函数

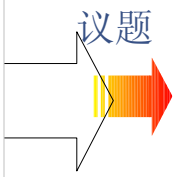
DB2 应用开发入门电子书

第 3 章，第 3.6 节 : 触发器 : 概述

视频

db2university.com 课程 AA001EN

第 12 课 : DB2 应用开发



议题

- **DB2** 应用开发概述
- 服务器端开发
  - 存储过程
  - 用户定义的函数
  - 触发器
- 客户端开发
  - 嵌入式 **SQL**
  - 静态与动态 **SQL** 对比
  - **CLI/ODBC**
  - **JDBC / SQLJ / pureQuery**

DB2 应用开发概述

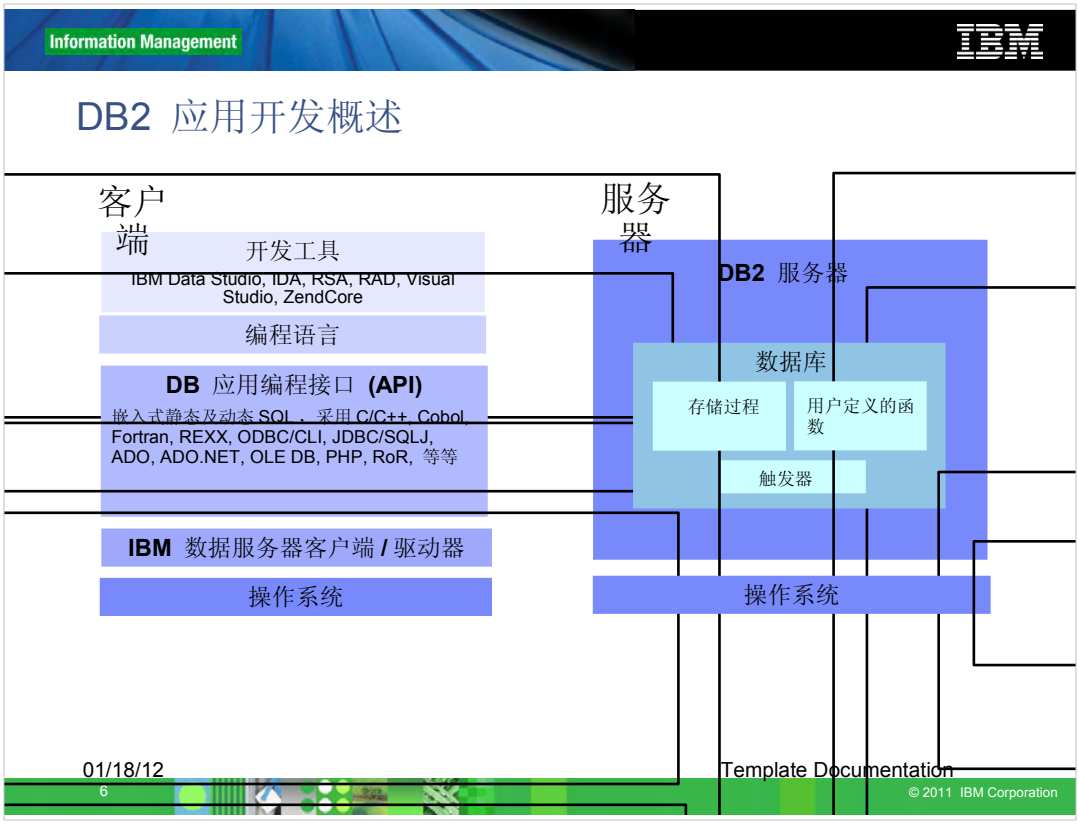
服务器端开发（在 **DB2** 数据库服务器上）：

- 例程（存储过程、用户定义的函数）
- 数据库对象（触发器）

客户端开发（在客户端上）：

- 可能需要安装 DB2 客户端或驱动器
- 数据库应用（使用 C/C++ 、 .NET 、 Cobol 、 Java 等语言）

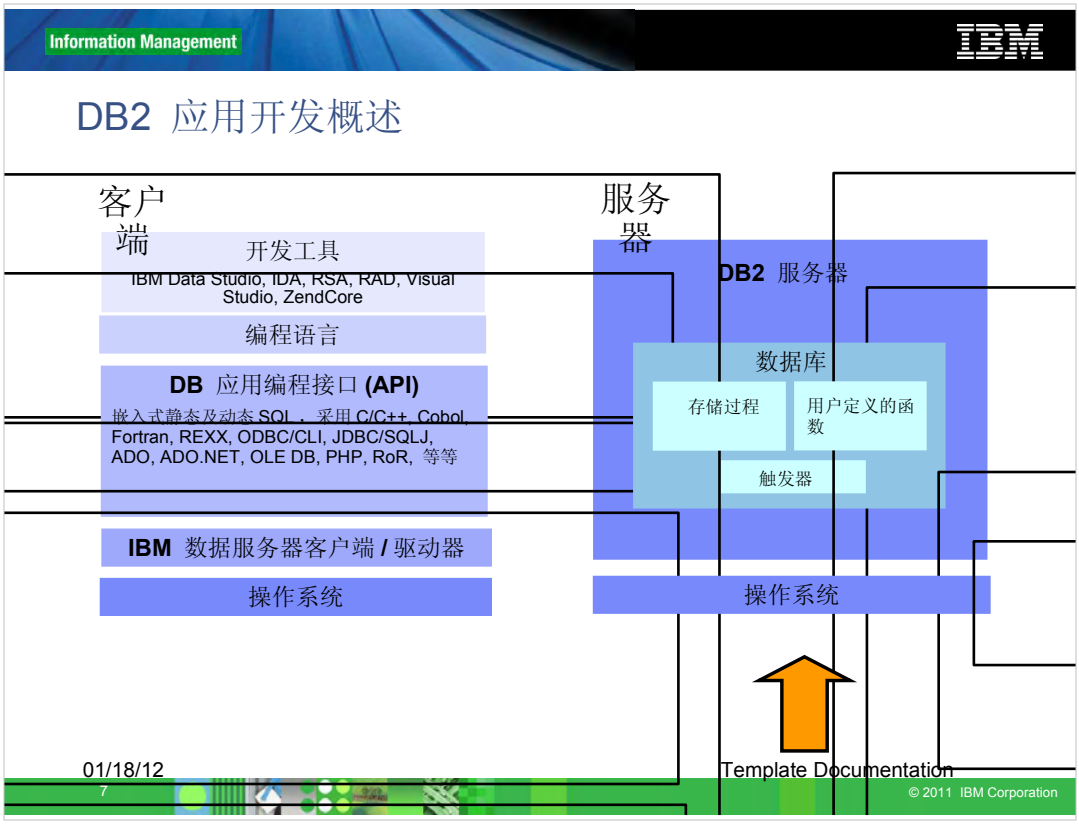
DB2 应用开发分为两部分，第一部分是服务器端开发（存储过程、用户定义的函数，等等），第二部分是客户端开发（例如：从 Java 程序中访问 DB2 ）



本图中，左侧代表客户机，应用程序员在这里开发和运行他的程序。在这个客户机中，除了操作系统之外，或许还要安装 IBM 数据服务器客户端，这取决于所开发的应用的类型。IBM 数据服务器客户端包含了所需要的连接驱动器，例如JDBC驱动器以及ODBC/CLI驱动器。这些驱动器也可以从 IBM DB2 Express-C网站上单独下载，地址为： ibm.com/db2/express

通过使用IBM Data Studio、InfoSphere Data Architect (IDA)、Rational Software Architect (RSA)、Rational Application Developer (RAD) 等编程工具，你可以利用你喜欢的语言开发你的应用。IBM 数据服务器客户端中也含有支持这些语言的API库，这样，当你接入DB2 服务器时，所有的程序指令都利用这些API适当地转换为DB2所理解的SQL或者XQuery语句。

图的右侧是DB2 服务器，它含有一个数据库。在该数据库中，有存储过程、用户定义的函数和触发器。我们将在后面详细介绍这些对象，我们首先来集中讨论这一部分。



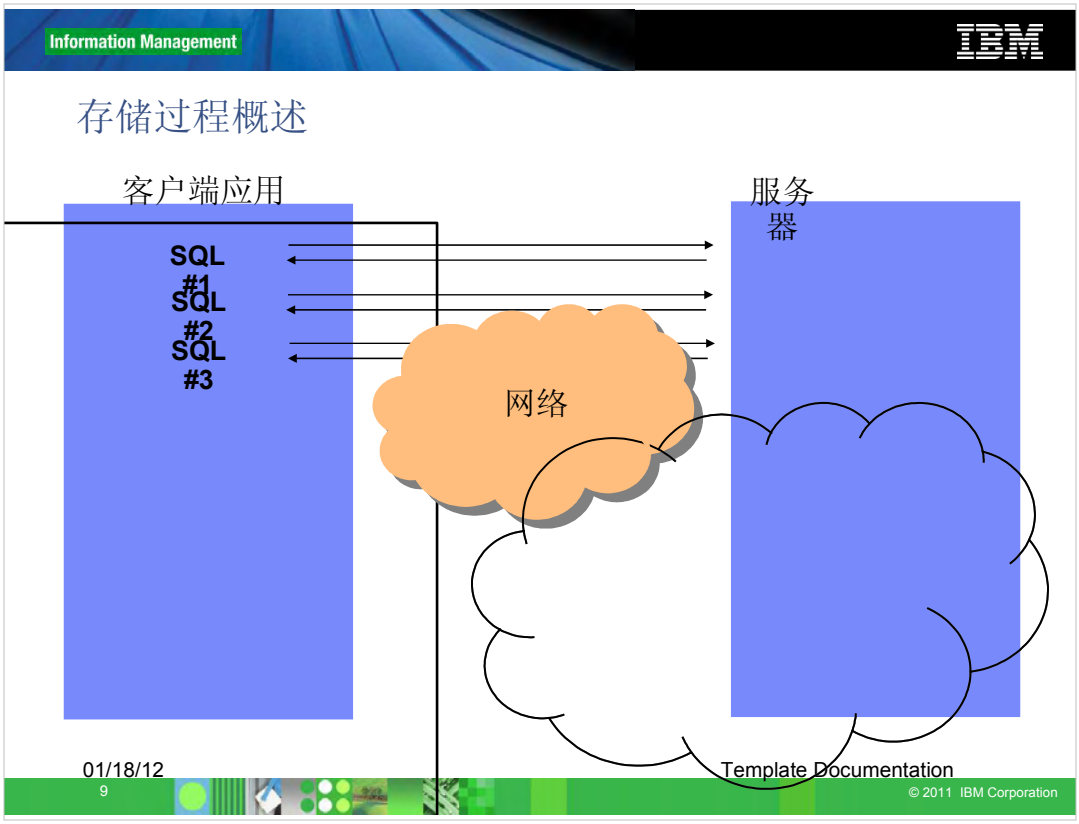
本图中，左侧代表客户机，应用程序员在这里开发和运行他的程序。在这个客户机中，除了操作系统之外，或许还要安装 IBM 数据服务器客户端，这取决于所开发的应用的类型。IBM 数据服务器客户端包含了所需要的连接驱动器，例如JDBC驱动器以及ODBC/CLI驱动器。这些驱动器也可以从 IBM DB2 Express-C网站上单独下载，地址为： ibm.com/db2/express

通过使用IBM Data Studio、InfoSphere Data Architect (IDA)、Rational Software Architect (RSA)、Rational Application Developer (RAD) 等编程工具，你可以利用你喜欢的语言开发你的应用。IBM 数据服务器客户端中也含有支持这些语言的API库，这样，当你接入DB2 服务器时，所有的程序指令都利用这些API适当地转换为DB2所理解的SQL或者XQuery语句。

图的右侧是DB2 服务器，它含有一个数据库。在该数据库中，有存储过程、用户定义的函数和触发器。我们将在后面详细介绍这些对象，我们首先来集中讨论这一部分。

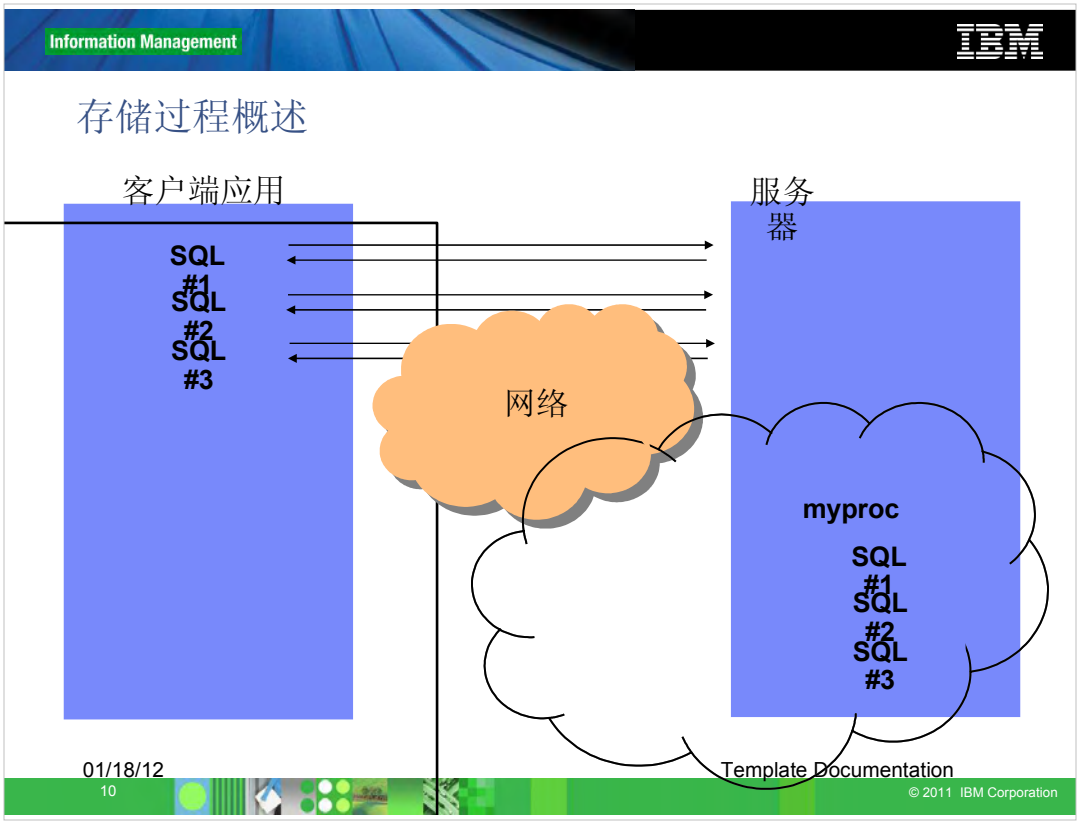
议题

- DB2 应用开发概述
- 服务器端开发
  - 存储过程
  - 用户定义的函数
  - 触发器
- 客户端开发
  - 嵌入式 SQL
  - 静态与动态 SQL 对比
  - CLI/ODBC
  - JDBC / SQLJ / pureQuery

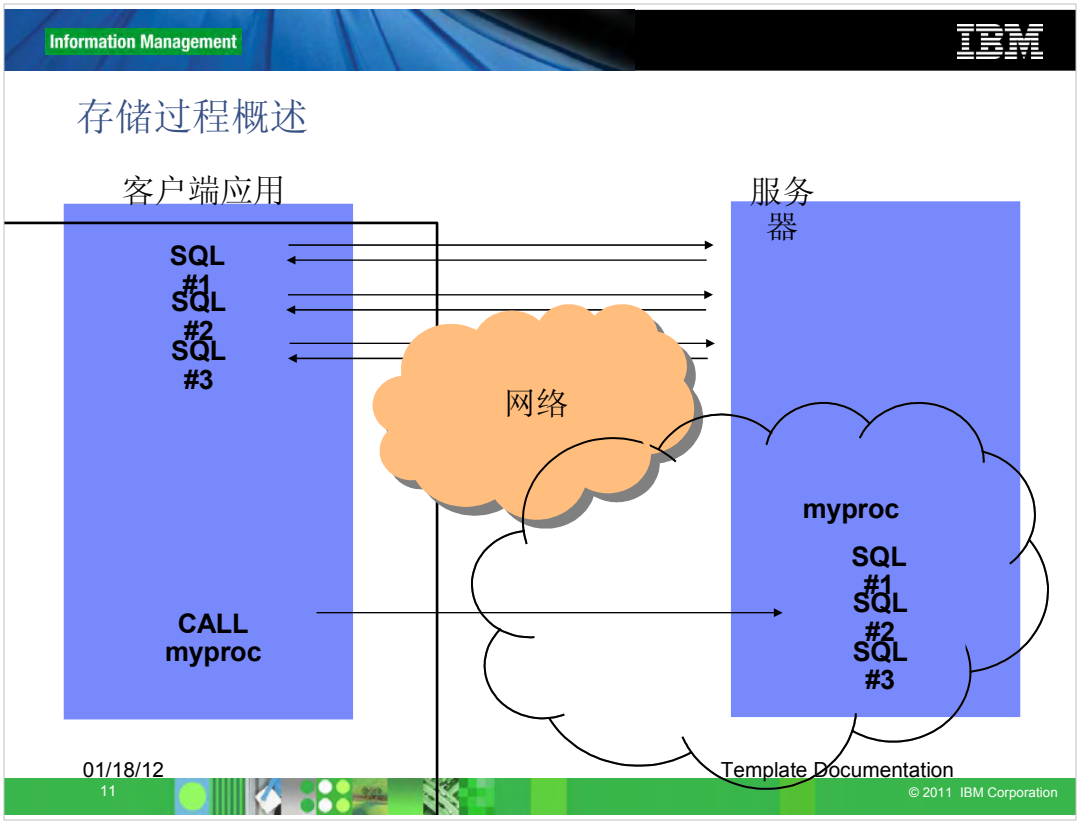


本图解释了存储过程的工作原理以及为什么需要它们。

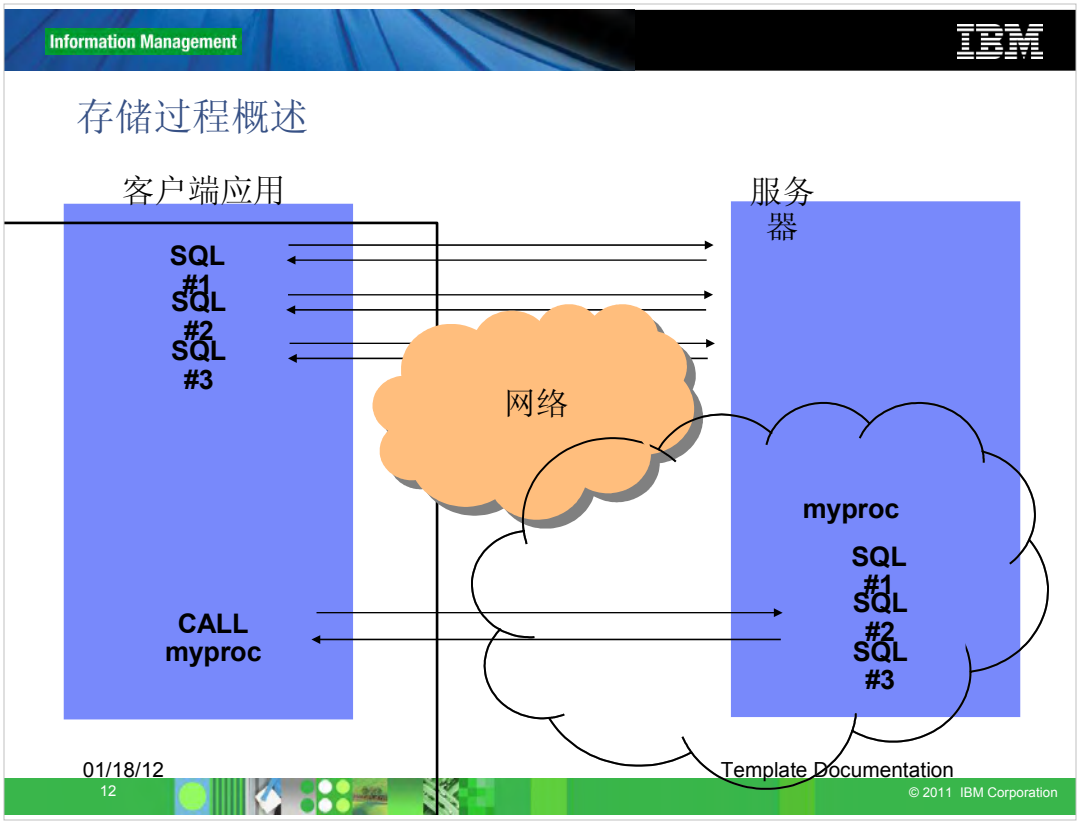
图的左上角，你可以看到有多个 SQL 语句先后执行。每个 SQL 被从客户端送入数据服务器，然后数据服务器把结果返回给客户端。如果许多 SQL 语句都像这样执行，网络流量就会急剧增加。



另一方面，在底部，你可以看到一种替代方法，它产生的网络流量较少。这第二种方法调用存储在服务器上的存储过程 ***myproc***，它里面含有相同的 SQL；



而在客户端（左侧）， CALL 语句用来调用该存储过程。



第二种方法更有效率，因为只有一个 call 语句在网络之间传递，最后把一个结果集返回给客户端。

存储过程也有助于提高数据库的安全性。例如，你可以让用户仅通过存储过程来访问表和视图；这有助于锁定服务器，使得用户无法访问不该访问的信息。这之所以能够实现，是因为用户不需要对他们通过存储过程访问的表或视图拥有显式的特权；他们只需要被授予足够的特权来调用存储过程即可。

存储过程概述

通常包含一个或多个 SQL 语句以及程序（业务）逻辑

由 DB2 执行和管理（服务器端对象）

可以使用 SQL PL 、 C/C++ 、 Java 、 Cobol 、 CLR 支持的语言、 OLE 、 PL SQL 等进行编写

采用存储过程的好处包括 :

- 有助于代码重用的集中的业务逻辑
- 提高安全性
- 提高性能

本研讨班主要讨论 SQL PL 过程，因为它非常流行，而且具有良好的性能，也比较简单

创建你的第一个存储过程

使用命令行处理器：

```
db2=> connect to sample
db2=> create procedure p1 begin end
```

使用 IBM Data Studio
（演示）

通过 CLP 创建的过程什么事情也做不了。它只是用于解释的目的。要注意，你首先需要接入数据库，因为这是存储过程驻留的地方。

基本的存储过程结构

CREATE PROCEDURE proc_name *[({ 可选参数 })]*
[可选过程属性]
< 语句 >

[可选参数]

IN	输入参数
OUT	输出参数
INOUT	输入和输出参数

示例：

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

基本存储过程结构

[可选过程属性 *]*
LANGUAGE SQL
RESULT SETS <n> （在返回结果集时需要）

< 语句 > 可以是单一语句，也可以是由 BEGIN [ATOMIC] ... END 分组的一系列语句

LANGUAGE SQL

这个属性指出了存储过程将使用的语言。LANGUAGE SQL 是默认值。对于其他语言，例如 Java 或 C，则分别使用LANGUAGE JAVA 或 LANGUAGE C。

RESULT SETS <n>

如果你的存储过程将返回 n个结果集，就需要这个属性。

存储过程中的复合语句是由多个过程指令和 SQL 语句构成的语句，由关键字 BEGIN 和 END 所包围。如果 ATOMIC 关键字跟在 BEGIN 关键字之后，该复合语句就被视为一个单元，就是说，只有该复合语句中的所有指令或语句都成功运行了，整个复合语句才算是成功了。如果有一个语句运行不成功，则所有的操作都被回滚。

注：必须遵守所示顺序。

例如，如果你在声明变量之前先声明游标，你就会得到一条错误消息。

变量声明和赋值

DECLARE var_name <data type> [DEFAULT value]

示例：

```
DECLARE temp1 SMALLINT DEFAULT 0;
DECLARE temp2 VARCHAR(10) DEFAULT 'hello';
DECLARE temp3 DATE DEFAULT '1998-12-25';
```

SET var_name = value

示例：

SET total = 100;

等同于 **VALUES(100) INTO total;**

SET total = NULL;

任何变量都可以设置为 **NULL**

SET total = (select sum(c1) from T1);

如果存在一行以上的话，则给出条件

SET first_val = (select c1 from T1 fetch first 1 row only)

仅从表中抓取第一行

SET sch = CURRENT SCHEMA;

示例：带有参数的存储过程

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

从命令行处理器上调用该过程：

```
db2=> call P2 (3, 4, ?)
```


注：

在从应用中（Data Studio 除外）调用存储过程时，需要用一个问号来代表 OUT 参数。

示例：处理游标的存储过程

```
CREATE PROCEDURE sum_salaries(OUT sum INTEGER)
LANGUAGE SQL
BEGIN
  DECLARE p_sum INTEGER;
  DECLARE p_sal INTEGER;
  DECLARE SQLSTATE CHAR(5) DEFAULT '00000';
  DECLARE c CURSOR FOR
            SELECT SALARY FROM EMPLOYEE;
  SET p_sum = 0;
  OPEN c;
  FETCH FROM c INTO p_sal;
  WHILE(SQLSTATE = '00000') DO
     SET p_sum = p_sum + p_sal;
     FETCH FROM c INTO p_sal;
  END WHILE;
  CLOSE c;
  SET sum = p_sum;
END
```

SQLCODE 和 SQLSTATE

访问时需要显式声明：

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

只能在最外层声明，在每次运行后由 DB2 自动设置

SQLCODE

- **= 0,** 成功
- **> 0,** 成功，但有警告
- **< 0,** 不成功
- **= 100,** 未找到数据
  - 例如， **FETCH** 语句未返回数据

SQLSTATE

- **SQLSTATE '00000' =** 成功
- **SQLSTATE '02000' =** 未找到
- **SQLSTATE '01XXX' =** 警告
- 任何其他 **=** 异常

示例：从 Java 应用中调用存储过程

```
try
{
  // Connect to sample database
  String url = "jdbc:db2:sample";
  con = DriverManager.getConnection(url);
  CallableStatement cs = con.prepareCall("CALL
  trunc_demo(?, ?)");
  // register the output parameters
  callStmt.registerOutParameter(1, Types.VARCHAR);
  callStmt.registerOutParameter(2, Types.VARCHAR);
  cs.execute();
  con.close();
}
catch (Exception e)
{
  /* exception handling logic goes here */
}
```

更多的例子见于：

C:\Program Files\IBM\SQLLIB\samples

议题

- DB2 应用开发概述
- 服务器端开发
  - 存储过程
  - 用户定义的函数
  - 触发器
- 客户端开发
  - 嵌入式 SQL
  - 静态与动态 SQL 对比
  - CLI/ODBC
  - JDBC / SQLJ / pureQuery

用户定义的函数

函数一定会返回值

已有一些系统内置函数，“开箱即用”

例如 : SUM(), AVG(), DIGITS(), 等等

可以使用下述语言创建用户定义的函数 :

SQL PL, C/C++, Java, CLR, OLE, 等等

在本研讨班中，我们主要讨论 SQL PL 函数，因为它简单、流行

用户定义的函数 (UDF) 是一种数据库应用对象，它把一组输入值映射为一组输出值。例如，一个函数可以用英寸测量值作为输入，然后以厘米返回结果。

DB2 支持使用 SQL PL 、 PL/SQL 、 C/C++ 、 Java 、 CLR （通用语言运行时）和 OLE （对象链接和嵌入）（即： .NET 技术）来创建函数。

你可以把 UDF 视为一种允许用户根据其需要扩展 SQL 语言的方式。你从 **SELECT** 或 **VALUES** 语句中调用函数。

函数的类型

- 标量函数
  - 返回一个值
  - 不能改变数据库状态（即，不允许执行 INSERT 、 UPDATE 、 DELETE 语句）
    - 示例 : COALESCE( ), SUBSTR( )
- 表函数
  - 以表的格式返回值
  - 在查询语句的 FROM 子句中调用
  - 可以改变数据库的状态（即，允许执行 INSERT 、 UPDATE 、 DELETE 语句）
  - 示例 : SNAPSHOT_DYN_SQL( ), MQREADALL( )
- 其他类型的函数（本课程不讨论）：
  - 行函数
  - 列函数

标量函数

标量函数获得输入值并返回一个值
它们不得被用来修改表数据

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


在上面的程序中，函数名称是***deptname***，它根据 employee id 返回 employee 的 department号码。

调用标量函数

标量的用户定义函数可以被需要标量值的 SQL 语句所调用，或者用在 VALUES 子句中

```
SELECT DEPTNAME('000010') FROM SYSIBM.SYSDUMMY1

VALUES DEPTNAME('000010')
```


标量 UDF 可以使用 VALUES 语句调用。它也可以由需要获得标量值的SQL语句调用。例如，可以从DB2命令窗口或者从Linux或UNIX终端试试执行如下命令：

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

或者

db2 "select (deptname ('000300')) from sysibm.sysdummy1"

注意，第二个例子使用了SYSIBM.SYSDUMMY1。这是一个特殊的哑表，它只有一行、一列，它用于确保只有一个值被返回。如果你使用任何其他有多行的表来执行这个SELECT语句，则该此表有多少行，函数就被调用多少次。

表 UDF （用户定义的函数）

返回一个表

用于查询语句的 FROM 子句中

通常用于返回一个表及保存审计记录

示例：用于列举部门中雇员的函数

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

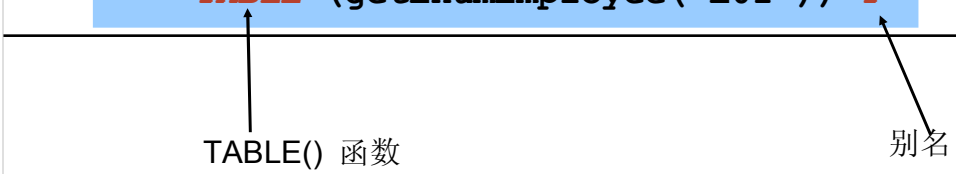

表函数返回包含许多列的表。你可以使用查询语句中的FROM子句调用它们。与标量函数不同的是，表函数可以修改数据库的状态；因此，允许使用INSERT、UPDATE和DELETE语句。有一些内置的表函数，例如SNAPSHOT_DYN_SQL( ) 和 MQREADALL( )。表函数类似于视图，但表函数允许使用数据修改语句（INSERT、UPDATE和DELETE），所有更为强大。它们通常用于返回一个表及保存审计记录。

调用 UDF

在 SQL 语句的 FROM 子句中使用

必须应用 TABLE() 函数，必须起别名。

```
SELECT * FROM
    TABLE (getEnumEmployee('E01')) T
```



表UDF必须从SQL语句的FROM子句中调用，因为它返回表。必须使用特殊的TABLE()函数，而且调用之后必须提供别名。

议题

- DB2 应用开发概述
- 服务器端开发
  - 存储过程
  - 用户定义的函数
  - 触发器
- 客户端开发
  - 嵌入式 SQL
  - 静态与动态 SQL 对比
  - CLI/ODBC
  - JDBC / SQLJ / pureQuery

触发器

触发器是针对表定义的数据库对象，在进行 INSERT 、 UPDATE 或 DELETE 操作时被触发。

自动激活（“启动”）

导致触发器启动的操作被称为触发 SQL 语句

触发器是与表相关联的数据库对象，它们定义了在对表进行INSERT、UPDATE或DELETE时将进行的操作。触发器是自动激活的。导致触发器启动的操作被称为“触发SQL语句”。

触发器的类型

BEFORE

在插入、更新或删除某一行之前被激活

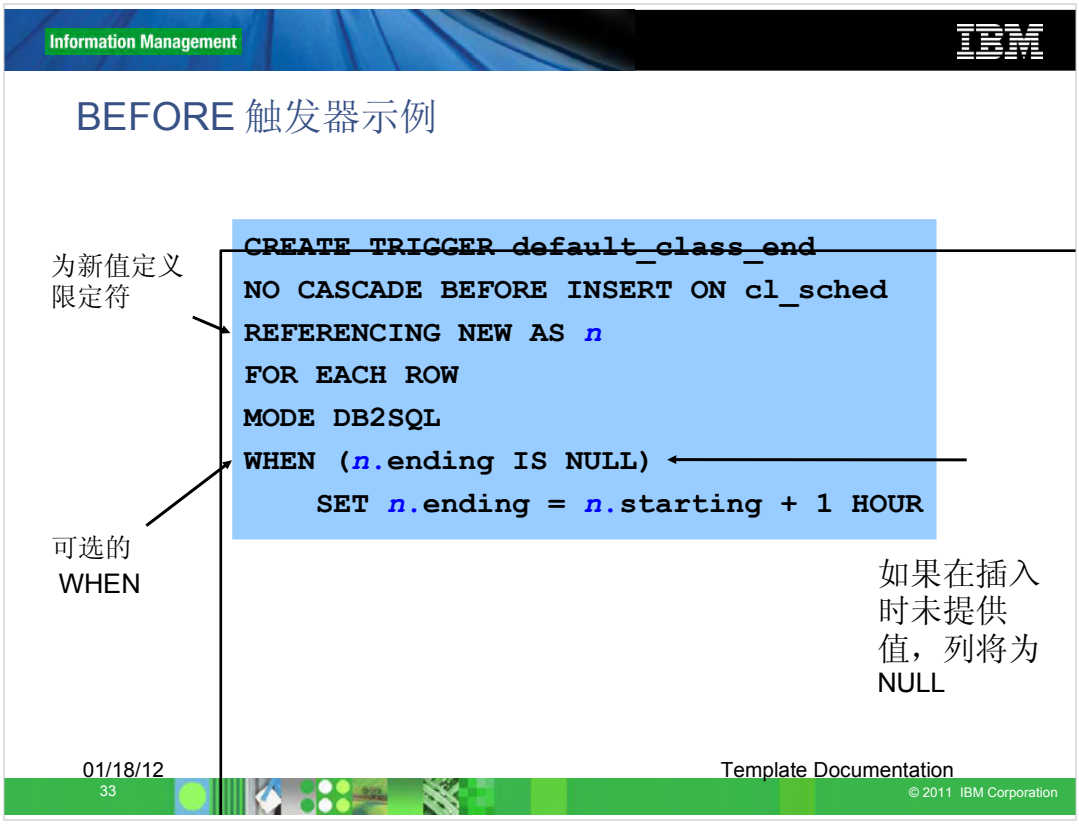
AFTER

在触发的 SQL 语句已经执行并成功完成之后被激活

INSTEAD OF

针对视图定义

在触发器中定义的逻辑得到执行，而不是 (*instead of*) 触发的 SQL 语句得到执行



触发器***default_class_end*** 将在对表 CL_SCHED 执行 INSERT SQL 语句之前被触发。此表是SAMPLE数据库的一部分，因此，你可以创建该触发器并在接入该数据库之后亲自测试该触发器。触发器定义中的变量 ***n*** 代表了插入INSERT语句中的新值，也就是待插入的值。此触发器将检查被插入到表中的数据的有效性。如果列ENDING在INSERT语句中没有值的话，触发器将保证它具有列STARTING加1小时这样的值。为测试该触发器，你可以执行：

db2 insert into cl_sched (class_code, day, starting) values ('abc',1,current time)

db2 select * from cl_sched

AFTER 触发器示例

类似于 BEFORE 触发器，除了 INSERT 、 UPDATE 和 DELETE 得到支持之外

前提条件：

CREATE TABLE audit (mytimestamp timestamp, comment varchar (1000))

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


触发器***audit_emp_sal*** 用于对EMPLOYEE 表的SALARY列进行审计。当有人对该列进行修改时，触发器将被激活，向另一个称为AUDIT的表中写入有关对salary进行修改的信息。而OLD as o NEW as n 这一行说明，前缀 ***o*** 将被用来代表该表中的旧值或现有值，而前缀 ***n*** 将被用来代表来自UPDATE语句的新值。因此，***o.salary*** 代表了salary的旧值或现有值，而 ***n.salary*** 代表了salary列的更新后的值。

INSTEAD OF 触发器示例

在修改视图时激活它

前提条件：

```
CREATE TABLE countries (id int, country varchar(50),
                        region varchar (50), average_temp int)
CREATE VIEW view1 (id, continent, temperature) as
       SELECT id, region, average_temp from countries
```

```
CREATE TRIGGER update_view1
       INSTEAD OF UPDATE ON view1
       REFERENCING OLD AS o NEW AS n
       FOR EACH ROW
       MODE DB2SQL
       BEGIN ATOMIC
          UPDATE countries
          SET region = n.region
          WHERE region = o.region;
       END
```


Instead of 触发器是为视图定义的。由于视图是利用SELECT语句动态定义的，而且会访问一个或多个表，因此视图无法被更新。但是，通过这类触发器，你可以让用户感觉到视图被更新了，因为触发器中定义的逻辑得到执行，而不是触发SQL语句。例如，如果你对某个视图进行一次更新操作，instead of 触发器将被激活，它实际上对构成此视图基本表进行了更新。

触发器无法从IBM Data Studio中创建，除非你使用“Script”文件夹。它们可以从控制中心或者从命令行工具（DB2命令窗口、命令行处理器）中创建。

议题

- DB2 应用开发概述
- 服务器端开发
  - 存储过程
  - 用户定义的函数
  - 触发器
- 客户端开发
  - 嵌入式 SQL
  - 静态与动态 SQL 对比
  - CLI/ODBC
  - JDBC / SQLJ / pureQuery

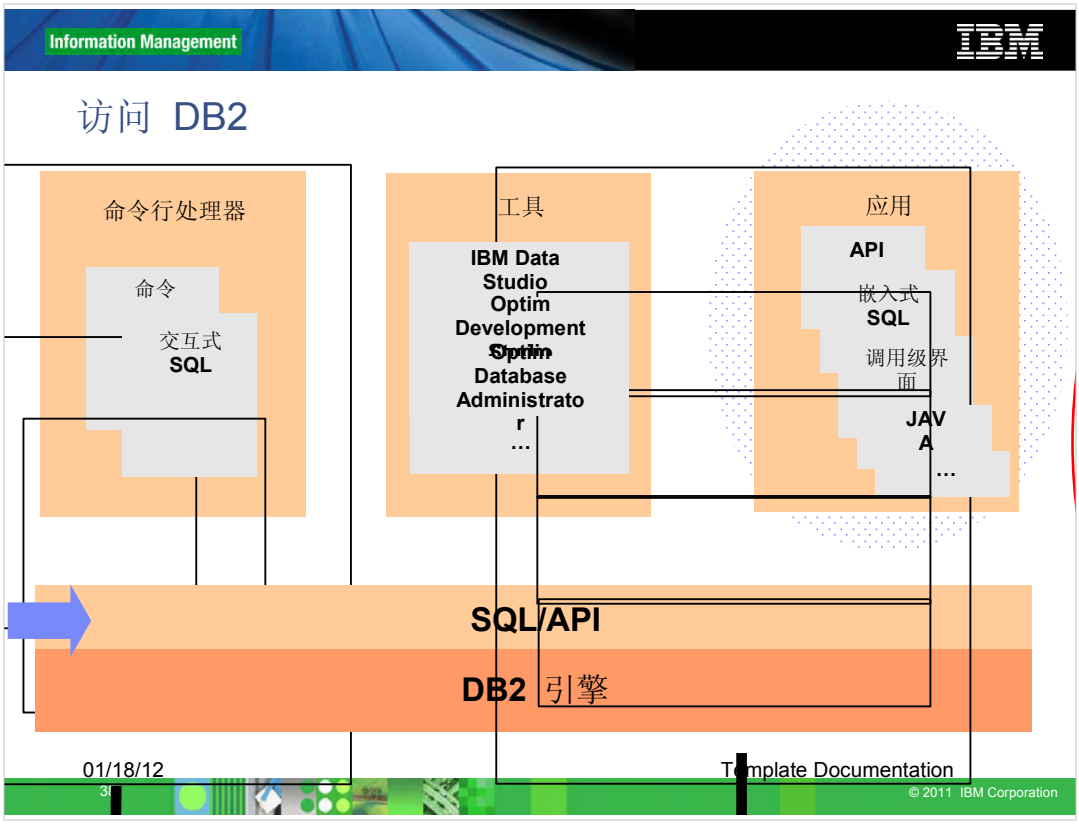
DB2 应用开发概述

服务器端开发（在 **DB2** 数据库服务器上）：

- 例程（存储过程、用户定义的函数）
- 数据库对象（触发器）

客户端开发（在客户端上）：

- 可能需要安装 DB2 客户端或驱动器
- 数据库应用（使用 C/C++ 、 .NET 、 Cobol 、 Java 等语言）



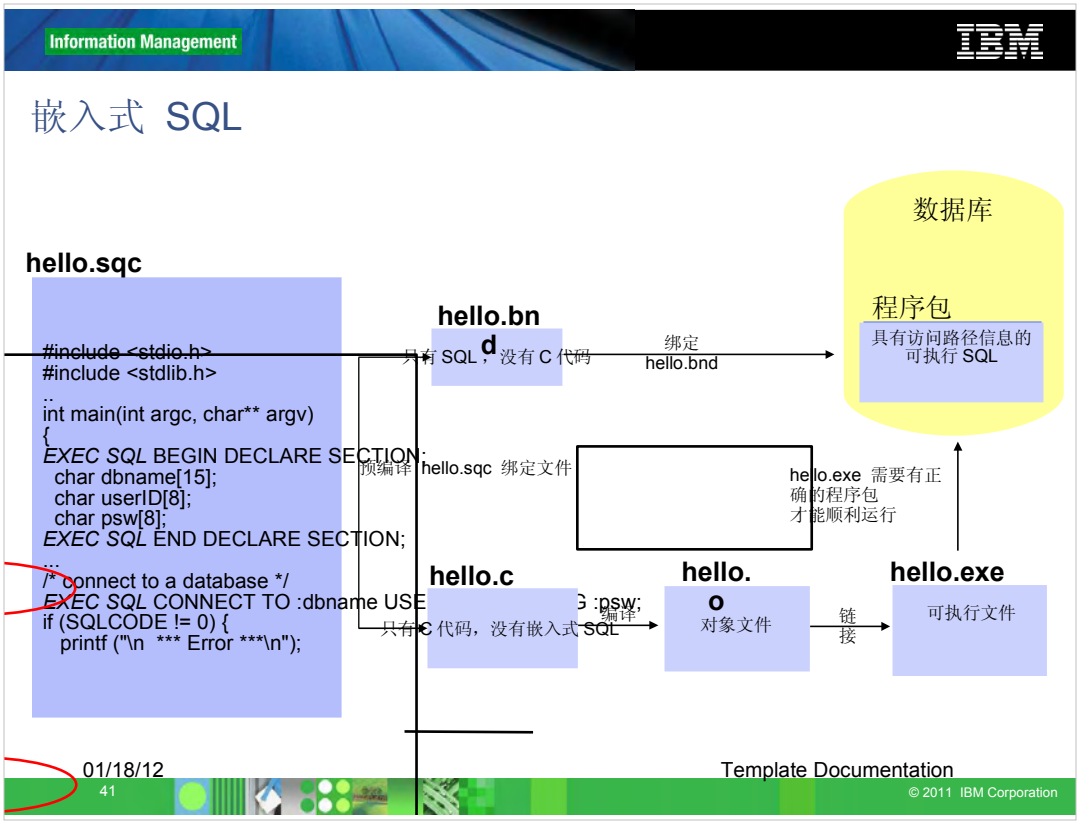
最后， SQL/API 层把所有的内容转换为 DB2 能够理解的 SQL 。

应用开发自由度

Ruby on Rails
C/C++ (ODBC 和静态 SQL)
JDBC 和 SQLJ
Borland
Python
PHP
Perl
.NET 语言语言
OLE-DB
ADO
Web 服务
SQL
MS Office: Excel, Access, Word

议题

- DB2 应用开发概述
- 服务器端开发
  - 存储过程
  - 用户定义的函数
  - 触发器
- 客户端开发
  - 
  - 嵌入式 **SQL**
  - 静态与动态 **SQL** 对比
  - **CLI/ODBC**
  - **JDBC / SQLJ / pureQuery**



嵌入式 ***SQL***应用是把 SQL 嵌入到C、C++或者COBOL等宿主语言的应用。嵌入式 SQL 应用可以包含静态或动态的SQL。

在本图中，C 语言程序hello.sqc包含了嵌入的 SQL。面向C语言的嵌入式 SQL API把 EXEC SQL（图*1.2*中突出显示的）用作定界符，以便让DB2中的 PRECOMPILER（预编译器）识别出待转换的SQL语句，而不是把它们当做实际的C代码。

你或许还注意到，在 hello.sqc 中，有些变量以冒号做前缀，例如 ***:dbname*** 、 ***:userID*** 和 ***:psw*** 。这些变量被称为主变量。主变量是在嵌入式 SQL 语句中应用的，来自宿主语言的变量。

执行**precompile**命令（也称为 **prep**命令）时带有 **bindfile** 选项将生成两个文件：一个是 hello.bnd 绑定文件，它仅含有SQL语句；另一个是hello.c文件，它仅含有C代码。绑定文件将使用 **bind** 命令进行编译，以获得一个 ***package***（程序包），它存储在数据库中。程序包中含有编译过的/可执行的 SQL 语句以及DB2在检索数据时将遵循的访问路径。要想执行 **bind** 命令，必须存在与数据库之间的连接。在该图的下面，hello.c文件像任何常规的C程序一样进行编译和连接。所产生的可执行文件hello.exe必须与存储在数据库中的程序包相匹配才能保证顺利执行。

议题

- DB2 应用开发概述
- 服务器端开发
  - 存储过程
  - 用户定义的函数
  - 触发器
- 客户端开发
  - 嵌入式 SQL
  - 静态与动态 **SQL** 对比
  - CLI/ODBC
  - JDBC / SQLJ / pureQuery

静态 SQL

SQL 语句在预编译时是完全已知的。

```
SELECT lastname, salary FROM employee
```

语句中所引用的列 (lastname, firstname) 和表 (employee) 的名称在预编译时是完全已知的。

主机变量值可以在运行时刻指定（但它们的数据类型仍然必预先编译）。

```
SELECT lastname, salary
  FROM employee
 WHERE firstname = :fname
```

你在运行你的应用之前对静态执行的 SQL 语句进行预编译、绑定及编译。

静态 SQL 最适用于那些统计值不发生太大变化的数据库。

静态 ***SQL*** 语句是在（预编译）之时SQL结构已经完全清楚的语句。

在这第二个例子中，一个主变量 ***:fname*** 被用作嵌入式 SQL语句的一部分。虽然主变量的值在运行之前是未知的，但其数据类型是可以从程序中知悉的，而且所有的其他对象（列名称、表名称）也都提前知道了。DB2软件通过对这些主变量的估计来提前计算存取方案；因此，这种情况仍然被视为静态SQL。

你在运行你的应用之前对静态执行的SQL语句进行预编译 (precompile)、绑定 (bind) 和编译 (compile)。静态SQL最适用于那些统计值不发生太大变化的数据库。

动态 SQL

这种 SQL 在运行时刻构建和执行。

```
SELECT ?, ? FROM ?
```

语句中所引用的列和表的名称在运行时刻之前是未知的。

存取方案在运行时确定。

正常情况下，静态 SQL 比动态 SQL 有更好的表现，因为存取方案是提前计算好的。

对于统计值频繁变化的表，动态 SQL 或许能够提供更准确的存取方案。

在使用动态 SQL 时，可以利用参数标记 (?) 来减少计算存取方案所需要的时间（参见下面的例子）

在本例中，语句所有用的列及表的名称在运行之前都是未知的。因此，存取方案只能在运行时间利用当时获得统计值进行计算。此类语句被视为动态 ***SQL***语句。

有些编程API，例如JDBC和ODBC，总是使用动态SQL，无论SQL语句中是否包含已知的对象。例如，语句**SELECT lastname, salary FROM employee** 中所有的列名称和表名称都是事先知道的，但在JDBC 或 ODBC中，你无法预编译这样的语句。这些语句的所有存取方案都是在运行时间计算。

动态 SQL

示例：

情形 **1:**

```
EXECUTE IMMEDIATELY SELECT name from EMP where dept = 1
EXECUTE IMMEDIATELY SELECT name from EMP where dept = 2
```

情形 **2:**

```
strcpy(hVStmtDyn, "SELECT name FROM emp WHERE dept = ?");
PREPARE StmtDyn FROM :hVStmtDyn;
EXECUTE StmtDyn USING 1;
EXECUTE StmtDyn USING 2;
```

在情形 1 中，每个语句被视为不同的 SQL，因此 DB2 将会为每个语句计算访问路径。

在情形 2，只有一个 SQL 语句：

```
"SELECT name FROM emp WHERE dept = ?"
```

因此，存取方案只需被计算一次，并被缓存在内存中。

一般而言，有两个语句可以用来把SQL语句处理为动态的：

PREPARE: 此语句准备或编译SQL语句，以便计算用以检索数据的存取方案

EXECUTE: 此语句执行 SQL

另一种方式，你也可以在单一语句中执行 **PREPARE** 和 **EXECUTE**： **EXECUTE IMMEDIATELY**

静态与动态 SQL 对比

嵌入式 SQL 应用支持静态的和动态的 SQL

静态 SQL 嵌入 SQL C 程序的示例

```
EXEC SQL SELECT name, dept
           INTO :name, :dept
           FROM staff WHERE id = 310;
printf( …)
```

动态 SQL 嵌入 SQL C 程序的示例

```
...
strcpy(hostVarStmtDyn,
  "UPDATE staff SET salary = salary + 1000 WHERE dept = ?");
EXEC SQL PREPARE StmtDyn FROM :hostVarStmtDyn;
EXEC SQL EXECUTE StmtDyn USING :dept;
```


就性能而言，静态SQL一般要好于动态SQL，因为静态SQL中的存取方案是在预编译时确定的，而不是在运行时间确定的。但是，如果运行环境中有大量的INSERT和DELETE活动，在预编译时计算出来的统计值可能并不是最新的，因此，静态SQL的存取方案可能不是最优的。在这种情况下，动态SQL或许是个不错的选择，但需要频繁运行RUNSTATS命令来收集最新统计信息。

议题

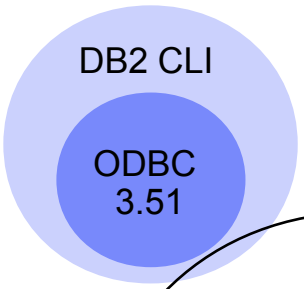
- DB2 应用开发概述
- 服务器端开发
  - 存储过程
  - 用户定义的函数
  - 触发器
- 客户端开发
  - 嵌入式 SQL
  - 静态与动态 SQL 对比
  - **CLI/ODBC**
  - JDBC / SQLJ / pureQuery

CLI / ODBC

CLI = 调用级接口 (Call Level Interface)

DB2 CLI 在被 ODBC Driver Manager 加载时可以用作 ODBC 驱动器

DB2 CLI 遵守 ODBC 3.51



DB2 CLI 符合 ODBC 3.51，在由 ODBC 驱动器管理器加载时可以用作 ODBC 驱动器。该图能够帮助你了解 DB2 CLI 对 ODBC 的支持。

CLI / ODBC

为了运行 CLI/ODBC 应用，你所需要的所有东西就是 DB2 CLI 驱动器。该驱动器可以从以下任一方式安装：

- IBM 数据服务器客户端
- IBM 数据服务器运行时刻客户端
- 面向 ODBC 和 CLI 的 IBM 数据服务器驱动器

为了开发 CLI/ODBC 应用，你需要 DB2 CLI 驱动器以及适当的库。它们只能在以下地点找到：

- IBM 数据服务器客户端

CLI / ODBC

CLI/ODBC 的特性 :

- 代码易于在多种 RDBMS 供应商的产品之间移植
- 与嵌入式 SQL 不同的是，无需预编译器或主变量 (host variable)
- 它运行动态 SQL
- 它非常流行

议题

- DB2 应用开发概述
- 服务器端开发
  - 存储过程
  - 用户定义的函数
  - 触发器
- 客户端开发
  - 嵌入式 SQL
  - 静态与动态 SQL 对比
  - CLI/ODBC
  - **JDBC / SQLJ / pureQuery**

JDBC / SQL / pureQuery

JDBC 的特性：

与在 ODBC 中一样，代码易于在多种 RDBMS 供应商的产品之间移植

动态 SQL

非常流行

SQLJ

以 Java 语言编写的嵌入式 SQL

静态 SQL

不是十分流行

pureQuery

基于 Eclipse 的插件，用于按照对象来管理关系数据

IBM 开发 Java 数据库应用的范式

2007 年中期新推出，随同 Optim Development Studio 一起提供

Java Database Connectivity (JDBC) 是一套 Java 编程 API，它对使用及访问数据库的方法进行了标准化。在JDBC中，代码代码易于在多种 RDBMS 供应商的产品之间移植。唯一需要对代码进行改变的地方一般是加载哪个JDBC驱动器以及使用哪个连接串。JDBC 仅使用动态 SQL，它非常流行。

SQLJ 是在Java程序中嵌入SQL的标准。它主要用于静态SQL，虽然它可以与JDBC进行互操作，如图所示。虽然它一般情况下比JDBC程序更紧凑，而且能提供更好的性能，但它并不普及。SQLJ 程序在被编译之前必须首先通过一个预处理器（SQLJ 转换器）来运行。

在图中（见下一张幻灯片），可能需要 DB2 客户端，也可能不需要，这取决于所采用的 JDBC 驱动器的类型。

pureQuery 是一套IBM基于 Eclipse的插件，用于按照对象来管理关系数据。自2007推出以来，pureQuery 可以自动生成代码在你的面向对象的代码与关系数据库对象之间建立对象-关系映射 (ORM)。你首先利用Optim™ Development Studio (ODS) 创建一个Java工程，然后接入一个DB2数据库，再让 ODS 发现所有的数据库对象。通过 ODS GUI，你可以选择一张表，然后生成 pureQuery 代码，它会把任何基础性的关系表实体转变为Java对象。生成代码以创建相关的 SQL 语句以及包含这些语句的父Java对象。所生成的Java对象以及所包含的SQL语句都可以进一步优化。利用，pureQuery，你可以决定在运行SQL时是采用静态模式，还是采用动态模式。pureQuery 支持Java 和 .NET。

JDBC / SQLJ – 支持的驱动器

驱动器类型	驱动器名称	被打包为	支持	需要最低层次的面向 **Java** 的 **SDK**
类型 2	DB2 JDBC Type 2 Driver for Linux, UNIX and Windows（已淘汰）	db2java.zip	JDBC 1.2 和 JDBC 2.0	1.4.2
类型 2 和类型 4	IBM Data Server Driver for JDBC and SQLJ	db2jcc.jar and sqlj.zip	兼容 JDBC 3.0	1.4.2
		db2jcc4.jar and sqlj4.zip	JDBC 4.0 及更早版本	6

类型 **2** 驱动器需要安装 **DB2** 客户端

“ 已淘汰**”**意味着它仍然得到支持，但不再增强了

请注意，相同的文件（例如 **db2jcc.jar** ）支持类型 **2** 和类型 **4**

虽然有多种JDBC驱动器，例如类型1、2、3 和 4；但类型1和3通常并不实用，而且 DB2已经放弃了对这些类型的支持。对于类型2，有两种驱动器，我们后面将简单介绍，但其中一个也被淘汰了。

如幻灯片所示，类型2和类型4得到 DB2 软件的支持。类型2驱动器需要安装DB2客户端，因为该驱动器需要使用它来建立与数据库之间的通信。类型4是纯粹的Java客户端，因此不需要DB2客户端，但此驱动器必须安装在JDBC应用运行的机器上。

如前所述，而且本幻灯片也显示，类型2有两类不同的驱动器；但是， DB2 JDBC Type 2 Driver for Linux, UNIX and Windows，即文件名为db2java.zip的驱动器，已经被淘汰了。

JDBC / SQLJ – 支持的驱动器

db2java.zip, db2jcc.jar, sqlj.zip, db2jcc4.jar 和 **sqlj4.zip** 包含在下述产品中：

- 面向 Linux、 UNIX 和 Windows 服务器的 IBM DB2
- IBM 数据服务器客户端
- IBM 数据服务器运行时刻客户端
- 面向 JDBC 和 SQLJ 的 IBM 数据服务器驱动器

当安装 DB2 服务器、DB2客户端或者面向JDBC和SQLJ的 IBM 数据服务器驱动器时，符合JDBC 3.0的db2jcc.jar 文件和 sqlj.zip 文件将会自动添加到你的classpath中。

如果你对本课程所讨论的内容有什么技术方面的问题的话，可以利用 db2university.com 课程 AA001EN 中的论坛。 同学、教员以及 IBM 人员可以帮助你！

在本演示中，我们将向你展示如何解决在使用 DB2 过程中可能遇到的故障。

议题

概述

帮助 (?) 命令

DB2 信息中心

管理通知日志

db2diag.log

DB2 Express-C 论坛

回顾 OS 命令

搜索 APAR 和技术注解 (technote)

支持性阅读材料和视频

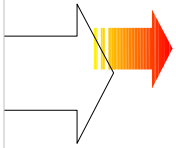
阅读材料

- DB2 Express-C 入门电子书
  - 附录 A: 故障解决

视频

- db2university.com 课程 AA001EN
  - 第 13 课 : 故障解决

议题



概述

帮助 (?) 命令

DB2 信息中心

管理通知日志

db2diag.log

DB2 Express-C 论坛

回顾 OS 命令

搜索 APAR 和技术注解 (technote)

本图简单介绍了发生问题时可以采取的行动。未必一定要遵循这个顺序。

当你遇到问题时，如果有SQLCODE，你首先可以试试利用帮助命令（“?”）获得有关此代码的详细信息。下面几张幻灯片我们将详细讨论这个命令。

你能够得到最多答案的一个不错的信息源是 DB2 手册（也称为“信息中心”）。

也会得到一些诊断日志，例如管理日志，而且db2diag.log也能够就问题所在提供有价值的线索。

此时你可能希望搜索DB2 Express-C 论坛寻找别人过去遇到的问题的答案，或者向社区寻求帮助。如果你运气不佳，你还可以使用操作系统命令进行搜索。例如，或许是磁盘空间不够了。

然后，你可以在产品中搜索APAR（缺陷报告）。

当你在诊断过程中遇到更多消息时，一定要Google它们！

议题

概述

帮助 (?) 命令

DB2 信息中心

管理通知日志

db2diag.log

DB2 Express-C 论坛

回顾 OS 命令

搜索 APAR 和技术注解 (technote)

帮助 (?) 命令

```
db2 ? SQL0104N
db2 ? SQL104N
db2 ? SQL-0104
db2 ? SQL-104
db2 ? SQL-104N
```


为了获得有关所收到的错误代码的更多信息，可以输入问号，然后输入该代码。问号 (?) 调用DB2的帮助命令。这里是几个调用该命令寻求帮助的例子。例如，如果你收到SQL错误代码 “-104”。所示出的各行是等价的。

SQLCODE是在每个SQL语句执行完毕之后收到的代码。其值的含义如下：

SQLCODE = 0; 命令成功执行

SQLCODE > 0; 命令成功执行，但返回了警告信息

SQLCODE < 0; 命令执行失败，返回错误消息

SQLSTATE是一个含五个字符的串，它符合ISO/ANSI SQL92标准。前两个字符被称为SQLSTATE类代码：

类代码 00 意味着命令被成功执行。

类代码 01 意味着警告消息。

类代码 02 意味着未找到条件。

所有其他类代码均被视为错误。

议题

概述

帮助 (?) 命令

DB2 信息中心

管理通知日志

db2diag.log

DB2 Express-C 论坛

回顾 OS 命令

搜索 APAR 和技术注解 (technote)

DB2 信息中心

含有DB2在线手册。

有一个搜索域

可以本地安装，也可以通过Internet访问

Internet版本是最新的：

http://publib.boulder.ibm.com/infocenter/db2luw/v9r7/index.jsp

你问题的多数答案都可以在DB2 信息中心找到！

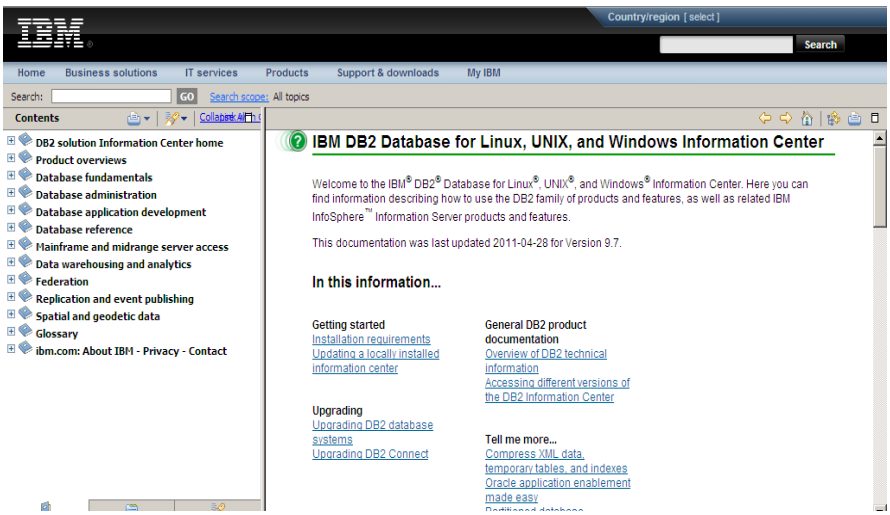
DB2 信息中心含有 DB2 在线手册。你可以使用搜索域来加快搜索，但要注意所搜索到的内容的上下文，因为信息中心可能同时提供相关产品的信息，而不仅仅限于 DB2 。

请注意，对其他版本的 DB2 信息中心有单独的链接。例如，对于 DB2 9.5 ，该链接是：

http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp

一定要查找你所使用的 DB2 版本的信息中心。

DB2 信息中心



这是 DB2 信息中心的一个屏幕截图。如前所述，使用搜索域可以快速找到你所寻找的东西。

议题

概述

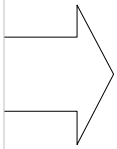
帮助 (?) 命令

DB2 信息中心

管理通知日志

db2diag.log

DB2 Express-C 论坛



回顾 OS 命令

搜索 APAR 和技术注解 (technote)

管理通知日志

日志以及在故障点的诊断信息

在Linux/UNIX平台上， 管理通知日志是一份名为 instance.nfy的文本文件

在windows中，所有的管理通知消息都写入事件日志中。可以使用事件查看器 (Event Viewer) 来查阅。

DBM 配置参数*notifylevel* 规定了待记录的信息的级别：

0 -- 不采集管理通知消息（不推荐）

1 -- 致命的或者不可恢复的错误

2 -- 需要立即采取的行动

3 -- 重要的信息，不需要立即行动（默认）

4 -- 信息消息

DB2 管理通知日志提供了在故障发生时有关错误的诊断信息。在Linux和UNIX平台上，管理通知日志是一份文本文件，名为 <instance name>.nfy（例如 “db2inst1.nfy”）。在 Windows 中，所有的管理通知消息都被写入Windows事件日志。

DBM配置参数**notifylevel** 能够让管理员指定待记录的信息的级别：

0 -- 不采集管理通知消息（不推荐）

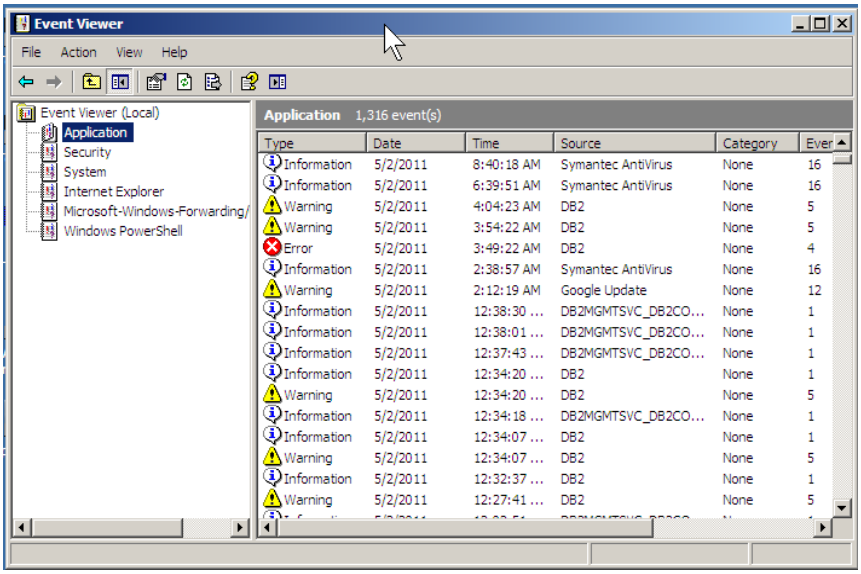
1 -- 致命的或者不可恢复的错误

2 -- 需要立即采取的行动

3 -- 重要的信息，不需要立即行动（默认）

4 -- 信息消息

管理通知日志



Windows 事件查看器以及关于 **DB2** 管理通知日志的项目

本屏幕截图示出了事件查看器（ Event Viewer ），其中列出了来自 DB2 （ DB2 管理通知日志）的多种错误 / 警告。

议题

- 概述
- 帮助 (?) 命令
- DB2 信息中心
- 管理通知日志
- **db2diag.log**
- DB2 Express-C 论坛
- 回顾 OS 命令
- 搜索 APAR 和技术注解 (technote)

db2diag.log

包含在db2diag.log中的信息

- 发生错误时的时间戳。利用它可以很快找到错误
- 问题所涉及的应用名称和DB2函数。
- 解释错误原因的诊断消息。
- 任何可用的的支持数据，例如SQLCA数据结构以及指向任何额外的转储或陷阱文件位置的指针。

如果你能够重建故障：

- 删除/重命名 db2diag.log
- 重建故障。将生成新的db2diag.log文件

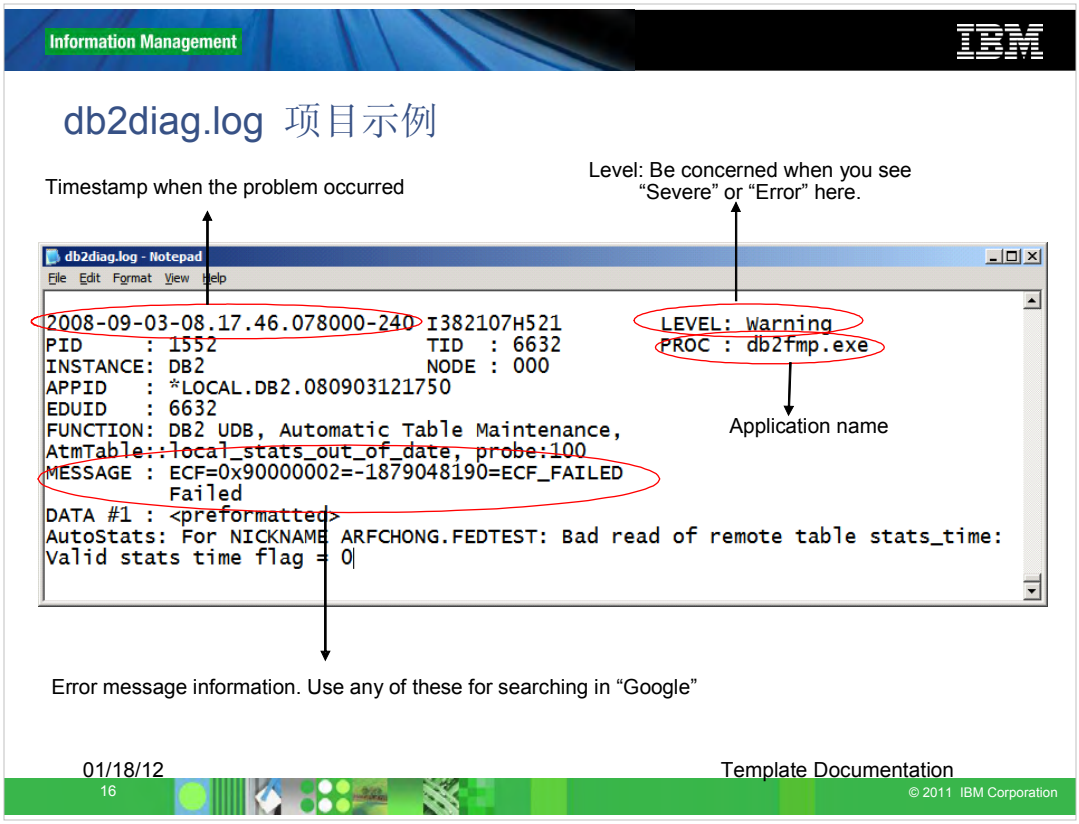
db2diag.log 提供了比 DB2 管理通知日志更详细的信息。它通常仅由IBM DB2技术支持人员或者有经验的DBA使用。db2diag.log中的信息包括：

• 报告错误的DB2代码位置。

• 应用标示符，它能够让你比对服务器的和客户端的db2diag.log文件中与某个应用相关的项目。

• 用于解释错误原因的诊断消息（起始于 "DIA"）。

• 任何可用的支持数据，例如 SQLCA 数据结构以及指向任何额外的转储或陷阱文件位置的指针。



本张幻灯片示出了在查看 db2diag.log 项目时应当注意的一些方面。首先，你应当根据时间戳过滤这些项目，假设你知道问题发生时间的话。然后丢弃那些不属于错误的项目。有时，项目可能会告诉你哪个可执行命令报告了错误。

这里是一个工具，名为“ db2diag”，它可以根据你所规定的标准过滤 db2diag.log 。

一旦你把范围缩小到报告了问题的项目上，你就可以查找手册或者 Internet 来获得更多线索。

定位 db2diag.log

Windows Vista 及以后版本

ProgramData\IBM\DB2\

Windows XP/2003（默认）

C:\Documents and Settings\All Users\Application Data\IBM\DB2\DB2COPY1\<instance name>

Linux/UNIX（默认）

INSTHOME/sqllib/db2dump (INSTHOME is the home directory of the instance owner)

你可以通过 **dbm cfg** 中的 ***diagpath*** 来改变 **db2diag.log**的位置

例如：db2 update dbm cfg using diagpath <path>

诊断文本的详细程度是由**dbm cfg**中的 ***diaglevel***来决定的

范围介于 0 至 4（默认为 3）

最详细的程度是4

db2diag.log 默认位于这里列出的目录中，你可以修改 dbm cfg 参数 diagpath 来提供你希望的存储路径。

diaglevel dbm cfg 参数用来规定你打算收集多少信息。

议题

概述

帮助 (?) 命令

DB2 信息中心

管理通知日志

db2diag.log

DB2 Express-C 论坛

回顾 OS 命令

搜索 APAR 和技术注解 (technote)

DB2 Express-C 论坛

免费的社区援助

www.ibm.com/developerworks/forums/dw_forum.jsp?forum=805&cat=19

主要采用英语

IBM DB2 Express-C团队监控该论坛，虽然主要是由社区提供帮助

你也可以通过DB2 Express-C网站访问它：

www.ibm.com/db2/express

（点击按钮进入论坛）

DB2 Express-C 论坛是一个免费的社区论坛，在这里， DB2 社区可以帮助你解决 DB2 的问题。虽然这是一个社区论坛，但 IBM 人员监控该论坛，并尽量帮助“遇到麻烦”的成员。

你无需注册就可以查阅论坛中的项目。但跟帖需要注册，这是免费的。

DB2 Express-C 论坛可以通过 DB2 Express-C 主页进入，地址为 ibm.com/db2/express

议题

概述

帮助 (?) 命令

DB2 信息中心

管理通知日志

db2diag.log

DB2 Express-C 论坛

回顾 OS 命令

搜索 APAR 和技术注解 (technote)

这里提供了一些操作系统命令，它们或许有助于确定问题所在。
Ipconfig 和 ping 是在发生连接问题时经常使用的命令。

议题

- 概述
- 帮助 (?) 命令
- DB2 信息中心
- 管理通知日志
- db2diag.log
- DB2 Express-C 论坛
- 回顾 OS 命令
- **搜索 APAR 和技术注解 (technote)**

搜索 APAR 和技术注解

APAR是 IBM对产品中缺陷的承认。

APAR通过修补包(fixpack)进行修补，虽然APAR描述可能会提供某种规避方法

如果你在使用DB2 Express-C（无保证的免费许可证），你无法应用修补包。你只能等待更新DB2 Express-C影像，其中含有相关的修补

如果你知道APAR号码，可以访问 www.ibm.com 并把该号码输入搜索域中，以阅读详细信息。

Google也会提供帮助！

有时候你遇到的问题可能是由 DB2 中的缺陷引起的。 IBM 经常发布修补包，其中包含对缺陷的修补代码（也称为 APAR ）。修补包文档中含有修补包所包含的修补程序列表。

我们总是建议，当你开发新应用时，一定要使用最新的修补包，以便能够享受到最新的修补程序带来的好处。

要查看你当前的版本以及修补包级别，可以打开 DB2 命令窗口或 Linux 外壳然后键入 db2level 。

请注意，修补包以及正式的 IBM DB2 技术支持并不向 DB2 Express-C 提供。对于 DB2 Express-C ，修补程序将整合到影像本身中，而不是采用修补包的方式。

如果你知道 APAR 号码，可以进入 www.ibm.com ，把此号码输入搜索域中，就可以阅读详细信息了。

Google 也会为你提供帮助！

如果你对本课程所讨论的内容有什么技术方面的问题的话，可以利用 db2university.com 课程 AA001EN 中的论坛。 同学、教员以及 IBM 人员可以帮助你！

Information Management

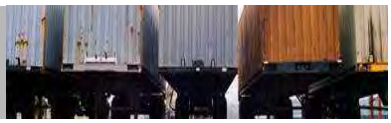
IBM

下一个前沿：大数据

IBM 信息管理云计算能力中心
IBM （加拿大）研究院

世界在变化，越来越 ...

物联化



互联化

智能化

由此导致的信息爆炸使得我们需要一种新型智能

... 来协助建立更加智慧的地球

我们的世界越来越物联化、互联化，并由于这两个特点而更加智能化。我们获得关于事物更多的信息。你现在可以利用 32 位及 64 位微处理器技术来做许多的事情，并可以采集到有关物理基础架构（带有传感器）、业务流程、人类交互等的更多信息。你现在可以完成仅在 10 年前还不可能完成的事情。支持我们所有人的互连基础架构在可用带宽方面是非凡的。在物联及连接两方面，通过挖掘这一新的信息宝藏，我们能够使事情大幅度智能化。这就是智慧地球这一思想所要实现的愿景。因此，我们来看看在当前的信息管理能力背景下，我们如何让一切更加智慧。

这种爆炸式增长也反映在巨大的非结构化信息数量上，这代表了不同的挑战。我们都理解组织良好的结构化数据。我们与它打交道已经有好几十年了。它是信息技术的核心，过去 60 年为我们带来了可编程计算的进步。我们只是刚刚进入非结构化领域，我们需要了解其有多大潜力，我们如何获得此类信息，以及我们如何处理它。想想社交网站 (Twitter, Facebook, LinkedIn，等等) 、博客、点击流、即时消息、电子邮件、电子传感器数据等正在产生的巨量信息。如果我们能够有效地过滤这些数据，把方方面面恰当地聚集到一起，把它们与现有的运营数据结合起来，并对它们进行有效的分析，我们岂不是可以更好地利用这些信息？

IDC 参考：

http://idcdocserv.com/925

http://www.computer.org/portal/web/news/home/-/blogs/2613266;jsessionid=abbfded1402383e107abfa2641d6

当今的组织机构仅利用了它们所拥有的数据的一小部分

想想如何分析所有的数据，并在这些新的、非传统的数据类型中寻找深刻观点，这是多么大的挑战

设想你能够分析推特每天产生的 12B TB 的数据，以发现人们对你的产品的评价，以及谁在你的目标人群中具有关键影响力。你能设想挖掘此类数据来寻找新的市场机会吗？

如果医院能够利用每时每刻从 ICU 病房中患者身上收集的成千上万传感器读数，从而比传统技术提前几天发现患者病情可能正在恶化的细微指征，那该多好。

设想一家绿色能源公司能够利用巨量的天气数据以及海量的运营数据来优化资产位置和利用率，从而使它们的环境友好能源比传统能源更具有成本竞争力。

设想你能够在几分钟之内做出风险决策，例如某人是否有资格获得一笔抵押贷款，这是通过分析包括实时交易数据在内的多种来源的数据完成的，而此时该客户还仍然在电话的那头或者还在办公室中，这是何等高的效率。

设想执法机构能够实时分析音频及视频馈入，无需手工干预就能够发现可以活动，那该多好。

随着新的数据源在数量、种类及速度方面不断扩张，这些数据的潜力也不断上升，从而实现各个行业决策流程的革命化。

需要提及的综合性大数据战略的一个关键部分：有效的大数据平台和方法需要与你的 IT 基础架构的其他部分相整合。

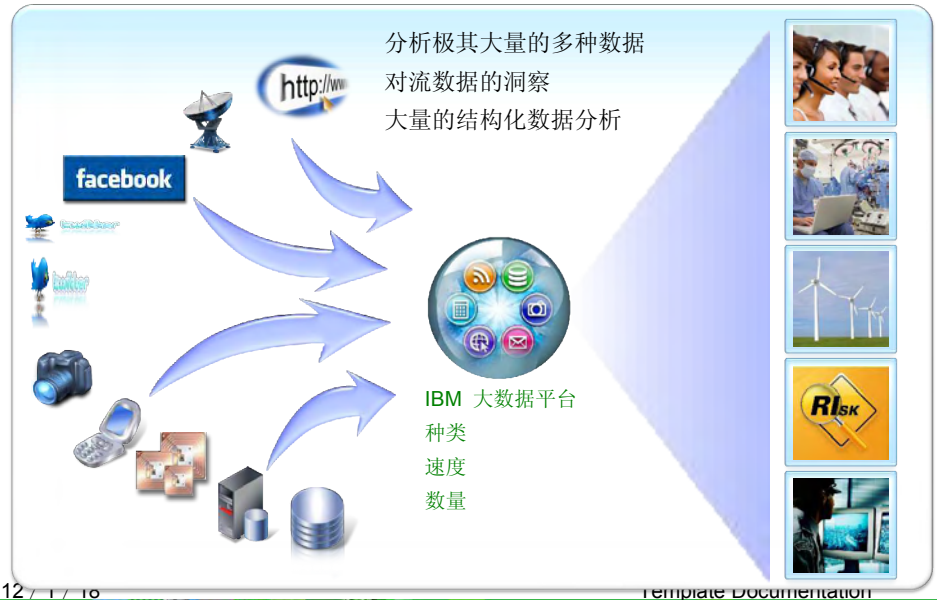
在体系结构方面你最不需要的东西就是另一套技术或者数据孤岛。

IBM 的大数据技术应当与你现有的数据仓库和分析技术通力合作并扩展它们的价值。

传统的和非传统的数据源的价值，以及传统的和非传统的技术的价值，都会随着它们集成的一起而提高。

解决方案 – IBM 的大数据平台

以任何速度把任何数据来源集中到一起，以获得洞察力



IBM 的大数据平台是所有大数据问题的答案。它能够帮助你把任何数量、任何类型的数据源以任何速度整合到一起，从而回答过去望尘莫及的业务问题。

关键点

大数据平台构建于开源基础之上。 IBM 一直在支持开源运动，因为 IBM 相信， Hadoop 技术是解决 Internet 规模分析的正确选择。但 IBM 的方法更加成熟，它建立在企业级平台技术基础之上。

开源 – 构建于 Hadoop （映射简化 , HDFS ）、 HBase （ Hadoop 数据库）、 Pig （对大数据集的分析，数据分析程序的高级语言）、 Lucene （全文本检索）、 Jawl （面向 Javascript Object Notation 的查询语言）之上。

IBM 通过两种企业引擎使这些开源工具更加成熟，可以让它们处理大量的数据并分析多种类型的数据。流分析用来管理流流动并根据流数据应用各种分析工具 – 挖掘、数学、视频，等等。 Internet 规模的分析工具用于依原样静态（ at rest ）存储数据，并应用分析能力，例如根据数据集进行文本分析。

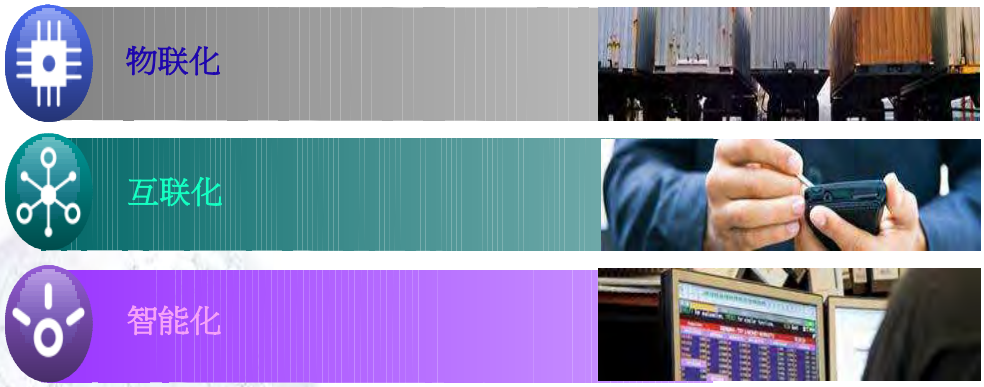
用户环境：这是让此平台更加成熟并向现有人员（而不仅仅是能够编写映射简化程序的专业编程人员）展示大数据威力的重要方面。此开发环境旨在为开发和测试大数据分析工具及应用提供成熟的环境。还有最终用户可视化能力用来探索及分析数据。

集成 – 这是大数据平台的一个重要方面 – 它必须集成，以便“把大数据带给企业” – 洞察必须整合到数据仓库、数据库、应用等之中。实现这一点的一种关键机制是信息集成 – 这包括治理这些数据。

证据点和统计

Tera Echos 公司 –‘我们的开发人员交付应用的速度提高了 45% ，这全依赖于流处理语言的敏捷性’ – 表明了一套成熟的开发环境和语言如何加快 BD 应用的开发速度。

你如何了解包括 Hadoop 在内的大数据的更多信息？



了解 IBM 大数据平台及技术的详细信息：
http://www-01.ibm.com/software/data/bigdata/

Hadoop: 大数据背后的技术部分：
http://www-01.ibm.com/software/ebusiness/jstart/hadoop/

在这些链接中可以找到关于大数据的更多内容。未来在大数据上。欢迎了解更多内容，以便准备好使用大数据来解决现实的编程问题，并创建更加智慧的地球。

如果你对本课程所讨论的内容有什么技术方面的问题的话，可以利用 db2university.com 课程 AA001EN 中的论坛。 同学、教员以及 IBM 人员可以帮助你！