第1章 概述

教学目标

教学过程

教学目标

- 理解使用数据库的原因
- 掌握数据库的基本概念和特点
- 理解**Microsoft SQL Server**简史
- 理解**Microsoft SQL Server**系统的体系结构
- 理解数据库和数据库对象的特点
- 理解管理工具的特点
- 理解数据库管理员的任务

教学过程

1.1 为什么使用数据库？

1.2 什么是数据库？

1.3 Microsoft SQL Server简史

1.4 Microsoft SQL Server系统的体系结构

1.5 数据库和数据库对象的特点

1.6 管理工具

1.7 数据库管理员

1.1 为什么使用数据库？

	A	B	C	D
1	员工姓名	员工E-mail	主管姓名	部门主管E-mail
2	赵　松	zhao@abcom.com.cn	周语	fengty@abcom.com.cn
3	钱大河	qiandh@abcom.com.cn	沈　洋	shenyang@abcom.com.cn
4	孙　策	sunce@abcom.com.cn	冯同	fengty@abcom.com.cn
5	李江息	leejx@abcom.com.cn	张子美	zhangzm@abcom.com.cn
6	周小娟	zhouxj@abcom.com	林　霖	linlin@abcom.com.cn
7	吴词人	wuperson@abcom.com.cn	徐　曼	xuman@abcom.com.cn
8	郑依复	zhengyf@abcom.com.cn	沈　洋	shenTomson@abcom.com.cn
9	王洛旺	wanglwang@abcom.com.cn	张子美	zhangzm@abcom.cn
10	姜　泰	jiangtai@abcom.com.cn	???	???

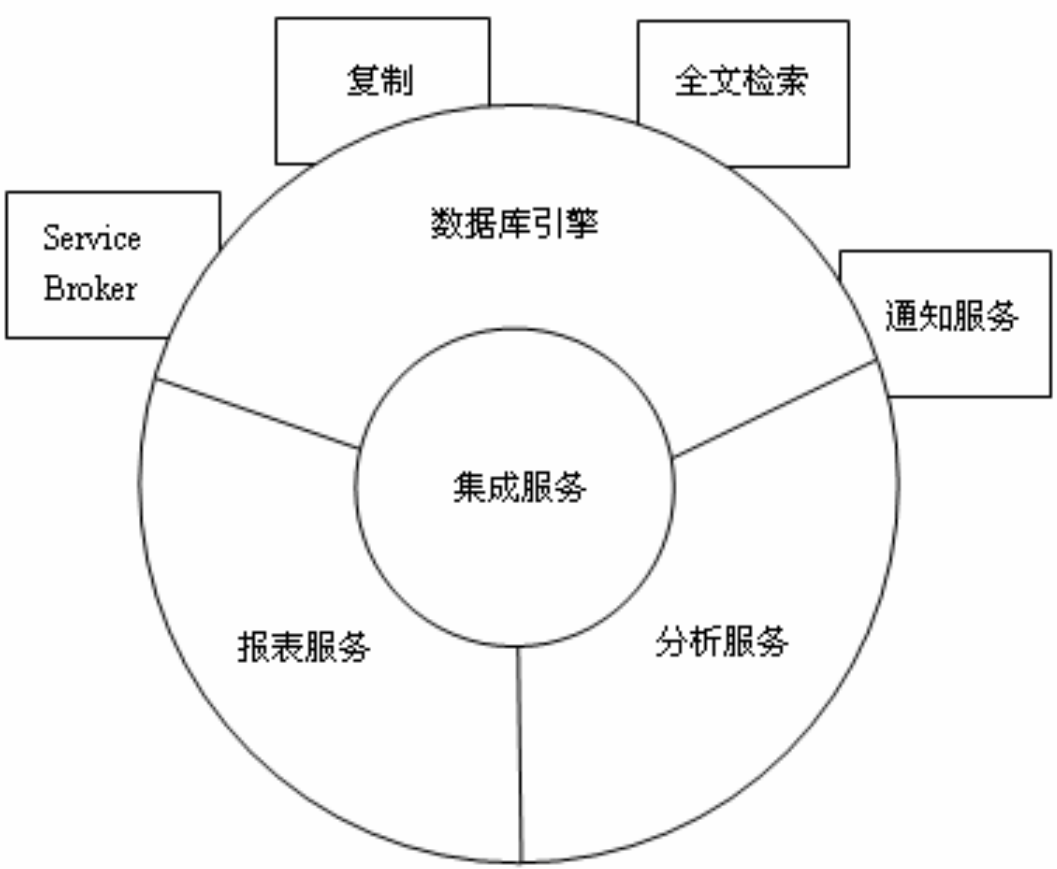
1.2 什么是数据库？

- 数据库是采用计算机技术统一管理的相关数据的集合，数据库能为各种用户共享，具有最小冗余度、数据之间联系密切、有较高数据独立性等特点。
- 数据库管理系统(database management system，DBMS)是位于用户与操作系统之间的一层数据管理软件，它为用户或应用程序提供访问数据库的方法，包括数据库的建立、查询、更新以及各种数据库控制等。
- 数据库系统是实现有组织地、动态地存储大量关联数据、方便多用户访问的计算机软件、硬件和数据资源组成的系统，是采用数据库技术的计算机系统。
- 数据库产品是由专门开发DBMS的厂商提供的。

1.3 Microsoft SQL Server简史

- 1987年，赛贝斯公司发布了Sybase SQL Server系统
- 1988年，微软公司参加赛贝斯SQL Server系统开发
- 1992年，联合开发Windows NT环境的SQL Server系统
- 1993年，微软与赛贝斯公司联合开发正式结束
- 1995年，微软发布Microsoft SQL Server 6.0系统
- 1998年，微软推出Microsoft SQL Server 7.0系统。
- 2000年，微软发布Microsoft SQL Server 2000系统。
- 2005年，微软发布Microsoft SQL Server 2005系统。

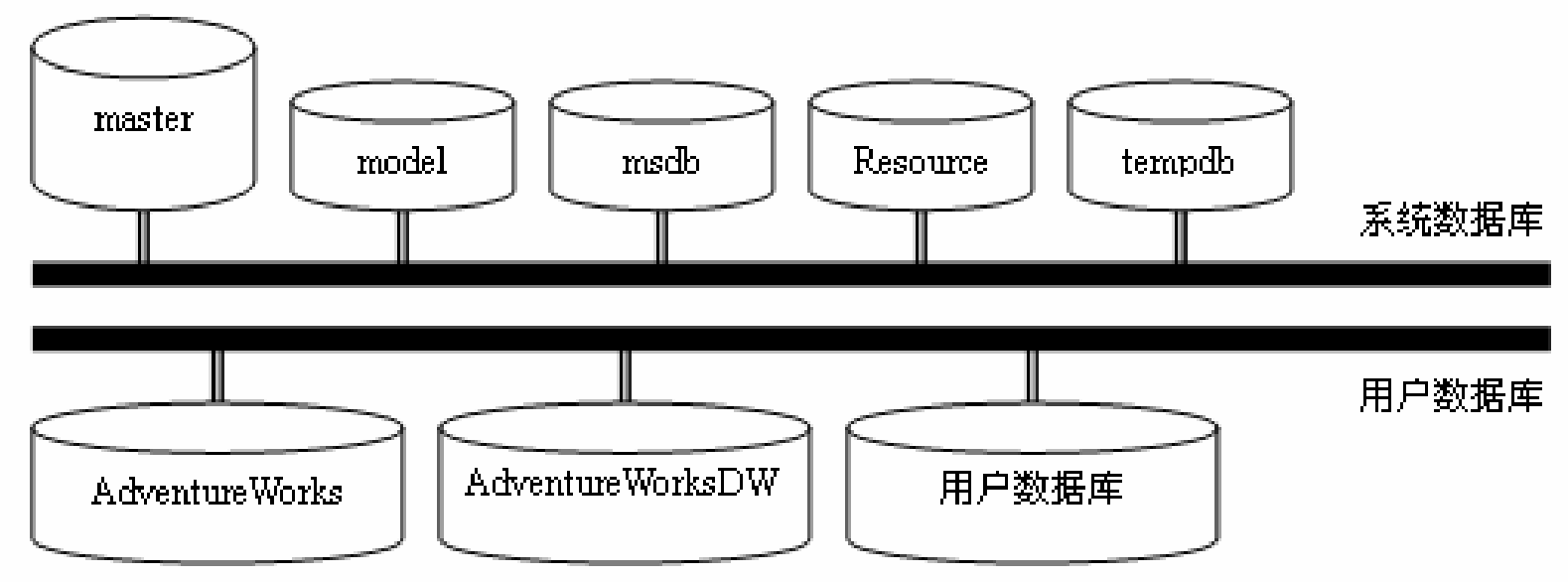
1.4 Microsoft SQL Server系统的体系结构



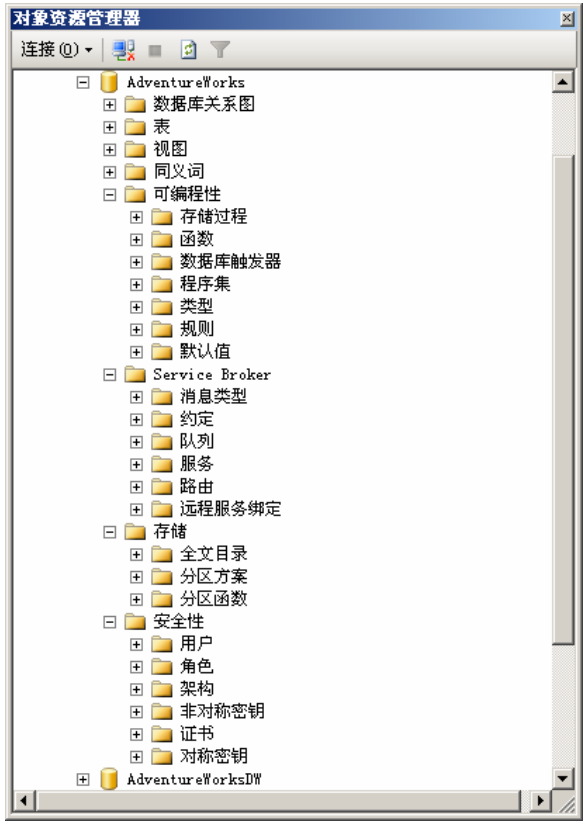
1.5 数据库和数据库对象的特点

- 数据库的类型和特点
- 数据库对象的类型和特点

数据库的类型和特点


master
model
msdb
Resource
tempdb
系统数据库
用户数据库
AdventureWorks
AdventureWorksDW
用户数据库

数据库对象的类型和特点



1.6 管理工具

- SQL Server配置管理器
- Microsoft SQL Server Management Studio
- SQL Server Profiler
- 数据库引擎优化顾问
- 大量的命令行实用工具

1.7 数据库管理员

- 任务一：安装和配置。
- 任务二：容量规划。
- 任务三：应用架构设计。
- 任务四：管理数据库对象
- 任务五：存储空间管理。
- 任务七：安全管理。
- 任务六：备份和恢复。
- 任务八：性能监视和调优。
- 任务九：调度作业。
- 任务十：网络管理。
- 任务十一：高可用性和高可伸缩性管理
- 任务十二：故障解决。

第2章 安装规划和配置

教学目标

教学过程

教学目标

- 理解为什么要进行安装规划
- 掌握如何进行安装规划
- 理解系统的版本特点
- 理解和掌握安装过程中的关键步骤
- 理解和掌握为什么要进行升级规划
- 掌握如何注册服务器
- 掌握服务器选项的类型和设置方式

教学过程

- 2.1 安装规划
- 2.2 安装过程
- 2.3 验证安装结果
- 2.4 升级规划
- 2.5 注册服务器
- 2.6 配置服务器选项

2.1 安装规划

- 安装规划是指在安装Microsoft SQL Server系统之前对系统的安装目的、环境需求、并发用户、安装版本、服务器位置、安装过程中特殊要求等内容进行统筹安排。

安装目的

- 安装目的是指安装**Microsoft SQL Server 2005**系统支持业务工作的顺利、高效的、安全的进行。
- **Microsoft SQL Server**系统是一个可以在多种行业领域中管理业务数据的大型数据库管理系统。
- 如果用户的环境是一个经常有数百个用户并发访问的生产环境
- 用在业务操作环境中的系统与用在分析环境中的系统是不同的

系统版本

- Microsoft SQL Server 2005系统提供了6个不同的版本，即
  - Express版
  - 工作组版
  - 标准版
  - 企业版
  - 开发人员版
  - 企业评估版。
- 经常使用的是前面4个版本。

环境需求

- 环境需求是指系统安装时对硬件、操作系统、网络等环境的要求，
- 这些要求也是**Microsoft SQL Server**系统运行所必须的条件。

安装位置和安全模式

- 在实际安装前，还应该考虑这两个问题：确定安装文件的根目录和确定选用的系统安全模式。这两个问题都与今后的使用息息相关。
- 安装文件的根目录是**Microsoft SQL Server**系统存储程序文件的位置
- **Microsoft SQL Server**系统有两种安全模式，即**Windows**认证模式和混合模式。

2.2 安装过程

- 虽然说Microsoft SQL Server 2005系统具有很好的易用性，安装时可以按照安装向导的逐步提示执行安装操作，但是用户应该对安装过程中的选项有深刻理解，只有这样才能完全按照自己的要求顺利完成安装操作。
- 下面针对安装过程中涉及的实例名、服务帐户、身份验证模式、排序规则设置等关键内容进行分析。

2.3 验证安装结果

- 安装结束之后，怎样才能知道系统安装成功呢？为了确保安装是正确的，我们也可以采用一些验证方法。
- 常用的验证方法包括：
  - 检查Microsoft SQL Server系统的服务和工具是否存在
  - 应该自动生成的系统数据库和样本数据库是否存在
  - 相关系统目录和文件是否正确等。

2.4 升级规划

- 分析和评估升级需求
- 确定升级内容和选择升级路线
- 模拟升级过程
- 制定详细升级计划和灾难恢复计划
- 执行升级操作
- 测试升级结果
- 完成升级总结报告

2.5 注册服务器

- 注册服务器就是为**Microsoft SQL Server**客户机/服务器系统确定一个数据库所在的机器，该机器作为服务器可以为客户端的各种请求提供服务。
- 服务器组是服务器的逻辑集合，可以利用**Microsoft SQL Server Management Studio**工具把许多相关的服务器集中在一个服务器组中，方便对多服务器环境的管理操作。

2.6 配置服务器选项

- 服务器选项用于确定**Microsoft SQL Server 2005**系统运行行为、资源利用状况。
- 既可以使用**sp_configure**系统存储过程配置服务器选项，也可以使用**SQL Server Management Studio**工具设置。

服务器选项

- 与以前版本相比，**Microsoft SQL Server 2005**系统的服务器选项有了比较大的变化，新增了许多选项，有些选项被废弃了。
- **Microsoft SQL Server 2005**系统提供的**60**多个服务器选项的名称和对应的取值范围如表**2-1**所示。

使用sp_configure系统存储过程配置选项

- sp_configure系统存储过程可以用来显示和配置服务器的各种选项。
- sp_configure的基本语法形式如下：
  - sp_configure 'option_name', 'value'

使用SQL Server Management Studio配置选项

- 在SQL Server Management Studio工具的“对象资源管理器”中右击将要设置的服务器名称，从弹出的快捷菜单中选中“属性”选项，则出现如图2-18所示的“服务器属性 - ABCSERVER”对话框。
- 可以在该对话框中完成大多数选项的配置

第3章 管理安全性

教学目标

教学过程

教学目标

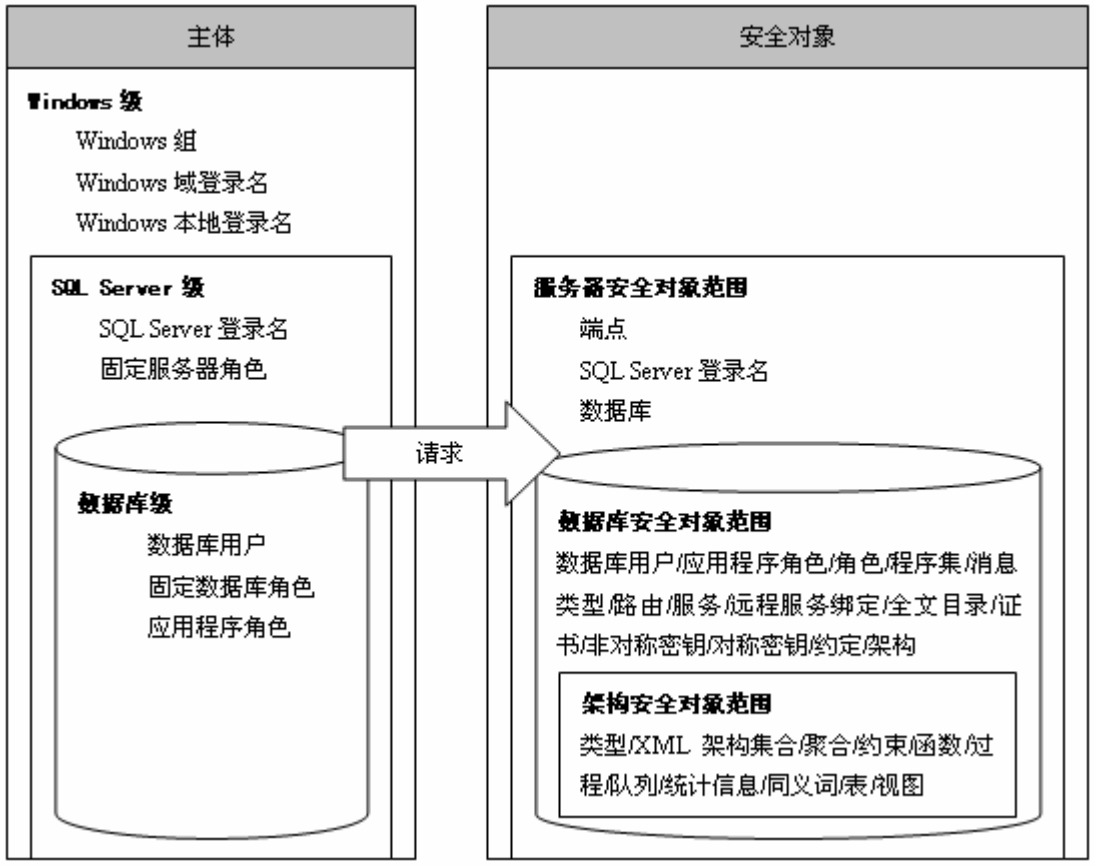
- 理解数据库安全性问题和安全性机制之间的关系
- 掌握管理和维护登录名
- 理解SQL Server系统的密码策略
- 掌握固定服务器角色的特点和管理
- 掌握管理和维护数据库用户
- 掌握管理和维护架构
- 掌握权限类型和权限管理
- 理解系统内置的加密机制

教学过程

3.1 概述

- 第一个安全性问题：当用户登录数据库系统时，如何确保只有合法的用户才能登录到系统中呢？这是一个最基本的安全性问题，也是数据库管理系统提供的基本功能。在**Microsoft SQL Server 2005**系统中，这个问题是通过身份验证模式和主体解决的。
- 第二个安全性问题：当用户登录到系统中，他可以执行哪些操作、使用哪些对象和资源呢？这也是一个非常基本的安全问题，在**Microsoft SQL Server 2005**系统中，这个问题是通过安全对象和权限设置来实现的。
- 第三个安全性问题：数据库中的对象由谁所有？

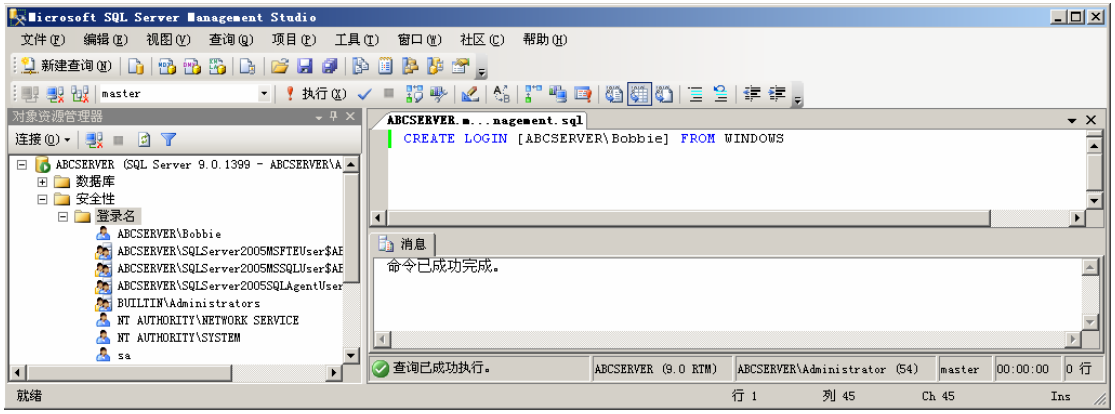
服务器安全对象范围

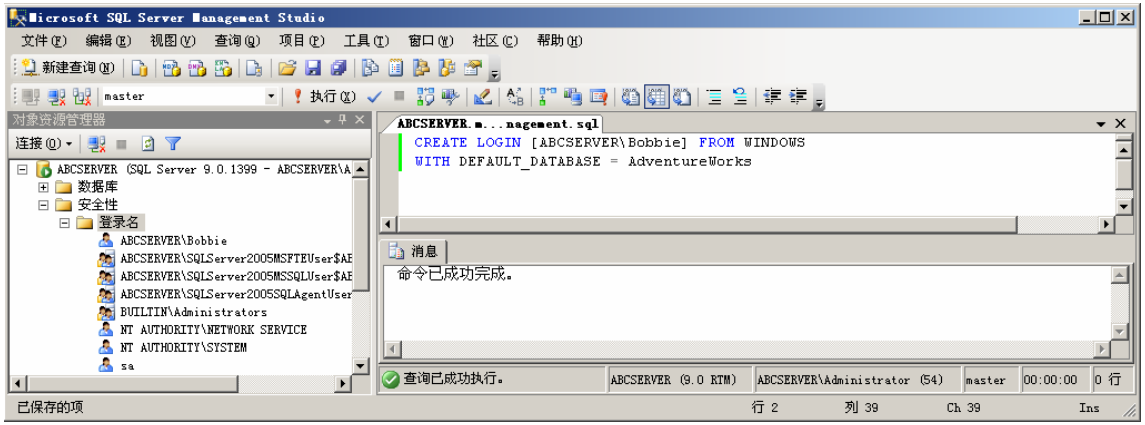

主体
安全对象
Windows 级
Windows 组
Windows 域登录名
Windows 本地登录名
SQL Server 级
SQL Server 登录名
固定服务器角色
数据库级
数据库用户
固定数据库角色
应用程序角色
请求
服务器安全对象范围
端点
SQL Server 登录名
数据库
数据库安全对象范围
数据库用户/应用程序角色/角色/程序集/消息类型/路由/服务/远程服务绑定/全文目录/证书/非对称密钥/对称密钥/约定/架构
架构安全对象范围
类型/XML 架构集合/聚合/约束/函数/过程/队列/统计信息/同义词/表/视图

3.2 管理登录名

- 管理登录名包括创建登录名、设置密码策略、查看登录名信息、修改和删除登录名。下面讲述登录名管理的内容。
- 注意，**sa**是一个默认的**SQL Server**登录名，拥有操作**SQL Server**系统的所有权限。该登录名不能被删除。
- 当采用混合模式安装**Microsoft SQL Server**系统之后，应该为**sa**指定一个密码。

创建登录名


Microsoft SQL Server Management Studio
文件(F) 编辑(E) 视图(V) 查询(Q) 项目(P) 工具(T) 窗口(W) 社区(C) 帮助(H)
新建查询(N)
master
执行(X)
对象资源管理器
连接(O)
ABCSERVER (SQL Server 9.0.1399 - ABCSERVER\A
数据库
安全性
登录名
ABCSERVER\Bobbie
ABCSERVER\SQLServer2005MSFTEUser$AE
ABCSERVER\SQLServer2005MSSQLUser$AE
ABCSERVER\SQLServer2005SQLAgentUser
BUILTIN\Administrators
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
sa
ABCSERVER.m...nagement.sql
CREATE LOGIN [ABCSERVER\Bobbie] FROM WINDOWS
消息
命令已成功完成。
查询已成功执行。
ABCSERVER (9.0 RTM)
ABCSERVER\Administrator (54)
master
00:00:00
0 行
就绪
行 1
列 45
Ch 45
Ins


Microsoft SQL Server Management Studio
文件(F) 编辑(E) 视图(V) 查询(Q) 项目(P) 工具(T) 窗口(W) 社区(C) 帮助(H)
新建查询(N)
master
执行(X)
对象资源管理器
连接(O)
ABCSERVER (SQL Server 9.0.1399 - ABCSERVER\A
数据库
安全性
登录名
ABCSERVER\Bobbie
ABCSERVER\SQLServer2005MSFTEUser$AE
ABCSERVER\SQLServer2005MSSQLUser$AE
ABCSERVER\SQLServer2005SQLAgentUser
BUILTIN\Administrators
NT AUTHORITY\NETWORK SERVICE
NT AUTHORITY\SYSTEM
sa
ABCSERVER.m...nagement.sql
CREATE LOGIN [ABCSERVER\Bobbie] FROM WINDOWS
WITH DEFAULT_DATABASE = AdventureWorks
消息
命令已成功完成。
查询已成功执行。
ABCSERVER (9.0 RTM)
ABCSERVER\Administrator (54)
master
00:00:00
0 行
已保存的项
行 2
列 39
Ch 39
Ins

密码策略

- **Windows**的密码策略包括了密码复杂性和密码过期两大特征。
- 密码的复杂性是指通过增加更多可能的密码数量来阻止黑客的攻击。
- 密码过期策略是指如何管理密码的使用期限。

维护登录名

- 登录名创建之后，可以根据需要修改登录名的名称、密码、密码策略、默认的数据库等信息，可以禁用或启用该登录名，甚至可以删除不需要的登录名。
- **ALTER LOGIN**语句用来修改登录名的属性信息。
- 修改登录名的名称与删除、重建该登录名是不同的。

3.3 固定服务器角色

- 固定服务器角色也是服务器级别的主体，他们的作用范围是整个服务器。
- 固定服务器角色已经具备了执行指定操作的权限，可以把其他登录名作为成员添加到固定服务器角色中，这样该登录名可以继承固定服务器角色的权限。
- 下面，首先讲述**Microsoft SQL Server 2005**系统提供的固定服务器角色的特点，然后分析如何处理登录名与固定服务器角色之间的关系。

固定服务器角色的特点

- **Microsoft SQL Server 2005**系统提供了**8**个固定服务器角色，这些固定服务器角色的清单和功能描述如表**3-1**所示。

固定服务器角色和登录名

- 在**Microsoft SQL Server**系统中，可以把登录名添加到固定服务器角色中，使得登录名作为固定服务器角色的成员继承固定服务器角色的权限。对于登录名来说，可以判断其是否某个固定服务器角色的成员。用户可以使用**sp_addsrvrolemember**、**sp_helpsrvrolememeber**、**sp_dropsrvrolemember**等存储过程和**IS_SRVROLEMEMBER**函数来执行有关固定服务器角色和登录名之间关系的操作。

3.4 管理数据库用户

- 数据库用户是数据库级的主体，是登录名在数据库中的映射，是在数据库中执行操作和活动行动者。
- 在**Microsoft SQL Server 2005**系统中，数据库用户不能直接拥有表、视图等数据库对象，而是通过架构拥有这些对象。
- 数据库用户管理包括创建用户、查看用户信息、修改用户、删除用户等操作。

创建用户

- 可以使用**CREATE USER**语句在指定的数据库中创建用户。由于用户是登录名在数据库中的映射，因此在创建用户时需要指定登录名。
- 例如，可以使用如图**3-13**所示的命令在**AdventureWorks**数据库中创建对应于**Peter**登录名的用户。

维护用户

- 可以使用**ALTER USER**语句修改用户。修改用户包括两个方面，第一，可以修改用户名；第二可以须改用户的默认架构。
- 如果用户不再需要了，可以使用**DROP USER**语句删除数据库中的用户

3.5 管理架构

- 架构是形成单个命名空间的数据库实体的集合。
- 架构是数据库级的安全对象，也是**Microsoft SQL Server 2005**系统强调的新特点，是数据库对象的容器。
- 管理架构包括创建架构、查看架构的信息、修改架构及删除架构等。

创建架构

- 使用**CREATE SCHEMA**语句不仅可以创建架构，而且在创建架构的同时还可以创建该架构所拥有的表、视图并且可以对这些对象设置权限。下面讲述如何创建架构。
- 图**3-17**是一个创建架构的最简单的示例。在这个示例中，仅仅指定**companyGManager**作为架构的名称，没有明确指定该架构的所有者。这时，该架构的所有者为当前执行该项操作的用户。

修改和删除架构

- 修改架构是指将特定架构中的对象转移到其他架构中。可以使用**ALTER SCHEMA**语句完成对架构的修改。需要注意的是，如果要更改对象本身的结构，那么应该使用针对该对象的**ALTER**语句。
- 如果架构已经没有存在的必要了，可以使用**DROP SCHEMA**语句删除架构。删除架构是需要注意，如果架构中包含有任意的对象，那么删除操作失败。只有当架构中不再包含有对象时，才可以被删除。

3.6 数据库角色

- 数据库角色是数据库级别的主体，也是数据库用户的集合。数据库用户可以作为数据库角色的成员，继承数据库角色的权限。
- 数据库管理人员可以通过管理角色的权限来管理数据库用户的权限。
- **Microsoft SQL Server 2005**系统提供了一些固定数据库角色和**public**特殊角色。
- 下面详细描述数据库角色的特点和管理方式。

管理数据库角色

- 管理数据库角色包括创建数据库角色、添加和删除数据库角色成员、查看数据库角色信息、修改和删除角色等。
- 可以使用**CREATE ROLE**语句创建角色。
- 如果希望为角色添加成员，那么可以使用**sp_addrolemember**存储过程。
- 如果希望修改数据库角色的名称，那么可以使用**ALTER ROLE**语句。
- 如果某个角色确实不再需要了，那么可以使用**DROP ROLE**语句删除指定的角色。

固定数据库角色

- 就像固定服务器角色一样，固定数据库角色也具有了预先定义好的权限。使用固定数据库角色可以大大简化数据库角色权限管理工作。
- **Microsoft SQL Server 2005**系统提供了**9**个固定数据库角色，这些固定数据库角色清单和权限描述如表**3-2**所示。

public角色

- 除了前面介绍的固定数据库角色之外，Microsoft SQL Server系统成功安装之后，还有一个特殊的角色，这就是public角色。
- public角色有两大特点，第一，初始状态时没有权限；第二，所有的数据库用户都是他的成员。

3.7 管理应用程序角色

- 应用程序角色是一个数据库主体，它可以使应用程序能够用其自身的、类似用户的权限来运行。在使用应用程序时，可以仅仅允许那些经过特定应用程序连接的用户来访问数据库中的特定数据，如果不通过这些特定的应用程序连接，那么无法访问这些数据。这是使用应用程序角色实现安全管理的目的。
- 与数据库角色相比来说，应用程序角色有三个特点：
  - 第一，在默认情况下该角色不包含任何成员；
  - 第二，在默认情况下该角色是非活动的，必须激活之后才能发挥作用；
  - 第三，该角色有密码，只有拥有应用程序角色正确密码的用户才可以激活该角色。当激活某个应用程序角色之后，用户会失去自己原有的权限，转而拥有应用程序角色的权限。

CREATE APPLICATION ROLE语句

- 在Microsoft SQL Server 2005系统中，可以使用CREATE APPLICATION ROLE语句创建应用程序角色。该语句的语法形式如下所示：
  - CREATE APPLICATION ROLE application_role_name
  - WITH PASSWORD = 'password',
  - DEFAULT_SCHEMA = schema_name

3.8 管理权限

- 权限是执行操作、访问数据的通行证。只有拥有了针对某种安全对象的指定权限，才能对该对象执行相应的操作。
- 在**Microsoft SQL Server 2005**系统中，不同的对象有不同的权限。
- 为了更好地理解权限管理的内容，下面从权限的类型、常用对象的权限、隐含的权限、授予权限、收回权限、否认权限等几个方面讲述。

权限的类型

- 在Microsoft SQL Server 2005系统中，不同的分类方式可以把权限分成不同的类型。
- 如果依据权限是否预先定义，可以把权限分为预先定义的权限和预先未定义的权限。
- 如果按照权限是否与特定的对象有关，可以把权限分为针对所有对象的权限和针对特殊对象的权限。

常用对象的权限

- 上一节从权限的角度来看待对象，本节从对象的角度来看待权限。在使用**GRANT**语句、**REVOKE**语句、**DENY**语句执行权限管理操作时，经常使用**ALL**关键字表示指定安全对象的常用权限。
- 不同的安全对象往往具有不同的权限。安全对象的常用权限如表**3-3**所示。

授予权限

- 在Microsoft SQL Server 2005系统中，可以使用GRANT语句将安全对象的权限授予指定的安全主体。
- 在执行GRANT语句时，授权者必须具有带GRANT OPTION的相同权限，或具有隐含所授予权限的最高权限。

收回权限

- 如果希望从某个安全主体处收回权限，可以使用**REVOKE**语句。
- **REVOKE**语句是与**GRANT**语句相对应的，可以把通过**GRANT**语句授予给安全主体的权限收回。
- 也就是说，使用**REVOKE**语句可以删除通过**GRANT**语句授予给安全主体的权限。

否认权限

- 安全主体可以通过两种方式获得权限，第一种方式是直接使用**GRANT**语句为其授予权限，第二种方式是通过作为角色成员继承角色的权限。
- 使用**REVOKE**语句只能删除安全主体通过第一种方式得到的权限，要想彻底删除安全主体的特定权限必须使用**DENY**语句。
- **DENY**语句的语法形式与**REVOKE**语句非常类似。

3.9 SQL Server 2005内置的加密机制

- Microsoft SQL Server 2005系统不是简单的提供一些加密函数，而是把成熟的数据安全技术引进到数据库中，形成了一个清晰的内置加密层次结构。
- 在加密技术领域，根据加密密钥和解密密钥是否相同，可以把加密方式分为对称加密机制和非对称加密机制，其数据传输示意图如图3-33所示。

3.10 使用Microsoft SQL Server Management Studio工具

- 除了可以使用**Transact-SQL**语句执行有关安全的操作之外，使用**SQL Server Management Studio**图形工具也可以完成许多有关安全管理的操作。
- 需要注意的是，考虑到性能和安全等原因，大多数情况下建议使用**Transact-SQL**语句执行相关的操作。

第4章 管理数据库

教学目标

教学过程

教学目标

- 理解数据库的管理问题
- 理解数据库文件和文件组的基本特征
- 掌握数据库的物理存储方式和大小估算方法
- 使用**CREATE DATABASE**语句定义数据库
- 理解数据库选项的作用和设置方式
- 理解扩大数据库的原因和方法
- 理解收缩数据库的原因和方法
- 掌握文件组的管理方式
- 理解数据库快照的作用和特点
- 掌握优化数据库设计的方法

教学过程

4.1 概述
4.2 数据库文件和文件组的基本特征
4.3 定义数据库
4.4 修改数据库
4.5 管理数据库快照
4.6 其他相关操作
4.7 优化数据库

4.1 概述

- 为了有效地实现数据库的管理工作，我们必须至少解决8个方面的问题，这些问题包括
  - 数据库文件的存储问题
  - 数据库的大小问题
  - 确定数据库运行时的行为特征
  - 数据库的更改问题
  - 数据库的扩大问题
  - 数据库的收缩问题
  - 如何兼顾数据库的事务处理效率和决策支持效率问题
  - 数据库的性能优化问题等。

4.2 数据库文件和文件组的基本特征

- 本节主要讲述数据库文件的类型、事务的概念、文件组的作用、估算数据库文件大小的方法等内容。
- 一个数据库至少有一个数据文件和一个事务日志文件
- 数据文件又可以分成主数据文件和次数据文件两种形式
- 事务就是一个单元的工作，该单元的工作要么全部完成，要么全部不完成。
- 文件组就是文件的逻辑集合。为了方便数据的管理和分配，文件组可以把一些指定的文件组合在一起。
- 在**Microsoft SQL Server**系统中，可管理的最小物理空间是以页为单位的，每一个页的大小是**8KB**

4.3 定义数据库

- 定义数据库就是创建数据库和设置数据库选项。
- 本节从三个方面讲述定义数据库：创建数据库、设置数据库选项和查看数据库信息。

创建数据库

- 创建数据库就是确定数据库名称、文件名称、数据文件大小、数据库的字符集、是否自动增长以及如何自动增长等信息的过程。在一个**Microsoft SQL Server**实例中，最多可以创建**32767**个数据库。数据库的名称必须满足系统的标识符规则。在命名数据库时，一定要使数据库名称简短和有一定的含义。
- 具有**CREATE DATABASE**、**CREATE ANY DATABASE**或**ALTER ANY DATABASE**权限的用户才可以执行创建数据库的操作。
- 在**Microsoft SQL Server**系统中，既可以使用**CREATE DATABASE**语句创建数据库，也可以使用**SQL Server Management Studio**工具创建数据库。下面，主要介绍如何使用**CREATE DATABASE**语句创建数据库。

数据的状态和选项

- 为了理解数据库的运行特征，需要了解数据库的状态和选项。下面分别介绍数据库的状态和选项。
- 数据库总是存在某个特定的状态中，例如，**ONLINE**状态表示数据库处于正常的在线状态，可以对数据库执行正常的操作。数据库的状态清单和特征描述如表**4-1**所示。
- 设置数据库选项是定义数据库状态或特征的方式。在**Microsoft SQL Server 2005**系统中，共有大约**40**个数据库选项，这些选项可以分为**13**个类型。数据库选项清单和功能描述如表**4-2**所示。

查看数据库信息

- 在Microsoft SQL Server 2005系统中，可以使用一些目录视图、函数、存储过程查看有关数据库的基本信息。
- sys.databases数据库和文件目录视图可以查看有关数据库的基本信息，sys.database_files可以查看有关数据库文件的信息，sys.filegroups可以查看有关数据库文件组的信息，sys.master_files可以查看数据库文件的基本信息和状态信息。
- DATABASEPROPERTYEX函数可以查看指定数据库的指定选项的信息，一次只能返回一个选项的设置。

4.4 修改数据库

- 数据库创建之后，根据需要，可以使用**ALTER DATABASE**语句对数据库进行修改。
- 除了前面讲过的设置数据库选项之外，修改操作还包括更改数据库名称、扩大数据库、收缩数据库、修改数据库文件、管理数据库文件组、修改字符排列规则等。
- 下面详细讨论这些内容。

更改数据库名称

- 数据库创建之后，一般情况下不要更改数据库的名称，因为许多应用程序都可能使用了该数据库的名称。数据库名称更改之后，需要修改相应的应用程序。但是，如果确实需要更改数据库名称，也可以使用**ALTER DATABASE**语句做到。
- 使用**ALTER DATABASE**语句更改数据库名称的语法形式如下所示：
  - **ALTER DATABASE database_name MODIFY NAME = new_database_name**

扩大数据库

- 在Microsoft SQL Server系统中，如果数据库的数据量不断膨胀，可以根据需要扩大数据库的尺寸。
- 有三种扩大数据库的方式。
  - 第一种方式是设置数据库为自动增长方式，可以在创建数据库时设置。
  - 第二种方式是直接修改数据库的数据文件或日志文件的大小
  - 第三种方式是在数据库中增加新的次要数据文件或日志文件。

收缩数据库

- 如果数据库的设计尺寸过大了，或者删除了数据库中的大量数据，这时数据库会白白耗费大量的磁盘资源。根据用户的实际需要，可以收缩数据库的大小。
- 在**Microsoft SQL Server**系统中，有三种收缩数据库的方式。
  - 第一种方式是设置数据库为自动收缩，这可以通过设置**AUTO_SHRINK**数据库选项实现。
  - 第二种方式是收缩整个数据库的大小，这可以通过使用**DBCC SHRINKDATABASE**命令完成。
  - 第三种方式是收缩指定的数据文件，这可以使用**DBCC SHRIKNFILE**命令实现。除了这些命令方式之外，也可以使用**SQL Server Management Studio**工具来收缩数据库。

修改数据库文件

- 用户可以根据需要使用**ALTER DATABASE**语句修改数据库中指定的文件。这些修改操作包括增加数据文件、在指定的文件组中增加指定文件、增加日志文件、删除指定的文件、修改指定的文件等。增加数据文件、修改指定的文件等操作已经讲过了，下面通过一些示例讲述有关数据库文件的其他操作。

管理文件组

- 文件组是数据库数据文件的逻辑组合，它可以对数据文件进行管理和分配，以便提高数据库文件的并发使用效率。
- **Transact-SQL**语言没有提供独立的管理文件组的命令，只能通过**ALTER DATABASE**语句提供了管理文件组的命令。
- 这些管理文件组的命令包括新建文件组、设置默认的文件组、设置文件组的属性、修改文件组、删除文件组等。下面，详细研究管理文件组的操作。

4.5 管理数据库快照

- 数据库快照提供了源数据库在创建快照时刻的只读、静态视图。数据库快照可以有效地支持报表数据汇总、数据分析等只读操作。数据库快照也是**Microsoft SQL Server 2005**系统的一个显著特征。
- 如果源数据库中包含了未提交事务，那么这些事务不包含在数据库快照中。需要说明的是，数据库快照必须与源数据库在同一个服务器实例上。

4.6 其他相关操作

- 除了前面讲述的数据库操作之外，数据库管理操作还包括
  - 分离数据库
  - 附加数据库
  - 删除数据库
- 下面讨论这些操作。

4.7 优化数据库

- 在创建数据库时，有两个基本目标：提高数据库的性能和提高数据库的可靠性。提高数据库的性能就是提高操纵数据库的速度。
- 提高数据库的可靠性就是数据库中某个文件破坏之后，数据库依然可以正常使用的能力。
- 一般地，可以通过选择如何放置数据文件和日志文件、如何使用文件组、如何使用**RAID**等技术来优化数据库和数据库文件。

第5章 Transact-SQL语言

教学目标

教学过程

教学目标

- 理解**Transact-SQL**语言和**SQL**语言之间的关系
- 理解**Transact-SQL**语言的特点和执行方式
- 理解数据定义语言的类型和特点
- 理解数据操纵语言的类型和特点
- 理解数据控制语言的类型和特点
- 理解事务管理语言的类型和特点
- 理解附加语言元素的类型和特点

教学过程

- 5.1 概述
- 5.2 Transact-SQL语言的特点和执行方式
- 5.3 数据定义语言
- 5.4 数据操纵语言
- 5.5 数据控制语言
- 5.6 事务管理语言
- 5.7 附加的语言元素
- 5.8 数据类型
- 5.9 内置函数

5.1 概述

- **Transact-SQL**语言是微软公司在**Microsoft SQL Server**系统中使用的语言，是对**SQL**语言的一种扩展形式。
- **Transact-SQL**语言有**4**个特点：一是一体化的特点，集数据定义语言、数据操纵语言、数据控制语言、事务管理语言和附加语言元素为一体。二是有两种使用方式，即交互使用方式和嵌入到高级语言中的使用方式。三是非过程化语言，只需要提出“干什么”，不需要指出“如何干”，语句的操作过程由系统自动完成。四是，类似于人的思维习惯，容易理解和掌握。

5.2 Transact-SQL语言的特点和执行方式

- 在Microsoft SQL Server 2005系统中，根据Transact-SQL语言的功能特点，可以把Transact-SQL语言分为5种类型，即数据定义语言、数据操纵语言、数据控制语言、事务管理语言和附加的语言元素。
- 在Microsoft SQL Server 2005系统中，主要使用SQL Server Management Studio工具来执行Transact-SQL语言编写的查询语句。除此之外，还可以使用sqlcmd实用工具来执行Transact-SQL语句。

5.3 数据定义语言

- 数据定义语言用于创建数据库和数据库对象，为数据库操作提供对象。例如，数据库以及表、触发器、存储过程、视图、索引、函数、类型、用户等都是数据库中的对象，都需要通过定义才能使用。在**DDL**中，主要的**Transact-SQL**语句包括**CREATE**语句、**ALTER**语句、**DROP**语句。

5.4 数据操纵语言

- 数据操纵语言主要是用于操纵表、视图中数据的语句。当我们创建表对象之后，初始状态时该表是空的，没有任何数据。如何向表中添加数据呢？这时需要使用**INSERT**语句。如何检索表中数据呢？可以使用**SELECT**语句。如果表中数据不正确的，那么可以使用**UPDATE**语句进行更新。当然，也可以使用**DELETE**语句删除表中的数据。实际上，**DML**语言正是包括了**INSERT**、**SELECT**、**UPDATE**、**DELETE**等语句。

5.5 数据控制语言

- 数据控制语言(DCL)主要用来执行有关安全管理的操作，该语言主要包括GRANT语句、REVOKE语句和DENY语句。GRANT语句可以将指定的安全对象的权限授予相应的主体，REVOKE语句则删除授予的权限，DENY语句拒绝授予主体权限，并且防止主体通过组或角色成员继承权限。

5.6 事务管理语言

- 在Microsoft SQL Server系统中，可以使用BEGIN TRANSACTION、COMMIT TRANSACTION、ROLLBACK TRANSACTION等事务管理语言(TML)语句来管理显式事务。
- 其中，BEGIN TRANSACTION语句用于明确地定义事务的开始，COMMIT TRANSACTION语句用于明确地提交完成的事务。
- 如果事务中出现了错误，那么可以使用ROLLBACK TRANSACTION语句明确地取消定义的事务。

5.7 附加的语言元素

- 除了前面介绍的语句之外，**Transact-SQL**语言还包括了附加的语言元素。这些附加的语言元素主要包括标识符、变量和常量、运算符、表达式、数据类型、函数、控制流语言、错误处理语言、注释等。
- 下面，详细研究这些内容。

标识符

- 在**Transact-SQL**语言中，数据库对象的名称就是其标识符。在**Microsoft SQL Server**系统中，所有的数据库对象都可以有标识符，例如服务器、数据库、表、视图、索引、触发器、约束等。大多数对象的标识符是必须的，例如，创建表时必须为表指定标识符。但是，也有一些对象的标识符是可选的，例如，创建约束时用户可以不提供标识符，其标识符由系统自动生成。
- 按照标识符的使用方式，可以把这些标识符分为常规标识符和分割标识符两种类型。

变量和常量

- 在Microsoft SQL Server 2005系统中，变量也被称为局部变量，是可以保存单个特定类型数据值的对象。
- 常量是表示特定数据值的符号，常量也被称为字面量。常量的格式取决于它所表示的值的数据类型。

运算符

- 运算符是一种符号，用来指定要在一个或多个表达式中执行的操作。
- 在**Microsoft SQL Server 2005**系统中，可以使用的运算符可以分为算术运算符、逻辑运算符、赋值运算符、字符串串联运算符、按位运算符、一元运算符、比较运算符等。

表达式

- 在**Transact-SQL**语言中，表达式是由标识符、变量、常量、标量函数、子查询、运算符等的组合。在**Microsoft SQL Server 2005**系统中，表达式可以在多个不同的位置使用，这些位置包括查询中检索数据的一部分、搜索数据的条件等。
- 表达式可以分为简单表达式和复杂表达式两种类型。

控制流语言

- 一般地，结构化程序设计语言的基本结构是顺序结构、条件分支结构和循环结构。顺序结构是一种自然结构，条件分支结构和循环结构都需要根据程序的执行状况对程序的执行顺序进行调整。
- 在**Transact-SQL**语言中，用于控制语句流的语言被称为控制流语言。**Microsoft SQL Server 2005**系统提供了**8**种控制流语句，这些语句如表**5-7**所示。

错误捕捉语言

- 为了增强程序的健壮性，必须对程序中可能出现的错误进行及时地处理。
- 在**Transact-SQL**语言中，可以使用两种方式处理发生的错误：使用**TRY…CATCH**构造和使用**@@ERROR**函数。

注释

- 所有的程序设计语言都有注释。注释是程序代码中不执行的文本字符串，用于对代码进行说明或暂时仅用正在进行诊断的部分语句。一般地，注释主要描述程序名称、作者名称、变量说明、代码更改日期、算法描述等。
- 在**Microsoft SQL Server**系统中，支持两种注释方式，即双连字符**(--)**注释方式和正斜杠星号字符对**(/*...*/)**注释方式。

5.8 数据类型

- 本节将从六个方面研究**Transact-SQL**语言的数据类型。首先，分析数据类型的概念、特点和主要类型。然后，讲述数字数据类型的主要内容和特点。之后，描述字符数据类型的使用方式。接下来，研究日期和时间数据类型的输入输出特点。接着，分析二进制数据类型的特点。最后，讲述其他数据类型的内容和特点。

数据类型的类型和特点

- 在Microsoft SQL Server 2005系统中，需要使用数据类型的对象包括：表中的列、视图中的列、定义的局部变量、存储过程中的参数、Transact-SQL函数、存储过程的返回值等。
- Microsoft SQL Server 2005系统提供了28种数据类型。这些数据类型可以分为数字数据类型、字符数据类型、日期和时间数据类型、二进制数据类型以及其他数据类型。

数字数据类型

- 使用数字数据的数据类型被称为数字数据类型。这些数据类型的数字可以参加各种数学运算。我们还可以为这些数据类型继续进行分类。
- 从这些数字是否有小数，可以把这些数据类型分为整数类型和小数类型。
- 从这些数字的精度和位数是否可以明确地确定，可以把这些数据类型分为精确数字类型和近似数字类型。
- 从是否可以表示金额，可以分为货币数字类型和非货币数字类型。

字符数据类型

- 字符数据类型用于存储固定长度或可变长度的字符数据。
- 在**Microsoft SQL Server 2005**系统中，提供了**CHAR**、**VARCHAR**、**TEXT**、**NCHAR**、**NVARCHAR**、**NTEXT**等**6**种数据类型。前**3**种数据类型是非**Unicode**字符数据，后**3**种是**Unicode**字符数据。

时间和日期数据类型

- 如果希望存储日期和时间数据，那么可以使用**DATETIME**或**SMALLDATETIME**数据类型。这两种数据类型的差别在于其表示的日期和时间范围不同、时间精确度也不同。
- **DATETIME**数据类型可以表示的范围是**1753**年**1**月**1**日至**9999**年**12**月**31**日，时间精确度是**3.33**毫秒。**SMALLDATETIME**数据类型可以表示的范围是**1900**年**1**月**1**日至**2079**年**12**月**31**日，时间精确度是**1**分钟。

二进制数据类型

- 二进制数据类型包括**BINARY**、**VARBINARY**、**IMAGE**等**3**种数据类型，可以用于存储二进制数据。
- 其中，**BINARY**可以用于存储固定长度的二进制数据，**VARBINARY**用于存储可变长度的二进制数据。
- 微软建议使用**VARBINARY(MAX)**代替**IMAGE**数据类型

其他数据类型

- 除了前面介绍的数据类型之外，**Microsoft SQL Server 2005**系统还提供了**CURSOR**、**SQL_VARIANT**、**TABLE**、**TIMESTAMP**、**UNIQUEIDENTIFIER**、**XML**等数据类型。使用这些数据类型可以完成特殊数据对象的定义、存储和使用。

5.9 内置函数

- 可以把Microsoft SQL Server 2005系统提供的内置函数分为13种类型，每一种类型的内置函数都可以完成某种类型的操作，
- 这些类型的函数名称和主要功能如表5-8所示。

第6章 表

教学目标

教学过程

教学目标

- 理解设计表时应该考虑的因素
- 理解表的基本特点和类型
- 掌握使用**CREATE TABLE**语句创建表
- 修改表的结构
- 理解标识符列的作用和特点
- 掌握已分区表的作用和管理方式

教学过程

6.1 设计表时应该考虑的因素

6.2 表的基本特点和类型

6.3 创建和修改表

6.4 已分区表

6.1 设计表时应该考虑的因素

- 因素一，考虑表将要存储哪些数据对象，绘制出ER图。
- 因素二，考虑表中将要包含的列，这些列的数据类型、精度等属性是什么？
- 因素三，考虑列的属性，例如哪些列允许空值，哪些列不允许空值？
- 因素四，考虑表是否使用主键，如果使用则在何处使用主键？
- 因素五，考虑是否使用约束、默认值、规则，以及在何处使用这些对象？
- 因素六，考虑是否使用外键，在何处使用外键？
- 因素七，考虑是否使用索引，在何处使用索引，使用什么样的索引？

6.2 表的基本特点和类型

- 本节讲述两方面的内容，首先分析和描述表的基本特点，然后讨论表的分类方式和表的类型。

表的基本特点

- 表是关系模型中表示实体的方式，是用来组织和存储数据、具有行列结构的数据库对象。
- 一般而言，表具有下列一些基本特点：代表实体、由行和列组成、行和列的顺序是不重要的等等。
- 下面，详细讲述这些特点。

表的类型

- 在Microsoft SQL Server 2005系统中，按照表的作用，可以把表分为4种类型，即
  - 普通表
  - 已分区表
  - 临时表
  - 系统表
- 每一种类型的表都有自己的作用和特点。

6.3 创建和修改表

- 本节主要围绕着创建和修改表展开讨论。
- 内容包括创建表、增加和删除列、修改列的属性、设置标识符列、查看表的信息、删除表等。

创建表

- 在Microsoft SQL Server 2005系统中，既可以使用CREATE TABLE语句创建表，也可以使用可视化的SQL Server Management Studio图形工具。下面主要研究如何使用CREATE TABLE语句创建表。
- CREATE TABLE语句一种经常使用的创建表的方法，也是一种最灵活、最强大的创建表的方式。

增加或删除列

- 表创建之后，用户可以根据需要使用**ALTER TABLE**语句修改表的结构。在表中增加新列、删除已有的列是常见的修改表结构的操作。
- 当用户向表中增加一个新列时，**Microsoft SQL Server**为表中该列在已有数据的每一行中的相应位置插入一个数据值。因此，当向表中增加一个新列时，最好为该新列定义一个默认约束，使该列有一个默认值。如果该新列没有默认约束，并且表中已经有了其他数据，那么必须指定该新列允许空值，否则，系统将产生一个错误信息。

更该列的数据类型

- 使用**ALTER TABLE**语句除了可以增加新列和删除列之外，还可以对列的属性进行更改。本节主要讲述如何更改列的数据类型。使用**ALTER TABLE**语句更该列的数据类型的基本语法形式如下所示：
- **ALTER TABLE table_name ALTER COLUMN column_name new_type_name**

创建和修改标识符列

- 标识符列表示惟一地标识表中的每一行数据的符号。
- 在**Microsoft SQL Server 2005**系统中，可以创建两种类型的标识符列，即**IDENTITY**列和**ROWGUIDCOL**列。
- 下面，详细研究这两种标识符列的创建和修改方式。

IDENTITY列

- 使用IDENTITY属性的列是IDENTITY列，每一个表中最多只能有一个IDENTITY列。定义IDENTITY属性时需要指定两个值：种子值和增量值。这样，表中第一行的IDENTITY列的值是种子值，其他行的IDENTITY列的值是在前一行的值的基础上增加一个增量值得到的。
- IDENTITY属性的语法形式如下所示：
  - IDENTITY (seed, increment)

ROWGUIDCOL列

- ROWGUIDCOL列是全局唯一标识符列。每一个表中最多可以创建一个ROWGUIDCOL列。从理论上来看，分布在Internet上的两个不同的计算机中的ROWGUIDCOL列的值出现相同的现象的概率是微乎其微的。在创建表时，可以使用UNIQUEIDENTIFIER数据类型定义ROWGUIDCOL列。

查看表的信息

- 表创建之后，可以使用许多函数、存储过程查看有关表的各种信息。
- **COLUMNPROPERTY**函数可以用于查看有关表中的列的信息，这些信息包括是否为空、是否计算得到的列、是否具有**IDENTITY**属性、是否**ROWGUIDCOL**列等。
- **sp_depends**存储过程可以用于查看指定表的依赖对象，这些依赖对象包括依赖于表的视图、存储过程等。
- 使用**sp_help**存储过程可以查看有关表结构的信息。

删除表

- 删除表就是将表中数据和表的结构从数据库中永久性地去除。表被删除之后，就不能再恢复该表的定义。删除表可以使用**DROP TABLE**语句来完成，该语句的语法形式如下：
  - **DROP TABLE table_name**

使用图形工具执行有关表的操作

- 在Microsoft SQL Server 2005系统中，可以使用可视化工具执行有关表的操作，这些操作包括创建表、修改表的结构、查看依赖关系、查看有关属性信息等。

6.4 已分区表

- 如果一个表中包含了大量的、以多种不同方式使用的数据，且一般地查询不能按照预期的成本完成，那么应该考虑使用已分区表。
- 已分区表是指按照数据水平方式分区，将数据分布于一个数据库的多个不同的文件组中。在对数据进行查询或更新时，这些已分区表将被视为独立的逻辑单元。
- 注意：只有**Microsoft SQL Server 2005**企业版支持已分区功能。

分区函数和分区方案

- 在对表进行分区之前，必须考虑如何创建分区函数和分区方案。
- 分区函数定义如何根据某些列中的值将表中的数据行映射到一组分区
- 分区方案则将分区函数指定的分区映射到文件组中。

第7章 操纵数据

教学目标

教学过程

教学目标

- 理解操纵数据需要解决的问题
- 掌握使用**INSERT**语句插入数据
- 掌握使用**UPDATE**语句更新数据
- 掌握使用**DELETE**语句删除数据
- 掌握使用**SELECT**语句检索数据
- 理解分组、子查询、连接、集合运算、**CTE**等检索特点
- 理解数据加密的方式和特点

教学过程

7.1 概述

- 表创建之后，表只是一个空表。
- 如何向表中添加数据呢？如果表中已有数据了，但是数据不合适或不正确，那么如何更新这些数据呢？如果表中的数据不再需要了，那么如何删除这些过时的数据呢？如何按照用户需要，将表中的数据检索出来呢？这些问题都是数据操纵问题。
- 用户可以使用**INSERT**、**UPDATE**、**DELETE**、**SELECT**等语句来解决这些数据操纵问题。

7.2 插入数据

- 表创建之后往往只是一个空表，因此向表中插入数据是在表结构创建之后，首先需要执行的操作。向表中插入数据，应该使用**INSERT**语句。该语句包括了两个子句，即**INSERT**子句和**VALUES**子句。**INSERT**子句指定要插入数据的表名或视图名称，它可以包含表或视图中列的列表。**VALUES**子句指定将要插入的数据。
- 一般地，使用**INSERT**语句一次只能插入一行数据。**INSERT**语句的基本语法形式如下所示：
  - **INSERT INTO table_or_view_name (column_list)**
  - **VALUES (expression)**

7.3 更新数据

- 可以使用**UPDATE**语句更新表中已经存在的数据。**UPDATE**语句既可以一次更新一行数据，也可以一次更新许多行，甚至可以一次更新表中的全部数据行。
- 在**UPDATE**语句中，使用**WHERE**子句指定要更新的数据行满足的基本条件，使用**SET**子句给出新的数据。新数据既可以是常量，也可以是指定的表达式。
- **UPDATE**语句的基本语法形式如下：
  - **UPDATE table_or_view_name**
  - **SET column_name = expression, …**
  - **WHERE search_condition**

7.4 删除数据

- 当表中的数据不再需要时，可以删除。一般情况下，使用**DELETE**语句删除数据。**DELETE**语句可以从一个表中删除一行或多行数据。
- 删除数据的**DELEE**语句的基本语法形式如下：
  - **DELETE**
  - **FROM table_or_name**
  - **WHERE search_condition**

7.5 检索操作概述

- 如果我们希望检索表中数据，可以使用SELECT语句。在SELECT语句中，有三个基本的组成部分：SELECT子句、FROM子句和WHERE子句。
- SELECT子句用于指定将要检索的列名称，FROM子句指定将要检索的对象，WHERE子句则用于指定数据应该满足的条件。

7.6 使用SELECT子句检索数据

- 在SELECT语句中，可以在SELECT子句中选择指定的数据列、使用文字串、改变列标题、执行数据运算、使用ALL关键字、使用DISTINCT关键字等。

选择指定的数据列

- 选择指定的数据列是指可以在**SELECT**子句中指定将要检索的列名称。选择指定的列名称要注意几点，第一，这些列名称应该与表中定义的列名称一致，否则就可能出错或者得到意想不到的结果；第二，列名称之间的顺序既可以与表中定义的列顺序相同，也可以不相同；第三，**SELECT**语句的检索结果只是影响数据的显示，对表中数据的存储没有任何的影响。

使用文字串

- 通常，直接阅读**SELECT**语句的检索结果，是一件头疼的事情，因为显示出来的数据，只是一些不连贯的、阅读性不强的信息。为了提高**SELECT**语句检索结果的可读性，可以通过在**SELECT**关键字后面增加文字串。通常情况下，使用单引号将文字串引起来。

改变列标题

- 在默认情况下，在数据检索结果中所显示出来的列标题就是在定义表时使用的列名称。但是，在检索过程中可以根据用户的需要改变显示的列标题。实际上，改变列标题也就是为指定的列定义一个别名。改变列标题有两种方法，一种方法是使用等号(=)，另一种方法是使用AS关键字。

数据运算

- 数据运算就是指对检索的数据进行各种运算。也就是说，可以在**SELECT**关键字后面列出的列项中使用各种运算符和函数。这些运算符和函数包括算术运算符、数学函数、字符串函数、日期和时间函数、系统函数等。

使用ALL和DISTINCT关键字

- 在SELECT语句中，可以在SELECT子句中通过使用ALL或DISTINCT关键字控制查询结果集的显示样式。ALL关键字表示检索所有的数据，包括重复的数据行。DISTINCT关键字表示仅仅显示那些不重复的数据行，重复的数据行只是显示一次。由于ALL关键字是默认值，所以当没有显式使用ALL或DISTINCT关键字时，隐含着使用ALL关键字。

7.7 排序

- 在使用**SELECT**语句时，排序是一种常见的操作。排序是指按照指定的列或其他表达式对结果集进行排列顺序的方式。**SELECT**语句中的**ORDER BY**子句负责完成排序操作。
- 在排序时，既可以按照升序排列，也可以按照降序排列。关键字**ASC**表示升序，**DESC**表示降序，默认情况下是升序。

7.8 使用WHERE子句选择数据

- 在SELECT语句中，WHERE子句指定将要搜索的数据行的条件。也就是说，只有满足WHERE子句条件的数据行才会出现在结果集中。
- 这些搜索条件可以分为
  - 简单搜索条件
  - 模糊搜索条件
  - 复合搜索条件

简单搜索条件

- 在**WHERE**子句中，简单搜索条件是指使用比较运算符、范围、列表、合并以及取反等运算方式形成的搜索条件。

模糊搜索条件

- 在检索字符数据时，通常提供的检索条件是不十分准确的，例如这种搜索条件仅仅是包含、类似某种样式的字符。在**WHERE**子句中，可以使用**LIKE**关键字实现这种灵活的模糊搜索条件。
- **LIKE**关键字用于检索与特定字符串匹配的字符数据。**LIKE**关键字后面可以跟一个列值的一部分而不是一个完整的列值，从而形成**LIKE**子句。**LIKE**子句的语法形式如下：
  - **match_expression [NOT] LIKE pattern [ESCAPE escape_character]**

复合搜索条件

- 在**WHERE**子句中可以使用逻辑运算符把若干个搜索条件合并起来，组成复杂的复合搜索条件。
- 这些逻辑运算符包括**AND**，**OR**和**NOT**。

7.9 聚合技术

- 聚合技术是指对一组数据进行聚合运算得到聚合值的过程。在聚合运算中主要是使用聚合函数。在**Microsoft SQL Server 2005**系统中，一般情况下，可以在三个地方使用聚合函数，即**SELECT**子句、**COMPUTE**子句和**HAVING**子句。本节主要讲述如何在**SELECT**子句和**COMPUTE**子句中使用聚合函数，有关**HAVING**子句使用聚合函数的内容在下一节介绍

SELECT子句中的聚合

- 在SELECT子句中可以使用聚合函数进行运算，运算结果作为新列出现在结果集中。在聚合运算的表达式中，可以包括列名、常量以及由算术运算符连接起来的函数。
- 例如，在如图7-34所示的示例中，在SELECT子句中使用聚合函数计算了Production.Product表中的数据量以及有关标准成本的最大值、最小值、平均值、标准偏差、方差等。注意COUNT函数的特点。

COMPUTE子句中的聚合

- 需要指出的是，当在SELECT子句中出现聚合函数时，结果集中的数据全是聚合值，没有明细值。这是使用SELECT子句计算聚合值的缺点。能否解决这种问题呢？能，解决问题的方法就是使用COMPUTE子句。

7.10 分组技术

- 聚合函数只能产生一个单一的汇总数据，使用**GROUP BY**子句，则可以生成分组的汇总数据。**GROUP BY**子句把数据组织起来分成组。一般情况下，可以根据表中的某一列进行分组，通过使用聚合函数，对每一个组可以产生聚合值。
- 如果希望过滤某些分组，可以使用**HAVING**子句排。分组技术是指使用**GROUP BY**子句完成分组操作的技术。
- 如果在**GROUP BY**子句中没有使用**CUBE**或**ROLLUP**关键字，那么表示这种分组技术是普通分组技术。

普通分组技术

- **GROUP BY**子句、**HAVING**子句和聚合函数一起完成对每一个组生成一行和一个汇总值。
- 在使用**GROUP BY**子句和**HAVING**子句的过程中，要求考虑一些条件

ROLLUP和CUBE关键字

- 在GROUP BY子句中，可以使用ROLLUP或CUBE关键字获得附加的分组数据，这些附加的分组数据是通过各组之间的组合得到的。使用ROLLUP关键字可以得到各组的单项组合，但是CUBE关键字可以得到各组之间的任意组合。在结果集中，通过组组合起来的组名称是NULL，可以使用GROUPING函数来判断该组是否是经过组合得到的。实际上，使用CUBE关键字可以生成多维数据。
- 下面，通过一个示例讲述这两个关键字的特点。

7.11 连接技术

- 在设计表时，为了提高表的设计质量，经常把相关数据分散在不同的表中。但是，在使用数据时，需要把这些数据集中在一个查询语句中。连接技术可以满足这种客观需求。
- 在**Microsoft SQL Server 2005**系统中，这种连接操作又可以细分为交叉连接、内连接、外连接等。下面分别研究这些连接技术。

交叉连接

- 交叉连接也被称为笛卡尔乘积，返回两个表的乘积。在检索结果集中，包含了所连接的两个表中所有行的全部组合。
- 例如，如果对**A**表和**B**表执行交叉连接，**A**表中有**5**行数据，**B**表中有**12**行数据，那么结果集中可以有**60**行数据。

内连接

- 内连接把两个表中的数据连接生成一个第3个表，在这个第3个表中，仅包含那些满足连接条件的数据行。在内连接中，使用**INNER JOIN**连接运算符，并且使用**ON**关键字指定连接条件。
- 内连接是一种常用的连接方式，如果在**JOIN**关键字前面没有明确指定连接类型，那么默认的连接类型是内连接。

外连接

- 在外连接中，不仅包括那些满足条件的数据，而且某些表不满足条件的数据也会显示在结果集中。也就是说，外连接只限制其中一个表的数据行，而不限制另外一个表中的数据。
- 在**Microsoft SQL Server 2005**系统中，可以使用的**3**种外连接关键字，即**LEFT OUTER JOIN**、**RIGHT OUTER JOIN**和**FULL OUTER JOIN**。

7.12 子查询技术

- SELECT语句可以嵌套在其他许多语句中，这些语句包括SELECT、INSERT、UPDATE或DELETE等，这些嵌套的SELECT语句被称为子查询。当一个查询依赖于另外一个查询结果时，那么可以使用子查询。在某些查询中，查询语句比较复杂不容易理解，因此为了把这些复杂的查询语句分解成多个比较简单的查询语句形式时，可以使用子查询方式。
- 使用子查询方式完成查询操作的技术是子查询技术。

7.13 集合运算技术

- 查询语句的结果集往往是一个包含了多行数据集合。集合之间可以进行并、差、交等运算。
- 在**Microsoft SQL Server 2005**系统中，两个查询语句之间也可以进行集合运算。其中，**UNION**运算符表示并集运算，**EXCEPT**运算符从左查询中返回右查询中没有找到的重复值，**INTERSECT**运算符则返回左右两个查询语句都包含的所有非重复值。
- 需要注意的是，在集合运算时，所有查询语句中的列的数量和顺序必须相同，且数据类型必须兼容。

7.14 公用表表达式

- 在Microsoft SQL Server 2005系统中，可以使用公用表表达式(common table expression，CTE)。CTE是定义在SELECT、INSERT、UPDATE或DELETE语句中的临时命名的结果集，CTE也可以用在视图的定义中。
- 在CTE中，可以包括对自身的引用，因此这种表达式也被称为递归CTE。

7.15 PIVOT和UNPIVOT

- PIVOT和UNPIVOT都是Microsoft SQL Server 2005系统中新增的关系运算符，提供了一种把列数据转换为行数据的方式。PIVOT运算符把表达式中某一列中的唯一数据转换为输出中的多个列，UNPIVOT运算符则相反。
- 在图7-50所示的示例中，PIVOT运算符与SELECT语句一起使用，创建了一个新的结果集。COUNT (PurchaseOrderID)合计了每个轴转值的订单数，FOR关键字指定了要轴转的列是EmployeeID，IN关键字列表中确定的值是轴转列中被用作列标题的值。

7.16 加密表中数据

- 下面，通过一个示例讲述如何加密表中的数据。
- 该示例的加密过程如图**7-51**所示。

第8章 索引和查询优化

教学目标

教学过程

教学目标

- 理解索引的优点和缺点
- 理解堆的结构特点
- 理解聚集索引和非聚集索引的特点
- 理解索引的类型
- 使用**CREATE INDEX**语句创建索引的方式
- 理解索引统计信息的特点和获得方式
- 理解查询优化的方式

教学过程

8.1 概述

8.2 索引的类型和特点

8.3 创建索引

8.4 索引维护

8.5 查询优化

8.1 概述

- 在Microsoft SQL Server系统中，可管理的最小空间是页。一个页是8KB字节的物理空间。插入数据的时候，数据就按照插入的时间顺序被放置在数据页上。一般地，放置数据的顺序与数据本身的逻辑关系之间是没有任何联系的。因此，从数据之间的逻辑关系方面来讲，数据是乱七八糟堆放在一起的。
- 数据的这种堆放方式称为堆。当一个数据页上的数据堆放满之后，数据就得堆放在另外一个数据页上，这时就称为页分解。
- 索引是一种与表或视图关联的物理结构，可以用来加快从表或视图中检索数据行的速度。

8.2 索引的类型和特点

- 在Microsoft SQL Server 2005系统中，有两种基本的索引类型：聚集索引和非聚集索引。除此之外，还有惟一性索引、包含性列索引、索引视图、全文索引、XML索引等。在这些索引类型中，聚集索引和非聚集索引是数据库引擎中索引的基本类型，是理解惟一性索引、包含性列索引、索引视图的基础，本节主要研究者两种索引类型。
- 另外，为了更好地理解索引结构，有必要对堆结构有所了解。
- 最后，简单介绍一下系统访问数据的方式。

堆

- 堆是不含聚集索引的表，表中的数据没有任何的顺序。堆的信息记录在**sys.partitions**目录视图中。每一个堆都可能有多个不同的分区，每一个分区都有一个堆结构，每一个分区在**sys.partitions**目录视图中都有一行，且**index_id=0**。也就是说，每一个堆都可能有多个堆结构。

聚集索引

- 聚集索引是一种数据表的物理顺序与索引顺序相同的索引，非聚集索引则是一种数据表的物理顺序与索引顺序不相同的索引。
- 聚集索引的叶级和非叶级构成了一个特殊类型的B树结构。B树结构中的每一页称为一个索引节点。索引的最低级节点是叶级节点。在一个聚集索引中，某个表的数据页是叶级，在叶级之上的索引页是非叶级。在聚集索引中，页的顺序是有序的。

非聚集索引

- 非聚集索引与聚集索引具有相同的B树结构，但是，在非聚集索引中，基础表的数据行不是按照非聚集键的顺序排序和存储，且非聚集索引的叶级是由索引页而不是由数据页组成。
- 非聚集索引既可以定义在表或视图的聚集索引上，也可以定义在表或视图的堆上。非聚集索引中的每一个索引行都是由非聚集键值和行定位符组成，该行定位符指向聚集索引或堆中包含该键值的数据行。如果表或视图中没有聚集索引(堆)，则行定位符是指向行的指针RID，RID由文件标识符ID、页码和页上的行数生成。

其他类型的索引

- 除了聚集索引和非聚集索引之外，**Microsoft SQL Server 2005**系统还提供了一些其他类型的索引或索引表现形式，这些内容包括
  - 惟一性索引
  - 包含性列索引
  - 索引视图
  - 全文索引
  - **XML**索引

访问数据的方式

- 第一种方法是表扫描，就是指系统将指针放在该表的表头数据所在的数据页上，然后按照数据页的排列顺序，一页一页地从前向后扫描该表数据所占有的全部数据页，直至扫描完表中的全部记录。
- 第二种方法是使用索引查找。

8.3 创建索引

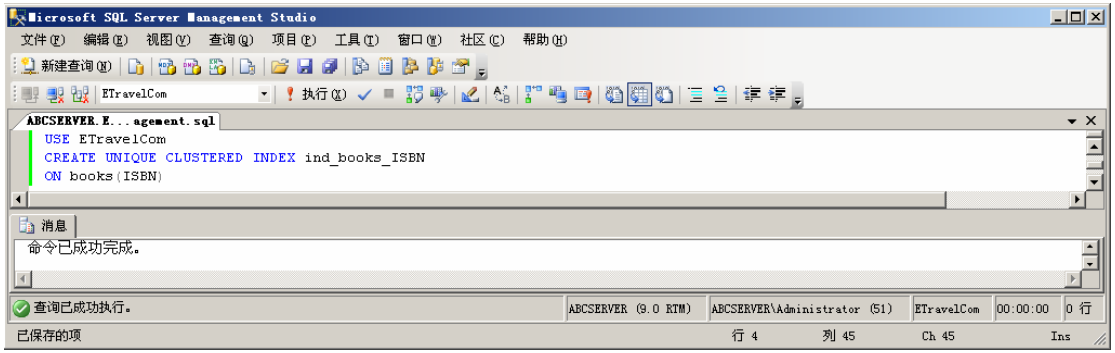
- 在Microsoft SQL Server 2005系统中，既可以直接创建索引，也可以间接创建索引。当直接创建索引时，既可以使用CREATE INDEX语句，也可以使用图形工具。

直接方法和间接方法

- 可以把创建索引的方式分为直接方法和间接方法。
- 直接创建索引的方法就是使用命令和工具直接创建索引。
- 间接创建索引就是通过创建其他对象而附加创建了索引，例如在表中定义主键约束或惟一性约束时，同时也创建了索引。
- 虽然，这两种方法都可以创建索引，但是，它们创建索引的具体内容是有区别的。

使用CREATE INDEX语句

- 在Microsoft SQL Server 2005系统中，使用CREATE INDEX语句可以在关系表上创建索引



数据库引擎优化顾问

- 使用**Microsoft SQL Server 2005**的数据库引擎优化顾问，用户可以方便地选择和创建索引、索引视图和分区的最佳集合。数据库引擎优化顾问分析一个或多个数据库的工作负荷和实现，其中工作负荷是对要优化的一个或多个数据库执行的一组**Transact-SQL**语句。数据库引擎优化顾问的输入是由**SQL Server Profiler**生成的跟踪文件、指定的跟踪表或工作负荷。数据库引擎优化顾问的输出是修改数据库的物理设计结构的建议，其中物理设计结构包括聚集索引、非聚集索引、索引视图、分区等。

查看索引信息

- 在Microsoft SQL Server 2005系统中，可以使用一些目录视图和系统函数查看有关索引的信息。
- 这些目录视图和系统函数如表8-1所示。

8.4 索引维护

- 索引在创建之后，由于数据的增加、删除、更新等操作使得索引页发生碎块，因此为了提高系统的性能，必须对索引进行维护。
- 这些维护包括查看碎块信息、维护统计信息、分析索引性能、删除重建索引等。

查看索引统计信息

- 索引统计信息是查询优化器用来分析和评估查询、确定最优查询计划的基础数据。一般地，用户可以通过常用的方式访问指定索引的统计信息。
  - 一种方式是使用**DBCC SHOW_STATISTICS**命令
  - 另一种是使用图形化工具

查看索引碎片信息

- 可以使用两种方式查看有关索引的碎片信息，使用**sys.dm_db_index_physical_stats**系统函数和使用图形化工具。注意，**sys.dm_db_index_physical_stats**系统函数替代了以前版本中的**DBCC SHOWCONTIG**命令。

维护索引统计信息

- 统计信息是存储在**Microsoft SQL Server**中的列数据的样本。这些数据一般地用于索引列，但是还可以为非索引列创建统计。**Microsoft SQL Server**维护某一个索引关键值的分布统计信息，并且使用这些统计信息来确定在查询进程中哪一个索引是有用的。查询的优化依赖于这些统计信息的分布准确度。查询优化器使用这些数据样本来决定是使用表扫描还是使用索引。
- 当表中数据发生变化时，**Microsoft SQL Server**周期性地自动修改统计信息。索引统计被自动地修改，索引中的关键值显著变化。

8.5 查询优化

- 在很多情况下，为了达到同样的结果，可以写出多个不同的查询形式。但是，不同的查询形式往往消耗的时间不相同，因此有不同的性能。如何提高查询语句的性能呢？下面，介绍**Microsoft SQL Server**查询优化器和优化隐藏的特点。
- 在查询语句中，**Microsoft SQL Server**系统是如何判断是否使用索引或使用哪些索引呢？一般地，系统是根据索引的选择性和索引类型。如果索引列的选择性很高，也就是说，索引列中的只有很少几行数据将被选中，那么应该使用索引。系统如何得到选择性呢？这就需要系统的统计信息来确定。下面，通过一个示例讲述系统是如何选择索引执行查询操作的。

第9章 数据完整性

教学目标

教学过程

教学目标

- 理解操纵数据时的问题和解决机制
- 理解约束的基本概念和类型
- 掌握管理**DEFAULT**约束技术
- 掌握管理**CHECK**约束技术
- 掌握管理主键约束技术
- 掌握管理**UNIQUE**约束技术
- 掌握管理外键约束技术
- 理解禁止约束的场景

教学过程

- 9.1 概述
- 9.2 约束的概念和类型
- 9.3 管理约束
- 9.4 禁止约束

9.1 概述

- 本节讨论两个方面的内容。首先，分析操纵数据时经常遇到的问题。其次，提出解决这些问题的方法。
- 当操纵表中数据时，由于种种原因，我们经常遇到一些问题。
- 诸如此类的问题，不能仅仅依靠数据录入人员和操纵人员的认真和负责，而是应该有一套保障机制：要么防止这些问题发生，要么发生这些问题是可以及时地发现。数据完整性就是解决这些问题的机制。
- 数据完整性就是指存储在数据库中的数据的一致性和准确性。在评价数据库的设计时，数据完整性的设计是数据库设计好坏的一项重要指标。
- 在**Microsoft SQL Server 2005**系统中，有**3**种数据完整性类型，即域完整性、实体完整性和引用完整性。

9.2 约束的概念和类型

- 约束是通过限制列中数据、行中数据和表之间数据来保证数据完整性的非常有效的方法。约束可以确保把有效的数据输入到列中和维护表和表之间的特定关系。
- **Microsoft SQL Server 2005**系统提供了**5**种约束类型，即**PRIMARY KEY(**主键**)**、**FOREIGN KEY(**外键**)**、**UNIQUE**、**CHECK**、**DEFAULT**约束。
- 每一种数据完整性类型，例如域完整性、实体完整性和引用完整性，都由不同的约束类型来保障。表**9-1**描述了不同类型的约束和完整性之间的关系。

9.3 管理约束

- 本节详细研究各种**DEFAULT**、**CHECK**、主键、**UNIQUE**、外键等约束的特点、创建方式、修改等内容。

DEFAULT约束

- 当使用INSERT语句插入数据时，如果没有为某一个列指定数据，那么DEFAULT约束就在该列中输入一个值。
- 例如，在记录了人事信息的person表的性别列中定义了一个DEFAULT约束为“男”。那么当向该表中输入数据时，如果没有为性别列提供数据，那么DEFAULT约束把缺省值“男”自动插入到该列中。因此，DEFAULT约束可以实现保证域完整性。
- 定义DEFAULT约束的基本语法在CREATE TABLE语句中和在ALTER TABLE语句中的形式不完全相同。

CHECK约束

- CHECK约束用来限制用户输入某一个列的数据，即在该列中只能输入指定范围的数据。CHECK约束的作用非常类似于外键约束，两者都是限制某个列的取值范围，但是外键是通过其他表来限制列的取值范围，CHECK约束是通过指定的逻辑表达式来限制列的取值范围。

主键约束

- 主键约束在表中定义一个主键值，这是惟一确定表中每一行数据的标识符。在所有的约束类型中，主键约束是最重要的一种约束类型，也是使用最广泛的约束类型。该约束强制实体完整性。一个表中最多只能有一个主键，且主键列不允许空值。

UNIQUE约束

- **UNIQUE**约束指定表中某一个列或多个列不能有相同的两行或两行以上的数据存在。这种约束通过实现惟一性索引来强制实体完整性。当表中已经有了一个主键约束时，如果需要在其他列上实现实体完整性，又因为表中不能有两个或两个以上的主键约束，所以只能通过创建**UNIQUE**约束来实现。

外键约束

- 外键约束强制引用完整性。外键约束定义一个列或多个列，这些列可以引用同一个表或另外一个表中的主键约束列或**UNIQUE**约束列。实际上，通过创建外键约束可以实现表和表之间的依赖关系。一般情况下，在**Microsoft SQL Server**关系型数据库管理系统中，表和表之间经常存在着大量的关系，这些关系都是通过定义主键约束和外键约束实现的。

9.4 禁止约束

- 在表上定义约束之后，就可以依靠约束来保证表中数据的完整性。但是，在某些特殊的情况下，需要禁止在已有的数据上应用定义的约束或者禁止在加载数据时应用约束。下面讨论有关禁止约束的内容

禁止在已有的数据上应用约束

- 一般地，当在一个已经有数据的表上定义约束时，**Microsoft SQL Server**系统自动检查这些数据是否满足约束条件。然而，在某些特殊情况下，还可以禁止约束检查已经存在的数据是否满足约束的定义。
- 禁止在已有的数据上应用约束，实际上就是说这些约束对表中已有的数据不起作用。这种禁止只有在向表中增加约束时才能指定。

禁止在加载数据时应用约束

- 对于**CHECK**约束和外键约束，除了可以禁止对已有的数据应用约束之外，也可以在加载数据时禁止应用这些定义的约束。
- 也就是说，在更新表中数据或向表中添加数据时，可以不判断这些数据是否与所定义的**CHECK**约束或外键约束冲突。

第10章 视图

教学目标

教学过程

教学目标

- 数据管理中存在的问题和解决思路
- 视图的概念、特点和类型
- 使用**CREATE VIEW**语句创建视图
- 创建索引视图
- 查看和加密视图定义文本
- 通过视图修改基表中的数据
- 使用图形工具管理视图

教学过程

10.1 概述

10.2 视图的概念、特点和类型

10.3 创建视图

10.4 通过视图修改数据

10.5 使用图形化工具定义视图

10.1 概述

- 数据是存储在表中，对数据的操纵主要是通过表进行的。但是，仅仅通过表操纵数据会带来一系列的性能、安全、效率等问题。下面，对这些问题进行分析。

问题

- 从业务数据角度来看，由于数据库设计时考虑到数据异常等问题，同一种业务数据有可能被分散在不同的表中，但是对这种业务数据的使用经常是同时使用的。
- 从数据安全角度来看，由于工作性质和需求不同，不同的操作人员只是需要查看表中的部分数据，不能查看表中的所有数据。
- 从数据的应用角度来看，例如，一个报表中的数据往往来自于多个不同的表中。在设计报表时，需要明确地指定数据的来源途径和方式。

10.2 视图的概念、特点和类型

- 视图是查看数据库表中数据的一种方式。视图提供了存储预定义的查询语句作为数据库中的对象以备以后使用的能力。视图是一种逻辑对象，是一种虚拟表。除非是索引视图，否则视图不占物理存储空间。在视图中被查询的表称为视图的基表。大多数的**SELECT**语句都可以用在视图的创建中。

特点

- 使用视图有许多优点，例如集中用户使用的数据、掩码数据的复杂性、简化权限管理以及为向其他应用程序输出而重新组织数据等等。

类型

- 在Microsoft SQL Server 2005系统中，可以把视图分成三种类型，即标准视图、索引视图和分区视图。
- 一般情况下的视图都是标准视图，它是一个虚拟表，不占物理存储空间。如果希望提高聚合多行数据的视图性能，那么可以创建索引视图。索引视图是被物理化的视图，它包含有经过计算的物理数据。通过使用分区视图，可以连接一台或多台服务器中成员表中的分区数据，使得这些数据看起来就象来自一个表中一样。

10.3 创建视图

- 在Microsoft SQL Server 2005系统中，主要使用CREATE VIEW语句创建视图。
- 只能在当前数据库中创建视图。
- 当创建视图时，Microsoft SQL Server首先验证视图定义中所引用的对象是否存在。

10.4 通过视图修改数据

- 无论在什么时候修改视图的数据，实际上都是在修改视图的基表中的数据。在满足一定的限制条件下，可以通过视图自由地插入、删除和更新基表中的数据。
- 在修改视图时，要注意下列一些条件：
  - 不能同时影响两个或两个以上的基表。
  - 某些列不能修改，包括通过计算得到值的列、有内置函数的列或有合计函数的列等。
  - 如果影响到表中那些没有缺省值的列，那么可能引起错误。
  - 如果在视图定义中指定了**WITH CHECK OPTION**选项，那么系统验证所修改的数据。

10.5 使用图形化工具定义视图

- 除了使用**CREATE VIEW**语句之外，也可以使用图形化工具定义视图。
- 在**SQL Server Management Studio**环境中的“对象资源管理器”中，打开指定的服务器实例，打开“数据库”节点，打开指定的数据库例如**AdventureWorks**数据库节点，选中“视图”节点。右击“视图”节点，从弹出的快捷菜单中选择“新建视图”命令，则出现如图**10-9**所示的“添加表”对话框。

第11章 存储过程、触发器和函数

教学目标

教学过程

教学目标

- 理解存储过程的特点、类型和作用
- 使用**CREATE PROCEDURE**语句创建存储过程
- 理解存储过程的执行方式
- 理解**DML**触发器的特点和创建方式
- 理解**DML**触发器的工作原理
- 使用**CREATE TRIGGER**语句创建**DML**触发器
- 理解**DDL**触发器的特点和创建方式
- 理解用户定义函数的类型和特点
- 使用**CREATE FUNCTION**语句创建用户定义函数

教学过程

11.1 存储过程

11.2 触发器

11.3 用户定义函数

11.1 存储过程

- 存储过程的特点和类型
- 创建存储过程的规则
- 使用**CREATE PROCEDURE**语句创建存储过程
- 执行存储过程
- 修改和删除存储过程
- 存储过程的执行过程
- 查看存储过程的信息

存储过程的特点和类型

- 存储过程是一个可重用的代码模块，可以高效率地完成指定的操作。在**Microsoft SQL Server 2005**系统中，既可以使用**Transact-SQL**语言编写存储过程，也可以使用**CLR**方式编写存储过程。使用**CLR**编写存储过程是**Microsoft SQL Server 2005**系统与**.NET**框架紧密集成的一种表现形式。
- 在**Microsoft SQL Server 2005**系统中，提供了三种基本的存储过程类型，即用户定义的存储过程、扩展存储过程和系统存储过程。

创建存储过程的规则

- 在设计和创建存储过程时，应该满足一定的约束和规则。只有满足了这些约束和规则，才可以创建有效的存储过程。
- 虽然说在**CREATE PROCEDURE**语句中可以包括任意数量和类型的**Transact-SQL**语句，但是某些特殊的语句是不能包含在存储过程定义中的。

使用CREATE PROCEDURE语句创建存储过程

- CREATE PROCEDURE语句的基本语法形式如下所示：
  - CREATE PROCEDURE procedure_name
  - parameter_name data_type, …
  - WITH procedure_option
  - AS
  - sql_statement

执行存储过程

- 在Microsoft SQL Server 2005系统中，可以使用EXECUTE语句执行存储过程。EXECUTE语句也可以简写为EXEC。如果将要执行的存储过程需要参数，那么应该在存储过程名称后面带上参数值。

修改和删除存储过程

- 在Microsoft SQL Server 2005系统中，可以使用ALTER PROCEDURE语句修改已经存在的存储过程。
- 修改存储过程，而不是删除和重建存储过程，其目的是保持存储过程的权限不发生变化。
- 例如，如果修改HumanResources.GetEmployeeInfo存储过程，那么与该存储过程对象相关的权限将不会发生任何变化。
- 但是，如果删除HumanResources.GetEmployeeInfo存储过程并且重新创建同名的存储过程，那么该存储过程对象相关的权限都需要重新定义。

存储过程的执行过程

- 语法分析阶段是指创建存储过程时，系统检查其创建语句的语法正确性的过程。
- 解析阶段是指某个存储过程首次执行时，查询处理器从**sys.sql_modules**目录视图中读取该存储过程的文本并且检查该过程引用的对象名称是否存在的过程。
- 编译阶段是指分析存储过程和生成存储过程执行计划的过程。
- 执行阶段是指执行驻留在过程高速缓冲存储区中的存储过程的执行计划的过程。

查看存储过程的信息

- 在Microsoft SQL Server 2005系统中，可以使用系统存储过程和目录视图查看有关存储过程的信息。
- 如果希望查看存储过程的定义信息，那么可以使用sys.sql_modules目录视图、OBJECT_DEFINITION元数据函数、sp_helptext系统存储过程等。
- 除此之外，使用sp.sql_dependencies、sp_depends等系统存储过程可以查看存储过程的依赖信息。
- 使用sys.objects、sps.procedures、sys.parameters、sys.numbered_procedures等目录视图可以查看有关存储过程的名称、参数等信息。

11.2 触发器

- 触发器的概念和类型
- DML触发器的类型
- 创建DML触发器
- DML触发器的工作原理
- 一个DML触发器示例
- DDL 触发器

触发器的概念和类型

- 一般地认为，触发器是一种特殊类型的存储过程，它包括了大量的**Transact-SQL**语句。
- 按照触发事件的不同，可以把**Microsoft SQL Server 2005**系统提供的触发器分成两大类型，即**DML**触发器和**DDL**触发器。

DML触发器的类型

- 按照触发器事件类型的不同，可以把Microsoft SQL Server 2005系统提供的DML触发器分成3种类型，即INSERT类型、UPDATE类型和DELETE类型。这也是DML触发器的基本类型。

创建DML触发器

- DML触发器是一种特殊类型的存储过程，所以DML触发器的创建和存储过程的创建方式有很多相似的地方。可以使用CREATE TRIGGER语句创建DML触发器。在CREATE TRIGGER语句中，指定了定义触发器的基表或视图、触发事件的类型和触发的时间、触发器的所有指令等内容。
- 创建DML触发器的CREATE TRIGGER语句的基本语法形式如下：
  - CREATE TRIGGER trigger_name
  - ON table_name_or_view_name
  - WITH ENCRYPTION
  - { FOR | AFTER | INSTEAD OF } {[DELETE] [,] [INSERT] [,] [UPDATE] }
  - AS sql_statement

DML触发器的工作原理

- 当向表中插入数据时，**INSERT**触发器触发执行。当**INSERT**触发器触发时，新的记录增加到触发器表中和**inserted**表中。该**inserted**表是一个逻辑表，保存了所插入记录的拷贝，允许用户参考**INSERT**语句中数据。触发器可以检查**inserted**表，来确定该触发器的操作是否应该执行和如何执行。在**inserted**表中的那些记录，总是触发器表中一行或多行记录的冗余。

一个DML触发器示例

- 为了更加全面地掌握开发触发器的步骤和技术，本节通过一个具体的示例，全面讲述使用**Transact-SQL**语言开发和创建触发器的技术。
- 一般地，开发触发器的过程包括用户需求分析、确定触发器的逻辑结构、编写触发器代码和测试触发器。

DDL 触发器

- DDL触发器与DML触发器有许多类似的地方，例如可以自动触发完成规定的操作、都可以使用CREATE TRIGGER语句创建等，但是也有一些不同的地方。
- 例如，DDL触发器的触发事件主要是CREATE、ALTER、DROP以及GRANT、DENY、REVOKE等语句，并且触发的时间条件只有AFTER，没有INSTEAD OF。
- 一般地，DDL触发器主要是用于下面一些操作：
  - 防止对数据库架构进行某些更改。
  - 希望数据库中发生某种情况以便相应数据库架构中的更改。
  - 记录数据库架构中的更改或事件。

11.3 用户定义函数

- 用户定义函数的特点
- 创建用户定义函数时的考虑
- 使用**CREATE FUNCTION**语句
- 查看用户定义函数的信息

用户定义函数的特点

- 在Microsoft SQL Server系统中，使用用户定义函数可以带来许多好处，这些好处包括允许模块化设计，只需创建一次函数并且将其存储在数据库中，以后便可以在程序中调用任意次。用户定义函数可以独立于程序源代码进行修改。执行速度更快，就像存储过程一样，使用Transact-SQL编写的用户定义函数通过缓存计划并在重复执行时重用它来降低Transact-SQL代码的编译开销。

创建用户定义函数时的考虑

- 在Microsoft SQL Server 2005系统中，可以分别使用CREATE FUNCTION、ALTER FUNCTION、DROP FUNCTION语句来实现用户定义函数的创建、修改和删除。在创建用户定义函数时，每个完全限定用户函数名称(schema_name.function_name)必须惟一。
- 函数的BEGIN END块中的语句不能有任何副作用。

使用CREATE FUNCTION语句

- 在Microsoft SQL Server 2005系统中，使用CREATE FUNCTION语句可以创建标量函数、内联表值函数、多语句表值函数。
- 需要说明的是，如果RETURNS子句指定了一种标量数据类型，则该函数为标量值。如果RETURNS子句指定了TABLE，则该函数为表值函数。根据函数主体的定义方式，表值函数可以分为内联函数或多语句函数。
- 内联函数可以用于获得参数化视图的功能。

查看用户定义函数的信息

- **Microsoft SQL Server 2005**系统提供了几个可以用于查看用户定义函数的信息的系统存储过程和目录视图。使用这些工具，可以查看用户定义函数的定义、获取函数的架构和创建时间、列出指定函数所使用的对象等信息。
- 可以使用**sys.sql_modules**、**OBJECT_DEFINITION**、**sp_helptext**等工具查看用户定义函数的定义，可以使用**sys.objects**、**sys.parameters**、**sp_help**等工具查看有关用户定义函数的信息，可以使用**sys.sql_dependencies**、**sp_depends**等工具查看用户定义函数的依赖关系。

第12章 备份和还原

教学目标

教学过程

教学目标

- 理解备份和还原的原因和作用
- 理解数据库的恢复模式
- 理解备份前的准备工作和备份特点
- 执行备份操作
- 理解备份方法和备份策略
- 理解还原前的准备工作和还原特点
- 执行还原操作

教学过程

12.1 概述

- 备份就是制作数据库结构和数据的拷贝，以便在数据库遭到破坏的时候能够修复数据库。数据库的破坏是难以预测的，因此必须采取能够还原数据库的措施。一般地，造成数据丢失的常见原因包括：
  - 软件系统瘫痪
  - 硬件系统瘫痪
  - 人为误操作
  - 存储数据的磁盘被破坏
  - 地震、火灾、战争、盗窃等灾难

数据库的恢复模式

- 完整恢复模式
- 大容量日志记录的恢复模式
- 简单恢复模式

12.3 备份基础

- 备份就是制作数据库结构和数据的拷贝。在执行备份操作之前，应该做好相应的计划工作、明确备份的对象、理解备份的动态特点等。
- 下面详细介绍这些内容。

备份前的计划工作

- 确定备份的频率。
- 确定备份的内容。
- 确定使用的介质。
- 确定备份工作的负责人。
- 确定使用在线备份还是脱机备份。
- 是否使用备份服务器。
- 确定备份存储的地方。
- 确定备份存储的期限。

备份的对象

- 在备份的时候，应该确定备份的内容。备份的目的是当系统发生故障或瘫痪之后，应该能够将系统还原到发生故障之前的状态。因此，有必要将系统的全部信息都备份下来。
- 从大的方面上讲，应该备份两方面的内容，一方面是备份记录系统信息的系统数据库，另一方面是备份记录用户数据的用户数据库。

备份的动态特点

- 在Microsoft SQL Server系统中，备份既可以是静态的，也可以是动态的。备份是静态的，表示备份数据库时不允许用户使用数据库。如果说备份是动态，那么在备份数据库时，允许用户继续在数据库中操作。

12.4 执行备份操作

- 在执行备份操作之前，应该创建数据库的备份文件。
- 备份文件既可以是永久性的，也可能是临时性的。
- 然后，把指定的数据库备份到备份文件上。

创建永久性的备份文件

- 执行备份的第一步是创建将要包含备份内容的备份文件。为了执行备份操作，在使用之前所创建的备份文件称为永久性的备份文件。这些永久性的备份文件也称为备份设备。
- 有两种创建永久性备份文件的方法：使用**sp_addumpdevice**系统存储过程；使用**SQL Server Management Studio**。

创建临时性的备份文件

- 除了创建永久性的备份文件或备份设置之外，还可以创建临时性的备份文件。
- 在执行数据库备份过程中产生的备份文件称为临时性的备份文件。

使用多个备份文件来存储备份

- 在执行数据库备份过程中，**Microsoft SQL Server**系统可以同时向多个备份文件写备份内容。这时的备份称为并行备份。如果使用多个备份文件，那么数据库中的数据就分散在这些备份文件中。
- 在执行一次备份过程中，使用到的一个或多个备份文件称为备份集。

BACKUP语句

- BACKUP DATABASE { database_name | @database_name_var }
- TO < backup_device > [,...n]

备份方法

- **Microsoft SQL Server 2005**系统提供了四种基本的备份方法，来满足企业和数据库活动的各种需要。这四种备份方法是：完全数据库备份、增量数据库备份、事务日志备份和数据库文件或文件组备份。
- 这些备份方法的不同组合就产生了不同的备份策略。

12.5 还原

- 备份是一种灾害预防操作，还原则是一种消除灾害的操作。备份是还原的基础，还原是为了实现备份的目的。
- 本节讲述还原数据库的基本概念还原数据库的具体操作。

还原的特点

- 数据库还原就是指加载数据库备份到系统中的进程。在进行数据库还原时，系统首先进行一些安全性检查，这些安全性检查包括：
  - 指定的数据库是否存在
  - 数据库文件是否变化
  - 数据库文件是否兼容
  - 重建数据库及其相关的文件

验证备份的内容

- 在还原数据库之前，应该验证使用的备份文件是否有效和查看备份文件中的内容是否自己所需要的内容。
- 可以使用下面的**RESTORE**语句验证备份的内容：
  - **RESTORE HEADERONLY**
  - **RESTORE FILELISTONLY**
  - **RESTORE LABELONLY**
  - **RESTORE VERIFYONLY**

RESTORE语句

- RESTORE DATABASE
  { database_name |
  @database_name_var }
- [FROM <backup_device> [,...n]]

从不同的备份中还原数据库

- 如果数据库遭到了破坏，那么可以从完全数据库备份中来还原。这种还原也是所有还原操作的基础。
- 如果只使用一个完全数据库的备份，那么可以在还原时使用**RECOVERY**选项。
- 如果有多个将要还原的内容，那么在执行完全数据库还原时使用**NORECOVERY**选项。

第13章 事务

教学目标

教学过程

教学目标

- 并发性的概念和并发性问题的特点
- 事务的概念、类型和特点
- 事务管理技术
- 理解锁的作用
- 掌握定制锁技术
- 掌握查看和理解锁信息技术

教学过程

13.1 概述

13.2 事务的特点、类型和管理

13.3 使用锁

13.1 概述

- 在Microsoft SQL Server 2005系统中，解决并发性问题采取了事务和锁机制。
- 事务和锁是两个紧密联系的概念。事务就是一个单元的工作，包括一系列的操作，这些操作要么全部成功，要么全部失败。
- 锁就是保护指定的资源，不被其他事务操作。

13.2 事务的特点、类型和管理

- 事务是Microsoft SQL Server系统的重要特征，一方面保证了系统的备份和恢复，另一方面实现了数据一致性机制。
- 下面将详细描述事务的概念、工作原理、事务的类型等内容。

事务的概念

- 事务是指一个单元的工作。作为一个逻辑单元，它必须具备4个属性：
  - 自动性
  - 一致性
  - 独立性
  - 持久性

事务的工作原理

- 事务确保数据的一致性和可恢复性。事务开始之后，事务所有的操作都陆续写到事务日志中。
- 系统自动生成一个检查点机制，这个检查点周期地发生。检查点的周期是系统根据用户定义的时间间隔和系统活动的频度由系统自动计算出来的时间间隔。
- 检查点周期地检查事务日志，如果在事务日志中事务全部完成，那么检查点将事务日志中的该事务提交到数据库中，并且在事务日志中做一个检查点提交标记。如果在事务日志中事务没有完成，那么检查点将事务日志中的该事务不提交到数据库中，并且在事务日志中做一个检查点未提交标记。

使用事务时的考虑

- 在使用事务时，原则上应该使事务尽可能短并且要避免事务嵌套。事务应该尽可能短，这是因为比较长的事务增加了事务占用数据的时间，使其他必须等待访问该事务锁定数据的事务延长了等待访问数据的时间。
- 在使用事务时，为了使事务尽可能短，应该采取一些相应的方法。

事务的类型

- 根据系统的设置，可以把事务分成两种类型。一种是系统提供的事务，另一种是用户定义的事务。系统提供的事务是指在执行某些语句时，一条语句就是一个事务。这时要知道，一条语句的对象既可能是表中的一行数据，也可能是表中的多行数据，甚至是表中的全部数据。因此，只有一条语句构成的事务也可能包含了对多行数据的处理。

管理事务

- BEGIN TRANSACTION
- BEGIN DISTRIBUTED TRANSACTION
- COMMIT TRANSACTION
- ROLLBACK TRANSACTION
- SAVE TRANSACTION
- SET IMPLICIT_TRANSACTION

13.3 使用锁

- 锁是实现事务的手段。
- 实际上锁是保护事务和数据的方式，这种保护方式类似于日常生活中使用的锁。
- 下面介绍**Microsoft SQL Server**系统中锁的特点。

锁的概念

- 锁就是防止其他事务访问指定资源的手段。锁是实现并发控制的主要方法，是多个用户能够同时操纵同一个数据库中的数据而不发生数据不一致现象的重要保障。
- 一般来说，锁可以防止脏读、不可重复读和幻觉读。

SQL Server的空间特点

- 在Microsoft SQL Server系统中，最小的空间管理单位是页，一个页有8K。所有的数据、日志、索引都存放在页上。另外，使用页还有一个限制，这就是表中的一行数据必须在同一个页上，不能跨页。页上面的空间管理单位是Extent，一个Extent是8个连续的页。表和索引的最小占用单位是Extent。数据库是由一个或多个表或索引组成的，即是由多个Extent组成。

可以锁定的资源

- 在Microsoft SQL Server中可以锁定的资源有多种，这些可以锁定的资源分别是行、页、Extent、表和数据库，他们对应的锁分别是行级锁、页级锁、Extent级锁、表级锁和数据库级锁。数据行存放在页上，页存放在Extent上，一个表有若干个Extent组成，而若干个表组成了数据库。
- 在这些可以锁定的资源中，最基本的资源是行、页和表，而Extent和数据库是特殊的可以锁定的资源。

锁的类型和其兼容性

- 锁定资源的方式有两种基本形式，一种形式是读操作要求的共享锁，另一种形式是写操作要求的排它锁。除了这两种基本类型的所，还有一些特殊情况的锁，例如意图锁、修改锁和模式锁。在这些各种类型的锁中，某些类型的锁之间是可以兼容的，但多数类型的锁之间是不兼容的。

死锁问题

- 在事务和锁的使用过程中，死锁是一个不可避免的现象。在两种情况下发生死锁。第一种情况是当两个事务分别锁定了两个单独的对象，这时每一个事务都要求在另外一个事务锁定的对象上获得一个锁，因此每一个事务都必须等待另外一个事务释放占有的锁，这时就发生了死锁。这种是最典型的死锁形式。
- 死锁的第二种情况是当在一个数据库中时，有若干个长时间运行的事务执行并行的操作，当查询分析器处理一种非常复杂的查询例如连接查询时，由于不能控制处理的顺序，有可能发生死锁现象。

会话级锁

- 会话级锁的定制包括两个方面，事务隔离等级和锁超时限制。事务隔离等级保护指定的事务，该事务隔离允许对一个会话中的全部事务设置隔离等级。当设置隔离等级时，就为会话中的全部语句指定了默认的锁定行为。
- **Microsoft SQL Server**系统支持**5**种事务隔离等级，这**5**种等级分别是：
  - **READ UNCOMMITTED**
  - **READ COMMITTED**
  - **REPEATABLE READ**
  - **SNAPSHOT**
  - **SERIALIZABLE**

表级锁

- 定制表级锁就是指通过为表指定一个或多个选项来设置表级锁的行为。
- 实际上，定制表级锁，就是使用一种优化隐藏的方式。
- 优化隐藏就是指在**FROM**子句后面，附加上有关行为选项，提高系统识辨操作的能力。

第14章 自动化管理任务

教学目标

教学过程

教学目标

- 自动化管理任务的必要性
- 自动化管理任务的组件
- 作业的基本概念、作用和管理
- 作业管理技术
- 操作员管理技术
- 警报的特点和类型
- 警报管理技术

教学过程

14.1 概述

14.2 作业

14.3 警报

14.1 概述

- 作为一种分布式数据库管理系统，完成许多自动化管理任务是必不可少的功能。自动化管理任务是指系统可以根据预先的设置自动地完成某些任务和操作。
- 一般地，把可以自动完成的任务分成两大类：一类是执行正常调度的任务，另一类是识别和回应可能遇到的问题的任务。
- 执行正常调度的任务，例如在**Microsoft SQL Server**系统中执行一些日常维护和管理的任务，可以包括备份数据库、传输和转换数据、维护索引、维护数据一致性等。
- 另一类任务识别和回应可能遇到的问题，例如对**Microsoft SQL Server**系统出现的错误以及定义监测可能存在问题的性能条件。

14.2 作业

- 作业就是为了完成指定任务而执行的一系列操作。作业管理包括创建作业、定义作业步骤、确定每一个作业步骤的动作流程逻辑、调度作业、创建将要通知的操作员，以及检查和配置作业的历史。在**Microsoft SQL Server**系统中，既可以使用**SQL Server Management Studio**创建作业和操作员，也可以使用系统存储过程创建作业。下面主要介绍如何使用**SQL Server Management Studio**工具管理作业。

定义作业

- 在**SQL Server Management Studio**主窗口左端的树状结构中，打开指定的服务器实例，打开“**SQL Server**管理”节点，右击“作业”节点，则弹出一个快捷菜单，如图**16-2**所示。

定义操作员

- 操作员是指定的用户对象。可以使用**SQL Server Management Studio**创建操作员。在**SQL Server Management Studio**主窗口左端的树状结构中，打开指定的服务器实例，打开“**SQL Server**代理”节点，右击“操作员”节点，则弹出一个快捷式菜单。从中选择“新建操作员”选项，则出现如图**16-11**所示的“新建操作员”对话框。

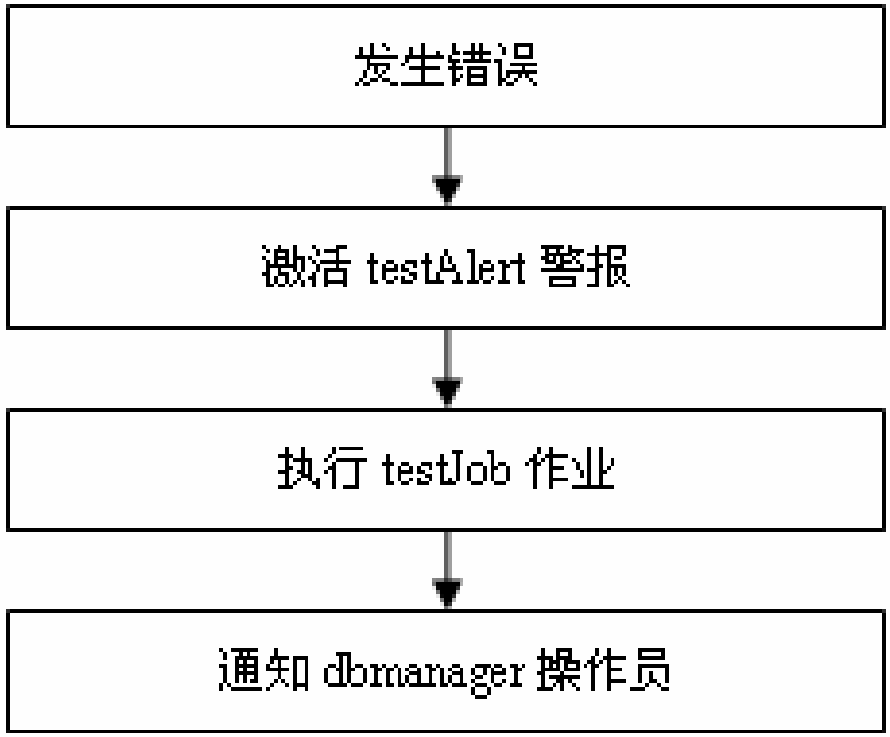
执行和脚本化作业

- 作业创建之后，除了按照其调度方式执行之外，还可以由用户手动执行。在**SQL Server Management Studio**主窗口中，右击作业**testJob**，则弹出一个快捷菜单，如图**14-12**所示。

14.3 警报

- 警报是联系写入Windows事件日志中的Microsoft SQL Server错误消息和执行作业或发送通知的桥梁。下面介绍有关警报的概念及其创建方法等内容。
- 警报负责回应Microsoft SQL Server系统或用户定义的已经写入到Windows应用程序日志中的错误或消息。警报管理包括创建警报、指定错误的代号和严重等级、提供错误消息的文本，以及确定是否将发生的错误或消息写入Windows的应用程序日志中。

警报的执行过程示意图



第15章 性能监视和调整

教学目标

教学过程

教学目标

- 监视Microsoft SQL Server系统的原因和目标
- 影响系统性能的因素
- 性能调整的策略框架和步骤
- 监视和调整系统性能的工具
- Windows系统监视器的作用和使用方式
- SQL Server Profiler工具的作用和使用方式
- C2审核的作用和方式
- 常用的性能监视和调整任务

教学过程

15.1 概述

15.2 影响系统性能的因素

15.3 性能监视和调整的策略

15.4 性能监视和调整的工具

15.5 SQL Server Profiler

15.6 标准审核和C2审核

15.7 常用的监视和调整任务

15.1 概述

- 优化SQL Server的应用程序
- 最小化用户执行查询的响应时间
- 最大化系统的吞吐量
- 检查数据的一致性等

15.2 影响系统性能的因素

- 影响系统性能的因素非常多，为了更好地分析这些影响系统性能的因素，可以把这些因素分成六大类：
  - 服务器硬件类
  - 操作系统类
  - 网络类
  - **SQL Server系统类**
  - 数据库应用程序类
  - 客户应用程序类

15.3 性能监视和调整的策略

- 制订监视和调整系统的策略
- 选择调整性能的方案
- 开发性能监视和调整的具体方法
- 建立系统的性能基线
- 检测性能的瓶颈
- 了解通常的监视任务

15.4 性能监视和调整的工具

- Microsoft SQL Server和Microsoft Windows包括了一些用于监视服务器活动的工具。理解这些工具的特点和合理地使用这些工具，才能做好系统的监视和调整工作。
- 由于SQL Server Profiler以及标准审核和C2审核的重要性，后面有专门2节介绍他们的特点和使用方式。

Windows事件查看器

- “Windows事件查看器”工具用于确认引发性能问题的事件。可以使用该工具提供的信息进行深入地研究和分析。
- 使用“Windows事件查看器”工具可以查看三种事件日志，即应用程序日志、系统日志和安全性日志。这三种事件日志的特点如表15-1所示。

Windows系统监视器

- 如果希望跟踪服务器的活动信息和性能统计，那么可以使用“Windows系统监视器”工具。“Windows系统监视器”工具有许多不同的性能计数器，每一个性能计数器都标志着计算机资源的使用状况。使用“Windows系统监视器”工具可以监视有关Microsoft SQL Server的信息如下：
  - SQL Server的读入/写出
  - SQL Server的内存使用状况
  - SQL Server的用户连接信息
  - SQL Server的锁信息
  - 复制活动状况

Transact-SQL语句

- 除了使用图形化工具之外，还可以使用某些**Transact-SQL**语句监视**Microsoft SQL Server**的性能，这些语句包括系统存储过程、全局变量、**SET**语句、**DBCC**语句和跟踪标志等。

SQL编辑查询器窗口

- **SQL编辑查询器窗口是SQL Server Management Studio工具上的执行查询语句的窗口。除了具备执行查询语句的功能之外，还具备监视系统性能的功能。使用SQL编辑查询器窗口可以监视的系统性能包括：**
  - 显示查询执行规划
  - 显示服务器活动跟踪
  - 显示服务器端的统计信息
  - 显示客户机端的统计信息

15.5 SQL Server Profiler

- **SQL Server Profiler**工具可以用来跟踪服务器和数据库的各种活动。可以把这些活动捕捉到表中、文件中或某个脚本文件，以便以后分析使用。
- 使用**SQL Server Profiler**工具的过程包括创建跟踪、运行和重现跟踪等。

15.6 标准审核和C2审核

- 标准审核是指通过**SQL Server Management Studio**工具或者其他工具设置审核级别进行的对**Microsoft SQL Server**系统登录操作的审核活动。
- **C2**审核是一种可以审核更加广泛活动的审核方式。

15.5 常用的监视和调整任务

- 常用的监视和调整任务包括监视内存的使用状况、监视线程和处理器的使用状况、监视硬盘的输入/输出、监视锁的信息、监视性能差的查询语句。
- 常用的监视系统性能的计数器如表15-6所示。

第16章 Service Broker

教学目标

教学过程

教学目标

- 同步通信方式和异步通信方式的特点
- **Service Broker体系架构的特点**
- 定义**Service Broker**对象
- 操纵**Service Broker**对象
- 启用**Service Broker**系统
- 开发**Service Broker**应用程序

教学过程

16.1 概述

16.2 Service Broker体系架构

16.3 开发Service Broker应用程序的工具

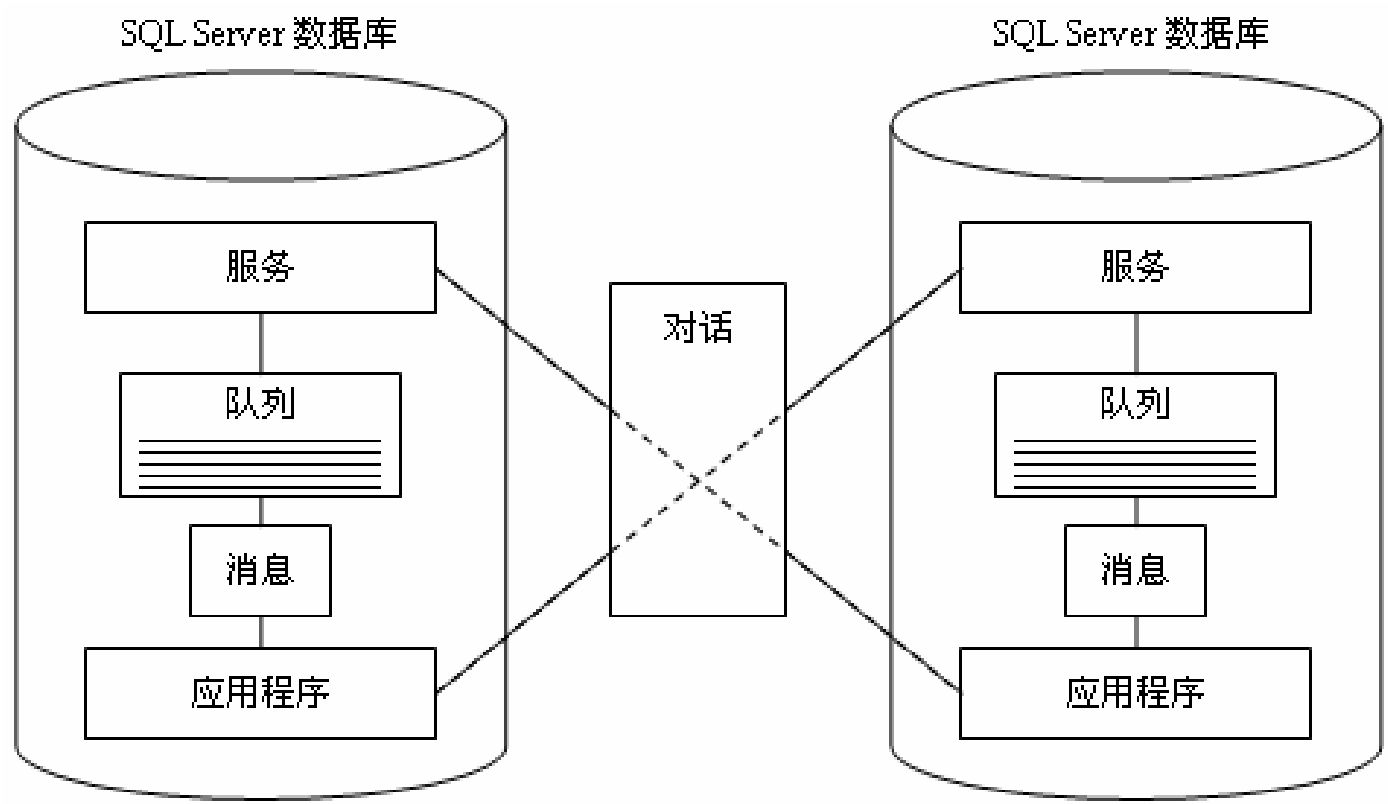
16.4 Service Broker应用程序示例

16.5 查看Service Broker信息

16.1 概述

- 在数据库系统中，同步通信方式和异步通信方式都是重要的，都是满足用户需求的重要方式。不同的场景下需要不同的通信方式。
- 在使用数据库的场景中，大多数情况下我们采用了同步通信方式。
- 如果采用异步通信机制，那么可以有效地解决通信效率低下的问题。

16.2 Service Broker体系架构



16.3 开发Service Broker应用程序的工具

- 下面，从3个不同的方面研究如何开发Service Broker应用程序。这3个方面分别是如何定义Service Broker对象，如何操纵Service Broker对象和如何启用Service Broker。
- 在Microsoft SQL Server 2005系统中，用户可以使用Transact-SQL语言中的DDL语句定义Service Broker对象，这些语句和特点如表16-1所示。

16.4 Service Broker应用程序示例

- 本节介绍一个简单的**Service Broker**应用程序示例。首先，讲述如何创建所需的**Service Broker**对象，然后分析如何使用这些对象。该示例是一个简单的异步通信系统，它将一个文本消息放置到输入队列中，然后再从队列中读取消息。

16.5 查看Service Broker信息

- Microsoft SQL Server 2005系统提供了多个系统视图，可以用来检索有关Service Broker对象及其当前状态的信息，这些系统视图如表16-3所示。

第17章 报表服务

教学目标

教学过程

教学目标

- 为什么要引入报表服务
- 报表服务体系结构的特点和主要组件的作用
- 报表服务器的结构特点和主要功能
- 报表服务支持6种呈现扩展类型
- 报表管理器的作用和使用方式
- 报表设计器的作用和使用方式
- 报表模型设计器的作用和使用方式
- 报表生成器的作用和使用方式

教学过程

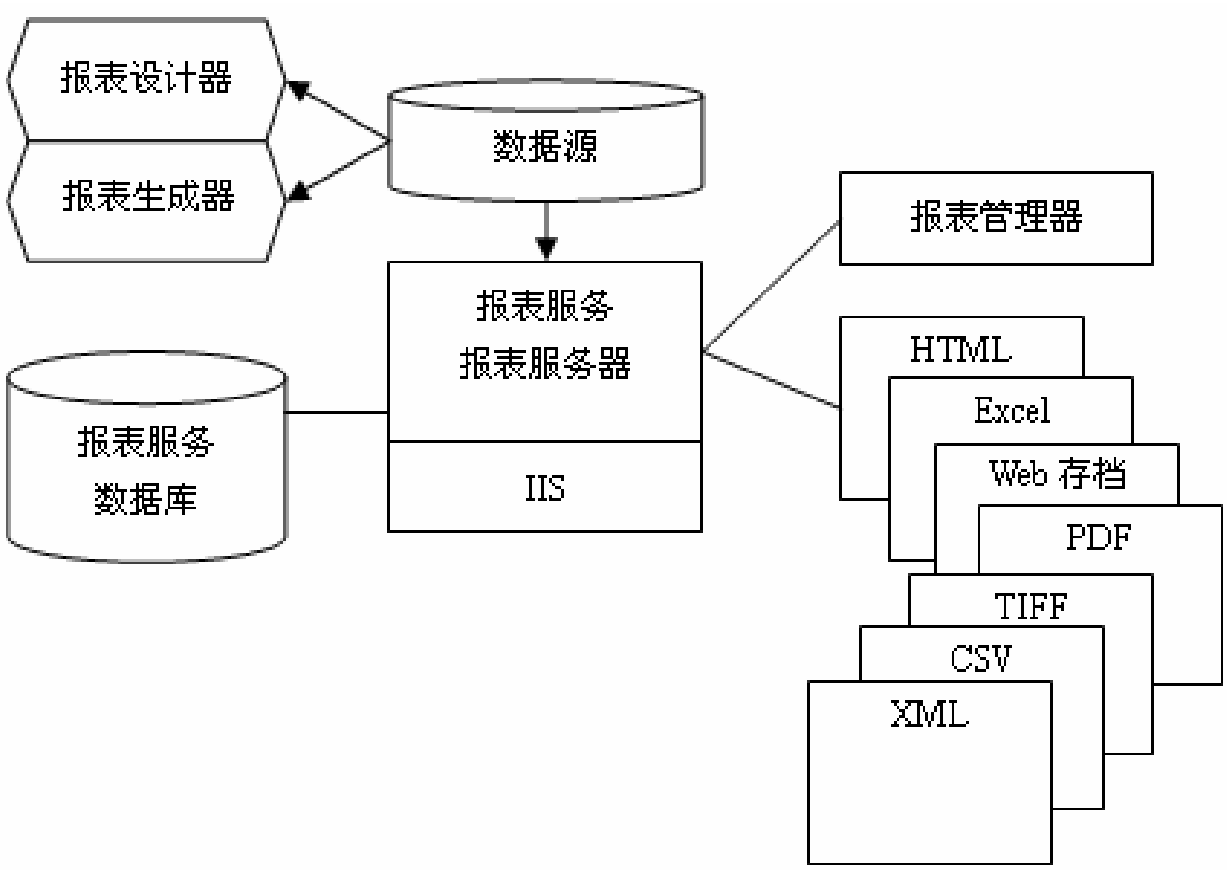
17.1 概述

- 报表对于管理人员来说是极为重要的，报表是管理人员了解组织状态、发现管理中的问题、安排管理工作、总结经验教训的重要手段。
- 虽然说数据库中存储了组织的大量业务信息，但是如何从大量的业务数据中及时发现潜在的问题，依然是摆在管理人员面前亟需解决的问题。管理人员不可能总是从数据库中查看数据，也不可能所有的管理人员都能随时查看各种业务数据，因此，作为业务状态载体的管理报表无疑是管理人员的管理手段和工具。

报表服务

- 虽然说报表是一个重要的管理职能，但是许多数据库产品却没有很好地重视这种职能。对于Microsoft SQL Server系统来说，在发布2000版本时系统依然不具备制作报表的功能。只是到了2003年，微软才发布了一个称为报表服务的组件。从2005版本开始，Microsoft SQL Server系统的报表服务在功能方面有了极大的提高。从这个变化中可以看出，微软的Microsoft SQL Server系统向管理领域深入发展的决心和趋势。

17.2 报表服务体系结构



17.3 报表服务器

- 报表服务器是报表服务的主要组件。报表服务器可以 **Microsoft Windows** 服务和 **Web** 服务两种形式实现，可以为处理和呈现报表提供优化的并行处理基础结构。**Web** 服务公开了一组客户端应用程序可用来访问报表服务器的编程接口。**Windows** 服务可提供初始化、计划和传递服务以及服务器维护功能。这些服务协同工作，构成单个报表服务器实例。
- 报表服务器的结构和功能如图**17-2**所示。

17.4 报表管理器

- 报表管理器是基于 **Web** 的报表访问和管理工具，可以通过浏览器进行访问。用户可以使用报表管理器通过 **HTTP** 连接从远程位置管理单个报表服务器实例，还可以使用报表管理器的报表查看器和导航功能。报表管理器的主窗口如图**17-3**所示。

17.5 报表编制工具

- 在Microsoft SQL Server 2005系统中，报表编制工具包括报表设计器、报表模型设计器和报表生成器。
- 下面分别讲述这些工具的特点和使用方式。

报表设计器

- 报表设计器是一种带有设计界面的图形化工具，可以用于预览和发布报表。报表设计器的环境提供了一些分栏的窗口可以支持用户交互式地设计报表，这些窗口包括数据窗格、布局窗格、报表单元、语言窗格等。在报表设计器中，既可以通过使用向导工具设计报表，也可以手工设计报表。
- 用户可以从**Business Intelligence Development Studio**工具中启动报表设计器。

报表模型设计器

- 在如图**17-4**所示的“新建项目”对话框中，从“模板”列表框中选择“报表模型项目”，可以启动报表模型设计器。报表模型中要做的第一件事情是向该项目中添加数据源，接下来是添加数据源视图。可以在数据源视图中选择包含的表和视图。可以在设计界面中添加或删除表、添加关系、创建指定的查询、替换表、浏览表数据等。
- 报表模型设计器的一个窗口如图**17-11**所示。

报表生成器

- 一旦创建了报表模型并将其发布到服务器之后，报表生成器就可以用于设计并运行一个基于该报表模型的报表。报表生成器可以用于创建表、矩阵或图表报表，使用报表布局模板，选择预先定义的报表模型。用户还可以向报表中添加文本和格式，创建新字段并对报表执行计算，预览、打印完整的报表等。

第18章 集成服务

教学目标

教学过程

教学目标

- 为什么要引入集成服务
- 异构数据的特征和面临的问题
- 数据仓库的特征和面临的问题
- **SSIS**体系架构的特点
- **DTP**的架构特点和作用
- **DTR**的架构特点和作用
- 使用**SSIS**导入/导出向导管理包
- 使用**SSIS**设计器管理包
- 配置和部署**SSIS**包

教学过程

18.1 概述

18.2 集成服务的体系架构

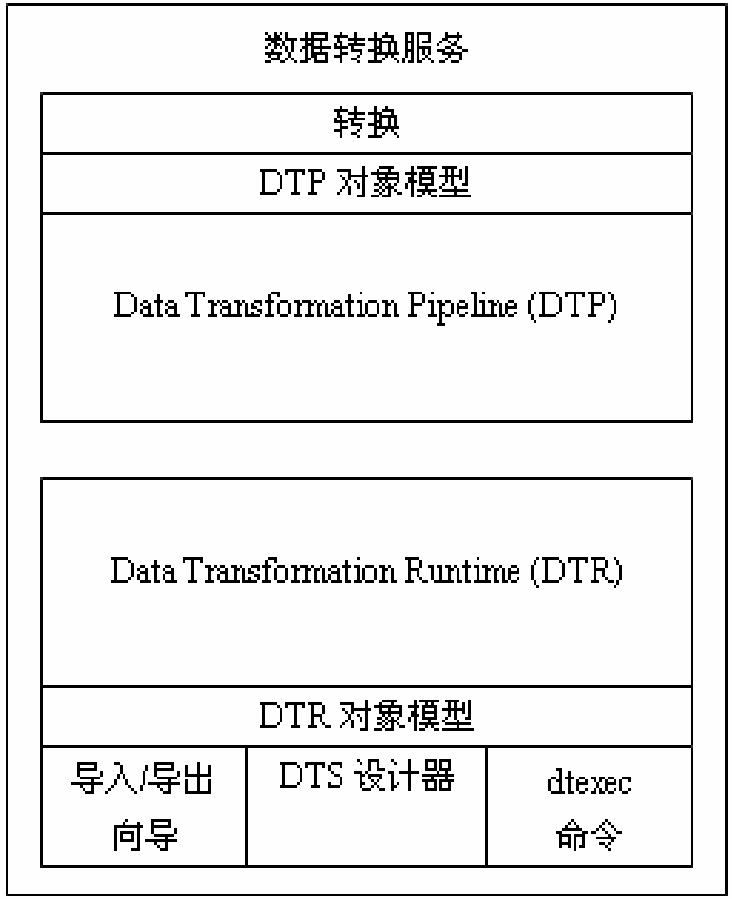
18.3 创建包

18.4 部署包

18.1 概述

- 在数据库的实际应用中，我们经常面临两大问题，一是如何有效地解决异构数据问题，二是如何有效地创建数据仓库和向数据仓库加载数据。
- **Microsoft SQL Server 2005**系统提供的集成服务**(SQL Server Integration Services，SSIS)**可以比较好地解决异构数据问题和数据仓库加载问题。
- 早在**Microsoft SQL Server 7.0/2000**系统中，微软提供了一个**DTS**服务，用于解决异构问题和加载数据问题。但是，在**2005**系统中，微软重写了集成服务，对原有的**DTS**进行了巨大的改变，目的是使其成为企业级的**ETL**平台**(extract, transformation, and loading**，抽取、转换和加载**)**。

18.2 集成服务的体系架构



18.3 创建包

- **Microsoft SQL Server 2005**系统提供了**3**种创建包的方法，即**SSIS**导入**/**导出向导、**SSIS**设计器和**DTR**对象模型。
- 下面，介绍前两种方法的特点。

使用SSIS导入/导出向导

- SSIS导入/导出向导提供了一系列对话框，可以帮助用户完成选择数据源、目标和传输对象的过程。该向导允许用户有选择地保存和执行SSIS包。
- 可以使用两种方式启动SSIS导入/导出向导。一种方式是在SQL Server Management Studio工具中，通过选择SQL Server实例引擎、数据库，从右击弹出的菜单中选择“任务”→“导入数据”或“导出数据”，可以启动SSIS导入/导出向导。另外一种方式，在命令行中输入dtswizard命令，也可以启动SSIS导入/导出向导。

使用SSIS设计器

- 虽然SSIS导入/导出向导可以方便地传输数据和创建包，但是对于ETL操作来说，这种传输方式比较简单，很难满足复杂的应用场景。因为ETL作业不仅仅是简单的从一个目标传输到另外一个目标，而是需要组合来自多个数据源的数据、对这些数据进行处理、将这些数据映射到新的列中，并且提供各种不同的数据清洗和验证作业。
- SSIS设计器可以比较好地完成这种复杂的ETL作业。作为一种图形化的工具，SSIS设计器可以用于构建、执行和调试SSIS包。

18.4 部署包

- 部署包有两个基本步骤，即生成集成服务项目以创建包部署实用工具，并且将在生成集成服务项目时所创建的部署文件夹复制到目标计算机，然后运行包安装向导来安装这些包。
- SSIS支持通过使用包配置来对包进行部署。

第19章 分析服务

教学目标

教学过程

教学目标

- 为什么要引入分析服务
- **OLTP**系统和数据仓库系统的区别和联系
- 数据仓库和数据集市之间的关系
- 雪花维度模型的特点
- **MOLAP/ROLAP/HOLAP**存储结构的特点
- 聚合数据的特点和方法
- 数据访问的特点和方法
- 使用**BIDS**工具执行分析服务操作

教学过程

19.1 概述

19.2 数据仓库的基本概念

19.3 Business Intelligence Development Studio

19.1 概述

- 数据仓库和以数据仓库为基础的分析系统，无疑是数据库技术发展的更高阶段。因此，许多数据库厂商纷纷推出自己的数据仓库和分析系统产品。微软公司也不例外。**Microsoft SQL Server 2005**系统包含了功能强大、技术先进的分析服务，它可以帮助用户构建数据仓库和分析数据仓库中的数据。

19.2 数据仓库的基本概念

- 下面，着重讨论数据仓库和分析服务领域中的一些基本概念，这些概念包括**OLTP**和数据仓库系统、数据仓库和数据集市、数据仓库设计和维度模型、多维数据集和存储模型、数据聚合、数据访问等内容。

OLTP和数据仓库系统

- OLTP是在线事务处理(online transaction processing)的简称。OLTP主要用在各种事务处理领域，例如会计核算、商品销售等都是典型的事务处理事件。普通的数据库系统就是一种OLTP系统。OLTP注重对业务数据记录的支持。
- 数据仓库把企业中的所有数据集中到一起存储，用户可以运行各种查询语句和报表来使用数据仓库中的数据。也就是说，数据仓库可以作为用户进行分析和决策的基础。

数据仓库和数据集市

- 数据仓库可以是包含了企业所有数据的数据库，可以由用户按照统一的方式进行访问。在经常情况下，企业可能拥有产生在不同时间的、存储在不同数据库或文件的、由不同的数据库管理系统管理的大量数据。这些数据库管理系统可能是关系型的，但是也可能是层次数据库系统或网状数据库系统。
- 数据集市只是包含企业部门级的数据，并且只有一部分用户使用。

数据仓库设计和维度模型

- 合理的数据库设计可以大大提高数据库的性能，同样，合理的数据仓库设计也可以大大提高数据仓库的性能。在数据库设计时，一般使用**ER**模型。在数据仓库设计时，需要使用维度模型。也就是说，维度模型是数据仓库的结构基础。
- 在数据仓库中，每一个维度模型都有一个包含了度量数据的表和若干个描述维度的表。前者被称为事实表，后者被称为维度表。

多维数据集和存储模型

- 数据仓库支持多种不同类型的存储结构。许多数据存储类型是基于被称为多维数据集的多维数据库。多维数据集是数据仓库数据的子集，可以组织成多维结构。在定义多维数据集时，需要选择一个事实表和确认该表中感兴趣的数值列，然后选择可以为数据提供描述性信息的维度表。
- 有三种典型的存储模型，即**MOLAP(multidimensional OLAP，多维OLAP)、ROLAP(relational OLAP，关系型OLAP)和HOLAP(Hybrid OLAP，混合OLAP)。**

数据聚合

- 数据是按照最详细的格式存储在事实表中，各种报表可以充分利用这些数据。一般的查询语句在查询事实表时，一次操作经常涉及成千上万条记录，但是通过使用汇总、平均、极值等聚合技术可以大大降低数据的查询数量。因此，来自事实表中的底层数据应该事先经过聚合存储在中间表中。
- 因为这种中间表存储了聚合信息，所以被称为聚合表，这种处理过程被称为聚合过程。

数据访问

- 一般地，用户使用三种访问技术访问数据仓库中的数据。这三种数据访问技术分别是报表、多维分析和数据挖掘。
- 报表是最简单的访问技术。报表是使用查询语句得到的表格数据或矩阵数据。这是最常见的数据访问方式。
- 多维分析是指可以利用数据仓库进行多个维度的计算、比较和分析，可以交互式地考虑所有的可能情况。
- 数据挖掘是通过对大量数据的研究和分析，企图发现更多的以前未知的信息和模式。管理人员可以使用这些信息和模式进行管理上的决策。

19.3 Business Intelligence Development Studio

- 分析服务的主要组件是**Business Intelligence Development Studio(BIDS)**。**BIDS**是一个管理工具，为集成服务、报表服务、分析服务、数据挖掘等提供了一个集成平台。基于**Visual Studio 2005**的**BIDS**支持用户开发商业智能应用程序，用户可以在该平台中进行编写代码、调试、版本控制等工作。
- **BIDS**工具具有易用性的特点，提供了许多向导工具帮助用户完成诸如创建数据源、创建数据源视图、创建多维数据集等工作。
- 从数据仓库存储结构角度来看，**BIDS**工具支持用户根据需要创建**MOLAP**、**ROLAP**和**HOLAP**等不同的结构，增强了系统的灵活性。

9种数据挖掘技术或算法

- Microsoft Naïve Bayes是一种分类算法，用于预测性建模。
- Microsoft关联规则算法是基于关联模型的，对建议引擎非常有用。
- Microsoft聚类分析算法使用迭代技术将数据集中的事例分组为包含类似特征的分类。在浏览数据、标识数据中的异常及创建预测时，这些分组十分有用。
- Microsoft决策树算法是分类和回归算法，用于对离散和连续属性进行预测性建模。
- Microsoft神经网络算法通过构造多层感知器网络创建分类和回归挖掘模型。
- Microsoft逻辑回归算法是Microsoft神经网络算法的变化形式。
- Microsoft顺序分析和聚类分析算法研究包含可通过指定路径或顺序链接到的事件的数据。
- Microsoft时序算法用于创建数据挖掘模型以预测连续列。
- Microsoft线性回归算法是Microsoft决策树算法的变体，该算法不创建拆分，从而执行线性回归。

第20章 XML

教学目标

教学过程

教学目标

- 为什么要使用XML语言
- 类型化数据和非类型化数据的特点
- XML数据类型的特点
- XML架构的作用和使用方式
- XQuery/exist/modify技术的特点
- FOR XML子句的类型和使用方式
- XML索引的类型和特点
- OPENXML函数的作用和使用方式

教学过程

20.1 概述

- 从Microsoft SQL Server 2000版本开始，微软公司引入了XML技术。这时，可以在系统中通过FOR XML子句和OpenXML函数使用XML数据。
- Microsoft SQL Server 2005系统通过引入更多的功能增强了对XML数据的支持。Microsoft SQL Server系统提供了XML数据类型可以用来存储XML数据。XQuery和XSD(eXtensible schema definition，可扩展的架构定义)支持这种XML数据。并且这种XML数据与Microsoft SQL Server 2005关系型数据库引擎紧密集成的。例如，Microsoft SQL Server 2005提供了XML触发器、XML数据复制、大容量的XML数据插入等操作的支持。

20.2 XML数据类型

- XML数据类型是Microsoft SQL Server 2005系统为了增强XML技术支持而引入的新功能。就像INT、CHAR等数据类型一样，XML数据类型可以用在表中列的定义中、变量的定义中和存储过程的参数定义中。
- XML数据类型既可以存储类型化数据，也可以存储非类型化数据。如果存储在XML列中的数据没有与XSD架构关联，那么这种数据就是非类型化数据。如果存储在XML列中的数据与XSD架构关联，那么这种数据是类型化数据。当插入类型数据时，系统将根据定义的XSD架构检查数据的一致性和完整性。

20.3 查询XML数据

- 对于XML列中的数据，可以使用相应的技术对其进行操纵，这些技术包括XQuery技术、Exist技术和Modify技术。
- 下面介绍这些技术。

使用XQuery技术

- XQuery是一种可以查询结构化或半结构化的XML数据的语言。由于Microsoft SQL Server 2005系统提供了对XML数据类型的支持，因此可以将XML文档存储在数据库中，然后使用XQuery语句进行查询。
- XQuery基于现有的XPath查询语言，并且支持迭代、排序结果以及构造必须的XML的功能。
- Transact-SQL支持XQuery语言的子集。

使用XML数据类型方法

- Microsoft SQL Server 2005系统提供了一些内置的可以用于XML数据类型的方法。与普通关系型数据不同的是，XML数据是分层次的，具有完整的结构和元数据。XML数据类型方法可以用于钻取存储在XML数据类型中的XML文档的内容。
- 这些方法包括Exist方法、Modify方法、Query方法、Value方法等。
- Query方法在前面一节中已经涉及了，下面主要讲述Exist方法和Modify方法。

20.4 使用FOR XML子句

- 使用FOR XML可以把Microsoft SQL Server 2005系统的表中数据检索出来并且自动表示成XML的格式。在Microsoft SQL Server 2000版本中，FOR XML有三种模式，即RAW、AUTO和EXPLICIT。在Microsoft SQL Server 2005系统中，由于增加了XML数据类型，因此也增强了FOR XML的功能，这些增强功能TYPE模式、PATH模式、嵌套FOR XML查询、内联XSD架构等。
- 下面详细研究这些内容。

FOR XML RAW

- FOR XML RAW是最简单的FOR XML模式，该模式将查询结果集中的每一行转换为带有通用标识符<row>或可能提供元素名称的XML元素。在默认情况下，行集中非NULL的每列值都将映射为<row>元素的一个属性。也就是说，RAW模式表示元素名称是row，属性名称是列名称或列的别名。

FOR XML AUTO

- 使用**FOR XML AUTO**也可以返回**XML**文档。但是，使用**AUTO**关键字和使用**RAW**关键字得到的**XML**文档形式是不同的。使用**AUTO**关键字，**Microsoft SQL Server**使用表名称作为元素名称，使用列名称作为属性名。**SELECT**关键字后面的列的顺序用于确定**XML**文档的层次。

使用FOR XML EXPLICIT

- 使用**FOR XML EXPLICIT**子句可以准确地得到用户需要的**XML**文档。但是，**FOR XML EXPLICIT**子句和前面讲过的两个子句不同。前面讲过的那些子句可以直接用在**SELECT**子句中，但是如果把**FOR XML EXPLICIT**子句直接用在**SELECT**子句中，就会产生如图**20-19**所示的错误信息。

使用TYPE指令

- 由于SQL Server 2005系统支持XML数据类型，因此可以通过指定TYPE指令，将FOR XML查询结果返回为XML数据类型。这样可以方便在服务器上处理FOR XML的查询结果。

使用FOR XML PATH

- 作为一种新增功能，**FOR XML PATH**子句比**FOR XML ROW**或**FOR XML AUTO**子句的功能强大，并且比**FOR XML EXCPLICIT**子句更加简单。**FOR XML PATH**子句允许用户指定**XML**树状数据中的路径。**FOR XML PATH**子句可以更加简单地完成**FOR XML EXCPLICIT**子句具备的功能。

嵌套的FOR XML查询

- Microsoft SQL Server 2000系统限定FOR XML子句只能用在查询语句的顶层，不能在子查询中使用FOR XML子句。但是，Microsoft SQL Server 2005系统增强了这方面的功能，用户可以在子查询中使用FOR XML子句，从而实现了嵌套的FOR XML查询。

内联XSD架构生成

- 在FOR XML子句中，可以请求在查询返回结果的同时返回一个内联架构。如果需要XSD架构，可以使用XMLSCHEMA关键字。需要注意的时，只能在RAW和AUTO模式中指定XMLSCHEMA，不能在EXPLICIT模式和PATH模式中指定内联XSD架构。

20.5 使用XML索引

- XML数据类型支持最大2GB的数据，这是一个相当大的数据。当查询XML数据时，XML数据会对系统性能带来巨大的影响。为了提高XML查询的性能，可以在具有XML数据类型的列上创建索引。
- XML索引可以分为两个类别，即主XML索引和辅助XML索引。

20.6 使用OPENXML函数

- 前面已经讲过，使用**FOR XML**可以把**Microsoft SQL Server**系统中的数据生成**XML**文档，使用**OPENXML**则是使用**FOR XML**的逆过程。也就是说，使用**OPENXML**可以从**XML**文当中返回数据的行集。

第21章 开发CLR数据库对象

教学目标

教学过程

教学目标

- .NET CLR与Microsoft SQL Server的集成的意义
- CLR体系结构的特点
- 启动Microsoft SQL Server对CLR的支持
- 创建CLR数据库对象的一般过程
- 使用Visual Studio 2005创建CLR触发器
- 使用CREATE ASSEMBLY语句创建程序集

教学过程

21.1 概述

21.2 CLR体系结构

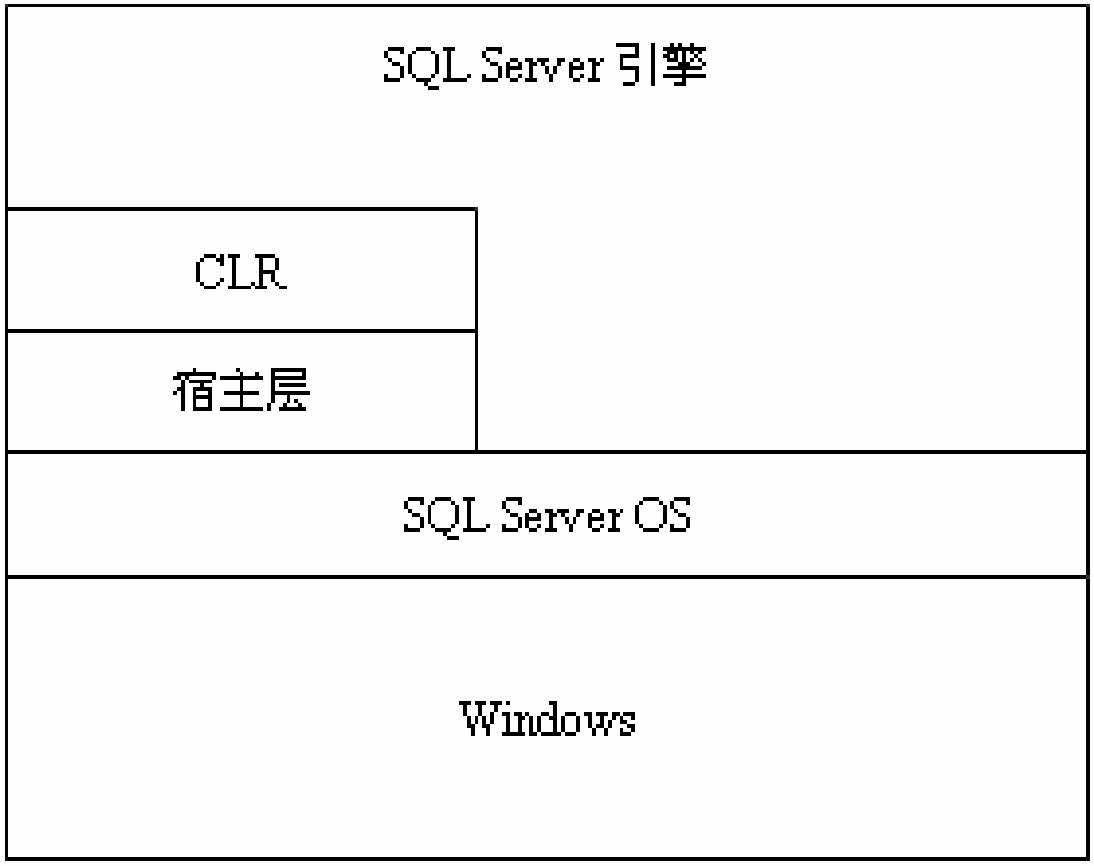
21.3 CLR数据库对象创建过程

21.4 创建CLR触发器数据库对象

21.1 概述

- .NET CLR与Microsoft SQL Server 2005的集成支持使用任何.NET语言开发存储过程、用户定义函数、触发器、聚集、用户自定义类型等。这种集成不尽仅是表面的工作，实际上，Microsoft SQL Server 2005的数据库引擎内置了CLR。
- 要创建CLR数据库对象，首先必须使用Visual Studio 2005创建一个DLL。然后，将该DLL作为程序集导入Microsoft SQL Server系统中。最后，将该程序集链接到诸如存储过程、触发器等数据库对象上。

21.2 CLR体系结构



21.3 CLR数据库对象创建过程

- 要创建.NET CLR数据库对象，首先必须使用任何一种.NET语言编写托管代码，并将其编译到动态链接库(dynamic link library，DLL)中。实现该过程的最常见的方法是使用Visual Studio 2005创建一个新的SQL Server项目，然后构建该项目，即创建DLL文件。
- 一旦创建了.NET DLL文件，则需要向SQL Server注册该DLL文件，即使用CREATEASSEMBLY语句创建一个虚拟的名称为程序集的SQL Server数据库对象。从本质上来看，该程序集封装了.NET DLL。
- 然后，在Microsoft SQL Server中使用DDL语言创建一个新的数据库对象，例如触发器或存储过程，使其指向新建的程序集。

21.4 创建CLR触发器数据库对象

- 下面，以创建CLR触发器数据库对象的完整过程为例，讲述如何使用Visual Studio 2005工具创建CLR数据库对象。