第5章 SQL Server 2008数据库基础

Microsoft SQL Server 2008 是一个数据库平台，能够满足各种企事业单位对构建网络数据库的需求，同时还具有功能强大、安全可靠等特点，可用于大型联机事务处理、数据仓库及电子商务等。本章首先对 SQL Server 2008 进行简单概述，然后详细介绍 SQL Server 2008 数据库、数据表的管理，以及如何在数据库中创建及操作视图和索引，同时对 SQL Server 2008 的其他功能进行简要介绍。

本章学习重点：

(1) SQL Server 2008 的基本知识；

(2) SQL Server 2008 的数据库管理；

(3) SQL Server 2008 的数据表管理；

(4) SQL Server 2008 数据表的基本操作；

(5) SQL Server 2008 的视图和索引操作；

(6) SQL Server 2008 的存储过程、触发器、备份与恢复、安全管理以及数据格式转换等其他功能。

5.1 SQL Server 2008 概述

5.1.1 SQL Server 2008 简介

SQL Server 2008 是 Microsoft 公司于 2008 年推出的一个可信任的、高效的、智能的数据平台。它提供了可依靠的技术和能力来接受不断发展的对于管理数据和给用户发送全面洞察的挑战，具有在关键领域方面的显著优势。它是微软数据平台愿景中的一个主要部分，旨在满足目前和将来管理和使用数据的需求。

SQL Server 2008 除了具有 SQL Server 以前版本所具有的特点，例如，真正的客户机/服务器体系结构、图形化用户界面、丰富的编程接口工具、与 Windows NT 的完全集成、良好的伸缩性、对 Web 技术的支持以及提供数据仓库功能等，还推出了许多新特性和关键改进。

1) 增加了集成服务

SSIS(SQL Server 集成服务)是一个嵌入式应用程序，用于开发和执行 ETL(解压缩、转换和加载)包。SSIS 代替了 SQL Server 2000 的 DTS。整合服务功能既包含了实现简单的

导入/导出包所必需的 Wizard 导向插件、工具以及任务，也有非常复杂的数据清理功能。SQL Server 2008 SSIS 的功能有很大改进和增强，比如它的执行程序能够更好地并行执行。在 SSIS 2005，数据管道不能跨越两个处理器。而 SSIS 2008 能够在多处理器机器上跨越两个处理器 。而且它在处理大件包上面的性能得到了提高。SSIS 引擎更加稳定，锁死率更低。

2）改进了 Lookup 功能

Lookup 是 SSIS 一个常用的获取相关信息的功能。比如从 CustomerID 查找 Customer Name，获取数据集。Lookup 在 SSIS 中很常见，可以处理上百万行的数据集，因此性能可能很差。SQL 2008 对 Lookup 的性能做出了很大改进，而且能够处理不同的数据源，包括 ADO. NET、XML、OLEDB 和其他 SSIS 压缩包。

3）改进了报表服务

SSRS（SQL Server 报表服务）的处理能力和性能得到改进，使得大型报表不再耗费所有可用内存。另外，在报表的设计和完成之间有了更好的一致性。SQL SSRS 2008 还包含了跨越表格和矩阵的 TABLIX。Application Embedding 允许用户单击报表中的 URL 链接调用应用程序。

4）改进了数据库镜像

（1）页面自动修复：SQL Server 2008 通过请求获得一个从镜像合作机器上得到的出错页面的重新复制，使主要的和镜像的计算机可以透明地修复数据页面上的 823 和 824 错误。

（2）提高了性能：SQL Server 2008 压缩了输出的日志流，以便使数据库镜像所要求的网络带宽达到最小。

5）加强了可支持性

（1）SQL Server 2008 包括了新增加的执行计数器，可以更细粒度地对 DBMS 日志记录的不同阶段所耗费的时间进行计时。

（2）SQL Server 2008 包括动态管理视图（Dynamic Management View）和对现有视图的扩展，以此来显示镜像会话的更多信息。

6）支持 Microsoft Office 渲染

SQL Server 2008 提供了新的 Microsoft Office 渲染，使得用户可以从 Word 里直接访问报表。此外，现有的 Excel&reg 渲染器被极大地增强了，用以支持嵌套数据区域、子报表和合并单元格等功能。这使得用户可以维护显示保真度和改进 Microsoft Office 应用中所创建的报表的全面可用性。

7）Microsoft SharePoint&reg 集成

SQL Server 2008 报表服务将 Microsoft Office SharePoint Server 2007 和 Microsoft SharePoint Services 深度集成，提供了企业报表和其他商业洞察的集中发送和管理，使得用户可以访问包含了与他们直接在商业门户中所做的决策相关的结构化和非结构化信息的报表。

8）支持地理数据类型

这个功能支持存储符合行业空间标准（开放地理空间联盟（Open Geospatial Consortium，OGC））的平面空间数据。这使得开发人员可以通过存储与设计平面表面和自然平面数据（例如内部空间等）相关联的多边形、点和线来实现“平面地球”。

9）支持几何数据类型

此功能支持存储地理空间数据并能对其进行相关操作。使用纬度和经度的组合以及地理数据和行业标准椭圆体（例如用于全球 GPS 解决方案的 WGS84）来定义地球表面的区域。

5.1.2 SQL Server 2008 常用工具

Microsoft SQL Server 2008 系统提供了大量的管理工具，实现了对系统进行快速、高效的管理。这些管理工具主要包括 Microsoft SQL Server Management Studio、Business Intelligence Development Studio、SQL Server Configuration Manager、SQL Server Profiller、Database Engine Tuning Advisor 以及大量的命令行和实用工具。其中，最重要的工具是 Microsoft SQL Server Management Studio。本节将介绍这些工具的主要作用和特点。

1. Microsoft SQL Server Management Studio

Microsoft SQL Server Management Studio 是 Microsoft SQL Server 2008 提供的一种新的集成环境。Microsoft SQL Server 2008 将服务器管理和业务对象创建合并到以下两种集成环境中：Microsoft SQL Server Management Studio 和 Business Intelligence Development Studio。这两种环境使用解决方案和项目进行管理和组织，同时还提供了完全的源代码管理功能，能够与 Visual Studio 2008 集成。

Microsoft SQL Server Management Studio 是一个集成环境，用于访问、配置、控制、管理和开发 SQL Server 的所有工作。实际上，Microsoft SQL Server Management Studio 组合了大量的图形工具和丰富的脚本编辑器，大大方便了技术人员和数据库管理员对 SQL Server 系统的各种访问，是 SQL Server 2008 最重要的管理工具组件。Microsoft SQL Server Management Studio 将 SQL Server 2000 的查询分析器和服务管理器的各种功能组合到一个单一的环境中，此外还提供一个环境统一管理分析服务（Analysis Service）、集成服务、报表服务和 XQuery。此环境为开发者提供了一个熟悉的体验环境，为数据库管理人员提供了一个实用工具，使用户能够通过简单的图形工具和丰富的脚本完成任务。

SQL Server 管理平台不仅能够配置系统环境和管理 SQL Server，而且由于它能够以层叠列表的形式来显示所有的 SQL Server 对象，因而，所有 SQL Server 对象的建立与管理工作都可以通过它来完成。通过 SQL Server Management Studio 可以完成的操作有：管理 SQL Server 服务器，建立和管理数据库，建立与管理表、视图、存储过程、触发程序、角色、规则、默认值等数据库对象以及用户定义的数据类型，备份数据库和事务日志，恢复数据库，复制数据库，设置任务调度，设置报警，提供跨服务器的拖放控制操作，管理用户账户，建立 Transact-SQL 命令语句等。

要打开 Microsoft SQL Server 2008 的 SQL Server Management Studio，可以通过“开始”菜单，选择 Microsoft SQL Server 2008 程序组中的 SQL Server Management Studio。

要使用 SQL Server Management Studio，首先必须在对话框中注册。在“服务器类型”、“服务器名称”、“身份验证”选项中分别输入或选择正确的信息（默认情况下不用选择，因为安装的时候已经设置完毕），连接服务器界面如图 5-1 所示，然后单击“连接”按钮，即可登录

到 SQL Server Management Studio。

图 5-1 连接服务器界面

Microsoft SQL Server Management Studio 由多个管理和开发工具组成，主要包括已注册的服务器、对象资源管理器、查询编辑器、模板资源管理器、解决方案资源管理器等。主界面如图 5-2 所示。

图 5-2 Microsoft SQL Server Management Studio 主界面

“已注册的服务器”，可以完成注册服务器和将服务器组合成逻辑组的功能。通过该窗口可以选择“数据库引擎服务器”、“分析服务器”、“报表服务器”、“集成服务器”等。当选中某个服务器时，可以从右键快捷菜单中选择执行查看服务器属性、启动和停止服务器、新建服务器组、导入导出服务器信息等操作。

“对象资源管理器”可以完成类似 SQL Server Enterprise Manager 工具的操作。具体地，“对象资源管理器”可以完成如下操作：

(1) 注册服务器；

(2) 启动和停止服务器;

(3) 配置服务器属性;

(4) 创建数据库,以及创建表、视图、存储过程等数据库对象;

(5) 生成 Transact-SQL 对象,创建脚本;

(6) 创建登录账户;

(7) 管理数据库对象权限;

(8) 配置和管理复制;

(9) 监视服务器活动、查看系统日志等。

"查询编辑器"是以前版本中的 Query Analyzer 工具的替代物,用于编写和运行 Transact-SQL 脚本。与 Query Analyzer 工具总是工作在连接模式下不同的是,"查询编辑器"既可以工作在连接模式下,也可以工作在断开模式下。另外,如同 Visual Studio 工具一样,"查询编辑器"支持彩色代码关键字、可视化地显示语法错误、允许开发人员运行和诊断代码等功能。因此,"查询编辑器"的集成性和灵活性大大提高了。

"模板资源管理器"提供了执行常用操作的模板,其主界面如图 5-3 所示。用户可以在此模板的基础上编写符合自己要求的脚本。例如在"模板资源管理器"窗口中打开 Database 节点,可以生成诸如 attach database、Bring Database Online、Create Database on Multiple Filegroups 等操作的模板。

图 5-3 模板资源管理器界面

"解决方案资源管理器"提供指定解决方案的树状结构图。解决方案可以包含多个项目,允许同时打开、保存、关闭这些项目。解决方案中的每一个项目还可以包含多个不同的文件或其他项(项的类型取决于创建这些项所用到的脚本语言)。

2. Business Intelligence Development Studio

Business Intelligence Development Studio(SQL Server 2008 商业智能开发平台)是一个集成开发环境,用于开发商业智能构造,例如多维数据集、数据源、报告、Integration Services 软件包等,其主界面如图 5-4 所示。SQL Server 2008 商业智能开发平台包含了一些项目模板,这些模板可供开发特定构造的上下文。

利用商业智能开发平台开发项目时,可以将其作为某个解决方案的一部分进行开发,而

该解决方案独立于具体的服务器。例如，可以在同一个解决方案中包括 Analysis Services 项目和 Reporting Services 项目。在开发过程中，可以将对象部署到测试服务器中进行测试，然后将项目结果部署到一个或者多个临时服务器或生产服务器上。

图 5-4 Business Intelligence Development Studio 主界面

SQL Server 2008 商业智能开发平台可用于开发商业智能应用程序，例如开发并使用 Analysis Services、Integration Services 或 Reporting Services 的安防。如果要实现使用 SQL Server 数据库服务的解决方案，或者要管理并使用 SQL Server、Analysis Services、Integration Services 或 Reporting Services 的现有解决方案，则应当使用 SQL Server Management Studio。

3. SQL Server Configuration Manager

SQL Server Configuration Manager(SQL Server 配置管理器)用于管理与 SQL Server 相关的服务，SQL Server 实用的网络协议以及 SQL Server 客户端计算机管理网络连接的配置，可以通过“开始”菜单启动 SQL Server Configuration Manager 来实现，如图 5-5 所示。

4. SQL Server Profiler

SQL Server Profiler(SQL Server 分析器)是一个图形化的管理工具，用于监督、记录和检查 SQL Server 2008 数据库的使用情况。对于系统管理员来说，它是一个连续、实时地捕捉用户活动情况的间谍。

可以通过多种方法来启动 SQL Server Profiler，以支持在各种情况下收集跟踪输出。例如，可以通过“开始”菜单启动 SQL Server Profiler。SQL Server Profiler 启动以后，选择“文件”→“新建跟踪”命令，打开如图 5-6 所示的跟踪属性窗口。

在“常规”选项卡中，可以设置跟踪名称和跟踪提供程序名称与类型，使用的模板，保存的地址，以及是否启用跟踪停止时间等。

图 5-5 SQL Server Configuration Manager 主界面

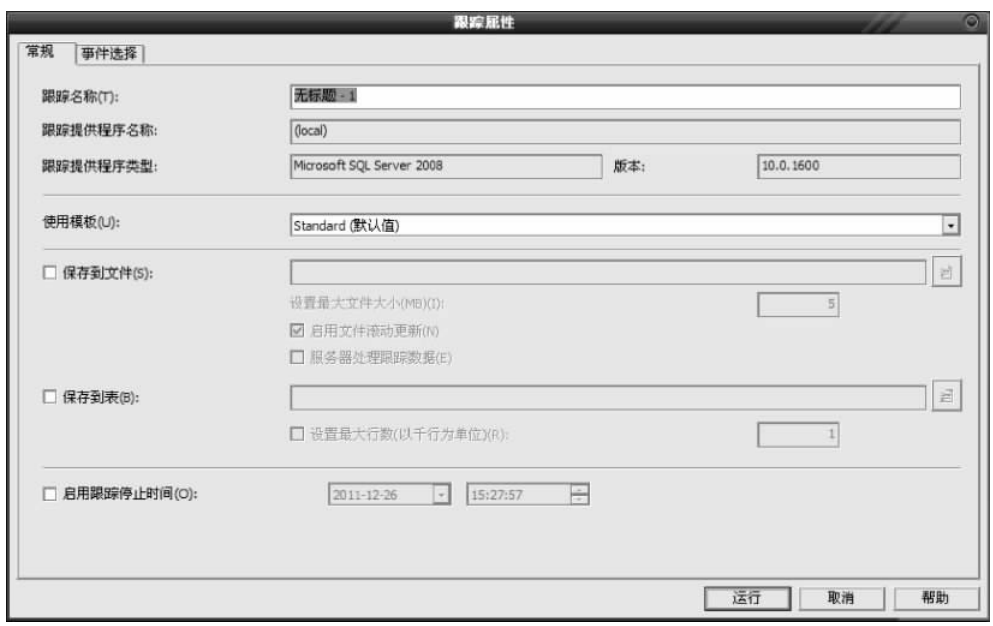


图 5-6 跟踪属性窗口

在“事件选择”选项卡中,可以设置需要跟踪的事件和事件列,如图 5-7 所示。

SQL Server Profiler 是用于捕捉来自服务器的 SQL Server 2008 事件的工具,这些事件保存在一个跟踪文件中,可以在以后对该文件进行分析,也可以在试图诊断某个问题时,用它来重播一系列的步骤。SQL Server Profiler 可以支持如下活动:

(1) 逐步分析有问题的查询,以便找到问题的原因;

(2) 查找并诊断执行速度慢的查询;

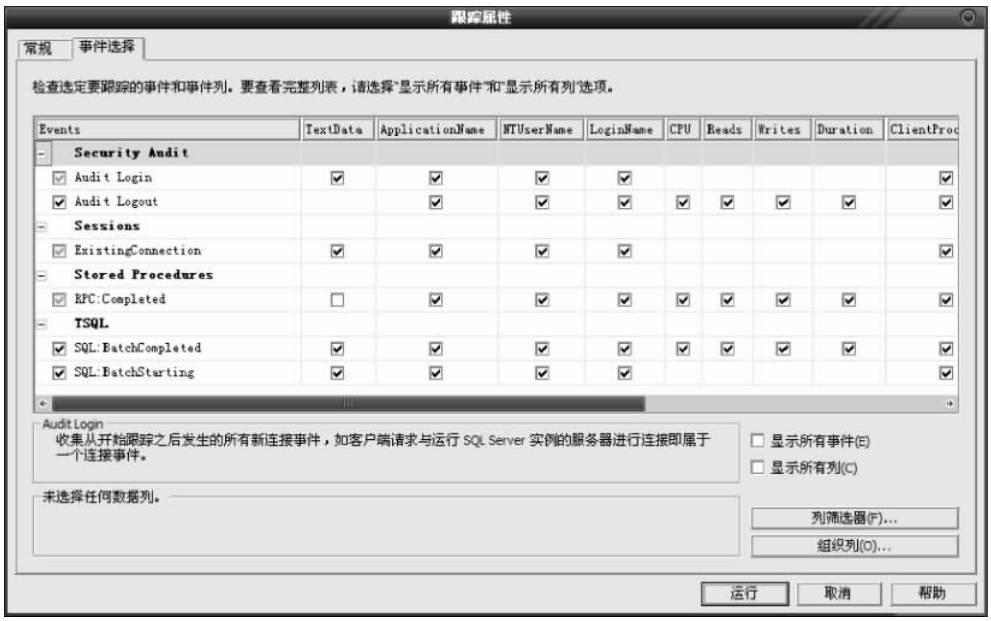


图 5-7 SQL Server Profiler"事件选择"选项卡

(3) 捕获导致某个问题的一系列 Transact-SQL 语句,然后利用所保存的跟踪,在某台测试服务器上复制此问题,接着,在该服务器上诊断问题;

(4) 监视 SQL Server 的性能以便优化工作负荷;

(5) 使性能计数器与诊断关联;

(6) SQL Server Profiler 还支持对 SQL Server 实例上执行的操作进行审核,审核将记录与安全相关的操作,方便安全管理员以后复查。

5. Database Engine Tuning Advisor

Database Engine Tuning Advisor(数据库引擎优化顾问)工具可以帮助用户分析工作负荷、提出创建高效率索引的建议等功能。

借助数据库引擎优化顾问,用户不必详细了解数据库的结构就可以选择和创建最佳的索引、索引视图、分区等。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,可以通过 Microsoft SQL Server Management Studio 中的查询编辑器创建 Transact-SQL 脚本工作负荷,也可以使用 SQL Server Profiler 中的优化模板来创建跟踪文件和跟踪表工作负荷。数据库引擎优化顾问(Database Engine Tuning Advisor)的界面如图 5-8 所示。

使用数据库引擎优化顾问工具可以执行下列操作:

(1) 通过使用查询优化器分析工作负荷中的查询,推荐数据库的最佳索引组合;

(2) 为工作负荷中引用的数据库推荐对齐分区和非对齐分区;

(3) 推荐工作负荷中引用的数据库的索引视图;

(4) 分析所建议的更改将会产生的影响,包括索引的使用、查询在工作负荷中的性能;

(5) 推荐为执行一个小型的问题查询集而对数据库进行优化的方法;

(6) 允许通过指定磁盘空间约束等选项对推荐进行自定义；

(7) 提供对所给工作负荷的建议执行效果的汇总报告。

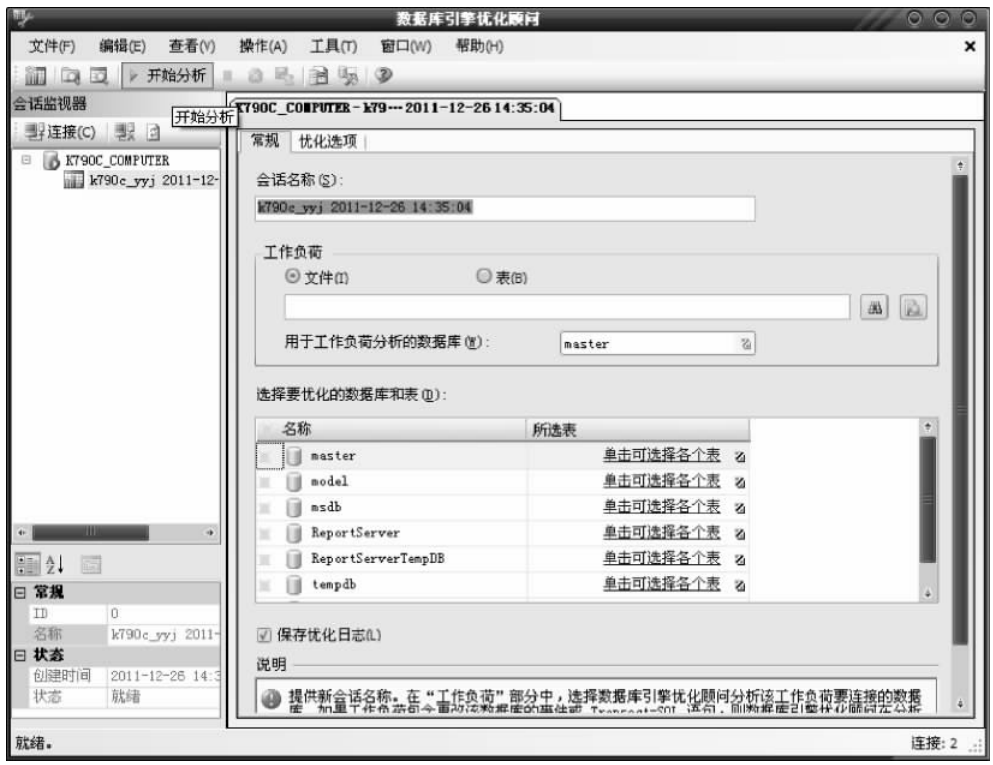


图 5-8 “数据库引擎优化顾问”主界面

6. 实用工具

在 Microsoft SQL Server 2008 系统中，不仅提供了大量的图形化工具，还提供了大量的命令行实用工具。这些命令行实用工具包括 bcp、dta、dtexec、dtutil、Microsoft. AnalysisServices. Deployment、nscontrol、osql、profiler90、rs、rsconfig、rskeymgmt、sac、sqlagent90、sqlcmd、SQLdiag、sqlmaint、sqlservr、sqlwb、tablediff 等。

(1) bcp 实用工具可以在 Microsoft SQL Server 2008 实例和用户指定格式的数据文件之间进行大容量的数据复制。也就是说，使用 bcp 实用工具可以将大量数据导入 Microsoft SQL Server 表中，或者将表中的数据导出到数据文件中。

(2) dta 实用工具是数据库引擎优化顾问的命令提示符板。通过使用 dta 实用工具，用户可以在应用程序和脚本中使用数据库引擎优化顾问功能，从而扩大了数据库引擎优化顾问的作用范围。

(3) dtexec 实用工具用于配置和执行 Microsoft SQL Server 2008 Integration Services (SSIS)包。用户通过使用 dtexec 实用工具，可以访问所有 SSIS 包的配置信息和执行功能，这些信息包括连接、属性、变量、日志、进度指示器等。

(4) dtutil 实用工具的作用类似 dtexec 实用工具，也是执行与 SSIS 包有关的操作。但

是,该工具主要用于管理 SSIS 包,这些管理操作包括验证包的存在性及对包进行复制、移动、删除等操作。

(5) Microsoft. AnalysisServices. Deployment 实用工具执行与 Microsoft SQL Server 2008 Analysis Services (SSAS)有关的部署操作。该工具的输入文件是在生成分析服务项目时生成的 XML 类型文件。这些文件可以提供对象定义、部署目标、部署选项、配置设置等。该工具通过使用指定的部署选项和配置设置,将对象定义部署到指定的部署目标。

(6) nscontrol 实用工具与 Microsoft SQL Server 2008 Notification Services 服务有关,用于管理、部署、配置、监视、控制通知服务实例,并且提供了创建、删除、使能、修复、注册等与通知服务实例相关的命令。

(7) osql 实用工具可以用来输入和执行 Transact-SQL 语句、系统过程、脚本文件等。该工具通过 ODBC 与服务器进行通信。实际上,在 Microsoft SQL Server 2008 系统中,sqlcmd 实用工具可以替代 osql 实用工具。

(8) profiler90 实用工具是启动 SQL Server Profiler 工具的命令行命令。使用该工具可以方便地在应用中对 SQL Server Profiler 工具进行启动和使用。

(9) rs 实用工具与 Microsoft SQL Server 2008 Reporting Services 服务有关,可以用于管理和运行报表服务器的脚本。通过使用该工具,用户可以轻松地实现报表服务器部署与管理任务的自动化执行。

(10) rsconfig 实用工具也是与报表服务相关的工具,可以用来对报表服务连接进行管理。例如,该工具可以在 RSReportServer. config 文件中加密并存储连接和账户,确保报表服务可以安全地运行。

(11) rskeymgmt 实用工具也是与报表服务相关的工具,可以用来提取、还原、创建、删除对称密钥。该密钥可用于保护敏感报表服务器数据不受未经授权的访问,从而提高报表服务器数据的安全性。

(12) sac 实用工具与 Microsoft SQL Server 2008 外围应用设置相关,可以用来导入、导出这些外围应用设置,大大方便了多台计算机上的外围应用设置。例如,可以使用 Microsoft SQL Server 2008 系统提供的外围应用配置图形工具先配置一台计算机,然后使用 sac 将该计算机的配置导出到一个文件中。接着,可以使用 sac 实用程序将所有 Microsoft SQL Server 2008 组件的设置应用到本地或远程计算机的其他 Microsoft SQL Server 2008 实例中。

(13) sqlagent90 实用工具用于从命令提示符处启动 SQL Server Agent 服务。需要注意的是,一般地,应该从 SQL Server Management Studio 工具中或在应用程序中使用 SQL-DMO 方法来运行 SQL Server Agent 服务。只有在对 SQL Server Agent 服务进行诊断或提供程序定向到命令提示符时,才使用该工具。

(14) sqlcmd 实用工具可以在命令提示符处输入 Transact-SQL 语句、系统过程和脚本文件。实际上,该工具是作为 osql 实用工具和 isql 实用工具的替代工具而新增的,它通过 OLE DB 与服务器进行通信。

(15) SQLdiag 用于对 SQL Server 系统进行诊断。利用该工具可以收集 SQL Server 系统的有关性能诊断信息,这些信息包括 Windows 性能日志、Windows 事件日志、SQL Server 事件探查器跟踪、SQL Server 阻塞和配置信息等。从而有助于技术支持人员排除

SQL Server 运行过程中出现的故障。

(16) sqlmaint 可以执行一组指定的数据库维护操作,包括 DBCC 检查、数据库备份、事务日志备份、更新统计信息、重建索引,并且生成报表以及把这些报表发送到指定的文件和电子邮件账户。

(17) sqlservr 的作用是在命令提示符下启动、停止、暂停、继续 Microsoft SQL Server 的实例。如果希望从应用程序中启动 Microsoft SQL Server 实例,则使用该工具将是一个不错的选择。

(18) sqlwb 可以在命令提示符下打开 SQL Server Management Studio,并且可以与服务器建立连接,打开查询、脚本、文件、项目、解决方案等。

(19) tablediff 用于比较两个表中的数据是否一致,对于排除复制中出现的故障非常有用。用户可以在命令提示符下使用该工具执行比较任务。

7. PowerShell

PowerShell 是 Microsoft SQL Server 2008 系统的新功能,是一个脚本和服务器导航引擎。用户可以使用该工具导航服务器上的所有对象,就好像它们是文件系统中目录结构的一部分一样,甚至可以使用诸如 dir、cd 类型的命令。

5.2 SQL Server 2008 数据库管理

5.2.1 数据库概述

数据库是以文件的形式存储在磁盘上的。SQL Server 2008 数据库中的每个数据库由多个操作系统文件组成,数据库的所有数据、对象和数据库操作日志都存储在这些操作系统文件中。

1. 文件类型

SQL Server 数据库通过数据文件保存与数据库相关的数据和对象。在 SQL Server 2008 中有两种类型的数据文件:

(1) 主数据文件。主数据文件是数据库的起点,其中包含了数据库的初始信息,并记录数据库还拥有哪些文件。每个数据库有且只能有一个主数据文件。主数据文件是数据库必须的文件,Microsoft 建议的主数据文件的扩展名是 .mdf。

(2) 次要数据文件。除主数据文件以外的所有其他数据文件都是次要数据文件。次要数据文件不是数据库必须的文件。Microsoft 建议的次要数据文件的扩展名是 .ndf。

2. 数据文件结构

数据文件的结构按照层次可以划分为页和区,每个数据文件由若干个大小为 64KB 的区组成,每个区由 8 个 8KB 大小的连续空间组成,这些连续空间称为页。

(1) 页。在 SQL Server 中,页是数据存储的基本单位。为数据库中的数据文件分配的磁盘空间可以从逻辑上划分成带有连续编号的页(编号从 0 开始)。磁盘 I/O 操作在页级执

行，SQL Server 读取或写入的是所有的数据页。

（2）区。区是 SQL Server 分配给表和索引的基本单位。区有统一区和混合区这两种类型。

3. 数据库文件组

为了更好地实现数据库文件的组织，以便于分配和管理，SQL Server 从 7.0 版本开始允许将多个文件归纳为一组，并赋予一个名称，这就是数据库文件组。通过设置文件组，可以有效提高数据库的读写性能，例如可以将文件组中的文件设立在不同的磁盘上，然后指定文件组同时对不同磁盘上的数据进行处理，从而提高存取的速度。

在使用数据库文件组时，需要注意的是一个数据库文件不能存在于多个数据文件组中，事务日志文件不属于任何文件组，文件组只实现对数据文件的管理。

SQL Server 数据库文件组分为以下 3 类：

（1）主文件组：包括主数据文件和所有没有被包含在其他文件组里的文件。数据库的系统表都包含在主文件组里。

（2）自定义文件组：包括所有在使用 CREATE DATABASE 或 ALTER DATABASE 时使用 FILEGROUP 关键字约束的文件。

（3）默认文件组：容纳所有在创建时没有指定文件组的表、索引以及 text、ntext、image 数据类型的数据。只能有一个文件组被指定为默认文件组，默认情况下，主文件组被当作默认文件组。

4. 事务日志文件

在 SQL Server 2008 中，每个数据库至少拥有一个自己的日志文件（也可以拥有多个日志文件）。日志文件的大小最少是 1MB，默认扩展名是.ldf，用来记录数据库的事务日志，即记录所有事务以及每个事务对数据库所做的修改。

5. 系统数据库

系统数据库是在 SQL Server 2008 安装完成时系统自动建立的特殊数据库，包括 master、model、msdb 和 tempdb 共 4 个数据库。

（1）master 数据库：是 SQL Server 2008 中的主控数据库，记录了 SQL Server 系统的所有系统信息，包括所有的登录信息、系统设置信息、系统初始化信息、其他系统数据库和用户数据库的信息等。只要对上述信息做过修改，例如创建一个新的数据库、增加新的用户等，系统都会自动修改 master 数据库的相关数据表中的数据。

（2）model 数据库：是 SQL Server 2008 中的模板数据库，当创建一个用户数据库时，系统会自动将 model 数据库的内容复制到该数据库中。用户可以将一些自定义规则的数据，如用户自定义数据类型等建立在 model 数据库中，这样每次建立新的数据库时，新数据库都将自动拥有这些数据库对象，从而省略了很多重复的建立工作。

（3）msdb 数据库：主要被 SQL Server Agent 用于进行复制、作业调度以及管理报警等活动，它常用于通过调度任务排除故障。

（4）tempdb 数据库：用于存储用户创建的临时表、存储过程或全局变量。在 tempdb

数据库中存放的所有数据信息都是临时的，当与数据库的连接断开时，所有的临时表格和其他临时对象都将被自动删除。tempdb 数据库是一个全局资源，没有专门的权限限制，允许所有可以连接上 SQL Server 服务器的用户使用。

6. 系统数据表

SQL Server 2008 用系统数据表记录所有服务器活动的信息。在系统数据库和用户数据库中都存在着系统数据表，当用户创建一个新的数据库时，会发现系统已经自动生成了一些系统数据表。因本书篇幅有限，只简要介绍几个最重要的系统表，其他系统表的详细信息，请参阅 SQL Server 2008 的联机丛书。

(1) sysobjects 数据表：是 SQL Server 的主系统表，它记录所有数据库对象的相关信息，对每个数据库对象都用一行记录来标识。

(2) syscolumns 数据表：用来记录表、视图中的列、存储过程的参数信息。

(3) sysindexes 数据表：用来记录有关索引和建立索引的表的相关信息。

(4) sysusers 数据表：用来记录所有服务器用户的信息，包括 Windows NT 用户、Windows NT 用户组、SQL Server 用户或 SQL Server 角色的信息。

(5) sysdatabases 数据表：只出现在 master 数据库中，用来记录所有 SQL Server 数据库的相关信息。

(6) sysdepends 数据表：用来记录表、视图和存储过程之间的依赖关系。

对于系统数据表，不能使用 DELETE、INSERT、UPDATE 等 SQL 语句直接修改其中的内容，也不允许编写程序直接对系统表中的信息进行访问，否则可能导致数据库发生一些难以诊断的错误，甚至导致系统瘫痪。如果需要访问或修改系统表的信息，应该使用系统存储过程或 Transact-SQL 提供的系统函数。

5.2.2 创建数据库

数据库包括物理结构和逻辑结构，而数据库文件又包括数据文件和事务日志文件。所以在创建数据前要对数据库进行规划，如数据库名、数据库的大小、增量等。

在创建数据库时，通常考虑以下几个方面问题：

(1) 数据的逻辑结构：包括数据库名、数据库所有者；

(2) 数据的物理结构：包括数据文件和事务日志文件的逻辑名、物理名、初始大小、增长方式及最大容量。一般增量按照固定大小来设定，以固定时间内可能增长的空间大小作为增长步长，尽量不要用百分比作为增长方式；

(3) 数据库的用户：包括用户权限和数量问题；

(4) 数据库的性能：包括数据库的大小与硬件配置的平衡、是否使用文件组等；

(5) 数据库的维护：包括数据库的备份和恢复。

在 SQL Server 2008 中可以通过两种方法来创建数据库：使用 Transact-SQL 语句创建数据库；使用对象资源管理器创建用户数据库。

1. 利用 Transact-SQL 语句创建数据库

在 SQL Server 2008 的查询分析器中使用 Transact-SQL 语句 CREATE DATABASE

也可以创建新的数据库。CREATE DATABASE 语句格式如下：

```
CREATE DATABASE database_name
ON
{ [ PRIMARY ] ( NAME = logical_file_name ,
FILENAME = 'os_file_name'
[ , SIZE = size]
[ , MAXSIZE = { max_size | UNLIMITED } ]
[ , FILEGROWTH = growth_increment ])
} [ ,...n ]
LOG ON
{ [ PRIMARY ] ( NAME = logical_file_name ,
FILENAME = 'os_file_name'
[ , SIZE = size]
[ , MAXSIZE = { max_size | UNLIMITED } ]
[ , FILEGROWTH = growth_increment ])
} [ ,...n ]
```

在 Transact-SQL 语言的命令格式中，[ ]表示该项可以省略，[,...*n*]表示可以多次重复前面的内容，<>表示在实际编写语句时，用相应的内容替代，类似 A|B 的格式表示 A 和 B 只能选择一个，{}表示该项必选。

上述创建数据库的语句中各参数意义的简要介绍如下，更详细说明请参照联机丛书。

- database_name：新数据库的名称。
- ON：指定显式定义用来存储数据库数据部分的磁盘文件(数据文件)。
- PRIMARY：在主文件组中指定文件。
- LOG ON：指定用来存储数据库日志的磁盘文件(日志文件)。
- NAME：指定文件的逻辑名称。
- FILENAME：指定操作系统(物理)文件名称。
- os_file_name：创建文件时由操作系统使用的路径和文件名。
- SIZE：指定文件的大小。
- MAXSIZE：指定文件可增大到的最大大小。
- UNLIMITED：指定文件将增长到整个磁盘。
- FILEGROWTH：指定文件的自动增量。

【例 5-1】 用 CREATE DATABASE 语句创建 Test 数据库，所有参数均取默认值。

语句清单如下：

```
CREATE DATABASE Test
```

这是利用 CREATE DATABASE 语句创建数据库最简单的方法。将上述语句在 SQL Server 2008 新建查询的编辑窗格中输入，单击工具栏上的“执行”按钮，就可以快速建立 Test 数据库。

【例 5-2】 用 CREATE DATABASE 语句创建一个数据库，数据库名为 ToyUniverse，此数据库包含一个数据文件和一个事务日志文件，主要参数如表 5-1 所示，其他参数均取默认值。

表 5-1 参数表

选项		参数
数据库名称		ToyUniverse
数据文件	逻辑文件	ToyUniverse-Data
	物理文件名	D:\SQL2008\DataBase\ToyUniverse-DataMDF
	初始大小	10MB
	最大容量	不受限制
	增长量	5MB
事务日志文件	逻辑文件名	ToyUniverse-Log
	物理文件名	D:\SQL2008\DataBase\ToyUniverse-LogLDF
	初始大小	10MB
	最大容量	2000MB
	增长量	10%

语句清单如下：

```
USE master
GO
CREATE DATABASE ToyUniverse
ON PRIMARY
(
    NAME = ToyUniverse_Data,
    FILENAME = 'D:\SQL2008\DataBase\ToyUniverse_Data.MDF',
    SIZE = 10,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 5
)
LOG ON
(
    NAME = ToyUniverse_Log,
    FILENAME = 'D:\SQL2008\DataBase\ToyUniverse_Log.LDF',
    SIZE = 10,
    MAXSIZE = 2000,
    FILEGROWTH = 10 %
)
GO
```

2. 使用对象资源管理器创建用户数据库

具体步骤：依次选择“开始”→“所有程序”→SQL Server Management Studio→“数据库”→“新建数据库”，如图 5-9 所示。

使用对象资源管理器创建用户数据库，命名要符合 SQL Server 2008 命名规则：长度在 1～128 个字符之间，名称的第一个字符必须是字母或_、@、# 中的任意字符；中文名称不能包含空格，也不要包含 SQL Server 2008 的保留字(如 master)，在图 5-10 所示的“数据库名称”后的文本框中输入数据库名称，然后单击“确定”按钮即可。

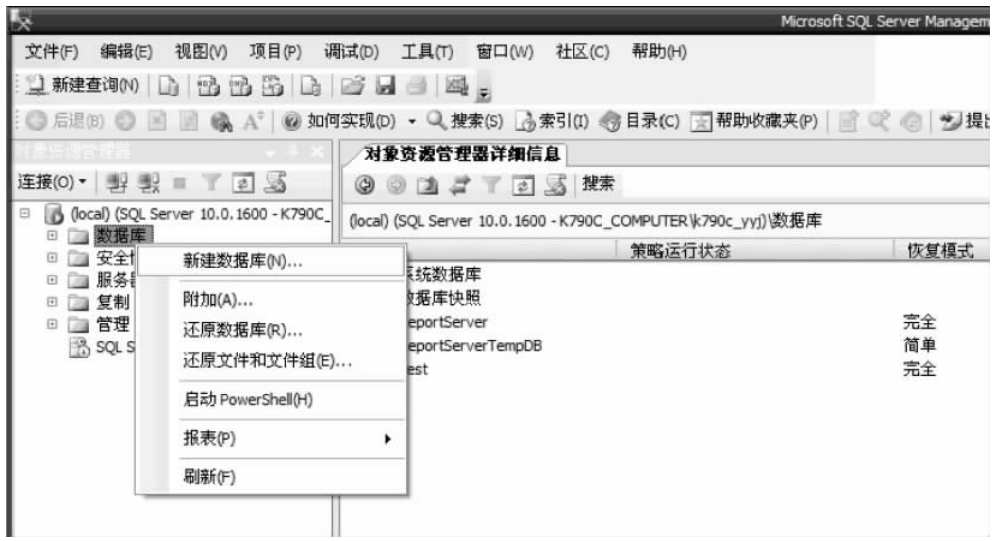


图 5-9 使用对象资源管理器新建数据库

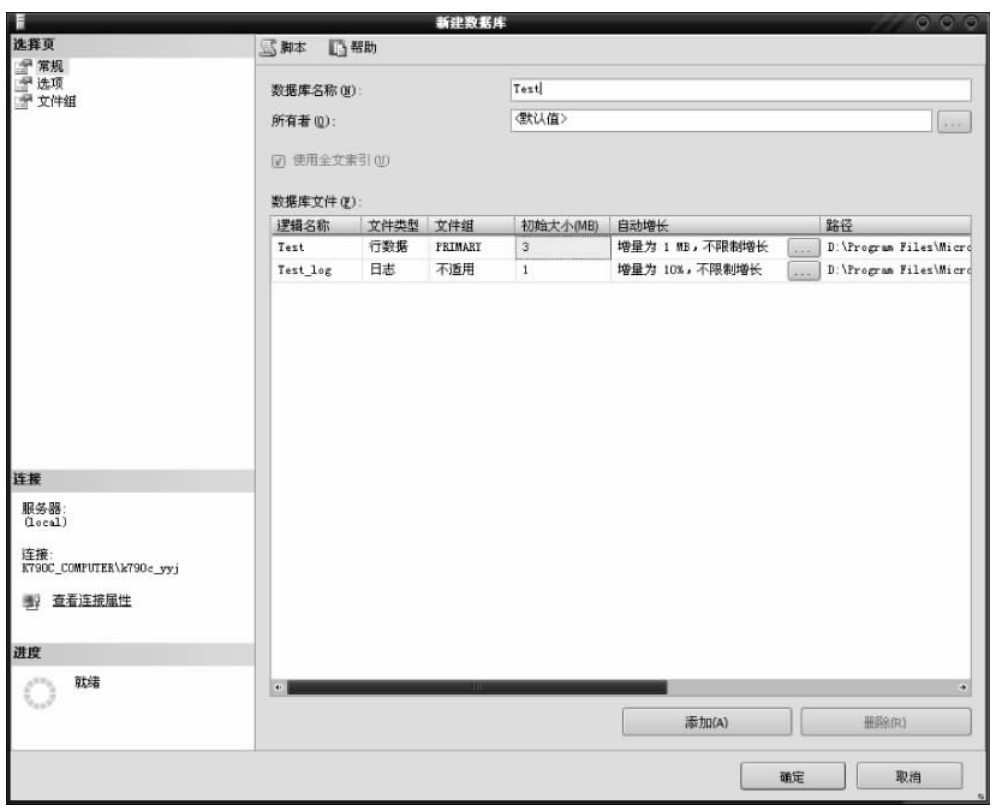


图 5-10 新建数据库名称

5.2.3 查看数据库

数据库创建之后，可以通过查看数据库操作来了解当前数据库的相关信息。数据库信息主要包括基本信息、维护信息和空间使用情况。可以通过 SQL Server 2008 的对象资源管理器和 Transact-SQL 语句两种方法来查看和修改数据库的配置信息。

1. 利用对象资源管理器查看数据库

(1) 依次选择“开始”→“所有程序”→Microsoft SQL Server 2008→SQL Server Management Studio，打开对象资源管理器窗口。

(2) 在“对象资源管理器”窗口展开数据库实例下的“数据库”节点，接着选中需要查看或配置的数据库并右击，从弹出的快捷菜单中选择“属性”命令，如图 5-11 所示。

(3) 在选定数据库的“属性”对话框中首先显示的是“常规”选项卡，如图 5-12 所示，在该选项卡中可以查看数据库的状态、所有者、创建日期等基本信息。要查看其他信息，可以分别单击“文件”、“文件组”、“选项”、“更改跟踪”、“权限”、“扩展属性”、“镜像”、“事务日志传送”选项卡。

图 5-11 数据库属性

2. 利用 Transact -SQL 语句修改用户数据库

1) 选择数据库

在 SQL Server 服务器上，可能存在多个用户数据库，用户只有连接上所要使用的数据

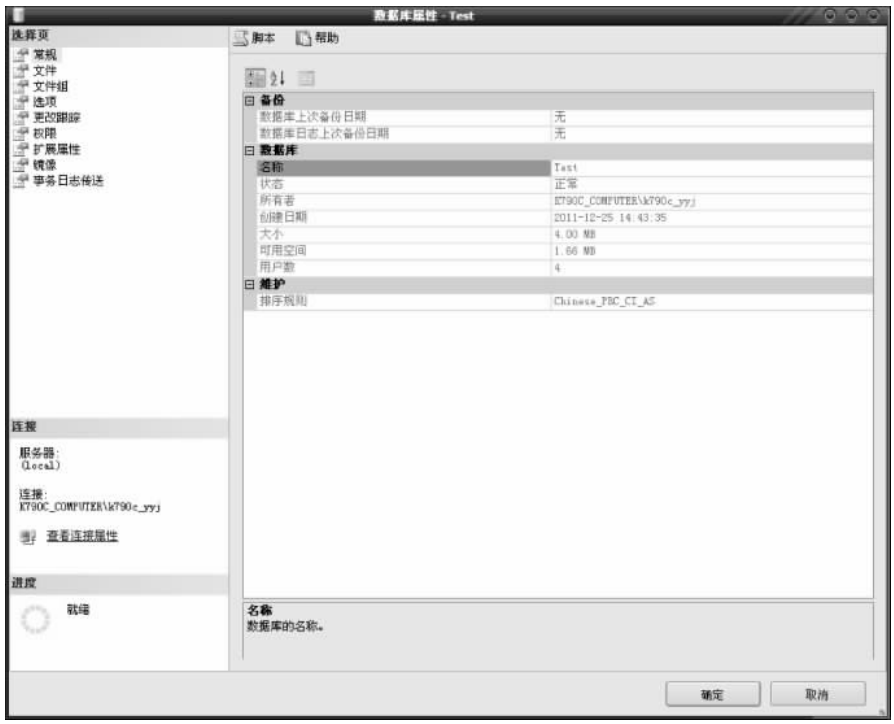


图 5-12 数据库属性常规选项卡

库,才能对该数据库中的数据进行操作。选择数据库语句的语法格式如下:

```
USE database_name
```

其中,database_name 为选择的数据库名称。

2) 查看数据库属性

数据库的属性信息都保存在系统数据库和系统数据表中,可以通过系统提供的存储过程来获取有关数据库的属性信息。

sp_helpdb:显示数据库和数据库参数信息。

sp_spaceused:查看数据库空间信息。

sp_options:查看数据库选项信息。

【例 5-3】 查询数据库 Test 的相关参数信息。

```
Exec sp_helpdb Test
```

依次选择“开始”→“所有程序”→ Microsoft SQL Server 2008→SQL Server Management Studio→“新建查询”,然后输入“Exec sp_helpdb Test”,单击执行,结果如图 5-13 所示。

【例 5-4】 查询数据库 Test 的空间信息。操作步骤与例 5-3 一致,语句为:

```
Use Test Exec sp_spaceused
```

【例 5-5】 查询数据库 Test 的选项信息。操作步骤与例 5-3 一致,语句为:

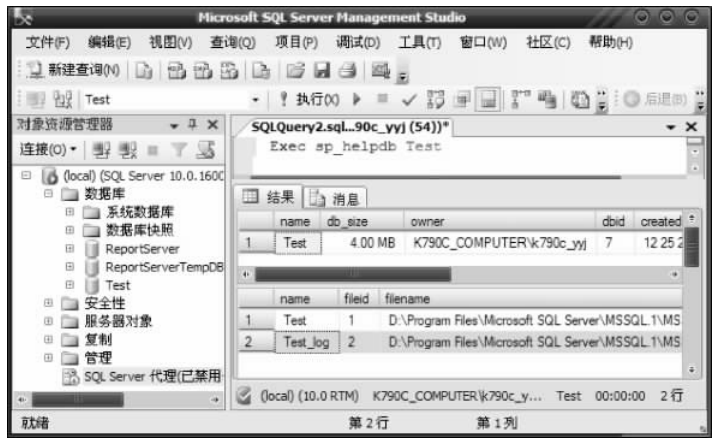


图 5-13 查询数据库信息

```
Exec sp_dboption student
```

5.2.4 修改数据库

对数据库信息的修改可以通过 SQL Server 2008 的 SQL Server Management Studio 或执行 Transact-SQL 语句这两种方法来实现。

1. 利用 SQL Server Management Studio 修改数据库

利用 SQL Server Management Studio 修改数据库信息的方法和利用 SQL Server Management Studio 来查看数据库信息一样，用同样的方法打开要修改数据库的“属性”对话框，在“文件”、“文件组”、“选项”、“更改跟踪”、“权限”、“扩展属性”、“镜像”、“事务日志传送”选项卡中修改需改变的属性即可。

2. 利用 Transact-SQL 语句修改数据库

修改数据库名字或其他属性可通过 Transact-SQL 提供的 ALTER DATABASE 语句来实现，只有 sysadmin 和 dbcreator 固定服务器角色成员以及 db_owner 固定数据库角色成员才能执行该语句。ALTER DATABASE 语句格式如下：

```
ALTER DATABASE database_name
{ ADD FILE <filespec> [ ,...n ] [ TO FILEGROUP
{ filegroup_name } ]
  | ADD LOG FILE <filespec> [ ,...n ]
  | REMOVE FILE logical_file_name
  | MODIFY FILE <filespec>
  | ADD FILEGROUP filegroup_name
  | REMOVE FILEGROUP filegroup_name
  | MODIFY FILEGROUP filegroup_name { filegroup_property |
    NAME = new_filegroup_name }
```

上述修改数据库的语句中各参数的意义简要介绍如下，更详细说明请参照联机丛书。

(1) ADD FILE：向数据库文件组添加新的数据文件。

(2) ADD LOG FILE：向数据库添加事务日志文件。

(3) REMOVE FILE：从 SQL Server 的实例中删除逻辑文件说明并删除物理文件。

(4) MODIFY FILE：修改某一文件的属性。

(5) ADD FILEGROUP：向数据库添加文件组。

(6) REMOVE FILEGROUP：从实例中删除文件组。

(7) MODIFY FILEGROUP：修改某一文件组的属性。

【例 5-6】 修改数据库 ToyUniverse 的数据文件 ToyUniverse_Data2 的属性，将其初始大小改为 10MB，最大容量改为 1000MB，增幅改为 10MB。

语句清单如下：

```
ALTER DATABASE ToyUniverse
MODIFY FILE
(
    NAME = ToyUniverse_Data2,
    SIZE = 10,
    MAXSIZE = 1000,
    FILEGROWTH = 10
)
GO
```

【例 5-7】 为数据库 ToyUniverse 添加一个数据库文件 ToyUniverse_Data2 和一个事务日志文件 ToyUniverse_Log2。

语句清单如下：

```
ALTER DATABASE ToyUniverse
ADD FILE
(
    NAME = ToyUniverse_Data2,
    FILENAME = 'D:\SQL2008\DataBase\ToyUniverse_Data2.NDF',
    SIZE = 5,
    MAXSIZE = UNLIMITED,
    FILEGROWTH = 5
)
GO
ALTER DATABASE ToyUniverse
ADD LOG FILE
(
    NAME = ToyUniverse_Log2,
    FILENAME = 'D:\SQL2008\DataBase\ToyUniverse_Log2.LDF',
    SIZE = 10,
    MAXSIZE = 2000,
    FILEGROWTH = 10 %
)
GO
```

5.2.5 删除数据库

如果数据库不再需要，为节省存储空间，可删除该数据库。数据库一旦被删除，即被永

久删除，文件和其数据都将从服务器上的磁盘中删除。对数据库的删除可以通过 SQL Server 2008 的对象资源管理器或执行 Transact-SQL 语句两种方法实现。

1. 利用对象资源管理器删除数据库

依次选择“开始”→“所有程序”→Microsoft SQL Server 2008→SQL Server Management Studio，打开对象资源管理器窗口。选择所要删除的数据库，右击并选择“删除”，然后在弹出的窗口单击“确定”即可完成删除操作。

2. 利用 Transact-SQL 语句删除用户数据库

使用 Transact-SQL 的 DROP DATABASE 语句可以删除用户数据库，其语法格式如下：

```
DROP DATABASE database_name
```

其中，database_name：指定要删除的数据库的名称。

【例 5-8】 删除已存在的 Test 数据库。

语句清单如下：

```
DROP DATABASE Test
```

5.3 SQL Server 2008 数据表管理

5.3.1 数据表概述

SQL Server 2008 是通过数据库来管理所有信息的，而在数据库中，用户真正关心并实际访问的数据则存储在各个数据表中，数据表是 SQL Server 中最重要的数据库对象。

1. 数据表概念

数据库是保存数据的集合，其目的在于存储和返回数据。如果没有数据库的表所提供的结构，这些任务是不可能完成的。数据库中包含一个或多个表，表是数据库的基本构造块。同时，表是数据的集合，是用来存储数据和操作数据的逻辑结构。表由行和列所构成，行被称为记录，是组织数据的单位；列被称为字段，每一列表示记录的一个属性。

在 SQL Server 2008 中，数据表分为永久数据表和临时数据表两种。永久数据表在创建后一直存储在数据库文件中，直至用户删除为止。而临时数据表则在用户退出或系统修复时被自动删除。

2. 数据类型

在数据表中的每一个数据列都会有特定的属性，而这些属性中最重要的就是数据类型(Data Type)，数据类型是用来定义储存在数据列中的数据，它限制了一个列中可以存储的数据的类型，在某些情况下甚至限制了该列中的可能值的取值范围。

在 SQL Server 2008 中，数据类型可以是系统提供的数据类型，也可以是用户自定义的

数据类型。

1）系统数据类型

SQL Server 系统数据类型有 7 类，如表 5-2 所示。

表 5-2 SQL Server 系统数据类型

数据类型分类	基 本 目 的
精确数值	存储带小数或不带小数的精确数值
近似数值	存储带小数或不带小数的数值
货币	存储带小数位的数值；专门用于货币值，最多可以有 4 个小数位
日期和时间	存储日期和时间信息，并强制实施特殊的年代规则
字符	存储基于字符的可变长度的值
二进制	存储以严格的二进制(0 和 1)表示的数据
专用数据类型	要求专门处理的复杂数据类型，诸如 XML 文档

说明：可变长度数据类型具有变动长度的特性，因为 VARCHAR 数据类型的存储长度为实际数值长度，若输入数据的字符数小于 n，则系统不会在其后添加空格来填满设定好的空间。反之固定长度的类型如果输入字符长度比定义长度小，则在其后添加空格来填满长度。

(1) 精确数值数据类型如表 5-3 所示。

表 5-3 精确数值数据类型

数 据 类 型	存 储 长 度	取 值 范 围	说 明
BIT	1B	0 或者 1	如果输入 0 或 1 以外的值，将被视为 1
INT	4B	−2^31 ～2^31−1	正/负整数
SMALLINT	2B	−32768～32767	正/负整数
TINYINT	1B	0～255	正整数
BIGINT	8B	−2^63 ～2^63−1	大范围的正/负整数
DECIMAL(p,s)	5～17B	−10^38 +1 ～ 10^38−1	最大可存储 38 位十进制数
NUMERIC(p,s)	5～17B	−10^38 +1 ～ 10^38−1	与 DECIMAL 等价

其中，p 表示可供存储的值的总位数(不包括小数点)，默认值为 18；s 表示小数点后的位数，默认值为 0。

(2) 近似数值数据类型如表 5-4 所示。

表 5-4 近似数值数据类型

数据类型	存储长度	取 值 范 围	说 明
FLOAT(p)	4B 或 8B	1.79E+308～−2.23E−308、0 和 2.23E−308～1.79E+308	存储大型浮点数
REAL	4B	−3.40E+38～−1.18E−38、0 和 1.18E−38～3.40E+38	后续被 float 替换

(3) 货币数据类型如表 5-5 所示。

表 5-5 货币数据类型

数据类型	存储长度	值 域	说 明
MONEY	8B	−922 337 203 685 477.5808～922 337 203 685 477.5807	存储大型货币值
SMALLMONEY	4B	−214 748.3648～214 748.3647	存储小型货币值

（4）日期和时间数据类型如表 5-6 所示。

表 5-6 日期和时间数据类型

数据类型	存储长度	取值范围	精度
DATE	3B	0001-01-01～9999-12-31	1 天
TIME	3B～5B	00:00:00.0000000～23:59:59.9999999	100 纳秒
SMALLDATETIME	4B	1900-01-01～2079-06-06	1 分钟
DATETIME	8B	1753-01-01～9999-12-31	0.003 33 秒
DATETIME2	6B～8B	0001-01-01 00:00:00.0000000～9999-12-3123:59:59.9999999	100 纳秒
DATETIMEOFFSET	8B～10B	0001-01-01 00:00:00.0000000～9999-12-3123:59:59.9999999	100 纳秒

（5）字符数据类型如表 5-7 所示。

表 5-7 字符数据类型

数据类型	存储长度	取值范围	说明
CHAR(*n*)	1～8000B	最多 8000 个字符	固定长度 ANSI 数据类型
NCHAR(*n*)	2～8000B	最多 4000 个字符	固定长度 Unicode 数据类型
VARCHAR(*n*)	1～8000B	最多 8000 个字符	可变长度 ANSI 数据类型
VARCHAR(max)	最大 2GB	最多 1 073 741 824 个字符	可变长度 ANSI 数据类型
NVARCHAR(*n*)	2～8000B	最多 4000 个字符	可变长度 Unicode 数据类型
NVARCHAR(max)	最大 2GB	最多 536 870 912 个字符	可变长度 Unicode 数据类型
TEXT	最大 2GB	最多 1 073 741 824 个字符	可变长度 Unicode 数据类型
NTEXT	最大 2GB	最多 536 870 912 个字符	可变长度 Unicode 数据类型

（6）二进制数据类型如表 5-8 所示。

表 5-8 二进制数据类型

数据类型	存储长度	说明
BINARY(*n*)	1～8000B	存储固定大小的二进制数据
VARBINARY(*n*)	1～8000B	存储可变大小的二进制数据
ARBINARY(max)	最大 2GB	存储可变大小的二进制数据
IMAGE	最大 2GB	存储可变大小的二进制数据

说明：在 Microsoft SQL Server 的未来版本中将删除 NTEXT、TEXT 和 IMAGE 数据类型。请避免在新开发工作中使用这些数据类型，并考虑修改当前使用这些数据类型的应用程序，改用 NVARCHAR(max)、VARCHAR(max)和 VARBINARY(max)。

（7）专用数据类型如表 5-9 所示。

表 5-9 专用数据类型

数据类型	说明
TABLE	用于存储结果集以进行后续处理，临时存储一组作为表值函数的结果集返回的行
ROWVERSION	通常用作给表行加版本戳的机制。存储大小为 8 个字节
TIMESTAMP	其数据类型为 ROWVERSION 数据类型的同义词
UNIQUWIDENTIFIER	一个 16 字节 GUID。用来全局标识数据库、实例和服务器中的一行
CURSOR	这是变量或存储过程 OUTPUT 参数的一种数据类型，这些参数包含对游标的引用
XML	存储 XML 数据的数据类型。可以在列中或者 XML 类型的变量中存储 XML 实例

说明：后续版本的 Microsoft SQL Server 将删除该功能。请避免在新的开发工作中使用该功能，并着手修改当前还在使用该功能的应用程序。

2）用户自定义数据类型

除了系统提供的数据类型外，用户还可以根据需要，在系统的数据类型基础上定制数据类型。当定义数据类型时，需要指定该数据类型的名称、基数据类型和是否为空。自定义数据类型可以保证数据的完整性，使开发团队保证数据的一致性。

(1) 用 Transact-SQL 语句创建用户自定义数据类型。

```
CREATE TYPE type_name
{
  FROM base_type
    [ ( precision [ , scale ] ) ]
        [ NULL | NOT NULL ]
}
[ ; ]
```

参数说明如下：

base_type ：用户自定义数据类型所基于的数据类型，由 SQL Server 2008 提供。

precision ：指定数据类型的精度。

scale：对于 decimal 或 numeric，指示小数点位数，它必须小于或等于精度值。

NULL|NOT NULL：指定此类型是否可容纳空值。如果未指定，则默认值为 NULL。

(2) 利用对象资源管理器创建用户自定义数据类型。

① 使用“Windows 身份验证”连接到数据库实例。

② 展开需要创建用户自定义数据类型的数据库，选择“可编程性”→“类型”，右击，然后从弹出的快捷菜单中选择“新建”→“用户定义数据类型”命令，打开“新建用户定义数据类型”对话框。

③ 在“新建用户定义数据类型”对话框中，可以定义类型的结构、名称、数据类型、精度、允许为空值等。

④ 完成设置后，单击“确定”按钮，创建用户自定义数据类型。

5.3.2 创建数据表

SQL Server 2008 提供了两种方法创建数据表：利用可视化数据库管理工具 SQL Server Management Studio；利用 Transact-SQL 语句。在 SQL Server 2008 中，每个数据库可以创建许多表，每个表至多可定义 1024 列。表和列的名称必须遵守标识符的规定，在特定表中必须是唯一的，但同一数据库的不同表中可使用相同的列名。必须为每列指定数据类型。尽管在一个数据库内对于每一个结构表的名称必须是唯一的，但如果为每张表指定了不同的结构，则可以创建多个具有相同名称的表。当必须使用某一表时，可以通过指定表的结构以及表的名称来区分这两个表。

1. 利用可视化数据库管理工具 SQL Server Management Studio 创建数据表

1）创建表结构

(1) 依次选择“开始”→“程序”→ Microsoft SQL Server 2008 → SQL Server

Management Studio，打开可视化数据库管理工具“SQL Server Management Studio”窗口。

（2）在窗口左边的树状窗格中，依次展开 Microsoft SQL Server→SQL Server 组→“数据库”节点，在列出的数据库列表中选中要在其上创建数据表的数据库并展开，在其下的“表”节点右击，在弹出的快捷菜单中选择“新建表”命令，打开表设计器窗口，如图 5-14 所示。

图 5-14　表设计器窗口

（3）表设计器窗口的上部网格用来设置字段的基本属性。在“列名”列中，输入字段名。字段名最长为 128 个字符，可以包含汉字、英文字母、数字、下划线以及其他符号，在同一个数据表中，字段名必须唯一。在“数据类型”列中，从下拉列表中选择一种数据类型。在“允许 NULL 值”列中，单击，选择该字段是否允许为空。在“长度”列中指定字段的长度。

（4）表设计器窗口的下部的“列”选项卡用来设置上部网格选中的字段的附加属性，这些属性不是必要的。在“描述”栏中输入该字段的文字描述。在“默认值”栏中输入字段的默认值，在插入记录时，如果没有指定该字段的值，系统自动使用默认值代替。如果字段的数据类型为 decimal 或 numeric，需要在“精度”和“小数位数”栏指明精度和小数位数。“标识”栏用来设置该字段是否具有自动编号属性，“标识种子”用来设置自动编号的起始编号，“标识递增量”则指明每个编号的变化量。对于具有自动编号属性的字段，在输入数据时不必也不能为它输入任何值，系统会按照上述设置自动产生一个数字作为它的值。如果是字符数据，还可在“排序规则”栏定义排序的规则。

（5）在建立的过程中，如果需要插入或删除某一字段，可以在表设计器窗口的上部网格中选中该字段，右击，在出现的快捷菜单中单击相应的命令即可。

（6）单击工具栏上的“保存”按钮，在弹出的“选择名称”对话框中，输入数据表的名称，然后单击“确定”按钮，完成数据表结构的创建。

2）设置约束

（1）设置主键约束。在打开的表设计器中，在要设置主键的列上右击，在弹出的快捷菜单中单击“设置主键”命令，或选中该列后，直接单击工具栏中的“设置主键”按钮，设置成功后，在该列前有一个钥匙样图标，说明该列已被指定为主键。

（2）设置唯一约束。在打开的表设计器中，右击，在弹出的快捷菜单中单击“索引/键”命令，或直接单击工具栏中的“管理索引和键”按钮，打开“属性”对话框。在“索引/键”对话框中单击“添加”按钮，在“列”下拉框中选择要设置唯一约束的列，然后在“是唯一的”下拉框中选择“是”，单击“关闭”按钮，完成唯一约束的设置，如图 5-15 所示。

（3）设置检查约束。在打开的表设计器中，右击，在弹出的快捷菜单中单击“CHECK 约束”命令，或直接单击工具栏中的“管理 CHECK 约束”按钮，打开“CHECK 约束”对话框。单击“添加”按钮，在“表达式”文本框中输入相应的约束条件，例如“语文>=0”，并根据需要确定是否选中“创建和重新启用时检查现有数据”、“强制用于复制”、“强制用于 INSERT 和 UPDATE”复选框，单击“关闭”按钮，完成检查约束的设置。

（4）设置外键约束。在打开的表设计器中右击，在弹出的快捷菜单中单击“关系”命令，

图 5-15 “索引/键”对话框

或直接单击工具栏中的“关系”按钮，打开“外键关系”对话框。单击“添加”按钮，在“主键表”下拉框中选择外键引用的表，并在其下的列表框中选择外键引用的列；在“表和列规范”子项目中选择要定义外键约束的列。然后根据需要设置““强制外键约束”、“强制用于复制”、“INSERT 和 UPDATE 规范”等选项，单击“关闭”按钮，完成外键约束的设置。

2. 利用 Transact-SQL 语句创建数据表

1）创建表结构

创建数据表结构可通过 Transact-SQL 提供的 CREATE TABLE 语句来实现。CREATE TABLE 语句格式如下：

```
CREATE TABLE
    [ database_name . [ schema_name ] . | schema_name . ] table_name
        ( { <column_definition> | <computed_column_definition>
                | <column_set_definition> }
        [ <table_constraint> ] [ ,...n ] )
    [ ON { partition_scheme_name ( partition_column_name ) | filegroup
        | "default" } ]
    [ { TEXTIMAGE_ON { filegroup | "default" } ]
    [ FILESTREAM_ON { partition_scheme_name | filegroup
        | "default" } ]
    [ WITH ( <table_option> [ ,...n ] ) ]
[ ; ]
```

上述创建数据表结构语句中各参数的意义简要介绍如下，更详细说明请参照联机丛书。

（1）database_name：在其中创建表的数据库的名称。如果未指定，则 database_name 默认为当前数据库。

（2）schema_name：新表所属结构的名称。

（3）table_name：新表的名称。表名必须符合标识符规则。

（4）column_definition：表中列的定义，一般包括列名、类型等定义。

（5）computed_column_definition：表中列的相关约束。

（6）table_constraint：新表的相关约束。

(7) ON { <partition_scheme> | filegroup | "default" }：指定存储表的分区结构或文件组。

(8) TEXTIMAGE_ON { filegroup | "default" }：指示 TEXT、NTEXT、IMAGE、XML、VARCHAR(max)、NVARCHAR(max)、VARBINARY(max)和 CLR 用户定义类型的列存储在指定文件组的关键字。

(9) FILESTREAM_ON { partition_scheme_name | filegroup | "default" }：指定 FILESTREAM 数据的文件组。

(10) table_option：新表的选项。

用 CREATE TABLE 语句创建数据表时，如果不在语句中指明数据库名，则默认在当前打开的数据库中创建数据表。使某数据库成为当前打开的数据库，可以使用“USE 数据库名”语句。

【例 5-9】 用 CREATE TABLE 语句在 Test 数据库创建如表 5-1 所示的学生成绩表。

语句清单如下：

```
use Test;
CREATE TABLE 学生成绩表
(
学号     char(10) NOT NULL,
姓名     char(10) NOT NULL,
语文     int DEFAULT 0,
数学     int DEFAULT 0,
英语     int DEFAULT 0,
总分     int DEFAULT 0,
平均分   int DEFAULT 0
)
```

2) 设置约束

设置约束只需在 CREATE TABLE 语句中 table_constraint 部分加入 CONSTRAINT 约束子句即可。

(1) 设置主键约束子句格式如下：

```
[ CONSTRAINT constraint_name ]
{
    { PRIMARY KEY }
        [ CLUSTERED | NONCLUSTERED ]
              (column [ ASC | DESC ] [ ,...n ] )
        [WITH FILLFACTOR = fillfactor | WITH ( <index_option> [ , ...n ] )]
        [ ON { partition_scheme_name (partition_column_name) | filegroup | "default" } ]
}
```

其中，CONSTRAINT constraint_name 用于指定约束的名称，必须是唯一的，如果不指定，系统会自动生成一个约束名；CLUSTERED | NONCLUSTERED 用于指定索引的类型，即聚集索引和非聚集索引。

【例 5-10】 用 CREATE TABLE 语句在 Test 数据库创建如表 5-1 所示的学生成绩表，并将学号设置为主键。

语句清单如下：

```
USE Test
CREATE TABLE 学生成绩表
(
学号     char(10) NOT NULL,
姓名     char(10) NOT NULL,
语文     int DEFAULT 0,
数学     int DEFAULT 0,
英语     int DEFAULT 0,
总分     int,
平均分   int,
CONSTRAINT PRIMARY_KEY_FIELD PRIMARY KEY (学号)
)
```

(2) 设置唯一约束子句格式如下：

```
[ CONSTRAINT constraint_name ]
{
    { UNIQUE }
        [ CLUSTERED | NONCLUSTERED ]
              (column [ ASC | DESC ] [ ,...n ] )
        [ WITH FILLFACTOR = fillfactor |WITH ( <index_option> [ , ...n ] ) ]
        [ ON { partition_scheme_name (partition_column_name)| filegroup |"default" } ]
}
```

【例 5-11】 用 CREATE TABLE 语句在 Test 数据库创建如表 5-1 所示的学生成绩表，并将学号设置为主键，姓名设置为具有唯一性。

语句清单如下：

```
USE Test
CREATE TABLE 学生成绩表
(
学号     char(10) NOT NULL,
姓名     char(10) NOT NULL,
语文     int DEFAULT 0,
数学     int DEFAULT 0,
英语     int DEFAULT 0,
总分     int,
平均分   int,
CONSTRAINT PRIMARY_KEY_FIELD PRIMARY KEY (学号),
CONSTRAINT UNIQUE_FIELD UNIQUE(姓名)
)
```

(3) 设置检查约束子句格式如下：

```
[ CONSTRAINT constraint_name ]
{
    CHECK [ NOT FOR REPLICATION ] ( logical_expression )
}
```

【例 5-12】 用 CREATE TABLE 语句在 Test 数据库创建如表 5-1 所示的学生成绩表，

并将学号设置为主键，姓名设置为具有唯一性，总分设置为必须大于等于 0。

语句清单如下：

```
USE Test
CREATE TABLE 学生成绩表
(
学号     char(10) NOT NULL,
姓名     char(10) NOT NULL,
语文     int DEFAULT 0,
数学     int DEFAULT 0,
英语     int DEFAULT 0,
总分     int,
平均分   int,
CONSTRAINT PRIMARY_KEY_FIELD PRIMARY KEY (学号),
CONSTRAINT UNIQUE_FIELD UNIQUE(姓名),
CONSTRAINT CHECK_FIELD CHECK(总分>=0)
)
```

(4) 设置外键约束子句格式如下：

```
[ CONSTRAINT constraint_name ]
{
    FOREIGN KEY ( column [ ,...n ] )
      REFERENCES referenced_table_name [ ( ref_column [ ,...n ] ) ]
      [ ON DELETE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
      [ ON UPDATE { NO ACTION | CASCADE | SET NULL | SET DEFAULT } ]
      [ NOT FOR REPLICATION ]
}
```

【例 5-13】 用 CREATE TABLE 语句在 Test 数据库创建如表 5-1 所示的学生成绩表，并将学号设置为主键，姓名设置为具有唯一性，总分设置为必须大于等于 0，并将表中的学号字段与学生信息表(假设已存在)的学号字段建立外键约束。

语句清单如下：

```
USE Test
CREATE TABLE 学生成绩表
(
学号     char(10) NOT NULL,
姓名     char(10) NOT NULL,
语文     int DEFAULT 0,
数学     int DEFAULT 0,
英语     int DEFAULT 0,
总分     int,
平均分   int,
CONSTRAINT PRIMARY_KEY_FIELD PRIMARY KEY (学号),
CONSTRAINT UNIQUE_FIELD UNIQUE(姓名),
CONSTRAINT CHECK_FIELD CHECK(总分>=0),
CONSTRAINT FOREIGN_FIELD
    FOREIGN KEY(学号)
    REFERENCES 学生信息表(学号)
)
```

5.3.3 查看数据表

在数据库中创建了数据表后，有时需要查看数据表的一些相关信息，例如数据表的结构、约束、数据等。可以利用 SQL Server 2008 的可视化数据库管理工具或 Transact-SQL 提供的存储过程和语句来查看数据表的各种信息。

1. 利用可视化数据库管理工具查看数据表

1）查看数据表结构

（1）打开 SQL Server Management Studio 窗口。

（2）打开数据表所在数据库的节点，选中表名，右击，在弹出的快捷菜单中选择“设计”命令，弹出如图 5-16 所示的表设计器窗口。在表设计器窗口中显示了该数据表结构的信息，如各字段的名称、类型、长度、是否为空等属性。

2）查看数据表约束

（1）打开 SQL Server Management Studio 窗口。

（2）打开数据表所在数据库的节点，选中表名，右击，在弹出的快捷菜单中选择“设计”命令，弹出同创建数据表时相同的窗口，再按照 5.3.2 节中介绍的设置数据表约束相同的方法，就可以看到该数据表的各种约束。

图 5-16 表设计器窗口

2. 利用 Transact-SQL 的存储过程和语句查看数据表

利用 Transact-SQL 提供的系统存储过程 sp_help 可以查看数据表的结构和约束。sp_help 存储过程的语法格式如下：

```
[EXECUTE] sp_help [数据表名]
```

【例 5-14】 用 sp_help 存储过程查看学生成绩表的结构和约束。

语句清单如下：

```
USE Test
EXEC sp_help 学生成绩表
```

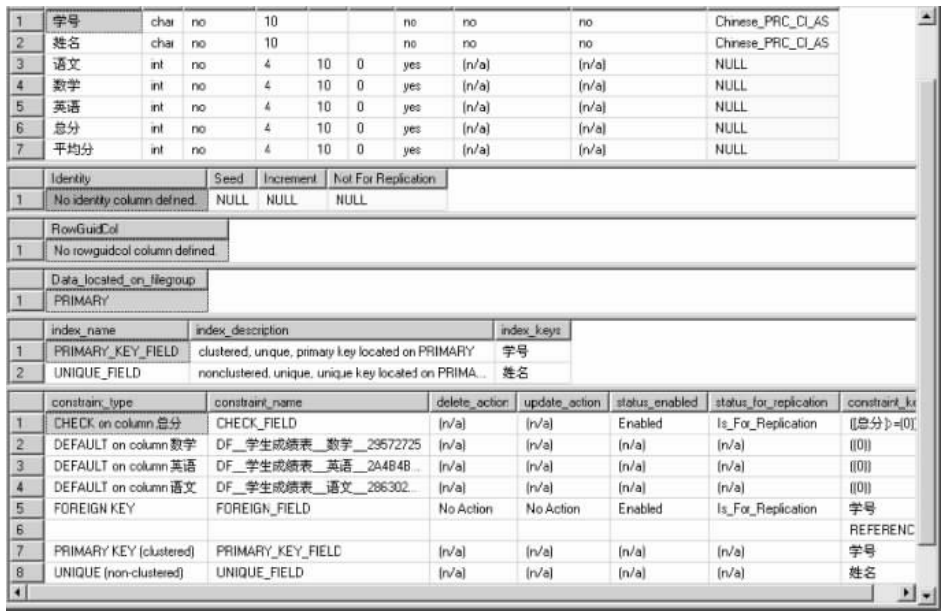
在 SQL 查询分析器中输入上述语句，执行结果如图 5-17 所示。

5.3.4 修改数据表

在数据库中创建了数据表后，有时需要对数据表的结构、约束或数据进行修改。利用 SQL Server 2008 的可视化数据库管理工具或 Transact-SQL 提供的语句可以修改数据表的各种信息。

1. 利用可视化数据库管理工具修改数据表

（1）打开 SQL Server Management Studio 窗口。



	学号	char	no	10			no	no	no	Chinese_PRC_CI_AS
2	姓名	char	no	10			no	no	no	Chinese_PRC_CI_AS
3	语文	int	no	4	10	0	yes	(n/a)	(n/a)	NULL
4	数学	int	no	4	10	0	yes	(n/a)	(n/a)	NULL
5	英语	int	no	4	10	0	yes	(n/a)	(n/a)	NULL
6	总分	int	no	4	10	0	yes	(n/a)	(n/a)	NULL
7	平均分	int	no	4	10	0	yes	(n/a)	(n/a)	NULL

	Identity	Seed	Increment	Not For Replication
1	No identity column defined.	NULL	NULL	NULL

	RowGuidCol
1	No rowguidcol column defined.

	Data_located_on_filegroup
1	PRIMARY

	index_name	index_description	index_keys
1	PRIMARY_KEY_FIELD	clustered, unique, primary key located on PRIMARY	学号
2	UNIQUE_FIELD	nonclustered, unique, unique key located on PRIMA...	姓名

	constraint_type	constraint_name	delete_action	update_action	status_enabled	status_for_replication	constraint_k
1	CHECK on column 总分	CHECK_FIELD	(n/a)	(n/a)	Enabled	Is_For_Replication	([总分]>=[0])
2	DEFAULT on column 数学	DF__学生成绩表__数学__29572725	(n/a)	(n/a)	(n/a)	(n/a)	((0))
3	DEFAULT on column 英语	DF__学生成绩表__英语__2A4B4B...	(n/a)	(n/a)	(n/a)	(n/a)	((0))
4	DEFAULT on column 语文	DF__学生成绩表__语文__286302...	(n/a)	(n/a)	(n/a)	(n/a)	((0))
5	FOREIGN KEY	FOREIGN_FIELD	No Action	No Action	Enabled	Is_For_Replication	学号
6							REFERENC
7	PRIMARY KEY (clustered)	PRIMARY_KEY_FIELD	(n/a)	(n/a)	(n/a)	(n/a)	学号
8	UNIQUE (non-clustered)	UNIQUE_FIELD	(n/a)	(n/a)	(n/a)	(n/a)	姓名

图 5-17 sp_help 执行结果

(2) 打开数据表所在数据库的节点,选中表名,右击,在弹出的快捷菜单中,选择“设计”命令,弹出同创建数据表时相同的窗口(图 5-14),利用 5.3.2 节介绍的创建方法,就可以对数据表的结构和约束进行修改。

(3) 如果要添加新字段,可在要插入字段的后一字段所在的列上右击,在出现的快捷菜单中单击“插入列”命令,即可在当前列前插入一个空行,以输入新的字段信息;如果要删除已有字段,可在要删除字段所在的列上右击,在出现的快捷菜单中单击“删除列”命令,即可将该列删除。

2. 利用 Transact-SQL 语句修改数据表

Transact-SQL 提供了 ALTER TABLE 语句来修改数据表的结构和约束,ALTER TABLE 的语法格式如下:

```
ALTER TABLE table
{ [ ALTER COLUMN column_name
    { new_data_type [ ( precision [ , scale ] ) ]
        [ COLLATE < collation_name > ]
        [ NULL | NOT NULL ]
        | {ADD | DROP } ROWGUIDCOL }
    ]
    | ADD
        { [ < column_definition > ]
        | column_name AS computed_column_expression
        } [ ,...n ]
    | [ WITH CHECK | WITH NOCHECK ] ADD
        { < table_constraint > } [ ,...n ]
    | DROP
```

```
        { [ CONSTRAINT ] constraint_name
            | COLUMN column_name } [ ,...n ]
    | { CHECK | NOCHECK } CONSTRAINT
        { ALL | constraint_name [ ,...n ] }
    | { ENABLE | DISABLE } TRIGGER
        { ALL | trigger_name [ ,...n ] }
}
```

上述修改数据表结构和约束语句中各参数的意义简要介绍如下，更详细说明请参照联机丛书。

(1) table：要更改的表的名称。

(2) ALTER COLUMN：指定要更改给定列。

(3) column_name：要更改、添加或除去的列的名称。

(4) new_data_type：要更改的列的新数据类型。

(5) precision：指定数据类型的精度。

(6) scale：指定数据类型的小数位数。

(7) COLLATE < collation_name >：为更改列指定新的排序规则。

(8) NULL | NOT NULL：指定该列是否可接受空值。

(9) [{ ADD | DROP } ROWGUIDCOL]：指定在指定列上添加或除去 ROWGUIDCOL 属性。

(10) ADD：指定要添加一个或多个列定义、计算列定义或者表约束。

(11) computed_column_expression：是一个定义计算列的值的表达式。

(12) WITH CHECK | WITH NOCHECK：指定表中的数据是否用新添加的或重新启用的 FOREIGN KEY 或 CHECK 约束进行验证。

(13) DROP { [CONSTRAINT] constraint_name | COLUMN column_name }：指定从表中删除 constraint_name 或者 column_name。

(14) { CHECK | NOCHECK} CONSTRAINT：指定启用或禁用 constraint_name。

(15) {ENABLE | DISABLE} TRIGGER：指定启用或禁用 trigger_name。

(16) trigger_name：指定要启用或禁用的触发器名称。

【例 5-15】 用 ALTER TABLE 语句修改学生成绩表的结构，将“姓名”字段的类型改为 varchar(8)型，删除“平均分”字段，增加“名次”字段，类型为 int 型。

语句清单如下：

```
USE Test;
ALTER TABLE 学生成绩表 ALTER COLUMN 姓名 varchar(8)
ALTER TABLE 学生成绩表 DROP COLUMN 平均分
ALTER TABLE 学生成绩表 ADD 名次 int
```

如果学生成绩表中建有约束，上述操作可能会失败，要先去除约束，才能执行成功。

【例 5-16】 用 ALTER TABLE 语句修改学生成绩表的结构，去除在例 5-11 中建立的名为 UNIQUE_FIELD 和 FOREIGN_FIELD 的约束，同时设置英语字段必须小于等于 100 的约束。

语句清单如下：

```
USE Test
ALTER TABLE 学生成绩表 DROP UNIQUE_FIELD, FOREIGN_FIELD
ALTER TABLE 学生成绩表 ADD CONSTRAINT ENGLISH_CHECK CHECK(英语<=100)
```

5.3.5 删除数据表

对于那些不再有用的数据表，可以将其删除。通过 SQL Server 2008 的可视化数据库管理工具或 Transact-SQL 语句都能达到删除数据表的目的。

1. 利用可视化数据库管理工具删除数据表

(1) 打开 SQL Server Management Studio 窗口。

(2) 打开数据表所在数据库的节点，选中表名，右击，在弹出的快捷菜单中选择“删除”命令，弹出“删除对象”对话框，如图 5-18 所示。在该对话框中单击“确定”按钮，就可以删除选定的数据表。需要注意的是，如果该表被其他表通过外键约束引用，除非先删除这些外键约束，否则不能删除该表。单击“显示依赖关系”按钮，可以查看该表所依赖的对象和依赖于该表的对象。

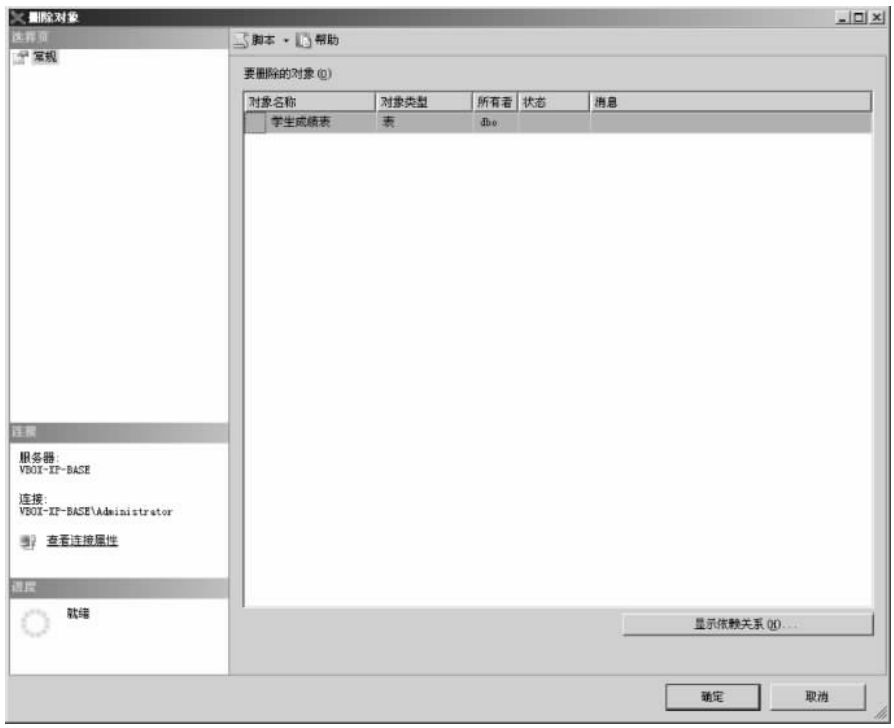


图 5-18 “删除对象”对话框

2. 利用 Transact-SQL 语句删除数据表

Transact-SQL 提供了 DROP TABLE 语句来删除数据表，不过 DROP TABLE 语句只能删除用户数据表，不能用来删除系统表。DROP TABLE 的语法格式如下：

```
DROP TABLE [数据表名]
```

【例 5-17】 用 DROP TABLE 语句删除学生成绩表。

语句清单如下：

```
USE Test
DROP TABLE 学生成绩表
```

5.4 SQL Server 2008 数据表基本操作

5.4.1 添加数据

数据表的结构一旦确定，就可以向新表中输入数据，生成记录。SQL Server 2008 提供了两种方法向数据表中添加数据：利用可视化数据库管理工具；利用 Transact-SQL 语句。

1. 利用可视化数据库管理工具添加数据

(1) 打开 SQL Server Management Studio 窗口。

(2) 打开数据表所在数据库的节点，选中表名，右击，在弹出的快捷菜单中，选择“编辑前 200 行”命令，在弹出的查询设计器结果窗格中输入新的数据即可，如图 5-19 所示。

图 5-19 向表中添加数据

2. 利用 Transact-SQL 语句添加数据

向新创建的数据表中输入数据可以通过 Transact-SQL 提供的 INSERT 语句来实现。INSERT 语句可以将一行或多行记录添加到一个已经存在的数据表中，INSERT 语句有两种格式。

1) INSERT…VALUES 语句

INSERT…VALUES 语句用来向数据库中添加一条新记录，其语法格式为：

```
INSERT [ INTO ] table_name
{ [ ( column_list ) ]
    { VALUES ( { DEFAULT | NULL | expression } [ ,...n] ) }
}
```

如果插入数据的顺序和个数与数据表的字段顺序和个数完全一致，可以省略 column_list，否则必须指明 column_list。

【例 5-18】 用 INSERT 语句向 Test 数据库的学生成绩表(表 5-1)添加一条记录：学号：200403；姓名：张杰；语文：78；数学：56；英语 50；总分：184；平均分：61。

语句清单如下：

```
INSERT INTO 学生成绩表 (学号 , 姓名 , 语文 , 数学 , 英语 , 总分 ,平均分)
VALUES ('200403' , '张杰' ,78,56,50,184,61)
```

或：

```
USE Test
INSERT INTO 学生成绩表 VALUES ('200403' , '张杰' ,78,56,50,184,61)
```

2）INSERT…SELECT 语句

INSERT…SELECT 语句用来向数据库中一次添加多条记录，这些记录是由 SELECT 子句生成的查询结果构成。INSERT…SELECT 语句语法格式为：

```
INSERT [ INTO ] table_name
{ [ ( column_list ) ]
    { SELECT [column_list] FROM table_name [WHERE search_conditions] }
}
```

【例 5-19】 用 INSERT 语句将学生信息表中性别为女的学生记录的学号和姓名添加到学生成绩表中。

语句清单如下：

```
USE Test
INSERT INTO 学生成绩表 (学号,姓名)
SELECT 学号,姓名 FROM 学生信息表 WHERE 性别 = '女'
```

5.4.2 查看数据

在数据表中添加数据后，可以利用可视化数据库管理工具和 Transact-SQL 语句来查看所添加的数据。

1. 利用可视化数据库管理工具查看数据

在可视化数据库管理工具中查看数据表中数据的方法与上一节中介绍的通过可视化数据库管理工具向数据表中添加数据的方法相同，详见 5.4.1 节相关内容。

2. 利用 Transact-SQL 语句查看数据

利用 Transact-SQL 提供的 SELECT 语句可以查看数据表中的所有数据，SELECT 语句的语法格式为：

```
SELECT * FROM 数据表名
```

【例 5-20】 用 SELECT 语句查看学生成绩表数据。

语句清单如下：

```
USE Test
SELECT * FROM 学生成绩表
```

5.4.3 修改数据

对数据表中的数据进行修改，是数据表的常见操作。SQL Server 2008 提供了两种方

法修改数据表中的数据：利用可视化数据库管理工具；利用 Transact-SQL 语句。

1. 利用可视化数据库管理工具修改数据

在可视化数据库管理工具中修改数据的方法同在可视化数据库管理工具中添加数据、查看数据的方法一样，在添加、查看数据的同时，就可以进行数据的修改，在此不做赘述。

2. 利用 Transact-SQL 语句修改数据

利用 Transact-SQL 提供的 UPDATE 语句可以修改数据表中的数据，UPDATE 语句的语法格式为：

```
UPDATE table_name
SET{ column_name = { expression | DEFAULT | NULL }
{ [ FROM { < table_source > } [ ,...n ] ]
        [ WHERE search_condition ] }
```

格式中的 table_name 指明要修改的数据表；使用 SET 子句指定要修改的列和修改后的数据；当没有 WHERE 子句时，表中所有记录的指定列都被修改；若修改的数据来自于另一个表，需要由 FROM 子句指明另一个表的表名。

【例 5-21】 用 UPDATE 语句修改学生成绩表的数据，将表中姓名为“李兵”的同学的语文成绩改为 90，英语成绩改为 79，总分为修改后 3 门成绩的总和。

语句清单如下：

```
USE Test
UPDATE 学生成绩表 SET 语文 = 90,英语 = 79 WHERE 姓名 = '李兵'
UPDATE 学生成绩表 SET 总分 = 语文 + 数学 + 英语 WHERE 姓名 = '李兵'
```

5.4.4 删除数据

对于数据表中无用的数据，可以进行删除。SQL Server 2008 提供了两种方法删除数据表中的数据：利用可视化数据库管理工具；利用 Transact-SQL 语句。

1. 利用可视化数据库管理工具删除数据

在可视化数据库管理工具中删除数据，先按照前面章节介绍的添加数据的方法打开如图 5-19 所示窗口。单击行数据前的小方块，选中要删除的行，右击，在出现的快捷菜单中，单击“删除”命令，可以一次性删除一条记录。

2. 利用 Transact-SQL 语句删除数据

利用 Transact-SQL 提供的 DELETE 语句和 TRUNCATE TABLE 语句可以删除一条或多条甚至全部数据表记录。

1) DELETE 语句

DELETE 语句的语法格式为：

```
DELETE [ FROM ] { table_name }
```

```
    [ FROM {<table_source>} [ ,...n ] ]
    [ WHERE search_condition ]
```

【例 5-22】 用 DELETE 语句删除学生成绩表中平均分不及格的记录。

语句清单如下：

```
USE Test
DELETE 学生成绩表 WHERE 平均分< 60
```

2）TRUNCATE TABLE 语句

TRUNCATE TABLE 语句可以删除数据表中的所有数据，但是表的结构仍然保留，即该表成为一个空记录的数据表，如同新创建的表格。TRUNCATE TABLE 语句的语法格式为：

```
TRUNCATE TABLE 数据表名
```

【例 5-23】 用 TRUNCATE TABLE 语句删除学生成绩表。

语句清单如下：

```
USE Test
TRUNCATE TABLE 学生成绩表
```

5.4.5 简单查询

对数据的查询是指从数据表中检索出符合用户需求的数据，是数据表操作中最常用的一种操作。SQL Server 2008 利用 Transact-SQL 的 SELECT 语句来实现对数据的各种查询，SELECT 语句也是 Transact-SQL 中使用频率最高的一条核心语句。

SELECT 语句可以从数据库中检索行，并允许从一个或多个表中选择一个或多个行或列。虽然 SELECT 语句的完整语法较复杂，但是其主要的语法格式可归纳如下：

```
SELECT select_list
[ INTO new_table ]
FROM table_source
[ WHERE search_condition ]
[ GROUP BY group_by_expression ]
[ HAVING search_condition ]
[ ORDER BY order_expression [ ASC | DESC ] ]
```

在 5.4.2 节中已经介绍过如何使用 SELECT 语句查看数据表的全部数据，即“SELECT * FROM 数据表名”，这也是 SELECT 语句最简单的形式。因 SELECT 语句较复杂，下面将通过对语句中各子句使用方法的介绍来说明 SELECT 语句的用法。

1. SELECT 子句

SELECT 子句用来指定由查询返回的列。其语法格式为：

```
SELECT [ ALL | DISTINCT ]
    [ TOP n [ PERCENT ] [ WITH TIES ] ]
    < select_list >
```

其中，各参数意义如下：

(1) ALL：指定在结果集中可以显示重复行。ALL 是默认设置。

(2) DISTINCT：指定在结果集中只能显示唯一行。

(3) TOP *n* [PERCENT]：指定只从查询结果集中输出前 *n* 行。*n* 是介于 0 和 4 294 967 295之间的整数。如果还指定了 PERCENT，则只从结果集中输出前百分之 *n* 行。

(4) WITH TIES：指定从基本结果集中返回附加的行，这些行包含与出现在 TOP *n* (PERCENT)行最后的 ORDER BY 列中的值相同的值。如果指定了 ORDER BY 子句，则只能指定 TOP…WITH TIES。

(5) < select_list >：为结果集选择的列。选择列表是以逗号分隔的一系列表达式。

【例 5-24】 现有 Test 数据库中的学生信息表，如表 5-10 所示。

表 5-10 学生信息表

学 号	姓 名	性 别	籍 贯	进校成绩
200401	李兵	男	湖北	580
200402	王燕	女	江苏	576
200403	张强	男	湖北	569
200404	王虎	男	山东	530
200405	谭梦	女	山东	548

(1) 查询表中的所有记录。

语句清单如下：

```
USE Test
SELECT * FROM 学生信息表
```

(2) 查询表中有哪些地区的生源。

语句清单如下：

```
USE Test
SELECT DISTINCT 籍贯 FROM 学生信息表
```

返回结果如下：

```
籍贯
-----------
湖北
江苏
山东
```

(3) 查询前 2 条记录的学号、姓名和进校成绩。

语句清单如下：

```
USE Test
SELECT TOP 2 学号,姓名,进校成绩 FROM 学生信息表
```

返回结果如下：

```
学号      姓名          进校成绩
------  ----------  ----------
200401    李兵          580
200402    王燕          576
```

2. INTO 子句

SELECT 语句可以将查询的结果放入另一个新数据表中，INTO 子句就是用于创建新的数据表并将查询结果插入新表中，其语法格式为：

```
SELECT select_list
[ INTO new_table ]
FROM table_source
```

参数 new_table 用于指定新生成的数据表的名称，新生成数据表的字段由查询结果的字段决定，即为 SELECT 语句中的 select_list 指出的字段。

【例 5-25】 将学生信息表中前 3 条记录的学号、姓名、籍贯信息放入学生生源表(未建)。

语句清单如下：

```
USE Test
SELECT TOP 3 学号,姓名,籍贯 INTO 学生生源表 FROM 学生信息表
```

3. WHERE 子句

当需要查询满足一定条件的记录时，使用 WHERE 子句。WHERE 子句的格式为：

```
SELECT select_list
FROM table_source
[ WHERE search_condition ]
```

其中，search_condition 用来指定搜索条件，对搜索条件可以包含的谓词数量没有限制。搜索条件由各种运算符连接的表达式构成，这些运算符主要有比较运算符、逻辑运算符、范围运算符、列表运算符、模式匹配运算符、空值运算符等。

1) 比较运算符

比较运算符用来比较两者的大小。比较运算符包括以下运算符：>(大于)、>=(大于等于)、<(小于)、<=(小于等于)、=(等于)、<>(不等于)、!=(不等于)、!>(不大于)、!<(不小于)。

【例 5-26】 查询学生信息表中进校成绩大于 570 的学生记录。

语句清单如下：

```
USE Test
SELECT * FROM 学生信息表 WHERE 进校成绩> 570
```

返回结果如下：

```
学号      姓名          性别    籍贯          进校成绩
------  ----------  ----  ----------  ----------
200401    李兵          男      湖北          580
200402    王燕          女      江苏          576
```

2）逻辑运算符

逻辑运算符用来连接多个条件，以构成一个复杂的查询条件。逻辑运算符共有以下 3 个：

（1）NOT：逻辑非，对给定的条件取反。

（2）AND：逻辑与，只有当两个条件同时成立时，结果为真，否则结果都为假。

（3）OR：逻辑或，只有当两个条件同时不成立时，结果为假，否则结果都为真。

这 3 个运算符的运算顺序为：NOT 的优先级最高，AND 次之，OR 的优先级最低。

【例 5-27】 查询学生信息表中进校成绩大于 570 的女学生的记录。

语句清单如下：

```
USE Test
SELECT * FROM 学生信息表 WHERE 进校成绩>570 AND 性别='女'
```

或：

```
USE Test
SELECT * FROM 学生信息表 WHERE 进校成绩>570 AND NOT 性别='男'
```

返回结果如下：

```
学号      姓名          性别   籍贯          进校成绩
------  ----------  ----  ----------  ----------
200402    王燕          女     江苏          576
```

3）范围运算符

范围运算符用来判断字段值是否在/不在指定范围内。范围运算符由关键字 BETWEEN…AND 和 NOT BETWEEN…AND 组成，其语法格式为：

```
表达式 [NOT] BETWEEN 最小值 AND 最大值
```

上述格式中表达式的类型与最小值和最大值的类型要一致，如果用的是 BETWEEN…AND 关键字，则当表达式大于或等于最小值而小于或等于最大值时，结果为真，否则为假。NOT BETWEEN…AND 则刚好相反，用于搜索不在指定范围的数据。范围运算符也可以用比较运算符结合逻辑运算符代替，但相对于后者，范围运算符更简洁易用。

【例 5-28】 查询学生信息表中进校成绩在 540～570 之间的学生的记录。

语句清单如下：

```
USE Test
SELECT * FROM 学生信息表 WHERE 进校成绩 BETWEEN 540 AND 570
```

或：

```
USE Test
SELECT * FROM 学生信息表 WHERE 进校成绩>=540 AND 进校成绩<=570
```

返回结果如下：

```
学号      姓名          性别   籍贯          进校成绩
------  ----------  ----  ----------  ----------
200403    张强          男     湖北          569
200405    谭梦          女     山东          548
```

4）列表运算符

列表运算符用来判断给定的列值是否在/不在所给定的子列表中。列表运算符由 IN 和 NOT IN 组成，其语法格式为：

```
表达式 [NOT] IN (值,值,…,值 n)
```

上述格式中如果表达式的值在(IN)或不在(NOT IN)括号中所列值的范围之内，则结果为真，否则为假。

【例 5-29】 查询学生信息表中籍贯为江苏或山东的学生的记录。

语句清单如下：

```
USE Test
SELECT * FROM 学生信息表 WHERE 籍贯 IN ('江苏','山东')
```

返回结果如下：

```
学号     姓名        性别   籍贯        进校成绩
------  ----------  ----  ----------  ----------
200402   王燕        女     江苏        576
200404   王虎        男     山东        530
200405   谭梦        女     山东        548
```

5）模式匹配运算符

模式匹配运算符用来判断给定的字符型数据是否与指定的字符通配格式相符。模式匹配运算符由 LIKE 和 NOT LIKE 组成，其语法格式为：

```
表达式 [NOT] LIKE '通配符'
```

其中，通配符主要有以下 4 种：

(1) %：可匹配任意长度(包括 0)的字符串。如 a%表示以 a 开头的字符串，%a 表示以 a 结尾的字符串。

(2) _：即下划线，可匹配任何单个字符。如 a_2 表示第 1 个字符为 a，第 2 个字符任意，第 3 个字符为 2。

(3) [ ]：可匹配指定范围或集合中的任何单个字符。如 a[123]表示第 1 个字符为 a，第 2 个字符可以为 1、2、3 中的任意一个；a[1-3]也表示相同的含义。

(4) [^]：可匹配不属于指定范围或集合的任何单个字符。如 a[^1-3]表示第 1 个字符为 a，第 2 个字符不能为 1、2、3 中的任何一个。

【例 5-30】 查询学生信息表中姓王的学生的记录。

语句清单如下：

```
USE Test
SELECT * FROM 学生信息表 WHERE 姓名 LIKE '王%'
```

返回结果如下：

```
学号     姓名        性别   籍贯        进校成绩
------  ----------  ----  ----------  ----------
200402   王燕        女     江苏        576
```

```
200404    王虎          男       山东          530
```

6）空值运算符

空值运算符用来判断所指定的列值是否为空值，即 NULL 值。NULL 值通常是在列的值还未输入时的值，与 0 或空格是不同的。空值运算符由 IS NULL 和 NOT IS NULL 组成，其语法格式为：

```
表达式 [NOT] IS NULL
```

【例 5-31】 查询学生信息表中未填籍贯的学生的记录。

语句清单如下：

```
USE Test
SELECT * FROM 学生信息表 WHERE 籍贯 IS NULL
```

4. ORDER BY 子句

ORDER BY 子句用于指定对查询结果排序，但是如果在 SELECT 子句中同时指定了 TOP 范围，则 ORDER BY 无效。ORDER BY 子句语法格式如下：

```
SELECT select_list FROM table_source
[ ORDER BY { order_by_expression [ ASC | DESC ] } [ ,...n ] ]
```

其中，order_by_expression 用于指定要排序的列，ASC 指定按递增顺序排序，DESC 指定按递减顺序排序。空值被认为是最小的值。

【例 5-32】 查询学生信息表中所有男生的记录，并将查询结果按进校成绩由高到低进行排序。

语句清单如下：

```
USE Test
SELECT * FROM 学生信息表 WHERE 性别 = '男'
ORDER BY 进校成绩 DESC
```

返回结果如下：

```
学号      姓名          性别   籍贯          进校成绩
------  ----------  ----  ----------  ----------
200401    李兵          男       湖北          580
200403    张强          男       湖北          569
200404    王虎          男       山东          530
```

SELECT 还有一些其他子句，如 GROUP BY、HAVING 等，将在下节中介绍。

5.4.6 统计查询

在利用 SELECT 进行查询时，不仅仅只能获得数据表中的原始数据，还可以对这些数据进行求和、求平均值、最大值、最小值等统计操作。SQL Server 2008 在 SELECT 语句中通过聚合函数、GROUP BY 子句和 COMPUTE 子句来实现数据的统计查询。

1. 聚合函数

聚合函数用来将查询结果进行各种统计，并将满足条件的记录汇总生成一条新的记录。SQL Server 2008 提供了多种聚合函数，常用的有 SUM、AVG、MAX、MIN、COUNT 5 种聚合函数。

1）SUM 函数

SUM 函数用来统计一个数值型字段的总和。SUM 只能用于数字列，空值将被忽略。SUM 函数的语法格式为：

```
SUM ( [ ALL | DISTINCT ] expression )
```

其中，参数 ALL 表示对所有的值进行 SUM 函数运算，ALL 是默认设置；DISTINCT 用来指定 SUM 返回唯一值的和；expression 是常量、列或函数，或者是算术、按位与字符串等运算符的任意组合。

【例 5-33】 分别统计学生成绩表中语文列的总分和数学列的总分。

语句清单如下：

```
USE Test
SELECT SUM(语文), SUM(数学) FROM 学生成绩表
```

返回结果如下：

```
-----  -----
166      176
```

从上面的返回结果无法看出统计的是什么内容，如果希望返回结果上显示统计的标题，可用以下语句清单来实现：

```
USE Test
SELECT 语文总和 = SUM(语文), 数学总和 = SUM(数学) FROM 学生成绩表
```

返回结果如下：

```
语文总和   数学总和
--------  -----
166         176
```

2）AVG 函数

AVG 函数用来返回组中值的平均值，空值将被忽略，其格式除函数名外与 SUM 函数一样。

【例 5-34】 分别统计学生成绩表中语文列的平均分和数学列的平均分。

语句清单如下：

```
USE Test
SELECT 语文平均分 = AVG(语文), 数学平均分 = AVG(数学) FROM 学生成绩表
```

返回结果如下：

```
语文平均分   数学平均分
---------  -----------
    83          88
```

3）MAX 函数

MAX 函数用来返回组中值的最大值，其格式除函数名外与 SUM 函数一样。

【例 5-35】 查询学生信息表中进校成绩最高的分数。

语句清单如下：

```
USE Test
SELECT 进校成绩最高分 = MAX(进校成绩) FROM 学生信息表
```

返回结果如下：

```
进校成绩最高分
------------
580
```

4）MIN 函数

MIN 函数用来返回组中值的最小值，其格式除函数名外与 SUM 函数一样。

【例 5-36】 查询学生信息表中进校成绩最低的分数。

语句清单如下：

```
USE Test
SELECT 进校成绩最低分 = MIN(进校成绩) FROM 学生信息表
```

返回结果如下：

```
进校成绩最低分
------------
530
```

5）COUNT 函数

COUNT 函数用于统计查询结果集中记录的个数。COUNT 函数的语法格式为：

```
COUNT ( { [ ALL | DISTINCT ] expression } | * )
```

其中，* 用于统计表中所有记录的个数，COUNT(*)不需要任何参数，但不能与 DISTINCT 一起使用。

【例 5-37】 查询学生信息表中共有多少位学生的记录以及这些学生来自多少个地区。

语句清单如下：

```
USE Test
SELECT 记录总数 = COUNT( * ),地区数 = COUNT(DISTINCT 籍贯) FROM 学生信息表
```

返回结果如下：

```
记录总数   地区数
------  ------
    5        3
```

2. GROUP BY 子句

GROUP BY 子句指定用来放置输出行的组，如果 SELECT 子句的< select list >参数中

包含聚合函数,则计算每组的汇总值。GROUP BY 子句的语法格式为:

```
GROUP BY [ ALL ] group_by_expression [ ,...n ]
      [ WITH { CUBE | ROLLUP } ]
[HAVING < search_condition >]
```

上述格式中参数含义如下:

(1) group_by_expression:对其执行分组的表达式。group_by_expression 也称为分组列,它可以是列或引用列的非聚合表达式。

(2) CUBE:指定在结果集内不仅包含由 GROUP BY 提供的正常行,还包含汇总行。在结果集内返回每个可能的组和子组组合的 GROUP BY 汇总行。结果集内的汇总行数取决于 GROUP BY 子句内包含的列数。由于 CUBE 返回每个可能的组和子组组合,因此不论指定分组列时所使用的是什么顺序,行数都相同。

(3) ROLLUP:指定在结果集内不仅包含由 GROUP BY 提供的正常行,还包含汇总行。按层次结构顺序,从组内的最低级别到最高级别汇总组。组的层次结构取决于指定分组列时所使用的顺序。更改分组列的顺序会影响在结果集内生成的行数。

(4) HAVING 子句:指定组或聚合的搜索条件。HAVING 通常与 GROUP BY 子句一起使用。如果不使用 GROUP BY 子句,HAVING 的行为与 WHERE 子句一样。

【例 5-38】 统计学生信息表中来自各个地区的学生人数。

语句清单如下:

```
USE Test
SELECT 籍贯,人数 = COUNT( * ) FROM 学生信息表 GROUP BY 籍贯
```

返回结果如下:

```
籍贯        人数
------  ----------
湖北         2
江苏         1
山东         2
```

若最后还想对上述汇总结果做一个总的汇总,可在上面的 SELECT 语句后加上 WITH CUBE 或是 WITH ROLLUP,语句执行后返回结果为:

```
籍贯            人数
----------  ------------
湖北            2
江苏            1
山东            2
NULL            5
```

因籍贯列不是数值型字段,无法进行汇总,所以用 NULL 来代替汇总结果。

【例 5-39】 统计学生信息表中来自某个地区超过 1 人以上的学生人数。

语句清单如下:

```
USE Test
SELECT 籍贯,人数 = COUNT( * ) FROM 学生信息表 GROUP BY 籍贯 HAVING COUNT( * )> 1
```

返回结果如下：

```
籍贯          人数
----------  -----------
湖北          2
山东          2
```

HAVING 子句在进行统计计算后产生的结果集中，选择所需要的行。

3. COMPUTE 子句

COMPUTE 子句用来对查询结果中的记录再进行汇总，生成合计作为附加的汇总列出现在结果集的最后。COMPUTE 子句与 BY 一起使用时，在结果集内生成控制中断和分类汇总。COMPUTE 子句的格式如下：

```
COMPUTE
    { { AVG | COUNT | MAX | MIN | STDEV | STDEVP| VAR | VARP | SUM }
            ( expression ) } [ ,...n ]
[ BY expression [ ,...n ]]
```

其中，STDEV、STDEVP、VAR、VARP 是统计学上的标准差和方差函数。

COMPUTE 子句与聚合函数类似，只是它除了同聚合函数一样显示汇总结果以外，还显示参加汇总记录的详细信息；COMPUTE BY 子句与 GROUP BY 子句也具有上述类似的不同。

【例 5-40】 用 COMPUTE 子句查询学生信息表中进校成绩最高的分数。

语句清单如下：

```
USE Test
SELECT 学号,姓名,进校成绩 FROM 学生信息表 COMPUTE MAX(进校成绩)
```

返回结果如下：

```
学号        姓名         进校成绩
------  -----------  -----------
200401      李兵
200402      王燕
200403      张强
200404      王虎         530
200405      谭梦         548
                         max
                    ============
                         580
```

与例 5-33 的查询结果相比，用 COMPUTE 子句除了最后列出汇总数据外，还将查询结果的详细信息显示出来。

【例 5-41】 用 COMPUTE BY 子句统计学生信息表中来自各个地区的学生人数。

语句清单如下：

```
USE Test
SELECT * FROM 学生信息表 ORDER BY 籍贯 COMPUTE COUNT(籍贯) BY 籍贯
```

返回结果如下：

```
学号      姓名        性别  籍贯        进校成绩
------  ----------  ---  ----------  ----------
200401    李兵        男    湖北        580
200403    张强        男    湖北        569
                          cnt
                          ===========
                          2
学号      姓名        性别  籍贯        进校成绩
------  ----------  ---  ----------  ----------
200402    王燕        女    江苏        576
                          cnt
                          ===========
                          1
学号      姓名        性别  籍贯        进校成绩
------  ----------  ---  ----------  ----------
200404    王虎        男    山东        530
200405    谭梦        女    山东        548
                          cnt
                          ===========
                          2
```

5.4.7 联接查询

在前面介绍的查询方法中都只涉及对一个表的查询，但在数据库中通常存在的不只是一张孤立的表，很多时候需要对多张数据表同时进行查询，SQL Server 2008 提供的SELECT 语句通过联接查询可以同时查询多张数据表。

1. 交叉联接

交叉联接将多张表不加任何约束地组合在一起，因此又称为非限制联接。交叉联接将第 1 张表的所有记录分别与第 2 张表的每条记录组成新记录，联接后结果集的行数就是两张表的行数的乘积，列数为两张表的列数的乘积，因此交叉联接也称作笛卡儿积。交叉联接有两种语法格式：

(1) SELECT select_list FROM table_source1, table_source2

(2) SELECT select_list FROM table_source1 CROSS JOIN table_source2

第 1 种格式使用 SELECT 的普通格式，没有新的关键字；第 2 种格式功能与第 1 种格式功能相同，只是引入了 CROSS JOIN 关键字。

【例 5-42】 对 Test 数据库中的学生信息表和学生成绩表进行交叉联接查询。

语句清单如下：

```
USE Test
SELECT * FROM 学生信息表,学生成绩表
```

或：

```
USE Test
```

```
SELECT * FROM 学生信息表 CROSS JOIN 学生成绩表
```

返回结果如下：

```
学号    姓名   性别   籍贯   进校成绩  学号    姓名   语文  数学  英语  总分  平均分
------ ------ ------ ---- -------- -------- ------ ---- ---- ---- ---- ------
200401  李兵   男     湖北  580      200401  李兵   89    96    78    263   88
200401  李兵   男     湖北  580      200402  王燕   76    80    85    241   80
200402  王燕   女     江苏  576      200401  李兵   89    96    78    263   88
200402  王燕   女     江苏  576      200402  王燕   76    80    85    241   80
200403  张强   男     湖北  569      200401  李兵   89    96    78    263   88
200403  张强   男     湖北  569      200402  王燕   76    80    85    241   80
200404  王虎   男     山东  530      200401  李兵   89    96    78    263   88
200404  王虎   男     山东  530      200402  王燕   76    80    85    241   80
200405  谭梦   女     山东  548      200401  李兵   89    96    78    263   88
200405  谭梦   女     山东  548      200402  王燕   76    80    85    241   80
```

从上述结果看出，交叉联接产生的结果通常没有多少实际意义，但在数据库的数学模式上有重要的作用。

2. 内联接

内联接将两个表中满足联接条件的记录组合在一起。内联接有两种语法格式：

(1) SELECT select_list FROM table_source1, table_source2
    WHERE table_source1.column_name = table_source2.column_name

(2) SELECT select_list FROM table_source1
    [INTER] JOIN table_source2
    ON table_source1.column_name = table_source2.column_name

第1种格式使用SELECT的普通格式，没有新的关键字，将联接条件直接写在WHERE子句中，称为旧式内联接。WHERE子句中的关系表达式也可以为不等式，但通常使用等式。

第2种格式功能与第1种格式功能相同，只是引入了INTER JOIN…ON关键字，同上一格式一样，ON后的等式也可以是不等式。

【例5-43】 从Test数据库中的学生信息表和学生成绩表中查询学生的进校成绩和总分。

语句清单如下：

```
USE Test
SELECT 学生成绩表.学号,学生成绩表.姓名,进校成绩,总分
FROM 学生成绩表,学生信息表 WHERE 学生信息表.学号 = 学生成绩表.学号
```

或：

```
SELECT 学生成绩表.学号,学生成绩表.姓名,进校成绩,总分 FROM 学生成绩表
JOIN 学生信息表 ON 学生信息表.学号 = 学生成绩表.学号
```

返回结果如下：

```
学号    姓名   进校成绩   总分
------ ------ ------- -----
200401  李兵      580     263
```

```
200402   王燕        576        241
```

上述语句中，因为学号和姓名是两个表都有的字段，因此在要查询字段名前必须加上表名作为前缀以区分；如果用“*”作为查询字段列表，则结果中会显示两个数据表中所有的字段（即有两个学号和姓名）；另外学生成绩表与学生信息表中学号相同的只有两位同学，所以结果为两条记录。

3. 外联接

仅当至少有一个同属于两表的行符合联接条件时，内联接才返回行。内联接消除与另一个表中的任何不匹配的行。而外联接会返回 FROM 子句中提到的至少一个表或视图的所有行，只要这些行符合任何 WHERE 或 HAVING 搜索条件。外联接包括左向外联接、右向外联接和完整外部联接 3 种联接方式。

1）左向外联接

左向外联接的结果集包括位于 LEFT OUTER JOIN 子句左边的表（以下称为左表）的所有行，而不仅仅是联接列所匹配的行。如果左表的某行在 LEFT OUTER JOIN 子句右边的表（以下称为右表）中没有匹配行，则在相关联的结果集行中右表的所有选择列表列均为 NULL 值。左向外联接的语法格式为：

```
SELECT select_list FROM table_source1 LEFT [OUTER] JOIN table_source2
ON table_source1.column_name =  table_source2.column_name
```

【例 5-44】 采用外联接的方法查询 Test 数据库中学生信息表里所有学生的性别、籍贯、平均分和总分信息。

语句清单如下：

```
USE Test
SELECT 学生信息表.学号, 学生信息表.姓名, 籍贯,平均分,总分
FROM 学生信息表
LEFT OUTER JOIN 学生成绩表 ON 学生信息表.学号 = 学生成绩表.学号
```

返回结果如下：

```
 学号      姓名       籍贯      平均分      总分
------  --------  -------  --------  --------
200401    李兵        湖北       88         263
200402    王燕        江苏       80         241
200403    张强        湖北       NULL       NULL
200404    王虎        山东       NULL       NULL
200405    谭梦        山东       NULL       NULL
```

上述左向外联接的语句中，将学生信息表作为左表，将学生成绩表作为右表，这样在结果集中才会返回所有学生信息表中的学生；而对于那些存在于学生信息表却不存在于学生成绩表中的数据（如后 3 名学生），在结果集中则用 NULL 来代替学生成绩表中的相关字段。

2）右向外联接

右向外联接的功能刚好与左向外联接相反，在结果集中返回所有右表的行，如果右表的某些行在左表中没有匹配行，则在相关联的结果集行中左表的所有选择列表列均为 NULL

值。右向外联接的语法格式为：

```
SELECT select_list FROM table_source1 RIGHT [OUTER] JOIN table_source2
ON table_source1.column_name = table_source2.column_name
```

【例 5-45】 采用外联接的方法查询 Test 数据库中学生成绩表里所有学生的性别、籍贯、平均分和总分信息。

语句清单如下：

```
USE Test
SELECT 学生信息表.学号, 学生信息表.姓名, 籍贯,平均分,总分
FROM 学生信息表
RIGHT OUTER JOIN 学生成绩表 ON 学生信息表.学号 = 学生成绩表.学号
```

返回结果如下：

```
学号      姓名      籍贯    平均分    总分
------  --------  ------  --------  --------
200401    李兵      湖北    88        263
200402    王燕      江苏    80        241
```

上述语句与例 5-41 中的语句相比，仅仅是将 LEFT 改成了 RIGHT，其返回结果就只返回右表(学生成绩表)中的所有行，而对于左表(学生信息表)中的其他学生记录，因不符合联接条件，所以不包含在查询结果集范围之内。

3) 完整外部联接

完整外部联接的结果集返回左表和右表中的所有行。当某行在另一个表中没有匹配行时，则另一个表的选择列表列为 NULL 值；如果表之间有匹配行，则整个结果集行包含基表的数据值。完整外部联接的语法格式为：

```
SELECT select_list FROM table_source1 FULL [OUTER] JOIN table_source2
ON table_source1.column_name = table_source2.column_name
```

为了说明完整外部联接的功能，假设已在学生成绩表中追加了一条学生记录：学号：200409，姓名：刘宇，语文：67，数学：76，英语：90，总分：233，平均分：78。以后出现的其他例子中学生成绩表内都包含该记录。

【例 5-46】 采用外联接的方法查询 Test 数据库中学生成绩表和学生信息表里所有学生的性别、籍贯、平均分和总分信息。

语句清单如下：

```
USE Test
SELECT 学生信息表.学号,学生信息表.姓名,学生成绩表.学号,
学生成绩表.姓名,籍贯,平均分,总分 FROM 学生信息表
FULL OUTER JOIN 学生成绩表 ON 学生信息表.学号 = 学生成绩表.学号
```

返回结果如下：

```
学号      姓名    学号     姓名     籍贯      平均分     总分
------  -----  ------  -------  -------  -------  --------
200401    李兵    200401  李兵     湖北      88         263
```

```
200402    王燕    200402    王燕    江苏    80      241
200403    张强    NULL      NULL    湖北    NULL    NULL
200404    王虎    NULL      NULL    山东    NULL    NULL
200405    谭梦    NULL      NULL    山东    NULL    NULL
NULL      NULL    200409    刘宇    NULL    78      233
```

从上述返回结果可以看出，两张数据表在人员上存在的不一致情况。

4. 联合查询

联合查询可以将两个或多个 SELECT 语句的结果组合成一个结果集，但这些被组合的结果集必须具有相同的结构，列数必须相同，并且相应的结果集列的数据类型也必须兼容。联合查询通过 UNION 运算符来实现，其语法格式为：

```
select_statement UNION [ALL] select_statement
```

结果集中字段的名称来自第一个 SELECT 语句。在合并结果集时，默认将从最后的结果集中删除重复的行，如果不希望删除重复的行，可在语句中加上 ALL 参数。

【例 5-47】 用联合查询方法查询 Test 数据库中学生成绩表和学生信息表里所有学生的学号和姓名信息。

语句清单如下：

```
USE Test
SELECT 学号,姓名 FROM 学生信息表
UNION SELECT 学号,姓名 FROM 学生成绩表
```

返回结果如下：

```
学号      姓名
------  --------
200401    李兵
200402    王燕
200403    张强
200404    王虎
200405    谭梦
200409    刘宇
```

5.4.8 嵌套查询

嵌套查询主要用于复杂的查询中。在 SQL 语言中，一个 SELECT…FROM…WHERE 语句称为一个查询块，将一个查询块嵌套在另一个查询块的 WHERE 或 HAVING 子句条件中的查询称为嵌套查询。嵌套查询中上层的查询块称为外层查询或父查询，下层查询块称为内层查询或子查询，子查询总是写在圆括号内。

SQL 语言允许多层嵌套，但在子查询中不能出现 ORDER BY 子句。嵌套查询的处理方法遵循由里向外的原则，即先处理最内层的子查询，再层层向外处理，直到最外层查询。使用嵌套查询可以达到比较测试、集成员测试、存在性测试等目的，从而使查询功能更丰富，方法更加灵活。

1. 比较查询

通过运用比较运算符将表达式的值与子查询返回的单值进行比较，可以减少中间变量的使用。

【例 5-48】 查询 Test 数据库中学生信息表里进校成绩高于平均进校成绩的学生的姓名和进校分数。

语句清单如下：

```
USE Test
SELECT 姓名,进校成绩 FROM 学生信息表
WHERE 进校成绩>(SELECT AVG(进校成绩) FROM 学生信息表)
```

返回结果如下：

```
姓名    进校成绩
----  --------
李兵       580
王燕       576
张强       569
```

2. 批量比较查询

批量比较查询除了运用各种比较运算符进行嵌套查询外，还需要用到 ANY/ALL 运算符。批量比较查询的语法格式为：

```
测试表达式 比较运算符 [ANY|ALL](子查询)
```

带有 ANY 运算符的嵌套查询，通过比较运算符将一个表达式的值与子查询返回的一列值中的每一个进行比较，只要有一次比较的结果为真，则测试结果返回真。带有 ALL 运算符的嵌套查询，则要求每次比较的结果都为真，只要有一次比较的结果为假，则测试结果返回假。

【例 5-49】 查询 Test 数据库中学生信息表里只要比任意一位山东学生进校成绩高的学生的姓名和进校分数。

语句清单如下：

```
USE Test
SELECT 姓名,进校成绩 FROM 学生信息表 WHERE
进校成绩> ANY(SELECT 进校成绩 FROM 学生信息表 WHERE 籍贯 = '山东')
```

返回结果如下：

```
姓名     进校成绩
-----  ---------
李兵       580
王燕       576
张强       569
谭梦       548
```

【例 5-50】 查询 Test 数据库中学生信息表里比所有山东学生进校成绩都高的学生的

姓名和进校分数。

语句清单如下：

```
USE Test
SELECT 姓名,进校成绩 FROM 学生信息表 WHERE
进校成绩> ALL(SELECT 进校成绩 FROM 学生信息表 WHERE 籍贯 = '山东')
```

返回结果如下：

```
姓名     进校成绩
-----  --------
李兵      580
王燕      576
张强      569
```

3. 集成员查询

集成员查询指通过运算符 IN 或 NOT IN 将一个表达式的值与子查询返回的一列值进行比较，如果使用的是 IN 运算符，则当表达式的值与此列中的任何一个值相等时，测试结果返回真，否则返回假。如果使用的是 NOT IN 运算符，结果刚好相反。集成员查询语法格式为：

```
测试表达式 比较运算符 [IN|NOT IN](子查询)
```

【例 5-51】 查询 Test 数据库中学生成绩表里在学生信息表中有记录的学生的学号和姓名。

语句清单如下：

```
USE Test
SELECT 学号,姓名 FROM 学生成绩表
WHERE 学号 IN(SELECT 学号 FROM 学生信息表)
```

返回结果如下：

```
 学号      姓名
-----  -------
200401    李兵
200402    王燕
```

【例 5-52】 查询 Test 数据库中学生成绩表里不在学生信息表中的学生的学号和姓名。

语句清单如下：

```
USE Test
SELECT 学号,姓名 FROM 学生成绩表
WHERE 学号 NOT IN(SELECT 学号 FROM 学生信息表)
```

返回结果如下：

```
学号      姓名
-------  --------
200409     刘宇
```

4. 存在性查询

存在性查询通过关键字 EXISTS 或 NOT EXISTS 检查子查询所返回的结果集是否不为空/为空，如果使用 EXISTS 关键字，则当子查询的结果集中包含一条或多条记录时，测试结果返回真，如果结果集中不包含任何记录，测试结果返回假。存在性查询的语法格式为：

```
WHERE [NOT] EXISTS(子查询)
```

【例 5-53】 用 EXISTS 查询实现例 5-49。

语句清单如下：

```
USE Test
SELECT 学号,姓名 FROM 学生成绩表 WHERE EXISTS
(SELECT * FROM 学生信息表 WHERE 学生成绩表.学号 = 学生信息表.学号)
```

运行上述语句，会返回同例 5-49 一样的结果。

5.4.9 自动生成 SQL 语句

前面章节中介绍的在数据表中添加数据、修改数据、删除数据、查询数据等操作，除了通过直接在 SQL Server 2008 的查询分析器中输入 Transact-SQL 语句来实现外，不太复杂的操作还可以通过可视化数据库管理工具中的查询设计器自动生成相应的 SQL 语句来实现。

1. 打开查询设计器

打开可视化数据库管理工具，单击“新建查询”新建一个查询，然后在菜单栏中选择“查询”→“在编辑器中设计查询”命令，然后添加需要进行查询的数据表，就可以打开“查询设计器”窗口，如图 5-20 所示。

图 5-20 “查询设计器”窗口

2. “查询设计器”窗口组成

“查询设计器”窗口由以下 3 个窗格组成，从上至下依次为：

(1) 关系图窗格：显示正在查询的表和其他表结构化对象。每个矩形代表一个表或表结构化对象，并显示可用的数据列以及表示每列如何用于查询的图标，在矩形之间连线表示在表之间进行联接。

(2) 网格窗格：包含一个类似电子表格的网格，用户可以在其中指定选项，比如要显示哪些数据列、要选择什么行、要进行何种排序等。

(3) SQL 窗格：显示用于查询的 SQL 语句。可以对设计器所创建的 SQL 语句进行编辑，也可以输入自己的 SQL 语句。对于不能用关系图窗格和网格窗格创建的 SQL 语句(如联合查询)，输入 SQL 语句尤其有益。

3. 查询设计器的使用方法

可以在查询设计器的任意窗格内进行操作以创建查询：可以通过在关系图窗格中选择某列，将该列输入到网格窗格中，或者使其成为 SQL 窗格中 SQL 语句的一部分等方法，指定要显示的列。关系图窗格、网格窗格和 SQL 窗格都是同步的，当在某一窗格中进行更改时，其他窗格自动反映所做的更改。

要进行多个表的联接操作，可以在关系图窗格中右击，在出现的快捷菜单中单击“添加表”命令，系统将列出该数据表所在数据库的所有数据表供用户选择。如果关系窗格中有多余的表格，可在该表格标题栏上右击，在出现的快捷菜单中单击“删除”命令，将其从关系图窗格中删除。

通过在 3 个子窗口中右击并选择“更改类型”，可以改变 SQL 语句的类型(查询、插入、更新、删除等)。例如选择“更新”，SQL 语句就会由 SELECT 语句变为 UPDATE 语句，不同类型的查询类型，看到的网格窗格中的列名也不同。

通过各个窗格共同设计的 SQL 语句完成后，可先在 SQL 窗格中右击，选择验证 SQL 语法，系统将自动检验 SQL 语句在语法上的正确性。设计的 SQL 语句通过语法正确性验证后，单击“确定”按钮，系统会将 SQL 语句自动添加到新建的查询中，执行生成的查询语句即可得到结果。

查询设计器的各个窗格还提供了“添加分组依据”、“排序”、“筛选”以及“显示/隐藏”等功能供用户使用。

【例 5-54】 用查询设计器自动生成以下 SQL 语句：

```
SELECT 学号,姓名, 进校成绩 FROM 学生信息表
WHERE 进校成绩> 570 AND 性别 = '女' ORDER BY 学号 DESC
```

在查询设计器中应按以下步骤进行：

(1) 设置要操作的数据表。右键选择“添加表”，然后添加学生成绩表和学生信息表。

(2) 设置查询类型。确认当前查询类型为“查询”，如果不是，按照前面介绍的方法，设置当前查询类型为“查询”。

(3) 设置要输出的字段。单击网格窗格列标题为“列”的第 1 行，在出现的下拉列表中

选择“学号”，系统自动在该行的“输出”列显示选中标志，表示该字段为要输出的字段，可以通过修改“输出”标志来确定该字段是否输出。在此列的第 2 行、第 3 行用相同的方法设置“姓名”和“进校成绩”为要输出的字段。

(4) 设置查询条件。在网格窗格第 3 行(“进校成绩”所在行)的“筛选器”列输入“>570”；单击网格窗格列标题为“列”的第 4 行，在出现的下拉列表中选择“性别”，在“筛选器”列输入“='女'”，即将查询条件设置完毕。因上述两个条件之间是 AND 关系，所以写在同一列，如果条件之间是 OR 关系，则将条件写在“或”列。

(5) 设置排序类型。单击网格窗格第 1 行(“学号”所在行)的“排序类型”列，在出现的下拉列表中选择“降序”。

(6) 上述设置完成后，网格窗格应如图 5-21 所示。SQL 窗格应出现上述要求实现的 SQL 语句。

	列		表	输出	排序类型	排序顺序	筛选器
	学号		学生成绩表	☑	降序	1	
	姓名		学生成绩表	☑			
	进校成绩		学生信息表	☑			
	总分		学生成绩表	☐			>570
	性别		学生信息表	☐			='女'

图 5-21 设置完成后的网格窗格

(7) 验证 SQL 语句的语法正确性。这一步不是必须的，但是为了减少运行时出现的错误，最好先执行该步再运行。在 SQL 窗格右击并选择“验证 SQL 语法”，验证 SQL 语句语法正确性。

(8) 运行 SQL 语句。单击“确定”按钮，系统会将生成的 SQL 语句写入查询窗口中，运行查询就可以获得结果。

5.5 SQL Server 2008 视图和索引

5.5.1 视图概述

1. 视图的概念

视图是从一个或多个数据表中导出的虚拟表，其内容由查询定义。同真实的表一样，视图包含一系列带有名称的列和行数据。但是，视图并不在数据库中以存储的数据值集形式存在，行和列的数据来自由定义视图的查询所引用的表(通常称为基表)，并且只在引用视图时才动态生成这些数据。

对其中所引用的基表来说，视图的作用类似于筛选。定义视图的筛选可以来自当前或其他数据库的一个或多个表，或者其他视图。通过视图进行查询没有任何限制，通过它们进行数据修改时的限制也很少。使用视图可以实现下列功能：

(1) 将用户限定在表中的特定行上。例如，只允许雇员看见工作跟踪表内记录其工作的行。

(2) 将用户限定在特定列上。例如，对于那些不负责处理工资单的雇员，只允许他们看

见雇员表中的姓名列、办公室列、工作电话列和部门列，而不能看见任何包含工资信息或个人信息的列。

(3) 将多个表中的列联接起来，使它们看起来像一个表。

(4) 聚合信息而非提供详细信息。例如，显示一个列的和，或列的最大值和最小值。

2. 视图的优点

与直接使用表相比，使用视图有很多优点，具体体现在以下几个方面：

(1) 简化用户对数据库的操作。视图可以屏蔽数据的复杂性，用户不用了解数据库的真实结构，就可以方便地使用和管理数据。在创建视图时，可以把经常使用的查询语句定义为视图，这样在每次执行相同的查询时，不用重新编写查询语句，只要使用一条简单的查询视图语句就可以实现相同的功能。

(2) 减少数据库中无用数据的干扰。视图可以使用户只关心自己感兴趣的某些特定数据信息，而对于那些暂时不需要或无用的数据在视图中可以不让其显示出来，从而减少数据库中无用数据对用户的干扰。

(3) 重新组织数据。视图可以对数据表中的数据进行水平或垂直分割，如果直接分割数据表，会改变数据表的结构，可能引起应用程序的执行错误；而使用视图进行分割，因为产生的是一张虚拟表，并没有对真实的数据表进行修改，不会产生上述错误。

(4) 提供简单有效的安全保护功能。通过定制不同用户对数据的访问权限，可以使视图用户只能查看和修改他们所能看到的数据，其他数据库或者表既看不见也不可访问。视图所引用表的访问权限和视图权限的设置互不影响。

5.5.2 创建视图

在 SQL Server 2008 中创建视图主要有以下限制：

(1) 只能在当前数据库中创建视图。

(2) 视图名称必须遵循标识符的规则，且对每个用户必须为唯一。此外，该名称不得与该用户拥有的任何表的名称相同。

(3) 可以在其他视图和引用视图的过程之上建立视图。允许嵌套 32 级视图。

(4) 不能将规则或 DEFAULT 定义与视图相关联。

(5) 定义视图的查询不可以包含 ORDER BY、COMPUTE 或 COMPUTE BY 子句或 INTO 关键字。

(6) 不能在视图上定义全文索引定义。

(7) 不能创建临时视图，也不能在临时表上创建视图。

在 SQL Server 2008 中主要有两种方法创建视图：使用图形化工具；使用 Transact-SQL 语句。

1. 使用图形化工具创建视图

例如为数据库 Test 创建一个视图，要求连接表 Students 和表 Courses。操作步骤如下：

(1) 打开 Microsoft SQL Server Management Studio 窗口。

(2) 选中数据库 Test 并展开,右击“视图”并选择“新建视图”,弹出“添加表”窗口,如图 5-22 所示。

(3) 在弹出的“添加表”窗口中可以看到,视图的基表可以是表,也可以是视图、函数和同义词。在表中选择表 Students 和表 Courses,如图 5-22 所示。

图 5-22 创建视图并添加表

(4) 单击“添加”按钮,如果还需要添加其他表,则可以继续选择添加基表;如果不再需要添加,则可以单击“关闭”按钮,关闭“添加表”窗口。

(5) 在视图窗口的“关系图窗格”中,显示了表 Students 和表 Courses 的全部列信息,在此可以选择视图查询的列,比如选择表 Students 中的列 Sno、Sname 和表 Courses 中的列 Cno、Cname、score,对应地,在“条件窗格”中就列出了选择的列。在“显示 SQL”空格中显示了两表的连接语句,表示了这个视图包含的数据内容,可以单击“执行 SQL”按钮 ,在“显示结果窗格”中显示查询出的结果集,如图 5-23 所示。

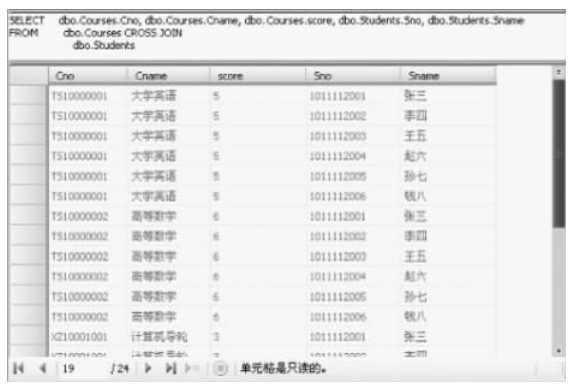


图 5-23 设置定义视图的查询条件

单击“保存”按钮,在弹出的“选择名称”窗口中输入视图名称“Students_view”,单击“确定”按钮,就可以看到“视图”节点下增加了一个视图 Students_view。

2. 利用 Transact-SQL 创建视图

Transact-SQL 提供了 CREATE VIEW 语句来创建视图,其语法格式为:

```
CREATE VIEW [ schema_name ] view_name [ (column [ ,...n ] ) ]
 [ WITH <view_attribute> [ ,...n ] ]
AS
 select_statement
[ WITH CHECK OPTION ]
<view_attribute> ::=
    { [ ENCRYPTION ]  [ SCHEMABINDING ] [ VIEW_METADATA ]  }
```

上述创建视图语句中各参数的意义简要介绍如下，更详细说明请参照联机丛书。

(1) schema_name：视图所属结构名。

(2) view_name：视图名。

(3) column：视图中的列名。对于视图中所使用的列名，一般只有列是从算术表达式、函数或常量派生出来的或者列的指定名称不同于来源列的名称时，才需要使用。

(4) select_statement：定义视图的 SELECT 语句。该语句可以使用多个表或其他视图。对于视图定义中的 SELECT 子句有几个限制。CREATE VIEW 语句不能包含 COMPUTE 或 COMPUTE BY 子句；不能包含 ORDER BY 子句，除非在 SELECT 语句的选择列表中也有一个 TOP 子句；不能包含 INTO 关键字；不能引用临时表或表变量。

(5) WITH CHECK OPTION：强制针对视图执行的所有数据修改语句都必须符合在 select_ statement 中设置的条件。

(6) ENCRYPTION：表示 SQL Server 加密包含 CREATE VIEW 语句文本的系统表列。

(7) SCHEMABINDING：将视图绑定到结构上。指定 SCHEMABINDING 时，select_ statement 必须包含所引用的表、视图或用户定义函数的两部分名称，即所有者和对象名称(owner. object)。

(8) VIEW_METADATA：指定为引用视图的查询请求浏览模式的元数据时，SQL Server 将向 DBLIB、ODBC 和 OLE DB API 返回有关视图的元数据信息。

视图一旦建立好后，可以像操作其他数据表一样对视图进行各种数据操作。

【例 5-55】 用 CREATE VIEW 语句创建名为 example_view 的视图，要求基表为：Test 数据库的学生信息表(Students)；在视图中显示学号(Sno)、姓名(Sname)、年龄(age)字段；视图查询的数据为所有男同学的信息。

语句清单如下：

```
USE Test
GO
CREATE VIEW example_view
AS SELECT     Sno, Sname, age
FROM          dbo.Students
WHERE     (sex = 'B')
```

在查询分析器中，CREATE VIEW 语句必须为查询的第一条执行语句，因此在打开数据库后，加上 GO 命令，让其立即执行，再执行 CREATE VIEW 语句。上述 SQL 语句将查询结果放在视图 example_view，因此在查询分析器的结果窗格中，没有任何输出结果。要查看 example_view 视图的内容，可以像查看一般数据表一样，用以下语句实现：

```
SELECT * FROM example_view
```

返回结果如图 5-24 所示。

5.5.3 查看视图

SQL Server 允许用户获得视图的一些有关信息，如视图的名称、视图的所有者、创建时

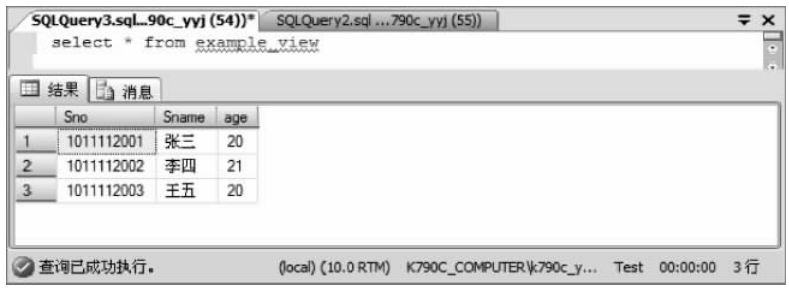


图 5-24 执行结果图示

间、视图的定义文本等。视图的信息存放在以下几个 SQL Server 系统表中：

(1) Sysobjects：存放视图名称等基本信息。

(2) Syscolumns：存放视图中定义的列。

(3) Sysdepends：存放视图的依赖关系。

(4) Syscomments：存放定义视图的文本。

可以通过企业管理器和 Transact-SQL 语句提供的系统存储过程来查看视图的各种信息。

1. 利用企业管理器查看视图

(1) 按照前面介绍的方法在企业管理器的树状窗格中，打开要查看视图所在的数据库节点，选中数据库节点下的“视图”节点，在右边的窗格中会列出所选数据库中的所有视图。

(2) 要查看视图的基本信息，可以在右边窗格要查看视图的名称上右击，在出现的快捷菜单中单击“属性”命令，或直接双击要查看的视图名，打开“查看属性”对话框。在该对话框中可以查看视图名称、创建日期、SQL 语句以及该视图的权限等信息。

(3) 要查看视图的相关性信息，可以在右边窗格要查看视图的名称上右击，在出现的快捷菜单中依次选择“所有任务”→“显示相关性”命令，打开“相关性”对话框。在该对话框的左、右窗格中分别显示了依赖于当前视图的对象以及当前视图依赖的对象。

(4) 要查看视图的数据信息，可以在右边窗格要查看视图的名称上右击，在出现的快捷菜单中依次选择“打开视图”→“返回所有行”命令，在打开的查询设计器结果窗格中会显示视图中所包含的数据信息。

2. 利用 Transact-SQL 存储过程查看视图

1) 查看视图的基本信息

在企业管理器中可以查询视图的基本信息。可以使用系统存储过程 sp_help 来显示视图的名称、拥有者、创建时间等信息。例如，查看视图 Students_view 的基本信息，可以使用如下语句：

```
sp_help Students_view
```

执行上述语句后，显示结果如图 5-25 所示。

2) 查看视图的文本信息

如果视图在创建或修改时没有被加密，那么可以使用系统存储过程 sp_helptext 来显示

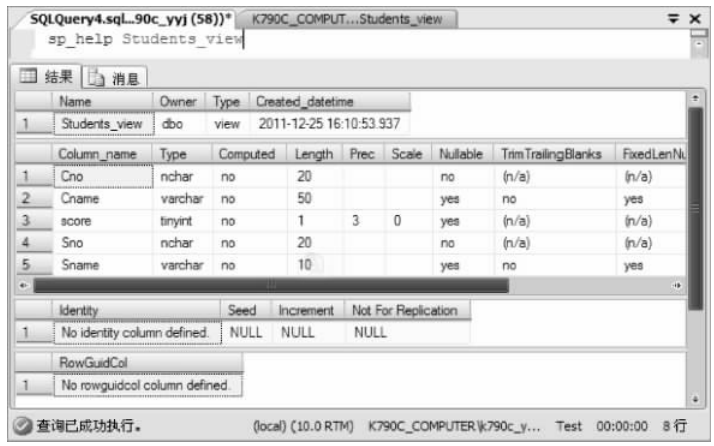


图 5-25 Students_view 的基本信息

视图定义的语句，否则，如果视图被加密，那么连视图的拥有者和系统管理员都无法看到它的定义。例如，查看视图 Students_view 的文本信息，可以使用如下语句：

```
sp_helptext Students_view
```

执行上面语句后，显示 Students_view 视图的文本信息如图 5-26 所示。

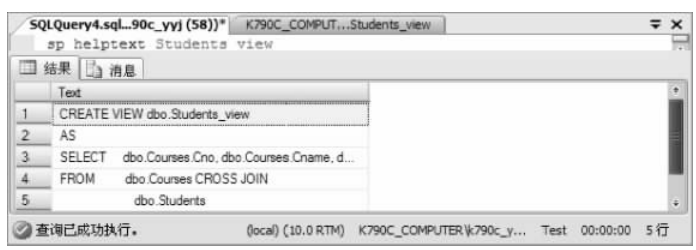


图 5-26 Students_view 视图的文本信息

如果查看的视图已被加密，则会返回该视图被加密的信息。

3）查看视图的依赖关系

有时候需要查看视图与其他数据库对象之间的依赖关系，比如视图在哪些表的基础上创建、又有哪些数据库对象的定义引用了该视图等。可以使用系统存储过程 sp_depends 查看。例如查看 Students_view 视图的依赖关系可以使用如下语句：

```
sp_depends Students_view
```

执行上面语句后，返回结果如图 5-27 所示。

5.5.4 修改视图

在查看视图的相关信息后，如果发现生成的视图不符合要求，可以进行修改。通过企业管理器和 Transact-SQL 语句可以达到修改视图的目的。

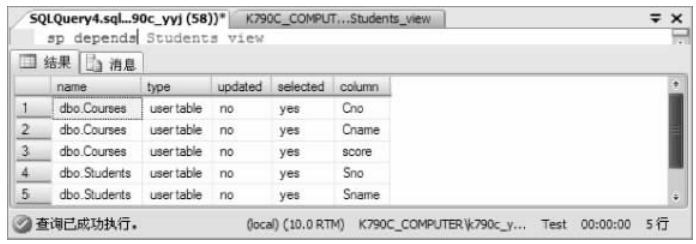


图 5-27 Students_view 的依赖关系

1. 利用企业管理器修改视图

用创建视图的方法修改视图。例如,使用图形工具修改视图数据库 Test 中的一个视图 Students_view,操作步骤如下:

(1) 在 SQL Server Management Studio 中,展开数据库 Test,再展开“视图”节点。

(2) 右击 Students_view 视图,从快捷菜单中选择相应的命令。这里可以选择“设计”命令,如图 5-28 所示。

图 5-28 修改视图

2. 利用 Transact-SQL 语句修改视图

使用 Transact-SQL 语句修改视图的定义需要使用 ALTER VIEW 语句。ALTER VIEW 语句的语法与 CREATE VIEW 的语法类似,其语法格式如下:

```
ALTER VIEW [ schema_name ] view_name [ (column [ ,...n ] ) ]
 [ WITH <view_attribute> [ ,...n ] ]
AS
 select_statement
[ WITH CHECK OPTION ]
<view_attribute> ::=
    { [ ENCRYPTION ]  [ SCHEMABINDING ] [ VIEW_METADATA ]  }
```

例如，需要修改所建视图 Students_view，使其只显示课程的编号和名称，并且不显示课程的学分，可以使用如下语句：

```
alter view Students_view
as
SELECT      dbo.Courses.Cno, dbo.Courses.Cname, dbo.Students.Sno, dbo.Students.Sname
FROM          dbo.Courses CROSS JOIN
                    dbo.Students
go
select * from Students_view
go
```

执行语句后，使用 SELECT 语句查询 Students_view，显示结果如图 5-29 所示。可以发现该视图不再显示课程学分信息。

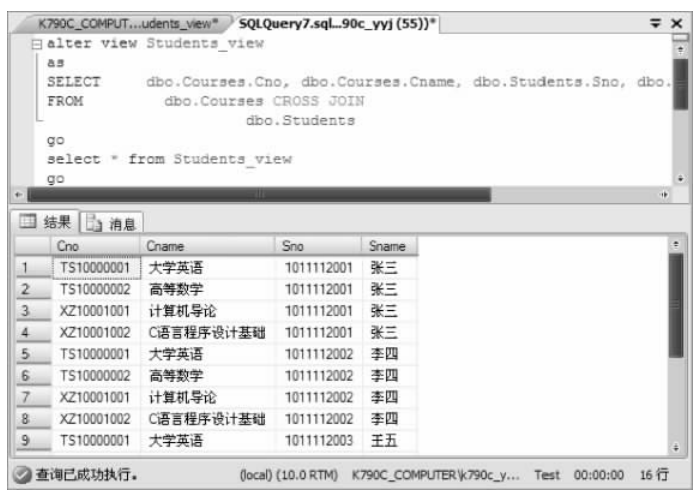


图 5-29 修改后的 Students_view 视图显示结果

【例 5-56】 对在例 5-55 中已经生成的 example_view 视图进行修改：显示的字段增加性别(sex)字段。

语句清单如下：

```
USE Test
GO
ALTER VIEW example_view AS SELECT Sno,Sname,age,sex
FROM Students
WHERE sex = 'B'
```

5.5.5 使用视图

在使用视图时，直接运行视图可以得到查询的结果数据集，这是视图最常用的功能。除此之外，在一定的条件下，还可以利用视图进行数据表数据的添加、修改和删除。由于视图本身是一张虚拟的数据表，并没有真正存储在磁盘上，因此对视图进行上述操作时，所做的改动都体现在生成视图的基表上。

1. 使用视图进行基表数据更改的条件

(1) 生成视图的 SELECT 语句不能包含计算字段。

(2) 生成视图的 SELECT 语句不能使用各种统计函数。

(3) 生成视图的 SELECT 语句不能使用 TOP、GROUP BY、DISTINCT 等子句。

(4) 如果视图的引用来自于多个基表,则当使用 UPDATE 语句修改视图中的数据时,不能同时影响多个表中的数据变化。

2. 使用视图进行基表数据更改的方法

符合上述条件的视图,可以通过对视图数据的添加、修改和删除操作,达到对基表数据进行更改的目的,其方法和 5.4 节中介绍的对数据表的操作相同,只要将数据表名换成视图名即可。

【例 5-57】 通过对 example_view 视图的操作删除学生信息表中所有男生信息。

语句清单如下:

```
USE Test
DELETE FROM example_view
```

因为在例 5-56 中修改后的 example_view 视图中存放的就是学生信息表中所有男生的信息,因此将 example_view 视图中的数据全部删除,就达到将学生信息表中所有男生信息删除的目的。

5.5.6 删除视图

对于那些不再有用的视图,可以将其删除。通过图形化操作工具或 Transact-SQL 语句都能达到删除数据表的目的。

1. 利用图形化操作工具删除视图

例如,使用图形工具修改和删除视图数据库 Test 中的一个视图 Students_view,操作步骤如下:

(1) 在 SQL Server Management Studio 中,展开数据库 Test,再展开“视图”节点。

(2) 右击 Students_view 视图,在右键菜单中选择“删除”命令,在打开的窗口单击“确定”按钮,即可完成删除操作。

2. 利用 Transact-SQL 语句删除视图

如果视图不再需要了,通过执行 DROP VIEW 语句,可以把视图的定义从数据库中删除。删除一个视图,就是删除其定义和赋予它的全部权限。删除一个表并不能自动删除引用该表的视图,因此,视图必须明确地删除。在 DROP VIEW 语句中,可以同时删除多个不再需要的视图。

DROP VIEW 语句的基本语法格式如下所示:

```
DROP VIEW view_name
```

删除一个视图后，虽然它所基于的表和数据不会受到任何影响，但是依赖于该视图的其他对象或查询将会在执行时出现错误。

与使用 alter 语句修改视图不同的是，删除视图后必须重新指定视图的权限，而修改视图，视图原来的权限不会发生变化。

【例 5-58】 用 DROP VIEW 语句删除 example_view。

语句清单如下：

```
USE Test
DROP VIEW example_view
```

5.5.7 索引概述

1. 索引的概念

可以利用索引快速访问数据库表中的特定信息。索引是一种特殊类型的数据库对象，是对数据库表中一个或多个列的值进行排序的结构。索引有助于更快地获取信息。

索引提供指针以指向存储在表中指定列的数据值，然后根据指定的排序次序排列这些指针。数据库使用索引的方式与使用书的目录很相似：通过搜索索引找到特定的值，然后跟随指针到达包含该值的行。

在数据库关系图中，可以为选定的表创建、编辑或删除索引。当保存附加在此索引上的表或包含此表的数据库关系图时，索引同时被保存。

通常情况下，只有当经常查询索引列中的数据时，才需要在表上创建索引。索引将占用磁盘空间，并且降低添加、删除和更新行的速度。不过在多数情况下，索引所带来的数据检索速度的优势大大超过它的不足之处。然而，如果应用程序非常频繁地更新数据，或磁盘空间有限，那么最好限制索引的数量。

在创建索引前，必须确定要使用哪些字段作为索引，通常建立索引选择字段的一般原则是：

(1) 对经常用来查询的字段建立索引。

(2) 对数据表中的主键建立索引。

(3) 对数据表中的外键建立索引。

(4) 对经常用于联接的字段建立索引。

2. 索引的类型

在 SQL Server 中只有两种类型的索引(聚集和非聚集)，但从内部角度上来说，有 3 种不同的类型：

(1) 聚集索引。

(2) 堆上的非聚集索引。

(3) 聚集索引上的非聚集索引。

索引在聚集表或者在堆上创建。

聚集表是具有聚集索引的表。聚集表中的数据以指定顺序存储。

堆是不含有任何聚集索引的表。它基于组合该行的扩展盘区、页和行偏移(放置在最顶

端)而建立唯一标识或行 ID。只在没有可用簇键的情况下(非聚集索引)才需要行 ID。

除了从内部角度考虑之外,在 SQL Server 2008 中,还有以下两种特殊的索引:全文索引和 XML 索引。

1) 聚集索引

聚集索引对于特定表是唯一的,一张表只能有一个聚集索引。聚集索引基于数据行的键值在表内排序和存储这些数据行。每个表只能有一个聚集索引,因为数据行本身只能按一个顺序存储。

聚集索引的特殊之处在于叶层聚集索引,就是实际数据,也就是说,数据根据索引排序标准重新排序,然后以相同物理顺序存储。

如果需要在索引结构的中间插入记录,就会发生标准页拆分。原有页的最后一半记录被移到一个新页上,并在新页或原有页的合适位置插入新记录。如果新记录逻辑上处于索引结构的末尾,就创建新页,但只有新记录被添加到新页中,如图 5-30 所示。

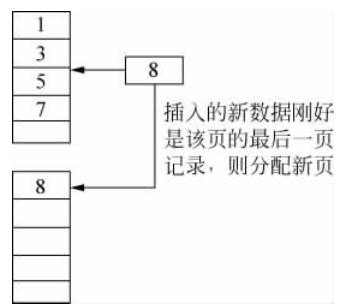


图 5-30 新记录被添加到新页中

2) 堆上的非聚集索引

堆上的非聚集索引与聚集索引的工作方式相似。但是,它们之间存在一些明显的区别。叶级不是数据,相反,它是一个可从中获得指向该数据的指针的级别。该指针以 RID 的形式出现,RID 由索引指向的特定行的区段、页以及行偏移量构成。即使叶级不是实际数据,使用叶级也仅仅比使用群集索引多一个步骤。因为 RID 具有行的位置的全部信息,所以可以直接到达数据。

但是不要误解“只多一步”意味着只存在细微差别,堆上的非聚集索引运行的速度与聚集索引的速度一样快。就聚集索引而言,数据以索引的物理存放。这意味着对于一定范围的数据,当发现该行拥有数据头时,就有好机会,因为该页的其他行也有数据头(这不是说,因为数据和数据头存储在一起,所以从物理上已经得到了下一条记录)。就堆而言,数据除了以索引的方式链接以外,不再以其他方式链接。从物理角度看,这绝对不是一种排序。这意味着,从物理读的角度看,系统已经从所有的文件中提取了记录。实际上,极有可能(或者更有可能)要从相同页上分多次提取数据。

3) 聚集表上的非聚集索引

聚集表上的非聚集索引仍然有许多相似性,但也有差别。就像堆上的非聚集索引,索引的非叶层看上去与聚集索引相似。只有等到了叶层,才能看见差别。

在叶层,聚集表上的非聚集索引与其他两种索引结构相比,简直是天壤之别。对于聚集索引,当到达叶层时,就可以找到实际数据。对于堆上的非聚集索引,还没有找到真实数据,但找到了直接访问数据的标识符(只多了一步)。对于聚集表上的非聚集索引,找到了簇键,也就是,找到了足够的信息,并且可以利用聚集索引。

这里是两种完全不同的查询方法。

在关系表的例子中,从一定范围的查询开始,在索引中进行了单个查询,而且浏览非聚集索引并能找到满足标准的数据连续范围(LIKE 'T%')。对于这种形式的查找,能立即进入索引中的特定场所,这种查找方法称为搜索(Seek)。

第 2 种查询是利用聚集索引。这种形式的查询速度非常快,问题是必须多次发生。可以看到,SQL Server 从第一个查询的索引中提取清单(所有名字都以“T”字开头的清单),但这个清单与连续模式的簇键在逻辑上并不匹配,需要单独查找每条记录。

4) 全文索引

全文索引是一种特殊类型的基于标记的功能性索引,由 Microsoft SQL Server 全文引擎(MSFTESQL)服务创建和维护。全文索引是基于索引文本中的各个标记来创建倒排、堆积且压缩的索引结构。

创建和维护全文索引的过程称为索引填充。Microsoft 支持下列全文索引填充:

(1) 完全填充。完全填充一般发生在首次填充全文目录或全文索引时。随后可以使用更改跟踪填充或增量填充来维护这些索引。

(2) 基于更改跟踪的填充。对基于更改跟踪的填充,SQL Server 会记录在设置了全文索引的表中修改过的行。这些更改会被传播到全文索引。

更改跟踪填充要求对相应的全文进行初步填充。若要自动将更改传播到全文索引,则应在 CREATE FULLTEXT INDEX 语句中使用 AUTO 选项。通过指定 MANUAL 选项,可以按计划手动传播更改或通过使用 SQL Server 代理来传播更改,也可以自动手动传播更改。

更改跟踪需要少量的开销。如果不希望 SQL Server 跟踪更改,则应使用 CHANGE TRACKING OFF 选项。

(3) 基于增量时间戳的填充。对基于增量时间戳的填充,增量填充会在全文索引中更新上次填充的当时或之后添加、删除或修改的行。增量填充要求索引表必须具有 timestamp 数据类型的列,如果时间戳列不存在,则无法执行增量填充。对不含时间戳列的表请求增量填充会导致完全填充操作。

如果影响表全文索引的任意元数据自上次填充以来发生了变化,增量填充请求将作为完全填充来执行。这包括更改任意列、索引或全文索引的定义。

填充结束时,SQL 收集器会记录一个新的时间戳值。此值等于 SQL 收集器所观察到的最大时间戳值。以后再启动增量填充时,就会使用此值。

5) XML 索引

XML 索引是 SQL Server 2005 中的新增功能。XML 是相对非结构化的数据。它利用标记来标识数据,并且可以与模式关联,给基于 XML 的数据提供类型或验证信息。XML 的非结构化特性需要“导航”或者“路径”信息的概念,以在 XML 文档中查找数据“节点”。在另一方面,索引尝试提供数据的特定结构和顺序,这在一定程度上有冲突。XML 索引分为主 XML 索引和辅助 XML 索引,如表 5-11 所示。

表 5-11 XML 索引

名 称	功 能
主 XML 索引	在 XML 索引上创建的第一个索引必须声明为主索引。当创建主索引时,SQL Server 创建一个新的群集索引,这个群集索引将基表的群集索引与来自任何指定的 XML 节点的数据组合在一起
辅助 XML 索引	这里没有任何特别之处,非常类似于指向群集索引的群集键的非群集索引,辅助 XML 索引以很相似的方法指向主 XML 索引。一旦创建了主 XML 索引,就能在 XML 列上创建多达 248 个以上的 XML 索引

在 SQL Server 中，可以在类型为 XML 的列上创建索引。这样做的主要要求如下：

(1) 在包含需要索引的 XML 表上必须具有群集索引。

(2) 在创建辅助索引之前(稍后将有更多介绍)，必须先在 XML 数据列上创建主 XML 索引。

(3) XML 索引只能在 XML 类型的列上创建(而且 XML 索引是可以在该类型的列上创建的唯一一种索引)。

(4) XML 列必须是基表的一部分，不能在视图上创建索引。

5.5.8 创建索引

在 SQL Server 2008 中，有些索引是系统自动建立的，例如在创建数据表时，如果设置了主键或唯一性约束，系统就会自动根据这些字段建立主键索引或唯一索引。如果用户要自己创建索引，SQL Server 2008 主要提供了两种方法：使用图形化工具；利用 Transact-SQL 语句。

1. 使用图形工具创建索引

下面为数据库 Test 中的表 Student 创建一个不唯一性的非聚集索引 Students_name_index，操作步骤如下：

(1) 在 SQL Server Management Studio 中，依次展开左边树结构，“服务器”→“数据库”→Test→“表”→Students，右击“索引”节点，在弹出的菜单中选择“新建索引”命令。

(2) 在“新建索引”窗口的“常规”页面，可以配置索引的名称、选择索引的类型、是否是唯一索引等，如图 5-31 所示。

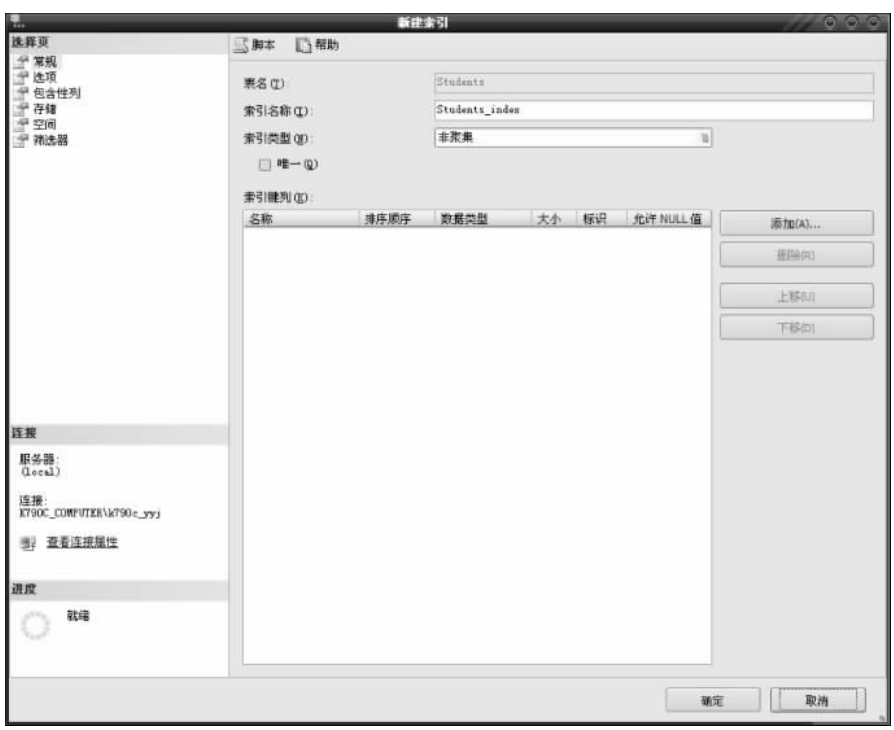


图 5-31 新建索引

(3) 单击“添加”按钮，打开“从 dbo. Students 中选择列”窗口，在窗口中的“表列”列表中启用 Sname 复选框，如图 5-32 示。

(4) 单击“确定”按钮，返回“新建索引”窗口，然后再单击“新建索引”窗口的“确定”按钮，“索引”节点下便生成了一个名 Students_name_index 的索引，如图 5-33 所示，说明该索引创建成功。

图 5-32　选择索引列

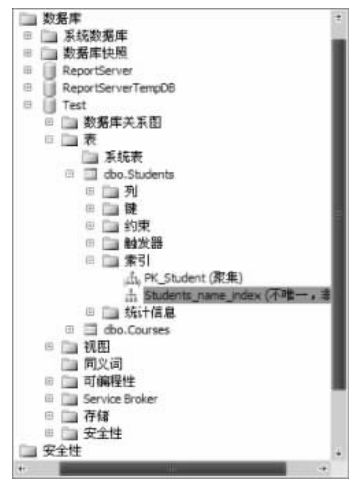


图 5-33　创建好的索引

2. 利用 Transact-SQL 语句创建索引

使用 Transact-SQL 提供的 CREATE INDEX 语句来创建索引，是最基本的索引创建方式，并且这种方法最具有适应性，可以创建出符合自己需要的索引。在使用这种方式创建索引时，可以使用许多选项，例如指定数据页的充满度、进行排序、整理统计信息等，从而优化索引。使用这种方法，可以指定索引类型、唯一性、包含性和复合性，也就是说，既可以创建聚集索引，也可以创建非聚集索引；既可以在一个列上创建索引，也可以在两个或两个以上的列上创建索引。

在 Microsoft SQL Server 2008 系统中，使用 CREATE INDEX 语句可以在关系表上创建索引，其基本的语法形式如下：

```
CREATE [UNIQUE] [CLUSTERED] [NONCLUSTERED] INDEX index_name
ON table_or_view_name (colum [ASC | DESC] [,...n])
[INCLUDE (column_name[,...n])]
[WITH
(   PAD_INDEX = {ON | OFF}
|   FILLFACTOR = fillfactor
|   SORT_IN_TEMPDB = {ON | OFF}
|   IGNORE_DUP_KEY = {ON | OFF}
|   STATISTICS_NORECOMPUTE = {ON | OFF}
|   DROP_EXISTING = {ON | OFF}
|   ONLINE = {ON | OFF}
|   ALLOW_ROW_LOCKS = {ON | OFF}
```

```
| ALLOW_PAGE_LOCKS = {ON | OFF}
| MAXDOP = max_degree_of_parallelism)[,...n]]
  ON {partition_schema_name(column_name) | filegroup_name | default}
```

上述创建索引语句中各主要参数的意义简要介绍如下，更详细说明请参照联机丛书。

(1) UNIQUE：该选项表示创建唯一性的索引，在索引列中不能有相同的两个列值存在。

(2) CLUSTERED：该选项表示创建聚集索引。

(3) NONCLUSTERED：该选项表示创建非聚集索引。这是 CREATE INDEX 语句的默认值。

(4) 第一个 ON 关键字表示索引所属的表或视图，这里用于指定表或视图的名称和相应的列名称。列名称后面可以使用 ASC 或 DESC 关键字，指定是升序还是降序排列，默认值是 ASC。

(5) INCLUDE：该选项用于指定将要包含到非聚集索引的叶级中的非键列。

(6) PAD_INDEX：该选项用于指定索引的中间级页，也就是说为非叶级索引指定填充度。这时的填充度由 FILLFACTOR 选项指定。

(7) FILLFACTOR：该选项用于指定叶级索引页的填充度。

(8) SORT_IN_TEMPDB：该选项为 ON 时，用于指定创建索引时产生的中间结果，在 tempdb 数据库中进行排序。为 OFF 时，在当前数据库中排序。

(9) IGNORE_DUP_KEY：该选项用于指定唯一性索引键冗余数据的系统行为。当为 ON 时，系统发出警告信息，违反唯一性的数据插入失败。为 OFF 时，取消整个 INSERT 语句，并且发出错误信息。

(10) STATISTICS_NORECOMPUTE：该选项用于指定是否重新计算分发统计信息。为 ON 时，不自动计算过期的索引统计信息。为 OFF 时，启动自动计算功能。

(11) DROP_EXISTING：该选项用于是否可以删除指定的索引，并且重建该索引。为 ON 时，可以删除并且重建已有的索引。为 OFF 时，不能删除重建。

(12) ONLINE：该选项用于指定索引操作期间基础表和关联索引是否可用于查询。为 ON 时，不持有表锁，允许用于查询。为 OFF 时，持有表锁，索引操作期间不能执行查询。

(13) ALLOW_ROW_LOCKS：该选项用于指定是否使用行锁，为 ON，表示使用行锁。

(14) ALLOW_PAGE_LOCKS：该选项用于指定是否使用页锁，为 ON，表示使用页锁。

(15) MAXDOP：该选项用于指定索引操作期间覆盖最大并行度的配置选项。主要目的是限制执行并行计划过程中使用的处理器数量。

【例 5-59】 用 CREATE INDEX 语句为学生信息表(Students)建立非聚集索引 example_index，使表中的数据先按姓名的升序索引，再按年龄的降序索引。

语句清单如下：

```
USE Test
CREATE NONCLUSTERED INDEX example_index
ON Students(Sname ASC,age DESC)
```

5.5.9 操作索引

索引创建后，可以对其进行查看、修改、删除等操作。

和操作视图一样，对索引的操作方法也有两种，使用方便的图形化工具和使用Transact-SQL语句。在本节中，将主要使用Transact-SQL语句管理索引。

1. 查看索引

Transact-SQL提供了sp_helpindex系统存储过程用来查看索引信息，其语法格式为：

```
[EXEC] sp_helpindex table_name
```

【例5-60】 查看学生信息表中的索引信息。

语句清单如下：

```
USE Test
exec sp_helpindex Students
```

返回结果如图5-34所示。

结果 消息

	index_name	index_description	index_keys
1	example_index	nonclustered located on PRIMARY	sex, age(-)
2	PK_Student	clustered, unique, primary key located on PRIMARY	Sno
3	Students_name_index	nonclustered located on PRIMARY	Sname

图5-34 执行结果图示

2. 修改索引

Transact-SQL提供了ALTER INDEX修改索引，语法格式如下。

(1) 重新生成索引：

```
ALTER INDEX index_name ON table_or_view_name REBUILD
```

(2) 重新组织索引：

```
ALTER INDEX index_name ON table_or_view_name REORGANIZE
```

(3) 禁用索引：

```
ALTER INDEX index_name ON table_or_view_name DISABLE
```

上述语句中，index_name表示所要修改的索引名称，table_or_view_name表示当前索引基于的表或视图名。

下面看一个具体实例，使用ALTER INDEX语句将表Students中的Students_name_index索引修改为禁止访问，可以使用如下语句：

```
ALTER INDEX Students_name_index ON Students Disable
```

3. 删除索引

Transact-SQL 提供了 DROP INDEX 语句一次性删除数据表的一个或多个索引，其语法格式为：

```
DROP INDEX < table or view name >.< index name >
```

也可以使用以下格式：

```
DROP INDEX < index name > ON < table or view name >
```

【例 5-61】 删除学生信息表中的 example_index 索引。

语句清单如下：

```
USE Test
DROP INDEX Students.example_index
```

5.6 SQL Server 2008 的其他功能

5.6.1 存储过程

1. 存储过程概述

存储过程是一组 Transact-SQL 语句和可选控制流语句的预编译集合，以一个确定的名称存储在数据库内，可由应用程序调用执行，而且允许用户声明变量、有条件执行以及其他强大的编程功能，类似于其他编程语言里的过程。像前面介绍过的 sp_help、sp_helptext、sp_depends、sp_helpindex 等都是 Transact-SQL 已编译好并存储在数据库中的系统存储过程，用户也可以编写自定义的存储过程。

使用存储过程有很多优点，主要体现在以下几个方面：

(1) 允许模块化程序设计。只需创建过程一次并将其存储在数据库中，以后即可在程序中调用该过程任意次。存储过程可由在数据库编程方面有专长的人员创建，并可独立于程序源代码而单独修改。

(2) 执行速度更快。如果某操作需要大量 Transact-SQL 语句或需重复执行，存储过程将比 Transact-SQL 批语句的执行要快。在创建存储过程时可以对其进行分析和优化，并可在首次执行该过程后使用该过程内存中的版本。而每次运行非存储过程中的 Transact-SQL 语句时，都要从客户端重复发送，并且在 SQL Server 每次执行这些语句时，都要对其进行编译和优化。

(3) 减少网络流量。一个需要数百行 Transact-SQL 语句的操作由一条执行存储过程语句的单独语句就可实现，而不需要在网络中发送数百行语句。

(4) 可提供安全机制。即使对于没有直接执行存储过程中语句权限的用户，也可授予他们执行该存储过程的权限。

2. 创建存储过程

如果用户要自己创建存储过程，SQL Server 2008 提供了两种方法：使用图形化工具；使用 Transact-SQL 语句。

1）使用图形化工具创建存储过程

使用 SQL Server 2008 的 SQL Server Management Studio 工具创建存储过程的操作步骤如下：

（1）打开 SQL Server Management Studio 窗口，连接到 Test 数据库。

（2）依次展开“服务器”→“数据库”→Test→“可编程性”节点。

（3）从列表中右击“存储过程”节点，选择“新建存储过程”命令，然后将出现如图 5-35 所示的显示 CREATE PROCEDURE 语句的模板，可以修改要创建的存储过程的名称，然后加入存储过程所包含的 SQL 语句。

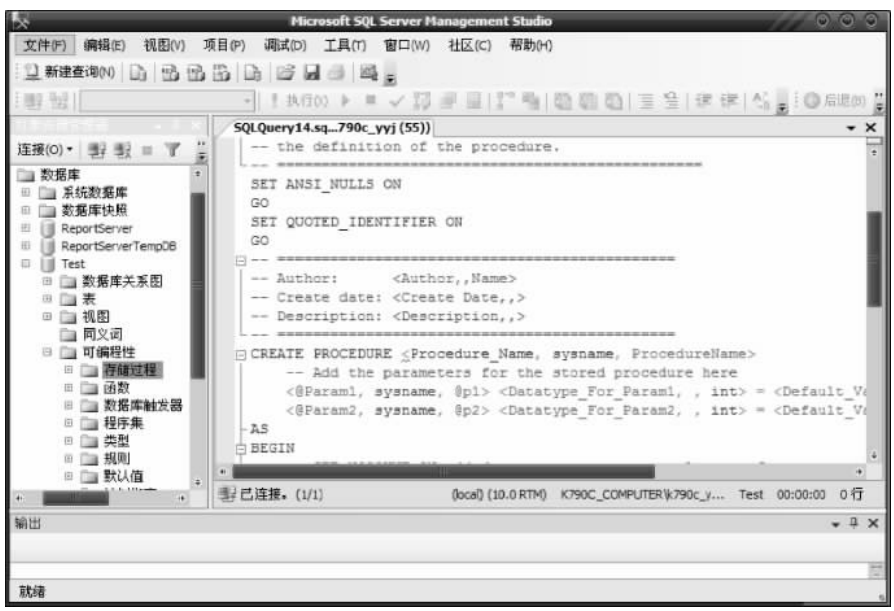


图 5-35 创建存储过程

（4）修改完后，单击“执行”按钮，即可创建一个存储过程。

2）利用 Transact-SQL 语句创建存储过程

Transact-SQL 提供了 CREATE PROCEDURE 语句创建存储过程，其语法格式为：

```
CREATE PROCDURE procedure_name[;number]
[{@parameter data_type}
[VARYING][ = default][OUTPUT]][,...n]
[WITH
{RECOMPILE|ENCRYPTION|RECOMPILE,ENCRYPTION}]
[FOR REPLICATION]
AS sql_statement[...n]
```

上述创建存储过程语句中各主要参数的意义简要介绍如下，更详细说明请参照联机

丛书。

(1) procedure_name：新存储过程的名称。

(2) ；number：是可选的整数，用来对同名的过程分组，以便用一条 DROP PROCEDURE 语句即可将同组的过程一起除去。例如，名为 orders 的应用程序使用的过程可以命名为 orderproc;1、orderproc;2 等。DROP PROCEDURE orderproc 语句将除去整个组。

(3) @parameter：过程中的参数。在 CREATE PROCEDURE 语句中可以声明一个或多个参数。用户必须在执行过程时提供每个所声明参数的值(除非定义了该参数的默认值)。存储过程最多可以有 2100 个参数。

(4) data_type：参数的数据类型。

(5) default：参数的默认值。

(6) OUTPUT：表明参数是返回参数。

(7) RECOMPILE：表明 SQL Server 不会缓存该过程的计划，该过程将在运行时重新编译。

(8) ENCRYPTION：表示 SQL Server 加密 syscomments 表中包含 CREATE PROCEDURE 语句文本的条目。

(9) FOR REPLICATION：指定不能在订阅服务器上执行为复制创建的存储过程。

(10) AS：指定过程要执行的操作。

(11) sql_statement：过程中要包含的任意数目和类型的 Transact-SQL 语句。

3. 执行存储过程

创建了存储过程后，该存储过程可以供各种应用程序调用，也可以在 SQL Server 2008 的查询分析器中执行。简单的执行方法是直接使用 EXECUTE 语句来执行存储过程。事实上 EXECUTE 语句还有更复杂的语法格式：

```
[ { EXEC | EXECUTE } ]
{
[ @return_status = ]
{ procedure_name [;number] | @procedure_name_var }
@parameter = [ { value | @variable [ OUTPUT ] | [ DEFAULT ] } ]
[,...n]
[ WITH RECOMPILE ]
}
```

上述语句中参数的含义具体如下：

(1) @return_status：是一个可选的整型变量，保存存储过程的返回状态。

(2) procedure_name：是拟调用的存储过程的名称。

(3) ;number：是可选的整数，用于将相同名称的过程进行组合，使得它们可以用一句 DROP PROCEDURE 语句除去。

(4) @procedure_name_var：局部定义变量名，代表存储过程名称。

(5) @parameter：过程参数，在 CREATE PROCEDURE 语句中定义。参数名称前必须加上符号@。

(6) value：过程中参数的值。如果参数名称没有指定，参数值必须以 CREATE PROCEDURE 语句中定义的顺序给出。

(7) @variable：用来保存参数或者返回参数的变量。

(8) OUTPUT：指定存储过程必须返回一个参数。

(9) DEFAULT：根据过程的定义，提供参数的默认值。

(10) WITH RECOMPILE：强制编译新的计划。

4. 删除存储过程

Transact-SQL 提供了 DROP PROCEDURE 语句来删除存储过程，其语法格式为：

```
DROP PROCEDURE { procedure } [ ,...n ]
```

通常在新建一个存储过程时，如果存在同名的存储过程会造成创建失败，进而会造成那些直接调用存储过程的应用程序出错，因此通常在创建一个存储过程时，须先判断是否存在同名的存储过程，如果存在，就先删除该存储过程，然后再新建，其一般格式为：

```
IF EXISTS(SELECT name FROM sysobjects WHERE name = 'procedure_name' AND type = 'P')
DROP PROCEDURE procedure_name
```

上述语句先检验在 sysobjects 系统表中是否存在名字相同的存储过程记录，如果存在，则删除原有的存储过程。

5. 综合实例

【例 5-62】 在 Test 数据库中建立存储过程 example_proc，要求实现在学生信息库中查询指定姓名学生的年龄。其中学号作为输入参数，在执行存储过程时由用户指定。

(1) 创建存储过程语句清单如下：

```
USE Test
IF EXISTS(SELECT name FROM sysobjects
WHERE name = 'example_proc' AND type = 'P')
DROP PROCEDURE example_proc
GO
CREATE PROCEDURE example_proc
@name VARCHAR(10),
@age INT OUTPUT
AS
SELECT @age = age FROM Students WHERE Sname = @name
```

上述创建存储过程的语句中，首先判断是否存在同名的存储过程，如果存在则将其删除，接着在创建存储过程的 CREATE PROCEDURE 语句中指明存储过程名为 example_proc，同时定义两个变量，name 为输入参数，age 为输出变量，最后在 AS 子句中指明存储过程所要执行的功能，即在学生信息表中查找姓名字段为变量 name 的值的年龄，并将查找的结果赋给 age 变量以供输出。

(2) 执行存储过程语句清单如下：

```
USE Test
```

```
DECLARE @age_out INT
EXEC example_proc '张三',@age_out OUTPUT
SELECT @age_out
```

上述执行存储过程的语句中，先用 DECLARE 语句声明了变量 age_out，用来接收存储过程的输出数据，然后在 EXEC 语句中指明要执行的存储过程名 example_proc，输入参数的值为'张三'（该值会传递给存储过程的 name 变量），如果有多个参数，按顺序给出每个值，同时还指明 age_out 变量为接收输出数据的变量，最后通过 SELECT 语句显示输出的数据，对于输出数据的处理不一定要用该语句输出，用户一旦得到输出数据 age_out，可以根据需求进行各种不同的处理。

上述语句在查询分析器中执行后，最后输出结果如图 5-36 所示。

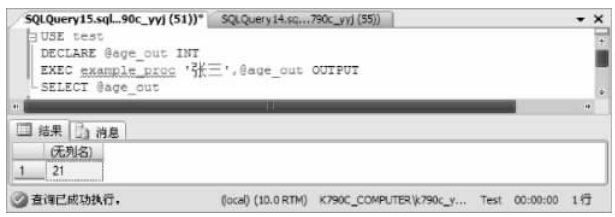


图 5-36　执行存储结果显示

以上介绍的只是简单存储过程的创建和执行过程，要编写出更加复杂、功能更全的存储过程，需要一定的 Transact-SQL 语言编程能力。因篇幅所限，关于 Transact-SQL 语言编程知识，用户可自行参考 SQL Server 的联机丛书。

5.6.2　触发器

1. 触发器概述

触发器是一种特殊类型的存储过程，它与前面介绍的一般存储过程不同，不是通过存储过程名被直接调用，而是通过事件进行触发而被执行。例如当用 UPDATE、INSERT 或 DELETE 等修改操作对指定表中的数据进行修改时，会导致触发器生效。触发器可以查询其他表，而且可以包含复杂的 SQL 语句，主要用于强制复杂的业务规则或要求。触发器与数据表紧密相连，甚至可以看作是数据表定义的一部分，它有助于强制引用完整性，以便在添加、更新或删除表中的行时保留表之间已定义的关系。

使用触发器有不少优点，主要体现在以下几个方面：

（1）触发器具有自动执行的特点。在对表的数据做了任何修改（比如手工输入或者应用程序采取的操作）之后，触发器立即被激活。

（2）触发器可以通过数据库中的相关表进行层叠更改。例如，可以在学生信息表的学号列上写入一个删除触发器，以使其他表中的各匹配行采取删除操作。该触发器用学号列作为唯一键，在学生成绩表中对各匹配行进行定位。

（3）触发器可以强制限制。这些限制比用 CHECK 约束所定义的更复杂。与 CHECK 约束不同的是，触发器可以引用其他表中的列。

2. 创建触发器

如果用户要自己创建触发器，可以通过 SQL Server 2008 的企业管理器和 Transact-SQL 语句来实现。

1）利用企业管理器创建触发器

打开 Microsoft SQL Server Management Studio 后，选中要创建触发器的数据表，例如表 Students，展开后右击“触发器”节点，在右键菜单中选择“新建触发器”命令，弹出如图 5-37 所示的窗口。

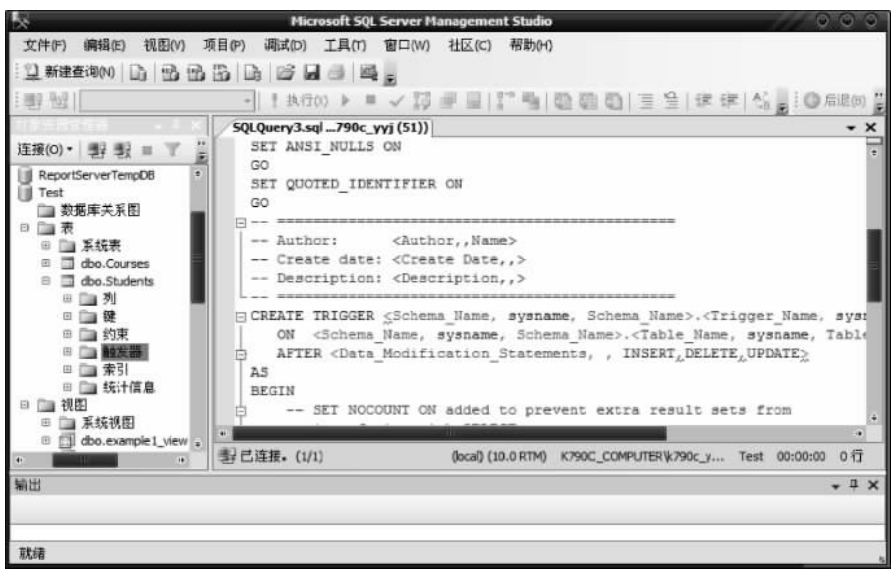


图 5-37 创建触发器

在“触发器属性”对话框的“文本”区域中填入生成触发器的 Transact-SQL 语句，触发器名由 SQL 语句中的 trigger_name 提供，编辑完后单击“执行”，完成触发器的创建。

2）利用 Transact-SQL 语句创建触发器

Transact-SQL 提供了 CREATE TRIGGER 语句来创建触发器，其语法格式为：

```
CREATE TRIGGER trigger_name
ON { table | view }
 [ WITH ENCRYPTION ]
{{ { FOR | AFTER | INSTEAD OF } { [DELETE][,][ INSERT ] [ , ] [ UPDATE ] }
    [ WITH APPEND ]
    [ NOT FOR REPLICATION ]
    AS
    [ { IF UPDATE ( column )
        [ { AND | OR } UPDATE ( column ) ]
            [ ...n ]
    | IF ( COLUMNS_UPDATED ( ) { bitwise_operator } updated_bitmask )
          { comparison_operator } column_bitmask [ ...n ]
    } ]
    sql_statement [ ...n ]
}}
```

上述创建触发器语句中，各主要参数的意义简要介绍如下，更详细说明请参照联机丛书。

(1) trigger_name：触发器的名称。

(2) table | view：在其上执行触发器的表或视图，有时称为触发器表或触发器视图。

(3) WITH ENCRYPTION：加密 syscomments 表中包含 CREATE TRIGGER 语句文本的条目。

(4) AFTER：指定触发器只有在触发 SQL 语句中指定的所有操作都已成功执行后才激发。所有的引用级联操作和约束检查也必须成功完成后，才能执行此触发器。未加此参数即表示 AFTER 触发器。

(5) INSTEAD OF：指定执行触发器而不是执行触发 SQL 语句，从而替代触发语句(如 INSERT、UPDATE 或 DELETE)的操作。

(6) { [DELETE] [,] [INSERT] [,] [UPDATE] }：指定在表或视图上执行哪些数据修改语句时将激活触发器的关键字。必须至少指定一个选项。

(7) WITH APPEND：指定应该添加现有类型的其他触发器。

(8) NOT FOR REPLICATION：表示当复制进程更改触发器所涉及的表时，不应执行该触发器。

(9) AS：触发器要执行的操作。

(10) sql_statement：触发器的条件和操作。触发器条件指定其他准则，以确定 DELETE、INSERT 或 UPDATE 语句是否导致执行触发器操作。CREATE TRIGGER 语句中使用两个特殊的表：deleted 和 inserted，它们是逻辑(概念)表，这些表在结构上类似于定义触发器的表(也就是在其中尝试用户操作的表)，用于保存用户操作可能更改的行的旧值或新值。

(11) IF UPDATE (column)：测试在指定的列上进行的 INSERT 或 UPDATE 操作，不能用于 DELETE 操作。

(12) IF (COLUMNS_UPDATED())：测试是否插入或更新了提及的列，仅用于 INSERT 或 UPDATE 触发器中。

(13) bitwise_operator：用于比较运算的位运算符。

(14) updated_bitmask：整型位掩码，表示实际更新或插入的列。

(15) comparison_operator：比较运算符。

(16) column_bitmask：要检查的列的整型位掩码，用来检查是否已更新或插入了这些列。

【例 5-63】 在学生信息表中创建一个 UPDATE 触发器 example_trig，要求当向学生信息表中插入和更新数据时，如果在性别字段中插入或修改性别后，触发该触发器，检查性别字段的内容是否为“男”或“女”，如果不是，给出提示。

(1) 创建触发器语句清单如下：

```
USE Test
IF EXISTS(SELECT name FROM sysobjects
WHERE name = 'example_trig' AND type = 'TR')
  DROP TRIGGER example_trig
```

```
GO
CREATE TRIGGER example_trig ON Students
FOR INSERT,UPDATE
AS
   DECLARE @sex CHAR(1)
   SELECT @sex = sex FROM Students
  IF @sex<>'B' AND @sex<>'G'
   BEGIN
    PRINT '性别输入有误,必须为男或女!'
    ROLLBACK TRANSACTION
END
```

上述触发器语句中的 ROLLBACK TRANSACTION 语句表示将数据回滚到刚刚所做的数据修改之前,即取消刚才的插入操作。

(2) 测试触发器语句清单如下:

```
USE Test
INSERT INTO Students VALUES('2010000010','小五','23','N')
```

运行上述语句,在结果窗格中会显示"性别输入有误,必须为男或女!",查看学生信息表记录会发现该条记录没有成功插入至数据表中。

【例 5-64】 在学生信息表中创建一个 AFTER 触发器 example_trig2,要求当删除学生信息表中的某个学生的记录时,该学生在学生成绩表中的记录也随之自动被删除。

(1) 创建触发器语句清单如下:

```
USE Test
IF EXISTS(SELECT name FROM sysobjects
WHERE name = 'example_trig2' AND type = 'TR')
  DROP TRIGGER example_trig2
GO
CREATE TRIGGER example_trig2 ON Students
FOR DELETE
AS
DELETE Students WHERE Sno = (SELECT Sno FROM Students)
```

(2) 测试触发器语句清单如下:

```
USE Test
DELETE Students WHERE Sno = '2010000001'
```

运行上述语句后,查看学生信息表,会发现表中的学号为"2010000001"的记录被删除。

3. 禁用和启用触发器

禁用触发器可以使已存在的触发器不触发,就好像不存在一样,但它并没有被从数据表中删除。启用触发器则是使触发器从禁用状态恢复为正常使用状态。禁用触发器和启用触发器都可以通过 Transact-SQL 的 ALTER TABLE(修改数据表)中的 DISABLE | ENABLE 语句来实现,其语法格式为:

```
ALTER TABLE table_name DISABLE | ENABLE TRIGGER trigger_name
```

例如要使学生信息表中的 example_trig 失效，可以用以下 SQL 语句实现：

```
USE Test
ALTER TABLE Students DISABLE TRIGGER example_trig
```

若要使学生信息表中的 example_trig 重新有效，可以用以下 SQL 语句实现：

```
USE Test
ALTER TABLE Students ENABLE TRIGGER example_trig
```

4. 删除触发器

Transact-SQL 提供了 DROP TRIGGER 语句来删除触发器，其语法格式为：

```
DROP TRIGGER trigger_name1[ ,...n ]
```

在前面的例子(例 5-63、例 5-64)中已经用到了该语句，同存储过程一样，在新建一个触发器时，如果存在同名的触发器会造成创建失败，因此通常在创建一个触发器时，先判断是否存在同名的触发器，如果存在，就先删除该触发器，然后再新建，其一般格式为：

```
IF EXISTS(SELECT name FROM sysobjects
WHERE name = 'trigger_name' AND type = 'TR')
      DROP TRIGGER trigger_name
```

上述语句先检验在 sysobjects 系统表中是否存在名字相同的触发器记录，如果存在，则删除原有的触发器。

5.6.3 数据备份与恢复

1. 数据备份与恢复概述

1) 数据备份与恢复的概念

数据库的备份与恢复是数据库管理员维护数据安全性和完整性的重要操作。数据备份是指复制数据库结构、对象和数据，以便在数据库遭到破坏时能够修复数据库；数据恢复是指将数据库备份加载到服务器，以恢复被破坏的数据库。

可能导致数据库中的数据被破坏的原因有很多，例如存储介质损坏、用户执行了误操作、服务器崩溃等，这些都可以依靠事先做好的备份来进行数据恢复。另外，有时需要将数据库从一个服务器上转移到另一服务器上，此时也可以通过在源服务器上进行数据备份，然后在目的服务器上进行恢复操作来完成数据库的复制。

2) 数据备份方法

SQL Server 2008 提供了高性能的备份和恢复功能，用户可以根据需求设计自己的备份策略，以保护存储在 SQL Server 2008 数据库中的关键数据。

SQL Server 2008 提供了 4 种数据库备份类型：

(1) 完整数据库备份：复制包括事务日志在内的整个数据库。这种方法在备份过程中需要花费的时间和空间最多，不宜经常使用。

(2) 差异数据库备份：在完整数据库备份之间执行，只备份最后一次全库备份后被修改的数据页，所占用的时间和空间较少。

(3) 事务日志备份：提供了连续的事务信息链，可支持从数据库、差异或文件备份中快速恢复。

(4) 文件组备份：当时间限制使得完整数据库备份不切实际时，可以使用 BACKUP 备份数据库文件和文件组，而不是备份完整数据库。若要备份一个文件而不是整个数据库时，请合理安排步骤以确保数据库中所有的文件按规则备份，同时必须进行单独的事务日志备份。在恢复一个文件备份后，使用事务日志将文件内容前滚，使其与数据库其余部分一致。

3) 数据恢复模型

SQL Server 2008 提供了 3 种数据恢复模型，其中每种恢复模型都能够在数据库发生故障时恢复相关的数据。不同的恢复模型在 SQL Server 备份、恢复的方式和性能方面存在差异，而且，采用不同的恢复模型对于避免数据损失的程度也不同。每个数据库必须选择 3 种恢复模型中的 1 种以确定备份数据库的备份方式。

(1) 简单恢复。使用简单恢复模型可以将数据库恢复到上次备份的即时点，但是无法将数据库还原到故障点或特定的即时点。若要还原到这些点，请选择完全恢复或大容量日志记录恢复。

(2) 完全恢复。该模型使用数据库备份和事务日志备份提供对存储介质故障的完全防范。如果一个或多个数据文件损坏，则该恢复可以还原所有已提交的事务，正在进行的事务将回滚。完全恢复提供将数据库恢复到故障点或特定即时点的能力。为保证这种恢复程度，包括大容量操作在内的所有操作都将完整地记入日志。

(3) 大容量日志记录恢复。该模型提供对存储介质故障的防范，并对某些大规模或大容量复制操作提供最佳性能和最少的日志使用空间。在大容量日志记录恢复模型中，这些大容量复制操作的数据丢失程度要比完全恢复模型严重。

4) 备份类型与恢复模型的关系

备份类型与恢复模型的关系如表 5-12 所示。

表 5-12 备份类型与恢复模型关系表

备份类型 / 模型	数据库	数据库差异	事务日志	文件或文件差异
简单	必须	可选	不允许	不允许
完全	必须(或文件备份)	可选	必须	可选
大容量日志记录	必须(或文件备份)	可选	必须	可选

2. 数据备份操作

对于备份数据的 4 种类型，在创建每一种备份时，所用到的操作都不相同。下面针对这 4 种类型，详细介绍使用 Transact-SQL 语句进行备份的操作(利用 SQL Server Management Studio 工具创建备份比较简单，这里不做介绍)。

1) 创建完整备份

使用 BACKUP 命令来备份数据库。对数据库进行完整备份的语法如下：

```
BACKUP DATABASE database_name
TO <backup_device> [...n]
```

```
[WITH
[[,] NAME = backup_set_name]
[ [,] DESCRIPITION = 'TEXT' ]
[ [,] {INIT | NOINIT } ]
[ [,]{ COMPRESSION | NO_COMPRESSION }
]
```

上述语句中一些参数选项的说明如下：

(1) database_name：指定了要备份的数据库。

(2) backup_device：为备份的目标设备，采用“备份设备类型＝设备名”的形式。

(3) WITH 子句：指定备份选项，这里仅给出两个，更多的备份选项可以参考 SQL Sever 联机丛书。

(4) NAME＝backup_set_name：指定了备份的名称。

(5) DESCRIPITION＝'TEXT'：给出了备份的描述。

(6) INIT|NOINIT：INIT 表示新备份的数据覆盖当前备份设备上的每一项内容，即原来在此设备上的数据信息都将不存在，NOINIT 表示新备份的数据添加到备份设备上已有的内容的后面。

(7) COMPRESSION|NO_COMPRESSION：COMPRESSION 表示启用备份压缩功能，NO_COMPRESSION 表示不启用备份压缩功能。

2) 创建差异备份

创建差异备份也可以使用 BACKUP 语句，进行差异备份的语法与完整备份的语法相似，其格式如下：

```
BACKUP DATABASE database_name
TO < backup_device > [...n]
WITH
DIFFERENTIAL
[[,] NAME = backup_set_name]
[ [,] DESCRIPITION = 'TEXT' ]
[ [,] {INIT | NOINIT } ]
[ [,]{ COMPRESSION | NO_COMPRESSION }
]
```

其中，WITH DIFFERENTIAL 子句指明了本次备份是差异备份。其他参数与完全备份参数完全一样，这里不再重复。

3) 创建事物日志备份

使用 BACKUP 语句创建事务日志备份，语法格式如下：

```
BACKUP LOG database_name
TO < backup_device > [...n]
WITH
[[,] NAME = backup_set_name]
[ [,] DESCRIPITION = 'TEXT' ]
[ [,] {INIT | NOINIT } ]
[ [,]{ COMPRESSION | NO_COMPRESSION }
]
```

其中，LOG 指定仅备份事务日志。该日志是从上一次成功执行的日志备份到当前日志的末尾。必须创建完整备份，才能创建第一个日志备份。其他各参数与完整备份语法中各参数完全一样，这里也不再重复。

4）创建文件组备份

可以使用 BACKUP 语句对文件组备份，具体的语法格式如下：

```
BACKUP DATABASE database_name
< file_or_filegroup > [...n]
TO < backup_device > [...n]
WITH options
```

其中，file_or_filegroup 指定了要备份的文件或文件组。如果是文件，则写作“FILE＝逻辑文件名”；如果是文件组，则写作“FILEGROUP＝逻辑文件组名”。WITH options 用于指定备份选项，与前几种备份设备类型相同。

3. 数据恢复操作

恢复数据库，就是让数据库根据备份的数据回到备份时的状态。当恢复数据库时，SQL Server 会自动将备份文件中的数据全部复制到数据库，并回滚任何未完成的事务，以保证数据库中的数据的完整性。

1）常规恢复

恢复数据前，管理员应当断开准备恢复的数据库和客户端应用程序之间的一切连接，此时，所有用户都不允许访问该数据库，并且执行恢复操作的管理员也必须更改数据库连接到 master 或其他数据库，否则不能启动恢复进程。

在执行任何恢复操作前，用户都要对事务日志进行备份，这样有助于保证数据的完整性。如果用户在恢复之前不备份事务日志，那么用户将丢失从最近一次数据库备份到数据库脱机之间的数据更新。

使用 SQL Server Management Studio 工具恢复数据库的操作步骤如下：

（1）打开 SQL Server Management Studio 工具，连接服务器。

（2）在对象资源管理器中，展开“数据库”节点，右击 Test 数据库，在弹出的命令菜单中选择“任务”→“还原”→“数据库”命令，打开“还原数据库”窗口。

（3）在“还原数据库”窗口中选中“源设备”单选按钮，然后单击 … 弹出一个“指定备份”对话框，在“备份媒体”选项中选择“备份设备”选项，然后单击“添加”按钮，选择之前创建的 Test_backup.bak 备份设备。然后选择相应的备份，单击“确定”按钮完成恢复操作，如图 5-38 所示。

2）时间点恢复

在 SQL Server 2008 中进行事务日志备份时，不仅给事务日志中的每个事务标上日志号，还给它们都标上一个时间。这个时间与 RESTORE 语句的 STOPAT 从句结合起来，允许将数据返回到前一个状态。但是，在使用这个过程时需要记住两点：

（1）这个过程不适用于完整与差异备份，只适用于事务日志备份。

（2）将失去 STOPAT 时间之后整个数据库上所发生的任何修改。

例如，一个数据库每天有大量的数据，每天 12 点都会定时做事务日志备份，10:00 的时

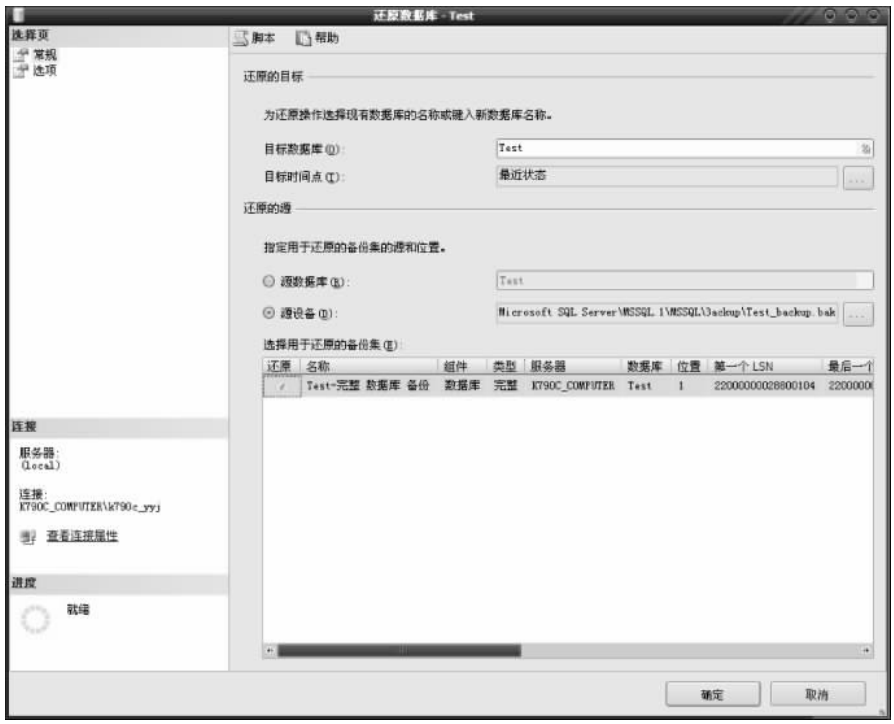


图 5-38 还原数据库

候服务器出现故障,误清除了许多重要的数据。通过对日志备份的时间点恢复,可以把时间点设置在 10:00:00,既可以保存 10:00:00 之前的数据修改,又可以忽略 10:00:00 之后的错误操作。

使用 SQL Server Management Studio 工具按照时间点恢复数据库的操作步骤如下:

(1) 打开 SQL Server Management Studio 工具,连接服务器。

(2) 在对象资源管理器中,展开“数据库”节点,右击任意一个用户数据库,在弹出的命令菜单中选择“任务”→“还原”→“数据库”命令,打开“还原数据库”窗口。

(3) 单击“目标时间点”文本框后面的“选项”按钮[…],打开“时点还原”窗口,启用“具体日期和时间”选项,输入具体时间“10:00:00”,如图 5-39 所示。

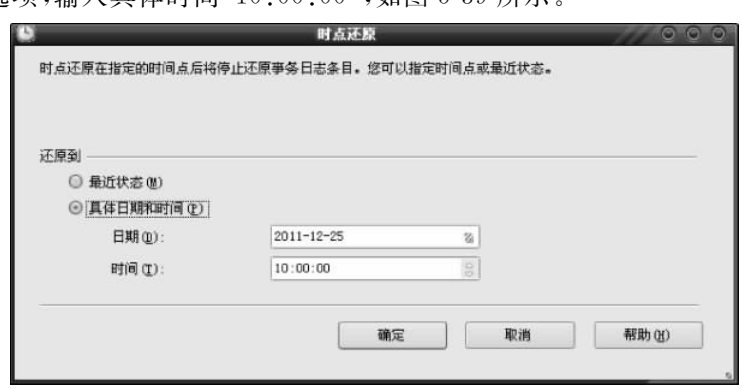


图 5-39 设置时点还原的日期和时间

(4) 设置完成后,单击“确定”按钮返回。然后还原备份,设置时间以后的操作将会被还原。

5.6.4 安全管理

数据库的安全性是指保护数据库以防止不合法的使用所造成的数据泄漏、更改或破坏。系统安全保护措施是否有效是数据库系统的主要性能指标之一。数据库的安全性和计算机系统的安全性(包括操作系统、网络系统的安全性)是紧密联系、相互支持的。

1. 安全机制

1) 安全等级

SQL Server 2008 的安全机制主要可以划分为 3 个等级:

(1) 服务器级别的安全机制。这个级别的安全性主要通过登录账户进行控制,要想访问一个数据库服务器,必须拥有一个登录账户。登录账户可以是 Windows 账户或组,也可以是 SQL Server 的登录账户。登录账户可以属于相应的服务器角色。至于角色,可以理解为权限的组合。

(2) 数据库级别的安全机制。这个级别的安全性主要通过用户账户进行控制,要想访问一个数据库,必须拥有该数据库的一个用户账户身份。用户账户是通过登录账户进行映射的,可以属于固定的数据库角色或自定义数据库角色。

(3) 数据对象级别的安全机制。这个级别的安全性通过设置数据对象的访问权限进行控制。如果是使用图形界面管理工具,可以在表上右击,选择“属性”→“权限”选项,然后启用相应的权限复选框即可。

上述每个安全等级就好像一道门,如果门没有上锁或用户拥有开门的钥匙,则用户就可以通过这道门而到达下一个安全等级。

2) 权限

数据库权限指明用户获得哪些数据库对象的使用权,以及用户能够对这些对象执行何种操作。用户在数据库中拥有的权限取决于用户账户的数据库权限和用户所在角色的类型两个因素。

(1) 对象权限:表示对特定的数据库对象的操作权限,它决定了能对表、视图等数据库执行哪些操作。对于表和视图,权限用来控制用户在表和视图上执行 SELECT、INSERT、UPDATE 和 DELETE 语句的能力;对于表和视图中的字段,权限用来控制用户在单个字段上执行 SELECT、UPDATE 和 REFERENCES 操作的能力;对于存储过程,权限用来控制用户执行 EXECUTE 语句的能力。

(2) 语句权限:决定了用户在数据库可以执行哪些 Transact-SQL 语句的权限。可用于语句权限的 Transact-SQL 语句有:

BACKUP DATABASE:备份数据库

BACKUP LOG:备份事务日志

CREATE DATABASE:创建数据库

CREATE DEFAULT:创建默认

CREATE FUNCTION:创建函数

CREATE PROCEDURE：创建存储过程

CREATE RULE：创建规则

CREATE TABLE：创建数据表

CREATE VIEW：创建视图

(3) 删除权限：通过删除某种权限可以停止以前授予或者拒绝的权限。使用REVOKE语句删除以前授予的或者拒绝的权限。删除权限是删除已授予的权限，并不是妨碍用户、组或者角色从更高级别集成已授予的权限。

3) 安全性角色

角色是SQL Server 2008用来集中管理数据库或者服务器的权限。数据库管理员将操作数据库的权限赋予角色。然后，数据库管理员再将角色赋给数据库用户或者登录账户，从而使数据库用户或者登录账户拥有了相应的权限。

(1) 服务器角色。SQL Server 2008提供了以下固定服务器角色：

sysadmin：可以在SQL Server中执行任何活动。

serveradmin：可以设置服务器范围的配置选项，关闭服务器。

setupadmin：可以管理链接服务器和启动过程。

securityadmin：可以管理登录和CREATE DATABASE权限，还可以读取错误日志和更改密码。

processadmin：可以管理在SQL Server中运行的进程。

dbcreator：可以创建、更改和除去数据库。

diskadmin：可以管理磁盘文件。

bulkadmin：可以执行BULK INSERT语句。

(2) 数据库角色。SQL Server 2008提供了以下固定数据库角色：

db_owner：在数据库中有全部权限。

db_accessadmin：可以添加或删除用户ID。

db_securityadmin：可以管理全部权限、对象所有权、角色和角色成员资格。

db_ddladmin：可以发出ALL DDL，但不能发出GRANT、REVOKE或DENY语句。

db_backupoperator：可以发出DBCC、CHECKPOINT和BACKUP语句。

db_datareader：可以选择数据库内任何用户表中的所有数据。

db_datawriter：可以更改数据库内任何用户表中的所有数据。

db_denydatareader：不能选择数据库内任何用户表中的任何数据。

db_denydatawriter：不能更改数据库内任何用户表中的任何数据。

public：在SQL Server 2008中每个数据库用户都属于public数据库角色。这个数据库角色不能被删除。

2. 安全管理

要想保证数据库数据的安全，必须搭建一个相对安全的运行环境。因此，对服务器安全性管理至关重要。在SQL Server 2008中，对服务器安全性管理主要通过更加健壮的验证模式、安全的登录服务器的账户管理以及对服务器角色的控制，从而更加有力地保证服务器的安全、便捷。

1）身份验证模式

SQL Server 2008 提供了 Windows 身份和混合身份两种验证模式，每一种身份验证都有一个不同类型的登录账户。无论哪种模式，SQL Server 2008 都需要对用户的访问进行两个阶段的检验：验证阶段和许可确认阶段。

（1）Windows 身份验证模式。当用户通过 Windows 用户账户连接时，SQL Server 使用操作系统中的 Windows 主体标记验证账户名和密码。也就是说，用户身份由 Windows 进行确认。SQL Server 不要求提供密码，也不执行身份验证，如图 5-40 所示。

图 5-40 Windows 身份验证

Windows 身份验证模式有以下主要优点：

① 数据库管理员的工作可以集中在管理数据库上面，而不是管理用户账户。对用户账户的管理可以交给 Windows 去完成。

② Windows 有更强的用户账户管理工具，可以设置账户锁定、密码期限等。如果不通过定制来扩展 SQL Server，SQL Server 则不具备这些功能。

③ Windows 的组策略支持多个用户同时被授权访问 SQL Server。

（2）混合身份验证模式。使用混合安全模式，SQL Server 2008 首先确定用户的连接是否使用有效的 SQL Server 用户账户登录。如果用户有有效的登录账户和使用正确的密码，则接受用户的连接；如果用户有有效的登录账户，但使用了不正确的密码，则用户的连接被拒绝。仅当用户没有有效地登录时，SQL Server 2008 才检查 Windows 账户的信息。在这种情况下，SQL Server 2008 将会确定 Windows 账户是否有连接到服务器的权限。如果账户有权限，连接被接受；否则，连接被拒绝。如图 5-41 所示。

2）管理登录账户

与两种验证模式一样，服务器登录也有两种情况：可用域账户登录，域账户可以是域或本地用户账户、本地组账户或通用的和全局的域组账户；可以通过指定唯一的登录 ID 和密码来创建 SQL Server 2008 登录，默认登录包括本地管理员组、本地管理员、sa、Network Service 和 SYSTEM。

（1）系统管理员组：SQL Server 2008 中管理员组在数据库服务器上属于本地组。这个组的成员通常包括本地管理员用户账户和任何设置为管理员本地系统的其他用户。在 SQL Server 2008 中，此组默认授予 sysadmin 服务器角色。

图 5-41 混合身份验证

(2) 管理员用户账户：管理员在 SQL Server 2008 服务器上的本地用户账户。该账户提供对本地系统的管理权限，主要在安装系统时使用。如果计算机是 Windows 域的一部分，管理员账户通常也有域范围的权限。在 SQL Server 2008 中，这个账户默认授予 sysadmin 服务器角色。

(3) sa 登录：是 SQL Server 系统管理员的账户。而在 SQL Server 2008 中采用了新的集成和扩展的安全模式，sa 不再是必须的，提供此登录账户主要是为了针对以前 SQL Server 版本的向后兼容性。与其他管理员登录一样，sa 默认授予 sysadmin 服务器角色。在默认安装 SQL Server 2008 的时候，sa 账户没有被指派密码。

(4) Network Service 和 SYSTEM 登录：它是 SQL Server 2008 服务器上内置的本地账户，而是否创建这些账户的服务器登录，依赖于服务器的配置。例如，如果已经将服务器配置为报表服务器，此时将有一个 Network Service 的登录账户，这个登录将是 mester、msdb、ReportServer 和 ReportServerTempDB 数据库的特殊数据库角色 RSExceRole 的成员。

在服务器实例设置期间，Network Service 和 SYSTEM 账户可以是为 SQL Server、SQL Server 代理、分析服务和报表服务器所选择的服务账户。在这种情况下，SYSTEM 账户通常具有 sysadmin 服务器和角色，允许其完全访问管理服务器实例。

只有获得 Windows 账户的用户才能建立与 SQL Server 2008 的信任连接(即 SQL Server 2008 委托 Windows 验证用户的密码)。如果正在为其创建登录的用户(比如 Guest 用户)无法建立信任连接，则必须为他们创建 SQL Server 账户登录。

3) 管理用户

要访问特定的数据库，还必须具有用户名。用户名在特定的数据库内创建，并关联一个登录名(当一个用户创建时，必须关联一个登录名)。通过授权给用户来指定用户可以访问的数据库对象的权限。可以这样想象，假设 SQL Server 是一个包含许多房间的大楼，每一个房间代表一个数据库，房间里的资料可以表示数据库对象。则登录名就相当于进入大楼的钥匙，而每个房间的钥匙就是用户名。房间中的资料则可以根据用户名的不同而有不同的权限。

下面利用系统存储过程 sp_grantdbaccess 来添加数据库用户，具体语法是：

```
CREATE USER user_name
  [ { { FOR | FROM }
```

```
    {
    LOGIN login_name
    | CERTIFICATE cert_name
    | ASYMMETRIC KEY asym_key_name
  }
  | WITHOUT LOGIN
]
[ WITH DEFAULT_SCHEMA = schema_name ]
```

其中的语法参数介绍如下：

(1) user_name：指定在此数据库中用于识别该用户的名称。user_name 是 sysname。其长度最多是 128 个字符。

(2) LOGIN login_name：指定要创建数据库用户的 SQL Server 登录名。login_name 必须是服务器中有效的登录名。当此 SQL Server 登录名进入数据库时，他将获取正在创建的数据库用户的名称和 ID。

(3) CERTIFICATE cert_name：指定要创建数据库用户的证书。

(4) ASYMMETRIC KEY asym_key_name：指定要创建数据库用户的非对称密钥。

(5) WITH DEFAULT_SCHEMA = schema_name：指定服务器为此数据库用户解析对象名时将搜索的第一个结构。

(6) WITHOUT LOGIN：指定不应将用户映射到现有登录名。

【例 5-65】 建立一个 SQL Server 的登录账户，然后将该账户添加为 Test 数据库的用户。语句清单如下：

```
USE master
GO
CREATE LOGIN admin
WITH PASSWORD = 'admin123456';
USE Test
CREATE USER admin FOR LOGIN admin;
GO
```

执行上述语句，就为 Test 数据库创建了一个名字为 admin 的用户，如图 5-42 所示。

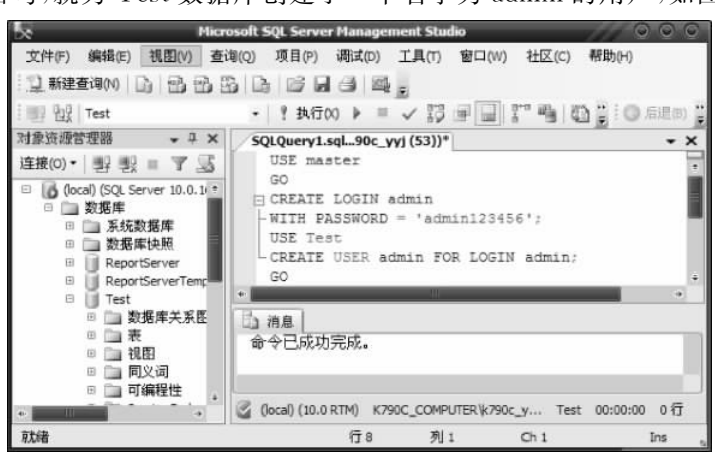


图 5-42 为数据库创建新用户

5.6.5 数据格式转换

1. 数据格式转换的内容

SQL Server 2008 的数据格式转换功能可以使 SQL Server 2008 数据库中的数据用于其他应用系统，也可以将其他应用系统的数据转换为自身所能识别的数据格式，因此它包括两个方面的内容。

(1) 导入数据：是指从 SQL Server 的外部数据源(如 Excel 表格)中检索数据，并将数据插入到 SQL Server 表的过程。

(2) 导出数据：是指将 SQL Server 实例中的数据析取为某些用户指定格式的过程，例如将 SQL Server 表的内容复制到 Microsoft Access 数据库中或 Excel 表格中。

2. 数据格式转换工具

为了解决数据转换问题，SQL Server 2008 提供了一个功能强大的转换组件：导入和导出数据向导工具。利用此转换组件，可以方便地完成数据导入和导出操作。

3. 利用导入和导出向导导入数据

下面以将 Microsoft Excel 2003 表中的数据导入 SQL Server 2008 数据库中为例来介绍如何利用导入和导出向导工具导入数据。假设有一个学生信息表格，如图 5-43 所示，现要将其导入至 SQL Server 2008 中 Test 数据库的学生成绩表中。

	A	B	C	D	E	F
1	学号	姓名	年龄	性别		
2	2010000020	张晓晓	20	G		
3	2010000021	李丽	21	G		
4	2010000022	杨一	19	B		
5	2010000023	吴磊	20	B		
6	2010000024	武清	23	B		
7	2010000025	赵依依	22	G		
8	2010000026	刘蓓	20	G		
9	2010000027	宋柯	22	B		

图 5-43 学生信息表. xls

(1) 依次选择“开始”→“程序”→Microsoft SQL Server 2008→“导入和导出数据(32 位)”命令，启动“SQL Server 导入和导出向导”。

(2) 单击“下一步”按钮，打开“选择数据源”对话框，设置导入的源数据。在对话框中的“数据源”下拉列表中选择 Microsoft Excel 97-2003 作为要导入数据的类型，在随之出现的“文件名”文本框中输入要导入的数据文件所在的位置和文件名，也可以单击旁边的按钮 ...，选择要导入的数据文件，此例应为“E:\...\桌面\学生信息表. xls”。

(3) 单击“下一步”按钮，打开“选择目的”对话框，设置导入的目的数据源，如图 5-44 所示。在对话框的“目标”下拉列表中选择 SQL Server Native Client 10.0，在“服务器”下拉列表中选择要导入到的数据库服务器，并设置登录该服务器的身份验证方式，最后选择要导入到的数据库名 Test。

(4) 单击“下一步”按钮，打开“指定表复制或查询”对话框，设置导入的方式。默认选择“复制一个或多个表或视图的数据(C)”。

(5) 单击“下一步”按钮，打开“选择源表和源视图”对话框，设置要导入的表格和导入到

图 5-44 “选择目标”对话框

的数据表。这里设置源为 Sheet1，目标为[dbo].[Students]，再单击“编辑映射”，设置 Sheet1 中列于数据表 Students 中的对应关系。

（6）单击“下一步”按钮，打开“查看数据类型映射”对话框，如图 5-45 所示。

图 5-45 “查看数据类型映射”对话框

(7) 单击“下一步”按钮,打开“保存、调度和复制包”对话框,选中“立即执行”复选框。

(8) 单击“下一步”按钮,打开“正在完成导入和导出向导”对话框,查看摘要信息无误后,单击“完成”按钮,就可以将“学生信息表.xls”文件中的记录追加至 Test 数据库的学生成绩表中。

4. 利用导入和导出向导工具导出数据

如果要将 Test 数据库中的学生成绩表中的数据导出到学生信息.xls 电子表格中,利用导入和导出向导工具导出数据的步骤与导入数据的步骤类似,只是在选择源数据和目的数据时刚好相反,应将 Test 数据库中的学生信息表作为源数据,而将“学生信息表.xls”表格作为目的数据,按照向导的提示,就可以完成数据的导出操作。

本章小结

本章介绍了 SQL Server 2008 的基本知识,包括 SQL Server 2008 的数据库管理、数据表管理、数据表基本操作、视图和索引的使用方法,并简要介绍了 SQL Server 2008 的存储过程、触发器、数据备份与恢复、安全管理和数据格式转换等常用功能的操作方法。本章在介绍基本理论的同时,通过大量典型例题的讲解来增强知识的应用性和可操作性。通过对本章的学习,读者应对 SQL Server 2008 数据库的基础知识有一定的了解和掌握,为今后的数据库开发奠定基础。

练习题

1. 单项选择题

(1) SQL Server 2008 中最重要也是最常用的管理工具是(　　)。

A. 服务管理器　　B. 对象资源管理器　　C. 查询分析器　　D. 事件探查器

(2) 每个数据库必须有且只有一个的文件是(　　)。

A. 主数据文件　　B. 辅助数据文件　　C. 事务日志文件　　D. 系统数据文件

(3) SQL Server 2008 数据文件中的页有几种类型(　　)。

A. 6　　B. 7　　C. 8　　D. 9

(4) 用于存储用户创建的临时表的数据库是(　　)。

A. master　　B. model　　C. msdb　　D. tempdb

(5) 在建立表格时给列赋默认值使用的关键字为(　　)。

A. DEFAULT　　B. SELECT　　C. INFO　　D. WHERE

(6) 以下系统表中用来记录表、视图和存储过程之间依赖关系的是(　　)。

A. sysobjects　　B. sysusers　　C. sysdepends　　D. sysdatabases

(7) 可以在 SQL Server 中执行任何活动的固定服务器角色是(　　)。

A. serveradmin　　B. sysadmin　　C. db_owner　　D. dbcreator

2. 多项选择题

(1) 可以在 SQL Server 2008 中执行的文件类型是(　　)。

A. 主数据文件　　B. 次要数据文件　　C. 辅助数据文件　　D. 事务日志文件

(2) 以下属于 SQL Server 2008 数据类型的有(　　)。

A. bigint　　B. image　　C. sql-variant　　D. table

(3) SQL Server 2008 中自带的两个实例数据库分别是(　　)。

A. model　　B. msdb　　C. pubs　　D. Northwind

(4) 定义视图的查询不能包含以下哪些子句或关键字(　　)。

A. ORDER BY　　B. COMPUTE BY　　C. INTO　　D. WHERE

(5) SQL Server 2008 提供了(　　)类型的索引。

A. 唯一索引　　B. 主键索引　　C. XML 索引　　D. 全文索引

(6) 以下操作中可能激活触发器的操作有(　　)。

A. INSERT　　B. SELECT　　C. DELETE　　D. SET

3. 简答题

(1) SQL Server 2008 与以前的版本相比,具有哪些新特性?

(2) SQL Server 2008 的数据完整性可以分为哪些类型? 每种类型有什么特点?

(3) 调用存储过程与直接执行 SQL 语句相比有哪些优点?

(4) SQL Server 2008 为数据备份提供了哪些不同的方法? 又提供了哪些数据恢复模型?

(5) SQL Server 2008 数据转换服务主要通过哪些转换工具来实现? 它们各自有什么特点?

4. 实验题

1) 实验 1

建立一个图书借阅管理数据库 library,库中包含以下数据表:

(1) 借书人表。包含的属性有:借书证号,姓名,性别,单位。

(2) 图书表。包含的属性有:书号,书名,数量,位置,出版社编号。

(3) 出版社表。包含的属性有:出版社编号,出版社名,电话,邮编,地址。

(4) 借阅表。包含的属性有:借书证号,书号,借书日期,还书日期。

其中,各表中属性的数据类型和宽度由学生综合考虑各表的功能及关系,自行定义。数据库建立后,在每张数据表中至少输入 10 条记录。

2) 实验 2

在实验 1 中建立的 library 数据库中实现以下操作:

(1) 查询“借书人表”中的所有记录。

(2) 将“图书表”中前 4 条记录的书号、书名、出版社编号信息放入“图书来源表”(未建)。

(3) 查询某一指定出版社编号的所有图书。

(4) 查询“借书人表”中姓“李”的学生的记录,并对查询的结果按借书证号升序排列。

(5) 统计“借书人表”中来自各个单位的学生人数。

3) 实验 3

在实验 1 中建立的 library 数据库中实现以下操作:

(1) 分别采用内联接、外联接的方法查询学生的借书情况,要求查询结果显示学生借书证号、姓名、书名、出版社名。

(2) 用联合查询方法查询“借书人表”和“借阅表”里所有学生的借书证号信息。

(3) 查询“图书表”里数量高于平均数量的图书的书号、书名和出版社编号。

(4) 查询“借阅表”里在“借书人表”中有记录的学生的借书证号信息。

(5) 利用查询设计器实现实验 2 和实验 3 的所有查询。

4) 实验 4

在实验 1 中建立的 library 数据库中实现以下操作:

(1) 创建名为 book_view 的视图,要求基表为“图书表”,在视图中显示书号、书名、位置字段,视图查询的数据为所有数量小于 3 本的图书信息。

(2) 创建名为 borrow_view 的视图,要求在视图中显示“借阅表”中所有借书人的借书证号、姓名、单位、所借书的书名、所借书对应的出版社名。

(3) 通过对 book_view 视图的操作删除“图书表”中所有数量小于 3 本的图书信息。

(4) 为“借书人表”建立非聚集索引 student_index,使表中的数据先按单位的升序索引,再按性别的降序索引。

5) 实验 5

在实验 1 中建立的 library 数据库中实现以下操作:

(1) 建立存储过程 publish_proc,要求实现在“出版社表”中查询指定出版社名的出版社编号、电话、邮编、地址。其中出版社名作为输入参数,在执行存储过程时由用户指定。

(2) 在“借书人表”中创建一个 AFTER 触发器 sex_trig,要求当向“借书人表”中插入和更新数据时,如果在性别字段中插入或修改性别,则触发该触发器,检查性别字段的内容是否为“男”或“女”,如果不是,给出提示。

(3) 完全备份 library 数据库,对 library 数据库中数据进行修改后,再将刚备份的数据库恢复过来。

(4) 将“借阅表”中的数据导出到 EXCEL 文件 borrow. xsl 中。