第3章 数据库和表的管理

前两章介绍了数据管理技术和数据库技术的基本概念和基础知识，为了在计算机上具体实现一个数据库系统，需要使用一个具体的数据库管理系统作为开发工具。

本章首先对常见的数据库管理系统，特别是 SQL Server 做一个总体的介绍，然后重点介绍 SQL Server 2012 最基础的操作，包括数据库的创建、修改与删除，表的创建、修改与删除，表中数据的维护等。

3.1 常见的关系数据库管理系统

数据库管理系统是一种操纵和管理数据库的大型软件，用于建立、使用和维护数据库。数据库管理系统提供多种功能，可使用户能够方便地定义和操纵数据；它对数据库进行统一的管理和控制，维护数据的安全性和完整性，以及进行多用户下的并发控制和恢复数据库。

为了开发各种基于数据库的应用程序，首先需要选择一个合适的数据库管理系统。选择数据库管理系统时需考虑以下几方面：构造数据库的难易程度，应用程序开发的难易程度，数据库管理系统的性能、可移植性和可扩展性，安全控制和故障恢复的能力等。目前，商品化的数据库管理系统以关系数据库为主，技术比较成熟。下面介绍五种常见的关系数据库管理系统。

1. SQL Server

SQL Server 最初是由 Microsoft、Sybase 和 Ashton-Tate 三家公司共同开发的，于 1988 年推出了第一个基于 OS/2 操作系统的版本。在 Windows NT 推出后，Microsoft 与 Sybase 在 SQL Server 的开发上选择了不同的平台。Microsoft 将 SQL Server 移植到 Windows NT 系统上，并专注于开发推广 Windows 操作系统上的 SQL Server 版本，而 Sybase 则专注于 SQL Server 在 UNIX 操作系统上的开发与应用。

在 Microsoft SQL Server 的发展历程中，其版本不断更新：1996 年 Microsoft 推出了 Microsoft SQL Server 6.5 版本；1998 年 Microsoft 推出了 Microsoft SQL Server 7.0 版本；2000 年发布的 SQL Server 2000 在数据库性能、可靠性、易用性方面做了重大改进；2005 年发布的 SQL Server 2005 可为各类用户提供完善的数据库解决方案；2008 年发布的 SQL Server 2008 安全性更强、延展性更好、管理能力更高，是一个全方位的数据管理平台。

SQL Server 界面友好、易学易用且功能强大，与 Windows 操作系统完美结合，可以构造网络环境数据库甚至分布式数据库，可以满足企业大型数据库应用的需要。SQL Server 具有如下特点。

(1) 支持客户机/服务器结构。

客户机/服务器(Client/Server,C/S)结构是指把 DBMS 与应用程序分开,网络上某些计算机专门用于执行 DBMS 功能,完成数据的管理功能,这些计算机称为数据库服务器;另外一些计算机安装 DBMS 的应用开发工具和相关数据库应用程序,这些计算机称为客户机。

在客户机/服务器结构中,DBMS 和数据库存放于数据库服务器上,应用程序和相关开发工具存放于客户机上(目前常用的客户机通常由两层构成,即应用服务器+客户端或者应用服务器+浏览器)。客户机负责管理用户界面、接收用户数据、处理应用逻辑、生成数据库服务请求,将该请求发送给服务器,数据库服务器进行处理后,将处理结果返回给客户机,并将结果按一定格式显示给用户。客户机/服务器结构的工作方式如图 3.1 所示。

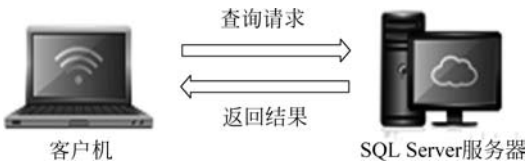

图 3.1　客户机/服务器结构的工作方式

SQL Server 是支持客户机/服务器结构的数据库管理系统。采用客户机/服务器结构后,数据库服务器仅返回用户所需的数据,这样网络上的数据流量将大大减少,可以加速数据的传输;数据集中存储在服务器上,而不是分散在各个客户机上,这使得所有用户都可以访问到相同的数据,而且数据的备份和恢复也很容易。

(2) 分布式数据库功能。

SQL Server 支持分布式数据库结构,可以将逻辑上是一个整体的数据库的数据分别存放在多个不同的 SQL Server 服务器上,客户机可以分别或同时在多个 SQL Server 服务器中存取数据,这样可以降低单个服务器的处理负担,提高系统执行效率。

分布式查询可以引用来自不同服务器的数据,而且这些对于用户来说是完全透明的,分布式数据库将保证任何分布式数据更新时的完整性。通过复制使用户能够维护多个数据副本,这些用户能够自主地进行工作,然后再将所做的修改合并到分布数据库中。

(3) 与 Internet 的集成。

SQL Server 的数据库引擎提供对 Web 技术的支持,使用户很容易将数据库中的数据发布到 Web 页面上。

(4) 具有很好的伸缩性与可用性。

同一个数据库引擎可以在多种版本的 Windows 操作系统上使用。SQL Server 提供的图形用户界面管理工具使得系统管理和数据库的操作更加直观、方便。

(5) 数据仓库功能。

SQL Server 提供了用于提取和分析数据,以进行联机分析处理(OLAP)的工具。

2. Oracle

Oracle 是甲骨文公司的一个关系数据库管理系统,它是在数据库领域一直处于领先地位的产品。可以说 Oracle 数据库系统是目前世界上流行的关系数据库管理系统,系统可移植性好、使用方便、功能强,适用于各类大、中、小、微型机环境。它是一种高效率、可靠性好、

适应高吞吐量的数据库解决方案。

Oracle 数据库产品具有以下优良特性。

(1) 支持大数据量、多用户的高性能的事务处理。Oracle 支持的最大的数据库,其大小可到几百千兆,可充分利用硬件设备。支持大量用户同时在同一数据库上执行各种数据应用。

(2) 兼容性。Oracle 产品采用标准 SQL,并经过美国国家标准技术所(NIST)测试,与 IBM SQL/DS、DB2、INGRES、IDMS/R 等兼容。

(3) 可移植性。Oracle 的产品可运行在很宽范围的硬件与操作系统平台上,它可以安装在 70 种以上不同的大、中、小型机上,并且可在 VMS、DOS、UNIX、Windows 等多种操作系统下工作。

(4) 高生产率。Oracle 产品提供了多种开发工具,能极大地方便用户进行进一步的开发。

(5) 安全性。获得最高认证级别的 ISO 标准安全认证。

3. MySQL

MySQL 是一个小型的关系数据库管理系统,开发者为瑞典 MySQLAB 公司,在 2008 年 1 月 16 日被 Sun 公司收购。MySQL 被广泛地应用在 Internet 上的中小型网站中。由于其体积小、速度快、总体拥有成本低,尤其是开放源代码这一特点,许多中小型网站选择 MySQL 作为网站数据库。

MySQL 的主要特性如下。

(1) MySQL 是开源软件,使用 C 和 C++编写,并使用了多种编译器进行测试,保证了源代码的可移植性。

(2) 支持 AIX、FreeBSD、HP-UX、Linux、Mac OS、Novell Netware、OpenBSD、OS/2 Wrap、Solaris、Windows 等多种操作系统。

(3) 为多种编程语言提供了 API,这些编程语言包括 C、C++、Eiffel、Java、Perl、PHP、Python、Ruby 和 Tcl 等。

(4) 支持多线程,充分利用 CPU 资源。

(5) 采用优化的 SQL 查询算法,能够有效地提高查询速度。

(6) 既能够作为一个单独的应用程序应用在客户端/服务器网络环境中,也能够作为一个库嵌入其他的软件中提供多语言支持。

(7) 提供 TCP/IP、ODBC 和 JDBC 等多种数据库连接途径。

(8) 提供用于管理、检查、优化数据库操作的管理工具。

(9) 可以处理拥有上千万条记录的大型数据库。

4. Sybase

1984 年,Mark B. Hiffman 和 Robert Epstern 创建了 Sybase 公司,并在 1987 年推出了 Sybase 数据库产品。Sybase 主要有三种版本,一是 UNIX 操作系统下运行的版本,二是 Novell Netware 环境下运行的版本,三是 Windows NT 环境下运行的版本。

Sybase 数据库的特点如下。

(1) 开放性。由于采用了客户机/服务器结构,应用被分布在多台机器上运行。更进一步,运行在客户端的应用不必是 Sybase 公司的产品。对于一般的关系数据库,为了让其他

语言编写的应用能够访问数据库，提供了预编译。Sybase 数据库不只是简单地提供了预编译，而且公开了应用程序接口 DB-LIB，鼓励第三方编写 DB-LIB 接口。由于开放的客户 DB-LIB 允许在不同的平台使用完全相同的调用，因而使得访问 DB-LIB 的应用程序很容易从一个平台向另一个平台移植。

(2) 可编程性。通过提供存储过程，创建了一个可编程数据库。存储过程允许用户编写自己的数据库子例程。这些子例程是经过预编译的，因此不必在每次调用时都进行编译、优化、生成查询规划，因而查询速度要快得多。

(3) 事件驱动的触发器。触发器是一种特殊的存储过程，通过触发器可以启动另一个存储过程，从而确保数据库的完整性。

(4) 多线索化。Sybase 数据库的体系结构的另一个创新之处就是多线索化。一般的数据库都依靠操作系统来管理与数据库的连接。当有多个用户连接时，系统的性能会大幅度下降。Sybase 数据库不让操作系统来管理进程，而是把与数据库的连接当作自己的一部分来管理。此外，Sybase 的数据库引擎还代替操作系统来管理一部分硬件资源，如端口、内存、硬盘，绕过了操作系统这一环节，提高了性能。

5. DB2

DB2 是美国 IBM 公司开发的一套关系数据库管理系统，它主要的运行环境为 UNIX(包括 IBM 自家的 AIX)、Linux、IBM i(旧称 OS/400)、Z/OS，以及 Windows 服务器版本。

DB2 主要应用于大型应用系统，具有较好的可伸缩性，可支持从大型机到单用户环境，可应用于所有常见的服务器操作系统。

DB2 提供了高层次的数据利用性、完整性、安全性、可恢复性，以及小规模到大规模应用程序的执行能力，具有与平台无关的基本功能和 SQL 命令。

DB2 采用了数据分级技术，能够使大型机数据很方便地下载到 LAN 数据库服务器，使得客户机/服务器用户和基于 LAN 的应用程序可以访问大型机数据，并使数据库本地化、远程连接透明化。

DB2 以拥有一个非常完备的查询优化器而著称，其外部连接改善了查询性能，并支持多任务并行查询。

DB2 具有很好的网络支持能力，每个子系统可以连接十几万个分布式用户，可同时激活上千个活动线程，尤为适用于大型分布式应用系统。

3.2 初识 SQL Server 2012

Microsoft 公司发布的 Microsoft SQL Server 2012，是一款典型的关系数据库管理系统。Microsoft SQL Server 2012 以其强大的功能、简便的操作和可靠的安全性，赢得了众多用户的认可，其应用也越来越广泛。

3.2.1 SQL Server 的发展与版本

1. 了解 SQL Server 2012

SQL Server 2012 是一个重大的新产品版本，它不仅继承了 Microsoft 产品的一贯特点，而且在性能、可靠性、实用性、可编程性、易用性等方面都远远超过了以前的版本，并且在

原有版本的基础上推出了许多新的特性和关键的改进。SQL Server 2012 不仅可以有效地执行大规模联机事务，而且还能完成数据仓库和电子商务应用等许多具有挑战性的工作。

(1) 提供高可用性。

AlwaysOn 是 SQL Server 2012 全新的高可用灾难恢复技术，它能够保障企业应用的正常运转，减少意外死机时间，还可以帮助企业在故障发生时快速恢复，同时能够提供实时读写分离，保证应用程序性能最大化。另外，AlwaysOn 能够跨域部署，将主从结点分别部署在不同地域，帮助全球性企业实现数据库的高可用性。

(2) 全面支持云技术与平台。

作为新一代的数据平台产品，SQL Server 2012 不仅延续了现有平台的强大能力，而且全面支持云技术与平台，能够快速构建相应的解决方案，实现私有云和公有云之间数据的扩展与应用迁移。

(3) 大数据处理。

针对大数据以及数据仓库，SQL Server 2012 提供了从数 TB 到数百 TB 全面端到端的解决方案。作为微软的信息平台解决方案，SQL Server 2012 可以帮助企业用户突破性地快速实现各种数据体验，将大数据变成企业的洞察力和行动力。

2. 选择 SQL Server 2012 的版本

SQL Server 2012 提供了以下版本供不同用户进行选择。

(1) 企业版(Enterprise)。作为高级版本，企业版提供了全面的高端数据中心功能，性能极为快捷，虚拟化不受限制，还具有端到端的商业智能，可为关键任务工作负荷提供较高级别服务，支持最终用户访问深层数据。

(2) 商业智能版(Business Intelligence)。商业智能版提供了综合性平台，可支持组织构建和部署安全，可扩展和易于管理的 BI 解决方案。它提供了基于浏览器的数据浏览与可见性等卓越功能、功能强大的数据集成，以及增强的集成管理。

(3) 标准版(Standard)。标准版提供了基本数据管理和商业智能数据库，使部门和小型组织能够顺利运行其应用程序并支持将常用开发工具用于内部部署和云部署，有助于以最少的 IT 资源获得高效的数据库管理。

(4) Web 版(Web)。Web 版是针对运行于 Windows 服务器中要求高可用、面向 Internet Web 服务的环境而设计的。该版本为实现低成本、大规模、高可用性的 Web 应用或客户托管解决方案提供了必要的支持工具。

(5) 开发版(Developer)。开发版支持开发人员基于 SQL Server 构建任意类型的应用程序。它包括企业版的所有功能，但有许可限制，只能用作开发和测试系统，而不能用作生产服务器。开发版是构建和测试应用程序的开发人员的理想之选。

(6) 简易版(Express)。简易版是入门级的简易数据库，是学习和构建桌面及小型服务器数据驱动应用程序的理想选择。如果以后需要使用更高级的数据库功能，则可以将 SQL Server Express 无缝升级到其他更高端的 SQL Server 版本。

3.2.2 SQL Server 2012 的主要组件

整套的 SQL Server 2012 由一系列的服务组件组成，各服务组件有其特有的功能，用户可按照功能需要安装不同的服务组件，以达到最佳的性能和最少的费用。

1. SQL Server 2012 的服务器组件

服务器组件主要包括以下五部分。

(1) SQL Server 数据库引擎。

SQL Server 数据库引擎包括数据库引擎(用于存储、处理和保护数据的核心服务),复制、全文搜索、用于管理关系数据和 XML 数据的工具以及 Data Quality Services(DQS)服务器。

(2) Analysis Services。

Analysis Services(分析服务)包括用于创建和管理联机分析处理(OLAP)以及数据挖掘应用程序的工具。

(3) Reporting Services。

Reporting Services(报表服务)包括用于创建、管理和部署表格报表、矩阵报表、图形报表以及自由格式报表的服务器和客户端组件。Reporting Services 还是一个可用于开发报表应用程序的可扩展平台。

(4) Integration Services。

Integration Services(集成服务)是一组图形工具和可编程对象,用于移动、复制和转换数据。另外,还包括 Integration Services 的 Data Quality Services 组件。

(5) Master Data Services。

Master Data Services(主数据服务,MDS)是针对主数据管理的 SQL Server 解决方案,可以配置 MDS 来管理任何领域(产品、客户、账户)。MDS 中可包括层次结构、各种级别的安全性、事务、数据版本控制和业务规则,以及可用于管理数据的 Excel 的外接程序。

2. SQL Server 2012 的管理工具

(1) SQL Server Management Studio。

SQL Server Management Studio(SSMS)是用于访问、配置、管理和开发 SQL Server 组件的集成环境。SQL Server Management Studio 使各种技术水平的开发人员和管理员都能使用 SQL Server。

(2) SQL Server Configuration Manager(SQL Server 配置管理器)。

SQL Server 配置管理器为 SQL Server 服务、服务器协议、客户端协议和客户端别名提供基本配置管理。

(3) SQL Server Profiler。

SQL Server Profiler 提供了一个图形用户界面,用于监视数据库引擎实例或 Analysis Services 实例。

(4) 数据库引擎优化顾问。

数据库引擎优化顾问可以协助创建索引、索引视图和分区的最佳组合。

(5) 数据质量客户端。

数据质量客户端提供了一个非常简单和直观的图形用户界面,用于连接到 DQS 数据库并执行数据清理操作。另外,它还允许用户集中监视在数据清理操作过程中执行的各项活动。

(6) SQL Server 数据工具。

SQL Server 数据工具(SSDT)提供 IDE 以便为以下商业智能组件生成解决方案:Analysis Services、Reporting Services 和 Integration Services。SSDT 还包含“数据库项

目”，为数据库开发人员提供集成环境，以便在 Visual Studio 内为任何 SQL Server 平台（无论是内部还是外部）执行其所有数据库设计工作。数据库开发人员可以使用 Visual Studio 中功能增强的服务器资源管理器，轻松创建或编辑数据库对象和数据或执行查询。

（7）连接组件。

安装用于客户端和服务器之间通信的组件，以及用于 DB-Library、ODBC 和 OLE DB 的网络库。

3. SQL Server 联机文档

SQL Server 2012 提供了大量的联机文档，用户可以查询到许多有价值的信息。一个优秀的 SQL Server 管理员和应用程序员应能熟练使用联机文档。

3.2.3 SQL Server 2012 管理平台

1. SQL Server Management Studio

SQL Server Management Studio 是一个集成环境，用于访问、配置、管理和开发 SQL Server 的所有组件。SQL Server Management Studio 组合了大量图形工具和丰富的脚本编辑器，使各种技术水平的开发人员和管理员都能访问 SQL Server。

启动 SQL Server Management Studio 的操作步骤如下。

（1）单击任务栏上的“开始”按钮，依次选择“所有程序”→Microsoft SQL Server 2012→SQL Server Management Studio 菜单命令，打开“连接到服务器”对话框，如图 3.2 所示。

图 3.2 “连接到服务器”对话框

（2）在“连接到服务器”对话框中，首先选择服务器类型。SQL Server Management Studio 提供了“数据库引擎”、Analysis Services、Reporting Services、Integration Services 四种服务器类型，这里选择“数据库引擎”服务器类型。接着选择服务器名称和身份验证方式，然后单击“连接”按钮。

（3）连接到服务器后，会进入 SQL Server Management Studio 窗口。SQL Server Management Studio 窗口由菜单栏、工具栏、对象资源管理器、查询编辑器、查询结果窗格等部分组成，如图 3.3 所示。

图 3.3 SQL Server Management Studio 窗口

(4) 单击“新建查询”按钮,新建“查询编辑器”。查询编辑器是非常实用的工具,主要用于输入、执行、保存 Transact-SQL 命令,实现数据库的查询管理。可以在查询编辑器中输入一条或多条 Transact-SQL 命令,也可以从对象资源管理器直接将对象拖曳到查询编辑器。单击工具栏中的“执行”按钮 或按 F5 键,可以执行 SQL 查询语句,并在查询结果窗格显示查询结果。

SQL 语句可以被保存或重新打开,SQL 文件的扩展名为.sql。保存 SQL 文件的步骤是:在菜单栏中选择“文件”→“保存”命令,选择文件存放地址,输入文件名。

2. SQL Server 配置管理器

SQL Server 配置管理器(Configuration Manager)负责配置管理各种 SQL Server 服务、网络配置协议、客户端协议和客户端别名,可以停止、启动或暂停各种 SQL Server 服务。

单击任务栏上的“开始”按钮,依次选择“所有程序”→Microsoft SQL Server 2012→“配置工具”→“SQL Server 配置管理器”命令,即可打开 Sql Server Configuration Manager 窗口,如图 3.4 所示。

图 3.4 SQL Server 配置管理器窗口

在 SQL Server 配置管理器中暂停、停止或启动 SQL Server 服务的方法如下：在 SQL Server 配置管理器窗口中，在左边的目录树中选择“SQL Server 服务”选项，在右边的服务内容列表区中选择 SQL Server(MSSQL SERVER)服务并右击，在弹出的快捷菜单中选择相应的命令即可启动、停止、暂停或继续选定的服务，如图 3.5 所示。

图 3.5　暂停、停止或启动 SQL Server 服务

SQL Server 服务暂停一般是在需要临时关闭数据库时进行。暂停服务后，用户已经提交的任务将继续执行，新的用户连接请求将被拒绝，暂停结束后可以恢复执行。

SQL Server 服务停止是从内存中清除所有有关的 SQL Server 服务进程，所有与之连接的用户提交的任务将停止，新的用户也不能登录。

在服务已经停止或暂停的情况下，需要相关服务时应启动 SQL Server 服务。

3.2.4　SQL 和 Transact-SQL

SQL 是结构化查询语言(Structured Query Language)的缩写，其功能包括数据查询、数据定义、数据操纵和数据控制等。SQL 简单易学，功能齐全，目前已成为关系数据库系统中使用最为广泛的语言。Transact-SQL 是 Microsoft SQL Server 使用的一种结构化查询语言。

1. 了解 SQL

SQL 是 1974 年由 Boyce 和 Chamberlin 提出的，并在 IBM 公司研制的关系数据库管理系统 System R 中应用。1986 年，美国国家标准局(ANSI)的数据库委员会批准了 SQL 作为关系数据库语言的美国标准。1987 年，国际标准化组织(ISO)将其采纳为国际标准。目前流行的关系数据库管理系统，如 Oracle、Sybase、SQL Server、DB2、MySQL 等都采用了 SQL 标准，而且很多数据库都对 SQL 语句进行了再开发和扩展。

SQL 具有简单、易学、综合、一体等鲜明的特点，主要有以下五方面。

(1) 一体化的语言。

SQL 集数据定义语言(DDL)、数据操纵语言(DML)、数据查询语言(DQL)、数据控制语言(DCL)的功能于一体，语言风格统一，可以独立完成数据库生命周期中的全部活动。

① 数据定义语言。用于定义数据的逻辑结构以及数据项之间的关系，包括创建、修改和删除表、索引、视图等。

② 数据操纵语言。用于更改数据库中的数据，包括增加新数据、删除旧数据、修改已有数据等。

③ 数据查询语言。用于按一定的查询条件从数据库对象中检索符合条件的数据。

④ 数据控制语言。用于控制对数据库中数据的操作，包括基本表和视图等对象的授权、完整性规则的描述、事务开始和结束控制语句等。

(2) 高度非过程化。

非关系数据模型的数据操纵语言是“面向过程”的，必须制定存取路径，而 SQL 只要用户提出“做什么”，无须了解存取路径，存取路径的选择以及 SQL 的操作过程由系统自动完成。

(3) 面向集合的操作方式。

非关系数据模型采用面向记录的操作方式，操作对象是一条记录；而 SQL 采用集合操作方式，不仅操作对象、查找结果可以是元组的集合，而且一次插入、删除、更新操作的对象也可以是元组的集合。

(4) 以同一种语法结构提供多种使用方式。

SQL 是独立的语言，能够独立地用于联机交互的使用方式；SQL 又是嵌入式语言，能够嵌入高级语言(如 C、C++、Java)程序中，供程序员设计程序时使用。

(5) 语言简洁，易学易用。

SQL 功能极强，但由于设计巧妙、语言十分简洁，完成核心功能只用了 9 个动词，如表 3.1 所示。SQL 语法接近英语，因此容易学习、容易使用。

表 3.1　SQL 的动词

SQL	动　　词
数据定义语言	CREATE,DROP,ALTER
数据操纵语言	INSERT,UPDATE,DELETE
数据查询语言	SELECT
数据控制语言	GRANT,REVOKE

2. Transact-SQL——SQL Server 中的 SQL

SQL 是一种数据库标准语言。在每个具体的数据库系统中，都对这种标准的 SQL 有一些功能上的调整(一般是扩展)，语句格式也有个别变化，从而形成了各自不完全相同的 SQL 版本。Transact-SQL(可简写为 T-SQL)就是 SQL Server 中使用的 SQL 版本。

Transact-SQL 对 SQL 的扩展主要包含以下三方面。

(1) 增加了流程控制语句，SQL 作为一种功能强大的结构化查询语言并没有包含流程控制语句，因此不能单纯使用 SQL 构造出分支或循环结构的程序。Transact-SQL 在这方面进行了扩展，增加了块语句、分支判断语句、循环语句、跳转语句等。

(2) 加入了局部变量、全局变量等许多新概念，可以写出更复杂的查询语句。

(3) 增加了新的数据类型，处理能力更强。

3.3 数据库的管理

在 SQL Server 中，数据库是存放数据的容器，在设计一个应用系统时，必须先设计数据库。数据库中的数据及其相关信息通常被存储在一个或多个文件中，而数据库管理系统为用户和数据库应用程序提供统一的接口来访问和控制数据，使得用户不需要直接访问数据库文件。

3.3.1 SQL Server 2012 数据库组成

数据库是 SQL Server 服务器管理的基本单位。从逻辑上看，SQL Server 数据库由数据表、视图、存储过程、触发器等逻辑组件组成，这些逻辑组件被称为数据库对象。从物理存储角度来看，数据库中的各种信息是以文件为单位存放在存储设备上的。

1. SQL Server 中的数据库对象

SQL Server 数据库对象通常用于提高数据库性能、支持特定的数据活动、保持数据完整性或保证数据的安全性。SQL Server 中常用的数据库对象如表 3.2 所示。

表 3.2 SQL Server 中常用的数据库对象

对　　象	作　　用
表	数据库中数据的实际存放处所
视图	定制复杂或常用的查询，以便用户使用；限定用户只能查看表中的特定行或列；为用户提供统计数据而不展示细节
索引	加快从表或视图中检索数据的效率
存储过程	提高性能；封装数据库的部分或全部细节；帮助在不同的数据库应用程序之间实现一致的逻辑
约束和触发器	确保数据库的数据完整性；强制执行业务规则
登录、用户、角色和组	保障数据安全的基础

2. SQL Server 中的数据库文件和文件组

在 SQL Server 中，数据库是由数据文件和事务日志文件组成的，一个数据库至少包含一个数据文件和一个事务日志文件。

(1) 数据文件。

数据文件(Database File)是存放数据库数据和数据库对象的文件。一个数据库可以有一个或多个数据文件，每个数据文件只属于一个数据库。

数据库的各个数据文件中，有且仅有一个数据文件被定义为主数据文件(Primary Database File)，其扩展名为.mdf，用来存储数据库的启动信息和部分或全部数据。其他数据文件被称为次数据文件(Secondary Database File)，扩展名为.ndf，用来存储主数据文件未存储的其他数据。

(2) 事务日志文件。

事务日志文件(Transaction Log File)是用来记录数据库更新信息(如使用 INSERT、UPDATE、DELETE 等语句对数据进行更改的操作)的文件，这些更新信息(日志)可用来恢复数据库。事务日志文件的扩展名为.ldf。每个数据库可以有一个或多个事务日志文件。

(3) 文件组。

文件组(File Group)是 SQL Server 中一个或多个数据文件的命名集合,数据库由一个或者多个文件组构成。SQL Server 通过文件组对数据文件进行管理,文件组构成了分配或用于数据库管理的单个单元。

主数据文件所在的文件组被称为主文件组,主文件组有且仅有一个。其余的文件组被称为次文件组,次文件组根据需要可以设置零到多个。

一个数据库的多个文件组中,有一个被指定为默认文件组(DEFALUT),创建数据库对象时,如果用户未指明将其放在哪一个文件组中,则系统将它放在默认文件组中。数据库首次创建时,主文件组是默认文件组。

(4) 文件和文件组的设计规则。

① 一个数据文件只能存在于一个文件组中,一个文件组也只能被一个数据库使用。

② 事务日志文件不分组,它不能属于任何文件组。

(5) 设计建议。

① 大多数数据库在只有单个数据文件和单个事务日志文件的情况下性能良好。

② 如果使用多个数据文件,需为附加文件创建第二个文件组,并将其设置为默认文件组。这样,主数据文件将只包含系统表和对象。

③ 若要使性能最大化,需在尽可能多的不同可用磁盘上创建文件或文件组,将争夺空间最激烈的对象置于不同的文件组中。

④ 使用文件组将对象放置在特定的物理磁盘上。

⑤ 将在同一连接查询中使用的不同表置于不同的文件组中,采用并行磁盘 I/O 对连接数据进行搜索,性能将得以改善。

⑥ 将最常访问的表和属于这些表的非聚集索引置于不同的文件组中。文件位于不同的物理磁盘上,并且采用并行 I/O,性能将得以改善。

⑦ 勿将事务日志文件置于已有其他文件和文件组的同一物理磁盘上。

3. SQL Server 2012 的系统数据库

SQL Server 2012 中有两类数据库:系统数据库和用户数据库。系统数据库存储有关 SQL Server 的系统信息,它们是 SQL Server 管理数据库的依据。如果系统数据库遭到破坏,那么 SQL Server 将不能正常启动。在安装了 SQL Server 2012 的系统中共创建四个可见的系统数据库和一个隐藏的系统数据库,如表 3.3 所示。

表 3.3 SQL Server 2012 中的系统数据库

系统数据库	说　明
master 数据库	记录 SQL Server 实例的所有系统级信息
msdb 数据库	用于 SQL Server 代理计划警报和作业
model 数据库	用作 SQL Server 实例上创建的所有数据库的模板。对 model 数据库进行的修改(如数据库大小、排序规则、恢复模式和其他数据库选项)将应用于以后创建的数据库中
resource 数据库	一个只读的、隐藏的数据库,包含 SQL Server 的系统对象。系统对象在物理上保留在 resource 数据库中,但在逻辑上显示在每个数据库的 sys 架构中
tempdb 数据库	一个工作空间,用于保存临时对象或中间结果集。一旦关闭 SQL Server,tempdb 数据库保存的内容将自动消失

3.3.2 数据库对象的标识符

为了提供完善的数据库管理机制，SQL Server 设计了严格的命名规则。在创建或引用数据库对象，如表、索引、约束等时，必须遵守 SQL Server 的命名规则，否则有可能发生一些难以预料和检查的错误。

1. 标识符分类

标识符是指 SQL Server 中的对象的名称，包括服务器、数据库、表、视图、列、索引、触发器、存储过程和约束等的名称。对象的标识符一般在创建对象时定义，作为引用对象的工具使用。

例如，下面的 SQL 语句创建了一个表，表的名字 student 是一个标识符，表中定义了两列，列的名字分别是 id 和 name，它们都是合法的标识符。

```
CREATE TABLE student
 (  id  INT  PRIMARY  KEY,
    name  VARCHAR(20)
)
```

SQL Server 有两种类型的标识符：**规则标识符(Regular Identifier)和界定标识符(Delimited Identifier)**。

(1) 规则标识符。

规则标识符严格遵守标识符有关格式的规定，因此在 T-SQL 语句中凡是规则标识符都不必使用界定符，即方括号([ ])和双引号(" ")来进行界定。如上述例子中使用的表名 student 就是一个规则标识符，在 student 上不必添加界定符。

(2) 界定标识符。

界定标识符是那些使用了方括号或双引号界定符号来进行位置限定的标识符，使用了界定标识符之后，既可以遵守标识符命名规则，也可以不遵守标识符命名规则。例如，

```
SELECT * FROM [my table] WHERE [order] = 10
```

在这个例子中，必须使用界定标识符，因为 FROM 子句中的标识符 my table 中含有空格，而 WHERE 子句中的标识符 order 是系统保留字。这两个标识符都不遵守标识符命名规则，必须使用界定符，否则无法通过代码编译。

2. 标识符的命名规则

SQL Server 标识符的命名遵循以下规则。

(1) 标识符包含的字符数必须为 1～128，不区分大小写。

(2) 标识符的首字符可以是：所有在统一码(Unicode)2.0 标准中规定的字符(如汉字以及其他一些语言字符)、26 个英文字母 a～z 和 A～Z，以及“_”“@”或“#”。

以某些特殊符号开头的标识符在 SQL Server 系统中具有特定的含义，如以“#”开头的标识符表示这是一个临时表或存储过程；以“##”开头的标识符表示这是一个全局的临时数据库对象；以“@”开头的标识符表示这是一个局部变量或是一个函数的参数；Transact-SQL 的全局变量以标志“@@”开头，为避免同这些全局变量混淆，建议不要使用“@@”作为标识符的开始。

(3) 标识符首字符后的字符可以是：所有在统一码(Unicode)2.0 标准中规定的字符、字母、数字，以及“_”“@”“$”“#”。

(4) 标识符不允许是 Transact-SQL 的保留字，标识符内部不允许有空格。

例如，以下都是合法的标识符。

```
student_name,学生姓名,@sum,#TempTable,price1
```

以下都是不合法的标识符。

```
Select,zhong guo,3level
```

其中，Select 不合法的原因是 Select 为 Transact-SQL 的保留字，zhong guo 不合法的原因是出现了空格；3level 不合法的原因是数字字符不能作为首字符。

3. 数据库对象的引用方式

在一个数据库中创建了一个数据库对象后，数据库对象的全名应该由服务器名、数据库名、所有者名和对象名 4 部分组成，格式如下：

```
[[[[server].][database].][owner_name].]object_name
```

在实际引用对象时，如果对象所在的服务器、数据库、所有者为当前的服务器、数据库和所有者，则可省略对应的部分，只留下空白的位置，如表 3.4 所示。

表 3.4 数据库对象的引用方式

引 用 方 式	含 义
server1.studentdb.dbo.T1	server1 服务器下，studentdb 数据库中，用户 dbo 所创建的表 T1
.studentdb.dbo.T1	当前服务器下，studentdb 数据库中，用户 dbo 所创建的表 T1
studentdb.dbo.T1	
..dbo.T1	当前服务器下，当前数据库中，用户 dbo 所创建的表 T1
.dbo.T1	
dbo.T1	
...T1	当前服务器下，当前数据库中，当前用户所创建的表 T1
..T1	
.T1	
T1	

3.3.3 数据库的创建

在 SQL Server 中，可以使用 SQL Server Management Studio 和 Transact-SQL 语句来创建数据库。

1. 使用 SQL Server 管理平台创建数据库

使用 SQL Server 管理平台创建数据库的步骤如下。

(1) 打开 SQL Server Management Studio，在“对象资源管理器”面板中展开服务器并右击“数据库”结点，在弹出的快捷菜单中选择“新建数据库”命令，如图 3.6 所示。

(2) 打开如图 3.7 所示的“新建数据库”窗口，在“常规”页的“数据库名称”文本框中输入新数据库名称(如学生成绩管理系统数据库)，在“所有者”下拉列表框中选择数据库所有

者，默认值为系统登录者。

图 3.6　选择“新建数据库”命令

图 3.7　“新建数据库”中的“常规”页面

在“数据库文件”列表区可以添加或删除数据库的数据文件和事务日志文件，修改数据文件或事务日志文件的逻辑名称、所属文件组、存放位置、文件初始大小、最大大小和增长率等内容。

(3) 在“新建数据库”窗口左侧的“选项页”列表区中选择“选项”命令，打开如图 3.8 所

示的“选项”页面，在其中可对数据库的排序规则、恢复模式等内容进行设置。

图 3.8 “新建数据库”中的“选项”页面

（4）在“新建数据库”窗口左侧的“选项页”列表区中选择“文件组”命令，打开如图 3.9 所示的“文件组”页面，在其中完成添加、删除文件组，设置默认文件组等操作。

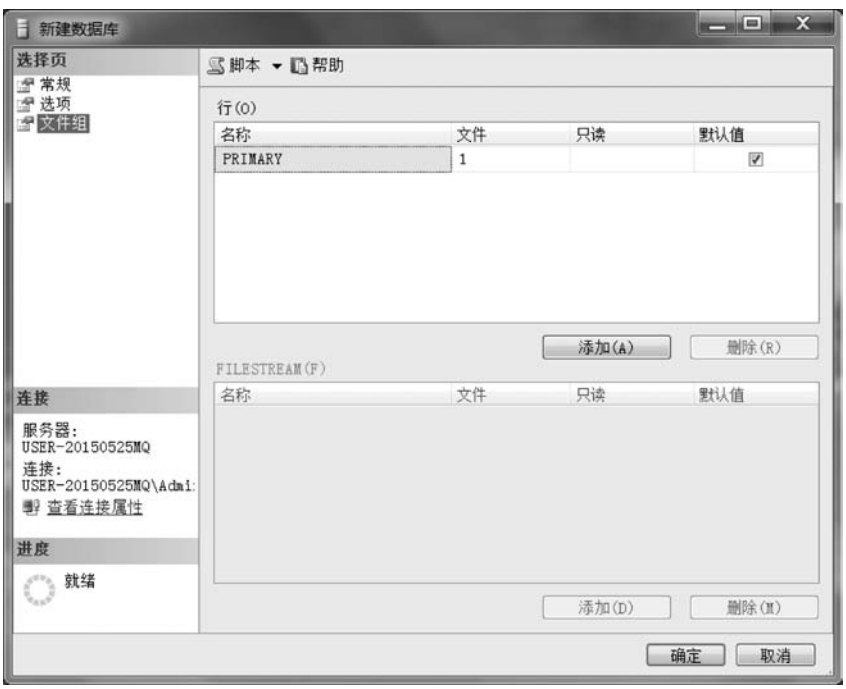

图 3.9 “新建数据库”中的“文件组”页面

(5) 所有的设置完成后单击“确定”按钮,完成数据库的创建工作。返回 SQL Server Management Studio 窗口,在“对象资源管理器”面板中的“数据库”结点下有了新建的数据库,如图 3.10 所示。

图 3.10　学生成绩管理系统数据库

注意:数据库创建完成后,有可能在“对象资源管理器”面板中看不到,此时可以在“对象资源管理器”中右击“数据库”结点,在弹出的快捷菜单中选择“刷新”命令,即可看到新建的数据库。

2. 使用 Transact-SQL 语句创建数据库

在 SQL Server Management Studio 窗口的工具栏上单击“新建查询”按钮,在右侧的查询编辑器的编辑区中,使用 CREATE DATABASE 语句即可创建数据库以及数据库文件。CREATE DATABASE 语句的基本语法格式如下:

```
CREATE DATABASE database_name
[ ON  [ PRIMARY ]  [ <filespec> [,...n] ]  [ , <filegroup> [,...n] ]
 [ LOG  ON { <filespec> [,...n] } ]
]
```

各项的含义如下。

(1) database_name 是新建数据库的名称,该名称需符合标识符的命名规则。

(2) ON:用于指定显式定义数据文件。当后面是以逗号分隔的、用以定义主文件组的数据文件的<filespec>项列表时,需要使用 ON。主文件组的文件列表可后跟以逗号分隔的、用以定义次文件组及其文件的<filegroup>项列表(可选)。

(3) PRIMARY:用于指定关联的<filespec>列表定义主数据文件,在主文件组的<filespec>项中指定的第一个文件将成为主文件。如果没有指定 PRIMARY,那么 CREATE DATABASE 语句中列出的第一个文件将成为主文件。

(4) LOG ON:用于指定显式定义事务日志文件。LOG ON 后跟以逗号分隔的、用以定义事务日志文件的<filespec>项列表。如果没有指定 LOG ON,系统将自动创建一个事务日志文件。注意,当没有 ON 子句时,也不能有 LOG ON 子句。

(5) <filespec>:用于对数据库的数据文件或事务日志文件的定义说明,其语法格式如下:

```
<filespec> ::=
```

```
(  NAME = logical_file_name,
  FILENAME = 'os_file_name'
  [ , SIZE = size [ KB | MB | GB | TB ] ]
  [ , MAXSIZE = { max_size [ KB | MB | GB | TB ] | UNLIMITED } ]
  [ , FILEGROWTH = growth_increment [ KB | MB | GB | TB | % ] ]
)
```

其中，logical_file_name 为文件在 SQL Server 中使用的逻辑名称，它在数据库中是唯一的，必须符合标识符的命名规则。

os_file_name 是文件由操作系统使用的带路径的文件名(或称为物理名称)。注意，该文件名是一个字符串，需使用单引号括起来。

size 用于指定文件的初始大小，单位可以是 KB、MB、GB 或 TB，默认为 MB。

max_size 用于指定文件可增大到的最大大小，UNLIMITED 指不限制文件的最大大小。

growth_increment 用于指定文件需要新空间时为文件添加的空间量。

(6) < filegroup >：对数据库文件组的定义说明，其语法格式如下：

```
<filegroup> ::=
FILEGROUP  filegroup_name  [DEFALUT]  <filespec>[,...n]
```

其中，filegroup_name 为文件组的逻辑名称，它在数据库中是唯一的，不能是系统提供的名称 PRIMARY，其名称必须符合标识符的命名规则。

DEFALUT 指定文件组为数据库的默认文件组。

注意：

(1) 上述语法格式只给出了 CREATE DATABASE 语句的基本形式，完整的语法还包括参数设置、附加数据库、创建数据库快照、指定数据库的排序规则等子句，具体可参考 SQL Server 的联机文档。

(2) Transact-SQL 语句不区分大小写，为了清晰，一般用大写表示系统保留字，用小写表示用户自定义的标识符。一条 Transact-SQL 语句可以写在一行上，也可以写在多行上。

(3) 在本书中，经常需要书写一些 Transact-SQL 语句，其语句的语法格式中使用的描述符及其含义可参见表 3.5。

表 3.5　Transact-SQL 语法格式所用的符号及其含义

描　述　符	含　　义
{}	表示该项是必选项
[]	表示该项可以省略，省略时该参数取默认值
|	表示在其左右两边任选一项，相当于 OR 的意思
[,...n]	表示前面的项可重复多次，项之间用逗号分隔
<标签>	语法块名称，对可在语句中的多个位置使用的过长语法单元部分进行标记
<标签>::=	对语法格式中出现的语法块进行定义

例 3-1　创建名称为“学生成绩管理系统数据库”的数据库，不指定文件。

```
CREATE DATABASE 学生成绩管理系统数据库
```

本例中，未对数据文件和事务日志文件做出说明，系统将根据默认值自动创建一个主数据文件和一个事务日志文件。

例 3-2 创建名称为 db1 的数据库，指定其主数据文件的逻辑名称和带路径的物理文件名，不指定主数据文件的 SIZE、MAXSIZE 和 FILEGROWTH 的值。

```
CREATE DATABASE db1
ON
(  NAME = db1_data,
   FILENAME = 'C:\DataBase\db1.mdf'
)
```

本例中，系统根据默认值指定主数据文件的 SIZE、MAXSIZE 和 FILEGROWTH 的值。本例未对事务日志文件做出说明，系统将根据默认值自动创建一个事务日志文件。

注意：只有当 C 盘下存在文件夹 DataBase 时，此语句才能执行成功。

例 3-3 创建名称为 db2 的数据库，指定其主数据文件的各项参数。

```
CREATE DATABASE db2
ON
(  NAME = db2_data,
   FILENAME = 'C:\DataBase\db2.mdf',
   SIZE = 5,
   MAXSIZE = 100,
   FILEGROWTH = 10
)
```

本例中，对主数据文件的 SIZE、MAXSIZE、FILEGROWTH 的值做出了指定，默认以 MB 为单位进行分配。

例 3-4 创建名称为 db3 的数据库，指定数据文件和事务日志文件。

```
CREATE DATABASE db3
ON
(  NAME = db3_data,
   FILENAME = 'C:\DataBase\db3data.mdf',
   SIZE = 10000KB,
   MAXSIZE = 500000KB,
   FILEGROWTH = 5 %
)
LOG ON
(  NAME = db3_log,
   FILENAME = 'C:\DataBase\db3log.ldf',
   SIZE = 5,
   MAXSIZE = 25,
   FILEGROWTH = 5
)
```

本例中，因为没有使用关键字 PRIMARY，第一个数据文件 db3_data 将成为主数据文件。

例 3-5 创建名称为 db4 的数据库，指定多个数据文件和事务日志文件。

```
CREATE DATABASE db4
```

```
ON  PRIMARY
(  NAME = db4_data1,
   FILENAME = 'C:\DataBase\db4data1.mdf',
   SIZE = 20,
   MAXSIZE = 100,
   FILEGROWTH = 20
),
(  NAME = db4_data2,
   FILENAME = 'C:\DataBase\db4data2.ndf',
   SIZE = 20,
   MAXSIZE = 100,
   FILEGROWTH = 20
),
(  NAME = db4_data3,
   FILENAME = 'C:\DataBase\db4data3.ndf',
   SIZE = 20,
   MAXSIZE = 100,
   FILEGROWTH = 20
)
LOG ON
(  NAME = db4_log1,
   FILENAME = 'C:\DataBase\db4log1.ldf',
   SIZE = 5,
   MAXSIZE = 25,
   FILEGROWTH = 5
),
(  NAME = db4_log2,
   FILENAME = 'C:\DataBase\db4log2.ldf',
   SIZE = 5,
   MAXSIZE = 25,
   FILEGROWTH = 5
)
```

本例为数据库创建了三个数据文件和两个事务日志文件。主数据文件是列表中的第 1 个文件，并使用 PRIMARY 关键字显示指定。注意 FILENAME 选项中所使用的文件扩展名：主数据文件使用.mdf，次数据文件使用.ndf，事务日志文件使用.ldf。

例 3-6 使用文件组创建名称为 db5 的数据库。

```
CREATE DATABASE db5
ON
/*默认的 PRIMARY 文件组,存放在 C 盘*/
PRIMARY
(  NAME = db5_data1,
   FILENAME = 'C:\DataBase\db5data1.mdf',
   SIZE = 10,
   MAXSIZE = 50,
   FILEGROWTH = 15
),
(  NAME = db5_data2,
   FILENAME = 'C:\DataBase\db5data2.ndf',
```

```
    SIZE = 10,
    MAXSIZE = 50,
    FILEGROWTH = 15
),
/* db5_group1 文件组,存放在 D 盘 */
FILEGROUP db5_group1
(  NAME = db5_data3,
    FILENAME = 'D:\DataBase\db5data3.ndf',
    SIZE = 20,
    MAXSIZE = 100,
    FILEGROWTH = 20
),
(  NAME = db5_data4,
    FILENAME = 'D:\DataBase\db5data4.ndf',
    SIZE = 20,
    MAXSIZE = 100,
    FILEGROWTH = 20
),
/* db5_group2 文件组,存放在 E 盘 */
FILEGROUP db5_group2
(  NAME = db5_data5,
    FILENAME = 'E:\DataBase\db5data5.ndf',
    SIZE = 20,
    MAXSIZE = 100,
    FILEGROWTH = 20
),
(  NAME = db5_data6,
    FILENAME = 'E:\DataBase\db5data6.ndf',
    SIZE = 20,
    MAXSIZE = 100,
    FILEGROWTH = 20
)
LOG ON
(  NAME = db5_log,
    FILENAME = 'C:\DataBase\db5log.ldf',
    SIZE = 5,
    MAXSIZE = 25,
    FILEGROWTH = 5
)
```

注意：此命令将文件分别存到不同盘中，因此 C、D、E 盘都要先建 DataBase 文件夹。

例 3-7 使用系统存储过程 sp_helpdb 查看“学生成绩管理系统数据库”的基本信息。

```
EXEC sp_helpdb 学生成绩管理系统数据库
```

本例的执行结果如图 3.11 所示。

注意：

(1) 存储过程是 SQL Server 服务器上一组预编译的 Transact-SQL 语句，用于完成某项任务，它可以接收参数，返回状态值和参数值。系统存储过程是指由 SQL Server 系统提供的存储过程。

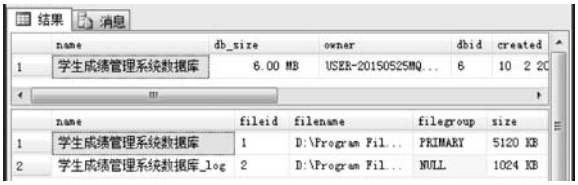

图 3.11　例 3-7 的执行结果

(2) 存储过程的执行方法为

```
[EXEC[UTE]] procedure_name parameter [,...n]
```

说明：EXEC 为关键字，表示执行后面的存储过程，它既可以是完整的单词 EXECUTE，也可以写成 EXEC，当此语句为批处理的第一条语句时，也可以省去不写。

3.3.4　数据库的修改

创建数据库后，可以对其原始定义进行修改。与创建数据库一样，在 SQL Server 中，可以使用 SQL Server Management Studio 和 Transact-SQL 语句两种方式来修改数据库。

1. 使用 SQL Server 管理平台修改数据库

对于已经建立的数据库，可以使用 SQL Server 管理平台来查看或修改数据库，具体步骤如下。

(1) 打开 SQL Server Management Studio，在“对象资源管理器”面板中右击要修改的数据库名称，在弹出的快捷菜单中选择“属性”命令，打开如图 3.12 所示的“数据库属性”窗口。

图 3.12　“数据库属性”窗口

（2）在“数据库属性”窗口的“常规”选项页中显示了当前数据库的基本信息，包括数据库的名称状态、所有者、创建日期、大小、可用空间、用户数以及备份和维护等内容，本页面的信息不能修改。

（3）“数据库属性”窗口的“文件”选项页显示了当前数据库的文件信息，如图 3.13 所示，可修改数据文件或事务日志文件的逻辑名称、所属文件组、存放位置、文件初始大小、最大大小和增长率等内容，还可以添加和删除文件。

图 3.13 “数据库属性”中的“文件”页面

（4）“数据库属性”窗口的“文件组”选项页显示数据库文件组的信息，用户可以在此页面上查看或修改文件组信息。

（5）“数据库属性”窗口的“选项”选项页显示数据库的选项信息，包括恢复选项、游标选项、杂项、状态选项和自动选项等。

（6）“数据库属性”窗口的“权限”选项页显示数据库的使用权限。

（7）“数据库属性”窗口的“扩展属性”选项页可以添加文本、输入掩码和格式规则，将其作为数据库对象或数据库本身的属性。

（8）“数据库属性”窗口的“镜像”选项页显示数据库的镜像设置属性，用户可以设置主体服务器和镜像服务器的网络地址及运行方式。

（9）“数据库属性”窗口的“事务日志传送”选项页显示数据库的日志传送配置信息，用户可以为当前数据库设置事务日志备份、辅助数据库及监视服务器。

2. 使用 Transact-SQL 语句修改数据库

在 SQL Server 中使用 ALTER DATABASE 语句来修改数据库。ALTER DATABASE 语句的基本语法格式如下：

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

各项的含义如下。

(1) database_name 是要修改的数据库的名称。

(2) MODIFY NAME 用于重命名数据库,new_database_name 是数据库的新名称。

(3) ADD FILE 用于向数据库添加新数据文件,其后的< filespec >的语法与 CREATE DATABASE 语句中的< filespec >的语法相同。TO FILEGROUP 用于将新添加的数据文件放到指定的文件组中。

(4) REMOVE FILE 用于删除数据文件,logical_file_name 是文件的逻辑文件名。注意,只有在文件为空的时候才能删除。

(5) MODIFY FILE 用于修改文件,其后的< filespec >的语法格式为如下:

```
<filespec> ::=
(  NAME = logical_file_name
  [ , NEWNAME = new_logical_name ]
  [ , FILENAME = 'os_file_name' ]
  [ , SIZE = size [ KB | MB | GB | TB ] ]
  [ , MAXSIZE = { max_size [ KB | MB | GB | TB ] | UNLIMITED } ]
  [ , FILEGROWTH = growth_increment [ KB | MB | GB | TB | % ] ]
)
```

其中,new_logical_name 是文件的新的逻辑名称,其余各项的含义与 ADD FILE 后的< filespec >中的各项相同。

(6) ADD LOG FILE 用于在向数据库中添加新的事务日志文件,其后的< filespec >的语法与 ADD FILE 后的< filespec >的语法相同。

(7) ADD FILEGROUP 用于向数据库添加新的文件组,filegroup_name 是文件组的名称。

(8) REMOVE FILEGROUP 用于删除文件组,filegroup_name 是文件组的名称,只有当文件组为空时才能删除。

(9) MODIFY FILEGROUP 用于修改文件组,filegroup_name 是文件组的名称。DEFAULT 是指设置该文件组为默认文件组; NAME = new_filegroup_name 是为文件组重命名; < filegroup_updatability_option >用于对文件组设置只读或读写属性,语法格式如下:

```
<filegroup_updatability_option> ::=
{ { READONLY | READWRITE } | { READ_ONLY | READ_WRITE } }
```

例 3-8 将数据库 db1 的名称改为 newdb。

```
ALTER DATABASE db1
MODIFY NAME = newdb
```

例 3-9 向数据库 db2 中添加一个文件组。

```
ALTER DATABASE db2
ADD FILEGROUP db2_group1
```

例 3-10 向数据库 db2 中添加两个数据文件,并将这两个文件放入文件组 db2_group1 中。

```
ALTER DATABASE db2
ADD FILE
(  NAME = db2_data2,
   FILENAME = 'C:\DataBase\db2data2.ndf'
),
(  NAME = db2_data3,
   FILENAME = 'C:\DataBase\db2data3.ndf'
) TO FILEGROUP db2_group1
```

例 3-11 修改数据库 db2 中的数据文件 db2_data2 的名称为 db2_newdatafile。

```
ALTER DATABASE db2
MODIFY FILE
(  NAME = db2_data2,
   NEWNAME = db2_newdatafile
)
```

例 3-12 删除数据库 db2 中的数据文件 db2_newdatafile。

```
ALTER DATABASE db2
REMOVE FILE db2_newdatafile
```

例 3-13 将数据库 db2 中的文件组 db2_group1 设置为默认文件组。

```
ALTER DATABASE db2
MODIFY FILEGROUP db2_group1 DEFAULT
```

3.3.5 数据库的删除

在 SQL Server 中,可以使用 SQL Server Management Studio 和 Transact-SQL 语句来删除数据库。

1. 使用 SQL Server 管理平台删除数据库

打开 SQL Server Management Studio,在“对象资源管理器”中右击要删除的数据库,在弹出的快捷菜单中选择“删除”命令,打开如图 3.14 所示的“删除对象”窗口。单击“确定”按钮完成数据库的删除。数据库的数据文件和事务日志文件也将同时被删除。

2. 使用 Transact-SQL 语句删除数据库

在 SQL Server 中,可以使用 DROP DATABASE 语句来删除数据库。DROP DATABASE 语句的语法格式如下:

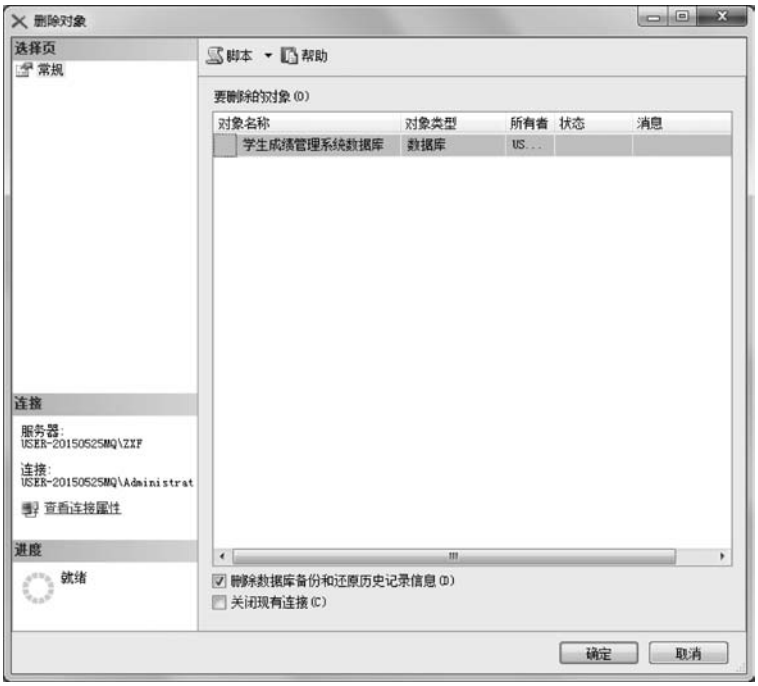

图 3.14 “删除对象”窗口

```
DROP DATABASE database_name [,...n]
```

其中,database_name 是要删除的数据库的名称。

例 3-14 删除“学生成绩管理系统数据库”。

```
DROP DATABASE 学生成绩管理系统数据库
```

例 3-15 删除数据库 db2 和 db3。

```
DROP DATABASE db2, db3
```

本例删除了两个数据库,两个数据库之间以逗号分隔。

3.3.6 数据库的备份与还原

数据库中的数据可能遭到丢失和破坏,这就有必要定时或在有需要的时候制作数据库的副本,即进行数据备份,以便在发生意外时能修复数据库,即进行数据库的恢复。

1. 数据库备份的类型

SQL Server 2012 支持四种基本类型的备份:完整备份、差异备份、事务日志备份、文件和文件组备份。

(1) 完整备份:备份整个数据库的所有内容,包括事务日志。该备份类型需要比较大的存储空间来存储备份文件,备份时间也比较长,在还原数据时,也只要还原一个备份文件。

(2) 差异备份:差异备份只备份上次完整备份后更改的数据。差异备份的速度一般比完整备份要快。在还原数据时,要先还原前一次做的完整备份,然后还原最后一次所做的差异备份,这样才能让数据库里的数据恢复到与最后一次差异备份时的内容相同。

(3) 事务日志备份：事务日志备份只备份事务日志里的内容。事务日志记录了上一次完整备份或事务日志备份后数据库的所有变动过程。在进行事务日志备份之前，必须要进行完整备份。与差异备份类似，事务日志备份生成的文件较小、占用时间较短，但是在还原数据时，除了要先还原完整备份之外，还要依次还原每个事务日志备份，而不是只还原最后一个事务日志备份(这是与差异备份的区别)。

(4) 文件和文件组备份：如果在创建数据库时，为数据库创建了多个数据库文件或文件组，可以使用该备份方式。使用文件和文件组备份方式可以只备份数据库中的某些文件。

2. 数据库的备份

备份数据库可以使用 SQL Server Management Studio 或 BACKUP 语句来实现，这里仅介绍使用 SQL Server Management Studio 备份数据库的方法，步骤如下。

(1) 打开 SQL Server Management Studio，在“对象资源管理器”中右击要备份的数据库，在弹出的快捷菜单中选择“任务”→“备份”命令，打开如图 3.15 所示的“备份数据库”窗口。

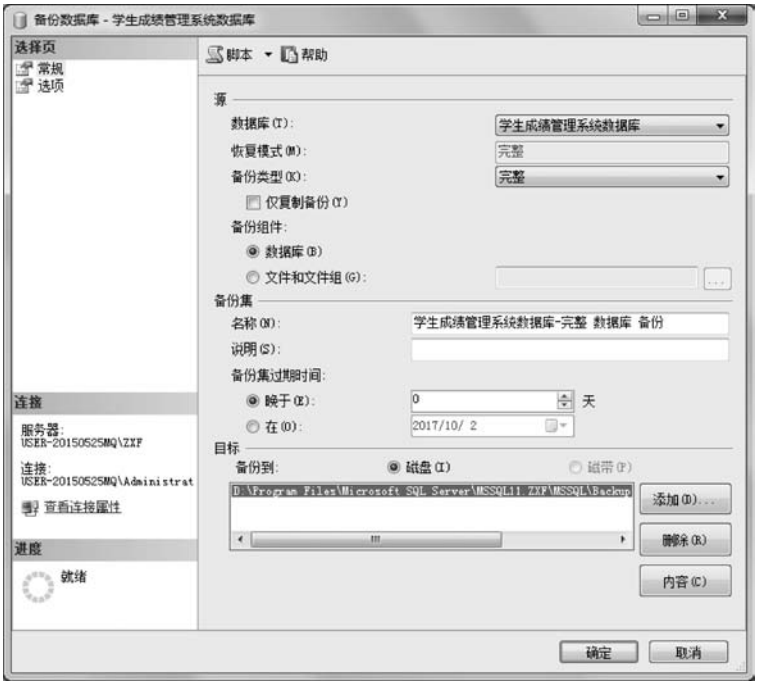

图 3.15 “备份数据库”窗口

(2) 在“备份数据库”对话框的“常规”选项页中，“数据库”下拉列表框中可以更改待备份的数据库；选择“备份类型”命令。如果是第一次备份，应该选择完整备份。在“备份集”名称文本框中可设置此备份的名称。“备份集过期时间”晚于 0 天表示永远过期；“目标”中可添加或删除备份设备(备份文件)。

(3) 设置完成后，单击“确定”按钮开始备份。

3. 数据库的还原

还原数据库可以使用 SQL Server Management Studio 或 RESTORE 语句来实现，这里仅介绍使用 SQL Server Management Studio 还原数据库，步骤如下。

(1) 打开 SQL Server Management Studio，如果要还原的数据库不存在，则在“对象资源管理器”中右击“数据库”结点，在弹出的快捷菜单中选择“还原数据库”命令，打开如图 3.16 所示的“还原数据库”窗口。如果要还原的数据库存在，则在“对象资源管理器”中右击该数据库结点，在弹出的快捷菜单中选择“任务”→“还原”→“数据库”命令，打开“还原数据库”窗口。

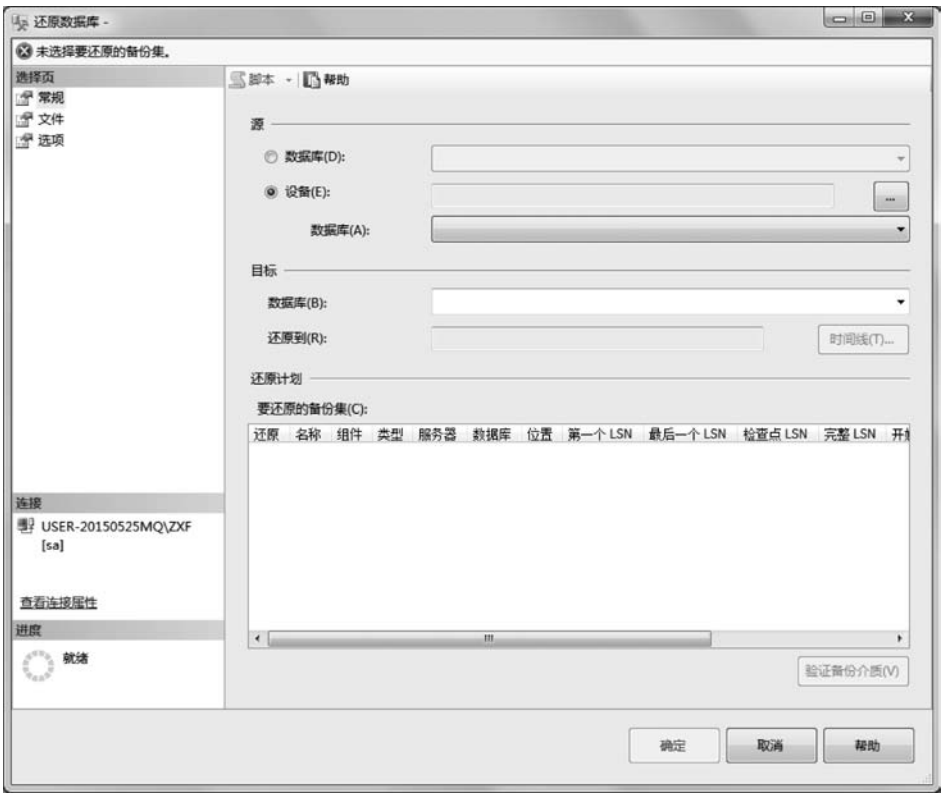

图 3.16 “还原数据库”窗口

(2) 在“还原数据库”窗口的“常规”选项页的“源”选项中选择“设备”，并单击“设备”右侧的 按钮，打开如图 3.17 所示的“选择备份设备”窗口，在对话框中添加在备份数据库时产生的备份文件，单击“确定”按钮。

(3) 返回“还原数据库”窗口，单击“确定”按钮即可还原数据库。

例 3-16 小明同学在机房做实验时创建了一个“学生选课”数据库，实验时间结束时，小明希望将此数据库备份到自己的 U 盘上，并在下次实验时还原数据库。

第 1 步：备份数据库到 U 盘。

在 SQL Server Management Studio 的“对象资源管理器”中右击“学生选课”数据库，在弹出的快捷菜单中选择“任务”→“备份”命令，打开如图 3.18 所示的“备份数据库”窗口，备份类型选择“完整”，备份组件选择“数据库”，备份集过期时间选择“晚于 0 天”。

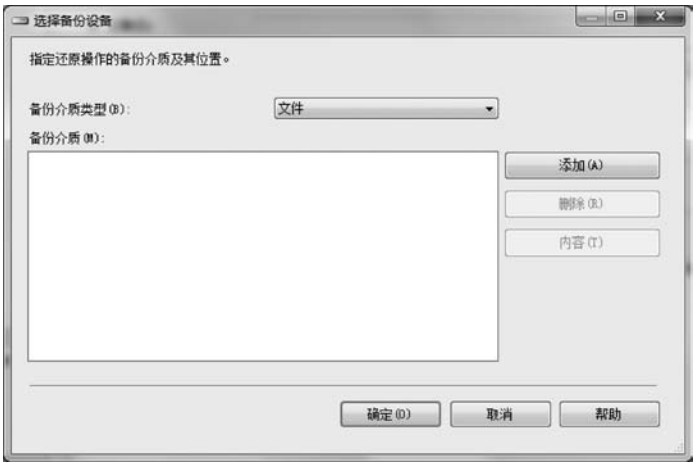

图 3.17 “选择备份设备”窗口

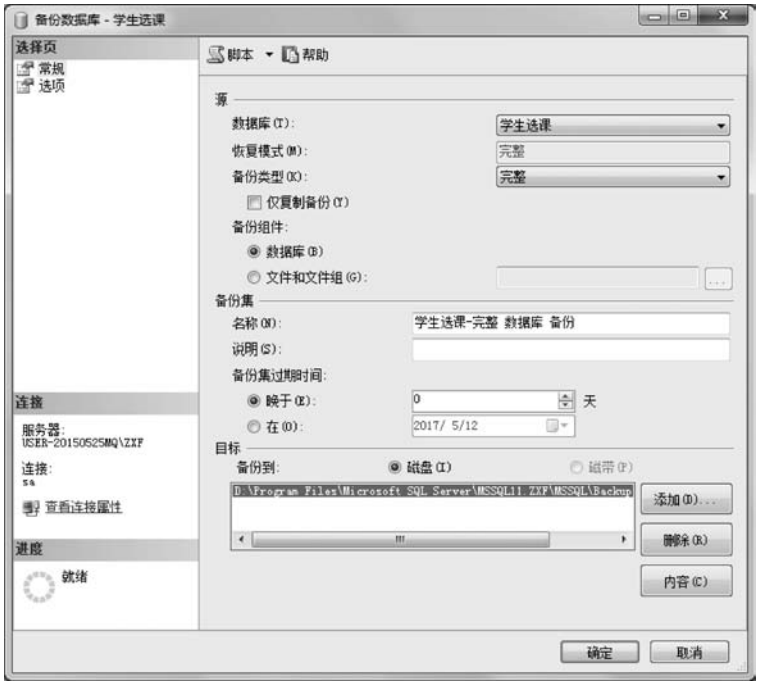

图 3.18 备份“学生选课”数据库

单击“删除”按钮，删除目标中的备份设备。单击“添加”按钮，弹出“选择备份目标”窗口，如图 3.19 所示。

单击省略号按钮，打开“定位数据库文件”窗口，为备份文件选择存放路径并命名，如图 3.20 所示。注意：文件的扩展名为.bak。

数据库备份执行成功后，就会在指定的路径下产生一个备份文件。

第 2 步：通过 U 盘上的备份文件还原数据库。

若“学生选课”数据库不存在，此时就需要进行数据库的还原，方法如下。

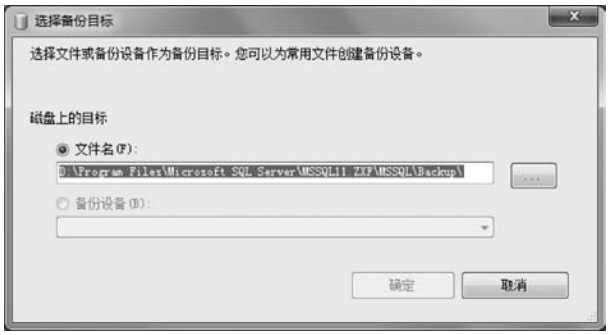

图 3.19 “选择备份目标”窗口

图 3.20 “定位数据库文件”窗口

在 SQL Server Management Studio 的“对象资源管理器”中右击“数据库”结点，在弹出的快捷菜单中选择“还原数据库”命令，打开如图 3.16 所示的“还原数据库-学生选课”窗口。

在“还原数据库”窗口的“常规”选项页的“源”选项中选择“设备”，并单击“设备”右侧的 按钮，打开如图 3.17 所示的“选择备份设备”窗口，在窗口中添加备份数据库时产生的备份文件，单击“确定”按钮，返回到“还原数据库”窗口。

添加了备份文件的“还原数据库-学生选课”窗口如图 3.21 所示，单击“确定”按钮即可还原数据库。还原数据库后即可在“对象资源管理器”中看到“学生选课”数据库。

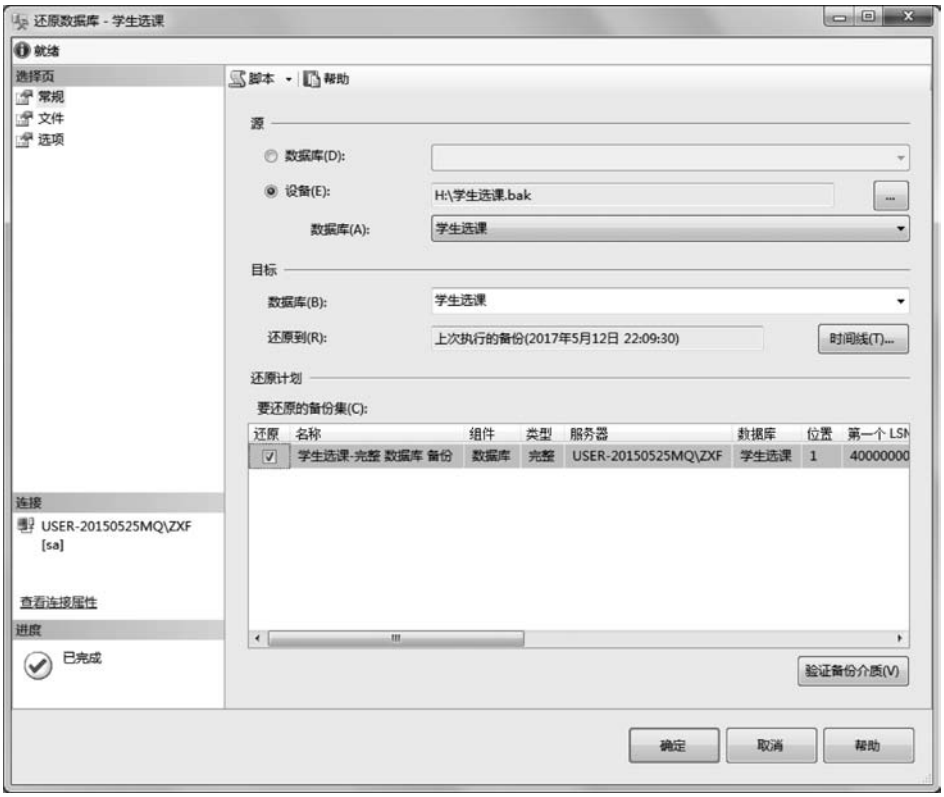

图 3.21　还原“学生选课”数据库

3.4　表的创建与管理

表是 SQL Server 中最重要的数据库对象，数据库中的所有数据都存放在表中。表由行和列组成，表的每一行是对一个实体的描述，也称为“记录”或“元组”；表的每一列表示实体的一个属性，也可以用表来存储表之间的联系。

3.4.1　数据类型

数据类型是数据的一种属性，表示数据所表示信息的类型。任何一种计算机语言都定义了自己的数据类型，当然，不同的程序设计语言都具有不同的特点，所定义的数据类型的种类和名称或多或少都有些不同。

在 SQL Server 中，每个列、局部变量、表达式和参数都具有一个相关的数据类型。SQL Server 提供了系统数据类型集，该类型集定义了可与 SQL Server 一起使用的所有数据类型。用户还可以使用 Transact-SQL 或 Microsoft .NET Framework 定义自己的数据类型。

SQL Server 2012 支持多种数据类型，主要包括数值数据、字符数据、二进制数据、日期和时间数据、逻辑数据和其他数据六大类数据类型。

1. 数值数据类型

SQL Server 的数值数据类型可分为以下四种类型。

(1) 整数数据类型。

整数数据类型用于存储无小数部分的数值数据，如年龄、数量等。整数数据类型可以用较少的字节存储较大的精确数值，只要有可能，对数值列应尽量使用整数数据类型。SQL Server 2012 有四个整数数据类型，分别是 tinyint、smallint、int 和 bigint，用于存储不同范围的值，如表 3.6 所示。

表 3.6　整数数据类型

数据类型	存储字节数	取值范围	使用说明
tinyint	1 字节	$0\sim2^8-1$，即 0～255	存储小范围的非负整数
smallint	2 字节	$-2^{15}\sim2^{15}-1$，即－32 768～32 767	存储正负整数
int	4 字节	$-2^{31}\sim2^{31}-1$， 即－2 147 483 648～2 147 483 647	存储正负整数
bigint	8 字节	$-2^{63}\sim2^{63}-1$， 即－9 223 372 036 854 775 808～ 9 223 372 036 854 775 807	存储非常大范围的正负整数

(2) 精确数值数据类型。

精确数值数据类型用于存储有多个小数位的数值。精确数值的“精度”是指小数点前后的所有位数个数，“小数位数”是指小数点后的位数个数。SQL Server 2012 有 decimal 和 numeric 两种精确数值数据类型，如表 3.7 所示，其中 P 表示精度、S 表示小数位数。

表 3.7　精确数值数据类型

数据类型	存储字节数	取值范围	使用说明
decimal[(P[,S])]	依据不同的精度，需要 5～17B	$-10^{38}+1\sim10^{38}-1$	P 的默认值为 18，最大可以存储 38 位十进制数；S 的默认值为 0，只能取 0～P 的值
numeric[(P[,S])]	依据不同的精度，需要 5～17B	$-10^{38}+1\sim10^{38}-1$	功能上等价于 decimal，并可以与 decimal 交换使用

(3) 近似数值数据类型。

近似数值数据类型用于存储浮点数据，近似数值数据不能精确地表示所有值。SQL Server 2012 有 float 和 real 两种近似数值数据类型，如表 3.8 所示。

表 3.8　近似数值数据类型

数据类型	存储字节数	取值范围	使用说明
float[(n)]	n 为 1～24 时，4B n 为 25～53 时，8B	$-1.79\times10^{308}\sim-2.23\times10^{-308}$ 0 $2.23\times10^{-308}\sim1.79\times10^{308}$	存储大型浮点数，默认精确到第 15 位
real	4B	$-3.40\times10^{38}\sim-1.18\times10^{-38}$ 0 $1.18\times10^{-38}\sim3.40\times10^{38}$	仍然有效，但为了满足 SQL-92 标准，已经被 float 替换了，精确到第 7 位数

float(*n*)中的 *n* 用于存储该数值尾数的位数。SQL Server 对此只使用两个值：如果指定 *n* 的值位于 1～24，SQL Server 就使用 24；如果指定 *n* 的值位于 25～53，SQL Server 就使用 53；当未指定 *n* 的值时，默认为 53。real 相当于 float(24)。

(4) 货币数据类型。

货币数据类型用于存储货币或现金值，SQL Server 提供了两种货币数据类型，分别是 money 和 smallmoney，这两种数据类型精确到它们所代表的货币单位的万分之一，如表 3.9 所示。

表 3.9　货币数据类型

数据类型	存储字节数	取值范围	使用说明
smallmoney	4B	−214748.3648～214748.3647	存储小型货币值，精确到小数点后 4 位
money	8B	−922337203685477.5808～922337203685477.5807	存储大型货币值，精确到小数点后 4 位

2. 字符数据类型

字符数据类型用于存储由字符构成的文本(字符串)，根据所采用的编码方案，又可以分为字符数据类型(采用 ANSI 编码)和 Unicode 字符数据类型(采用 Unicode 编码)。

(1) 字符数据类型。

字符数据类型采用 ANSI 编码，可用于存储汉字、英文字母、数字符号和其他各种符号。在 SQL 语句中书写字符型数据时，要用单引号(')将字符串括起来。例如，'张三'、'华中科技大学'等。字符数据类型有四种，如表 3.10 所示。

表 3.10　字符数据类型

数据类型	存储字节数	长度取值范围	使用说明
char[(*n*)]	*n* B	1～8000	固定宽度的 ANSI 数据类型，默认长度为 1
varchar[(*n*)]	实际字符长度＋2B	1～8000	可变宽度的 ANSI 数据类型，默认长度为 1
varchar(MAX)	实际字符长度＋2B	$1\sim2^{31}-1$	可变宽度的 ANSI 数据类型
text	实际字符长度＋2B	$1\sim2^{31}-1$	可变宽度的 ANSI 数据类型，已由 varchar(MAX)取代

char 为固定宽度的字符数据类型，在用 char(*n*)对列进行说明时，指示列长度为 *n*。如果不指定长度 *n*，系统默认长度为 1。多于列长度的输入从后面被截取，输入字符的长度短于指定字符长度时用空格填满。

varchar 为可变宽度的字符数据类型。varchar 数据类型的结构与 char 数据类型一致，区别是当输入 varchar 字符的长度小于 *n* 时不用空格来填满，而是按输入字符的实际长度存储。

varchar(MAX)和 text 用于存储数据量庞大且长度变化的字符文本数据。用户要求表中的某列能存储 255 个字符以上的数据，可使用 varchar(MAX)和 text 数据类型。

(2) Unicode 字符数据类型。

Unicode 是国际组织制定的可以容纳世界上所有文字和符号的字符编码方案。对于使用 Unicode 字符数据类型的数据，每个字符需占用 2 字节的存储空间。在 SQL 语句中书写 Unicode 字符数据时，一般需先写字母 N，再用单引号(')将字符串括起来，例如，N'张三'、N'华中科技大学'等。Unicode 字符数据类型有四种，如表 3.11 所示。

表 3.11　Unicode 字符数据类型

数 据 类 型	存储字节数	长度取值范围	使 用 说 明
nchar[(*n*)]	2*n*B	1～4000	固定宽度的 Unicode 数据类型，默认长度为 1
nvarchar[(*n*)]	2 * 实际字符长度＋2B	1～4000	可变宽度的 Unicode 数据类型，默认长度为 1
nvarchar(MAX)	2 * 实际字符长度＋2B	1～$2^{30}-1$	可变宽度的 Unicode 数据类型
ntext	2 * 实际字符长度＋2B	1～$2^{30}-1$	可变宽度的 Unicode 数据类型，已被 nvarchar(MAX)取代

3. 二进制数据类型

二进制数据类型用于存储二进制数据，如图形文件、Word 文档或 MP3 文件等。Image 数据类型可在数据页外部存储最多 2GB 的文件。Image 数据类型的首选替代数据类型是 varbinary(MAX)，可保存超过 8KB 的二进制数据，其性能通常比 image 数据类型好。SQL Server 2012 的新功能是可以在操作系统文件中通过 FileStream 存储选项存储 varbinary(MAX)对象。这个选项将数据存储为文件，同时不受 varbinary(MAX)的大小限制。

二进制数据类型有四种，如表 3.12 所示。在 SQL 语句中书写二进制数据时，需用 0x 开头的两个十六进制数构成 1 字节，如 0x5A、0x69B7。

表 3.12　二进制数据类型

数 据 类 型	存储字节数	长度取值范围	使 用 说 明
binary[(*n*)]	*n*B	1～8000	固定宽度的二进制数据类型，默认长度为 1
varbinary[(*n*)]	实际字符长度＋2B	1～8000	可变宽度的二进制数据类型，默认长度为 1
varbinary(MAX)	实际字符长度＋2B	1～$2^{31}-1$	可变宽度的二进制数据类型
image	实际字符长度＋2B	1～$2^{31}-1$	可变宽度的二进制数据类型，已由 varbinary(MAX)取代

4. 日期和时间数据类型

日期和时间数据类型用于存储日期和时间数据。SQL Server 2012 支持多种日期时间数据类型，包括 date、datetime、datetime2、datetimeoffset、smalldatetime 和 time，如表 3.13 所示。

表 3.13　日期和时间数据类型

数据类型	存储字节数	取值范围	精确度	格　式
date	3B	0001-01-01～9999-12-31	1day	YYYY-MM-DD
datetime	8B	1753-01-01～9999-12-31	0.00333s	YYYY-MM-DD hh:mm:ss[.nnn]
datetime2[(*n*)]	6～8B	0001-01-01 00:00:00.0000000～9999-12-31 23:59:59.9999999 *n* 指定秒的小数位数，取值为0～7，默认值为7	100ns	YYYY-MM-DD hh:mm:ss[.nnnnnnn]
datetimeoffset[(*n*)]	8～10B	9999年1月1日～12月31日 *n* 指定小数秒+/-偏移量，取值为0～7，默认值为7	100ns	YYYY-MM-DD hh:mm:ss[.nnnnnnn][+|-]hh:mm
smalldateTime	4B	1900-01-01～2079-06-06	1min	YYYY-MM-DD hh:mm:ss
time[(*n*)]	3～5B	00:00:00.0000000～23:59:59.9999999 *n* 指定秒的小数位数，取值为0～7，默认值为7	100ns	hh:mm:ss[.nnnnnnn]

date是单独表示日期的数据类型。time是单独表示时间的数据类型。datetime2是一种比datetime具有更大的日期范围和更好的精度的日期类型。datetimeoffset具有一个时区组成部分。

time、datetime2以及datetimeoffset的存储空间大小依赖于所选择的精度，可以通过0～7的整数来指定其精度，分别代表不同小数位数的秒值的精度。例如，time(0)表示秒的精度只有0位小数，即只能准确到1s；time(3)表示准确到1ms；而time(7)则表示准确到100ns。如果没有指定秒的小数部分的精度，则SQL Server默认将上述三种类型的精度设置为7。

在SQL语句中书写日期和时间型数据时，要用单引号(')将日期和时间括起来，例如，'2007-12-01'、'12:15:20'、'2008-01-3 12:15:11'等。

5. 逻辑数据类型

SQL Server的逻辑数据类型也称为位(bit)数据类型，适用于判断真/假的场合，长度为一字节，取值为1、0或NULL。

6. 其他数据类型

SQL Server还提供了一些特殊的数据类型，如表3.14所示。

表 3.14　其他数据类型

数据类型	存储字节数	使用说明
cursor	不适用	包含一个对游标的引用，可以用作变量或存储过程参数，创建表时不能使用
hierarchyid	1～892B+2B的额外开销	包含一个对层次结构中位置的引用
xml	最多2GB	可以以Unicode或非Unicode形式存储

续表

数据类型	存储字节数	使用说明
sql_variant	8016B	可能包含任何系统数据类型的值，除了 text、ntext、image、timestamp、xml、varchar(max)、nvarchar(max)、varbinary(max)、sql_variant 以及用户定义的数据类型。最大长度为 8000B 数据+16B(或元数据)
table	取决于表定义和存储的行数	存储用于进一步处理的数据集，其定义类似于 Create Table，主要用于返回表值函数的结果集，也可用于存储过程和批处理中
timestamp	8B	对于每个表来说是唯一的、自动存储的值，通常用于版本戳，该值在插入和每次更新时自动改变
uniqueidentifier	16B	用来存储一个全局唯一标识符(Globally Unique Identifier，GUID)，GUID 值可以从 newid()函数获得，这个函数返回的值对数据库来说是唯一的

3.4.2 表的创建

在设计数据库时，要根据数据库逻辑结构设计的要求，确定需要哪些表，各表中都有哪些列，表的各列的数据类型，表的主键、外键、约束、索引等。创建表就是定义一个新表的结构以及它与其他表之间的联系。

在 SQL Server 中，可以使用 SQL Server Management Studio 和 Transact-SQL 语句来创建表。

1. 使用 SQL Server 管理平台创建表

使用 SQL Server 管理平台创建表的步骤如下。

(1) 在 SQL Server Management Studio 中打开“对象资源管理器”面板，展开需要新建表的数据库，右击“表”结点，在弹出的快捷菜单中选择“新建表”命令，打开“表设计器”对话框。

(2) 在“表设计器”中输入各列的列名、数据类型、是否允许空值等信息，如图 3.22 所示。

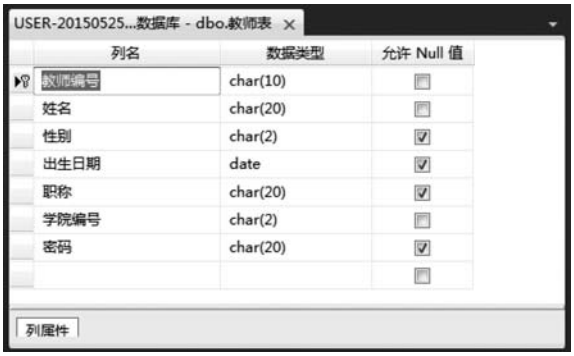

图 3.22 “表设计器”对话框

(3) 填写完成后，单击工具栏上的“保存”按钮，打开如图 3.23 所示的“选择名称”对话框，输入新建表的名称后，单击“确定”按钮，即可创建一个新表。

图 3.23 “选择名称”对话框

2. 使用 Transact-SQL 语句创建表

在 SQL Server 中,可以使用 CREATE TABLE 语句来创建表。CREATE TABLE 语句的语法格式如下:

```
CREATE TABLE table_name
({< column_definition > | < computed_column_definition >} [,...n]
    [,< table_constraint > [,...n] ]
)
```

各项的含义如下。

(1) table_name 是新表名称,该名称需符合标识符的命名规则。

(2) < column_definition >: 对表中列的定义说明,其语法格式如下。

```
< column_definition > :: = column_name data_type [< column_constraint >]
```

其中,column_name 为列名,data_type 为列的数据类型,< column_constraint >为此列的列级约束的定义。< column_constraint >的语法格式见 3.4.3 节。

(3) < computed_column_definition >: 对计算列的定义。计算列是指其值是由同一表中的其他列计算得到的,而非用户输入的。计算列不能作为 INSERT 或 UPDATE 语句的目标。

< computed_column_definition >的语法格式如下:

```
< computed_column_definition > :: =
column_name AS computed_column_expression
```

其中,computed_column_expression 是定义计算列的表达式,表达式可以是非计算列的列名、常量、函数、变量,也可以是由一个或多个运算符连接的上述元素的任意组合。

(4) < table_constraint >定义新表的表级约束,具体语法格式见 3.4.3 节。

例 3-17 在“学生成绩管理系统数据库”中创建“学生”表,包括“学号”“姓名”“性别”“籍贯”“出生日期”“专业班级”“入学时间”“学制”“学院编号”“密码”字段。

```
USE 学生成绩管理系统数据库
CREATE TABLE 学生
(   学号 char(10),
    姓名 char(20),
    性别 char(2),
    籍贯 char(20),
    出生日期 date,
    专业班级 char(30),
    入学时间 date,
```

```
    学制 int,
    学院编号 char(2),
    密码 char(20)
)
```

本例使用 USE 语句打开“学生成绩管理系统数据库”,使之成为当前数据库,然后在当前数据库中创建“学生”表。

例 3-18 创建员工工资表 salary,包括“姓名”“基本工资”“奖金”“总计”字段,其中“总计”字段是计算列,其值为基本工资和奖金之和。

```
CREATE TABLE salary
(   姓名 varchar(10),
    基本工资 money,
    奖金 money,
    总计 AS 基本工资 + 奖金
)
```

本例创建了 salary 表,定义了三个数值列。其中,“总计”列为计算列,其值由表达式计算而来,其数据类型为表达式的数据类型。

3. 使用 SQL Server 管理平台设计数据库关系

数据库关系图是 SQL Server 管理平台提供的一种很实用的工具。它将表和表间关系以及其他对象以图形方式表现出来,并且用户也可以通过它以图形的方式来增加、修改表和表间关系等数据库对象。

下面通过一个例子来说明数据库关系图的基本操作。

设有一个名为“教务管理系统”的数据库,目前数据库中有如下三个表,表结构如下:

```
院系表(院系编号,院系名称,办公地址,联系电话)
班级表(班级编号,班级名称,院系编号)
学生表(学号,姓名,性别,籍贯,出生日期,班级编号)
```

假设已设置“院系编号”为“院系表”的主键、“班级编号”为“班级表”的主键、“学号”为“学生表”的主键;“班级表”的“院系编号”字段为外键,对应的主键为“院系表”的“院系编号”字段。关系表的主键和外键设置方法见 3.4.3 节。

(1) 创建数据库关系图。

在 SQL Server Management Studio 中打开“对象资源管理器”面板,打开“教务管理系统”数据库,右击“数据库关系图”结点,在弹出的快捷菜单中选择“新建数据库关系图”命令,系统会打开一个“关系图设计器”窗口,并弹出“添加表”对话框。

“添加表”对话框里显示了“教务管理系统”数据库里面的所有表,将三张表都添加进“关系图设计器”后,“关系图设计器”中的关系图如图 3.24 所示。

(2) 在关系图中查看表结构和表间关系。

在关系图中,“院系表”和“班级表”之间存在一条连线,表明“院系表”和“班级表”之间存在联系(外键),连线一端有钥匙标志的表为主键表,连线另一端的表为外键表。

(3) 在关系图中创建表间关系。

“学生表”的“班级编号”字段应为“班级表”的“班级编号”字段的外键,下面在关系图中创建此外键(关系)。

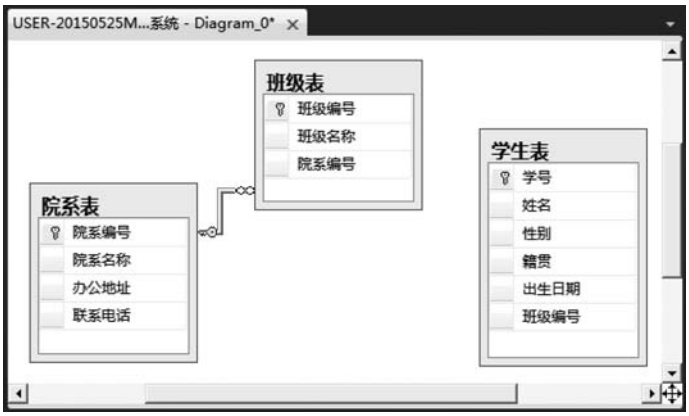

图 3.24　关系图

选中“班级表”的“班级编号”列，按住鼠标左键拖曳至“学生表”的“班级编号”列后松开，在弹出的“表和列”对话框中设置关系名称、主键表、主键字段、外键表、外键字段等信息后，单击“确定”按钮关闭“表和列”对话框。

在“外键关系”对话框中进行与外键相关的一些设置，单击“确定”按钮关闭对话框。

创建“班级表”与“学生表”的表间关系后的关系图如图 3.25 所示。

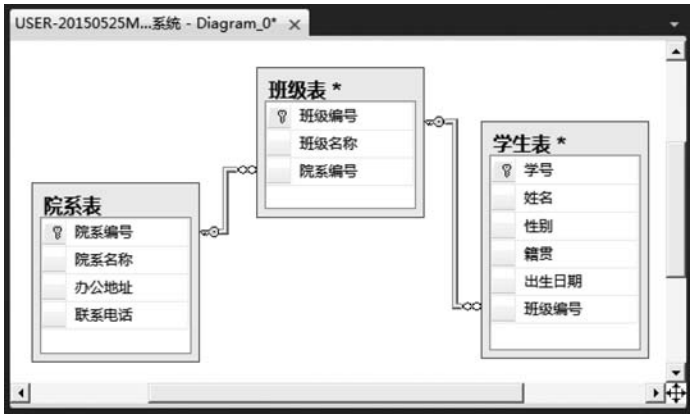

图 3.25　创建新关系后的关系图

(4) 在关系图中修改表结构和表间关系。

右击关系图的空白处，在弹出的快捷菜单中进行相应的选择，可完成新建表、向关系图中添加表等操作。

右击表或字段，在弹出的快捷菜单中进行相应的选择，可完成设置表结构(插入列、删除列、设置主键、设置 CHECK 约束等)、删除表等操作。

右击表间的连线，在弹出的快捷菜单中进行相应的选择，可以完成设置关系属性、从数据库中删除关系的操作。

(5) 保存关系图。

单击工具栏上的“保存”按钮，打开“选择名称”对话框，输入关系图的名称保存关系图。保存关系图后，在关系图中对表结构和表间关系的创建和修改将在数据库中实现。

3.4.3 定义表的约束

数据的完整性是指保护数据库中数据的正确性、有效性和相容性，防止错误的数据进入数据库。SQL Server 提供的完整性机制主要包括约束、触发器、存储过程等。本节介绍约束的概念和使用方法。

约束定义了关于列中允许值的规则，SQL Server 通过限制列中数据、行中数据和表之间的数据来保证数据的完整性。约束独立于表结构，作为数据库定义部分在 CREATE TABLE 语句中声明，可以在不改变表结构的基础上，通过 ALTER TABLE 语句添加或删除。当表被删除时，表所带的所有约束定义也随之删除。

在 CREATE TABLE 语句中定义约束时，可使用列级约束和表级约束两种方法进行定义。

(1) 列级约束是对某一个特定列的约束，包含在列定义中，直接跟在该列的其他定义之后，用空格分隔，不必指定列名。

(2) 表级约束不包含在列定义中，而是在所有列定义之后定义，与列定义用“,”分隔。定义表级约束时必须指出要约束的列的名称。

在 CREATE TABLE 语句中，如果约束是针对单个列的，则既可以使用列级约束，也可以使用表级约束；如果约束是针对多个列的，则必须使用表级约束。在 ALTER TABLE 语句中增加约束时，只能使用表级约束。

在 CREATE TABLE 语句或 ALTER TABLE 语句中定义约束时，可为约束命名，也可不为约束命名，不命名时系统将自动为约束命名。

在 SQL Server 中有六种约束，分别是空值/非空值约束、主键约束、外键约束、唯一性约束、检查约束和默认约束。

1. 空值/非空值约束

当用户往表中插入一行而未对其中的某列指定值时，该列将出现空值(NULL)。空值不同于空白(空字符串)或数值零，它通常表示未填写、未知(Unknown)、不可用或将在以后添加数据。空值会对查询命令或统计函数产生影响，实际应用中应尽量少使用空值。

因为每个空值均为未知，所以没有两个空值是相等的，不可以比较两个空值的大小或比较空值与任何其他数据的大小。判断某列中的值是否为空值，可以使用关键字 IS NULL 或 IS NOT NULL。

当某一字段一定要输入值才有意义的时候，可以为这一字段定义非空值(NOT NULL)约束，即不允许此列出现空值；当允许字段不输入值时，可以为这一字段定义空值(NULL)约束，即允许此列出现空值。当没有为字段定义空值或非空值约束时，系统允许字段值为空值，即具有空值约束。

空值/非空值约束只能定义列级约束，语法格式如下：

```
[CONSTRAINT constraint_name] [NOT] NULL
```

其中，constraint_name 为约束的名称，在定义空值/非空值约束时通常省略不写。如果不带 NOT，则为空值约束，否则为非空值约束。

例 3-19 在“学生成绩管理系统数据库”中创建“教师”表，包括“教师编号”“姓名”“性

别”“出生日期”“职称”“学院编号”“密码”字段。要求为“教师编号”“姓名”“学院编号”列设置非空值约束，为“性别”列设置空值约束。

```
CREATE TABLE 教师
(  教师编号 char(10) NOT NULL,
   姓名 char(20) NOT NULL,
   性别 char(2) NULL,
   出生日期 date,
   职称 char(20),
   学院编号 char(2) NOT NULL,
   密码 char(20)
)
```

本例中，并未为“出生日期”“职称”“密码”列设置空值/非空值约束，系统视这几列具有空值约束。

2. 主键约束

主键是被挑选出来，作为元组的唯一标识的候选关键字。它可以唯一确定表中的一行数据，或者说可以唯一确定一个实体。一个表只有一个主键，主键不允许为空值，且不同的两行的键值不能相同。

主键可以由一列，也可以由多列组成。如果一个表的主键由单列组成，则该主键约束既可以定义为该列的列级约束，也可以定义为表级约束。如果主键由两个或两个以上的列组成，则该主键约束必须定义为表级约束。

定义列级主键约束的语法格式如下：

```
[CONSTRAINT constraint_name]
PRIMARY KEY [CLUSTERED | NONCLUSTERED]
```

定义表级主键约束的语法格式如下：

```
[CONSTRAINT constraint_name]
PRIMARY KEY [CLUSTERED | NONCLUSTERED]
{(column_name [,...n])}
```

各项的含义如下。

(1) constraint_name 是约束名称，该名称需符合标识符的命名规则。

(2) [CLUSTERED | NONCLUSTERED]指系统创建主键索引时，索引为聚集索引(CLUSTERED)还是非聚集索引(NONCLUSTERED)，默认值为聚集索引。索引的概念见第 5 章。

(3) column_name 为构成主键的字段名。

例 3-20 在“学生成绩管理系统数据库”中创建“学院”表，包括“学院编号”“学院名称”“学院电话”“学院地址”字段。将“学院编号”字段设置为主键。

```
CREATE TABLE 学院
(  学院编号 char(2) PRIMARY KEY,
   学院名称 char(30),
   学院电话 char(12),
   学院地址 char(50)
)
```

本例中，主键由单列构成，可以定义为列级约束。定义约束时没有为约束取名，系统会自动为约束取名。在“对象资源管理器”中展开“学院”结点下的“键”结点，即可看到约束的名称。

例 3-21 在“学生成绩管理系统数据库”中创建“选课成绩”表，包括“学号”“课堂编号”“成绩”字段。将“学号”和“课堂编号”字段设置为主键，并为主键约束命名。

```
CREATE TABLE 选课成绩
(  学号 char(10),
   课堂编号 char(16),
   成绩 int,
   CONSTRAINT PK_选课成绩 PRIMARY KEY (学号, 课堂编号)
)
```

本例中，主键由多列构成，因此必须定义为表级约束。

除了可以通过 Transact-SQL 语句定义主键，还可以通过在 SQL Server Management Studio 中设计表结构时定义主键。例如，例 3-21 是通过 CREATE TABLE 命令创建“选课成绩表”并定义主键的，下面在 SQL Server Management Studio 中完成此功能。

(1) 打开“对象资源管理器”面板，打开“学生成绩管理系统数据库”，右击“表”结点，在弹出的快捷菜单中选择“新建表”命令。

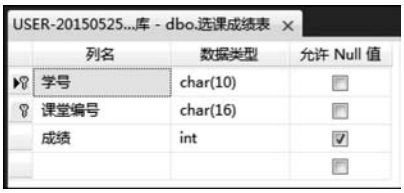

图 3.26 设置主键

(2) 在“表设计器”中输入各列的列名、数据类型、是否允许空值等信息。

(3) 定义主键，选中“学号”列（单击列名左侧的“行”按钮），再按住 Ctrl 键选中“课堂编号”列，在工具栏上单击“设置主键”按钮 即可设置主键。此时，“学号”和“课堂编号”的左侧会显示 图标，如图 3.26 所示。

(4) 单击工具栏上的“保存”按钮，打开“选择名称”对话框，输入新建表的名称“选课成绩”后，击“确定”按钮，则创建了“选课成绩”表。

3. 外键约束

外键约束定义了表与表之间的联系。通过将一个表中的一列或多列添加到另一个表中，创建两个表之间的联系，这个列就成为第二个表的外键（Foreign Key，FK），即外键是用于建立和加强两个表数据之间的联系的一列或多列，通过它可以强制参照完整性。

当一个表中的一列或多列的组合和其他表中的主键定义相同时，就可以将这列或这些列的组合定义为外键，并设定与它关联的表和列。这样，当向外键表插入数据时，如果与之相关联的表（称为主键表或主表）的关键字列中没有与插入的外键列值相同的值时，系统会拒绝插入数据。同时，如果主键表的某个元组的关键字值在外键表的外键列出现，则此元组不能从主键表中删除。

定义列级外键约束的语法格式如下：

```
[CONSTRAINT constraint_name] [FOREIGN KEY]
REFERENCES ref_table[(ref_column [,...n])]
[ON DELETE {CASCADE | NO ACTION}]
[ON UPDATE {CASCADE | NO ACTION}]
```

定义表级外键约束的语法格式如下：

```
[CONSTRAINT constraint_name] FOREIGN KEY (column_name [,...n])
REFERENCES ref_table [(ref_column [,...n])]
[ON DELETE {CASCADE | NO ACTION}]
[ON UPDATE {CASCADE | NO ACTION}]
```

各项的含义如下。

(1) constraint_name 是约束名称，该名称需符合标识符的命名规则。

(2) column_name 为外键列名。

(3) ref_table 为主键表名。ref_column 为主键表的主键列名，ref_column 可以省略不写。

(4) ON DELETE CASCADE 是指为外键设置级联删除，ON DELETE NO ACTION 是指不为外键设置级联删除，默认值为 NO ACTION。级联删除是指当主键表中的某行被删除时，外键表中所有相关行将自动被系统删除。

(5) ON UPDATE CASCADE 是指为外键设置级联修改，ON UPDATE NO ACTION 是指不为外键设置级联修改，默认值为 NO ACTION。级联修改是指当主键表中某行的键值被修改时，外键表中所有相关行的该外键值也将被自动修改为新值。

例 3-22 在创建“课程”表时，指明“学院编号”列为外键，对应的主键为“学院”表的“学院编号”列。

```
CREATE TABLE 课程
(  课程编号 char(8) PRIMARY KEY,
   课程名称 varchar(50),
   学时数 int,
   学分数 float,
   课程性质 char(10),
   课程介绍 text,
   学院编号 char(2) FOREIGN KEY REFERENCES 学院(学院编号)
)
```

注意：为了能将“课程”表中的“学院编号”列定义为外键，必须先定义“学院”表且将该表的“学院编号”列定义为主键。

例 3-23 在创建“课堂”表时，指明“教师编号”列为外键，对应的主键为“教师”表的“教师编号”列，并设置为级联修改；指明“课程编号”列为外键，对应的主键为“课程”表的“课程编号”列，并设置为级联删除。

```
CREATE TABLE 课堂
(  课堂编号 char(16) PRIMARY KEY,
   课堂名称 varchar(50),
   开课年份 char(10),
   开课学期 char(2),
   教师编号 char(10),
   课程编号 char(8),
   班级列表 char(80),
   课堂状态 int,
   最少开课人数 int,
```

```
    最多开课人数 int,
    成绩激活 int,
    FOREIGN KEY (教师编号) REFERENCES 教师(教师编号) ON UPDATE CASCADE,
    FOREIGN KEY (课程编号) REFERENCES 课程(课程编号) ON DELETE CASCADE
)
```

注意：教师和课程两个表必须事先建好，且已设置好各自的主键。

本例中，建立了两个表级外键约束。

尽管外键约束的主要目的是控制存储在外键表中的数据，但它还可以通过级联操作，使得当主键表中的数据被修改或删除后，外键表中的数据也相应地做相同的更新操作。

本例中，对“教师编号”列设置了级联修改，即当“教师”表中的“教师编号”值发生改变时，“课堂”表中对应的值也跟着改变。对“课程编号”列设置了级联删除，即当“课程”表中的某个课程被删除时，“课堂”表中的所有相关行也被自动删除。

4. 唯一性约束

唯一性(Unique)约束指定一个或多个列的组合的值具有唯一性，以防止在列中输入重复的值。

唯一性约束与主键约束有类似的功能，两者的区别如下。

(1) 一个表可以定义多个唯一性约束，但只能定义一个主键约束。

(2) 唯一性约束所在的列允许出现空值 NULL(只能出现一个)，但是主键约束所在的列不允许空值。

(3) 主键不可能(或很难)更新，但具有唯一性约束的列可以更新。

可见，主键约束强度大于唯一性约束，因此主键列无须再设定唯一性约束。

定义列级唯一性约束的语法格式如下：

```
[CONSTRAINT constraint_name]
UNIQUE [CLUSTERED | NONCLUSTERED]
```

定义表级唯一性约束的语法格式如下：

```
[CONSTRAINT constraint_name]
UNIQUE [CLUSTERED | NONCLUSTERED]
(column_name [,...n])
```

各项的含义如下。

(1) constraint_name 是约束名称，该名称需符合标识符的命名规则。

(2) [CLUSTERED | NONCLUSTERED]指系统创建唯一性索引时，索引为聚集索引(CLUSTERED)还是非聚集索引(NONCLUSTERED)，默认值为非聚集索引。索引的概念见第 5 章。

(3) column_name 为构成唯一性约束的字段名。

例 3-24 创建“学院”表时，要求“学院名称”具有唯一性。

```
CREATE TABLE 学院
(   学院编号 char(2) PRIMARY KEY,
    学院名称 char(30) UNIQUE,
    学院电话 char(12),
    学院地址  char(50)
)
```

例 3-25 为了避免不方便，在创建“学生”表时，要求同一个班级里没有同名的学生。

```
CREATE TABLE 学生
(  学号 char(10) PRIMARY KEY,
   姓名 char(20),
   性别 char(2),
   籍贯 char(20),
   出生日期 date,
   专业班级 char(30),
   入学时间 date,
   学制 int,
   学院编号 char(2),
   密码 char(20),
   UNIQUE (姓名, 专业班级)
)
```

5. 检查约束

检查(Check)约束对输入列或整个表中的值设置检查条件，以限制输入值，保证数据库的数据完整性。

当对具有检查约束的列进行插入或修改时，SQL Server 将用该检查约束的逻辑表达式对新值进行检查，只有满足条件(逻辑表达式返回 TRUE)的值才能填入该列，否则报错。可以为每列指定多个 CHECK 约束。

表级检查约束和列级检查约束的语法格式相同，定义检查约束的语法格式如下：

```
[CONSTRAINT constraint_name] CHECK (logical_expression)
```

各项的含义如下。

(1) constraint_name 是约束名称，该名称需符合标识符的命名规则。

(2) logical_expression 为对列值进行限制的逻辑表达式，逻辑表达式可以涉及表的多个列。

例 3-26 创建“教师”表时，要求“性别”列的取值只能是“男”或“女”。

```
CREATE TABLE 教师
(  教师编号 char(10) PRIMARY KEY,
   姓名 char(20) NOT NULL,
   性别 char(2) CHECK (性别 = '男' OR 性别 = '女'),
   出生日期 date,
   职称 char(20),
   学院编号 char(2) NOT NULL,
   密码 char(20)
)
```

例 3-27 创建“选课成绩”表时，要求“成绩”列的取值范围为 0～100。

```
CREATE TABLE 选课成绩
(  学号 char(10),
   课堂编号 char(16),
   成绩 int CHECK (成绩 >= 0 AND 成绩 <= 100),
   CONSTRAINT PK_选课成绩 PRIMARY KEY (学号, 课堂编号)
)
```

例 3-28 创建“学院”表时，要求“学院电话”列的取值需符合中国地区固定电话的编码格式。中国地区固定电话号码的编码格式是“区号-号码”。区号为三位数的号码是八位数，区号为四位数的则号码是七位数。区号的第一位数一定为零，第二位数一定不为零；号码的第一位数一定不为零。

```
CREATE TABLE 学院
(  学院编号 char(2) PRIMARY KEY,
   学院名称 char(30) UNIQUE,
   学院电话 char(12),
   学院地址  char(50),
   CHECK (学院电话 LIKE '0[1-9][0-9][0-9]-[1-9][0-9][0-9][0-9][0-9][0-9][0-9]'
       OR 学院电话 LIKE '0[1-9][0-9]-[1-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]')
)
```

在本例中，LIKE 运算符确定给定的字符串是否与指定的模式相匹配，模式可以使用通配符字符。本例中“[ ]”就是通配符，其含义是指定范围中的任何单个字符，[0－9]表示数字字符 0～9 中的任意一个，[1－9]表示数字字符 1～9 中的任意一个。

例 3-29 在“销售管理”数据库中创建“销售订单”表，包括“订单编号”“商品编号”“数量”“价格”“订货日期”“发货日期”“到货日期”字段。定义检查约束以保证发货日期在订货日期之后，到货日期在发货日期之后。

```
CREATE TABLE 销售订单
(  订单编号 char(10) PRIMARY KEY,
   商品编号 char(10),
   数量 float,
   价格 float,
   订货日期 date,
   发货日期 date,
   到货日期 date,
   CHECK (发货日期 > 订货日期 AND 到货日期 > 发货日期)
)
```

6. 默认约束

默认(Default)约束通过定义列的默认值来确保在用户没有为某列指定数据时，系统来指定列的值。

默认约束在对表执行 INSERT 语句时起作用，每列中只能有一个默认约束。默认值可以是常量，也可以是表达式，还可以为 NULL。

定义默认约束的语法格式如下：

```
[CONSTRAINT constraint_name]
DEFAULT constant_expression [FOR column_name]
```

各项的含义如下。

(1) constraint_name 是约束名称，该名称需符合标识符的命名规则。

(2) constant_expression 为默认值取值的常量表达式。

(3) column_name 为默认约束所作用的列。

注意：FOR column_name 子句只能在 ALTER TABLE 语句中使用，在 CREATE

TABLE 语句中不能使用。因此，在 CREATE TABLE 语句中只能定义列级约束，不能定义表级约束。

例 3-30 创建“课程”表时，为“课程性质”列设置默认值“必修”。

```
CREATE TABLE 课程
(  课程编号 char(8) PRIMARY KEY,
   课程名称 varchar(50),
   学时数 int,
   学分数 float,
   课程性质 char(10) CONSTRAINT DF_课程_课程性质 DEFAULT '必修',
   课程介绍 text
)
```

例 3-31 在“销售管理”数据库中创建“销售订单”表，包括“订单编号”“商品编号”“数量”“价格”“订货日期”“发货日期”“到货日期”字段。为“订货日期”列设置默认值。

```
CREATE TABLE 销售订单
(  订单编号 char(10) PRIMARY KEY,
   商品编号 char(10),
   数量 float,
   价格 float,
   订货日期 date DEFAULT getdate(),
   发货日期 date,
   到货日期 date
)
```

在本例中，为“订货日期”列设置的默认值为 getdate()。getdate()为 SQL Server 提供的系统函数，返回值为当前的日期。设置默认约束后，每当在“销售订单表”中添加一行记录时，如果用户没有给出“订货日期”字段的值，则系统会为该字段赋予当前日期。

3.4.4 表的修改

所谓表的修改是指在创建表之后，修改表结构以及添加、删除约束等。在 SQL Server 中，可以使用 SQL Server Management Studio 和 Transact-SQL 语句来修改表。

1. 使用 SQL Server 管理平台修改表

在 SQL Server Management Studio 中的打开“对象资源管理器”面板，展开“数据库”结点下的“表”结点。右击要修改的数据表，从快捷菜单中选择“设计表”命令，则会弹出修改数据表结构的“表设计器”对话框，如图 3.27 所示。

在“表设计器”中可执行增加列、删除列、修改列属性等操作，修改完成后，单击“保存”按钮 可保存修改。

2. 使用 Transact-SQL 语句修改表

在 SQL Server 中，可以使用 ALTER TABLE 语句来修改表结构、添加/删除约束。

(1) 使用 ALTER TABLE 语句修改表结构。

语法格式如下：

```
ALTER TABLE table_name
ADD {< column_definition > | < computed_column_definition >} [,...n]
```

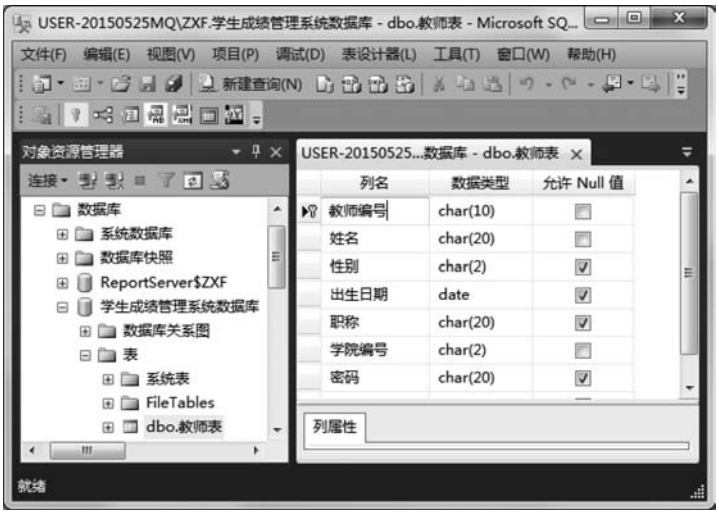

图 3.27　在“表设计器”中修改表结构

```
| ALTER COLUMN column_name data_type [NULL|NOT NULL]
| DROP COLUMN column_name [,...n]
```

从上面的语法格式可以看出，ALTER TABLE 有三个子句（ADD 子句、ALTER COLUMN 子句和 DROP COLUMN 子句），一条 ALTER TABLE 语句中只能写一个子句。

ADD 子句的功能是为表增加一列或多列。< column_definition >和< computed_column_definition >的具体语法格式与 CREATE TABLE 语句中的< column_definition >和< computed_column_definition >相同。

例 3-32　为“学生”表添加“电话”和“邮箱”列。

```
ALTER TABLE 学生
ADD 电话 char(20), 邮箱 char(30)
```

ALTER COLUMN 子句的功能是修改列定义。使用 ALTER COLUMN 子句时要注意以下五点。

① ALTER COLUMN 子句不能修改列名。

② 如果列中已有数据，则不能减少该列的宽度，也不能改变其数据类型。

③ 不能将含有空值的列的定义修改为 NOT NULL 约束。

④ 只能修改 NULL/NOT NULL 约束，其他类型的约束在修改之前必须先将约束删除，然后再重新添加修改过的约束定义。

⑤ 当修改列的 NULL/NOT NULL 约束而不修改列的数据类型时，在列定义中要定义该列的数据类型；当修改列的数据类型而不修改列的 NOT NULL 约束时，要在列定义中写 NOT NULL。

例 3-33　修改“学生”表的“邮箱”列的长度为 40。

```
ALTER TABLE 学生
ALTER COLUMN 邮箱 char(40)
```

例 3-34 将“学生”表的“性别”列设置为非空值约束。

```
ALTER TABLE 学生
ALTER COLUMN 性别 char(2) NOT NULL
```

DROP COLUMN 子句的功能是删除一列或多列。

例 3-35 删除“学生”表的“电话”和“邮箱”列。

```
ALTER TABLE 学生
DROP COLUMN 电话, 邮箱
```

(2) 使用 ALTER TABLE 语句添加/删除约束。

语法格式如下：

```
ALTER TABLE table_name
ADD < constraint_definition > [,...n]
| DROP CONSTRAINT constraint_name [,...n]
```

ALTER TABLE 语句通过 ADD 子句为表添加一个或多个约束，在 ALTER TABLE 语句中添加的约束都是表级约束，各种约束的具体语法格式见 3.4.3 节。

例 3-36 为“课堂”表添加主键约束，定义主键为“课堂编号”列。

```
ALTER TABLE 课堂
ADD CONSTRAINT PK_课堂 PRIMARY KEY (课堂编号)
```

例 3-37 为“教师”表的“学院编号”列添加外键约束，对应的主键为“学院”表的“学院编号”列。

```
ALTER TABLE 教师
ADD CONSTRAINT FK_教师_学院编号 FOREIGN KEY(学院编号)
REFERENCES 学院(学院编号)
```

例 3-38 设“学生”表中有一个“身份证号码”列，为此列添加唯一性约束。

```
ALTER TABLE 学生
ADD CONSTRAINT UQ_学生_身份证号码 UNIQUE (身份证号码)
```

例 3-39 为“学生”表的“邮箱”列添加检查约束，要求必须出现“@”符号。

```
ALTER TABLE 学生
ADD CONSTRAINT CK_学生_邮箱 CHECK (邮箱 LIKE '%@%')
```

本例中，通配符“%”的含义是包含 0 个或多个字符的任意字符串。

例 3-40 为“教师”表的“密码”列设置默认约束，默认值为“123456”。

```
ALTER TABLE 教师
ADD CONSTRAINT DF_教师_密码 DEFAULT '123456' FOR 密码
```

ALTER TABLE 语句中通过 DROP CONSTRAINT 子句可删除表的一个或多个约束。

例 3-41 删除“学生”表中的约束名为“CK_学生_邮箱”的约束。

```
ALTER TABLE 学生
DROP CONSTRAINT CK_学生_邮箱
```

3. 使用系统存储过程修改数据库对象名

ALTER TABLE 语句中的 ALTER COLUMN 子句无法修改列名或约束名。当需要改名时，可使用系统存储过程 sp_rename 来实现。系统存储过程 sp_rename 的功能是更改当前数据库中用户创建对象（如表、列、索引、视图或用户定义数据类型等）的名称。sp_rename 的调用格式如下：

```
EXEC sp_rename 'object_name', 'new_name'[, 'object_type']
```

各项的含义如下。

（1）object_name 是用户对象（表、视图、列、约束、存储过程、触发器、数据库或数据类型）的当前名称。如果要重命名的对象是表中的一列，那么 object_name 必须为 table_name.column_name 形式。如果要重命名的是索引，那么 object_name 必须为 table_name.index_name 形式。

（2）new_name 是指定对象的新名称。

（3）object_type 是要重命名的对象的类型，其取值有如下选择。

① COLUMN：要重命名的列。

② DATABASE：要重命名的、用户定义的数据库。

③ INDEX：用户定义的索引。

④ OBJECT：在 sysobjects 中跟踪的类型的项目。例如，OBJECT 可用来重命名约束（PRIMARY KEY、FOREIGN KEY、UNIQUE、CHECK、DEFAULT）、用户表、视图、存储过程和触发器等对象。

⑤ USERDATATYPE：通过执行 sp_addtype 而添加的用户定义数据类型。

例 3-42 将“课程”表中的“学院编号”列的名称改为“开课院系”。

```
EXEC sp_rename '课程.学院编号', '开课院系'
```

本例中，对于当前的列名“学院编号”，必须在前面加上表名，否则系统无法判断此列属于哪个表。

例 3-43 将在例 3-36 中创建的主键约束“PK_课堂”更名为“PK_课堂_课堂编号”。

```
EXEC sp_rename 'PK_课堂', 'PK_课堂_课堂编号'
```

3.4.5 表的删除

在 SQL Server 中，可以使用 SQL Server Management Studio 和 Transact-SQL 语句来删除表。

1. 使用 SQL Server 管理平台删除表

打开 SQL Server Management Studio，在“对象资源管理器”中右击要删除的表，在弹出的快捷菜单中选择“删除”命令，打开如图 3.28 所示的“删除对象”窗口。单击“显示依赖关系”按钮，弹出“依赖关系”对话框，其中列出了表所依赖的对象和依赖于表的对象，有对象依赖于表时不能删除表。单击“确定”按钮完成表的删除。

2. 使用 Transact-SQL 语句删除表

在 SQL Server 中，可以使用 DROP TABLE 语句来删除表。DROP TABLE 语句的语

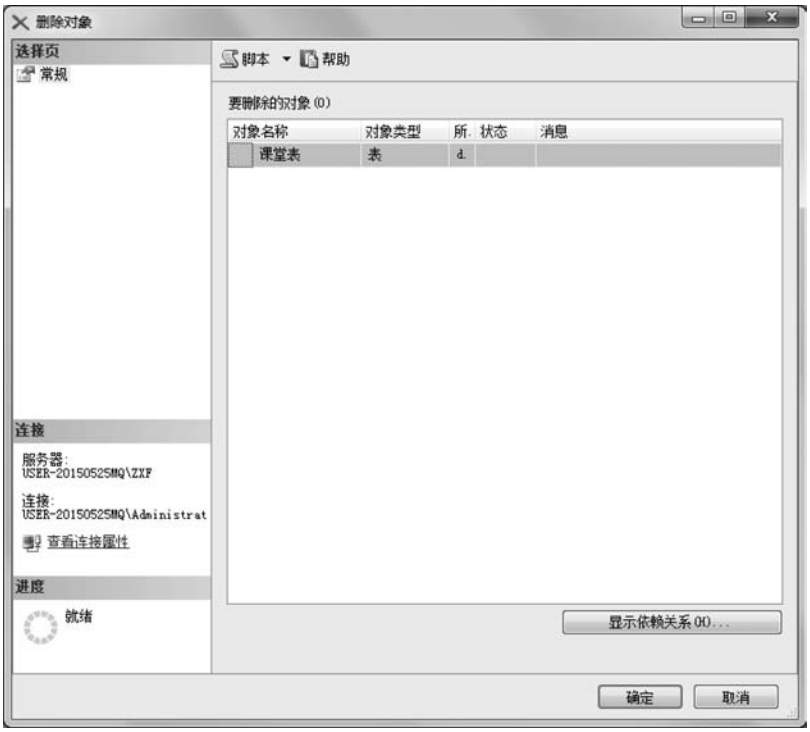

图 3.28 “删除对象”窗口

法格式如下：

```
DROP TABLE table_name [,...n]
```

其中，table_name 是要删除的表。

例 3-44 删除“学生成绩管理系统数据库”中的“课堂”表。

```
DROP TABLE 课堂
```

例 3-45 设已完成例 3-37 的操作，为“教师”表的“学院编号”列添加了外键约束，对应的主键为“学院”表的“学院编号”列。要通过 DROP TABLE 语句删除“学院”表，是否能够删除成功？

```
DROP TABLE 学院
```

该语句无法成功执行，运行结果如下：

```
无法删除对象'学院',因为该对象正由一个 FOREIGN KEY 约束引用。
```

3.5 表中数据的维护

数据库的主要用途是存储数据并使授权的应用程序和用户能够使用这些数据。在数据库中的表对象建立后，用户对表的访问可分为数据查询和数据操纵两类。其中，数据查询是指检索数据但不更改数据，数据操纵则以三种方式更改数据，分别是向表中添加若干行数

据、修改表中的数据和删除表中的若干行数据。本节介绍数据操纵功能的实现方法。

3.5.1 使用 SQL Server 管理平台维护表中数据

打开 SQL Server Management Studio，在“对象资源管理器”中右击要更改数据的表，在弹出的快捷菜单中选择“编辑前 200 行”命令，打开如图 3.29 所示的数据编辑窗口。

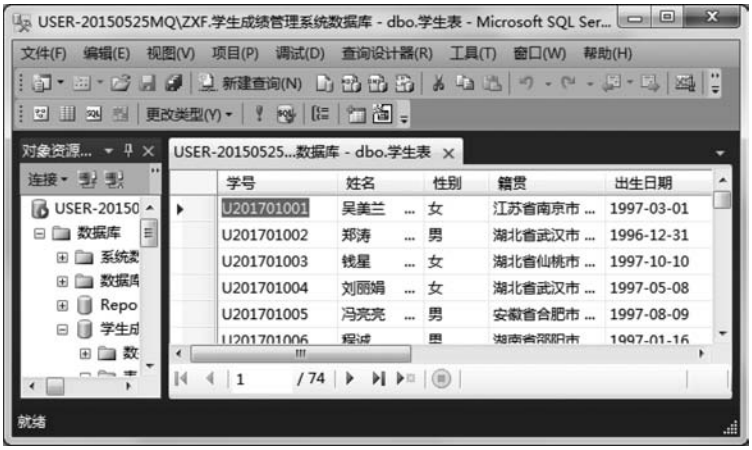

图 3.29 数据编辑窗口

1. 插入数据

数据编辑窗口中的最后一行为空行，所有的字段值都为 NULL，在此行写入新行的各个字段的值即可插入数据。当输入一个新记录的数据后，会自动在最后出现一行新的空白行，用户可以继续输入多个数据记录。

注意：为了保证数据完整性，输入数据应一行一行地输入，输入完一个记录的所有列值（具有空值约束或默认值约束的列、计算列可以不输入）后，再输入下一行的数据。

2. 修改数据

单击要修改数据值的单元格，向单元格中输入新值后，原数据被新数据覆盖。

3. 删除数据

用鼠标选中一行或多行数据并右击，在弹出的快捷菜单中选择“删除”命令，或者按 Del 或 Delete 键，在弹出的删除提示框中选择“是”，则记录被删除。

3.5.2 使用语句维护表中数据

1. 插入数据

使用 INSERT 语句插入数据，INSERT 语句通常有两种形式，一种是插入一个元组。另一种是插入子查询的结果，后者可以一次插入多个元组。本章只介绍前者。INSERT 语句的语法格式如下：

```
INSERT [INTO] table_name [(column_name [,...n])]
VALUES (value [,...n])
```

各项的含义如下。

(1) table_name 是要插入数据的表名，该表必须已存在。

(2) column_name 是要在其中插入数据的列名。column_name 可以写多个构成列名列表,其间以逗号分隔。

(3) value 为插入的数据值。多个 value 之间以逗号分隔,构成值列表。

注意:

(1) column_name 和 value 是一一对应的关系,即第一个 value 值赋予第一个 column_name,第二个 value 值赋予第二个 column_name,以此类推,因此 column_name 和 value 在个数和数据类型上必须一致。

(2) 对于列名列表中没有出现的列,如果此列具有默认值,则新添加的元组在该列将具有默认值,否则将赋 NULL 值。如果某列没有出现在列名列表中,同时具有 NOT NULL 约束,并且没有默认值,则此条 INSERT 语句将执行失败。

(3) 在 INSERT 语句中可以不写列名列表,此时值列表中的值将按创建表时的列顺序和个数赋予各个列。

例 3-46 开学初,郑涛同学(学号为"U201701002")选修了 Python 这门选修课(对应的课堂编号为"2017-2018-2-B009"),此时还没有考试,因此还没有成绩。将此信息写入"选课成绩"表中。

```
INSERT INTO 选课成绩(学号, 课堂编号)
VALUES ('U201701002', '2017 - 2018 - 2 - B009')
```

本例中,没有为新记录的"成绩"字段赋值,如果此字段具有默认值,则会被赋予默认值;如果没有默认值,则会被赋予 NULL 值。

例 3-47 土木学院(学院编号为"02")新进了一名教师,姓名为"徐强",性别为"男",出生日期为 1993 年 7 月 1 日,职称为助教,密码为"19930701",分配给他的教师编号为"T013"。将此教师的信息写入"教师"表中。

```
INSERT INTO 教师
VALUES ('T013', '徐强', '男', '1993 - 7 - 1', '助教', '02', '19930701')
```

本例为新记录的所有字段依次赋值,因此可以省略列名列表。

2. 修改数据

修改数据是指修改表中指定元组的指定列的值。修改操作由 UPDATE 语句完成,其语法格式如下:

```
UPDATE table_name
SET {column_name = expression [,...n]}
[WHERE < search_condition >]
```

各项的含义如下。

(1) table_name 是要修改数据的表名,该表必须已存在。

(2) column_name 是要修改数据的列名。

(3) expression 为一个表达式,是赋予列的新值。

(4) < search_condition >为一个条件表达式,满足条件的元组才会被修改数据。

注意: WHERE 子句可省略,此时将修改该表所有元组对应的列值。

例 3-48 将教师编号为 T013 的教师的性别由“男”改为“女”。

```
UPDATE 教师 SET 性别 = '女' WHERE 教师编号 = 'T013'
```

本例使用了 WHERE 子句,修改了表中的一条记录中的“性别”字段的值。

例 3-49 将“选课成绩”表中的所有成绩乘以 0.9。

```
UPDATE 选课成绩 SET 成绩 = 成绩 * 0.9
```

本例没有使用 WHERE 子句,因此更改了“选课成绩”中的所有记录的“成绩”字段的值。

例 3-50 将“学院”表中“学院电话”字段的所有本地电话号码前加上区号。判断本地电话号码的条件是:“电话”列的字符串长度为 8。

```
UPDATE 学院 SET 学院电话 = '027 - ' + 学院电话 WHERE LEN(学院电话) = 8
```

本例中,LEN( )为 SQL Server 提供的系统函数,返回值为字符串的长度。语句中的运算符“+”为字符串运算符,用于连接两个字符串数据。

3. 删除数据

使用 DELETE 语句删除数据,DELETE 语句可以删除表中的一行或多行数据,其语法格式如下:

```
DELETE [FROM] table_name
[WHERE < search_condition >]
```

各项的含义如下。

(1) table_name 是要删除数据的表名,该表必须已存在。

(2) < search_condition >为一个条件表达式,满足条件的元组才会被删除。

注意:WHERE 子句可省略,此时将删除该表的所有元组。

例 3-51 删除“教师”表中“教师编号”为 T013 的教师记录。

```
DELETE 教师 WHERE 教师编号 = 'T013'
```

本例使用了 WHERE 子句,删除了一条记录。

例 3-52 删除“选课成绩”表中的所有记录。

```
DELETE 选课成绩
```

本例没有使用 WHERE 子句,因此删除了表中的所有记录。

本 章 小 结

本章介绍了 SQL Server 2012 数据库管理系统及其基本操作。

(1) SQL Server 2012 是一个基于客户机/服务器应用模式的关系型数据库管理系统。用户可以通过图形化的管理工具和 Transact-SQL 两种方式浏览和修改数据库中的数据,配置数据库系统参数。

(2) 数据库是 SQL Server 服务器管理的基本单位。从逻辑上看,SQL Server 数据库是由数据表、视图、存储过程、触发器等逻辑组件组成的,这些逻辑组件称为数据库对象。从物

理存储角度来看，数据库中的各种信息是以文件为单位存放在存储设备上的。

(3) 在 SQL Server 2012 中有两类数据库——系统数据库和用户数据库。系统数据库存储有关 SQL Server 的系统信息，它们是 SQL Server 管理数据库的依据。SQL Server 2012 有五个系统数据库，分别是 master、tempdb、model、msdb 和 resource。用户创建的数据库称为用户数据库。创建、修改和删除用户数据库有两种常用方法：一是使用 SQL Server 管理平台；二是使用 CREATE DATABASE 语句、ALTER DATABASE 语句和 DROP DATABASE 语句。

(4) 表是数据库中数据的实际存储处，每个表代表一个实体集或实体集间的联系。表由行和列组成，每行标识一个实体，每列代表实体的一个属性。可使用 SQL Server 管理平台或 CREATE TABLE 语句、ALTER TABLE 语句、DROP TABLE 语句来创建表、修改表结构和删除表。可使用 SQL Server 管理平台或 INSERT 语句、UPDATE 语句、DELETE 语句来对表中的数据进行插入、修改和删除。

(5) 约束是数据库维护数据完整性的一种机制。SQL Server 2012 支持空值/非空值约束、主键约束、外键约束、唯一性约束、检查约束、默认约束六种约束。约束可以在 CREATE TABLE 语句中定义，也可以在 ALTER TABLE 语句中添加和删除。

习　题　3

一、单选题

(1) 以下关于 SQL Server 数据库的叙述中，错误的是_____。

A. 从用户角度观察数据库，看到的是多种逻辑组件(数据库对象)

B. 从存储角度观察数据库，看到的是数据文件和事务日志文件

C. 数据表中的数据可以存放在事务日志文件中

D. 用户无须直接访问数据库文件

(2) 以下不属于 SQL Server 数据库对象(逻辑组件)的有_____。

A. 表　　B. 文件　　C. 视图　　D. 索引

(3) SQL Server 的数据文件可分为_____。

A. 重要文件和次要文件　　B. 主数据文件和次数据文件

C. 初始文件和最大文件　　D. 初始文件和增长文件

(4) 事务日志用于保存_____。

A. 程序运行过程　　B. 程序的执行结果

C. 对数据的更新操作　　D. 对数据的查询操作

(5) 安装 SQL Server 后，数据库服务器中已经自动建立了系统数据库，以下_____不是系统数据库。

A. master 数据库　　B. pubs 数据库

C. model 数据库　　D. msdb 数据库

(6) 在 SQL Server 中，model 数据库是_____。

A. 数据库系统表　　B. 数据库模板

C. 临时数据库　　D. 示例数据库

(7) 下列标识符中,符合 SQL Server 标识符命名规则的有_____。

A. zhong guo　　B. @sum　　C. 1_student　　D. create

(8) 在 SQL 语法中,用来插入和修改数据的命令是_____。

A. INSERT,UPDATE　　B. DELETE,INSERT

C. DELETE,UPDATE　　D. CREATE,INSERT

(9) 若要删除 book 表中的所有数据,以下语句正确的是_____。

A. DROP FROM book　　B. DROP TABLE book

C. DELETE FROM book　　D. DELETE * FROM book

(10) 参照完整性要求有关联的两个或两个以上表之间数据的一致性。参照完整性可以通过建立_____来实现。

A. 主键约束　　B. 默认约束　　C. 唯一性约束　　D. 外键约束

二、填空题

(1) SQL Server 中,如果没有指定文件组,则_____是默认文件组。

(2) SQL Server 数据库分为_____数据库和用户数据库。

(3) master 数据库记录 SQL Server 系统的所有_____信息,如初始化信息、所有的登录账户和系统配置设置等。

(4) 删除"教务管理"数据库的语句为_____ DATABASE 教务管理。

(5) 在 Transact-SQL 语句中,表示字符串型常量数据时,应使用_____符号将字符串括起来。

(6) SQL 语言集数据查询、数据操纵、数据定义和数据控制功能于一体,其中,CREATE、DROP、ALTER 语句分别实现_____功能。

(7) 修改数据表的字段名称可使用系统存储过程_____。

(8) 利用 CREATE TABLE 语句创建表 Student:

```
CREATE TABLE Student
(   SNO CHAR(10) NOT NULL,
    SNAME CHAR(6) NOT NULL,
    AGE INT,
    NOTE CHAR(20)
)
```

则 SNAME 列的类型为_____型,列宽度为_____。

(9) 表设计器的"允许空"单元格用于设置该字段是否可输入空值,实际上就是创建该字段的_____约束。

(10) 一张表的主键个数为_____。

三、判断题

(1) 每个数据库有且仅有一个主数据文件。

(2) 主数据文件可以不在主文件组中。

(3) 一个数据库可以没有次数据文件。

(4) 在 SQL Server 中,数据信息和日志信息不能放在同一文件中。

(5) 如果用户定义的标识符与系统保留字重名,则需要使用双引号(" ")或方括号([ ])

进行分隔处理。

(6) 在 Transact-SQL 语句中,不区分英文字母的大小写(字符串常量除外)。

(7) 空字符串是空值(NULL)。

(8) 删除数据库时,组成该数据库的所有磁盘文件将同时被删除。

(9) 一条 DROP DATABASE 语句只能删除一个数据库。

(10) ALTER TABLE 语句可以修改表中数据。

四、应用题

(1) 设有一个“图书出版”数据库,其中有“图书”和“出版社”两个数据表,表模式如下:

图书(书号,类型,书名,作者,单价,出版社号)
出版社(出版社号,出版社名称,电话)

使用 Transact-SQL 语句完成下列操作。

① 创建“图书出版”数据库。

② 创建“图书”表。

③ 删除“图书”表的“类型”字段,增加“出版日期”字段。

④ 在“图书”表中增加一条记录:

书号为 B001,书名为大数据分析导论,出版日期为 2020 年 9 月 1 日,作者为金大卫,单价为 59.90 元,出版社号为 P302。

⑤ 在“出版社”表中,将出版社名称为“清华出版社”的记录的出版社名称改为“清华大学出版社”。

⑥ 在“图书”表中删除作者为“唐七公子”的所有图书。

⑦ 删除“图书”表。

(2) 利用上一小题创建的“图书”表和“出版社”表,使用 ALTER TABLE 语句为两张表添加或删除约束。

① 为“出版社”表添加主键约束,该约束由“出版社号”单列组成。

② 为“出版社”表添加约束,使“出版社名称”列具有唯一性。

③ 为“图书”表添加外键约束,使之与“出版社”表建立关联。

④ 为“图书”表添加约束,使其“单价”列的默认值为 0。

⑤ 为“图书”表添加约束,确保“单价”列值大于或等于 0。

⑥ 为“图书”表添加名称为“DF_图书_出版日期”的约束,使得出版日期的默认值为当天日期。

⑦ 删除名称为“DF_图书_出版日期”的约束。