第 3 章　创建和管理 SQL Server 2005 数据库

数据库是一个应用系统的核心，设计并管理好数据库是进行项目开发的首要任务。本章主要介绍创建、删除、修改数据库的方法及实用的数据库管理技术，包括备份和还原数据库、分离和附加数据库、数据库中数据的导入和导出等。

3.1　系统数据库概述

SQL Server 2005 数据库包括表、视图、索引、存储过程和触发器等对象。数据库提供对这些对象的管理，各个对象的简要说明见表 3-1。

表 3-1　SQL Server 2005 数据库对象

数据库对象	说　　明
表	用来存储数据，是行和列构成的集合
视图	由表或其他视图导出的虚表
索引	为数据快速检索提供支持的一种数据结构，且可以保证数据唯一性
存储过程	存放在服务器的一组预先编译好的 Transact-SQL 语句
触发器	一种特殊的存储过程，在用户表数据发生变化时将被自动执行
约束	用于定义表中列完整性的规则

通过 SQL Server 2005“对象资源管理器”窗口，可以查看当前数据库内的各种数据库对象，如图 3-1 所示。

图 3-1　“对象资源管理器”窗口

3.1.1　SQL Server 数据库的文件组成

SQL Server 2005 一方面将数据组织成表、视图等逻辑对象；另一方面，为了数据库管理员管理数据的方便，SQL Server 2005 数据库又表现为各种数据库文件，所有数据库都至少包含一个主数据库文件和一个日志文件。此外，还可以包含零个或多个辅助数据库文件，这称为数据库的物理结构。

1. 文件

SQL Server 数据库主要由两种文件组成：数据库文件和事务日志文件。

1)数据库文件

数据库文件用来存放数据库数据和数据库对象。一个数据库可以有一个或多个数据库文件,一个数据库文件只能属于一个数据库。当有多个数据库文件时,有一个文件被定义为主数据库文件(primary database file),扩展名为.mdf,它用来存储数据库的启动信息和部分或全部数据。一个数据库只能有一个主数据库文件。其他数据库文件被称为次数据库文件(secondary database file),扩展名为.ndf,用来存储那些不能存储在主数据库文件中的数据。如果主数据库文件足够大,能够容纳数据库中的所有数据,则该数据库不需要次数据库文件。但如果数据量非常大,则需要多个次数据库文件,或者在不同硬盘中存储次数据库文件,这样可以同时对几个硬盘进行数据存取,提高了数据处理的效率,对于服务器型的计算机尤为有用。

2)事务日志文件

事务日志文件包含用于恢复数据库的日志信息。一个数据库必须至少有一个事务日志文件。日志文件最小为 512 KB,扩展名为.ldf。

SQL Server 中采用提前写事务的方式,即对数据库的修改先写入事务日志,再写入数据库。SQL Server 有个特点,即在执行数据更改时会设置一个开始点和一个结束点,如果尚未到达结束点而因某种原因使操作中断,则在 SQL Server 重新启动时会自动恢复已修改的数据,使其返回未被修改的状态。可见,数据库被破坏后可以用事务日志恢复数据库内容。

2.文件组

文件组是为了管理和分配数据而将多个数据库文件集合起来形成的一个整体。每个文件组有一个组名。通常为一个硬盘驱动器创建一个文件组,将特定的表、索引与文件组相关联,则对这些表的存储、查询和修改等操作都在该文件组中。使用文件组可以提高表中数据的查询性能。

文件组可分为以下两类:

(1)主文件组:包含主要文件的文件组,所有系统表都被分配到该文件组中。

(2)用户自定义文件组:用户首次创建数据库或修改数据库时明确创建的文件组。

说明:与数据文件一样,文件组分为主文件组和次文件组。一个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。

3.1.2　系统数据库

SQL Server 2005 支持在一台服务器上创建多个数据库,每个数据库可以存储相关的数据。从数据库管理的角度来看,SQL Server 2005 数据库分为系统数据库、数据库快照和用户数据库。其中,数据库快照是源数据库的只读、静态视图,每个数据库快照都与创建快照时存在的源数据库在事务上一致。

SQL Server 2005 中的系统数据库主要包括以下几种。

1.master 数据库

SQL Server 2005 中,master 数据库用来记录与数据库有关的系统信息,包括登录信息、系统配置、数据库错误信息、SQL Server 初始化信息、系统中其他系统数据库和用户数据库的相关信息等。因此,对 master 数据库的任何修改都可能影响系统运行。

提示:通常不在 master 数据库中保留任何非系统级信息。当对 SQL Server 进行任何修改时,最好备份 master 数据库,以便数据库崩溃时进行恢复和补救。

2. model 数据库

model 数据库是 SQL Server 2005 中的模板数据库,包含 19 个系统表和一些视图,其中包含的各个系统表为每个用户数据库所共享。创建一个用户数据库时,系统会将 model 数据库中的内容复制到新建的数据库中去。用户可以利用 model 数据库的模板特性,通过更改 model 数据库的设置将常用的数据库对象复制到 model 数据库中,这样可以大大简化数据库及其对象的创建、设置工作,节省用户大量的时间。

3. msdb 数据库

msdb 数据库供 SQL Server 代理程序调度警报和作业以及记录各种操作,同时存储所有备份历史。SQL Server Agent 可能会使用这个库。

4. tempdb 数据库

tempdb 数据库用作系统的临时存储空间,主要作用有:存储用户建立的临时表和临时存储过程,存储用户说明的全局变量值,为数据排序创建临时表,存储用户利用游标说明所筛选出来的数据。当退出 SQL Server 时,用户在 tempdb 数据库中建立的所有对象都将被删除。每次 SQL Server 启动时 tempdb 数据库都将被重建,恢复到系统设定的初始状态,因此永远不要在 tempdb 数据库中建立需要永久保存的表。使用 tempdb 数据库不需要特殊的权限,不管 SQL Server 中安装了多少个数据库,tempdb 数据库只有一个。

3.2 创建数据库

创建数据库有两种方式:通过 SQL Server Management Studio 或者通过 Transact-SQL 语句。创建数据库时,必须确定数据库名、所有者(即创建数据库的用户)、数据库大小和存储数据库的文件。需要注意的是,必须是系统管理员或者被授权使用 CREATE DATABASE 语句的用户才能创建数据库。

3.2.1 使用 SQL Server Management Studio 创建数据库

假设用户以管理员的身份登录,下面以创建"图书管理数据库"library 为例,说明使用 SQL Server Management Studio 创建数据库的一般操作步骤:

(1)启动 SQL Server Management Studio,登录到指定的服务器。

(2)在"对象资源管理器"窗口中,选择要创建数据库的服务器并展开,右击"数据库"文件夹,在弹出的快捷菜单中选择"新建数据库"命令,如图 3-2 所示。

图 3-2 选择"新建数据库"命令

(3)弹出"新建数据库"窗口,如图 3-3 所示,默认显示"常规"选项,在此,用户需要输入新建数据库所需的各项参数,包括数据库名称、所有者和文件路径等。

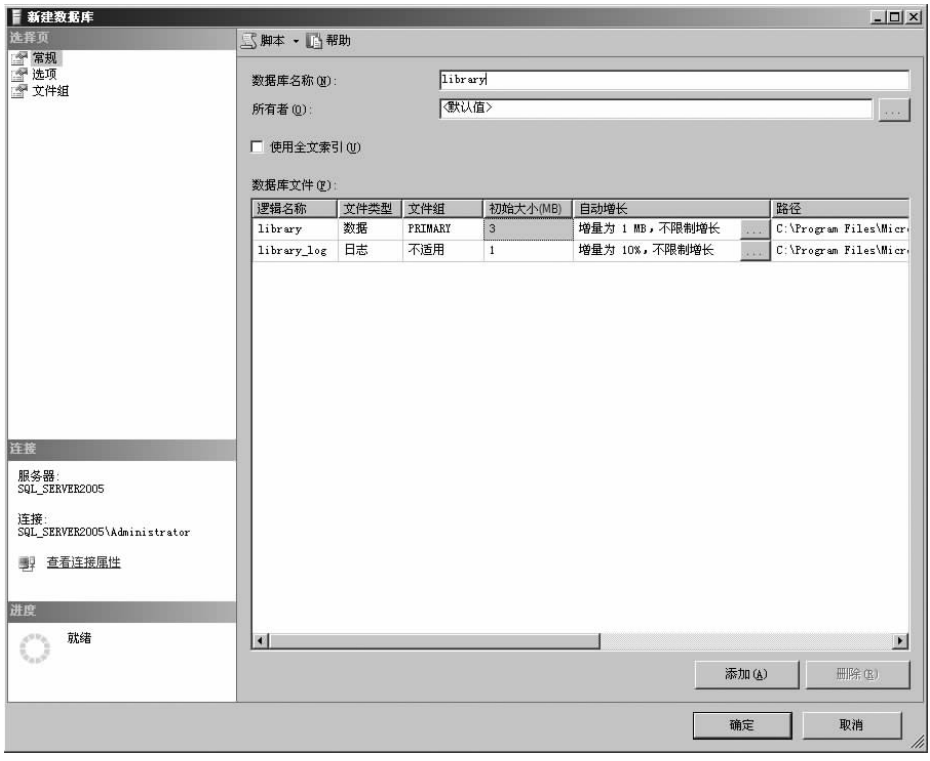

图 3-3　“新建数据库”窗口——“常规”选项

- 在“数据库名称”文本框中输入要创建的数据库名，本例输入 library。
- 在“所有者”文本框中选择数据库所有者。数据库所有者对数据库具有完全操作权限，默认值为当前登录到 SQL Server 服务器的登录名。若要更改所有者，单击“所有者”文本框后面的 ... 按钮，出现“选择数据库所有者”对话框，如图 3-4 所示。在“选择这些对象类型”列表框中给出了所有者的类型，单击旁边的“对象类型”按钮进行修改；在“输入要选择的对象名称(示例)”列表框中显示要选择的对象名称，单击“浏览”按钮，将打开“查找对象”对话框，如图 3-5 所示，其中列出了匹配所选类型的对象，可单击对象前的复选框，选中要作为数据库所有者的对象，然后单击“确定”按钮返回；在“选择数据库所有者”对话框中，单击“确定”按钮，返回“新建数据库”窗口。本例中使用默认设置。
- 系统默认的数据文件大小为 3 MB，但当存入大量数据使得数据文件超过初始设定值时，系统将按照“自动增长”栏中设定的数值对数据文件规模进行扩大。当数据量较大且一次写入的数据很多时，可以考虑对“自动增长”项进行修改。单击“自动增长”项后的 ... 按钮，弹出如图 3-6 所示的“更改 library 的自动增长设置”对话框，用户可根据个人要求进行设置。
- 默认情况下，系统自动使用指定的数据库名作为前缀来创建主数据库文件和日志文件，如本例中 library.mdf 和 library_log.ldf。
- 默认情况下，系统认为数据库文件只有一个。但如果需要建立辅助数据库文件，可单击“新建数据库”窗口中的“添加”按钮，为新建的辅助数据库文件命名、指定文件类型以及大小、文件分组等，如图 3-7 所示。

图 3-4 “选择数据库所有者”对话框

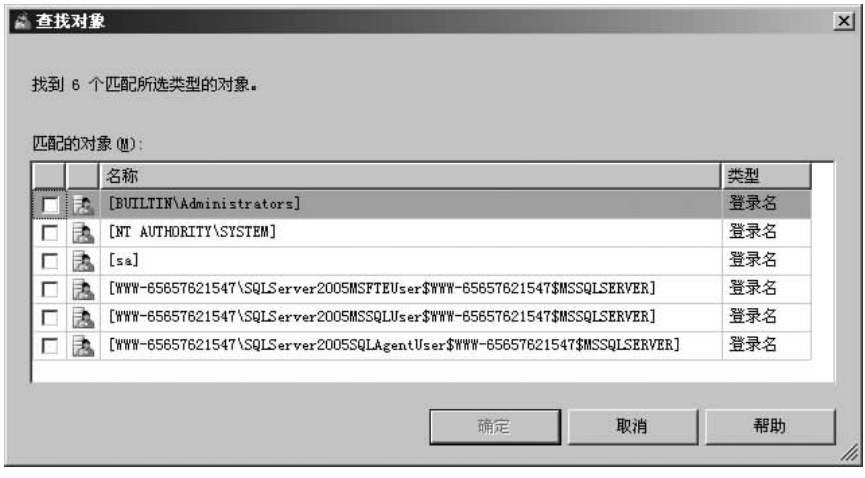

图 3-5 “查找对象”对话框

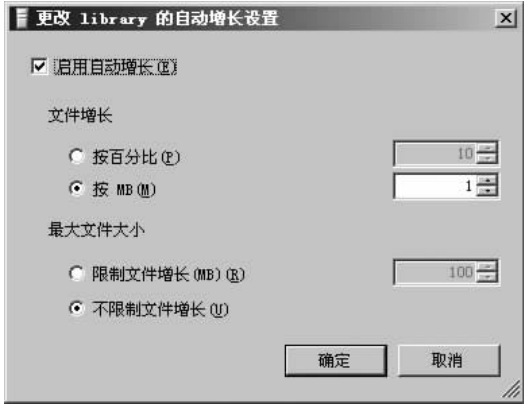

图 3-6 “更改 library 的自动增长设置”对话框

• 在“新建数据库”窗口中选中“使用全文索引”复选框表示为数据库中的变长复杂数据类型列建立索引。

图 3-7　添加数据库文件并设置其参数

(4)设置好所有参数后，单击“确定”按钮即可创建所定义的数据库。此时，“对象资源管理器”窗口的数据库树状菜单中会出现刚才所建的名为 library 的数据库，如图 3-8 所示。

图 3-8　新创建的数据库 library

3.2.2　使用 Transact-SQL 创建数据库

Transact-SQL 语句中的 CREATE DATABASE 命令用来创建数据库。该命令的基本语法如下：

```
CREATE DATABASE database_name
[ON
      [PRIMARY][<filespec>[,…n]]
      [,<filegroup>[,…n]]
]
[LOG ON{<filespec>[,…n]}]
[
      [COLLATE collation_name]
      [FOR LOAD|FOR ATTACH]
]
<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|%]]
)[,…n]
}
<filegroup>::=
{
    FILEGROUP filegroup_name[DEFAULT]
    <filespec>[,…n]
}
```

该命令中各参数说明如下：

- database_name：新数据库的名称。数据库名称在 SQL Server 的实例中必须唯一，并且必须符合标识符命名规则，最多可以包含 128 个字符。
- ON：显式定义用来存储数据库数据部分的磁盘文件(数据文件)，其后跟以逗号分隔的、用以定义主文件组的数据文件的<filespec>项列表。主文件组的文件列表项后跟以逗号分隔的、用以定义用户文件组及文件的<filegroup>项列表，该列表项为可选项。
- PRIMARY：指定关联的<filespec>列表定义主文件。在主文件组的<filespec>项中指定的第一个文件将成为主文件。一个数据库只能有一个主文件；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。
- LOG ON：显式定义用来存储数据库日志的磁盘文件(即事务日志文件)。LOG ON 后跟以逗号分隔的、用以定义日志文件的<filespec>项列表。如果没有指定 LOG ON，将自动创建一个日志文件，其大小为该数据库的所有数据文件大小总和的 25%或 512 KB，取两者之中的较大者。
- COLLATE collation_name：指定数据库的默认排序规则。排序规则名称既可以是 Windows 排序规则名称，也可以是 SQL 排序规则名称。如果没有指定排序规则，则将 SQL Server 实例的默认排序规则分配为数据库的排序规则。
- FOR LOAD：用于兼容早期版本的 SQL Server。
- FOR ATTACH：指定通过附加一组现有的操作系统文件来创建数据库。FOR ATTACH 必须有一个指定主文件的<filespec>项。
- NAME：用于为<filespec>定义的数据库文件指定逻辑名称，存储时未指定逻辑名称则使用 database_name 作为逻辑名称。
- FILENAME：用于为数据库文件指定物理名称。如果未指定该名称则将 NAME 值后缀.mdf 或后缀.ldf 作为数据库文件的物理文件名，存储路径可根据需要来决定。
- SIZE：用于指定数据库文件的初始大小，如果没有指定，则采用默认大小，即数据文件 3 MB，日志文件 1 MB。
- MAXSIZE：指定数据库文件能够增长到的最大值。如果取值为 UNLIMITED 表示无穷大，实际上由于受磁盘控件的限制，SQL Server 2005 中指定 UNLIMITED 的日志文件最大值为 2 TB，数据文件的最大值为 16 TB。
- FILEGROWTH：用于指定文件大小增长的幅度，可以用百分比或者绝对值。设置为 0 时表明关闭自动增长功能。

• FILEGROUP：用于定义文件组。其中，filegroup_name 为文件组的逻辑名称，在数据库中必须是唯一的，不能是系统提供的名称 PRIMARY 或 PRIMARY_LOG。当设参数 DEFAULT 时，表示将该文件组设置为数据库中默认的文件组。

【例 3-1】 建一个名为 librarySQL 的数据库。该数据库有 3 个数据库文件，2 个文件组。主文件组包括主数据库文件 librarySQL_data，文件大小为 5 MB，按 15%增长，最大为 80 MB；第二个文件组名为 Secondary，包括文件 librarySQL2_data，文件初始大小为 2 MB，按 10%增长，最大为 100 MB；日志文件大小为 3 MB，最大为 30 MB，按 1 MB 增长。

操作步骤如下：

(1)单击 SQL Server Management Studio 中工具栏上的“新建查询”按钮，或者执行“文件”→“新建”→“数据库引擎查询”命令，打开一个新的查询编辑器。在工具栏“可用数据库”下拉列表框中选择 master 数据库。

(2)如图 3-9 所示，在查询编辑器窗口中输入以下语句：

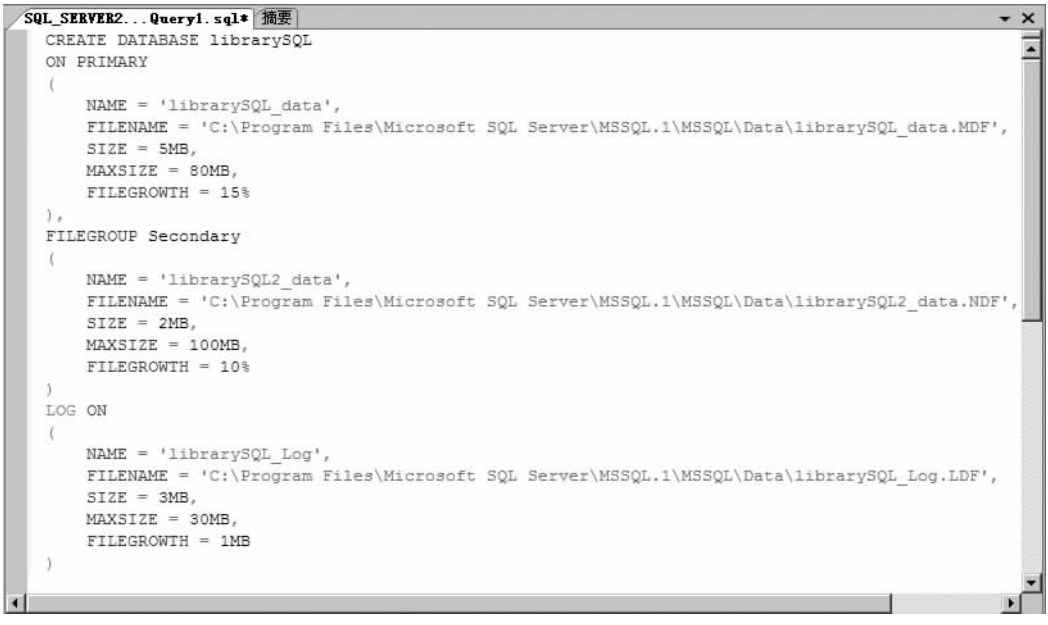

图 3-9　查询编辑器

```
CREATE DATABASE librarySQL
ON PRIMARY
(
    NAME=′librarySQL_data′,
    FILENAME=′C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\
librarySQL_data.MDF′,
    SIZE=5MB,
    MAXSIZE=80MB,
    FILEGROWTH=15%
),
FILEGROUP Secondary
```

```
(
    NAME='librarySQL2_data',
    FILENAME='C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\librarySQL2_data.NDF',
    SIZE=2MB,
    MAXSIZE=100MB,
    FILEGROWTH=10%
)
LOG ON
(
    NAME='librarySQL_Log',
    FILENAME='C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\librarySQL_Log.LDF',
    SIZE=3MB,
    MAXSIZE=30MB,
    FILEGROWTH=1MB
)
```

(3)单击工具栏上的“执行”按钮 执行(X) 或者按 F5 快捷键,执行上述语句。

(4)在查询编辑器的“结果”窗格中将显示相关信息,告知用户数据库创建是否成功。

(5)命令成功执行后,在 SQL Server Management Studio 界面的“对象资源管理器”中,单击“刷新”按钮即可看到新建的数据库。

3.3 管理数据库

管理数据库除了可改变数据的属性外,实际应用中可能还需要更改或删除整个数据、导入和导出数据、分离和附加数据库等。在 SQL Server 2005 提供的 SQL Server Management Studio 中,可以轻松地执行这些操作。当然,某些操作也可以通过 Transact-SQL 语句来完成。

3.3.1 数据库属性设置

1. 使用 SQL Server Management Studio 修改数据库属性

SQL Server 2005 中,通过 SQL Server Management Studio 修改数据库属性的一般操作步骤为:

(1)启动 SQL Server Management Studio,登录到指定的服务器。

(2)在“对象资源管理器”窗口中选中要修改属性的数据库,右击,在弹出的快捷菜单中选择“属性”命令。

(3)弹出“数据库属性”窗口,如图 3-10 所示,在“选择页”中,单击“文件”选项。此时,在右侧窗口中可以修改数据库的所有者、设置是否使用全文索引、修改数据库文件属性等,操作方法与建立数据库时相同。

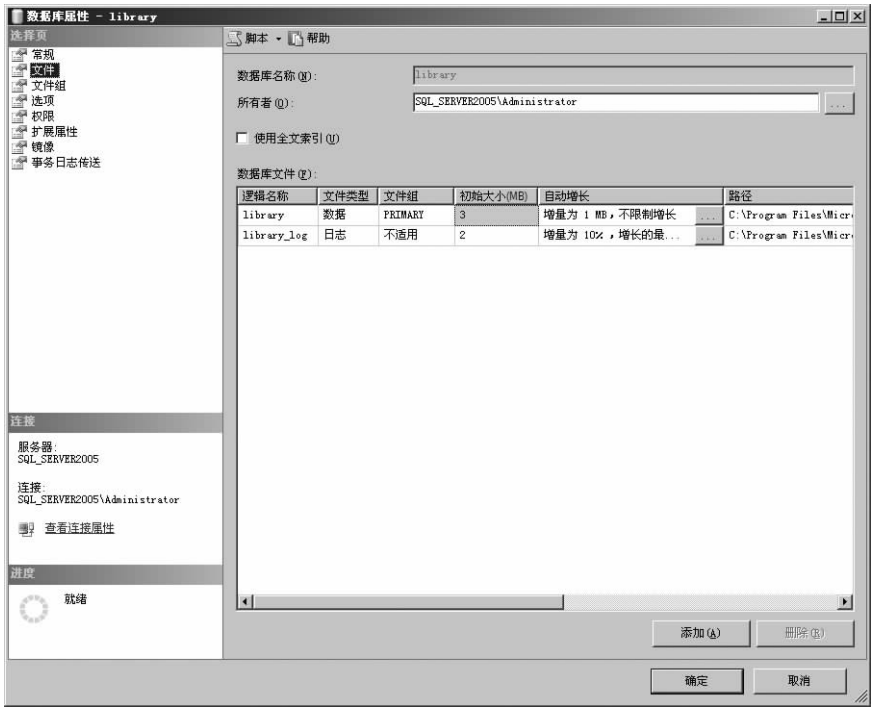

图 3-10　“数据库属性”窗口

(4)单击“选择页”中的“文件组”选项,可以添加或删除文件组。

(5)单击“选择页”中的“选项”选项,可以修改排序规则、恢复模式和兼容级别等。

(6)在其他选择页中可以查看或修改数据库的其他各项属性。

(7)在如图 3-10 所示的窗口左侧的“连接”中,单击“查看连接属性”超链接,可以查看数据库连接属性,如图 3-11 所示。

(8)单击“数据库属性”窗口右下角的“确定”按钮,完成对数据库的修改。

2. 使用 ALTER DATABASE 命令修改数据库属性

ALTER DATABASE 命令可以增加或删除数据库中的文件,修改数据库的属性设置等。但是,只有数据库管理员或者被授权使用 ALTER DATABASE 语句的数据库所有者才有权执行该命令。该命令的常用语法格式为:

```
ALTER DATABASE database_name
{
    <add_or_modify_files>
    |<add_or_modify_filegroups>
    |MODIFY NAME=new_database_name
    |COLLATE collation_name
}
<add_or_modify_files>::=
{
    ADD FILE<filespec>[,…n]
```

图 3-11 查看数据库连接属性

```
    [TO FILEGROUP{filegroup_name|DEFAULT}]
    |ADD LOG FILE<filespec>[,…n]
    |REMOVE FILE logical_file_name
    |MODIFY FILE<filespec>
}
<add_or_modify_filegroups>::=
{
    |ADD FILEGROUP filegroup_name
    |REMOVE FILEGROUP filegroup_name
    |MODIFY FILEGROUP filegroup_name
        {
            <filegroup_updatability_option>
            |DEFAULT
            |NAME=new_filegroup_name
        }
}
```

该命令中有关修改文件的主要参数说明如下：

- ADD FILE：向数据库添加数据文件。关键字 TO FILEGROUP 指定要将该文件添加到的文件组，默认为主文件组。
- ADD LOG FILE：向数据库添加事务日志文件。

- REMOVE FILE：从数据库中删除数据文件。当删除一个数据文件时，逻辑文件与物理文件同时被删除。
- MODIFY FILE：修改数据文件的属性，可以修改的文件属性包括 FILENAME、SIZE、MAXSIZE 和 FILEGROWTH。但需要注意的是，一次只能修改一个属性。

说明：有关 ALTER DATABASE 命令的更详细说明请参阅联机帮助文档。

【例 3-2】 修改数据库 librarySQL，向其中添加一个包含文件 librarySQL3_data(文件大小为 2 MB，按 10%增长，最大为 100 MB)的文件组 First，并将此文件组设置为默认文件组。

操作步骤如下：

(1)选中 librarySQL 数据库，右击，在弹出的快捷菜单中选择“新建查询”命令，打开查询编辑器窗口。

(2)在查询编辑器中输入如下语句：

```
ALTER DATABASE librarySQL
ADD FILEGROUP First
ALTER DATABASE librarySQL
ADD FILE
(
    NAME='librarySQL3_data',
    FILENAME='C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Data\librarySQL3_data.NDF',
    SIZE=2MB,
    MAXSIZE=100MB,
    FILEGROWTH=10%
)TO FILEGROUP First
ALTER DATABASE librarySQL
MODIFY FILEGROUP First DEFAULT
```

(3)单击工具栏上的“执行”按钮或者按 F5 快捷键，执行上述语句。在“结果”窗格中将显示相关信息，告知用户命令执行结果。

3.3.2　重命名数据库

在 SQL Server 2005 中，可以更改数据库的名称。在重命名数据库之前，应该确保没有人使用该数据库，而且该数据库设置为单用户模式。通过 SQL Server Management Studio 重命名数据库的一般操作步骤为：

(1)启动 SQL Server Management Studio，登录到指定的服务器。

(2)在“对象资源管理器”窗口中选择要重命名的数据库，右击，在弹出的快捷菜单中选择“重命名”命令，如图 3-12 所示，或者双击要重命名的数据库。

(3)数据库名处于可编辑状态，直接输入新的数据库名，按 Enter 键即可。

图 3-12　重命名数据库

提示:通过 ALTER DATABASE 命令重命名数据库的语句为:

```
ALTER DATABASE librarySQL
MODIFY NAME=newlibrary
```

3.3.3 删除数据库

当不再需要数据库或者数据库被转移到其他服务器上时,可以删除该数据库。数据库删除后,文件及其数据均从服务器上删除。一旦删除数据库即是永久删除。

建议在数据库删除之后备份 master 数据库。这是因为删除数据库将更新 master 数据库中的信息。如果必须还原 master,自上次备份 master 之后删除的所有数据库仍将引用这些不存在的数据库,从而导致产生错误消息。

【例 3-3】 删除【例 3-1】中创建的数据库 librarySQL。

操作步骤如下:

(1)启动 SQL Server Management Studio,登录到指定的服务器。

(2)选中数据库 librarySQL,右击,从弹出的快捷菜单中选择"删除"命令。

(3)弹出"删除对象"窗口,如图 3-13 所示,在该对话框中对删除操作进行设置,包括"删除数据备份和还原历史记录信息"以及"关闭现有连接"等。设置完成后,单击"确定" 按钮,系统显示成功后,librarySQL 数据库便从"数据库"中被删除了。

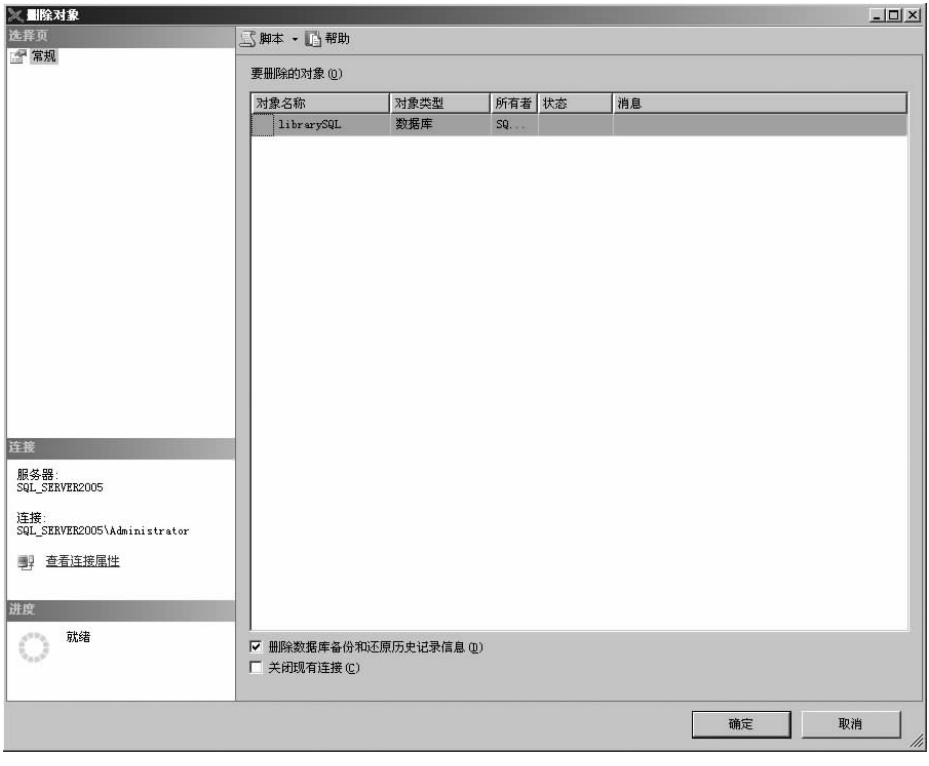

图 3-13 "删除对象"窗口

提示:通过 DROP DATABASE 命令实现【例 3-3】的命令为:

```
DROP DATABASE librarySQL
```

注意：当数据库正在被使用、恢复时，该数据库不能被删除。

3.3.4　导入和导出数据

使用 SQL Server Management Studio 的导入和导出向导，可以轻松地实现在数据源之间复制数据的操作。SQL Server 的导入和导出功能支持多种数据源类型，包括 Access 数据源、Oracle 数据源、Excel 文件乃至文本文件等。

1. 数据的导入

【例 3-4】 通过文本文件数据库导入学生成绩至数据库 student。

操作前提：

假设在现有的数据库服务器上已经创建好了数据库 student，并在其中创建表 ScoreInfo。ScoreInfo 表用于存放学生的考试成绩信息，其结构如表 3-2 所示。

表 3-2　ScoreInfo 表

字　　段	类　　型	描　　述
Semester	文本	学期
StudentNumber	文本	学生学号
Course	文本	课程
Score	数字	成绩

说明：关于在数据库中创建表的操作参照本书第 4 章的介绍。

操作步骤如下：

(1)启动 SQL Server Management Studio，登录到指定的服务器。

(2)选中数据库，右击，从弹出的快捷菜单中选择“任务”→“导入数据”命令。

(3)弹出“SQL Server 导入和导出向导”窗口，如图 3-14 所示。

图 3-14　“SQL Server 导入和导出向导”窗口

(4)单击“下一步”按钮,显示“选择数据源”窗口,默认显示“常规”选项,如图 3-15 所示。

图 3-15 选择数据源——“常规”选项

- 从“数据源”下拉列表框中选择“平面文件源”选项,即文本文件。
- 单击“文件名”右侧的“浏览”按钮,选择要导入数据的文本文件。
- 根据文本文件数据库的设置,在“格式”下拉列表框中选择“带分隔符”或者“固定宽度”等选项,也可进一步设置“文本限定符”、“标题行分隔符”、“要跳过的标题行数”等项。
- 单击窗口左侧的“高级”、“预览”选项可进行更详细的设置。
- 单击窗口左侧的“列”选项可以看到文本文件数据库中将要导入的数据,如图 3-16 所示。

图 3-16 选择数据源——“列”选项

(5)单击“下一步”按钮,显示“选择目标”窗口,如图 3-17 所示。在“目标”下拉列表框中选择 Microsoft OLE DB Provider for SQL Server,在“服务器名称”下拉列表框中选定服务器,选择身份验证类型,在“数据库”下拉列表框中选定数据库。

图 3-17　“选择目标”窗口

(6)单击“下一步”按钮,显示“选择源表和源视图”窗口,如图 3-18 所示。选择“源”和“目标”项,单击右侧的“编辑”按钮,弹出“列映射”窗口,如图 3-19 所示。

图 3-18　“选择源表和源视图”窗口

(7)在“列映射”窗口中设置需要映射到的数据库字段,以及数据库导入的方式(“删除”或“追加”)。设置完成后,单击“确定”按钮,返回如图 3-18 所示的窗口。

(8)单击“下一步”按钮,弹出“保存并执行包”窗口,如图 3-20 所示。在该窗口中设置

“立即执行”和“保存 SSIS 包”项。如果选中“保存 SSIS 包”单选按钮，则还需要设定是否保护敏感数据及保护级别等信息。

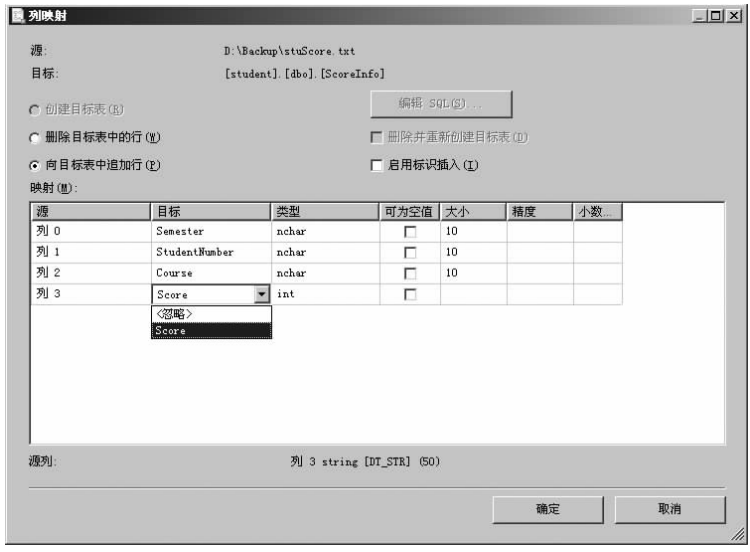

图 3-19 “列映射”窗口(1)

图 3-20 “保存并执行包”窗口

(9)单击“下一步”按钮，弹出“完成该向导”窗口，如图 3-21 所示。该窗口中显示前面各步骤中设置的结果，确认无误后，单击“完成”按钮。

(10)如果在步骤(8)中选中了“立即执行”复选框，则向导结束后可以看到导入数据的执行结果，如图 3-22 所示；如果某一步骤失败，可单击右侧“消息”列中的超链接，查看错误信息。

图 3-21　“完成该向导”窗口

图 3-22　数据导入执行过程

2. 数据的导出

【例 3-5】　导出数据库 student 中的数据至文本文件数据库。

操作步骤如下：

(1)启动 SQL Server Management Studio，登录到指定的服务器。

(2)选中数据库，右击，从弹出的快捷菜单中选择“任务”→“导出数据”命令，弹出“SQL Server 导入和导出向导”窗口，见图 3-14。

(3)单击“下一步”按钮，显示“选择数据源”窗口，如图 3-23 所示，选择“数据源”类型为 Microsoft OLE DB Provider for SQL Server；选择身份验证类型和要导出的数据库 student。

(4)单击“下一步”按钮，显示“选择目标”窗口，如图 3-24 所示。选择“目标”类型为“平面文件目标”，即文本文件；单击“文件名”右侧的“浏览”按钮，选择数据导出的文件。

图 3-23 “选择数据源”窗口

图 3-24 “选择目标”窗口

(5)单击“下一步”按钮,显示“指定表复制或查询”窗口,如图 3-25 所示。选中“复制一个或多个表或视图的数据”单选按钮。

(6)单击“下一步”按钮,显示“配置平面文件目标”窗口,如图 3-26 所示。在“源表或源视图”下拉列表框中选定数据库中的某个表;设置导出后文本文件中的“行分隔符”和“列分隔符”;单击“编辑转换”按钮,弹出“列映射”窗口,通过该窗口设定源字段到目标字段的映射,如图 3-27 所示;单击“预览”按钮,可查看选中源表或视图中的数据,设置完成后,单击“下一步”按钮。

图 3-25　“指定表复制或查询”窗口

图 3-26　“配置平面文件目标”窗口

(7)显示“保存并执行包”窗口(见图 3-20);选中“立即执行”复选框,单击“下一步”按钮,显示“完成该向导”窗口(见图 3-21)。单击“完成”按钮。

(8)如果步骤(7)中选择“立即执行”,则可以看到数据导出的执行过程和结果,如图 3-28 所示。导出成功后,右侧“消息”列中将显示导出了多少行数据;如果某一步失败,可单击“消息”列中相应的超链接,查看错误信息。

提示:其他类型数据源间数据的导入和导出操作与文本文件数据库到 SQL Server 数据库的导入和导出操作相似,读者可自行实践掌握。

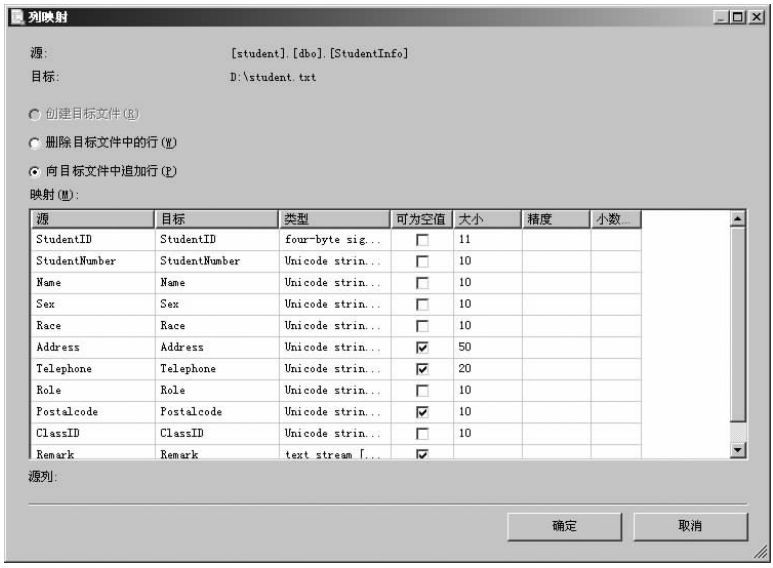

图 3-27 “列映射”窗口(2)

图 3-28 数据导出执行成功

3.3.5 生成脚本

当需要为通过 SQL Server Management Studio 创建的表或视图等数据库对象创建 Transact-SQL 语句时,可以通过使用 SQL Server Management Studio 生成脚本向导实现。这样,如果需要在其他服务器上创建相同结构的数据库时,只需执行脚本文件即可轻松创建各种数据库对象。

【例 3-6】　对数据库 student 中的表创建脚本文件。

操作步骤如下：

(1)启动 SQL Server Management Studio，登录到指定的服务器。

(2)选中数据库，右击，从弹出的快捷菜单中选择“任务”→“生成脚本”命令，弹出“脚本向导”窗口，如图 3-29 所示。

图 3-29　“脚本向导”窗口

(3)单击“下一步”按钮，显示“选择数据库”窗口，如图 3-30 所示，选择需要生成脚本的数据库。如果选中“为所选数据库中的所有对象编写脚本”复选框，可直接单击“完成”按钮。

图 3-30　“选择数据库”窗口

(4)单击“下一步”按钮,显示“选择脚本选项”窗口,如图 3-31 所示。该窗口列出各个选项供用户设置,选中某项在窗口下端会出现关于该选项的说明。没有特殊需求通常均采用默认值。

图 3-31 “选择脚本选项”窗口

(5)单击“下一步”按钮,显示“选择对象类型”窗口,如图 3-32 所示。该窗口的列表框中会列出所选数据库中包括的各种对象。选中列表项前端的复选框,这里选中“表”复选框。

图 3-32 “选择对象类型”窗口

(6)单击“下一步”按钮,显示“选择表”窗口,如图 3-33 所示,在“选择表”列表框中选中需要生成脚本的表名,可以一次选择多个表。

图 3-33　“选择表”窗口

(7)单击“下一步”按钮,显示“输出选项”窗口,如图 3-34 所示,在“脚本模式”中选择脚本保存的类型:“将脚本保存到文件”、“将脚本保存到剪贴板”、“将脚本保存到‘新建查询’窗口”。如果选择“将脚本保存到文件”,单击“浏览”按钮,选择保存后的文件位置和名称。

图 3-34　“输出选项”窗口

(8)单击“下一步”按钮,显示“脚本向导摘要”窗口,如图 3-35 所示,查看生成脚本的各项设置,确认无误后,单击“完成”按钮。

(9)显示“生成脚本进度”窗口,如图 3-36 所示。如果生成脚本成功,则在步骤(7)中设置的文件 script.sql 中写入 SQL 语句;如果某一部分出错,可单击“消息”列中相关错误信息查看。

生成脚本操作执行成功后,可找到脚本文件(.sql 文件)查看生成的 SQL 命令。

图 3-35 “脚本向导摘要”窗口

图 3-36 “生成脚本进度”窗口

3.3.6 分离和附加数据库

SQL Server 2005 中用户数据库可以从服务器上分离出来，然后再方便地附加到其他服务器上，从而实现在不同服务器上移植数据库。

1. 分离数据库

分离数据库的实现方法有两种：一是通过 SQL Server Management Studio 来实现，二是通过系统存储过程完成。这里介绍第一种方法。需要注意的是，系统数据库 master、tempdb和 model 不能执行分离操作。

在 SQL Server Management Studio 中分离数据库的一般操作步骤是：

(1)启动 SQL Server Management Studio，登录到指定的服务器。

(2)选中要分离的数据库,这里选择 library,右击,从弹出的快捷菜单中选择“任务”→“分离”命令,弹出“分离数据库”窗口,如图 3-37 所示。在“要分离的数据库”区中选中需要分离的数据库,并进行相关设置。

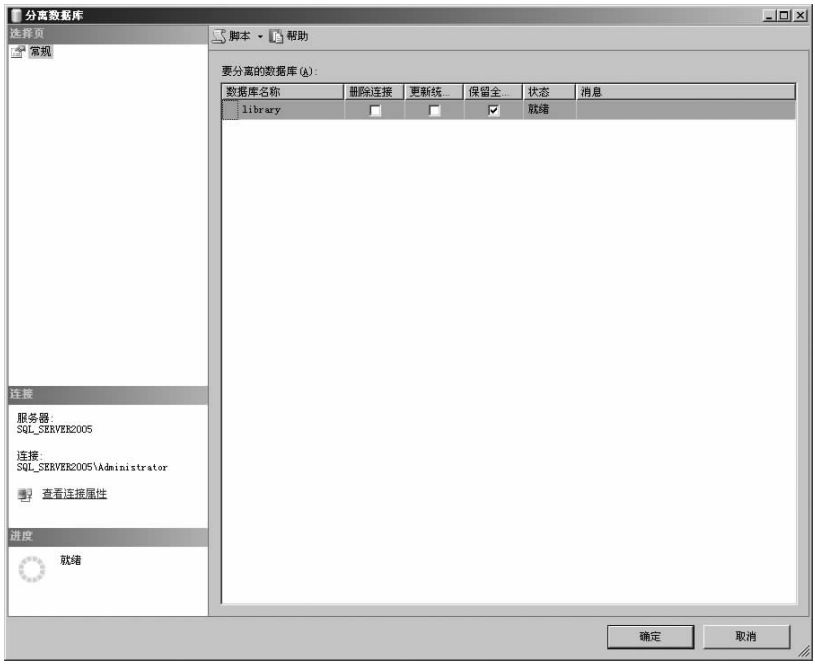

图 3-37　“分离数据库”窗口

- “删除连接”选项表示是否断开数据库连接。
- “更新统计信息”选项表示在分离数据库之前是否更新优化统计信息。
- “保留全文目录”选项表示是否保留与数据相关联的所有全文目录。
- “状态”选项显示数据库的状态是“就绪”或者“未就绪”。
- “消息”选项显示在数据库状态为“未就绪”时数据库的连接信息。

(3)设置完成后,单击“确定”按钮,则系统执行数据库分离任务。数据库分离成功后,分离的数据同时从“对象资源管理器”窗口的“数据库”下消失。

提示:分离数据库将从 SQL Server 中删除该数据库,但是组成该数据库的数据和事务日志文件中的数据完好无损。这些数据和事务日志文件用来将分离的数据库附加到其他服务器上。

2. 附加数据库

同分离数据库相似,附加数据库也可以通过两种方法实现:一是通过 SQL Server Management Studio 来实现;二是通过系统存储过程完成。在此,仅介绍第一种实现方法。

在附加数据库之前,一定要把需要附加的数据库所包含的全部数据文件和事务日志文件复制到要执行附加操作的服务器上。

在 SQL Server Management Studio 中附加数据库的一般操作步骤是:

(1)启动 SQL Server Management Studio,登录到指定的服务器。

(2)在“对象资源管理器”窗口中选定“数据库”,右击,从弹出的快捷菜单中选择“附加”命令,弹出“附加数据库”窗口,如图 3-38 所示。单击“要附加的数据库”下的“添加”按钮,弹

出“定位数据库文件”窗口，如图 3-39 所示。

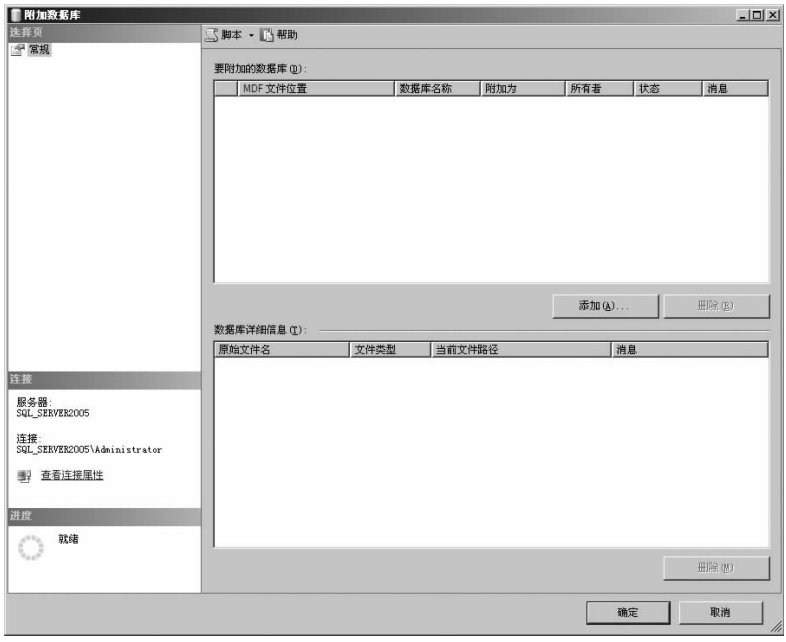

图 3-38 “附加数据库”窗口(1)

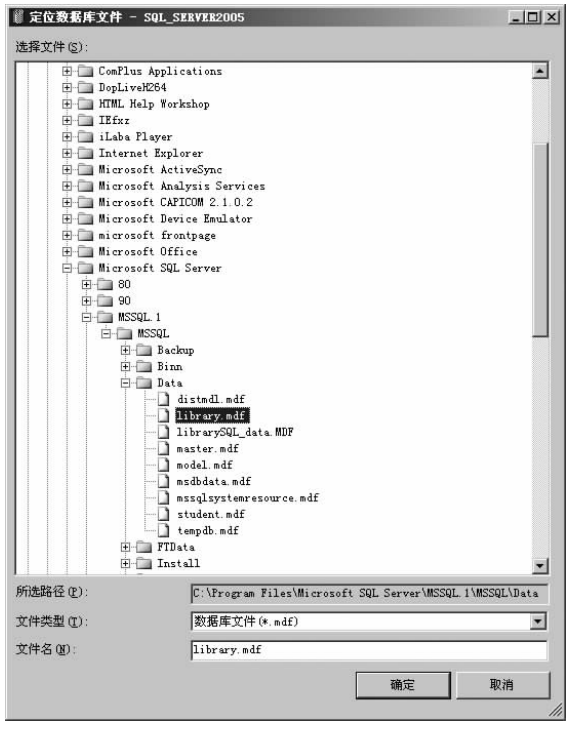

图 3-39 “定位数据库文件”窗口

(3)在“定位数据库文件”窗口中选中要附加的数据库文件，一般是. mdf 文件，如 library. mdf，单击“确定”按钮。

(4)返回“附加数据库”窗口,此时,在“要附加的数据库”区中显示附加的主数据库文件、数据库名称、所有者等信息;“数据库详细信息”中显示附加数据库包含的所有数据文件和事务日志文件,如图 3-40 所示。单击“确定”按钮,数据库 library 即被附加到 SQL Server 服务器上。

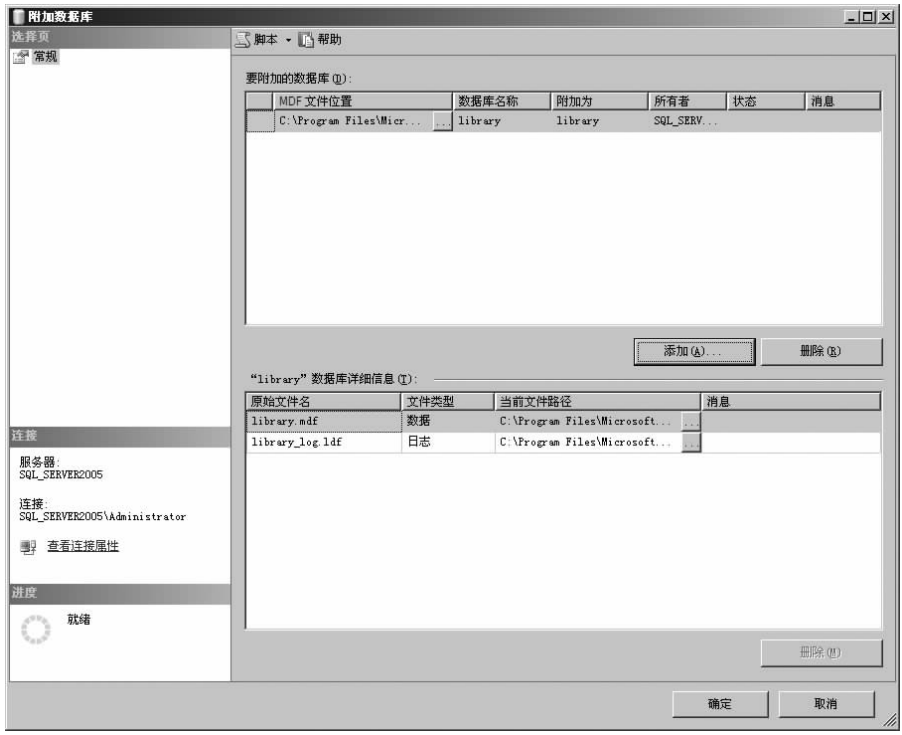

图 3-40　“附加数据库”窗口(2)

3.3.7　收缩日志文件

数据库中的事务日志映射在一个或多个物理文件上。从概念上讲,日志文件是一系列日志记录。从物理上讲,日志记录序列被有效地存储在实现事务日志的物理文件集中。SQL Server 数据库引擎在内部将每一个物理日志文件分成多个虚拟日志文件。虚拟日志文件没有固定大小,且物理日志文件所包含的虚拟日志文件数不固定。在创建或扩展日志文件时,数据库引擎动态选择虚拟日志文件的大小。虚拟日志文件的大小是现有日志大小和新文件增量大小之和。如果这些日志文件由于许多微小增量而增长到很大,则它们将具有很多虚拟日志文件。这会降低数据库启动以及日志备份和还原操作的速度。因此,有必要对事务日志文件进行收缩,提高事务日志文件备份和还原的速度。

在 SQL Server Management Studio 中收缩日志文件的一般操作步骤是:

(1)启动 SQL Server Management Studio,登录到指定的服务器。

(2)展开“对象资源管理器”窗口中的“数据库”,选中需要收缩的数据库,如 library,右击,从弹出的快捷菜单中选择“任务”→“收缩”→“文件”命令,弹出“收缩文件”窗口,如图 3-41 所示。在“文件类型”下拉列表框中选择“日志”,窗口中将显示该数据库中包含的日志文件名、位置、当前分配空间以及可用空间等。在“收缩操作”中选中“释放未使用的空

间”，或者“通过将数据迁移到同一文件组中的其他文件来清空文件”，或者“在释放未使用的空间前重新组织页”和相应的收缩后文件大小。设置完成后，单击“确定”按钮，完成对日志文件的收缩。

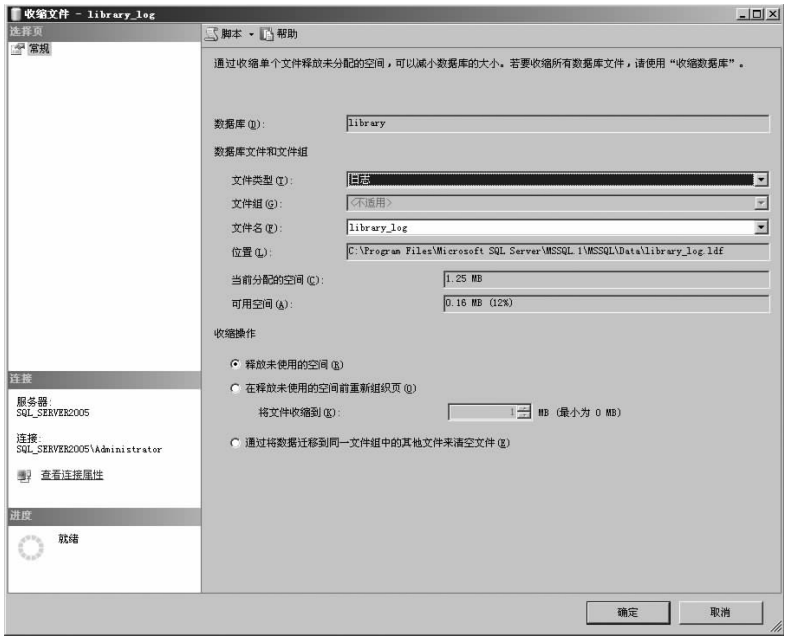

图 3-41 “收缩文件”窗口

3.4 优化数据库

数据库引擎优化顾问是 SQL Server 2005 中的新工具，使用该工具可以优化数据库，提高查询处理的性能。数据库引擎优化顾问检查指定数据库中处理查询的方式，然后建议如何通过修改物理设计结构(如索引、索引视图和分区)来改善查询处理性能。

作为 SQL Server 2005 中的新增项，数据库引擎优化顾问提供了一种基于图形用户界面 (GUI) 的方式来查看优化会话和优化建议报表。

(1)执行“工具”→“数据库引擎顾问”命令，弹出“连接到服务器”窗口。

(2)单击“连接”按钮，弹出如图 3-42 所示的 Database Engine Tuning Advisor 窗口。

(3)图 3-42 左侧的“会话监视器”中列出了已对此 SQL Server 实例执行的所有优化会话，打开数据库引擎优化顾问时，在窗格顶部将显示一个新会话；右侧窗格包含“常规”和“优化选项”选项卡。

- 在“常规”选项卡中，输入优化会话的名称，指定要使用的工作负荷文件或表，并选择要在该会话中优化的数据库和表。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。优化数据库时，数据库引擎优化顾问使用跟踪文件、跟踪表、Transact-SQL 脚本或 XML 文件作为工作负荷输入，如图 3-43 所示。
- 在“优化选项”选项卡上，可以选择希望数据库引擎优化顾问在分析过程中考虑的物理数据库设计结构(索引或索引视图)和分区策略。在此选项卡上，还可以指定数据

库引擎优化顾问优化工作负荷使用的最大时间，如图 3-44 所示。默认情况下，数据库引擎优化顾问优化工作负荷的时间为 1 小时。

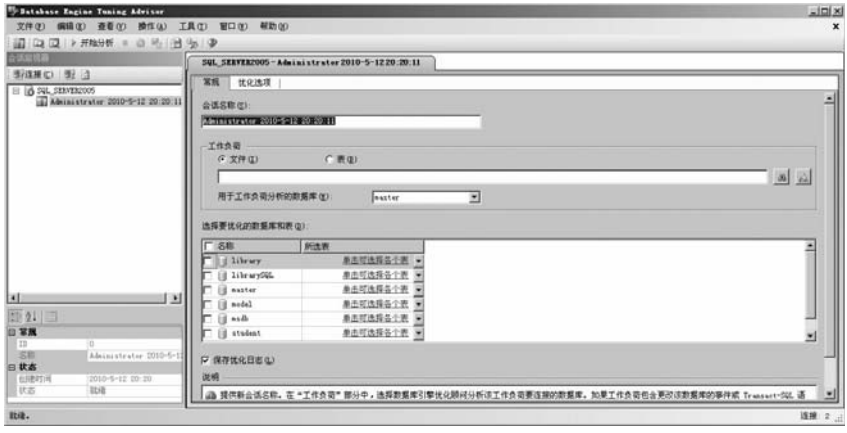

图 3-42　Database Engine Tuning Advisor 窗口

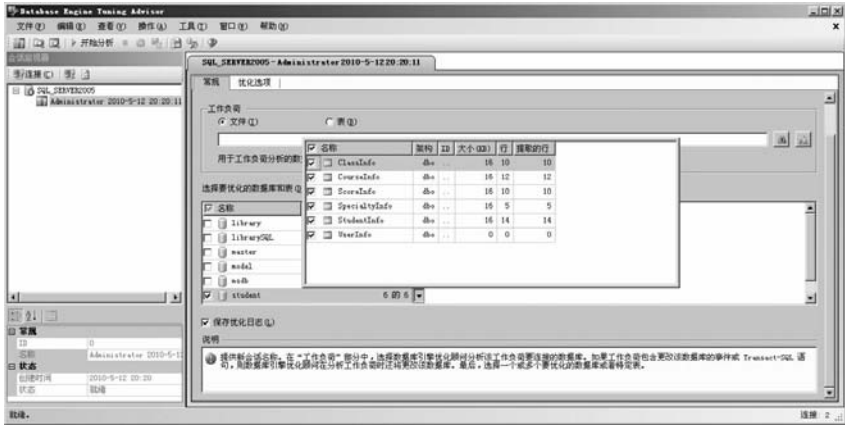

图 3-43　“常规”选项卡

图 3-44　“优化选项”选项卡

(4)在图 3-44 中，单击工具栏上的“开始分析”按钮，等待优化分析结束。

(5)优化分析结束后,在“会话监视器”窗口中双击要查看的会话。数据库引擎优化顾问将加载上一个优化会话中的会话信息,并显示“建议”选项卡;或者单击“报告”选项卡,在“优化摘要”窗格中查看有关此优化会话的信息。

实　训

【实训目的】

(1)熟悉两种创建数据库的方法。

(2)熟悉常用的数据库操作。

【实训内容】

(1)分别使用 SQL Server Management Studio 和 Transact-SQL 语句创建学生基本信息数据库 student。该数据库的主数据文件大小为 5 MB,最大为 30 MB,按 2 MB 增长;事务日志文件大小为 2 MB,最大为 10 MB,按 10%增长;两个文件组 PRIMARY 和Secondary,文件组 PRIMARY 包括主数据库文件和事务日志文件;文件组 Secondary 中仅包括一个数据文件:大小为 3 MB,最大 50 MB,按 15%增长。其他参数可自定。

(2)对题目(1)中创建的数据库执行分离和附加操作。

【实训总结】

结合自己操作的实际情况,认真撰写实训总结。

本章小结

本章主要内容包括 SQL Server 2005 数据库概述、数据库的创建、管理和优化等,要求掌握两种方法创建和管理数据库;SQL Server 2005 数据库的文件组成;在 SQL Server Management Studio 中分离和附加数据库的操作方法;SQL Server Management Studio 中执行数据的导入和导出操作;SQL Server 2005 数据库中生成脚本、收缩日志文件、优化等的操作过程。数据库是 SQL Server 2005 系统的重要对象,是存放数据库对象和数据的地方。掌握好数据库的创建、管理方法是在数据库中创建表、视图、存储对象等数据库对象的前提。

习　题　3

1. SQL Server 2005 的数据库对象有哪些?

2. SQL Server 2005 数据库由哪些文件组成?

3. 简述 SQL Server 2005 中各系统数据库的作用。

4. 导入和导出数据的作用是什么? 简述其操作步骤。

5. 分离和附加数据的作用是什么? 简述其操作步骤。

6. 总结自己在数据库创建和管理操作过程中遇到的错误,并分析其原因。

第 4 章　创建和管理 SQL Server 2005 数据表

创建好数据库后，就应该建立存储数据的表。表是 SQL Server 2005 中一种重要的数据库对象，用于存储数据库中的所有数据。就表而言，通常要定义表的结构、实现数据库完整性约束和建立索引。在完成以上任务以后，就可以向表中添加数据，实现对企业信息的电子化管理。

4.1　表　概　述

表是客观事物实体及实体间关系的具体表现形式，包含了数据库中的所有数据。表是一个二维数据结构，由行和列组成。如表 4-1 所示，每一行为一个记录，表示一个具体实体，如李慧、曾刚意等，分别代表一个学生；每一列为一个字段，表示实体的一个具体属性，如 StudentID(学号)、Sex(性别)等。在一张表中，字段不能够重名，字段值具有相同的数据类型。

表 4-1　student

StudentID	Name	Sex	Birthday	BirthPlace	ID	Dept	RegistrationDate
091007301	李慧	False	1991/4/1 0:00:00	洛阳	410303199104013216	D10	2009/9/1 0:00:00
091007302	曾刚意	True	1990/5/2 0:00:00	郑州	410104199005023581	D10	2009/9/1 0:00:00
091007303	李涛	True	1991/1/1 0:00:00	郑州	410104199101017611	D10	2009/9/2 0:00:00
091007304	陈超	True	1991/3/2 0:00:00	洛阳	410304199103021233	D10	2009/9/1 0:00:00
091007305	张秀娟	False	1990/3/5 0:00:00	洛阳	410301199003054694	D10	2009/9/2 0:00:00

SQL Server 2005 为定义字段数据类型提供了很多系统数据类型，除此以外，用户还可以根据需要自己定义新的数据类型。SQL Server 2005 中具有的系统数据类型如下：

bigint	binary	bit	char	cursor
datetime	decimal	float	image	int
money	nchar	ntext	numeric	nvarchar
real	smalldatetime	smallint	smallmoney	sql_variant
table	text	timestamp	tinyint	varbinary
varchar	uniqueidentifier	xml		

4.2 数据表的创建

在 SQL Server 2005 中，提供了两种方法来创建数据表：第一种方法是利用 Microsoft SQL Server Management Studio，另一种方法是利用 Transact-SQL 语句。本节将采用以上两种方法介绍数据表的创建。

4.2.1 使用 SQL Server Management Studio 创建表

1. 创建新表

创建新表就是要定义表的结构，为每一列定义字段名，并指定其数据类型，然后保存该表。

【例 4-1】 使用 SQL Server Management Studio，在数据库 library 中创建数据表 student 表。

student 表结构如表 4-2 所示。

表 4-2 student 表

列 名	数据类型	允 许 空
StudentID	char(9)	否
Name	varchar(10)	否
Sex	bit	否
Birthday	smalldatetime	否
BirthPlace	varchar(10)	否
ID	char(18)	否
Dept	char(3)	否
RegistrationDate	smalldatetime	否

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”，右击，如图 4-1 所示，在弹出的快捷菜单中选择“新建表”命令。

(3)此时打开表设计器窗口，如图 4-2 所示。在“列名”栏中输入表的字段名，在“数据类型”栏中选择字段的数据类型，在“允许空”栏中设置该字段的值是否允许为空(NULL)。

在图 4-2 中，每一个字段都对应一个“列属性”标签，其中各选项的含义如下：

- 名称：指定字段的名称，最大为 128 个字符，命名规则遵守标识符命名规则。
- 长度：指定字段数据类型的长度。
- 默认值或绑定：指定字段的默认值，在增加记录时，若没有在此字段上指定值，则用此默认值代替。
- 数据类型：指定字段的数据类型。

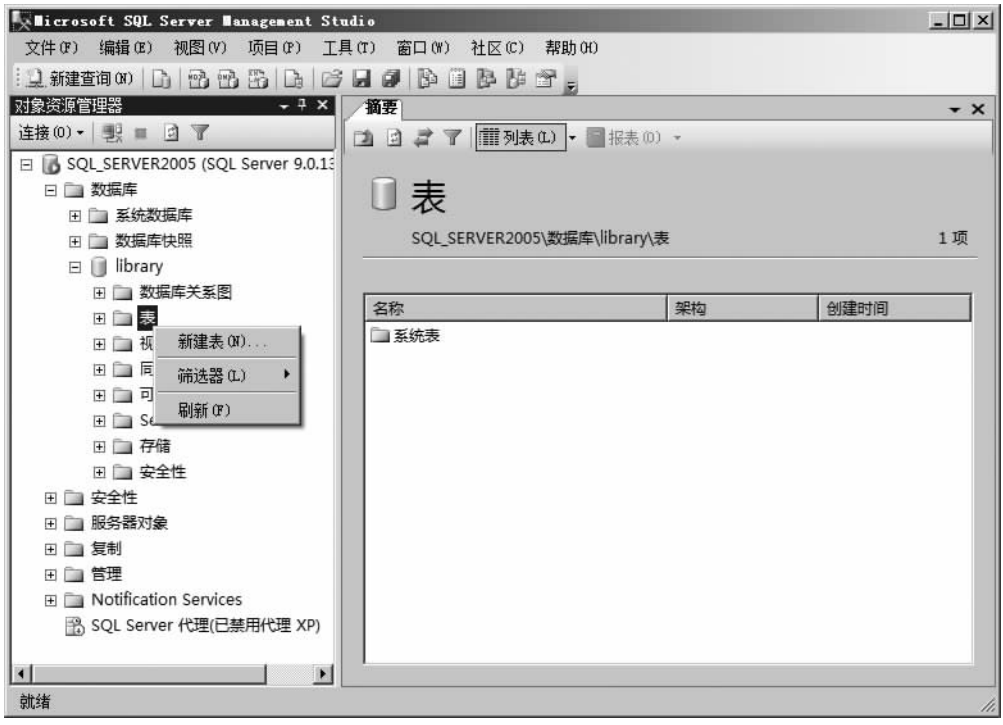

图 4-1　创建新表

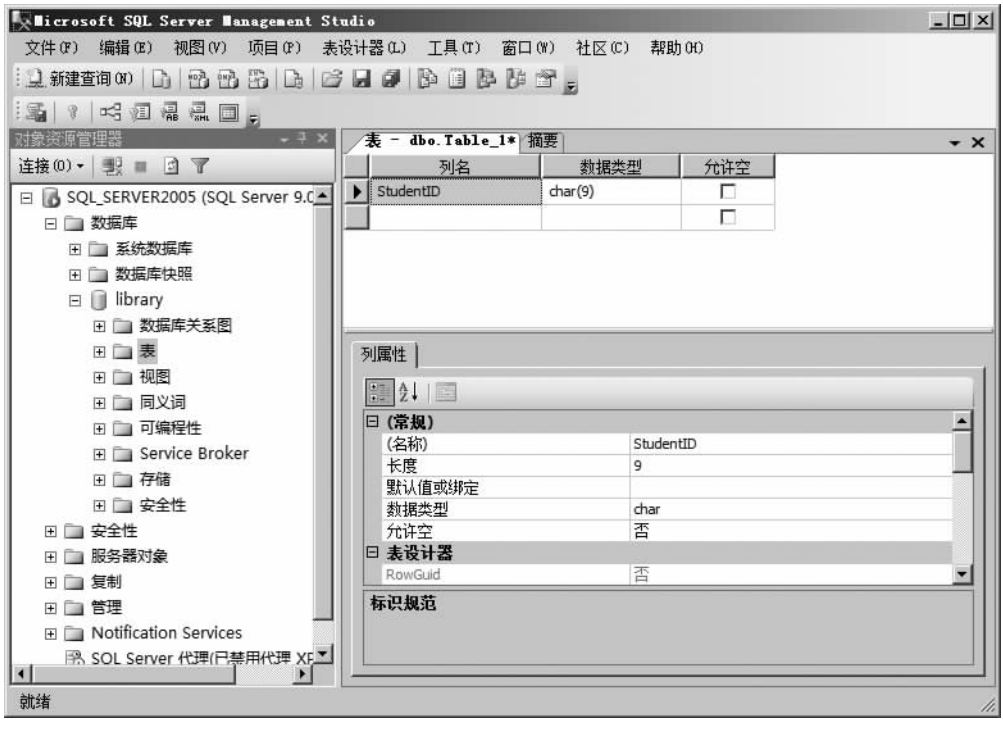

图 4-2　表设计器

• 允许空:指定在此字段上是否可以输入空值(NULL)。NULL 不同于空字符或 0,

NULL 只是表示此值暂时未知。

- RowGuid:当字段数据类型为 uniqueidentifier 时,可为字段增加 RowGuid 选项,使此字段产生一个全局唯一的值,即在全球联网的计算机中不产生重复的字段值。
- 排序规则:指定该字段的排序规则。
- 说明:指定字段的备注信息。

(4)单击工具栏上的保存按钮,出现如图 4-3 所示的对话框,输入表的名称 student,单击"确定"按钮。此时便建好了 student 表,但是表中还没有任何记录。

图 4-3 保存表

2. 为字段指定默认值

为表增加记录时,若没有为某字段指定值,则可以用默认值代替。默认值可以是 NULL、常量或系统函数。

【例 4-2】 使用 SQL Server Management Studio,在数据库 library 中创建 card 数据表,并为 StartDate(起效时间)、State(状态)、Maximum(最大允许借阅数)、BorrowedNum(已借数量)字段指定默认值。

card 表结构如表 4-3 所示。

表 4-3 card 表

列 名	数据类型	允 许 空	默 认 值
CardID	char(8)	否	
StartDate	smalldatetime	否	getdate()
ExpiredDate	smalldatetime	否	
State	char(4)	否	'正常'
Maximum	tinyint	否	(10)
BorrowedNum	tinyint	否	(0)
StudentID	char(9)	否	

操作步骤如下:

(1)启动 SQL Server Management Studio。

(2)在"对象资源管理器"中,依次展开 SQL_SERVER2005→"数据库"→library→"表",右击,在弹出的快捷菜单中选择"新建表"命令。

(3)此时打开表设计器窗口,输入表的字段名,选择字段的数据类型,在"允许空"栏设置该字段的值是否允许为空(NULL)。若为字段定义的有默认值,则在"默认值或绑定"后输入字段的默认值,如图 4-4 所示。

(4)单击工具栏上的保存按钮,输入表的名称 card,单击"确定"按钮。

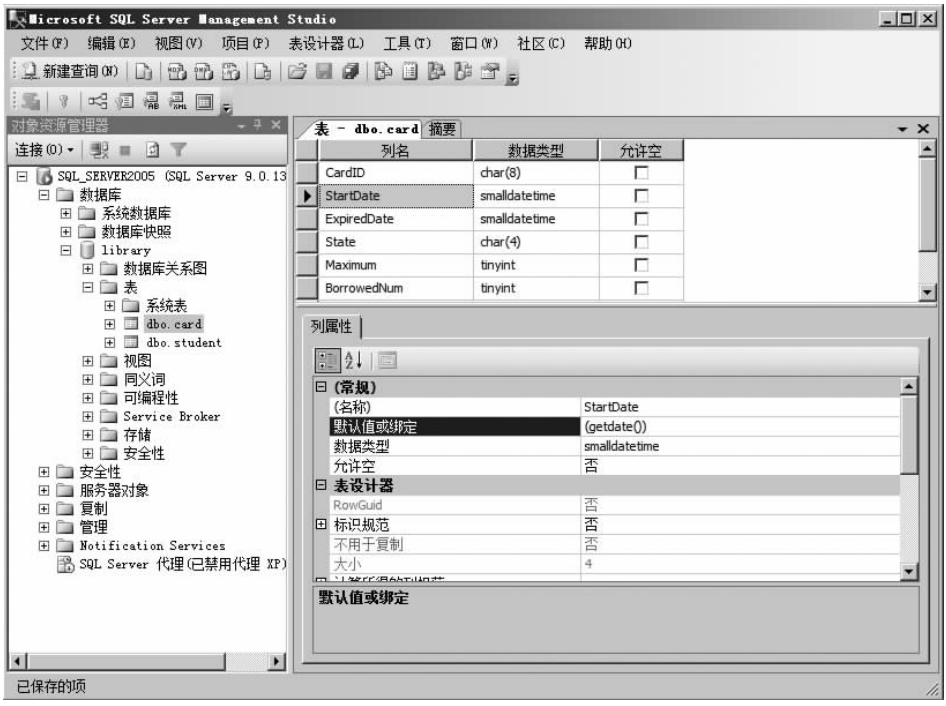

图 4-4　为字段指定默认值

3. 定义计算列

使用默认值选项为字段定义默认值时，不可以利用表中的字段。若表中某字段的值是由该表中其他字段的值通过计算得到的，则可以将该字段定义为计算列。在插入或更新记录时，不允许为计算列指定字段值。在没有为计算列指定“是持久的”选项时（即设置为“否”），计算列是未实际存储在表中的虚拟列，每当在查询中引用该列时，都将重新计算它们的值。若为计算列指定“是持久的”选项，则计算列实际存储在表中；若计算列所引用的字段值发生更新，则更新计算列的值。

【例 4-3】 使用 SQL Server Management Studio，在数据库 library 中创建 bookshelf 数据表，并将 ReturningTime 列定义为计算列。

bookshelf 表结构如表 4-4 所示。

表 4-4　bookshelf 表

列　　名	数据类型	允 许 空	计算所得的列规范
Barcode	char(8)	否	—
LendingTime	smalldatetime	是	—
LendingDays	smallint	是	—
ReturningTime		是	dateadd(dayofyear,[LendingDays],[LendingTime])
State	char(4)	否	—
ISBN	varchar(17)	否	—
CardID	char(8)	是	—
PlaceID	tinyint	否	—

提示：bookshelf 表记录了上架图书的借阅情况，因此，该书在没有借阅的情况下，LendingTime(借阅时间)、LendingDays(借阅天数)、ReturningTime(应归还时间)、CardID(借书证号)字段允许为空。若 LendingTime、LendingDays 字段有一个字段值为空，则 ReturningTime 字段的计算值也为空。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”，右击，在弹出的快捷菜单中选择“新建表”命令。

(3)此时打开表设计器窗口，输入表的字段名，选择字段的数据类型，在“允许空”栏设置该字段的值是否允许为空(NULL)。选中 ReturningTime 字段，在“计算所得的列规范”后输入计算表达式 dateadd(dayofyear,[LendingDays],[LendingTime])，如图 4-5 所示。

图 4-5 定义计算列

(4)单击工具栏上的保存按钮，输入表的名称 bookshelf，单击“确定”按钮。此时便建好了 bookshelf 表。

4.定义标识列

在定义表字段时，可以将某个字段定义为标识列。所谓标识列，是指字段的值由 SQL Server 2005 自动提供。定义标识列须同时指定种子和增量，或者两者都不指定；如果两者都未指定，则取默认值为(1,1)，即种子值(第一个 1)为 1，增量值(后一个 1)为 1。种子是添加表中的第一个记录所使用的值，增量是每次添加记录时，在上次增加记录时该字段值的基础上增加的幅度。例如，设种子、增量值为(3,2)，若上次添加记录时字段的值为 7，则下次添加记录时，字段的值为 9。如果为标识列指定了“不用于复制”选项，则复制代理执行插入记

录时，标识列中的值将不会增加。在一个表中，只能定义一个字段为标识列。

【例 4-4】 使用 SQL Server Management Studio，在数据库 library 中创建 place 数据表，并将 PlaceID 列定义为标识列，种子和增量分别为 1 和 1。

place 表结构如表 4-5 所示。

表 4-5　place 表

列　　名	数据类型	允 许 空	是 标 识	标识种子	标识增量
PlaceID	tinyint	否	是	1	1
Description	varchar(50)	否			

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”，右击，在弹出的快捷菜单中选择“新建表”命令。

(3)此时打开表设计器窗口，输入表的字段名，选择字段的数据类型，在“允许空”栏设置该字段的值是否允许为空(NULL)。选中 PlaceID 字段，在“是标识”后输入“是”，“标识种子”后输入 1，“标识增量”后输入 1，如图 4-6 所示。

图 4-6　定义标识列

(4)单击工具栏上的保存按钮，输入表的名称 place，单击“确定”按钮。此时便建好了 place 表。

4.2.2　使用 Transact-SQL 语句创建表

利用 Transact-SQL 语句创建表，需要利用 CREATE TABLE 语句，其语法格式为：

```
CREATE TABLE[database_name.[schema_name].|schema_name.]table_name
```

({<column_definition>|<computed_column_definition>}[<table_constraint>]
[,…n])
[ON{<partition_scheme>|filegroup|"default"}]
[TEXTIMAGE_ON{filegroup|"default"}]
[;]

该命令中各参数说明如下：

- database_name：指要在其中创建表的数据库名称。如果未指定，则 database_name 默认为当前数据库。
- schema_name：新表所属架构的名称。
- table_name：新建表的名称，最多可包含 128 个字符，表名必须符合标识符命名规则。
- ON：指定表存储的分区架构或文件组。如果指定了<partition_scheme>，则该表将成为已分区表，其分区存储在<partition_scheme>所指定的一个或多个文件组的集合中。如果指定了 filegroup，则该表将存储在命名的文件组中。如果指定了"default"，或者根本未指定 ON，则表存储在默认文件组中。CREATE TABLE 中指定表的存储机制后，不能再进行更改。
- TEXTIMAGE_ON：指定表中较大值列的存储机制，例如，text、ntext、image、xml、varchar(max)、nvarchar(max)、varbinary(max)和 CLR 用户定义类型的列。如果表中没有较大值的列，则不允许使用 TEXTIMAGE_ON。如果指定了<partition_scheme>，则不能指定 TEXTIMAGE_ON。如果指定了"default"，或者根本未指定 TEXTIMAGE_ON，则较大值的列存储在默认文件组中。CREATE TABLE 中指定较大值列的数据存储机制后，不能再进行更改。
- <table_constraint>：指定表约束。
- computed_column_definition：定义计算列的表达式，该列由同一表中的其他列通过表达式计算得到。表达式可以是非计算列的列名、常量、函数、变量，也可以是用一个或多个运算符连接的上述元素的任意组合。表达式不能是子查询。
- <column_definition>：字段定义，其语法如下：

  <column_definition>::=column_name<data_type>
  [COLLATE collation_name]
  [NULL|NOT NULL]
  [[CONSTRAINT constraint_name]DEFAULT constant_expression]
  |[IDENTITY[(seed,increment)][NOT FOR REPLICATION]]
  [ROWGUIDCOL]
  [<column_constraint>[…n]]

  其中参数说明如下：

  * column_name：表中字段的名称，最多包含 128 个字符，字段名必须遵循标识符命名规则，并在表中唯一。对于使用 timestamp 数据类型创建的列，可以省略 column_name。如果未指定 column_name，则 timestamp 列的名称将默认为 timestamp。
  * data_type：指定字段数据类型。
  * COLLATE：指定列的排序规则。排序规则名称可以是 Windows 排序规则名称或

SQL 排序规则名称。collation_name 只适用于 char、varchar、text、nchar、nvarchar 和 ntext 等数据类型列。如果没有指定该参数，则该列的排序规则是用户定义数据类型的排序规则(如果列为用户定义数据类型)或数据库的默认排序规则。

* NULL|NOT NULL：指定列是否允许输入空值。
* CONSTRAINT constraint_name：可选项，定义 DEFAULT 约束名。约束名称必须在表所属的架构中唯一。
* DEFAULT constant_expression：指定字段的默认值表达式。DEFAULT 定义可适用于除定义为 timestamp 或带 IDENTITY 属性的列以外的任何列。如果为用户定义类型列指定了默认值，则该类型应当支持从 constant_expression 到用户定义类型的隐式转换。constant_expression 可以是常量、NULL 或系统函数。
* IDENTITY[(seed,increment)][NOT FOR REPLICATION]：表示新列是标识列。可以将 IDENTITY 属性分配给 tinyint、smallint、int、bigint、decimal、numeric 等数据类型字段。不能对标识列使用默认值约束。seed 是种子，increment 是增量。如果为 IDENTITY 属性指定了 NOT FOR REPLICATION 子句，则复制代理执行插入时，标识列中的值将不会增加。
* ROWGUIDCOL：指示新列是 GUID 列。对于每个表，只能将一个数据类型为 uniqueidentifier 的列指定为 ROWGUIDCOL 列。ROWGUIDCOL 属性并不强制列中所存储值的唯一性。该属性也不会为插入表中的新列自动生成值。若要为每列生成唯一值，请在 INSERT 语句中使用 NEWID 系统函数，或使用 NEWID 函数作为该列的默认值。
* <column_constraint>：指定字段约束。

【例 4-5】 使用 CREATE TABLE 语句在数据库 library 中创建数据表 student 表，保存在 PRIMARY 组中，字段定义参见表 4-2。

语句如下：

```
CREATE TABLE[student](
  [StudentID][char](9) NOT NULL,
  [Name][varchar](10) NOT NULL,
  [Sex][bit]NOT NULL,
  [Birthday][smalldatetime]NOT NULL,
  [BirthPlace][varchar](10) NOT NULL,
  [ID][char](18) NOT NULL,
  [Dept][char](3) NOT NULL,
  [RegistrationDate][smalldatetime]NOT NULL,
)
ON[PRIMARY]
```

【例 4-6】 使用 CREATE TABLE 语句在数据库 library 中创建 book 数据表，保存在 PRIMARY 组中，字段定义参见表 4-6。

book 表结构如表 4-6 所示。

表 4-6 book 表

列 名	数据类型	允 许 空
ISBN	varchar(17)	否
BookName	varchar(50)	否
Author	varchar(50)	否
Publisher	varchar(50)	否
PubishDate	char(7)	否
Price	money	否
Class	varchar(10)	否

语句如下：

```
CREATE TABLE[book](
  [ISBN][varchar](17) NOT NULL,
  [BookName][varchar](50) NOT NULL,
  [Author][varchar](50) NOT NULL,
  [Publisher][varchar](50) NOT NULL,
  [PublishDate][char](7) NOT NULL,
  [Price][money]NOT NULL,
  [Class][varchar](10) NOT NULL,
) ON[PRIMARY]
```

【例 4-7】 使用 CREATE TABLE 语句在数据库 library 中创建 card 数据表，保存在 PRIMARY 组中，并为 StartDate(起效时间)、State(状态)、Maximum(最大允许借阅数)、BorrowedNum(已借数量)字段指定默认值。字段定义及默认值参见表 4-3。

语句如下：

```
CREATE TABLE[card](
  [CardID][char](8) NOT NULL,
  [StartDate][smalldatetime]NOT NULL CONSTRAINT[DF_card_StartDate]DEFAULT
(getdate()),
  [ExpiredDate][smalldatetime]NOT NULL,
  [State][char](4) NOT NULL CONSTRAINT[DF_card_State]DEFAULT ('正常'),
  [Maximum][tinyint]NOT NULL CONSTRAINT[DF_card_Maximum]DEFAULT ((10)),
  [BorrowedNum][tinyint]NOT NULL CONSTRAINT[DF_card_BorrowedNum]DEFAULT
((0)),
  [StudentID][char](9) NOT NULL,
) ON[PRIMARY]
```

【例 4-8】 使用 CREATE TABLE 语句在数据库 library 中创建 bookshelf 数据表，并将 ReturningTime 列定义为计算列。字段定义参见表 4-4。

语句如下：

```
CREATE TABLE[bookshelf](
```

```
  [Barcode][char](8) NOT NULL,
  [LendingTime][smalldatetime]NULL,
  [LendingDays][smallint]NULL,
  [ReturningTime]AS (dateadd(dayofyear,[LendingDays],[LendingTime])),
  [State][char](4) NOT NULL,
  [ISBN][varchar](17) NOT NULL,
  [CardID][char](8) NULL,
  [PlaceID][tinyint]NOT NULL,
) ON[PRIMARY]
```

【例 4-9】 使用 CREATE TABLE 语句在数据库 library 中创建 place 数据表，并将 PlaceID 列定义为标识列，种子和增量分别为 1 和 1。字段定义参见表 4-5。

语句如下：

```
CREATE TABLE[place](
  [PlaceID][tinyint]IDENTITY(1,1) NOT NULL,
  [Description][varchar](50) NOT NULL,
) ON[PRIMARY]
```

4.3 管　理　表

当数据表创建完成后，可以通过查看表属性、修改表结构、重命名表及删除表对表进行管理。管理操作可以通过 SQL Server 2005 管理控制器(即 SQL Server Management Studio)和Transact-SQL 语句实现，本节将重点介绍通过 SQL Server 2005 管理控制器管理表。

4.3.1 查看表的属性

【例 4-10】 使用 SQL Server Management Studio 查看表 student 的结构。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→student，右击，在弹出的快捷菜单中选择“属性”命令，即可打开表属性窗口，如图 4-7 所示。

4.3.2 修改表的结构

【例 4-11】 使用 SQL Server Management Studio 修改表 student 的结构，将 Sex 字段名修改为 Gender，数据类型定义为 char(1)。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→student，右击，在弹出的快捷菜单中选择“修改”命令，即可打开表设计器，选中 Sex 字段，

将其名称改为 Gender，数据类型改为 char(1)，如图 4-8 所示。

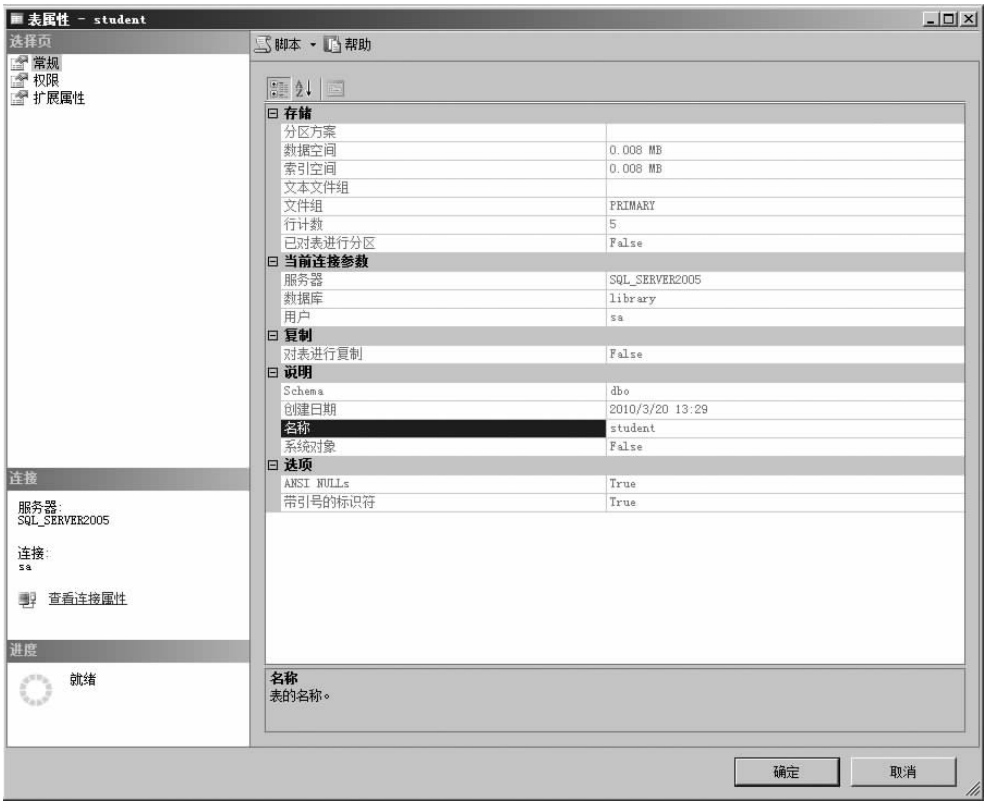

图 4-7 “表属性-student”窗口

图 4-8 修改表的结构

提示:修改表也可以通过 ALTER TABLE 语句实现。

4.3.3　重命名、删除表

【例 4-12】　使用 SQL Server Management Studio 将 student 表重命名为 teacher。

操作步骤如下:

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中,依次展开 SQL_SERVER2005→“数据库”→library→“表”→student,右击,在弹出的快捷菜单中选择“重命名”命令,输入 teacher 即可,如图 4-9 所示。

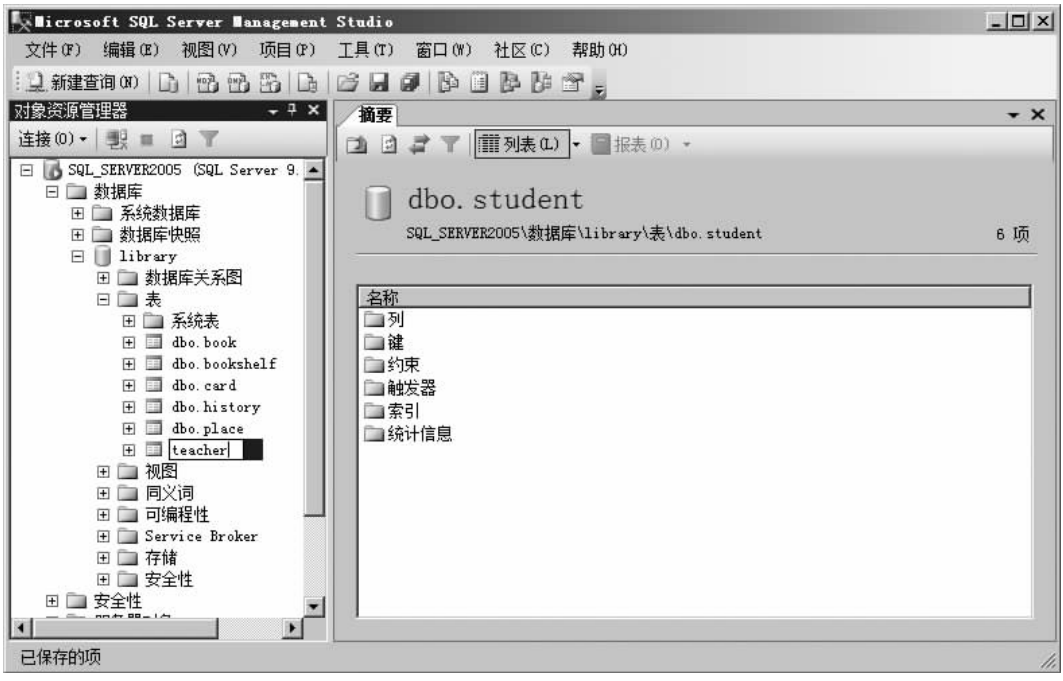

图 4-9　表重命名

【例 4-13】　使用 SQL Server Management Studio 将表 student 删除。

操作步骤如下:

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中,依次展开 SQL_SERVER2005→“数据库”→library→“表”→student,右击,在弹出的快捷菜单中选择“删除”命令,打开“删除对象”窗口,如图 4-10 所示。

(3)选中删除对象,单击“确定”按钮。

注意:删除表后,该表的结构定义、数据、约束和索引等都从数据库中永久删除。

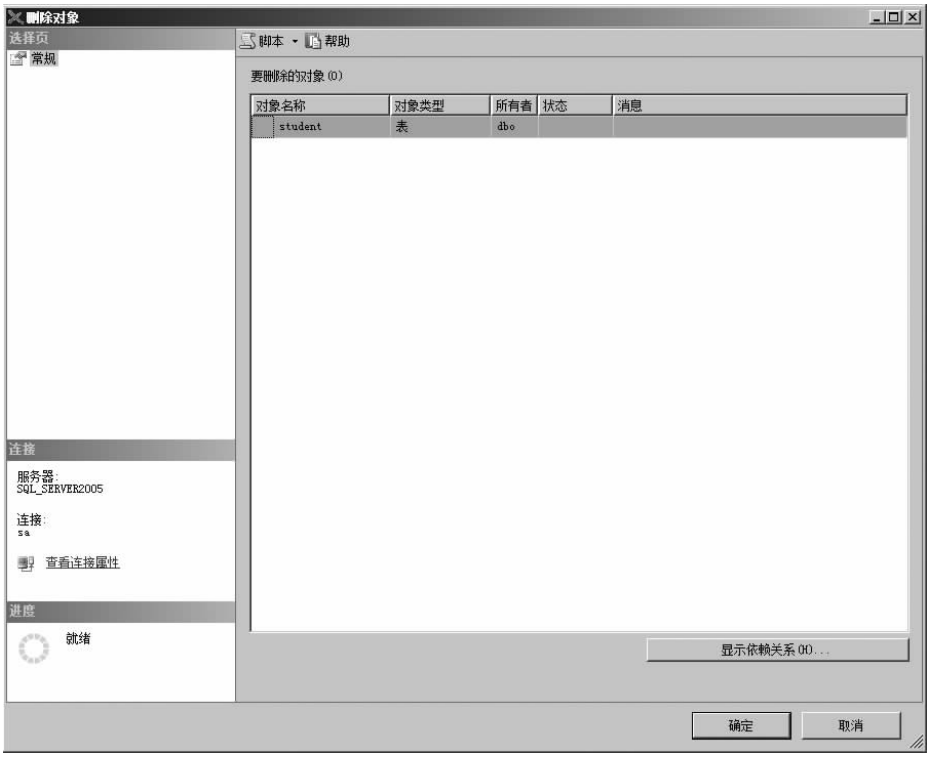

图 4-10 “删除对象”窗口

4.4 数据完整性设计

数据完整性就是确保数据库中数据的一致性和正确性,完整性包括数据的完整性和参照的完整性。在 SQL Server 2005 中,数据库完整性可以通过 PRIMARY KEY 约束、FOREIGN KEY 约束、UNIQUE 约束、CHECK 约束、DEFAULT 约束、允许空值实现。DEFAULT 定义及允许空值参见 4.2 节。

4.4.1 创建约束

1. PRIMARY KEY 约束

表通常具有包含唯一标识表中每一个记录的一个字段或一组字段。这样的一个字段或多个字段的组合称为表的主键(primary key),用于强制表的实体完整性。

PRIMARY KEY 约束中的列不允许为空值和重复的值。一个表只能有一个 PRIMARY KEY 约束,在创建表时允许 PRIMARY KEY 约束作为表定义的一部分,也可以在表创建后,为其添加 PRIMARY KEY 约束。为表中的现有列添加 PRIMARY KEY 约束时,应确保主键所对应的列没有空值或没有重复的值。

如果为表指定了 PRIMARY KEY 约束,则在默认情况下 SQL Server 2005 将为约束列创建唯一聚集索引,强制约束列值的唯一性。如果对多列定义了 PRIMARY KEY 约束,则

来自 PRIMARY KEY 约束定义所有列的任何值组合必须唯一。

【例 4-14】 使用 SQL Server Management Studio，在数据库 library 中将 student 表的 StudentID 字段设置为主键。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. student，右击，在弹出的快捷菜单中选择“修改”命令，打开表设计器。

(3)选中 StudentID 字段行，右击，如图 4-11 所示，在弹出的快捷菜单中选择“设置主键”命令。

图 4-11　设置主键

(4)单击工具栏中的保存按钮，将自动创建名为 PK_(表名)的主键，可以在 student 表下的键节点中看到该主键，如图 4-12 所示。

【例 4-15】 使用 SQL Server Management Studio，在数据库 library 中将 history 表的 ISBN、CardID、LendingTime 三个字段联合设置为主键，history 表的结构如表 4-7 所示。

表 4-7　history 表

列　　名	数据类型	允 许 空
ISBN	varchar(7)	否
CardID	char(8)	否
LendingTime	smalldatetime	否

图 4-12 PK_student 主键

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. history，右击，在弹出的快捷菜单中选择“修改”命令，打开表设计器。

(3)按住 Ctrl 键，依次选中 ISBN、CardID、LendingTime 三个字段行，右击，在弹出的快捷菜单中选择“设置主键”命令，如图 4-13 所示。

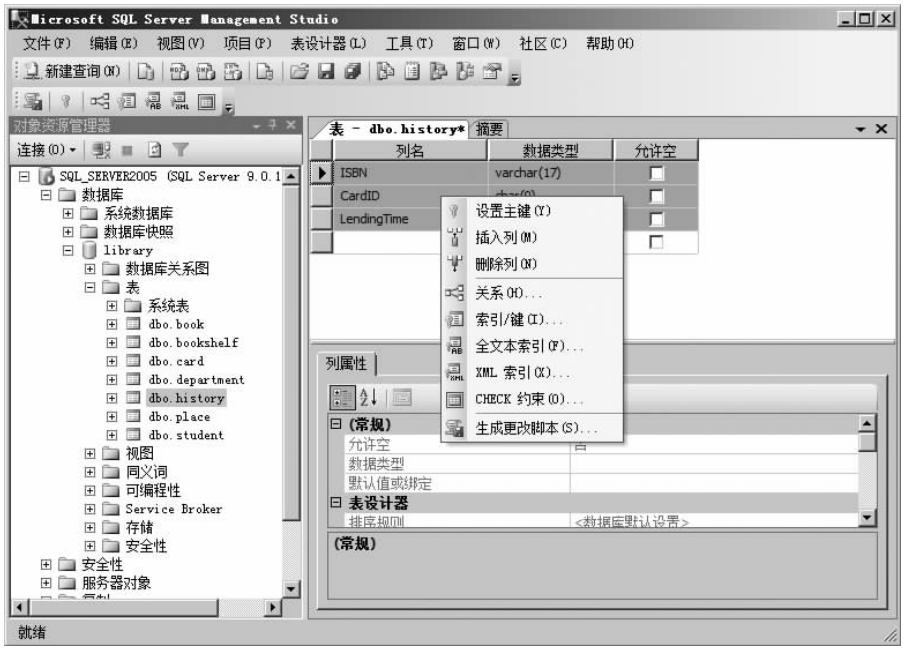

图 4-13 联合设置主键

表中创建过主键后，如果需要，也可以将主键删除。但要注意以下两种情况：

- 如果另一个表中的 FOREIGN KEY 约束引用了 PRIMARY KEY 约束，则必须先删除 FOREIGN KEY 约束。
- 表包含应用于自身的 PRIMARY XML 索引。

在 SQL Server 2005 中，也可以通过 ALTER TABLE 语句为表添加主键。该语句的语法如下：

```
ALTER TABLE[database_name.[schema_name].|schema_name.]table_name
[WITH{CHECK|NOCHECK}]ADD
{<table_constraint>
}[,…n]
table_constraint::=[CONSTRAINT constraint_name]
{
{PRIMARY KEY}
[CLUSTERED|NONCLUSTERED]
(column[ASC|DESC][,…n])
[ON{partition_scheme_name (partition_column_name…)|filegroup|"default"}]
}
```

该命令中各参数说明如下：

- database_name：要在其中创建表的数据库名称。
- schema_name：新表所属架构的名称。
- table_name：新建表的名称。
- WITH{CHECK|NOCHECK}：指定表中的数据是否用新添加的或重新启用的约束进行验证。如果未指定，对于新约束，假定为 WITH CHECK，对于重新启用的约束，假定为 WITH NOCHECK。
- ADD：指定添加一个或多个表约束。
- CONSTRAINT constraint_name：可选项，定义 PRIMARY KEY 约束名。约束名称必须在表所属的架构中唯一。
- PRIMARY KEY：指定增加的是 PRIMARY KEY 约束。
- CLUSTERED|NONCLUSTERED：指示为 PRIMARY KEY 约束创建聚集索引或非聚集索引。PRIMARY KEY 约束默认为 CLUSTERED。
- column[ASC|DESC]：新约束中使用的一个列，使用括号指定。[ASC|DESC]指定加入表约束中列的排序规则，默认值为 ASC(升序)。
- ON：为主键创建的索引指定存储位置。如果指定了 partition_scheme_name，则将对该索引进行分区，并将分区映射到由 partition_scheme_name 指定的文件组。如果指定了 filegroup，则将在命名文件组内创建索引。如果指定了"default"或者根本没有指定 ON，将与表在同一个文件组创建索引。

【例 4-16】 使用 ALTER TABLE 语句，在数据库 library 中将 student 表的 StudentID 字段设置为主键，主键遵守升序排序规则，并建立聚集索引。

语句如下：

```
ALTER TABLE[student]ADD CONSTRAINT[PK_student]PRIMARY KEY CLUSTERED
```

```
(
  [StudentID]ASC
)
ON[PRIMARY]
```

【例 4-17】 使用 ALTER TABLE 语句，在数据库 library 中将 history 表的 ISBN、CardID、LendingTime 三个字段联合设置为主键。

语句如下：

```
ALTER TABLE[history]ADD CONSTRAINT[PK_history]PRIMARY KEY CLUSTERED
(
  [ISBN]ASC,
  [CardID]ASC,
  [LendingTime]ASC
)
ON[PRIMARY]
```

在 SQL Server 2005 中，也可以在 CREATE TABLE 语句中为表指定主键。若主键只包含一个字段，可以直接在字段上定义主键约束。在字段上定义主键约束，则字段约束定义如下：

```
<column_constraint>::=
[CONSTRAINT constraint_name]
{{PRIMARY KEY}
[CLUSTERED|NONCLUSTERED]
[ON{partition_scheme_name (partition_column_name)|filegroup|"default"}]}
```

【例 4-18】 在数据库 library 中创建 student 表时将 StudentID 字段指定为主键，并在该字段上建立聚集索引。

语句如下：

```
CREATE TABLE[dbo].[student](
  [StudentID][char](9) NOT NULL CONSTRAINT[PK_student]PRIMARY KEY CLUSTERED ON
[PRIMARY],
  [Name][varchar](10) NOT NULL,
  [Sex][bit]NOT NULL,
  [Birthday][smalldatetime]NOT NULL,
  [BirthPlace][varchar](10) NOT NULL,
  [ID][char](18) NOT NULL,
  [Dept][char](3) NOT NULL,
  [RegistrationDate][smalldatetime]NOT NULL,
) ON[PRIMARY]
```

若主键包含多个字段，则需要在表上定义主键约束。表约束定义参见 ALTER TABLE 语句的 table_constraint 定义。

【例 4-19】 在数据库 library 中创建 history 表时，将 ISBN、CardID、LendingTime 三个字段联合设置为主键。

语句如下：

```
CREATE TABLE[dbo].[history](
  [ISBN][varchar](17) NOT NULL,
  [CardID][char](8) NOT NULL,
  [LendingTime][smalldatetime]NOT NULL,
CONSTRAINT[PK_history]PRIMARY KEY CLUSTERED
([ISBN]ASC,[CardID]ASC,[LendingTime]ASC) ON[PRIMARY]
) ON[PRIMARY]
```

2. UNIQUE 约束

UNIQUE 约束确保在非主键列中不输入重复的值。UNIQUE 约束与 PRIMARY KEY 约束不同，一个表可以有多个 UNIQUE 约束，但一个表只可以有一个 PRIMARY KEY 约束。UNIQUE 约束列允许输入 NULL 值，而 PRIMARY KEY 约束列不允许。

创建表时，可以创建 UNIQUE 约束作为表定义的一部分。如果表已经存在，可以添加 UNIQUE 约束(假设组成 UNIQUE 约束的列或列组合仅包含唯一的值)。

SQL Server 2005 将自动为 UNIQUE 约束列创建唯一索引。因此，如果试图在 UNIQUE 约束列上插入重复值，数据库将返回错误消息。默认情况下为 UNIQUE 约束列创建唯一的非聚集索引。若表中不存在聚集索引，也可以为 UNIQUE 约束列指定创建唯一聚集索引。

【例 4-20】 使用 SQL Server Management Studio 在数据库 library 中为 student 表的 ID 字段增加 UNIQUE 约束。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. student，右击，在弹出的快捷菜单中选择“修改”命令，打开表设计器。

(3)在“表设计器”菜单中，选择“索引/键”命令，打开“索引/键”对话框。

(4)单击“索引/键”对话框中的“添加”按钮，如图 4-14 所示，在属性编辑框中，将“类型”设置为“唯一键”，“列”改为 ID 字段(单击后面的 … 按钮，弹出“索引列”对话框，如图 4-15 所示，“列名”选择 ID，“排序顺序”选择升序)，单击“确定”按钮，如图 4-16 所示，然后单击“关闭”按钮。

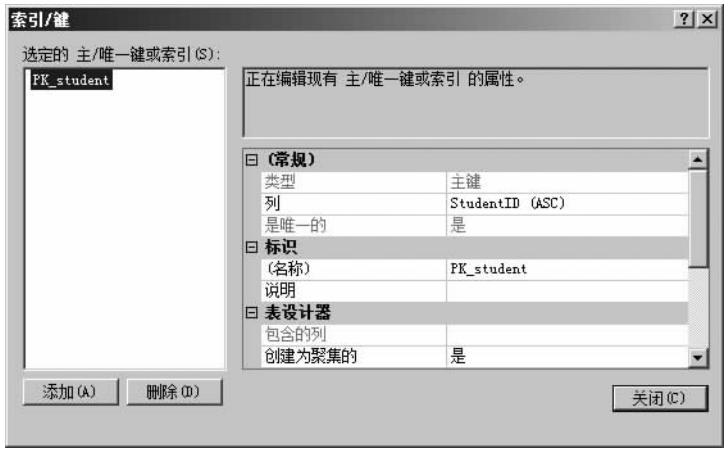

图 4-14　“索引/键”对话框

图 4-15 “索引列”对话框

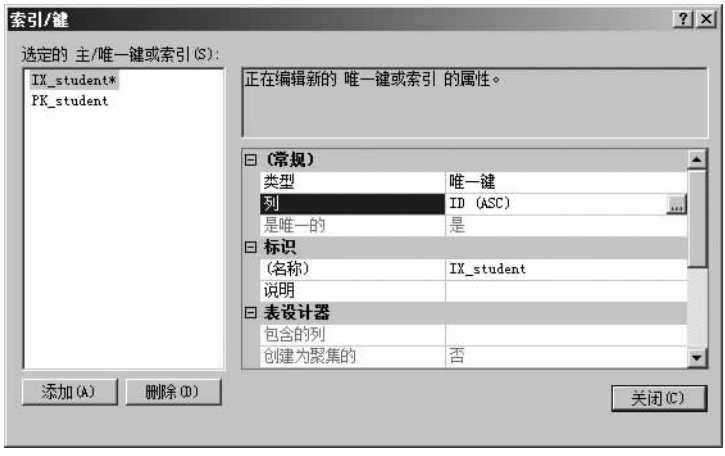

图 4-16 添加唯一键

(5)单击工具栏中的保存按钮,会在 student 表的“键”节点和“索引”下看到刚创建的 UNIQUE 约束。

提示:创建 UNIQUE 约束时,若没有为唯一键起名,则采用系统默认的名字,否则,可以在图 4-16 的名称字段中为唯一键命名。

【例 4-21】 使用 SQL Server Management Studio,在数据库 library 中为 book 表的 BookName、Author、Publisher 字段增加 UNIQUE 约束。

操作步骤如下:

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中,依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. book,右击,在弹出的快捷菜单中选择“修改”命令,打开表设计器。

(3)在“表设计器”菜单中,选择“索引/键”命令,打开“索引/键”对话框。

(4)单击“添加”按钮,在属性编辑框中,将“类型”设置为“唯一键”,“列”改为 BookName、Author、Publisher 字段,“是唯一的”设置为“是”,然后单击“关闭”按钮。

在 SQL Server 2005 中,也可以通过 ALTER TABLE 语句为表添加 UNIQUE 约束。该语句的语法与创建 PRIMARY KEY 约束的语法类似,只需要将关键字由 PRIMARY

KEY 改为 UNIQUE 即可。

【例 4-22】 使用 ALTER TABLE 语句，在数据库 library 中为 student 表的 ID 字段增加 UNIQUE 约束。

语句如下：

```
ALTER TABLE[student]ADD CONSTRAINT[IX_student]UNIQUE NONCLUSTERED
(
  [ID]ASC
) ON[PRIMARY]
```

在 SQL Server 2005 中，也可以在 CREATE TABLE 语句中为表中的某一列或若干列组合增加 UNIQUE 约束。其语法可参见为表中的某一列或若干列组合增加 PRIMARY KEY 约束。

【例 4-23】 在数据库 library 中创建 book 表时，为 BookName、Author、Publisher 字段增加 UNIQUE 约束。

语句如下：

```
CREATE TABLE[book](
  [ISBN][varchar](17) NOT NULL,
  [BookName][varchar](50) NOT NULL,
  [Author][varchar](50) NOT NULL,
  [Publisher][varchar](50) NOT NULL,
  [PublishDate][char](7) NOT NULL,
  [Price][money]NOT NULL,
  [Class][varchar](10) NOT NULL,
CONSTRAINT[IX_book]UNIQUE NONCLUSTERED
([Author]ASC,[Publisher]ASC,[BookName]ASC) ON[PRIMARY]
) ON[PRIMARY]
```

3. FOREIGN KEY 约束

PRIMARY KEY 约束和 UNIQUE 约束机制保障实体完整性，而 FOREIGN KEY 约束的主要目的是控制可以存储在外键表中的数据和控制对主键表中数据的更改，从而实现参照完整性。例如，如果在 student 表（主键表）中删除一个学生记录（代表注销一个学生），而这个学生的 StudentID 字段由 card 表（外键表）使用，则这两个表之间关联的完整性将被破坏；card 表中学生的借书证因为与 student 表中的数据没有链接而变得孤立了。

在外键表中，用 FOREIGN KEY 约束的列称为外键（foreign key）。外键的值可以包含空值，但是，包含空值的记录将跳过 FOREIGN KEY 约束验证。若要确保验证 FOREIGN KEY 约束的列，需将所有参与外键的列指定为 NOT NULL。FOREIGN KEY 约束并不仅仅可以与另一表的 PRIMARY KEY 约束相链接，它还可以定义为引用另一表的 UNIQUE 约束。

创建表时，可以创建 FOREIGN KEY 约束作为表定义的一部分。如果表已经存在，则可以添加 FOREIGN KEY 约束。

【例 4-24】 使用 SQL Server Management Studio，在数据库 library 中为 card 表的 StudentID 字段增加 FOREIGN KEY 约束。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. card，右击，在弹出的快捷菜单中选择“修改”命令，打开表设计器。

(3)在“表设计器”菜单中，选择“关系”命令，打开“外键关系”对话框，如图 4-17 所示。

图 4-17 “外键关系”对话框

(4)在“外键关系”对话框中，单击“添加”按钮，将增加一个关系，该关系将以系统提供的名称显示在“选定的关系”列表中，名称格式为 FK_＜tablename1＞_＜tablename2＞，其中 tablename1 是外键表名称，tablename2 是主键表名称。

(5)在“选定的关系”列表中单击新创建的关系，在关系编辑框中选中“表和列规范”选项，单击右侧的省略号，打开“表和列”对话框。

(6)在“表和列”对话框中，在“主键表”列选择主键表 student、主键列 StudentID；在“外键表”列选择外键表 card、外键列 StudentID，如图 4-18 所示，单击“确定”按钮。

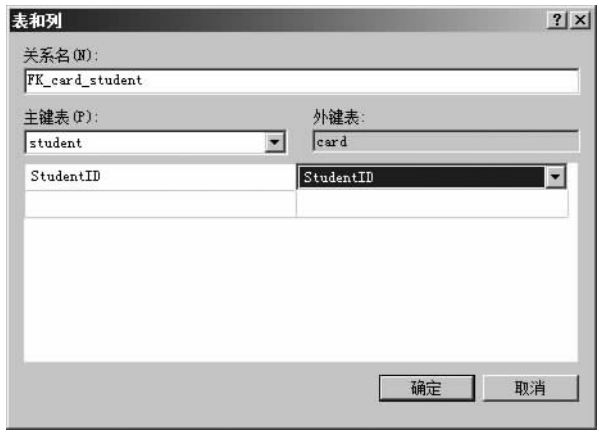

图 4-18 “表和列”对话框

(7)若要更改此关系名称,可编辑“名称”属性,否则,单击“关闭”按钮。

(8)单击工具栏中的保存按钮,在 card 表的“键”节点下即可看到刚创建的外键。

在 SQL Server 2005 中,也可以通过 ALTER TABLE 语句为表添加 FOREIGN KEY 约束。该语句的语法如下:

```
ALTER TABLE[database_name.[schema_name].|schema_name.]table_name
[WITH{CHECK|NOCHECK}]ADD
{<table_constraint>
}[,…n]
table_constraint::=[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]
}
```

该命令中各参数说明如下:

- FOREIGN KEY:指定约束为 FOREIGN KEY 约束。
- (column[,…n]):在外键表中指定 FOREIGN KEY 约束列。
- REFERENCES referenced_table_name[(ref_column[,…n])]:referenced_table_name 主键表,ref_column 是主键表中的主键或唯一键。
- ON DELETE{NO ACTION|CASCADE|SET NULL|SET DEFAULT}:指定主键表中的记录被删除时,若该记录在外键表中还存在被引用关系,则对主键表和外键表中的记录采取什么操作,默认值为 NO ACTION。若为 NO ACTION,则使 SQL Server 2005 引发错误,并对主键表执行回滚操作,恢复原来的记录;若为 CASCADE,将在主键表中删除记录的同时,从外键表中删除相应行;若为 SET NULL,将在删除主键表中记录的同时,外键表中的所有外键值都将设置为 NULL,执行此操作,外键列可为空值;SET DEFAULT,将在删除主键表中记录的同时,外键表中的所有外键值都将设置为默认值,执行此操作,所有外键列都必须有默认值定义,如果某个列可为空值,并且未设置显式的默认值,则将使用 NULL 作为该列的隐式默认值。
- ON UPDATE{NO ACTION|CASCADE|SET NULL|SET DEFAULT}:指定主键表中的记录被更新时,若该记录在外键表中还存在被引用关系,则对主键表和外键表中的记录采取什么操作,默认值为 NO ACTION。若为 NO ACTION,则数据库将引发错误,并回滚对主键表中相应记录的更新操作;若为 CASCADE,将在主键表中更新记录的同时,在外键表中更新相应记录;若为 SET NULL,则在更新主键表中记录的同时,外键表中的所有外键值都将设置为 NULL,执行此操作,外键列必须可为空值;SET DEFAULT,则在更新主键表中记录的同时,外键表中的所有外键值都将设置为默认值,执行此操作,所有外键列都必须有默认值定义,如果某个列可为空值,并且未设置显式的默认值,则将使用 NULL 作为该列的隐式默认值。

• NOT FOR REPLICATION:可以为 FOREIGN KEY 约束指定该参数。如果为约束指定了此子句,则当复制代理执行插入、更新或删除操作时,将不会强制执行此约束。

【例 4-25】 使用 ALTER TABLE 语句在数据库 library 中为 bookshelf 的 CardID、ISBN、PlaceID 字段增加 FOREIGN KEY 约束,分别参照 card 表的 CardID 字段、book 表的 ISBN 字段、place 表的 PlaceID 字段。对于 bookshelf 表中的 CardID、ISBN、PlaceID 字段,其 FOREIGN KEY 约束的 ON DELETE 设置为 NO ACTION,ON UPDATE 分别设置为 CASCADE、NO ACTION、NO ACTION。

语句如下:

```
ALTER TABLE[bookshelf]WITH CHECK ADD CONSTRAINT[FK_bookshelf_book]FOREIGN KEY
([ISBN])
REFERENCES[book]([ISBN])
ON DELETE NO ACTION
ON UPDATE NO ACTION
GO
ALTER TABLE[dbo].[bookshelf]WITH CHECK ADD CONSTRAINT[FK_bookshelf_card]FOR-
EIGN KEY([CardID])
REFERENCES[dbo].[card]([CardID])
ON DELETE NO ACTION
ON UPDATE CASCADE
GO
ALTER TABLE[dbo].[bookshelf]WITH CHECK ADD CONSTRAINT[FK_bookshelf_place]FOR-
EIGN KEY([PlaceID])
REFERENCES[dbo].[place]([PlaceID])
ON DELETE NO ACTION
ON UPDATE NO ACTION
```

在 SQL Server 2005 中,也可以在 CREATE TABLE 语句中为表指定外键。若外键只包含一个字段,可以直接在字段上定义外键约束。在字段上定义外键约束,需遵守以下语法:

```
<column_constraint>::=
[CONSTRAINT constraint_name]
[FOREIGN KEY]
REFERENCES[schema_name.]referenced_table_name[(ref_column)]
[ON DELETE{NO ACTION|CASCADE|SET NULL|SET DEFAULT}]
[ON UPDATE{NO ACTION|CASCADE|SET NULL|SET DEFAULT}]
[NOT FOR REPLICATION]
```

【例 4-26】 使用 CREATE TABLE 语句在数据库 library 中为 history 表的 ISBN 字段增加 FOREIGN KEY 约束,参照 book 表的 ISBN 字段,ON DELETE 和 ON UPDATE 均设置为 CASCADE。

语句如下:

```
CREATE TABLE[history](
```

```
[ISBN][varchar](17) NOT NULL CONSTRAINT[FK_history_book]FOREIGN KEY REFERENCES[book]([ISBN]) ON DELETE CASCADE ON UPDATE CASCADE,
[CardID][char](8) NOT NULL,
[LendingTime][smalldatetime]NOT NULL,
) ON[PRIMARY]
```

在表上定义外键约束，需要遵守表约束语法，可参阅 ALTER TABLE 的 table_constraint 生成式。

【例 4-27】 使用 CREATE TABLE 语句在数据库 library 中为 history 表的 ISBN 字段和 CardID 字段增加 FOREIGN KEY 约束，分别参照 book 表的 ISBN 字段和 card 表的 CardID 字段，ON DELETE 和 ON UPDATE 均设置为 CASCADE。

语句如下：

```
CREATE TABLE[history](
[ISBN][varchar](17) NOT NULL,
[CardID][char](8) NOT NULL,
[LendingTime][smalldatetime]NOT NULL,
CONSTRAINT[FK_history_book]FOREIGN KEY ([ISBN]) REFERENCES[book]([ISBN]) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT[FK_history_card]FOREIGN KEY ([CardID]) REFERENCES[card]([CardID]) ON DELETE CASCADE ON UPDATE CASCADE
) ON[PRIMARY]
```

4. CHECK 约束

CHECK 约束不接受计算结果为 FALSE 的值，从而限制列可接受的值，实现域的完整性。CHECK 约束通过定义逻辑表达式确定列的有效值，例如，Maximum<＝20。由于空值的计算结果为 UNKNOWN，所以表达式中存在这些值可能会越过约束验证，而在允许为空值的列中插入空值。

可以将多个 CHECK 约束应用于单个列，还可以在表级创建 CHECK 约束。创建表时，可以创建 CHECK 约束作为表定义的一部分。如果表已经存在，则可以添加 CHECK 约束。

【例 4-28】 使用 SQL Server Management Studio，在数据库 library 中为 card 表的 State 字段增加 CHECK 约束，限制输入“正常”和“挂失”两个值，限制 Maximum 字段最大值为 20。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo.card，右击，在弹出的快捷菜单中选择“修改”命令，打开表设计器。

(3)在“表设计器”菜单中，选择“CHECK 约束”命令，打开“CHECK 约束”对话框，如图 4-19 所示。

(4)单击“添加”按钮，在 CHECK 约束属性对话框中，选中“表达式”选项，单击右端的省略号按钮，弹出“CHECK 约束表达式”对话框，输入表达式“(State＝'正常' or State＝'挂失') and Maximum<＝20”，如图 4-20 所示，单击“确定”按钮。

(5)单击图 4-19 中的“关闭”按钮，然后单击工具栏中的保存按钮，即可在 card 表的约束

节点下看到刚创建的 CHECK 约束。

图 4-19 “CHECK 约束”对话框

图 4-20 “CHECK 约束表达式”对话框

在 SQL Server 2005 中,也可以通过 ALTER TABLE 语句为表添加 CHECK 约束。该语句的语法如下:

```
ALTER TABLE[database_name.[schema_name].|schema_name.]table_name
[WITH{CHECK|NOCHECK}]ADD
{<table_constraint>
}[,…n]
table_constraint::=[CONSTRAINT constraint_name]
{
    CHECK[NOT FOR REPLICATION](logical_expression)
}
```

该命令中各参数说明如下:

- CHECK[NOT FOR REPLICATION](logical_expression):定义 CHECK 约束。logical_expression 用于 CHECK 约束的逻辑表达式,返回 TRUE 或 FALSE。若返回值为 FALSE,则记录插入或更新失败。logical_expression 表达式不能引用其他表。
- NOT FOR REPLICATION:可以为 CHECK 约束指定该参数。如果为约束指定了此子句,则当复制代理执行插入、更新或删除操作时,将不会强制执行此约束。

【例 4-29】 使用 ALTER TABLE 语句在数据库 library 中为 card 表的 State 字段增加 CHECK 约束,限制输入“正常”和“挂失”两个值,限制 Maximum 字段最大值为 20。

语句如下:

```
ALTER TABLE[card]WITH CHECK ADD CONSTRAINT[CK_card]CHECK (([State]='正常'OR
[State]='挂失') AND[Maximum]<=(20))
```

在 SQL Server 2005 中，也可以在 CREATE TABLE 语句中为表指定 CHECK 约束。若 CHECK 约束只包含一个字段，可以直接在字段上定义 CHECK 约束。在字段上定义 CHECK 约束，需遵守以下语法：

```
<column_constraint>::={CHECK[NOT FOR REPLICATION](logical_expression)}
```

在表上定义 CHECK 约束，遵守以下语法：

```
<table_constraint>::=[CONSTRAINT constraint_name]{CHECK[NOT FOR REPLICA-
TION](logical_expression)}
```

【例 4-30】 使用 CREATE TABLE 语句在数据库 library 中为 card 表的 State 字段增加 CHECK 约束，限制输入“正常”和“挂失”两个值，限制 Maximum 字段最大值为 20。

语句如下：

```
CREATE TABLE[card](
  [CardID][char](8) NOT NULL,
  [StartDate][smalldatetime]NOT NULL,
  [ExpiredDate][smalldatetime]NOT NULL,
  [State][char](4) NOT NULL CHECK[State]='正常'OR[State]='挂失',
  [Maximum][tinyint]NOT NULL CHECK[Maximum]<=20,
  [BorrowedNum][tinyint]NOT NULL,
  [StudentID][char](9) NOT NULL,
) ON[PRIMARY]
```

或者，

```
CREATE TABLE[card](
  [CardID][char](8) NOT NULL,
  [StartDate][smalldatetime]NOT NULL,
  [ExpiredDate][smalldatetime]NOT NULL,
  [State][char](4) NOT NULL,
  [Maximum][tinyint]NOT NULL,
  [BorrowedNum][tinyint]NOT NULL,
  [StudentID][char](9) NOT NULL,
  CONSTRAINT[CK_card]CHECK (([State]='正常'OR[State]='挂失') AND[Maximum]<=(20))
) ON[PRIMARY]
```

4.4.2　创建规则

规则是一个向后兼容的功能，用于执行一些与 CHECK 约束相同的功能。使用 CHECK 约束是限制列值的首选标准方法。一个列只能应用一个规则，但可以应用多个 CHECK 约束。CHECK 约束是表的一部分，而规则是作为单独的对象创建，然后绑定到列上。列或别名数据类型只能被绑定一个规则。不过，列可以同时有一个规则以及一个或多个 CHECK 约束与之关联。在这种情况下，将评估所有限制。

提示：后续版本的 Microsoft SQL Server 已删除该功能，应避免在新的开发工作中使用

该功能。

1. 创建规则

```
CREATE RULE[schema_name.]rule_name AS condition_expression[;]
```

其中，condition_expression 定义规则的条件表达式。规则可以包括算术运算符、关系运算符、谓词(如 IN、LIKE、BETWEEN)和不引用数据库对象的内置函数；不能引用列、数据库对象、用户定义函数。condition_expression 包括一个变量。每个局部变量的前面都有一个“@”符号，该变量引用通过 UPDATE 或 INSERT 语句输入的值。

【例 4-31】 创建一个名为 max_value 的规则，其包含的局部变量定义为 max。

```
CREATE RULE max_value AS @max<=20
```

【例 4-32】 创建一个名为 list_value 的规则，其包含的局部变量定义为 list。

```
CREATE RULE list_value AS @list IN ('正常','挂失')
```

2. 绑定规则

```
sp_bindrule[@rulename=]'rule',[@objname=]'object_name'
```

该命令中各参数说明如下：

- [@rulename=]'rule'：绑定的规则名，要用单引号引起来。
- [@objname=]'object_name'：规则所应用的对象，可以是表、列、别名数据类型。不能将规则绑定到 text、ntext、image、varchar(max)、nvarchar(max)、varbinary(max)、xml、CLR 用户定义类型或 timestamp 列。无法将规则绑定到计算列。

【例 4-33】 将 max_value 规则应用于 card 表的 Maximum 字段上。

语句如下：

```
EXEC sp_bindrule 'max_value','card.Maximum'
```

【例 4-34】 将 list_value 规则应用于 card 表的 State 字段上。

语句如下：

```
EXEC sp_bindrule @rulename='list_value',@objname='card.State'
```

3. 解除和删除规则

(1)解除规则语法如下：

```
sp_unbindrule[@objname=]'object_name'
```

【例 4-35】 解除 card 表 Maximum 字段上的规则。

语句如下：

```
EXEC sp_unbindrule 'card.Maximum'
```

【例 4-36】 解除 card 表 State 字段上的规则。

语句如下：

```
EXEC sp_unbindrule 'card.State'
```

(2)删除规则语法如下：

```
DROP RULE{[schema_name.]rule_name}[,…n][;]
```

【例 4-37】 删除 max_value 规则。

语句如下：

```
DROP RULE max_value
```

4. 规则的查看

使用 SQL Server Management Studio 查看规则的操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“可编程性”→“规则”，如图 4-21 所示，即可看到数据库中所创建的规则。

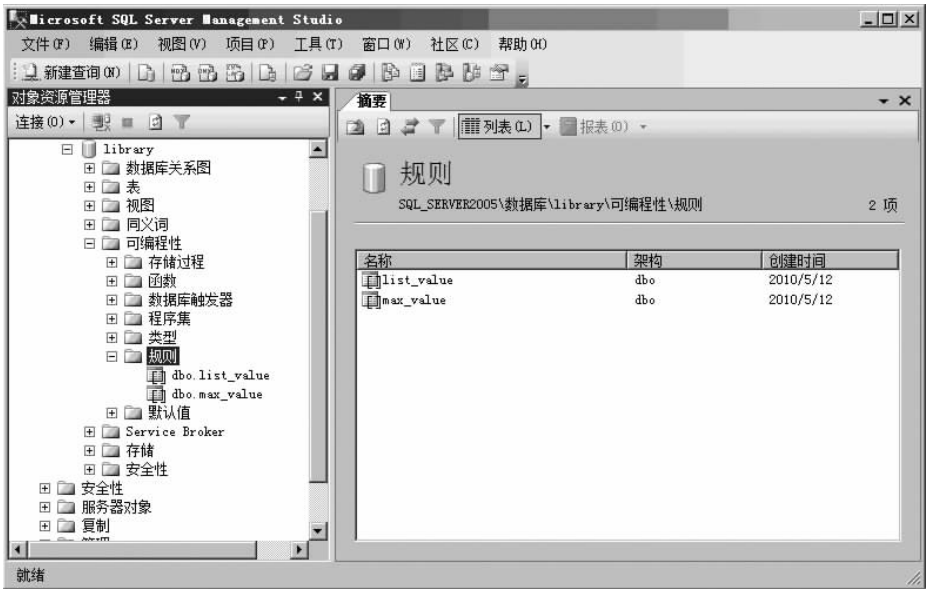

图 4-21　查看规则

4.4.3　创建默认值对象

为字段绑定默认值对象时，通常为数值列指定零作为默认值，为字符串列指定空字符作为默认值。

结合默认值定义列的为空性定义，字段中的值可归纳为如表 4-8 所示。

表 4-8　默认值

列 定 义	无输入，无 DEFAULT 定义	无输入，有 DEFAULT 定义	输入空值
允许空值	NULL	默认值	NULL
不允许空值	错误	默认值	错误

为字段指定默认值可以通过两种方法实现：一是在创建表时，为字段指定默认值，参见 4.2.1 中的相关内容；另一种是创建默认值对象，绑定到列或别名数据类型上。

注意：后续版本的 Microsoft SQL Server 版本中已删除 CREATE DEFAULT，请避免在新的开发工作中使用 CREATE DEFAULT。

1. 创建默认值对象

语法如下：

```
CREATE DEFAULT[schema_name.]default_name AS constant_expression[;]
```

该命令中各参数说明如下：

- default_name：默认值的名称，默认值名称必须遵守标识符命名规则。
- constant_expression：只包含常量值的表达式（它不能包括任何列或其他数据库对象的名称）。可以使用任何常量、内置函数或数学表达式，不能使用用户定义函数、包含别名数据类型的表达式。默认值必须与列数据类型兼容。

【例 4-38】 为 card 表的 State 字段创建默认值对象 default_State，默认值取值为“正常”。

语句如下：

```
CREATE DEFAULT default_State AS '正常'
```

【例 4-39】 为 card 表的 StartDate 字段创建默认值对象 default_date，默认取值为系统当前日期。

语句如下：

```
CREATE DEFAULT default_date AS getdate()
```

2. 绑定默认值对象

绑定默认值对象的语法如下：

```
sp_bindefault[@defname=]'default',[@objname=]'object_name'
```

该命令中各参数说明如下：

- [@defname=]'default'：指定默认值对象。
- [@objname=]'object_name'：表名和列名或者绑定默认值的别名数据类型。

【例 4-40】 将 default_State 默认值对象绑定到 card 表的 State 字段上。

语句如下：

```
EXEC sp_bindefault 'default_State','card.State'
```

【例 4-41】 将 default_date 默认值对象绑定到 card 表的 StartDate 字段上。

语句如下：

```
EXEC sp_bindefault @defname='default_date',@objname='card.StartDate'
```

3. 解除默认值对象

解除默认值对象的语法如下：

```
sp_unbindefault[@objname=]'object_name'
```

【例 4-42】 解除 card 表中 State 字段上的默认值对象。

语句如下：

```
EXEC sp_unbindefault 'card.State'
```

4. 删除默认值对象

删除默认值对象语法如下：

```
DROP DEFAULT{[schema_name.]default_name}[,…n][;]
```

【例 4-43】 删除 default_State 默认值对象。

语句如下：

```
DROP DEFAULT default_State
```

5. 查看默认值对象

使用 SQL Server Management Studio 查看默认值对象的操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中,依次展开 SQL_SERVER2005→“数据库”→library→“可编程性”→“默认值”,如图 4-22 所示,即可看到数据库中所创建的默认值对象。

图 4-22　查看默认值对象

4.5　关系的设计

在关系数据库中,关系可以防止冗余数据,关系通过匹配键列(通常是两个表中同名列)中的数据来发挥作用。在大多数情况下,关系将一个表的主键与另一表中的外键相匹配。

4.5.1　关系概述

在 SQL Server 2005 中,表之间存在 3 种类型的关系:一对一关系、一对多关系和多对多关系,所创建的关系类型取决于相关列的定义。student 表与 card 表是一对一关系,card 表与 bookshelf 表、book 表与 bookshelf 表、place 表与 bookshelf 表、card 表与 history 表、book 表与 history 表、department 表与 student 表是一对多关系。

4.5.2　创建关系图

在创建关系图时,需要事先定义好表的主键与外键。在 library 数据库中创建的关系如表 4-9 所示。

表 4-9 library 数据库中的关系

关 系 名	主 键	外 键
FK_card_student	student. StudentID	card. StudentID
FK_bookshelf_card	card. CardID	bookshelf. CardID
FK_bookshelf_book	book. ISBN	bookshelf. ISBN
FK_bookshelf_place	place. PlaceID	bookshelf. PlaceID
FK_history_card	card. CardID	history. CardID
FK_history_book	book. ISBN	history. ISBN
FK_student_department	department. DeptID	student. Dept

【例 4-44】 使用 SQL Server Management Studio 创建 library 数据库数据表之间的关系图。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→ “数据库关系图”，右击，在弹出的快捷菜单中选择“新建数据库关系图”命令，弹出“添加表”对话框，如图 4-23 所示。

图 4-23 “添加表”对话框

(3)在“添加表”对话框中，选择关系图中所要包含的表，按住 shift 键，单击 book 表和 student 表，选中所有的表。

(4)单击“添加”按钮，即可创建好 library 数据库数据表间的关系图；单击“关闭”按钮，将“添加表”对话框关闭，新建的关系如图 4-24 所示。

(5)单击工具栏中的保存按钮，在弹出的“选择名称”对话框中，为新建的关系图输入一个名称，如图 4-25 所示，单击“确定”按钮。

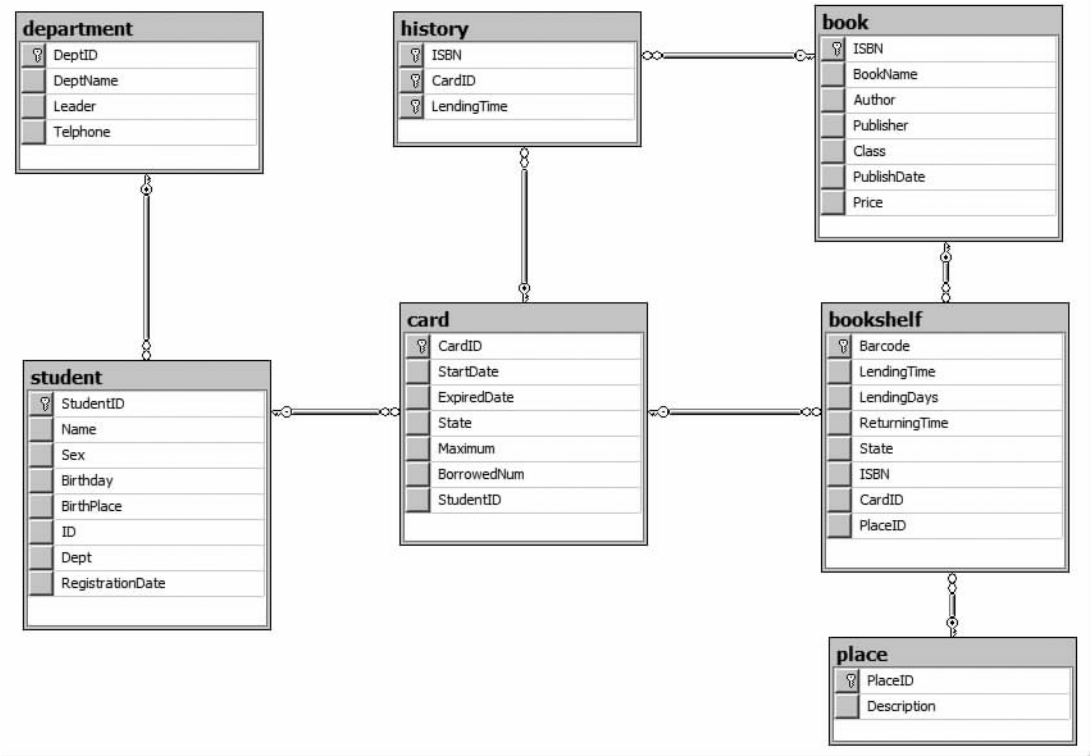

图 4-24　library 数据库关系图

图 4-25　“选择名称”对话框

4.6　索引的设计

索引是与表或视图关联的数据结构，包含表或视图中一列或多列的值，存储在磁盘上，可以加快从表或视图中检索记录的速度。对建有索引的表或视图进行查询时，先按索引的列的值在索引中搜索，查找到所需记录的存储位置后，再从该位置表或视图中提取匹配记录。建立索引可以提高对表或视图的查询性能，且可以减少磁盘 I/O 操作。

建立索引时还应考虑以下数据库准则：

(1)一个表如果建有大量索引会影响 INSERT、UPDATE 和 DELETE 语句的性能，因为更改表中的数据时，所有索引都需进行适当的调整。要避免对经常更新的表建立过多的索引，并且索引的列应尽可能少。

(2)对小表进行索引可能不会产生优化效果，因为查询优化器在遍历用于搜索数据的索引时，花费的时间可能比执行简单的表扫描还长。因此，小表的索引可能从来不用，但仍必

须在表中的数据更改时被维护。

4.6.1 索引概述

在 SQL Server 2005 中，可以创建的索引类型有：聚集索引、非聚集索引。聚集索引是根据索引列的值在表或视图中排序、存储记录。每个表只能有一个聚集索引，因为记录本身只能按一个顺序排序。只有当表包含聚集索引时，表中的记录才有序存储，否则记录存储在一个称为堆的无序结构中。非聚集索引具有独立于表或视图的结构，包含非聚集索引列的值，并且每个列值都有指向包含该列值的记录的指针。一个表可以包含多个非聚集索引。

索引还包含以下两个特性：唯一还是不唯一和包含性列。聚集索引和非聚集索引都可以是唯一的，这意味着任何两条记录都不能有相同的索引列值。另外，索引也可以是不唯一的，即多记录可以共享同一键值。包含性列指在创建索引时，索引不仅包含索引的列，还可以包含非索引的列。

在表上创建 PRIMARY KEY 约束和 UNIQUE 约束时，会自动创建索引。默认情况下，会在主键上创建唯一聚集索引，除非表中已存在聚集索引或在主键上指定了唯一的非聚集索引；在 UNIQUE 约束列上创建唯一非聚集索引，除非已明确指定唯一的聚集索引且表中不存在聚集索引。

4.6.2 创建索引

索引会在表定义创建 PRIMARY KEY 约束和 UNIQUE 约束时自动创建，也可以在表创建后创建。在 SQL Server 2005 中，可利用 SQL Server 2005 管理控制器(即 Management Studio)和 Transact-SQL 语句创建索引。

1. 使用 SQL Server Management Studio 创建索引

【例 4-45】 使用 SQL Server Management Studio 在 student 表的 StudentID 字段上创建唯一聚集索引 IX_student。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. student→“索引”，右击，在弹出的快捷菜单中选择“新建索引”命令，打开“新建索引”窗口，如图 4-26 所示。

(3)在“索引名称”文本框中输入 IX_Student，“索引类型”选择“聚集”，选中“唯一”复选框，单击“添加”按钮，打开“从 dbo. student 中选择列”窗口，如图 4-27 所示。

(4)选择 StudentID 字段，单击“确定”按钮，返回“新建索引”窗口后单击“确定”按钮即可。

【例 4-46】 使用 SQL Server Management Studio 在 student 表的 ID 字段上创建唯一非聚集索引 IX_student_ID。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. student，右击，在弹出的快捷菜单中选择“修改”命令，打开表设计器窗口。

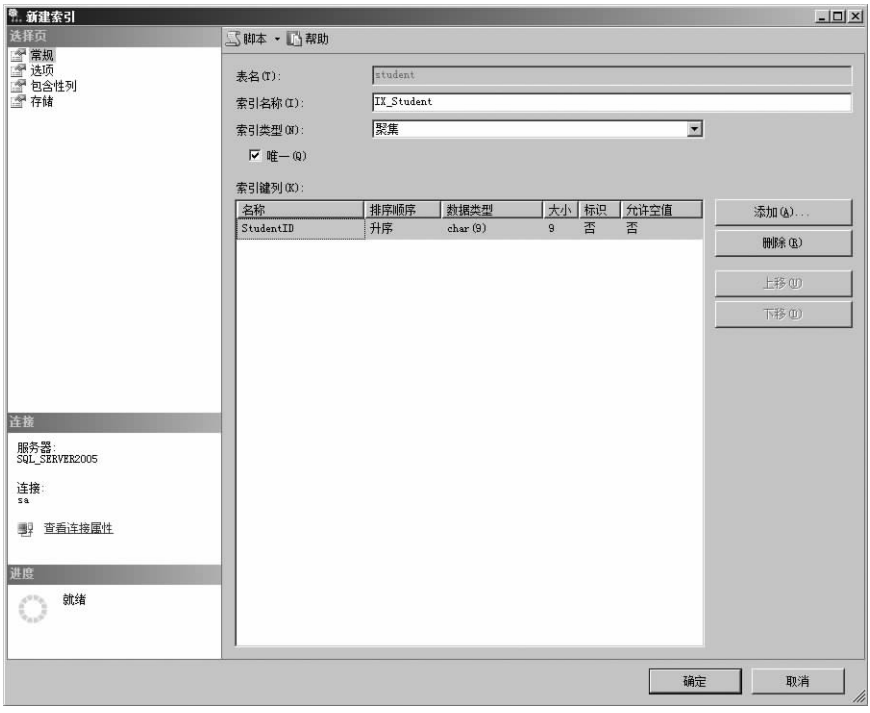

图 4-26　“新建索引”窗口

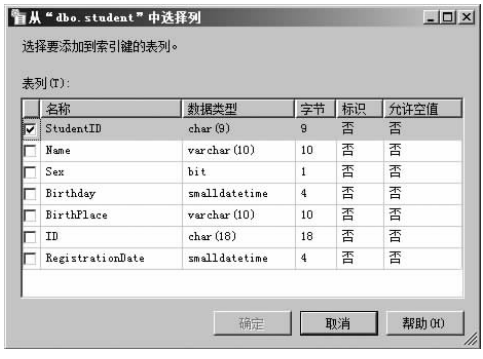

图 4-27　“从 dbo.student 中选择列”窗口

(3)在“表设计器”菜单中,选择“索引/键”命令,打开“索引/键”窗口,见图 4-14。

(4)单击“添加”按钮,则添加一个索引,在属性编辑框中,设置“类型”为“索引”、“列”指定为 ID 字段、是“唯一的”选择“是”、“创建为聚集的”选择“否”、名称改为 IX_student_ID 即可。

【例 4-47】 使用 SQL Server Management Studio 在 book 表的 Bookname、Publisher、Author 字段上创建唯一非聚集索引 IX_Book。

操作步骤如下:

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中,依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo.book→“索引”,右击,在弹出的快捷菜单中,选择“新建索引”命令,打开“新建索引”窗口。

(3)在“索引名称”文本框中输入 IX_Book,“索引类型”选择“非聚集”,选中“唯一”复选框,单击“添加”按钮,打开“从 bookshelf 中选择列”窗口。

(4)选择 Bookname、Author、Publisher 字段,单击“确定”按钮,再在“新建索引”窗口中单击“确定”按钮即可。

【例 4-48】 使用 SQL Server Management Studio 在 bookshelf 表的 CardID 字段上创建非唯一非聚集索引 IX_Bookshelf,并且包含 ISBN 字段。

操作步骤如下:

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中,依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. bookshelf→“索引”,右击,在弹出的快捷菜单中,选择“新建索引”命令,打开“新建索引”窗口。

(3)在“索引名称”文本框中输入 IX_Bookshelf,“索引类型”选择“非聚集”,选中“唯一”复选框,单击“添加”按钮,打开“从 bookshelf 中选择列”窗口。

(4)选择 CardID 字段,单击“确定”按钮,再在“新建索引”窗口选项页中选择“包含性列”,如图 4-28 所示。

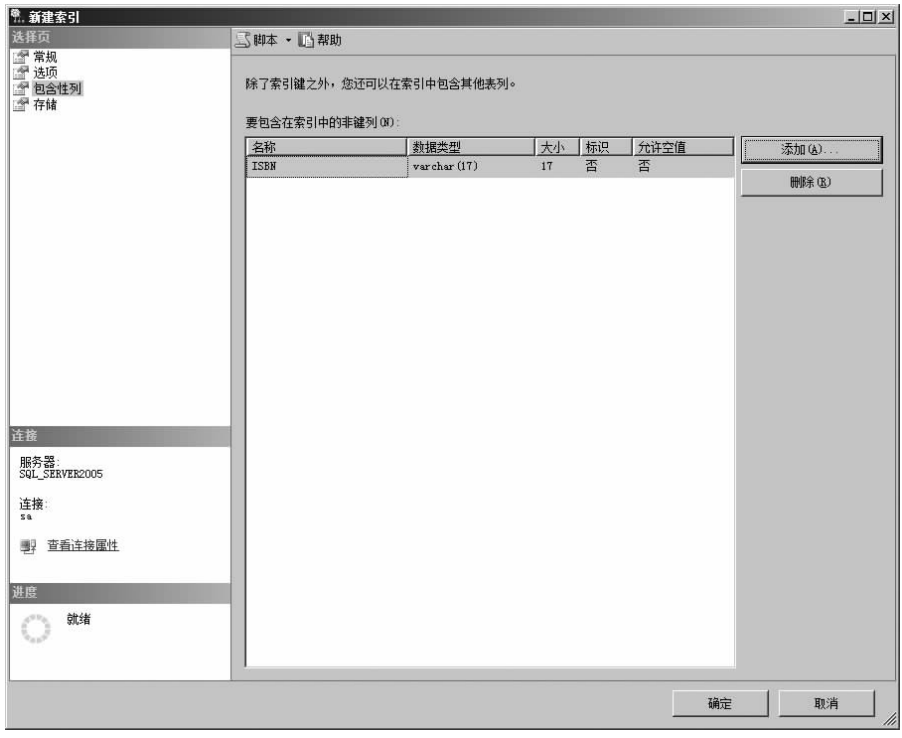

图 4-28 包含性列

(5)单击“添加”按钮,打开“从 bookshelf 中选择列”窗口,从中选择 ISBN 字段,单击“确定”按钮,再在“新建索引”窗口中单击“确定”按钮即可。

2. 使用 Transact-SQL 语句

使用 Transact-SQL 语句创建索引,可以利用 CREATE INDEX 语句创建索引。CREATE INDEX 语句创建索引语法格式如下:

```
CREATE[UNIQUE][CLUSTERED|NONCLUSTERED]INDEX index_name
  ON<table_or_view_name>(column[ASC|DESC][,…n])
  [INCLUDE (column_name[,…n])]
  [WITH (<relational_index_option>[,…n])]
  [ON{partition_scheme_name (column_name)|filegroup_name|default}]
```

该命令中各参数说明如下：

- UNIQUE：为表或视图创建唯一索引(不允许存在索引值相同的两行)。在创建索引时，如果数据已存在，SQL Server 会检查是否有重复值，如果存在重复的键值，将取消 CREATE INDEX 语句，并返回错误信息。如果存在唯一索引，产生重复键值的 UPDATE 或 INSERT 语句将显示错误信息。视图上的聚集索引必须是 UNIQUE 索引。
- CLUSTERED|NONCLUSTERED：聚集索引或非聚集索引。
- index_name：索引名，必须符合标识符命名规则。
- table_or_view_name：要建立索引的表或视图名称。
- column[ASC|DESC]：创建索引的列。[ASC|DESC]确定索引列按升序或降序排列，默认设置为 ASC。
- INCLUDE (column_name[,…n])：指定要添加到非聚集索引中的非索引列。列名不可以重复出现在索引列和包含性列中。
- WITH (<relational_index_option>[,…n])：指定索引选项。
- ON{partition_scheme_name (column_name)|filegroup_name|default}：指定索引存储方案，如果指定了 partition_scheme_name，则将对该索引进行分区，并将分区映射到由 partition_scheme_name 指定的文件组；如果指定了 filegroup，则将在命名文件组内创建索引；如果指定了 default 或者根本没有指定 ON，将与表在同一个文件组创建索引。

【例 4-49】 使用 CREATE INDEX 语句在 student 表的 StudentID 字段上创建唯一聚集索引 IX_student。

语句如下：

```
CREATE UNIQUE CLUSTERED INDEX[IX_student]ON[student]
([StudentID]ASC)ON[PRIMARY]
```

【例 4-50】 使用 CREATE INDEX 语句在 student 表的 ID 字段上创建唯一非聚集索引 IX_student_ID。

语句如下：

```
CREATE UNIQUE NONCLUSTERED INDEX[IX_student_ID]ON[student]
([ID]ASC)ON[PRIMARY]
```

【例 4-51】 使用 CREATE INDEX 语句在 book 表的 BookName、Publisher、Author 字段上创建唯一非聚集索引 IX_Book。

语句如下：

```
CREATE UNIQUE NONCLUSTERED INDEX[IX_Book]ON[book]
(
  [BookName]ASC,
```

```
    [Author]ASC,
    [Publisher]ASC
) ON[PRIMARY]
```

【例 4-52】 使用 CREATE INDEX 语句在 bookshelf 表的 CardID 字段上创建非唯一非聚集索引 IX_Bookshelf，并且包含 ISBN 字段。

语句如下：

```
CREATE NONCLUSTERED INDEX[IX_Bookshelf]ON[bookshelf]
([CardID]ASC)INCLUDE ([ISBN]) ON[PRIMARY]
```

4.6.3 修改和删除索引

1. 使用 SQL Server Management Studio 查看、修改和删除索引

【例 4-53】 使用 SQL Server Management Studio 查看在 bookshelf 表上创建的索引。

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. bookshelf→“索引”，即可看到 bookshelf 表上创建的所有索引，如图 4-29 所示。

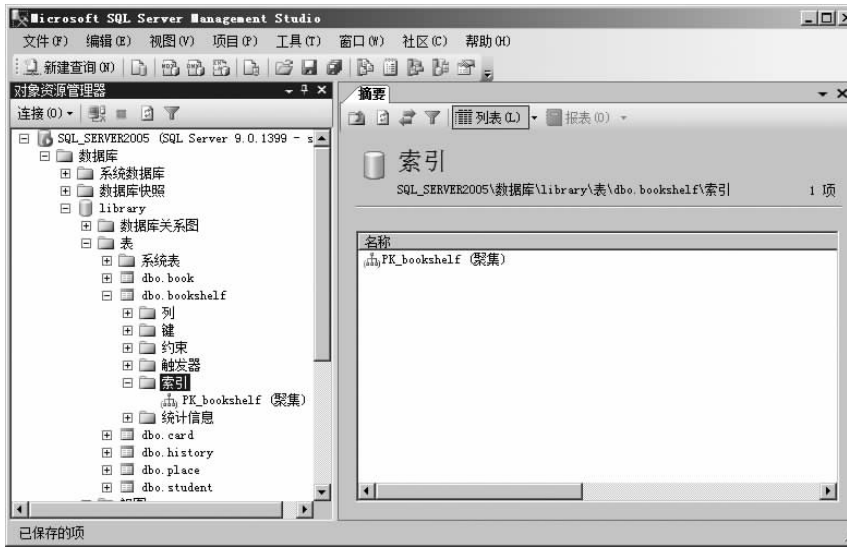

图 4-29 查看索引

索引可以通过两种渠道创建：一种是在创建表时定义 PRIMARY KEY 约束和 UNIQUE 约束间接产生索引；另一种是在表创建完毕后直接创建索引。前者会产生同名的键和索引，例如，在 bookshelf 表上定义 PRIMARY KEY 约束，会产生 PK_bookshelf 键，也会产生 PK_bookshelf 索引。对于前者索引的修改和删除需要通过表设计器窗口，修改表的 PRIMARY KEY 约束和 UNIQUE 约束，间接地修改或删除索引。而对后者的修改，可在“新建索引”窗口中完成。

【例 4-54】 使用 SQL Server Management Studio 修改和删除在 bookshelf 表上创建的 PK_bookshelf 索引。

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中,依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. bookshelf,右击,在弹出的快捷菜单中,选择“修改”命令,打开表设计器窗口。

(3)在“表设计器”菜单中,选择“索引/键”命令,然后选中 PK_bookshelf,即可在属性编辑框中进行修改。

(4)若要删除索引,则只需要选中 PK_bookshelf,然后在“索引/键”对话框中单击“删除”按钮,在删除索引的同时,也删除了对主键的定义。

【例 4-55】 使用 SQL Server Management Studio 修改和删除 student 表上创建的 IX_student。

操作步骤如下:

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中,依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. student→“索引”,右击 IX_student,在弹出的快捷菜单中,选择“属性”命令,打开“索引属性”窗口,如图 4-30 所示。

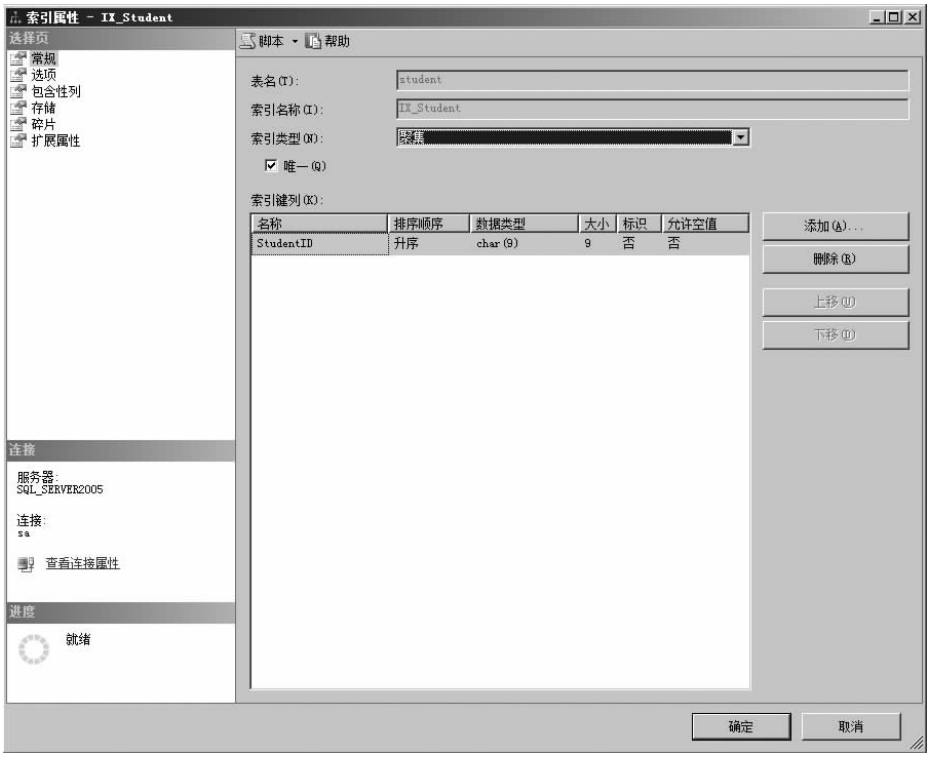

图 4-30　“索引属性”窗口

(3)在“索引属性”窗口中即可修改索引属性。

(4)若要删除 IX_student 索引,在第(2)步弹出的快捷菜单中,选择“删除”命令即可。

2. 禁用索引和重新生成索引

禁用表的聚集索引可以防止对数据的访问,数据仍保留在表中,但在删除或重新生成索引之前,无法对表中的数据进行操作。禁用非聚集索引后,将删除索引数据记录,但索引定义仍存在。

【例 4-56】 使用 SQL Server Management Studio 禁用 student 表上创建的 IX_student

索引。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. student→“索引”，右击 IX_student，在弹出的快捷菜单中，选择“禁用”命令。

【例 4-57】 使用 SQL Server Management Studio 重新生成 student 表上的 IX_student 索引。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. student→“索引”，右击 IX_student，在弹出的快捷菜单中，选择“重新生成”命令。

4.7 管理表数据

数据表创建完毕后，就需要向表中添加数据。可以使用下列工具来访问和更改 SQL Server 2005 实例数据库中的数据：SQL Server Management Studio、sqlcmd 实用工具、bcp 实用工具。本节讲解利用 SQL Server Management Studio 管理数据库中的数据。

【例 4-58】 使用 SQL Server Management Studio 查看 student 表中的数据。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中，依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. student，右击，在弹出的快捷菜单中，选择“打开表”命令，如图 4-31 所示。

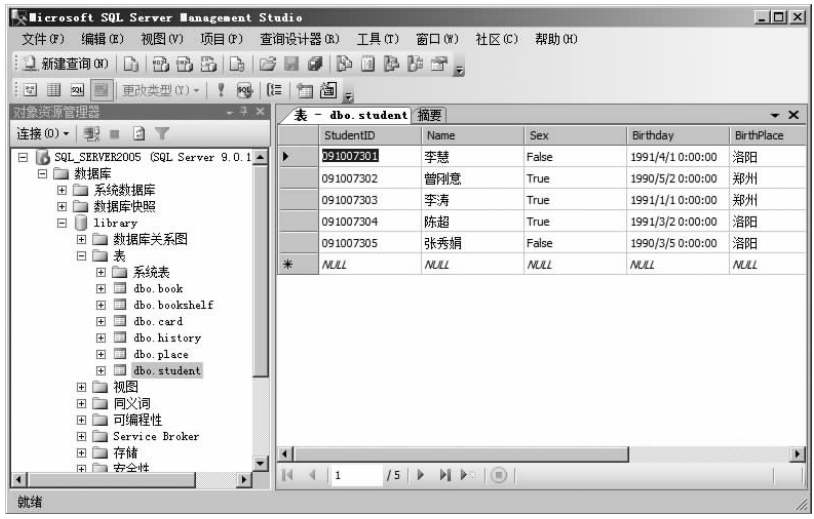

图 4-31 查看 student 表中的数据

【例 4-59】 使用 SQL Server Management Studio 删除 student 表中的记录。

操作步骤如下：

(1)启动 SQL Server Management Studio。

(2)在“对象资源管理器”中,依次展开 SQL_SERVER2005→“数据库”→library→“表”→dbo. student,右击,在弹出的快捷菜单中,选择“打开表”命令。

(3)选中要删除的行,右击,在弹出的快捷菜单中,选择“删除”命令。

提示:选中行可以每次单选一行;也可以按住 Ctrl 键,选中不连续的行;也可以按住 Shift 键,选中连续的行。

实　　训

【实训目的】

(1)使学生掌握数据库表创建过程。

(2)使学生掌握在表上创建索引的过程。

【实训内容】

(1)采用 Microsoft SQL Server Management Studio 和 Transact-SQL 语句两种方法在 library 数据库中创建员工信息表(employee),其结构如表 4-10 所示。

表 4-10　employee 表

列　　名	数据类型	允 许 空
EmployeeID	char(9)	否
Name	varchar(10)	否
Sex	bit	否
Birthday	smalldatetime	否
Birthplace	varchar(10)	否
ID	char(18)	否

(2)采用 SQL Server Management Studio 和 ALTER TABLE 语句两种方法将EmployeeID 字段定义为表的主键。

(3)采用 SQL Server Management Studio 和 ALTER TABLE 语句两种方法在 ID 字段上定义唯一约束。

【实训总结】

结合自己操作的实际情况,认真撰写实训总结。

本 章 小 结

本章主要学习了 SQL Server 2005 中表的知识,包括表结构的定义、数据完整性设计、关系设计、索引设计、管理表中的数据等内容。学习的重点是表的创建,学习的难点是结合数据完整性要求,恰当地在表中使用 PRIMARY KEY 约束、UNIQUE 约束、FOREIGN KEY 约束、CHECK 约束、默认值定义、设置字段值是否允许为空等功能。表是数据库数据存储

之所在，在实际数据库创建过程中是必不可少的环节，具有基础性地位，只有完成本章的学习，才可以开展后续章节的学习。

习 题 4

1. 简述 SQL Server 2005 所支持的系统数据类型。
2. 根据图书的属性，设计一张图书表，用来存储图书信息。
3. 简述 SQL Server 2005 可以实现的完整性约束种类。
4. 在 library 数据库中，定义各表的主键，并叙述创建主键的过程。
5. 在 library 数据库中，为表 card 和表 bookshelf 定义外键，并叙述创建外键的过程。
6. 在 library 数据库中，为表 card 的 State、Maximum、BorrowedNum 字段定义默认值。
7. 简述 SQL Server 2005 创建数据库关系图的过程。
8. 简述 SQL Server 2005 可以创建的索引类型。
9. 简述在 library 数据库 bookshelf 表的 ISBN 字段上创建非唯一非聚集索引的过程。