# 第5章 数据库和表的管理

**【本章导读】**

管理 SQL Server 的数据库和表有两种方式：一是使用 SSMS，二是使用 T-SQL 语句。本章主要讲述 SQL Server 数据库和表的结构、创建和管理，SQL Server 2019 提供的数据类型以及数据的更新和导入。

**【本章要点】**

- 数据库的基本结构。
- 数据库的创建和管理。
- 表的结构。
- SQL Server 2019 的系统数据类型。
- 表的创建和管理。
- 数据的插入、修改和删除。
- 数据导入。

## 5.1 数据库的创建


视频讲解

### 5.1.1 数据库的结构

**1. 数据库文件和文件组**

SQL Server 2019 用文件来存放数据库，即将数据库映射到操作系统文件上。数据库文件有以下 3 类。

(1) 主数据文件(Primary Database File，也称主文件)：主要用来存储数据库的启动信息、部分或全部数据，是数据库的关键文件。主数据文件是数据库的起点，包含指向数据库中其他文件的指针。每个数据库都有一个主数据文件。主数据文件的推荐文件扩展名是.mdf。

(2) 次要数据文件(Secondary Database File，也称辅助数据文件)：除主数据文件以外的所有其他数据文件都是次要数据文件。用于存储主数据文件中未存储的剩余数据和数据库对象。一个数据库可以没有，也可以有多个次要数据文件。次要数据文件的推荐文件扩展名是.ndf。

(3) 事务日志文件：简称日志文件，存放用来恢复数据库所需的事务日志信息，每个数据库必须有一个或多个日志文件。事务日志的推荐文件扩展名是.ldf。

SQL Server 2019 不强制使用.mdf、.ndf 和.ldf 文件扩展名,但使用它们有助于标识文件的各种类型和用途。

SQL Server 2019 中的文件通常有两个名称:逻辑文件名和物理文件名。逻辑文件名是在所有 T-SQL 语句中引用物理文件时所使用的名称。逻辑文件名与物理文件名一一对应,其对应关系由 SQL Server 系统维护。逻辑文件名必须符合 SQL Server 的标识符命名规则,而且在数据库中的逻辑文件名中必须是唯一的。物理文件名是包括目录路径的文件名。它必须符合操作系统文件的命名规则。

一般情况下,一个简单的数据库可以只有一个主数据文件和一个日志文件。如果数据库很大,则可以设置多个次要数据文件和多个日志文件,并将它们放在不同的磁盘上,以提高数据存取和处理的效率。

为便于分配和管理,可以将数据库对象和文件一起分成文件组,对它们整体进行管理。例如,可以将 3 个数据文件(data1.ndf、data2.ndf 和 data3.ndf)分别创建在 3 个磁盘上,这 3 个文件共同组成文件组 fgroup1。在创建表时,就可以指定一个表创建在文件组 fgroup1 上。这样该表的数据就可以分布在 3 个盘上,在对该表执行查询时,可以并行操作,从而有效地提高查询效率。

SQL Server 2019 中提供了两种类型的文件组:主文件组和用户定义文件组。

主文件组包含主数据文件和任何没有明确分配给其他文件组的数据文件。用户定义文件组是在 CREATE DATABASE 或 ALTER DATABASE 语句中使用 FILEGROUP 关键字指定的任何文件组。

一个文件组可以包含多个文件,但是一个文件只能属于一个文件组。每个数据库中均有一个文件组被指定为默认文件组。如果创建表或索引时未指定文件组,则将其分配到默认文件组。一次只能有一个文件组作为默认文件组。db_owner 固定数据库角色成员可以将默认文件组从一个文件组切换到另一个文件组。如果没有指定默认文件组,则将主文件组作为默认文件组。

日志空间与数据空间要分开管理,所以日志文件不包括在文件组内。

综上所述,SQL Server 的数据文件和文件组必须遵循以下规则。

(1) 一个文件和文件组只能被一个数据库使用。

(2) 一个文件只能属于一个文件组。

(3) 日志文件不能属于文件组。

用户可以指定数据库文件的存放位置,如果用户不指定,则数据库文件将被存放在系统的默认存储路径上。SQL Server 2019 的默认存储路径是 C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQLDATA。如果多个 SQL Server 实例在一台计算机上运行,则每个实例都会用不同的默认路径来保存在该实例中创建的数据库文件。

**2. 数据库对象**

SQL Server 2019 数据库中的数据在逻辑上被组织成一系列对象,当一个用户连接到数据库后,他所看到的是这些逻辑对象,而不是物理的数据库文件。

SQL Server 2019 中有以下数据库对象:表(Table)、视图(View)、存储过程 (Stored Procedures)、触发器(Triggers)、用户定义数据类型(User-defined Data Types)、用户自定义函数(User-defined Functions)、索引(Indexes)、规则 (Rules)、默认值(Defaults)等。

在 SQL Server 2019 中创建的每个对象都必须有一个唯一的完全限定对象名，即对象的全名。完全限定对象名由 4 个标识符组成：服务器名、数据库名、所有者名和对象名，各个部分之间由句点“.”连接。格式如下：

服务器名.数据库名.所有者名.对象名

```
server.database.Owner.object
```

使用当前数据库内的对象可以省略完全限定对象名的某部分，省略的部分系统将使用默认值或当前值，例如：

```
server.database..object          /*省略所有者名称 */
server..owner.object             /*省略数据库名称 */
database.owner.object            /*省略服务器名称 */
server…object                    /*省略数据库及所有者名称 */
owner.object                     /*省略服务器及数据库名称 */
object                           /*省略服务器、数据库及所有者名称 */
```

## 5.1.2　系统数据库

在没创建任何数据库之前，依次打开 SSMS 中“对象资源管理器”对话框中的“服务器”→“数据库”→“系统数据库”文件夹，可以看到 4 个系统数据库，如图 5-1 所示。


图 5-1　系统数据库

SQL Server 2019 的系统数据库分别是 master 数据库、tempdb 数据库、model 数据库和 msdb 数据库。

### 1. master 数据库

master 数据库记录 SQL Server 的所有系统级信息。包括实例范围内的元数据（如登录账户）、端点、连接服务器和系统配置设置。master 数据库也记录了所有其他数据库是否存在以及它们的数据库文件位置。另外，master 数据库还记录了 SQL Server 的初始化信息。因此，如果 master 数据库不可用，则 SQL Server 将无法启动。

### 2. tempdb 数据库

tempdb 数据库是连接到 SQL Server 实例的所有用户都可用的全局资源，它保存了所有临时表和临时存储过程。另外，它还用来满足所有其他临时存储的要求，如存储 SQL Server 生成的临时工作表。

每次启动 SQL Server 时，都要重新创建 tempdb，以便系统启动时，该数据库总是空的。在断开连接时，系统会自动删除临时表和存储过程，并且在系统关闭后没有活动连接。因此 tempdb 中不会有什么内容从一个 SQL Server 会话保存到另一个会话。

### 3. model 数据库

model 数据库是在 SQL Server 实例上创建的所有数据库的模板。因为每次启动 SQL Server 时都会创建 tempdb，所以 model 数据库必须始终存在于 SQL Server 系统中。model

数据库相当于一个模子,所有在系统中创建的新数据库的内容,在刚创建时都和 model 数据库完全一样。

**4. msdb 数据库**

msdb 数据库由 SQL Server 代理(SQL Server Agent)来计划警报和作业。

## 5.1.3 创建数据库

**1. 使用 SSMS 创建数据库**

在 SSMS 中创建数据库可以按照下列步骤来操作。

(1) 打开 SSMS,连接到相应的服务器。在“对象资源管理器”中,逐个展开将被使用的“服务器”→“数据库”,右击数据库,在弹出的快捷菜单中选择“新建数据库”命令,如图 5-2 所示。

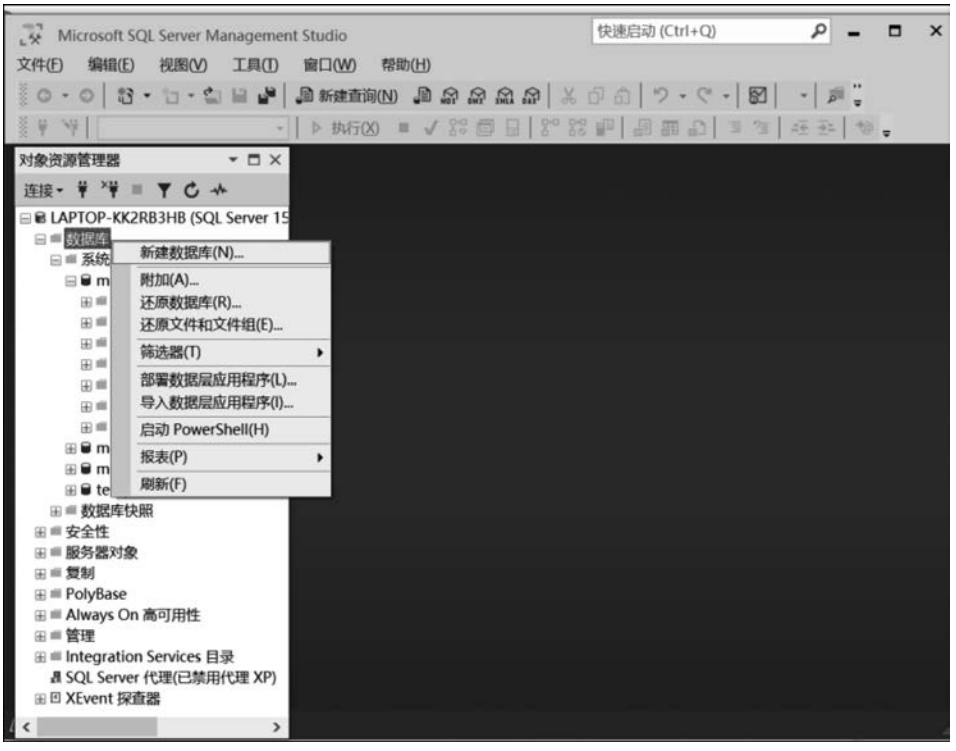


图 5-2 创建新数据库

(2) 在出现的如图 5-3 所示的“新建数据库”窗口中,左侧“选择页”包括“常规”“选项”和“文件组”3 项。默认显示的是“常规”选项,可以在该选项卡中设置新建数据库的名称、数据库的所有者、数据文件、事务日志文件等信息。在“数据库名称”文本框内输入“Study”,就建立了“Study”数据库,同时系统为数据库设置了两个必需的文件。

要添加一个数据库文件,可以在“新建数据库”窗口中单击“添加”按钮。现在为新的数据库文件设置“逻辑名称”为 study_log1,“文件类型”修改为“日志”类型,如图 5-4 所示。

再为数据库添加一个数据库文件 study_data1,如图 5-5 所示。可以为数据文件 study_data1 设置所在的文件组,单击“文件组”选项,可以显示本数据库所包含的所有文件组,因

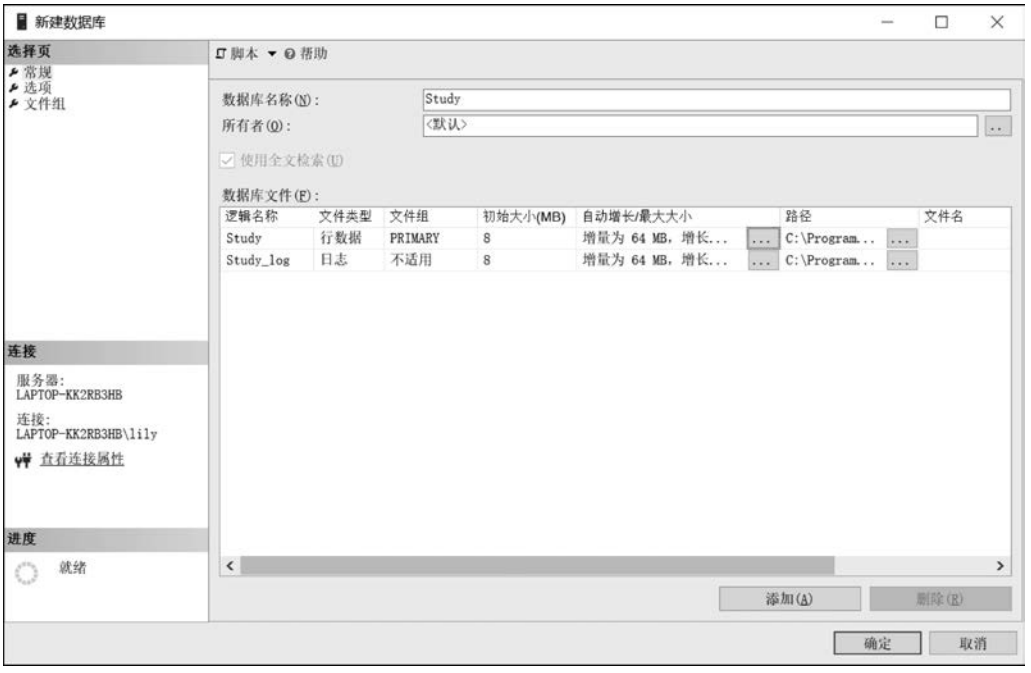


图 5-3　“新建数据库”对话框

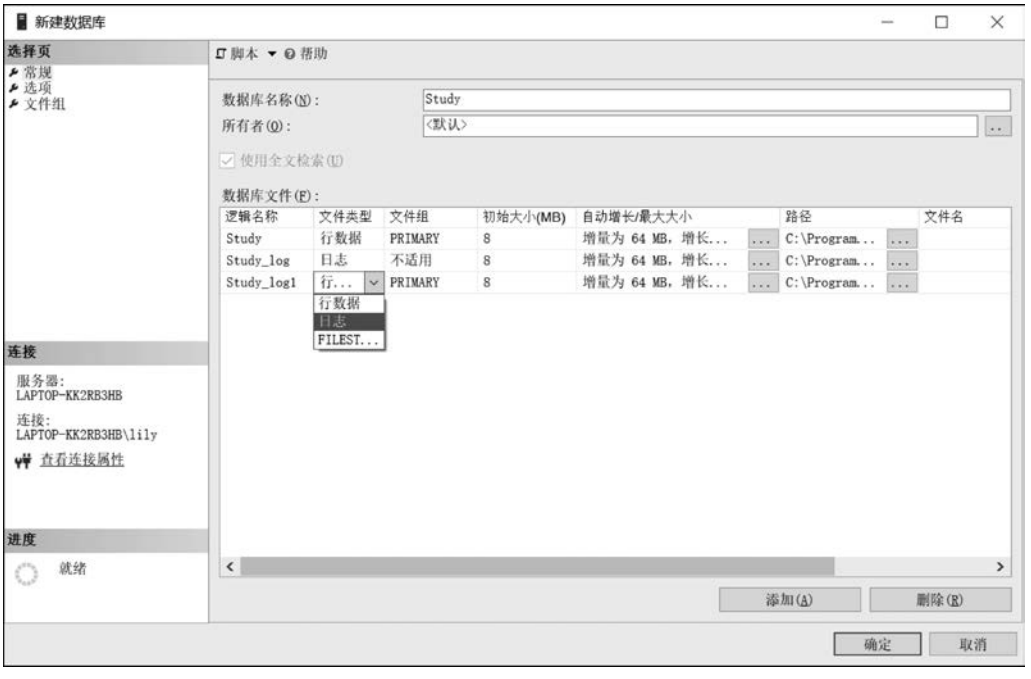


图 5-4　添加数据库文件-1

为是新建数据库，所以在这里只有两项。如果选择“<新文件组>”，将打开“Study 的新建文件组”对话框，可以为本数据库添加一个新的文件组，如图 5-6 所示。

要删除一个数据库文件，选中该文件，单击“删除”按钮即可。

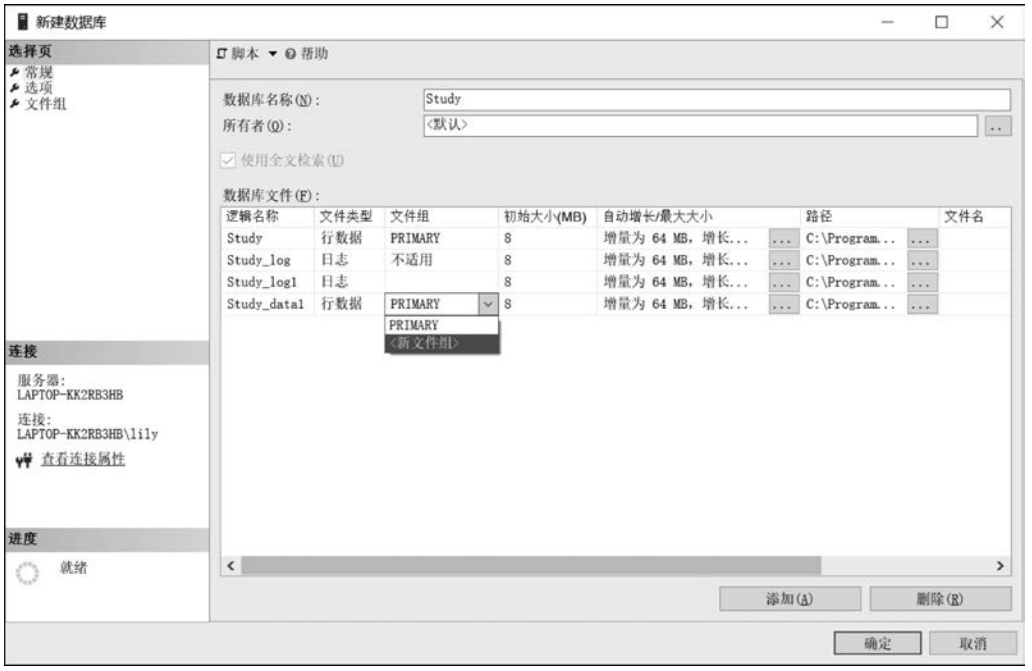


图 5-5　添加数据库文件-2

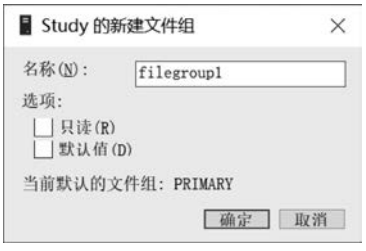


图 5-6　新建文件组

对于数据库文件的设置包括以下内容。

① 逻辑名称：标识一个数据文件。

② 文件类型：标识此文件的类型，包括行数据和日志两种类型。

③ 文件组：标识该文件所在的文件组名称，默认为 PRIMARY。

④ 初始大小：设置文件的初始大小。

⑤ 自动增长/最大大小：设置文件的自动增长方式和最大文件大小，可以通过单击其后的“…”来设置具体选项，如图 5-7 所示。

⑥ 路径：表示该文件所在的完整路径。

⑦ 文件名：数据库文件的物理文件名，即其在操作系统下的名称。

(3) 在“新建数据库”对话框左侧的“选择页”中单击“选项”，在出现的选项页面中可以设置数据库的选项信息，如排序规则、恢复模式、兼容性级别、包含类型和其他选项，如图 5-8 所示。

(4) 在“新建数据库”对话框左侧的“选择页”中单击“文件组”选项，可以显示当前数据

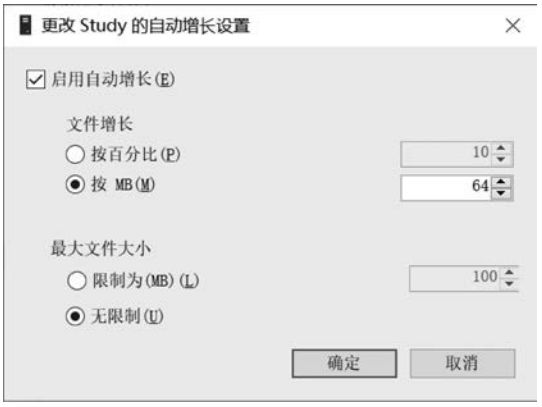


图 5-7　“更改 Study 的自动增长设置”对话框


图 5-8　设置数据库选项

库中的所有文件组信息，如图 5-9 所示。可以在此页查看或修改文件组的信息，也可以单击“添加文件组”按钮新建文件组。单击“删除”按钮，删除一个选中的文件组。当一个文件组被删除后，包含在其中的数据文件会自动添加到默认的文件组。需要注意的是，PRIMARY 文件组是不能被删除的。

（5）当设置完需要的信息后，单击“确定”按钮，系统会根据用户设置的信息完成数据库的建立。

在 SSMS 的“对象资源管理器”中，会出现新的数据库 Study，如图 5-10 所示。

可以根据用户设置的数据库文件存储路径找到创建的数据库文件。默认情况下，在本机的 C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQLDATA 下生成物理数据库文件，如图 5-11 所示。

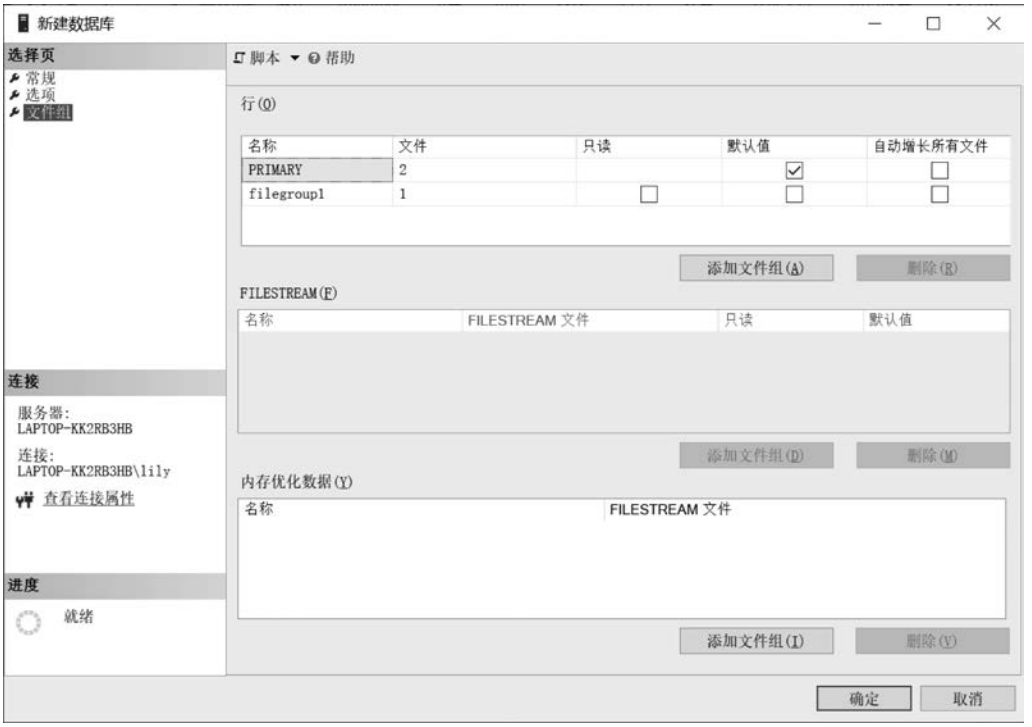


图 5-9　设置文件组

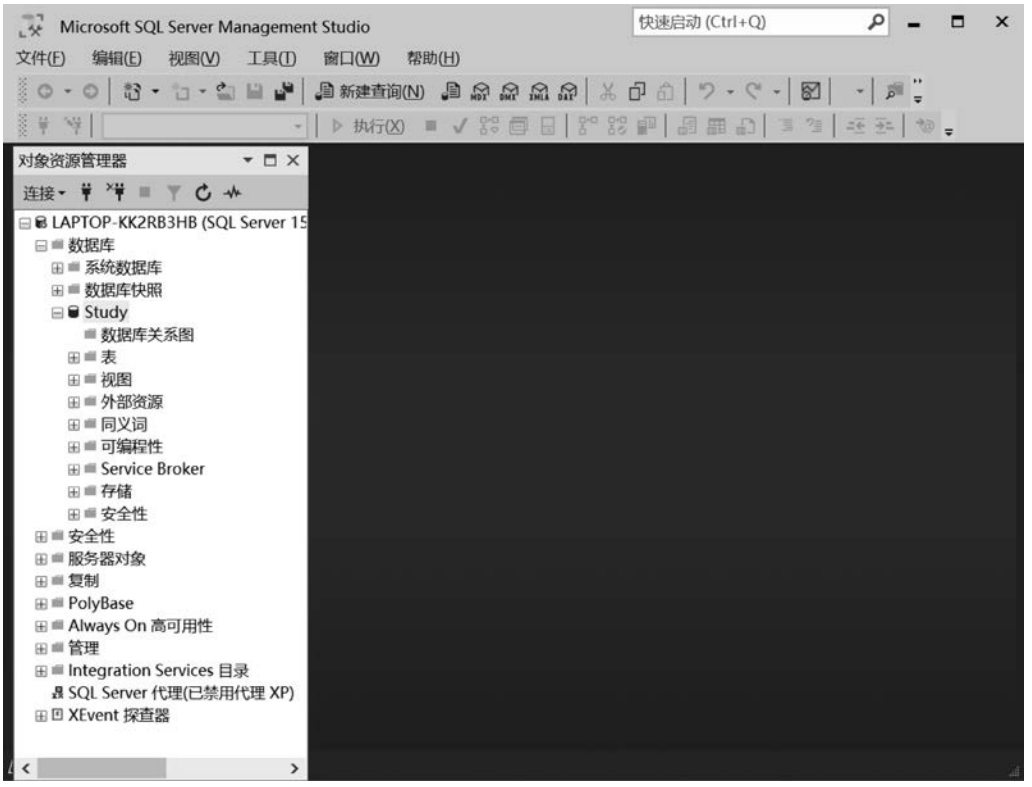


图 5-10　查看新建的数据库

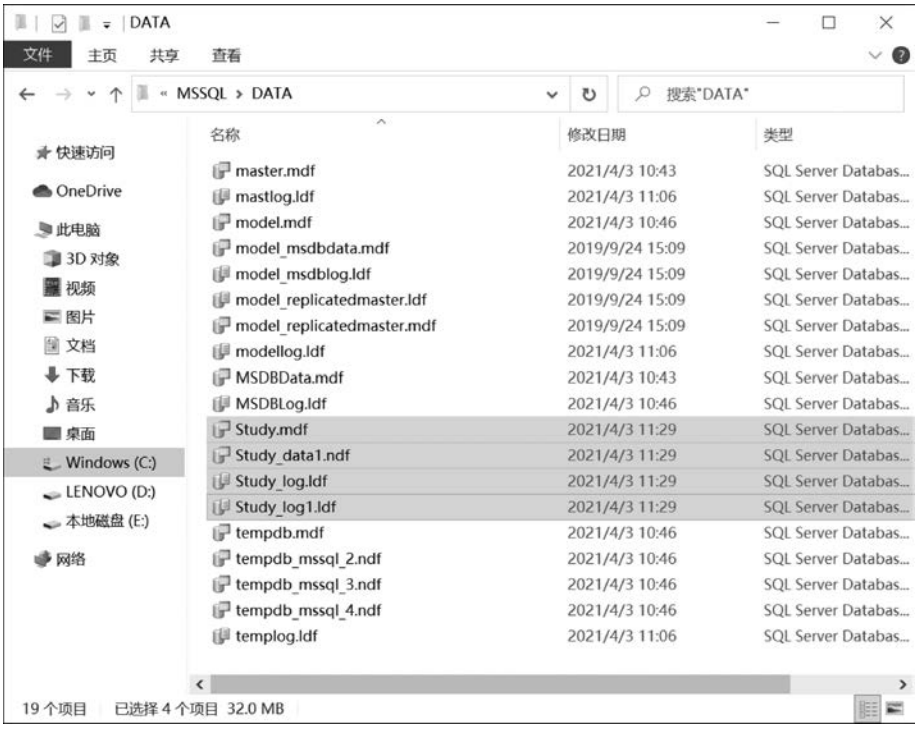


图 5-11　数据库文件及路径

### 2. 使用 T-SQL 语句创建数据库

T-SQL 提供了数据库创建语句 CREATE DATABASE。其语法格式如下。

```
CREATE DATABASE database_name              /* 指定数据库逻辑名称 */
    [ON
    [< filespec >[, …n]]
    [,< filegroup >[, …n]]]                /* 指定数据文件及文件组属性 */
    [LOG ON  {< filespec >[, …n]}]         /* 指定日志文件属性 */
    [COLLATE < collation_name >]           /* 指定数据库的默认排序规则 */
    [FOR LOAD|FOR ATTACH]                  /* FOR LOAD 从一个备份数据库向新建数据库加载数据;
FOR ATTACH 从已有的数据文件向数据库添加数据 */
```

**说明：**

(1) database_name 是所创建的数据库的逻辑名称。数据库名称在当前服务器中必须唯一，且符合标识符的命名规则，最多可以包含 128 个字符。

(2) ON 子句用于指定数据文件及文件组属性，具体属性值在< filespec >中指定。< filespec >格式如下。

```
< filespec >: : =   [PRIMARY]
                    (NAME = '逻辑文件名',
                     FILENAME = '存放数据库的物理路径和文件名'
                     [, SIZE = 数据文件的初始大小]
                     [, MAXSIZE = 指定文件的最大大小]
                     [, FILEGROWTH = 指出文件每次的增量])
```

<filegroup>项用以定义用户文件组及其文件。<filegroup>格式如下：

```
<filegroup>::= FILEGROUP 文件组名
```

(3) LOG ON 子句用于指定事务日志文件的属性，具体属性值在<filespec>中指定。

如果在定义时没有指定 ON 子句和 LOG ON 子句，系统将采用默认设置，自动生成一个主数据文件和一个事务日志文件，并将文件存储在系统默认路径上。

**【例 5-1】** 创建一个名为 BookSys 的数据库。

```
CREATE DATABASE BookSys
```

由于没有指定数据库文件名，在默认的情况下，命名主数据文件为 BookSys.mdf，事务日志文件名为 BookSys_log.log。同时由于按复制 model 数据库的方式来创建新的数据库，主数据文件和事务日志文件的大小都与 model 数据库的主数据文件和事务日志文件大小一致，并且可以自由增长。

**【例 5-2】** 创建一个名为 KEJI_DB 的数据库。要求有 3 个文件，其中，主数据文件为 10MB，最大大小为 50MB，每次增长 20%；辅助数据文件属于文件组 Fgroup，文件为 10MB，大小不受限制，每次增长 10%；事务日志文件大小为 20MB，最大大小为 100MB，每次增长 10MB。文件存储在 c:\db 路径下。

```
CREATE DATABASE  KEJI_DB                              /*数据库名*/
    ON PRIMARY                                        /*主文件组*/
    (NAME = 'KEJI_DB_Data1',                          /*主文件逻辑名称*/
    FILENAME = 'c:\db\KEJI_DB_Data1.mdf',             /*主文件物理名称*/
    SIZE = 10mb,
    MAXSIZE = 50mb,
    FILEGROWTH = 20 % ),
    FILEGROUP Fgroup                                  /*文件组*/
    (NAME = 'KEJI_DB_Data2',                          /*主文件逻辑名称*/
    FILENAME = 'c:\db\ KEJI_DB_Data2.ndf',            /*主文件物理名称*/
    MAXSIZE = UNLIMITED,                              /*增长不受限制*/
    SIZE = 10Mb,
    FILEGROWTH = 10mb)
    LOG ON
    (NAME = 'KEJI_DB_Log',                            /*日志文件逻辑名称*/
    FILENAME = 'c:\db\ KEJI_DB_Log.ldf',              /*日志文件物理名称*/
    SIZE = 20mb,
    MAXSIZE = 100mb,
    FILEGROWTH = 10mb)
```

**说明**：KEJI_DB 数据库的物理文件创建在 c:\db 路径下，在运行此语句前要首先确定此路径是存在的。

## 5.1.4 查看数据库信息

### 1. 使用 SSMS 查看数据库信息

在 SSMS 的“对象资源管理器”中，展开“服务器”→“数据库”，右击数据库 Study，在弹

出的快捷菜单中选择“属性”命令，打开如图 5-12 所示的“数据库属性”对话框来查看数据库的信息。

“数据库属性”对话框包含“常规”“文件”“文件组”“选项”“更改跟踪”“权限”“扩展属性”“镜像”“事务日志传送”和“查询存储”10 个选择页。可以通过它们查看、修改数据库的基本属性，在此不再具体说明。

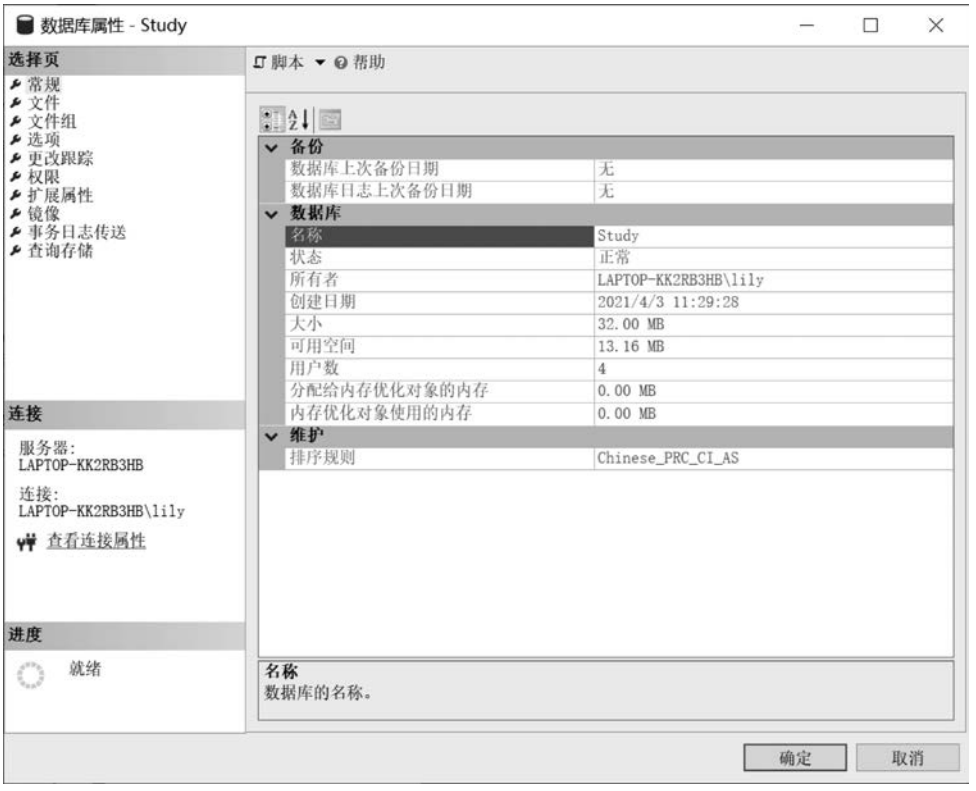


图 5-12　Study 的“数据库属性”对话框

### 2. 使用 T-SQL 语句查看数据库信息

(1) 使用系统存储过程 sp_helpdb 查看数据库信息。其语法格式如下。

```
sp_helpdb  [数据库名]
```

① 不指定数据库参数，将显示服务器中所有数据库的信息，如图 5-13 所示。

② 指定具体数据库参数，将显示指定数据库的信息，如图 5-14 所示。

(2) 使用系统存储过程 sp_databases。其语法格式如下。

```
sp_databases
```

此命令将显示服务器中所有可以使用的数据库的信息，如图 5-15 所示。

(3) 使用系统存储过程 sp_helpfile 查看数据库中文件的信息。其语法格式如下。

```
sp_helpfile [文件名]
```

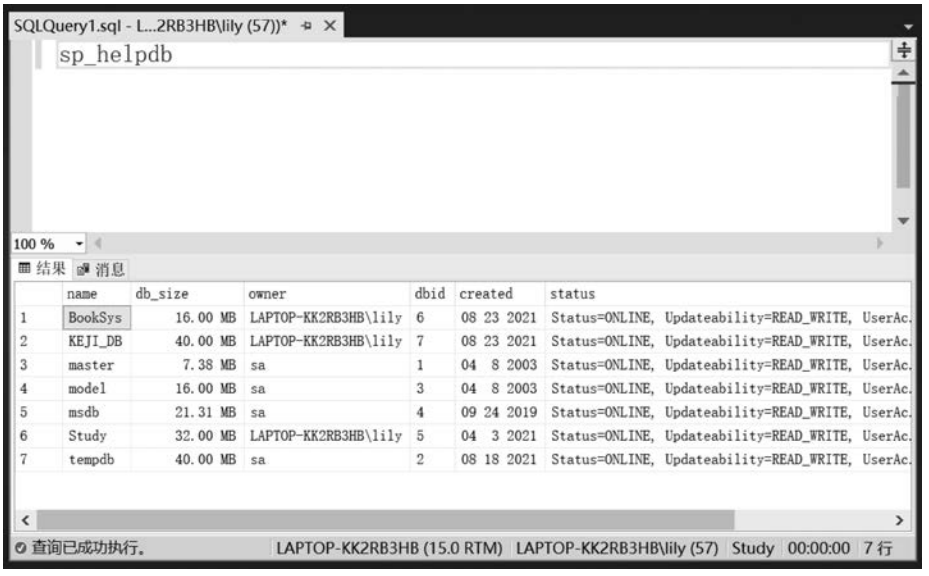


图 5-13　查看服务器中所有数据库的信息

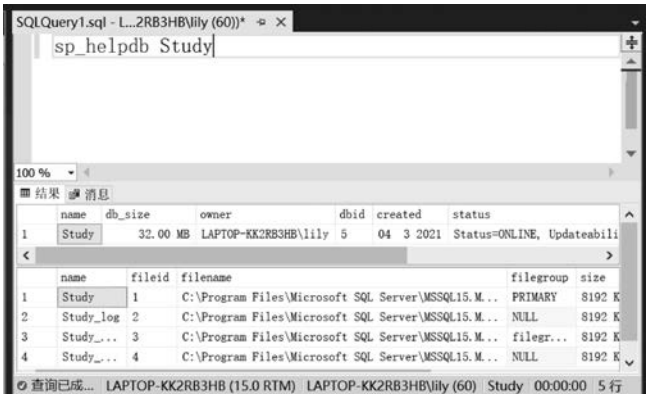


图 5-14　查看 Study 数据库的信息

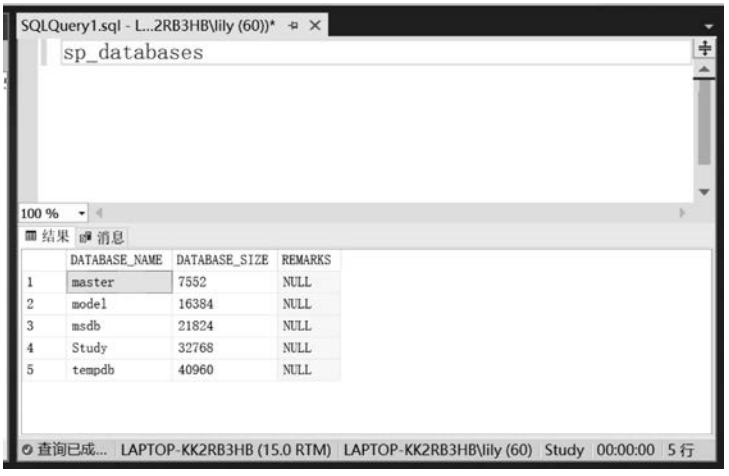


图 5-15　查看服务器中所有可用的数据库的信息

① 不指定文件名参数，将显示当前数据库中所有文件的信息，如图 5-16 所示。

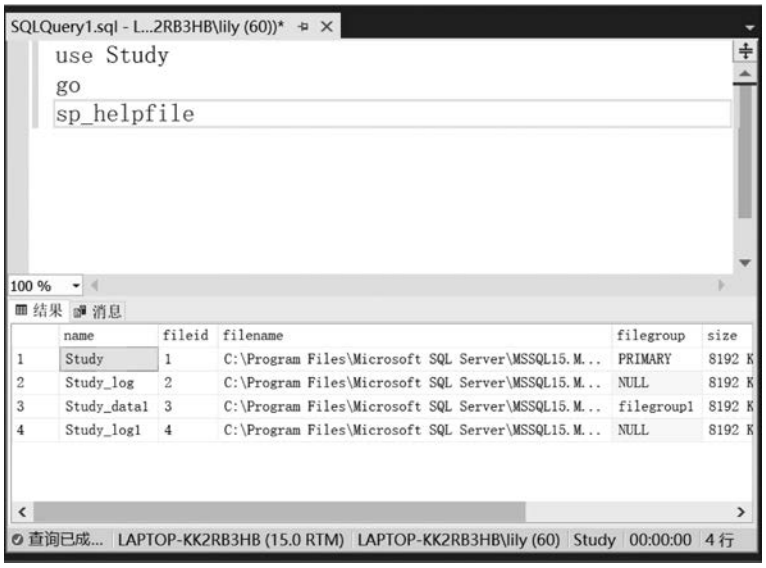


图 5-16　查看 Study 数据库中所有文件的信息

② 指定具体文件名参数，将显示数据库中指定文件的信息，如图 5-17 所示。

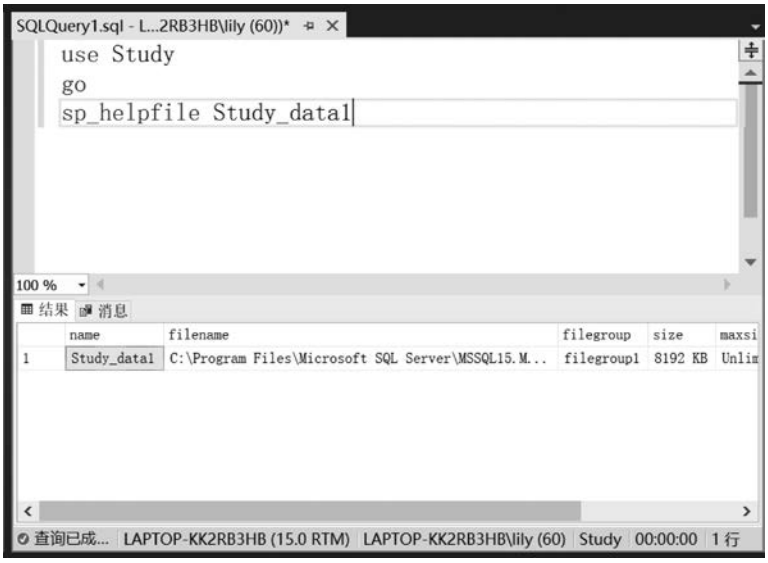


图 5-17　查看 Study 数据库中 study_data1 文件的信息

(4) 使用系统存储过程 sp_helpfilegroup 查看文件组信息。其语法格式如下。

```
sp_helpfilegroup [文件组名]
```

① 不指定文件组名参数，将显示数据库中所有文件组的信息。
② 指定具体文件组名参数，将显示数据库中指定文件组的信息。
其用法同 sp_helpfile。

### 5.1.5 修改数据库

修改数据库包括增减数据库文件、修改文件属性(包括更改文件名和文件大小等)、修改数据库选项等。

**1. 使用 SSMS 修改数据库**

在 SSMS 的“对象资源管理器”中,展开“服务器”→“数据库”,右击要修改的数据库如 Study,在弹出的快捷菜单中选择“属性”命令,打开“数据库属性”对话框来修改数据库的信息。

(1) 增减数据库文件和文件组。用户可以使用“文件”选项增减数据库文件或修改数据库文件属性。可以使用“文件组”选项增加或删除一个文件组,修改现有文件组的属性。具体操作在新建数据库时已经涉及,在此不再赘述。

(2) 修改数据库选项。使用“选项”选项可以修改数据库的选项。只需在图 5-18 中单击要修改的属性值后的下拉列表按钮,选择 True 或 False,就可以非常容易地更改当前数据库的选项值。


图 5-18 使用“选项”选项卡修改数据库选项

**2. 使用 T-SQL 语句修改数据库**

T-SQL 提供了数据库修改语句 ALTER DATABASE。其基本语法格式如下。

```
ALTER DATABASE database_name                    /* 指定要修改的数据库的名称 */
{ADD FILE < filespec >[, …n][TO FILEGROUP filegroup_name]
                                                /* 在文件组中增加数据文件 */
|ADD LOG FILE < filespec >[, …n]                /* 增加事务日志文件 */
```

```
|REMOVE FILE logical_file_name                    /* 删除数据文件 */
|ADD FILEGROUP filegroup_name                     /* 增加文件组 */
|REMOVE FILEGROUP filegroup_name                  /* 删除文件组 */
|MODIFY FILE <filespec>                           /* 修改文件属性 */
|MODIFY NAME = new_dbname                         /* 更新数据库名称 */
        }
```

**【例 5-3】** 为 KEJI_DB 数据库增加一个数据文件 KEJI_DB_Data3，物理文件名为 KEJI_ DB_Data3.ndf，初始大小为 5MB，最大大小为 50MB，每次扩展 1MB。

```
ALTER DATABASE KEJI_DB
ADD FILE
(NAME = 'KEJI_DB_Data3',
FILENAME = 'c:\db\KEJI_DB_Data3.ndf',
SIZE = 5MB,
MAXSIZE = 50MB,
FILEGROWTH = 1MB)
```

其中，ADD FILE 是指增加一个数据文件，还可以使用 ADD LOG FILE 增加事务日志文件。

**【例 5-4】** 将 KEJI_DB 数据库的第二个数据文件 KEJI_DB_data2 的初始大小修改为 40MB。

```
ALTER DATABASE KEJI_DB
MODIFY FILE
(NAME = 'KEJI_DB_data2',
SIZE = 40MB)
```

**【例 5-5】** 删除 KEJI_DB 的数据文件 KEJI_DB_Data3。

```
ALTER DATABASE KEJI_DB
REMOVE FILE KEJI_DB_Data3
```

**【例 5-6】** 使用 ALTER DATABASE 语句修改数据库选项，将 Study 数据库的自动缩减选项设置为 True。

```
ALTER DATABASE Study
SET AUTO_SHRINK ON
```

关于 ALTER DATABASE 语句的更详细的用法可以参考 SQL Server 2019 的联机丛书。

### 5.1.6　删除数据库

当不再需要用户定义的数据库，或者已将其移到其他数据库或服务器上时，可以删除该数据库。数据库删除之后，文件及其数据都从服务器的磁盘中删除。一旦删除数据库，它将被永久删除，并且不能进行检索，除非使用以前的备份。所以，删除数据库之前应格外小心。

可以删除数据库，而不管该数据库所处的状态。这些状态包括脱机、只读和可疑。

删除数据库时，应注意以下情况。

(1) 如果数据库涉及日志传送操作，应在删除数据库之前取消日志传送操作。

(2) 若要删除为事务复制发布的数据库,或删除为合并复制发布或订阅的数据库,必须首先从数据库中删除复制。

(3) 如果数据库已损坏,不能删除复制,则可以首先使用 ALTER DATABASE 语句将数据库设置为脱机,然后再删除数据库。

(4) 不能删除系统数据库。

(5) 删除数据库后,应备份 master 数据库,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 以来删除的任何数据库仍将引用这些不存在的数据库。这可能导致产生错误消息。

**1. 使用 SSMS 删除数据库**

在 SSMS 的"对象资源管理器"中找到要删除的数据库,右击选中的数据库,在弹出的快捷菜单中选择"删除"命令,进入如图 5-19 所示的"删除对象"窗口,单击"确定"按钮即可删除数据库。

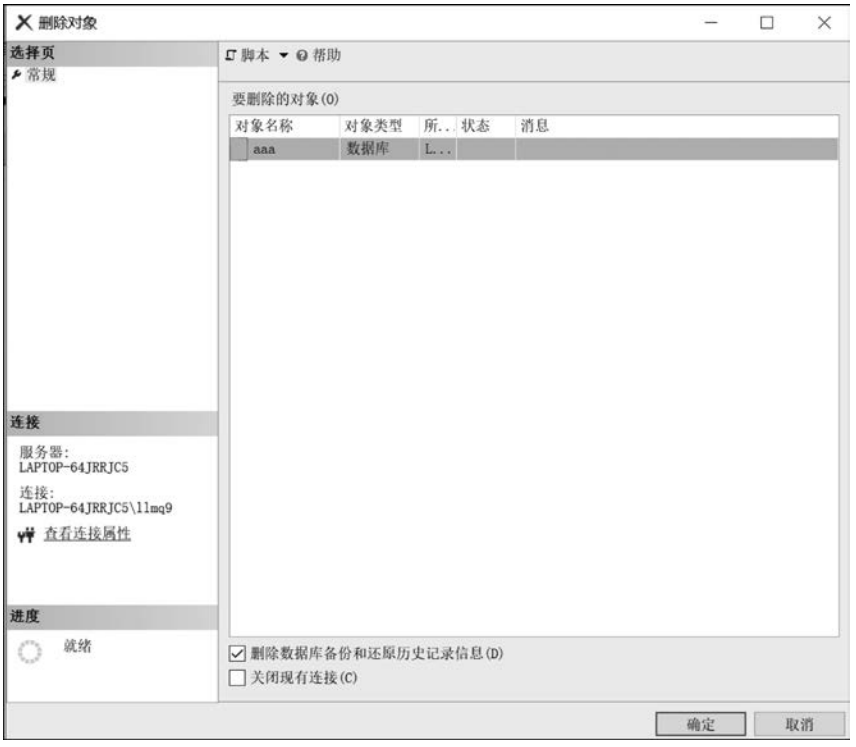


图 5-19 删除数据库

**2. 使用 T-SQL 语句删除数据库**

使用 T-SQL 提供的 DROP DATABASE 语句可以删除数据库,并且可以一次删除多个数据库。其语法格式如下。

```
DROP DATABASE database [,…n]
```

**【例 5-7】** 删除数据库 BookSys。

```
DROP  DATABASE BookSys
```

## 5.2　数据表的创建

表是用来存储数据和操作数据的逻辑结构。关系数据库中的所有数据都存储在表中，因此表是 SQL Server 数据库最重要的组成部分。在介绍表的创建之前，先要介绍 SQL Server 2019 提供的数据类型。


视频讲解

### 5.2.1　数据类型

SQL Server 为了实现 T-SQL 的良好性能，提供了丰富的数据类型。

**1. 数值型数据**

(1) bigint。bigint 型数据可以存放 $-2^{63}\sim2^{63}-1$ 范围内的整型数据。以 bigint 数据类型存储的每个值占用 8B，共 64 位，其中 63 位用于存储数字，1 位用于表示正负。

(2) int。int 也可以写作 integer，可以存储 $-2^{31}\sim2^{31}-1$ 范围内的全部整数。以 int 数据类型存储的每个值占用 4B，共 32 位，其中 31 位用于存储数字，1 位用于表示正负。

(3) smallint。smallint 型数据可以存储 $-2^{15}\sim2^{15}-1$ 范围内的所有整数。以 smallint 数据类型存储的每个值占用 2B，共 16 位，其中 15 位用于存储数字，1 位用于表示正负。

(4) tinyint。tinyint 型数据可以存储 0～255 范围内的所有整数。以 tinyint 数据类型存储的每个值占用 1B。

整数型数据可以在较少的字节里存储较大的精确数字，而且存储结构的效率很高。所以，平时在选用数据类型时，应尽量选用整数型数据类型。

(5) decimal 和 numeric。事实上，numeric 数据类型是 decimal 数据类型的同义词，decimal 可以简写为 Dec。在 T-SQL 中，numeric 与 decimal 数据类型在功能上是等效的。

使用 decimal 和 numeric 型数据可以精确指定小数点两边的总位数 $p$(precision，精度)和小数点右面的位数 $s$(scale，刻度)。

在 SQL Server 中，decimal 和 numeric 型数据的最高精度可以达到 38 位，即 $1\leqslant p\leqslant 38$，$0\leqslant s\leqslant p$。decimal 和 numeric 型数据的刻度的取值范围必须小于精度的最大范围。

SQL Server 分配给 decimal 和 numeric 型数据的存储空间随精度的不同而不同，一般说来，对应的比例关系如下。

| 精度范围 | 分配字节数 |
|---|---|
| 1～9 | 5 |
| 10～19 | 9 |
| 20～28 | 13 |
| 29～38 | 17 |

(6) real 和 float。real 型数据范围为 $-3.40E+38\sim1.79E+38$，存储时使用 4B，精度可以达到 7 位。float 型数据范围为 $-1.79E+38\sim1.79E+38$。利用 float 来表明变量和表列时可以指定用来存储按科学记数法记录的数据尾数的 bit 数。例如 float($n$)，$n$ 的范围是 1～53。当 $n$ 的取值为 1～24 时，float 型数据可以达到的精度是 7 位，用 4B 来存储；当 $n$ 的取值范围为 25～53 时，float 型数据可以达到的精度是 15 位，用 8B 来存储。

### 2. 字符数据类型

SQLServer 提供了 3 类字符数据类型,分别是 char、varchar 和 text。在这 3 类数据类型中,最常用的是 char 和 varchar 两类。

(1) char($n$)。利用 char 数据类型存储数据时,每个字符占用一个字节的存储空间。char 数据类型使用固定长度来存储字符,最长可以容纳 8000 个字符。利用 char 数据类型来定义表列或者变量时,应该给定数据的最大长度 $n$。如果实际数据的字符长度短于给定的最大长度,则多余的字节会以空格填充。如果实际数据的字符长度超过了给定的最大长度,则超过的字符将会被截断。在使用字符型常量为字符数据类型赋值时,必须使用单引号('')将字符型常量引起来。

(2) varchar($n$)。varchar 数据类型的使用方式与 char 数据类型类似。SQL Server 利用 varchar 数据类型来存储最长可以达到 8000 字符的变长字符。与 char 数据类型不同的是,varchar 数据类型的存储空间随存储在表列中的每一个数据的字符数的不同而变化。

例如,定义表列为 varchar(20),那么存储在该列的数据最多可以长达 20B。但是,在数据没有达到 20B 时并不会在多余的字节上填充空格,而是按实际占用的字符长度分配字节。

当存储在列中的数据值大小经常变化时,使用 varchar 数据类型可以有效地节省空间。

(3) text。当要存储的字符型数据非常庞大以至于 8000B 完全不够用时,char 和 varchar 数据类型都失去了作用,这时应该选择 text 数据类型。

text 数据类型专门用于存储庞大的变长字符数据。最大长度可以达到 $2^{31}-1$ 个字符,约 2GB。

**【例 5-8】** 建立一个以字符类型定义表列的表,然后向其中插入一行数据。

创建一个表格:

```
CREATE TABLE  chars_example
(char_1 char(5),
varchar_1 varchar(5),
text_1 text)
go
```

插入一行数据:

```
INSERT INTO chars_example
VALUES('abc', 'abc', 'ddddddddddddddddddddddddddd')
go
```

### 3. 日期/时间数据类型

SQL Server 提供的日期/时间数据类型可以存储日期和时间的组合数据。以日期和时间数据类型存储日期或时间的数据比使用字符型数据更简单,因为 SQL Server 提供了一系列专门处理日期和时间的函数来处理这些数据。如果使用字符型数据来存储日期和时间,只有用户本人可以识别,计算机并不能识别,因而也不能自动将这些数据按照日期和时间来进行处理。

日期/时间数据类型有 datetime 和 smalldatetime 两类。

(1) datetime。datetime 数据类型范围从 1753 年 1 月 1 日到 9999 年 12 月 31 日,可以

精确到千分之一秒。datetime 数据类型的数据占用 8B 的存储空间。

(2) smalldatetime。smalldatetime 数据范围从 1900 年 1 月 1 日到 2079 年 6 月 6 日，可以精确到分。smalldatetime 数据类型占 4B 的存储空间。

SQL Server 在用户没有指定小时以上精度的数据时，会自动设置 datetime 和 smalldatetime 数据的时间为 00:00:00。

在 SQL Server 2019 中，用户可以使用 GETDATE()函数来得到系统时间，使用 SET DATEFORMAT 命令设置日期格式。

**【例 5-9】** 设置日期格式为“月-日-年”。

```
Set DATEFORMAT MDY
```

本例设置的日期格式为“月-日-年”。M 表示月，D 表示日，Y 表示年。

SQL Server 2019 提供了多种日期表达式，其中，年可以是 4 位数或 2 位数，月和日可以用 2 位数或 1 位数。系统默认的日期格式是：年-月-日。常用的日期格式如下。

- 年
- 年月日
- 月-日-年
- 月/日/年
- 年-月-日

**4. 货币数据类型**

货币数据类型专门用于货币数据处理。SQL Server 提供了 money 和 smallmoney 两种货币数据类型。

(1) money。money 数据类型存储的货币值由 2 个 4B 整数构成。前面的一个 4B 表示货币值的整数部分，后面的一个 4B 表示货币值的小数部分。以 money 存储的货币值的范围为$-2^{63}\sim2^{63}-1$，可以精确到万分之一货币单位。

(2) smallmoney。由 smallmoney 数据类型存储的货币值由 2 个 2B 整数构成。前面的一个 2B 表示货币值的整数部分，后面的一个 2B 表示货币值的小数部分。以 smallmoney 存储的货币值的范围为$-2^{31}\sim2^{31}-1$，也可以精确到万分之一货币单位。

在把值加入定义为 money 或 smallmoney 数据类型的表列时，应该在最高位之前放一个货币记号 $ 或其他货币单位的记号，但是也没有严格要求。

**【例 5-10】** 建立一个以货币数据类型定义表列的表，然后向其中插入一行数据。

建立一个表格：

```
CREATE TABLE money_example2
(money_num money,
smallmoney_num smallmoney
)
go
```

插入一行数据：

```
INSERT INTO money_example2
VALUES ( $ 222.222, $ 333.333)
```

**5. 二进制数据类型**

二进制数据是一些用十六进制来表示的数据。例如,十进制数据 245 表示成十六进制数据就应该是 F5。在 SQL Server 中,可以使用 3 种数据类型来存储二进制数据,分别是 binary、varbinary 和 image。

二进制数据类型同字符数据类型非常相似。使用 binary 数据类型定义的列或变量,具有固定的长度,最大长度可以达到 8KB;使用 varbinary 数据类型定义的列或变量具有不固定的长度,其最大长度也不得超过 8KB;image 数据类型可以用于存储字节数超过 8KB 的数据,比如 Microsoft Word 文档、Microsoft Excel 图表及图像数据(包括. GIF、. BMP、. JPEG 文件)等。

一般说来,最好使用 binary 或 varbinary 数据类型来存储二进制数据。只有在数据的字节数超过 8KB 的情况下,才使用 image 数据类型。

在对二进制数据进行插入操作时,必须在数据常量前面增加一个前缀 0x。

**【例 5-11】** 建立一个以二进制数据类型定义表列的表,然后向其中插入一行数据。

建立一个表格:

```
CREATE TABLE binary_example
(bin_1 binary(5),
bin_2 varbinary(5))
go
```

插入一行数据:

```
INSERT INTO binary_example
VALUES (0xaabbccdd,0xaabbccddee)
INSERT INTO binary_example
VALUES (0xaabbccdde,0x)
go
```

**6. 双字节数据类型**

SQL Server 2019 提供的双字节数据类型共有 3 类,分别是 nchar、nvarchar、ntext。

(1) nchar($n$)。nchar($n$)是固定长度的双字节数据类型,括号里的 $n$ 用来定义数据的最大长度。$n$ 的取值范围是 1～4000,所以使用 nchar 数据类型所能存储的最大字符数是 4000 字符。由于存储的都是双字节字符,所以双字节数据的存储空间为:字符数×2(B)。

nchar 数据类型的其他属性及使用方法与 char 数据类型一样。例如,在有多余字节的情况下也会自动加上空格进行填充。

(2) nvarchar($n$)。nvarchar($n$)数据类型存储可变长度的双字节数据类型,括号里的 $n$ 用来定义数据的最大长度。$n$ 的取值为 0～4000。所以使用 nvarchar 数据类型所能存储的最大字符数也是 4000。nvarchar 数据类型的其他属性及使用方法与 varchar 数据类型一样。

(3) ntext。ntext 数据类型存储的是可变长度的双字节字符,ntext 数据类型突破了前两种双字节数据类型不能超过 4000 字符的规定,最多可以存储多达 $2^{30}-1$ 个双字节字符。ntext 数据类型的其他属性及使用方法与 text 数据类型一致。

**7. 图像、文本数据的使用**

为了方便用户存储和使用文本、图像等大型数据,SQL Server 2019 提供了 text、ntext

和 image 三种数据类型。

文本和图像数据在 SQL Server 中是用 text、ntext 和 image 数据类型来表示的。这 3 种数据类型很特殊，因为它们的数据量往往较大，所以它们不像表中其他类型的数据那样一行一行地依次存放在数据页中，而是经常被存储在专门的页中，在数据行的相应位置处只记录指向这些数据实际存储位置的指针。在 SQL Server 2019 以前的版本中，文本和图像数据都是这样与表中的其他数据分开存储的。SQL Server 2019 提供了将小型的文本和图像数据在行中存储的功能。

当将文本和图像数据存储在数据行中时，SQL Server 2019 不需要为访问这些数据而去访问另外的页，这使得读、写文本和图像数据可以与读写 varchar、nvarchar 和 varbinary 字符串一样快。

为了指定某个表的文本和图像数据在行中存储，需要使用系统存储过程 sp_tableoption 设置该表的 text in row 选项为 true。当指定 text in row 选项时，还可以指定一个文本和图像数据大小的上限值，这个上限值应为 24B～7000B。当同时满足以下两个条件时，文本和图像数据可以直接存储在行中。

(1) 文本和图像数据的大小不超过指定的上限值。

(2) 数据行有足够的空间存放这些数据。

当以上两个条件有一个不满足时，行中只存放指向这些数据实际存储位置的指针。

现建立表 text_example：

```
CREATE TABLE text_example
(bin_1 text,
bin_2 ntext)
```

以下语句将指定 text_example 表在行中存储文本和图像数据。

```
sp_tableoption 'text_example','text in row','true'
```

规定 text_example 表中不大于 1000B 的文本和图像数据直接在行中存储，可以执行以下语句。

```
sp_tableoption  'text_example', 'text in row', '1000'
```

如果不显式地指定上限，那么默认的上限为 256B，即在前一个例子中，行中存储文本和图像数据最大为 256B

以下语句指定 text_example 表不在行中存储文本和图像数据。

```
sp_tableoption 'text_example', 'text in row', 'false'
```

**8. 用户定义数据类型及使用**

用户定义的数据类型并不是真正的数据类型，它只是提供了一种加强数据库内部和基本数据类型之间一致性的机制。通过使用用户定义数据类型能够简化对常用规则和默认值的管理。

可以使用系统存储过程 sp_addtype 来创建用户定义数据类型。其语法格式如下。

```
sp_addtype type_name, system_type,  {'NULL'|'NOT NULL'}
```

默认为 NULL,可以为空。

凡是包含诸如“()”或“,”等分隔符的系统数据类型,如 Char(9),必须使用单引号括起来,即'Char(9)'。用户自定义数据类型在数据库中的命名必须唯一。只要命名唯一,甚至相同的类型定义也可以存储在同一个数据库中。

**【例 5-12】** 下面的例子创建了两个用户定义数据类型。

```
use model
go
exec sp_addtype telelephone,'varchar(24)','not null'
exec sp_addtype fax, 'varchar(24)','null'
```

可以直接使用用户定义数据类型来定义表列,就如同使用系统数据类型那样。但是用户自定义数据类型更常与默认值或规则等配合使用。

使用系统存储过程 sp_droptype 可以删除用户定义数据类型,其语法格式如下。

```
sp_droptype type_name
```

在实际使用中,如果用户定义数据类型正被某表中的某列使用,则不能立即删除,必须首先删除使用该数据类型的表。

也可以使用 SSMS 来创建用户定义数据类型,步骤如下。

(1) 选中要创建数据类型的数据库 Study,展开该结点。

(2) 选中树状结构上的“可编程性”→“类型”→“用户定义数据类型”。

(3) 右击,从弹出的快捷菜单中选择“新建用户定义数据类型”命令。在弹出的“新建用户定义数据类型”窗口中输入名称、系统数据类型、长度、可否为空等参数,如图 5-20 所示。

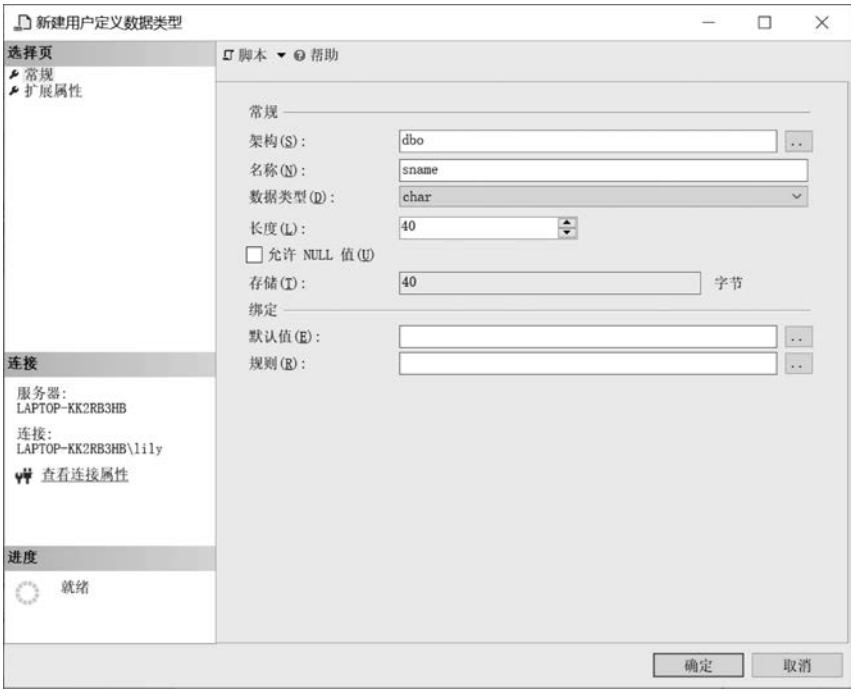


图 5-20 “新建用户定义数据类型”窗口

(4) 设置完毕后,单击“确定”按钮,即可完成用户定义数据类型的创建。

## 5.2.2　创建表结构

在创建表及其对象之前,最好先规划并确定表的下列特征。

(1) 表要包含的数据类型。

(2) 表中的列数,每一列中数据的类型和长度(如果必要)。

(3) 哪些列允许空值。

下面在数据库 Study 中创建以下 3 个表：Student(学生)、Course(课程)、Score(选课)。表的结构如表 5-1～表 5-3 所示。

**表 5-1　Student 表结构**

| 列　　名 | 数 据 类 型 | 可 否 为 空 | 备　　注 |
|---|---|---|---|
| sno | char(10) | Not null | 学号 |
| sname | varchar(40) | Not null | 姓名 |
| sage | tinyint | Null | 年龄 |
| ssex | char(4) | Null | 性别：男,女 |
| sbirthday | smalldatetime | Null | 出生日期 |
| depart | char(20) | Null | 系别 |
| class | char(10) | null | 班级 |

**表 5-2　Course 表结构**

| 列　　名 | 数 据 类 型 | 可 否 为 空 | 备　　注 |
|---|---|---|---|
| cno | char(10) | Not null | 课程号 |
| cname | varchar(40) | Not null | 课程名 |
| credit | char(4) | Null | 学分 |
| notes | varchar(200) | Null | 备注 |

**表 5-3　Score 表结构**

| 列　　名 | 数 据 类 型 | 可 否 为 空 | 备　　注 |
|---|---|---|---|
| sno | char(10) | Not null | 学号 |
| cno | char(10) | Not null | 课程号 |
| degree | tinyint | null | 成绩 |

表的创建也有两种方式：一是通过 SSMS 创建,二是用 T-SQL 命令创建。

### 1. 利用 SSMS 提供的图形界面创建表

利用 SSMS 提供的图形界面创建表,步骤如下。

(1) 在“对象资源管理器”的树形目录中找到要建表的数据库 Study,展开该数据库。

(2) 选择“表”,右击,在弹出的快捷菜单中选择“新建表”命令,打开表设计器,如图 5-21 所示。

(3) 表设计器的上半部分有一个表格,在这个表格中输入列的属性,表格的每一行对应设置一列。对每一列都需要进行以下设置。

① 列名：为每一列设定一个列名。

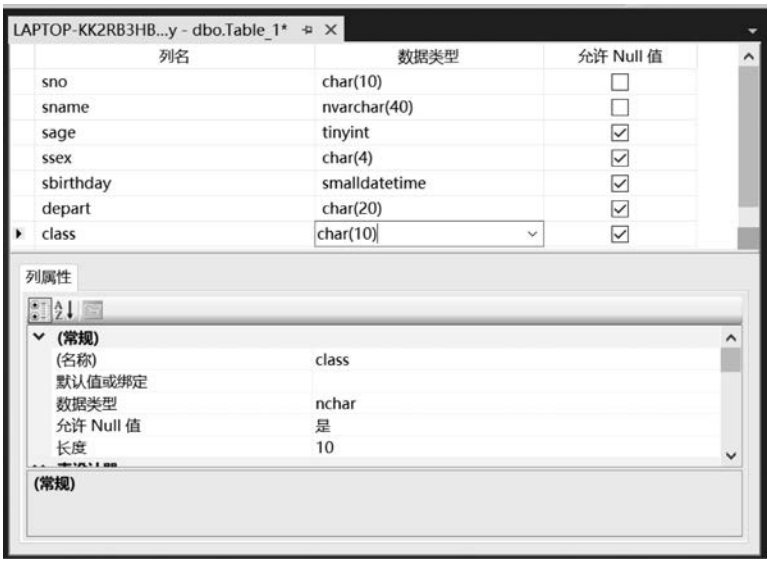


图 5-21　表设计器

② 数据类型：数据类型是一个下拉列表框，其中包括所有的系统数据类型和用户定义数据类型。用户可根据需要来选择数据类型和长度。

③ 允许 Null 值：单击该行的复选框，可以切换是否允许该列为空值的状态。选中状态表示允许为空值，空白表示不允许为空值，默认状态下是允许为空值的。

表设计器的下半部分是特定列的详细属性，包括是否是 IDENTITY 列、是否使用默认值等。

(4) 逐个定义好表中的列，单击工具栏中的“保存”按钮。若没有在表设计器中给出表的名称，会出现保存对话框，提示用户输入表的名称，这里输入“Student”，如图 5-22 所示。单击“确定”按钮，Student 表就建立完成了。


图 5-22　保存表结构

### 2. 使用 T-SQL 语句创建表

在 T-SQL 中，可以用 CREATE TABLE 命令来创建表，表中列的定义必须用括号括起来。一个表最多 1024 列。CREATE TABLE 的基本语法格式如下。

```
CREATE TABLE table_name
({column_name   datatype   NOT NULL|NULL})
```

**说明：**

(1) table_name 是所创建的表的名称，表名在一个数据库内必须唯一。

（2）column_name 是列名，列名在一个表内必须唯一。

（3）datatype 是该列的数据类型，可以使用系统数据类型，也可以使用用户定义数据类型。对于需要给定数据最大长度的类型，在定义时要给出长度。

（4）NOT NULL|NULL 指示该列可否输入空值，默认可以为空。

**【例 5-13】** 在 Study 数据库上创建课程表 Course 和成绩表 Score。

```
use Study
go
CREATE TABLE Course                    /*定义课程表 Course*/
(cno char(10) not null,                /*课程号列 cno,不能输入空值*/
 cname varchar(40) not null,           /*课程名列 cname,不能输入空值*/
 credit char(4),                       /*学分列 credit */
 notes varchar(200))                   /*备注列 notes*/
go

CREATE TABLE Score                     /*定义成绩表 Score*/
(sno char(10) not null,                /*学号列 sno,不能输入空值*/
 cno char(10) not null,                /*课程号列 cno,不能输入空值*/
 degree tinyint)                       /*成绩列 degree*/
go
```

## 5.2.3　查看表结构

### 1. 使用 SSMS 查看表属性

在 SSMS 的“对象资源管理器”中找到要查看的表所在数据库，打开树形菜单中的“表”结点，其下方会显示这一数据库中所有的表，其中系统表单独在“系统表”文件夹内。对于每个表，都会显示它的结构。

在列表中选择一个要查看的表，右击打开快捷菜单，选择“属性”命令，打开“表属性”窗口，如图 5-23 所示。可以在此查看、设置表的属性、权限和扩展属性等。

### 2. 使用 T-SQL 语句显示表结构

可以使用系统存储过程 sp_help 来查看表结构，包括表的所有者、类型（系统表还是用户表）、创建时间、表上每一列的名称、数据类型、表上定义的索引及约束等。

**【例 5-14】** 查看表 Course 的基本结构。

```
use Study
go
exec sp_help  Course
```

使用查询编辑器执行这一语句的结果如图 5-24 所示。

实际上，使用 sp_help 可以查看所有数据库对象的定义，除了表外，还包括视图、存储过程以及用户定义数据类型等，只需将不同数据库对象的名字作为 sp_help 的输入参数即可。另外，若执行不带参数的 sp_help 命令时，将返回当前数据库中的所有数据库对象。

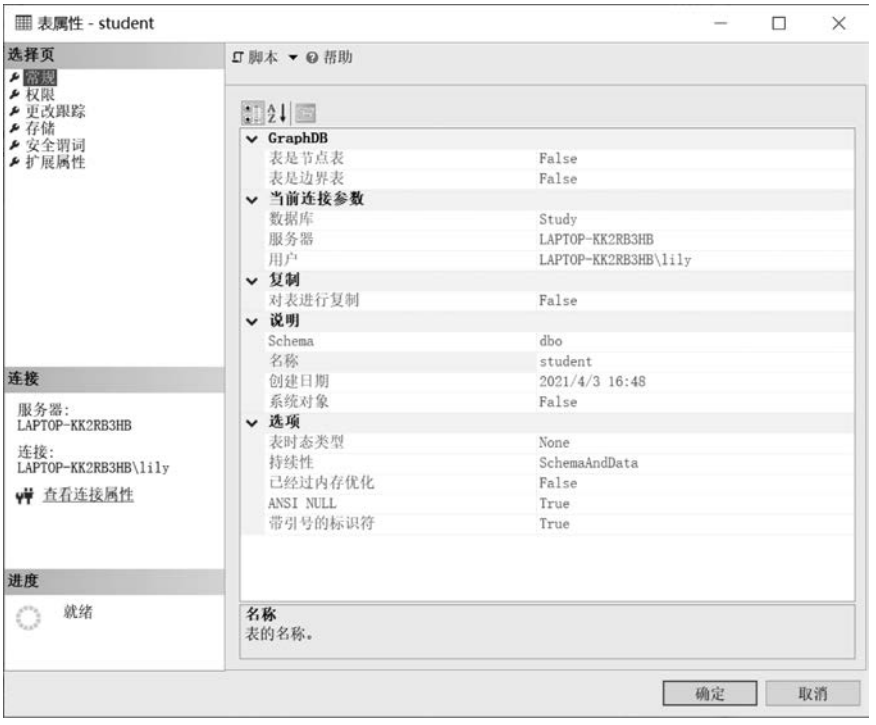


图 5-23　查看 Student 表的属性

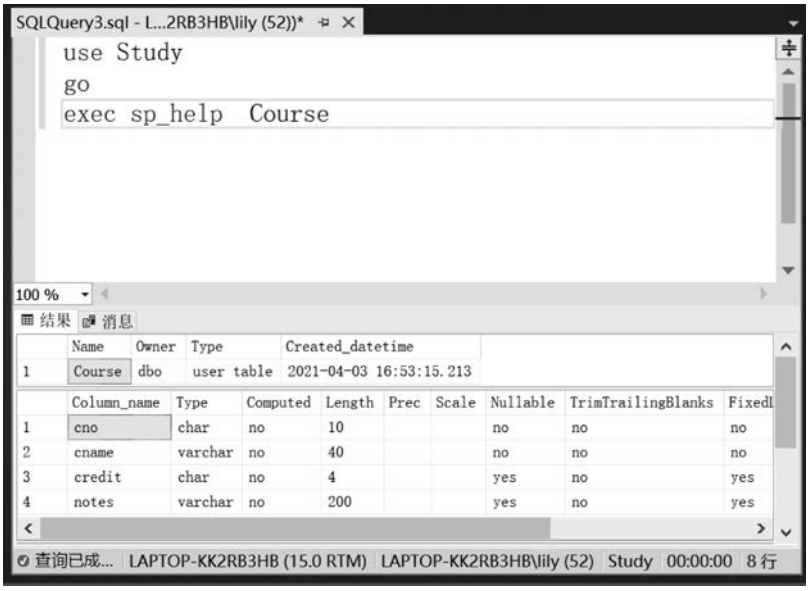


图 5-24　查看 Course 表的结构

**【例 5-15】** 查看 Study 数据库上的所有数据库对象。

```
use Study
go
exec sp_help
```

执行结果如图 5-25 所示。

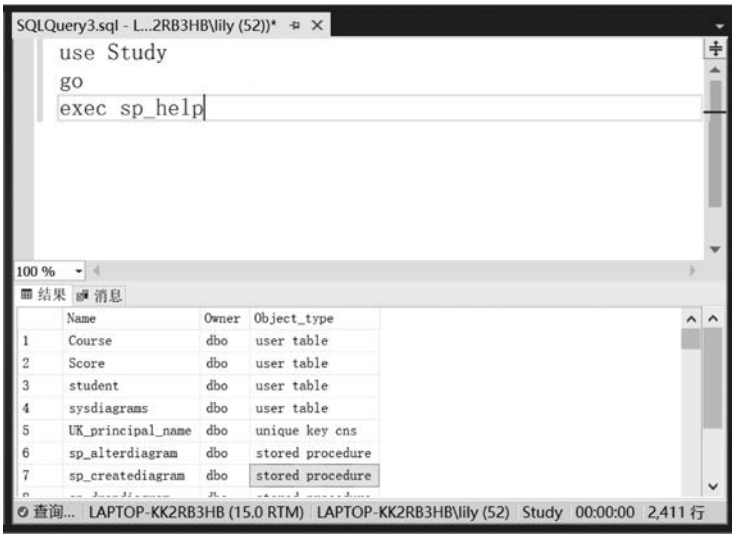


图 5-25　查看 Study 库的所有对象

## 5.2.4　修改表结构

创建完一个表以后，如果要修改表的结构，如添加、修改及删除列等，可以使用 SSMS 或 T-SQL 语句两种方式实现。

**1. 使用 SSMS 修改表**

在 SSMS 的“对象资源管理器”中找到要修改的表，右击，在弹出的快捷菜单中选择“设计”命令，将弹出如图 5-21 所示的表设计器。

此时可以像新建表时一样，向表中增加列、从表中删除列或修改列的属性，修改完毕后单击“保存”按钮即可。

选中某一列，右击，在弹出的快捷菜单中选择“删除列命令”，则可删除某一列。用与建表时相似的方法可以对列值进行修改。

**2. 使用 T-SQL 语句修改表**

（1）添加列。向表中增加一列时，应使新增加的列有默认值或允许为空值，SQL Server 将向表中已存在的行填充新增列的默认值或空值。如果既没有提供默认值也不允许为空值，那么新增列的操作将会出错，因为 SQL Server 不知道该怎么处理那些已经存在的行。

向表中添加列的语句格式如下。

```
ALTER TABLE  表名
ADD   列名 列的描述
```

**【例 5-16】** 向 Student 表中添加两个新的列：邮箱 Email 和电话 phone。

```
use Study
go
ALTER TABLE Student
ADD Email varchar(20) null,
```

```
    phone char(20) null
```

可以一次向表中添加多个列,各列之间用逗号分开即可。

(2) 删除列。删除一列的语句格式如下。

```
ALTER TABLE  表名
DROP COLUMN  列名
```

**【例 5-17】** 删除 Student 表的 Email 列。

```
use Study
go
ALTER TABLE Student
DROP COLUMN Email,phone
```

(3) 修改列定义。表中的每一列都有其定义,包括列名、数据类型、数据长度及是否允许为空值等,这些值都可以在表创建好以后修改。

修改列定义的语句格式如下。

```
ALTER TABLE  表名
ALTER COLUMN  列名 列的描述
```

**【例 5-18】** 将 Student 表的姓名列 sname 改为最大长度为 20 的 varchar 型数据,且不允许为空值。

```
use Study
go
ALTER TABLE Student
ALTER COLUMN sname varchar(20) not null
```

默认状态下,列是被设置允许空值的,将一个原来允许为空的列改为不允许为空时,必须在以下两个条件满足时才能成功。

① 列中没有存放空值的记录。

② 在列上没有创建索引。

### 5.2.5 删除表结构

当不再需要某个表时,可以将其删除。一旦一个表被删除,那么它的数据、结构定义、约束、索引等都将被永久地删除,以前用来存储数据和索引的空间可以用来存储其他的数据库对象。

#### 1. 使用 SSMS 删除表

使用 SSMS 删除一个表非常简单,只需在“对象资源管理器”中找到要删除的表,右击,在弹出的快捷菜单中选择“删除”命令。在打开的如图 5-26 所示的“删除对象”窗口中,单击“确定”按钮即可。

#### 2. 使用 T-SQL 语句删除表

在 T-SQL 语句中,DROP TABLE 语句可以用来删除表。其语法格式如下。

```
DROP TABLE  表名
```

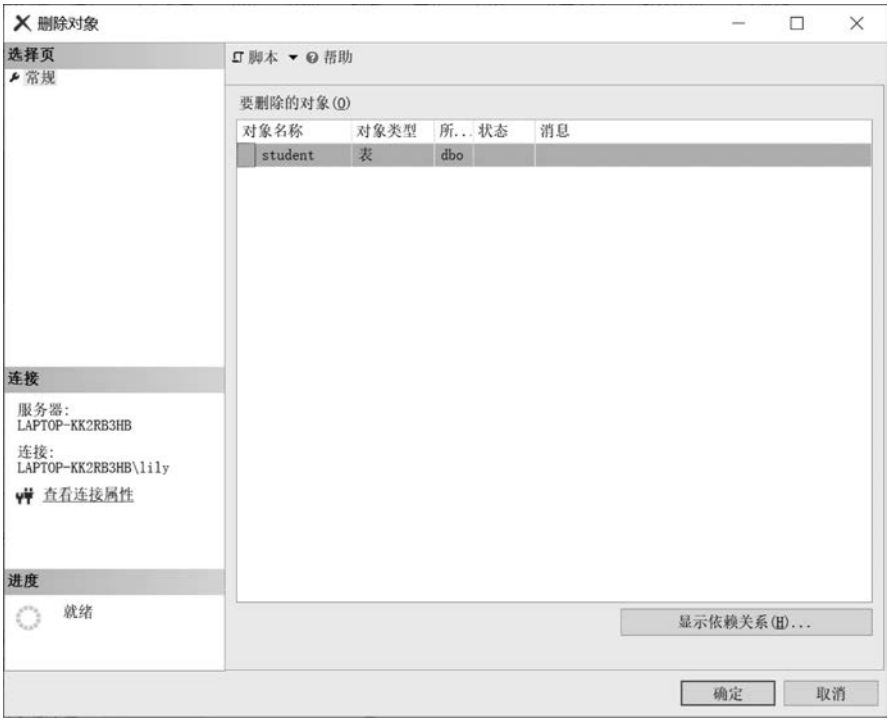


图 5-26　“删除对象”对话框

需要注意的是,DROP TABLE 语句不能用来删除系统表。

**【例 5-19】**　删除 Study 数据库中的表 table1。

```
use Study
go
DROP TABLE table1
```

## 5.3　数据更新


视频讲解

新创建的表中没有任何数据,可以通过数据更新操作向表中插入新数据,修改表中的数据和删除表中的数据。

### 5.3.1　向表中插入数据

使用 INSERT 语句,可以实现数据的插入操作。INSERT 语句的基本语法格式如下。

```
INSERT[INTO]表名[(列名)]
VALUES(表达式)
```

**1. 添加数据到一行中的所有列**

当将数据添加到一行中的所有列时,使用 VALUES 关键字来给出要添加的数据。INSERT 语句中无须给出表中的列名,只要在 VALUES 中给出的数据与用 CREATE TABLE 定义表时给定的列名顺序、类型和数量均相同即可。

【例 5-20】 向 Course 表中添加一条记录。

```
use Study
go
INSERT INTO Course
VALUES('080110H','数据库原理与应用','3','必修')
```

在查询编辑器中执行,返回的结果为:

```
(1 行受影响)
```

若对表中的结构不明确,即对列的顺序不明确,则要在表名后面给出具体的列名,而且列名顺序、类型和数量也要与 VALUES 中给出的数据一一对应。如上面的语句也可以写成如下形式。

```
INSERT INTO Course(cno,cname,credit,notes)
VALUES('080110H','数据库原理与应用','3','必修')
```

**2. 添加数据到一行中的部分列**

要将数据添加到一行中的部分列时,则必须同时给出要使用的列名以及要赋给这些列的数据。

【例 5-21】 向 Student 表中添加一条记录。

```
use Study
go
INSERT INTO Student(sno,sname,sage,ssex,sbirthday)
VALUES('05091101','李明',19,'男','1989-1-10')
```

在查询编辑器中执行,返回的结果为:

```
(1 行受影响)
```

对于这种添加部分列的操作,在添加数据前应确认表中的列是否允许为空,只有允许为空的列才能不出现在 VALUES 列表中。

**注意:**

(1) 输入数据的顺序和数据类型必须与表中列的顺序和数据类型一致。

(2) 列名与数据必须一一对应,当每列都有数据输入时,列名可以省略,但输入数据的顺序必须与表中列的定义顺序相一致。

(3) 可以不给全部列赋值,但没有赋值的列必须是可以为空的列。

(4) 插入字符型和日期型数据时要用单引号括起来。

**3. 添加多行数据**

通过在 INSERT 语句中嵌套子查询,可以将子查询的结果作为批量数据,一次向表中添加多行数据。查询语句将在第 6 章中讲解,在此只给出一个简单的例子。

【例 5-22】 添加批量数据。

创建一个新的数据表 stu_temp:

```
CREATE TABLE stu_temp
(sno char(10) not null,
sname char(20) not null,
```

```
sage tinyint)
go
```

假设 Student 中已有一批数据，可以从 Student 表中选择部分数据插入新表 stu_temp 中，此时将所有女生的信息插入新表 stu_temp 中。

```
INSERT INTO stu_temp
SELECT sno,sname,sage
FROM Student
WHERE ssex = '女'
```

### 5.3.2　修改表中数据

当数据添加到表中后，会经常需要修改，如客户的地址发生了变化、货品库存量的增减等。使用 UPDATE 语句可以实现数据的修改，其语法格式如下。

```
UPDATE  表名
SET  列名 = 表达式
[ WHERE 条件 ]
```

**【例 5-23】** 将所有课程的备注信息都改为“必修”。

```
use Study
go
UPDATE  Course
SET notes = '必修'
```

因为没有使用 WHERE 子句，所以对备注列的修改将影响到表中的每一行。

**【例 5-24】** 将课程号为 080110H 的课程改为“选修”。

```
use Study
go
UPDATE Course
SET notes = '必修'
WHERE cno = '080110H'
```

在此例中，只有满足 WHERE 子句条件的行被修改。

### 5.3.3　删除表中数据

当数据的添加工作完成以后，随着使用和对数据的修改，表中可能存在着一些无用的数据，这些无用数据不仅会占用空间，还会影响修改和查询数据的速度，所以应及时将它们删除。

#### 1. 使用 DELETE 语句删除数据

使用 T-SQL 中的 DELETE 语句可以删除数据表中的一个或多个记录。DELETE 语句最简单的形式如下。

```
DELETE  表名
[WHERE  条件]
```

其中，表名是要删除数据的表的名字。如果 DELETE 语句中没有 WHERE 子句限制，表中的所有记录都将被删除。

【例 5-25】 删除学号为 05091101 的学生的基本信息。

```
use Study
go
DELETE Student
WHERE sno = '05091101'
```

执行结果为：

```
(1 行受影响)
```

### 2. 使用 TRUNCATE TABLE 语句删除数据

TRUNCATE TABLE 语句提供了一种删除表中所有记录的快速方法，因为 TRUNCATE TABLE 语句不记录日志，只记录整个数据页的释放操作，而 DELETE 语句对每一行修改都记录日志，所以 TRUNCATE TABLE 语句总比没有指定条件的 DELETE 语句快。

【例 5-26】 删除所有学生的成绩记录。

```
TRUNCATE  TABLEScore
```

因为 TRUNCATE TABLE 操作是不进行日志记录的，所以建议在使用 TRUNCATE TABLE 语句之前先对数据库备份，数据库备份的操作将在第 10 章中介绍。


视频讲解

## 5.4 数据导入

在实际应用中，经常需要在不同或相同的数据源之间互相复制数据。SQL Server 2019 提供了功能非常强大的组件，可以将 SQL Server 2019 中的数据导出到其他数据源或从其他数据源导入数据到 SQL Server 2019 中。

导出操作可以看作导入操作的反操作，本节主要介绍如何使用 SQL Server 导入向导将 Excel 工作表中的数据导入 SQL Server 2019 中。

首先，新建一个 Excel 示例文件，在 Sheet1 工作表中输入需要导入的数据，文件名命名为 teacher.xls，并保存在预先建好的文件夹下，如图 5-27 所示。

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | tno | tname | tage | tsex | tdept | |
| 2 | 9001 | 刘云 | 35 | 女 | 信息系 | |
| 3 | 9002 | 周群 | 42 | 女 | 信息系 | |
| 4 | 9003 | 孙景琪 | 23 | 男 | 信息系 | |
| 5 | 8001 | 王乐洋 | 36 | 女 | 土木系 | |
| 6 | 8002 | 卢峰 | 28 | 男 | 土木系 | |
| 7 | 8003 | 宋彬 | 40 | 男 | 土木系 | |
| 8 | | | | | | |

图 5-27 新建 Excel 文件

下面使用 SQL Server 导入向导将此 Excel 文件中的数据导入 Study 数据库中，其操作步骤如下。

(1) 打开 SSMS，在“对象资源管理器”中展开“数据库”文件夹，右击要向其导入数据的数据库，如 Study，在弹出的快捷菜单中选择“任务”→“导入数据”命令。系统会启动 SQL

Server 导入和导出向导工具，并出现欢迎界面，如图 5-28 所示。

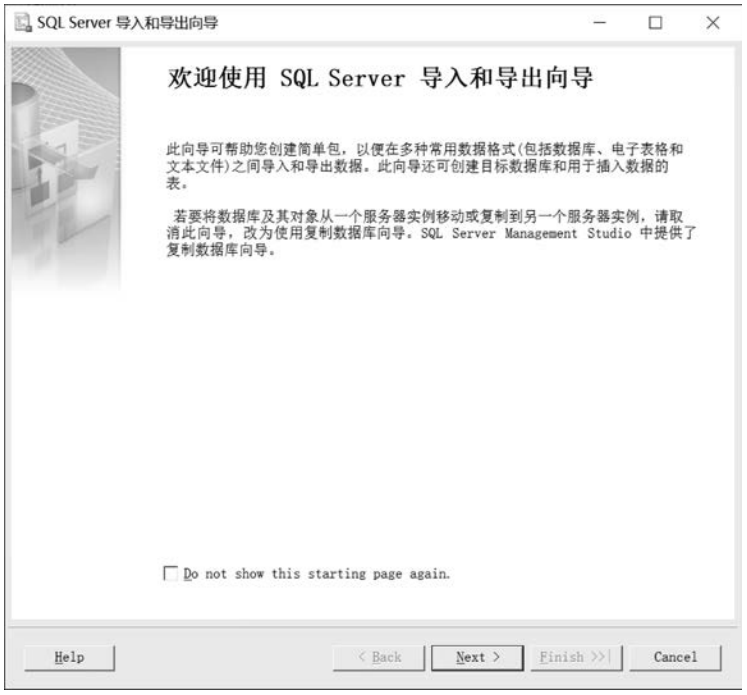


图 5-28　导入和导出向导欢迎界面

（2）单击 Next 按钮，出现“选择数据源”对话框，如图 5-29 所示。

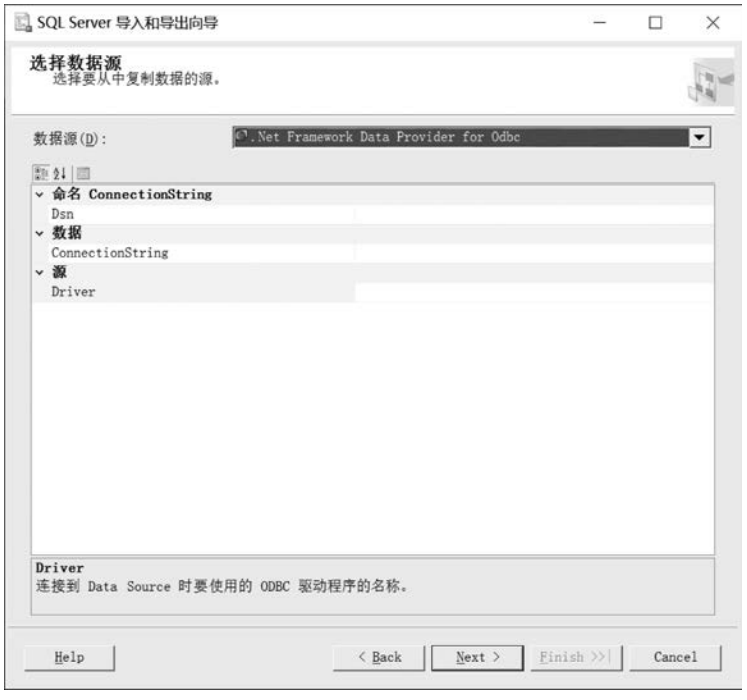


图 5-29　“选择数据源”对话框

在此对话框中可以设置数据源的相关信息,可用的数据源包括 Access、OLE DB 访问接口、SQL 本机客户端和 Excel 文件等。数据源不同,需要设置的身份验证模式、服务器名称、数据库名称和文件格式等选项也不同。根据实际操作中数据源的类型,在其后的下拉列表中选择相应的类型。在此选择 Microsoft Excel,如图 5-30 所示。

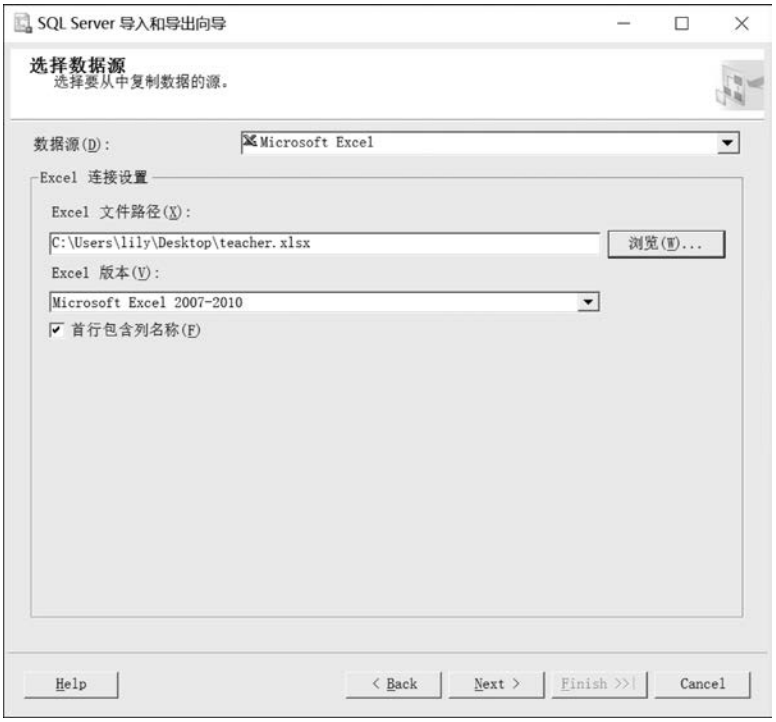


图 5-30 设置 Excel 数据源

在 Excel 连接设置中设置 Excel 文件路径和 Excel 版本信息,并选中“首行包含列名称”复选框,表示该文件中的首行为列名称信息。

(3) 设置完数据源的相关信息后,单击 Next 按钮,出现“选择目标”窗口,如图 5-31 所示。

在“目标”选项下拉列表中选择 Microsoft OLE DB Provider for SQL Server,表示选择 SQL Server 作为数据导入的目的地。选择相应的服务器名称、身份验证方式和数据导入的数据库 Study,单击 Next 按钮即可。

(4) 接下来出现“指定表复制或查询”窗口,如图 5-32 所示。

其中,“复制一个或多个表或视图的数据”选项可以用于指定复制源数据库中现有表或视图的全部数据。“编写查询以指定要传输的数据”选项可以用于编写 SQL 查询,以便对复制操作的源数据进行操纵或限制。在此选择第一项“复制一个或多个表或视图的数据”。

(5) 单击 Next 按钮,会出现“选择源表和源视图”窗口,如图 5-33 所示。

在此窗口中可以设置数据导入的“源”“目标”和“映射”等选项。本例中,选中数据所在的工作表 Sheet1 $,“目标”选项中会出现数据导入的默认目标[Study].[dbo].[Sheet1 $],可以对其进行修改,如将表名改为“teacher”。也可以选择已经存在的数据表作为数据导入的目的地。单击“编辑映射”按钮,将打开“列映射”对话框,如图 5-34 所示。


图 5-31　“选择目标”窗口


图 5-32　“指定表复制或查询”窗口

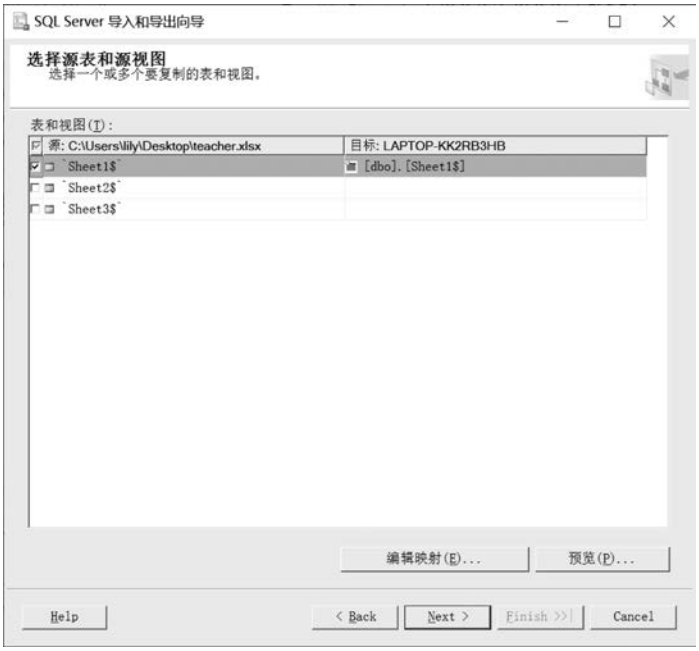


图 5-33 “选择源表和源视图”窗口

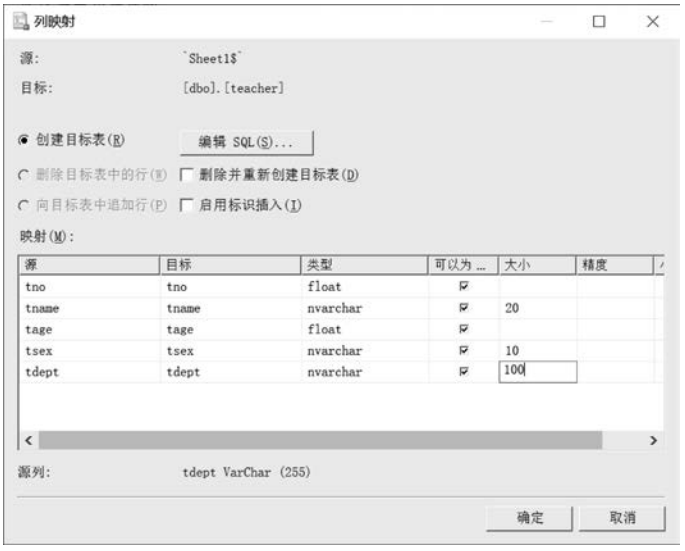


图 5-34 “列映射”窗口

在“列映射”窗口中,可以设置源和目标之间列的映射关系,来协调源和目标之间类型等的差异。还可以设置在数据导入时系统所做的工作,如对于不存在的目标表,可以选择“创建目标表”,而对于已存在的目标表,则可以根据需要选择“删除目标表中的行”“向目标表中追加行”或“删除并重新创建目标表”三种不同的操作。另外,还可以选择“启用标识插入”复选框来增加一个标识列。

设置完毕单击“确定”按钮,返回“选择源表和源视图”窗口,单击 Next 按钮,进入“保存并运行包”窗口。

（6）在“保存并运行包”窗口中，设置是否要立即执行，还可以设置将包保存到 SQL Server msdb 数据库或保存到文件系统，如图 5-35 所示。


图 5-35　“保存并运行包”窗口

（7）设置完毕后，单击 Next 按钮，打开完成该向导窗口，给出本次数据导入的信息，如图 5-36 所示。

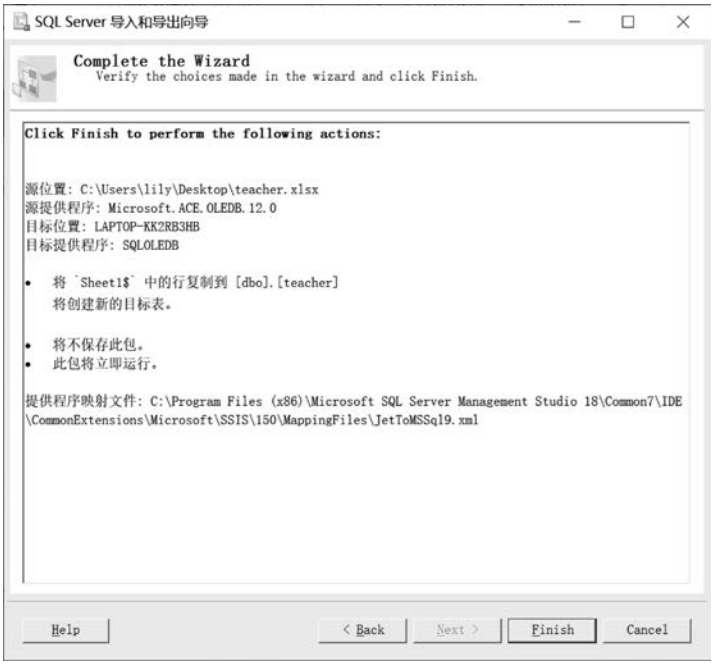


图 5-36　完成该向导窗口

单击 Finish 按钮，会出现导入数据的执行过程，并出现“执行成功”窗口，如图 5-37 所示。单击 Close 按钮，完成本次导入数据的操作。

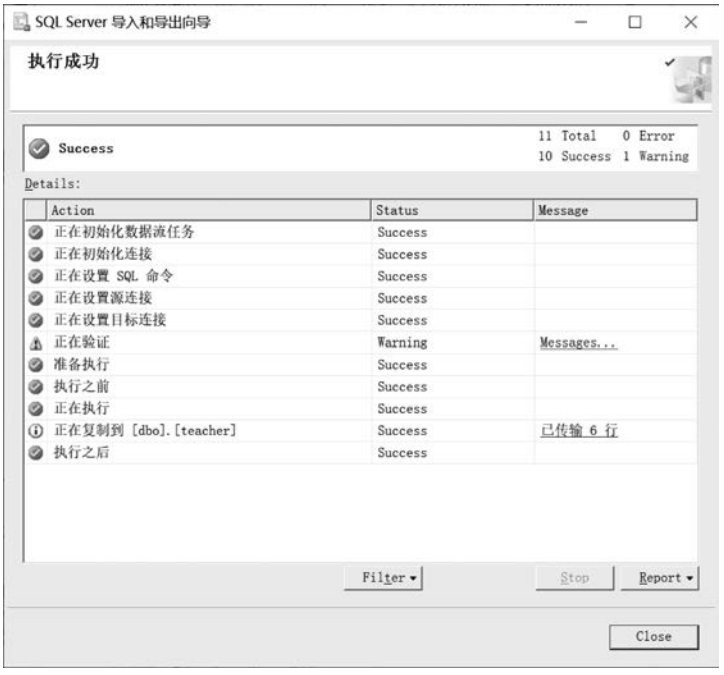


图 5-37 “执行成功”窗口

(8) 数据导入成功后，可以打开 Study 数据库，查看 teacher 表中的数据，如图 5-38 所示。

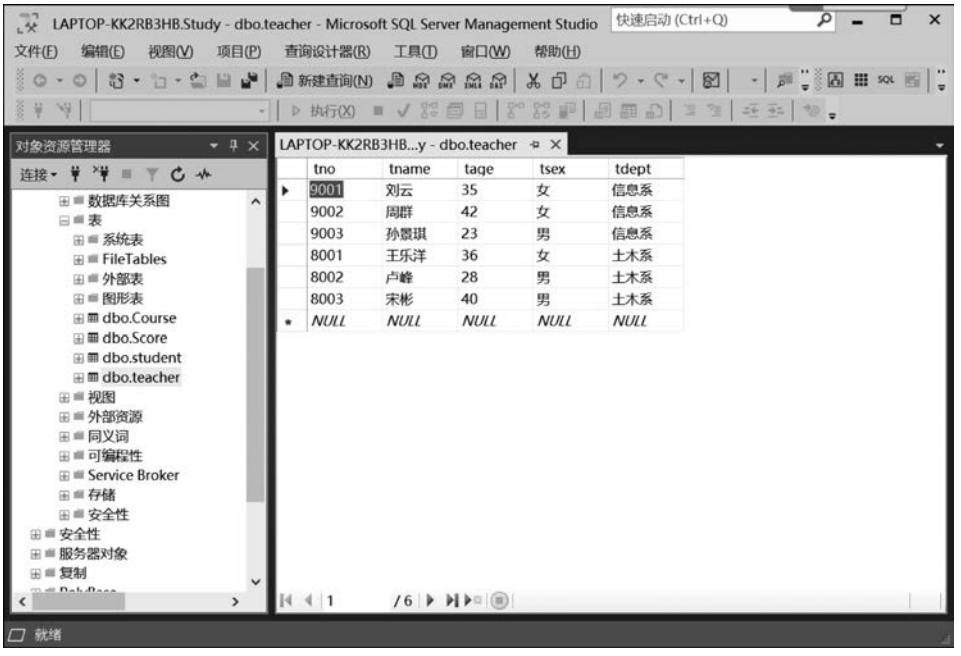


图 5-38 查看导入的结果

# 小结

SQL Server 2019 用文件来存放数据库，即将数据库映射到操作系统文件上。数据库文件有 3 类：主数据文件、次要数据文件和事务日志文件。用户可以使用 SSMS 和 T-SQL 语句两种方式创建、查看、修改和删除数据库。

SQL Server 为了实现 T-SQL 的良好性能，提供了丰富的数据类型，包括数值型、字符型、日期/时间型、货币型、二进制型、双字节型、图像、文本型以及用户自定义型数据。可以使用 SSMS 和 T-SQL 语句两种方式创建、显示、修改和删除数据表结构。

新创建的表中没有任何数据，可以用 T-SQL 语句插入、修改和删除数据，也可以用 SQL Server 2019 的数据导入功能实现批量数据导入。

# 习题

**一、填空题**

1. SQL Server 2019 提供的系统数据类型有_____、_____、_____、_____、_____和货币数据，也可以使用用户定义的数据类型。

2. SQL Server 2019 的数据库包含 3 类文件：_____、_____和_____。包含 4 个系统数据库：_____、_____、_____和 msdb 数据库。

3. 可以使用系统存储过程_____来查看表的定义，后面加上要查看的_____作为参数。

**二、操作题**

1. 创建用户定义的数据类型：编号(非空，长度为 8 的字符型)。

2. 创建图书数据库(BookSys)，并在数据库中建立图书信息(tsxx)、读者信息(dzxx)和借阅信息(jyxx)表，表结构如表 5-4～表 5-6 所示。要求图书编号、读者编号使用用户定义类型：编号。

**表 5-4　tsxx 表结构**

| 图书编号<br>(tusbh) | 书名<br>(shum) | 价格<br>(jiag) | 出版社<br>(chubs) | 出版日期<br>(chubrq) | 作者<br>(zuoz) |
|---|---|---|---|---|---|

**说明**：图书编号、书名不能为空。

**表 5-5　dzxx 表结构**

| 读者编号<br>(duzbh) | 姓名<br>(xingm) | 身份证号<br>(shenfzh) | 级别<br>(jib) |
|---|---|---|---|

**说明**：读者编号、姓名不能为空。

**表 5-6　jyxx 表结构**

| 读者编号<br>(duzbh) | 图书编号<br>(tusbh) | 借阅日期<br>(jieyrq) | 还书日期<br>(huansrq) | 是否续借<br>(shifxj) |
|---|---|---|---|---|

**说明**：图书编号、读者编号不能为空。

3. 完成如下操作。

(1) 向读者信息表中添加列：联系方式，可以为空。

(2) 修改列“出版社”的定义，长度修改为 200。

(3) 删除“联系方式”一列。

4. 完成如下数据操作。

(1) 向各表插入若干数据。

(2) 修改读者信息表中编号为 00001001 的读者的级别为 2 级。

(3) 删除借阅信息表中编号为 00001001 的读者借阅 10010001 图书的记录。

5. 完成如下操作。

导入一个 Excel 表到 SQL Server 2019 的 BookSys 数据库中。