第 3 章

操作数据表

在第 2 章中已经讲解了数据库的一些基本操作，那么，数据库中的数据如何存放呢？数据库相当于一个文件夹，在一个文件夹中可以存放多个文件。数据库中的文件称为数据表，也就是用来存储数据的容器。一个数据库有若干张数据表，每张数据表的名字都是唯一的，就像一个文件夹中的文件名都是唯一的一样。

本章主要知识点如下：

- 数据表中的数据类型；
- 如何创建数据表；
- 如何修改数据表；
- 如何删除数据表。

3.1 数据表中字段的数据类型

当我们在网站上注册一个用户的时候，需要输入哪些数据呢？通常有用户名、密码、邮箱、年龄和联系方式等。只要是注册时填入的数据，最终都将提交到数据库中存放。这些数据包含什么呢？输入注册信息的时候会有汉字、数字、字母以及特殊符号等。既然这些数据都能够存到数据库中，也就是说数据表中能存放这些类型的数据。实际上，在数据表中不仅可以存储这些数据类型，还可以存放很多其他数据类型。本节将详细讲解 SQL Server 数据表中使用的数据类型。

3.1.1 整型和浮点型

整型和浮点型都属于数值类型，也就是用来存放数字的一种数据类型。这个类型在日常生活中用得比较多，在什么情况下需要整数和小数呢？当存放年龄时需要整数，当存放金额时需要小数，当存放商品数量时需要整数等。那在 SQL Server 数据库中整数和浮点数用什么名称表示呢？首先，学习如表 3.1 所示的整数类型。

表 3.1 整数类型

数据类型	取值范围	说明
bit	存储 0 或 1	表示位整数，除了 0 和 1 之外，也可以取值 NULL
tinyint	$0\sim2^{8}-1$	表示小整数，占 1 个字节
smallint	$-2^{15}\sim2^{15}-1$	表示短整数，占 2 个字节
int	$-2^{31}\sim2^{31}-1$	表示一般整数，占用 4 个字节
bigint	$-2^{63}\sim2^{63}-1$	表示大整数，占用 8 个字节

从表 3.1 可以看出，整数类型主要包括 bit、tinyint、smallint、int 和 bigint，它们的取值范围是从小到大的。在实际应用中，要根据存储数据的大小选择数据类型，这样能够节省数据库的存储空间。这就像在超市结账时，根据选择物品的多少再购买购物袋一样。如果购买的东西多，就选择大号的；如果购买的东西少，就选择小号的。

接下来介绍表 3.2 所示的浮点型。

表 3.2 浮点型

数据类型	取值范围	说明
numeric(m,n)	$-10^{38}+1\sim10^{38}-1$	表示$-10^{38}+1\sim10^{38}-1$范围中的任意小数，numeric(m,n)中的 m 代表有效位数，n 代表小数要保留的小数位数。例如：numeric(7,2)表示是长度为 7 的数，并保留 2 位小数
decimal(m,n)	$-10^{38}+1\sim10^{38}-1$	与 numeric(m,n)的用法相同
real	$-3.40E+38\sim3.40E+38$	占用 4 个字节
float	$-1.79E+308\sim1.79E+308$	占用 8 个字节

从表 3.2 可以看出，如果要精确表示小数可以使用 numeric(m,n)或者 decimal(m,n)；如果不需要精确并且表示更多的小数位数可以使用 real 或者 float。总之是根据数据的大小和精度选择合适的浮点型。

3.1.2 字符串类型

字符串类型是数据表中存储数据最常用的数据类型。那么，什么样的数据可以用字符串类型表示呢？任何数据都可以说成是字符串类型，汉字、字母、数字、一些特殊字符甚至是日期形式都可以用字符串类型存储。用来表示字符串的数据类型是按照存储字符串的长度划分的。具体分类如表 3.3 所示。

表 3.3 字符串类型

数据类型	取值范围(字符)	说明
char(n)	1～8000	用来表示固定长度的字符串，如果存放的数据没有达到定义时长度，系统会自动用空格填充到该长度
varchar(n)	1～8000	用于表示变长的数据。1 个字符占 1 个字节，不用空格填充长度
varchar(max)	$1\sim2^{31}-1$	用于表示变长的数据。该数据类型表示的长度是输入数据的实际长度加上 2 个字节

续表

数据类型	取值范围(字符)	说明
text	$1\sim2^{31}-1$	用于表示变长的数据。1个字符占1个字节,最大可以存储2GB的数据
nchar(n)	1～4000	用于表示固定长度的双字节数据。1个字符占2个字节。与char类型一样,如果存放的数据没有达到定义时的长度,系统会自动用空格填充到该长度
nvarchar(n)	1～4000	用于表示变长的数据。与varchar(n)的区别就是1个字符需要占用2个字节来表示
nvarchar(max)	$1\sim2^{31}-1$	用于表示变长的数据。该数据类型表示的长度是输入数据的实际长度的2倍加上2个字节
ntext	$1\sim2^{31}-1$	用于表示变长的数据。1个字符占2个字节,最大可以存储2GB的数据
binary(n)	1～8000	用于表示固定长度的二进制数据。如果输入数据的长度没有达到定义的长度,用0X00填充
varbinary(n)	1～8000	用于定义一个变长的数据。存储的是二进制数据,输入的数据实际长度小于定义的长度也不需要填充值
image	$1\sim2^{31}-1$	用于定义一个变长的数据。image类型不用指定长度,可以存储二进制文件数据

通过学习表3.3中的数据类型,读者不难发现,实际上字符串类型可以大致分为三类:一是1个字符占用1个字节的字符串类型(char、varchar及text);二是1个字符占用2个字节的字符串类型(nchar、nvarchar及ntext);三是存放二进制数据的字符串类型(binary、varbinary及image)。每类中字符串类型又分为存放固定长度和可变长度的类型,在实际的应用中推荐读者在不确定字符的取值范围时使用可变长度的类型,而为固定字符长度的数据选择固定长度的数据类型,例如,身份证号、手机号等,这样就可以节省数据的存储空间。

说明:在字符串类型中的varchar(max)及nvarchar(max)类型是在SQL Server 2005版本上开始使用的。

3.1.3 日期时间类型

虽然日期时间可以用字符串类型表示,但是在SQL Server中还是为其准备了一套数据类型专门用于表示日期时间类型。通过日期时间类型可以将日期时间表示得更加准确了,在SQL Server中表示日期时间类型的数据类型主要有datetime和smalldatetime两种。具体的表示方法如表3.4所示。

表3.4 日期时间类型

数据类型	取值范围	说明
datetime	1753年1月1日～9999年12月31日	占用8个字节,精确到3.33毫秒
smalldatetime	1900年1月1日～2079年6月6日	占用4个字节,精确到分钟

虽然有了存储日期时间的数据类型,但还要清楚日期时间的存储格式。通常日期的输入格式有3种,英文+数字的格式、数字+分隔符的格式、数字格式。下面分别用这三种形

式表示 2018 年 5 月 1 日。

```
May 1 2018              -- 英文 + 数字格式
2018 - 5 - 1            -- 数字 + 分隔符格式
2018.5.1                -- 数字 + 分隔符格式
2018/5/1                -- 数字 + 分隔符格式
20180501                -- 数字格式
180501                  -- 数字格式
```

看了上面的例子，可以对日期类型的表示有所了解。在这 3 种表示方法中，“数字＋分隔符”的格式是最常用的，也是最灵活的。除了上面的 3 种数字＋分隔符的表示形式外，还可以按照月日年、日月年的顺序表示日期类型的数据。例如，5-1-2018、1-5-2018 等形式。

除了日期有固定的存储格式外，时间部分的数据也有固定的存储格式。通常时间类型的数据存储的格式都是按照“小时：分钟：秒.毫秒”存储的。例如，上午的 9 点 10 分 20 秒，可以用 9:10:20 表示。时间的表示可以分为 24 小时和 12 小时两种格式，如果是 12 小时的格式，用 am 表示上午，用 pm 表示下午。例如，晚上的 10 点 30 分 10 秒，可以用 10:30:10 pm 表示。

日期时间数据通常在一起存储，需要在日期格式后面加上一个空格然后加上时间格式表示。例如，2018 年 5 月 25 日下午 5 点 25 分 10 秒，就可以写成 2018-5-25 5:25:10 pm。存储日期时间类型时，要注意格式问题。另外，还要提醒读者在一个数据表中存储的日期时间的格式要统一，否则在查询数据时就会造成一些麻烦。

3.1.4 其他数据类型

在数据表存储数据除了上面讲的 3 类比较常用的数据类型外，还有一些不太常用的数据类型。例如，timestamp 类型、xml 类型、cursor 类型等。timestamp 类型时间戳类型，在更新数据时，系统会自动更新时间戳类型的数据，它也可以用于表示数据的唯一性。另外，在一张数据表中只能有一个时间戳类型的列。xml 类型可以存储之前学过的其他类型的数据，也可以存储 XML 文件格式的数据，它的存储空间最大是 2GB。cursor 类型是用于存储变量或者是存储过程输出的结果，它通常都用于存储查询结果，在存储过程中应用较多。

除了系统自带的数据类型外，如果用户觉得这些数据类型满足不了需求时也可以自定义数据类型。自定义数据类型很简单，具体的语法如下：

```
CREATE TYPE type_name
FROM datatype;
```

- type_name：自定义的数据类型名称。名称不能以数字开头。
- datatype：数据类型。自定义的数据类型，除了写数据类型外，还可以指定该类型是否为空值。

例 3-1 定义一个数据类型，用来表示字符串长度是 20 并且不能为空。

根据题目要求，仍然需要定义一个字符串类型，可以选择系统的字符串类型有很多，char、varchar、nchar 和 nvarchar 都是可以的。这里，选择一个可变长度的字符串类型

varchar，具体的语法如下：

```
CREATE TYPE usertype
FROM varchar(20) NOT NULL;
```

通过上面的语法可以为数据库新添加一个数据类型 usertype，在使用该类型时直接用 usertype 就可以了。

如果不需要自定义的数据类型了，也可以通过 DROP TYPE 语句将其删除。如果要删除在示例 3-1 中定义的数据类型 usertype，删除的语法如下：

```
DROP TYPE usertype;
```

对于自定义数据类型的应用，还将在下面的小节中详细讲解。

3.2 创建数据表

视频讲解

数据表在数据库中的地位就好像人的器官一样重要。数据库中如果一张数据表都不存在，那么数据库也就没有了存在的意义。在 SQL Server 2017 中除了一般的关系数据表外，还允许创建文件表、图形表、外部表。既然数据表如此重要，就让我们先学习数据表如何创建。

3.2.1 创建一般数据表的语法

所谓一般数据表，也是数据库中最常用的一类数据表，主要涉及的内容就是列名和数据类型。创建数据表的语法非常复杂，语句也非常多。在本小节中先学习创建数据表的基本语法格式，具体的语法如下：

```
CREATE TABLE table_name
(
  column_name1 datatype,
  column_name2 datatype,
  ...
);
```

- table_name：表名。在一个数据库中数据表的名字不能重复，且数据表不能用数字来命名。通常要将表名声明成有实际意义的名字，增强可读性。
- column_name1：列名。表中的列名也是不能重复的。
- datatype：数据类型。允许使用系统自带的数据类型也可以是用户自定义的数据类型。

3.2.2 创建简单数据表

有了在 3.2.1 小节中讲解的语法，就可以创建数据表了。但是，这个数据表只是最简单的一种形式，只有列名和数据类型没有其他的设置。不管多么简单的一张表，都要先弄清楚表中的列名和数据类型。假设要完成一张用户信息表的创建，表的列名和数据类型用表 3.5 表示。

表 3.5　用户信息表(userinfo)

编　　号	列　　名	数 据 类 型	说　　明
1	id	int	编号
2	name	varchar(20)	用户名
3	password	varchar(10)	密码
4	email	varchar(20)	邮箱
5	QQ	varchar(15)	QQ 号码
6	tel	char(11)	手机号

从表 3.5 可以看出,除了编号外都设置成了字符串类型。但是,字符串类型的长度设置略有不同。编号用整数表示,可以设置成自动增长的,以避免用户编号重复。为什么用户编号不能够重复呢? 其实,这就是为了避免出现多条重复的记录,如果重复的话就很难判断是哪个用户了。这就好像是每个人都共用同一个卡号的银行卡,那么如何知道给谁发工资了呢? 谁花钱了呢? 当然,也可以将其他字段设置成不重复的,使用第 4 章中介绍的唯一约束就可以很容易设置了。下面用例 3-2 演示如何创建用户信息表。

例 3-2　根据表 3.5 的列名信息创建用户信息表(userinfo)。

根据题目要求,创建用户信息表的语句如下所示。这里在 chapter3 数据库中创建数据表。如果没有 chapter3 数据库,请读者自行创建一个名为 chapter3 的数据库。本章所有数据表都将创建在该数据库中。

```
USE chapter3                    -- 打开 chapter3 数据库
CREATE TABLE userinfo
(
  id          int,
  name        varchar(20),
  password    varchar(10),
  email       varchar(20),
  QQ          varchar(15),
  tel         char(11)
);
```

执行上面的语法,可以在 chapter3 数据库中创建 userinfo 数据表,执行效果如图 3.1 所示。

图 3.1　创建表 userinfo

3.2.3 创建带标识列的数据表

所谓标识列，也可以称为自动变化值的列，是让字段按照某一个规律增加，这样可以做到该列的值是唯一的。在 SQL Server 数据库中，设置带标识列的前提是该列是一个整数类型的数据。另外在设置标识列时，还需要指定初始值以及每次增长多少共两个参数，需要注意的是，在设置增长值时既可以设置负数也可以设置正数，设置负数表示从初始值基础上递减，设置正数则表示从初始值基础上递增。具体的设置方式如下。

```
IDENTITY(minvalue,increment)
```

- minvalue：最小值，也可以说是该列第一个要使用的值。默认情况下是从 1 开始的。
- increment：每次增加值。默认情况下也是每次加 1。

如果采用默认的从 1 开始每次增加 1 的自增长方式，需要在列的定义后面直接使用 IDENTITY 关键字设置。

例 3-3 根据表 3.5 的字段信息创建用户信息表(userinfo1)，并将该表中的编号列(id)设置成自动增长列，编号从 1 开始每次增加 2。

根据题目要求，具体的创建表语句如下所示。该表仍然创建在 chapter3 数据库中。由于数据库中已经存在了 userinfo 的数据表，因此表的名字定义成 userinfo1。

```
USE chapter3                        -- 打开 chapter3 数据库
CREATE TABLE userinfo1
(
  id          int IDENTITY(1,2),   -- 设置自动增长字段
  name        varchar(20),
  password    varchar(10),
  email       varchar(20),
  QQ          varchar(15),
  tel         char(11)
);
```

执行上面的语法，在 chapter3 中创建表 userinfo1，执行效果如图 3.2 所示。

图 3.2 创建表 userinfo1

通过上面的例子，可以知道 IDENTITY 这个关键字放在什么位置。就是放在设置成标识列的数据类型的后面。

3.2.4 创建带自定义数据类型的数据表

如果要在表 3.5 所示的列信息中使用自定义数据类型，应该将哪些列设置成自定义数据类型呢？在表 3.5 中有两个列的数据类型使用了 varchar(20)，将 varchar(20)可以定义成一个自定义的数据类型。这样，不仅在这个表中，在整个数据库里如果再需要这种数据类型时，都可以直接使用自定义的数据类型。实际上，经常会将一个或多个表中经常出现的数据类型定义成自定义数据类型。

例 3-4 根据表 3.5 创建用户信息表(userinfo2)并使用用户自定义类型 usertype1。在创建用户信息表之前，先创建一个自定义数据类型 usertype1，类型是 varchar(20)。

根据题目要求，先创建自定义数据类型 usertype1，具体的语法如下：

```
USE chapter3
CREATE TYPE usertype1
FROM varchar(20);
```

执行上面的语法，在 chapter3 中创建了一个名为 usertype1 的数据类型。

在创建用户信息表(userinfo2)时，使用 usertype1 数据类型，具体的语法如下：

```
USE chapter3                          -- 打开 chapter3 数据库
CREATE TABLE userinfo2
(
  id       int,
  name     usertype1,
  password varchar(10),
  email    usertype1,
  QQ       varchar(15),
  tel      char(11)
);
```

执行上面的语法，在数据库 chapter3 中创建了表 userinfo2，执行效果如图 3.3 所示。

图 3.3 创建表 userinfo2

3.2.5 在其他文件组上创建数据表

前面的例 3-2～例 3-4 创建的数据表都存放在 chapter3 数据库的主文件组中了。在第 2 章学习数据库的操作时提到过在一个数据库中可以有多个文件组，但是只有一个主文件组，默认情况下数据文件都会存放在主文件组中，也可以指定文件存放在其他文件组中。不仅是数据文件，数据表也是可以指定存放的文件组的。具体的语法如下：

```
CREATE TABLE table_name
(
  column1_name datatype,
  column2_name datatype,
      …
)
ON filegroup_name;
```

这里 filegroup_name 是文件组的名字。

例 3-5 根据表 3.5 所示的字段信息创建用户信息表(userinfo3)，并将该数据表创建在 chapter3 数据库的 filegroup 文件组中。

根据题目要求，假设在创建 chapter3 数据库时，添加了文件组 filegroup。具体的语法如下：

```
USE chapter3                -- 打开 chapter3 数据库
CREATE TABLE userinfo3
(
  id        int,
  name      varchar(20),
  password  varchar(10),
  email     varchar(20),
  QQ        varchar(15),
  tel       varchar(15)
)
ON filegroup;
```

执行上面的语法，在 chapter3 数据库的 filegroup 文件组里创建了 userinfo3 数据表，执行效果如图 3.4 所示。

图 3.4 创建表 userinfo3

3.2.6 创建临时表

所谓临时表指不是数据库中永久存在的表，只是临时保存数据时使用的。临时表又分为本地临时表和全局临时表。本地临时表是以“#”开头的数据表，在当前登录用户下可用；全局临时表是以“##”开头的数据表，所有用户都可以使用。临时表的创建语法与一般的数据表创建是一样的，只是临时表通常都存放在 tempdb 数据库中。

例 3-6 创建一个临时表(#temptable)，表中的列信息如表 3.6 所示。

表 3.6 用户信息临时表(#temptable)

编 号	字 段 名	数 据 类 型	说 明
1	id	int	编号
2	name	varchar(20)	用户名
3	password	varchar(10)	密码

根据题目要求，创建临时表 temptable 的语法如下：

```
CREATE TABLE #temptable
(
  id        int,
  name      varchar(20),
  password  varchar(10)
)
```

执行上面的语法，在 tempdb 数据库中创建了一个名为#temptable 的临时表。执行效果如图 3.5 所示。

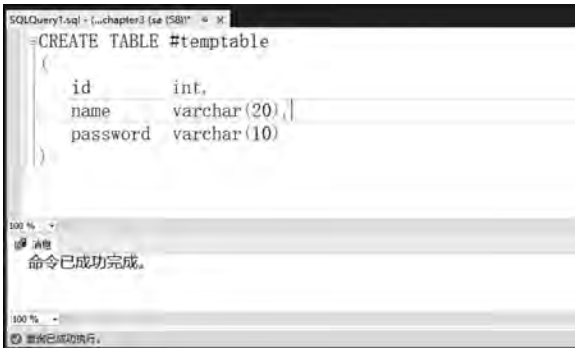

图 3.5 创建临时表#temptable

注意：虽然当前打开的数据库是 chapter3，但是临时表依然创建在 tempdb 数据库中。

3.2.7 使用 SSMS 创建数据表

创建数据表需要记住的语法比较多，现在有一个简单的方法创建数据表，既不用担心忘记数据类型的名称又不用怕记不住语法，这个方法就是使用 SSMS。下面例 3-7 是在 SSMS 中使用 SQL 语句创建用户信息表的过程。

例 3-7 在 SSMS 中，根据表 3.5 所示的用户信息表的列信息创建用户信息表 userinfo4 和 userinfo5，并分别按如下两个要求完成设置。

(1) 设置用户信息表中的编号列为标识列。

(2) 使用用户自定义的数据类型 usertype。

根据题目要求，要在 SSMS 中创建数据表，在本例中仍然将表创建在数据库 chapter3 中。无论创建的数据表有什么要求，都需要在表的设计页面中完成。下面先打开表的设计页面。在 SSMS 的“对象资源管理器”中找到 chapter3 数据库并展开文件夹，然后再右击其中的“表”节点，在弹出的快捷菜单中选择“新建”→“表”选项，出现如图 3.6 所示界面。

图 3.6 表设计器界面

图 3.6 中所示的界面是创建数据表的操作界面，被称为数据表的设计界面。所有关于数据表的操作都在该界面完成。下面使用该界面分别完成本题的两个小题。

1. 标识列

根据题目要求，要先录入用户信息表的基本内容，然后将表中的编号列设置为标识列。完成这个要求分为如下 3 个步骤。

(1) 按照表 3.5 的要求，录入用户信息表的信息。在图 3.6 所示的界面中录入用户信息表的列名和数据类型，其中数据类型可以通过下拉列表选择，录入后效果如图 3.7 所示。

在图 3.7 中可以看到，数据类型后面还有一列“允许 Null 值”，该列是做什么的呢？正如字面的意思就是设置该列是否允许不输入值，默认情况下，将其选中即可以不输入值。这种是否为空的限制也被称为非空约束。关于非空约束的定义将在第 4 章中详细讲解。

(2) 设置编号(id)列为标识列。设置某一列为标识列时，该列的数据类型必须是整数。在 SSMS 中设置标识在列的属性界面中完成。在图 3.7 所示界面中单击 id 所在的行，找到 id 的列属性，如图 3.8 所示。

图 3.7　录入用户信息表信息后的效果

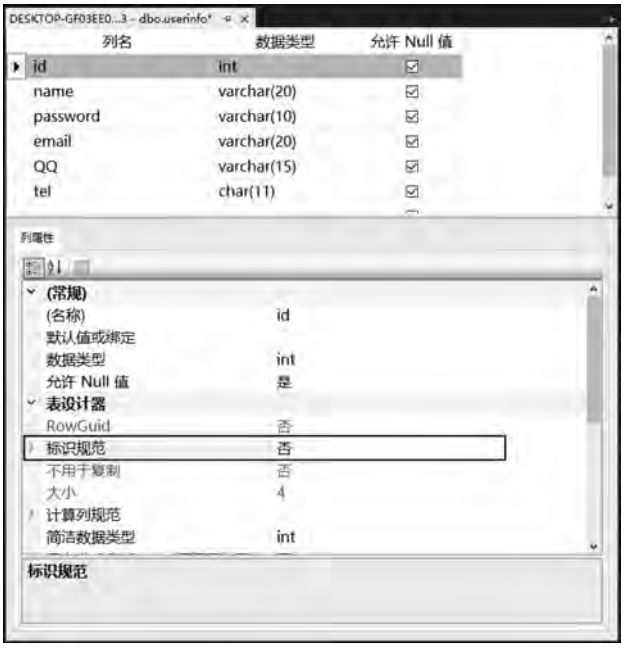

图 3.8　id 的列属性界面

在图 3.8 所示的界面中，“标识规范”选项就是用来设置标识列的。该选项值设成“是”，标识列就是设置成自动增长的；如果不加其他的设置，设置的标识列就是从 1 开始每次增加 1。双击图 3.8 中的“标识规范”选项（方框内选项）可以设置标识列的起始值以及增量，本例中起始值设置成 1，增量设置成 10，效果如图 3.9 所示。

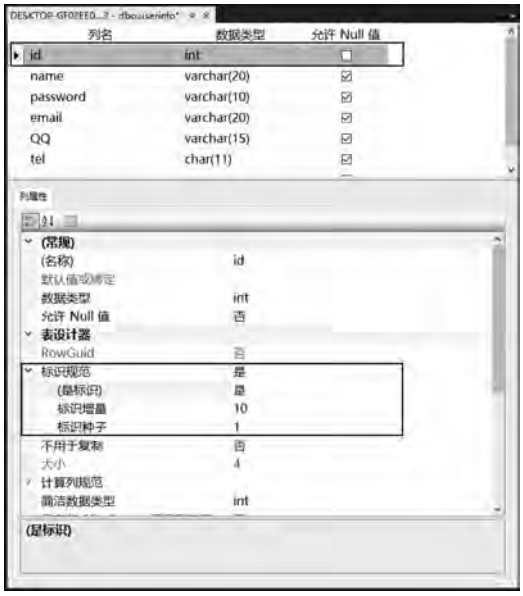

图 3.9 设置标识列

在图 3.9 中,“标识增量”选项是要设置的增量,“标识种子”选项是初始值。在将 id 设置成标识列后,该列中的“允许 Null 值”列就去除了选中状态,也就是不允许为空了。因为设置为标识列后是不会产生空值的。

(3) 给表命名。在完成了表的信息添加和标识列设置后,要保存表的信息。保存表的方法有很多,这里介绍几个常用的方法。最简单的方法是像保存文件一样用 Ctrl+S,还可以使用工具栏上的按钮来保存表信息。除了上面的两个方法外,用文件菜单下的“保存”选项也能完成保存的操作。不论使用哪种方法保存表的信息,都会弹出如图 3.10 所示的对话框。

在图 3.10 所示对话框中输入表的名称,这里按要求将默认的“表 1”更换为 userinfo4,单击“确定”按钮,即可完成对 userinfo4 的创建操作。还有一点需要注意,那就是名字不要与 chapter3 中的数据表重名。完成保存操作后,在 chapter3 数据库中的“表”节点下,就会出现 userinfo4 表的名字了,如图 3.11 所示。

图 3.10 保存表信息对话框

图 3.11 表节点下的 userinfo4

2. 用户自定义的数据类型

在设置自定义数据类型之前先要创建自定义数据类型。先创建一个用户自定义的数据类型，然后再使用该数据类型。具体操作分为如下两个步骤完成。

(1) 创建自定义数据类型 usertype。创建自定义数据类型前先要找到创建用户定义数据类型的位置，它就在 chapter3 数据库中的“可编程性”节点的“类型”节点里，如图 3.12 所示。

在图 3.12 所示界面中右击“用户定义数据类型”选项，在弹出的快捷菜单中选择“新建用户定义数据类型”选项，弹出如图 3.13 所示对话框。

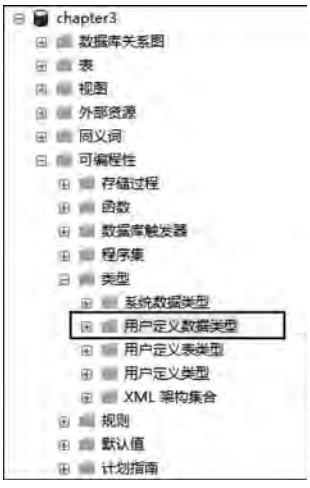

图 3.12　用户定义数据类型的位置

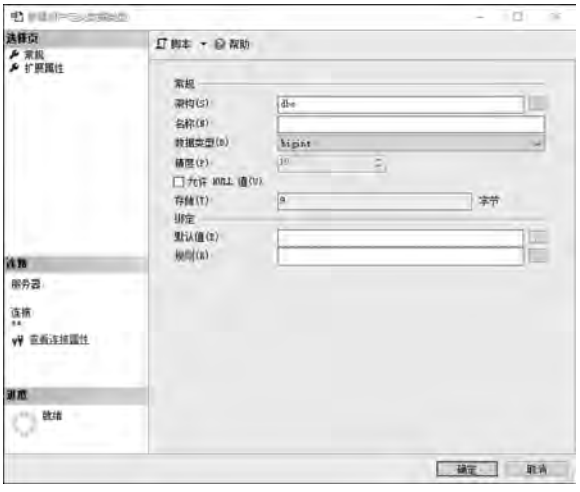

图 3.13　新建用户定义数据类型

在图 3.13 所示对话框中输入自定义数据类型的名称，然后选择一个数据类型并输入该数据类型的长度。这里自定义数据类型的“名称”是 usertype，“数据类型”是 varchar，“长度”是“20”，输入后的效果如图 3.14 所示。

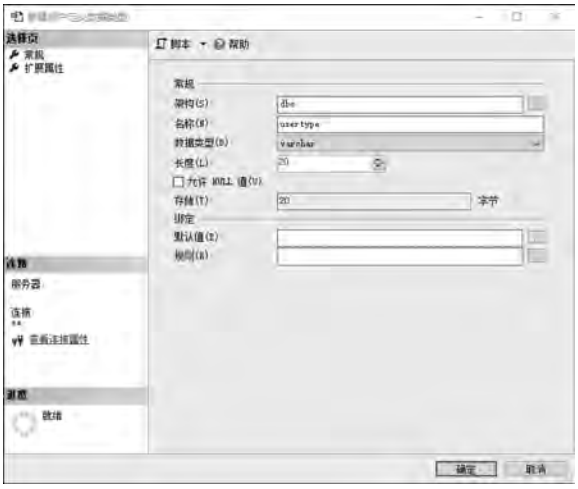

图 3.14　输入自定义类型后的效果

在图 3.14 所示对话框中单击“确定”按钮，即可完成自定义数据类型添加操作。

(2) 在表设计器中使用自定义数据类型。设置好自定义数据类型后，再使用该数据类型时与系统的数据类型是一样的。自定义的数据类型也会出现在表设计器数据类型的下拉列表框中。与(1)的方法一样，先将用户信息表的列信息输入表设计器中，然后将需要使用varchar(20)数据类型的列设置成用户自定义数据类型即可。完成操作后，将表的名称保存成 userinfo5。具体的操作与(1)的方法相同，表设计器的数据类型显示如图 3.15 所示。

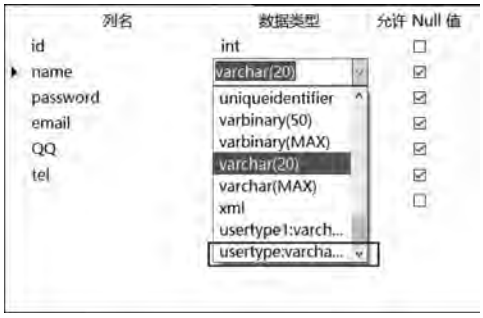

图 3.15 自定义数据类型显示的位置

从图 3.15 中可以看出，自定义的数据类型显示在数据类型列表的最后面。

通过例 3-7 的讲解，可以发现使用 SSMS 创建数据表还是很方便的。

3.2.8 使用 sp_help 查看数据表

数据表创建好后，如何查看数据表呢？用 SSMS 很容易查看数据表了。如果不使用 SSMS 如何查看数据表的信息呢？有很多方法，先介绍使用 sp_help 存储过程查看数据表的信息，查看的语法如下：

```
sp_help table_name;
```

table_name 是数据表的名称。在查看数据表之前，不要忘记用 USE 语句指定要使用的数据库。

例 3-8 使用存储过程 sp_help 查看用户信息表(userinfo)的表信息。

根据题目要求，查看的语法如下：

```
sp_help userinfo;
```

执行上面的语法，效果如图 3.16 所示。

在图 3.16 中查询结果分成 5 个部分，下面分别说明这 5 部分显示的信息是什么。

- 第 1 部分：显示表创建时的基本信息，包括数据表的名称、类型、创建时间以及拥有者。
- 第 2 部分：显示表中列的信息，包括列的名称、数据类型和长度等信息。
- 第 3 部分：显示表中标识列的信息。在表 userinfo 中 id 设置为标识列，起始值为 1，增量为 10。
- 第 4 部分：显示表中的全局唯一标识符列。在每一个数据表中只能有一个全局唯一

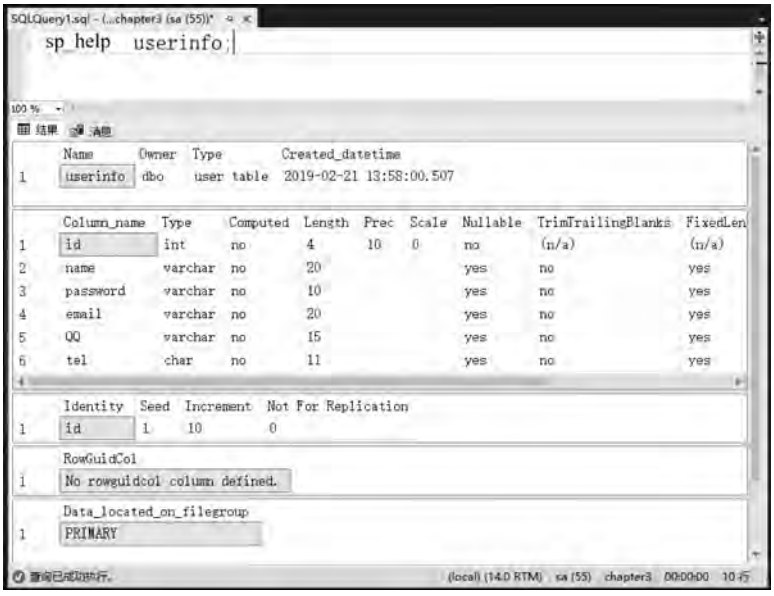

图 3.16　userinfo 表的信息

标识符列。在表 userinfo 中没有设置全局唯一标识符列。

- 第 5 部分：显示表存在的文件组。在本图中显示的是 userinfo 存放在主文件组(PRIMARY)中。

说明：使用存储过程 sp_help 不仅可以查看表的信息，也可以查看数据库的其他对象以及用户自定义的数据类型等信息。查询方法很简单，只需要执行 sp_help 存储过程即可，而不需要再添加表名了。直接使用 sp_help 查询的效果如图 3.17 所示。

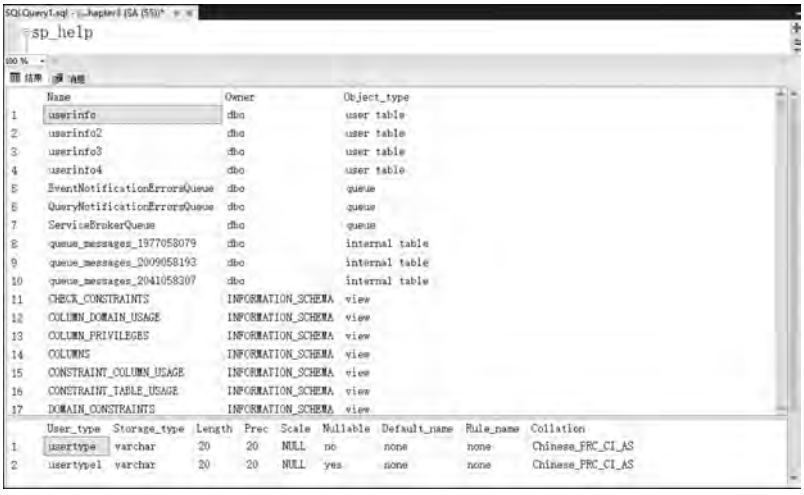

图 3.17　sp_help 查询的效果

图 3.17 的查询结果由两个部分组成，一部分用于显示数据库 chapter3 中所有的数据对象信息，另一部分用于显示数据库 chapter3 中自定义的数据类型信息。

3.2.9 使用 sys.objects 查看数据表

如果只需要知道数据表的创建信息，有一个相对简单点的方法，那就是使用系统表 sys.objects 查看。下面使用例 3-9 演示如何使用 sys.objects 查看表的信息。

例 3-9 使用系统表 sys.objects 查看 userinfo 表的信息。

根据题目要求，查看 userinfo 表的信息的 SQL 语句如下：

```
SELECT * FROM sys.objects WHERE name = 'userinfo';
```

执行上面的语法，将 userinfo 表的创建信息显示出来，效果如图 3.18 所示。

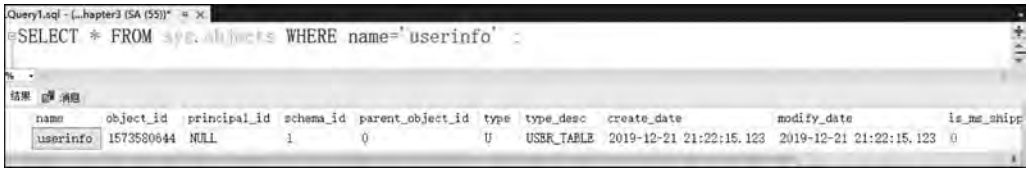

图 3.18 使用 sys.objects 查看 userinfo 表信息

从图 3.18 的查询结果中可以看出，使用 sys.objects 系统表可以查看到 userinfo 表的创建时间、修改时间以及表的类型等信息。学习了查看 userinfo 表的信息，发现在 sys.objects 中的 name 列就是要查找的数据表名称，也就是说，要查询哪个数据表信息就将 name 后的表名改成要查找的数据表即可。当然，也可以使用 sys.objects 系统表不加任何条件查看数据库中所有的数据表信息。

3.2.10 使用 information_schema.columns 查看数据表

在前面的两个小节中分别使用了存储过程和系统表查看表的信息，除了这两个方法外，还可以使用系统视图 information_schema.columns 查看表的信息。通过这个系统视图可以查看出表中的列信息，但是不包括表的创建信息了。

例 3-10 使用 information_schema.columns 查看 userinfo 表的信息。

根据题目要求，查看 userinfo 表的信息语法如下：

```
SELECT * FROM information_schema.columns WHERE table_name = 'userinfo';
```

执行上面的语法，效果如图 3.19 所示。

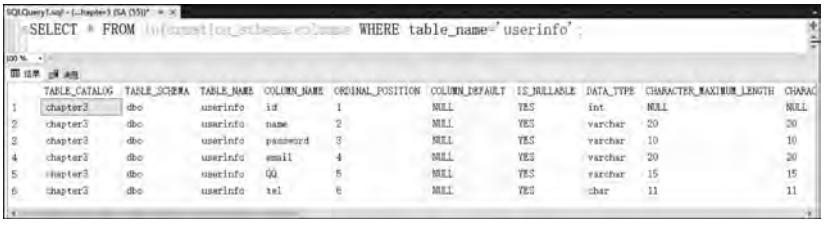

图 3.19 使用 information_schema.columns 查看 userinfo 表

从图 3.19 中可以看出，通过 information_schema.columns 视图可以查看到 userinfo 表所属的数据库名、列名以及列的数据类型等信息。

说明：前面讲过的使用 sp_help、sys.objects、information_schema.columns 三种方式查询表信息的方式，它们都在什么时候使用呢？sp_help 主要用于查询表中所有的信息，包括表的创建信息、列信息以及其他的信息；sys.objects 主要用于查询表的创建信息；information_schema.columns 用于查询表的列信息。

视频讲解

3.3 修改数据表

创建好的数据表可以修改它的哪些内容呢？其实所有内容都是可以修改的，包括修改列的数据类型、添加或减少表中的列、修改表中列的定义以及给表改名等。另外，还有一个好工具可以帮助我们修改数据表，也还是 SSMS。在本小节中，将为读者详细讲述如何修改数据表。

3.3.1 修改表中列的数据类型

修改数据类型是一件很容易的事情，请看下面的语法。

```
ALTER TABLE table_name
ALTER COLUMN column_name datatype;
```

- table_name：表名。要修改的数据表名，在执行修改语句时，也要先使用 USE 语句打开表所在的数据库。
- column_name：列名。数据表中的列名。如果不清楚表中的列名，可以先查看表的信息再进行修改。
- datatype：数据类型。给表中列新设置的数据类型。能够设置新类型的前提是该列中存放的值能够兼容新设置的类型。通常都是在表中还没有存放数据时修改数据类型。

例 3-11 修改用户信息表(userinfo)，将其中的用户名列(name)的数据类型改成 varchar(30)。

根据题目要求，具体的语法如下：

```
USE chapter3                    -- 打开 chapter3 数据库
ALTER TABLE userinfo
ALTER COLUMN name varchar(30);
```

执行上面的语句，将用户信息表(userinfo)中的姓名列(name)的数据类型更改成 varchar(30)了，执行效果如图 3.20 所示。

图 3.20 更改 name 列的长度

3.3.2 修改表中列的数目

如果一张数据表创建好了，根据实际的项目需求需要添加或删除一些列，可以用到修改表中列数目的语句了。修改表中列数目的语句比较简单。

(1) 向表中添加列。

```
ALTER TABLE table_name
ADD column_name datatype;
```

- table_name：表名。
- column_name：新添加的列名。列名不能与表中已经存在的列重名，因此在添加列时最好先查看表中现有列的信息。
- datatype：数据类型。

(2) 删除表中的列信息。

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

- table_name：表名。
- column_name：列名。删除后的列不能恢复，在删除前要考虑清楚。

例 3-12 向用户信息表(userinfo)中添加一个备注列(remark)，数据类型是 varchar(50)。

根据题目要求，查看了用户信息表后，发现备注列(remark)在用户信息表中没有重名，具体的语法如下：

```
USE chapter3;
ALTER TABLE userinfo
ADD remark varchar(50);
```

执行上面的语句，在用户信息表 userinfo 中增加了备注列(remark)，执行效果如图 3.21 所示。

图 3.21 添加 remark 列

例 3-13 删除在例 3-12 中为用户信息表(userinfo)添加的备注列(remark)。

根据题目要求，删除列的语法如下：

```
USE chapter3;
ALTER TABLE userinfo
DROP COLUMN remark;
```

执行上面的语句，将用户信息表(userinfo)中的备注列(remark)删除了，执行效果如图 3.22 所示。

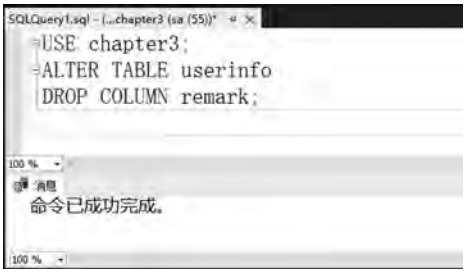

图 3.22　删除 remark 列

3.3.3　给表中的列改名

要想给表中的列改名，用 ALTER 语句不可以。那用什么语句给表中的列改名呢？在第 2 章中修改数据库的名字是用存储过程 sp_rename 完成的。这里也可以通过 sp_rename 修改列的名字，具体的语法如下：

```
sp_rename 'tablename.columnname','new_columnname';
```

- tablename.columnname：原来表中的列名。表中的列名要加上单引号。
- new_columnname：新列名。新列名也要加上单引号，并且不能与其他的列名重名。

例 3-14　将用户信息表(userinfo)中的用户名(name)列的名字更改成 username。

根据题目要求，对用户名列改名，username 在用户信息表中没有与之重复的，修改语法如下：

```
sp_rename 'userinfo.name','username';
```

执行上面的语法，将用户信息表(userinfo)中的用户名(name)列的名字改成 username 了，执行效果如图 3.23 所示。

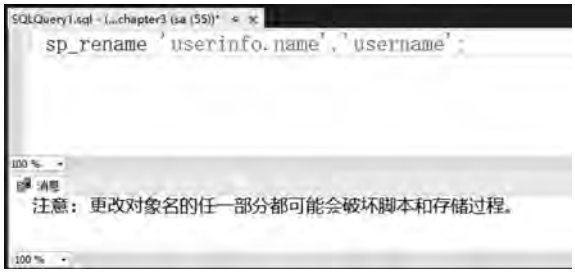

图 3.23　将列名 name 更改成 username

在图 3.23 中，虽然没有看到“执行成功”的字样，但是列名已经更改了。读者可以利用 SELECT * FROM information_schema.columns WHERE table_name='userinfo'语句查看效果。

3.3.4 重命名表

既然表中的列名可以更改，数据表的名字可以改吗？答案是肯定的，但是也不可以用 ALTER 语句修改。可用 sp_rename 语句来修改，这回修改语句与修改数据库名字的语句类似了，具体用法如下：

```
sp_rename old_tablename,new_tablename;
```

这里，old_tablename 是原来的表名，new_tablename 是修改后的表名。在更改表名的时候也要确认数据库中是否已有要创建的表名。另外，在修改表名之前还要将该表存在的数据库用 USE 打开。

例 3-15 将用户信息表(userinfo)的名字修改成(newuserinfo)。

根据题目要求，新表名 newuserinfo 在 chapter3 数据库中不存在，修改的语法如下：

```
sp_rename userinfo,newuserinfo;
```

执行上面的语句，在 chapter3 中就没有名为 userinfo 的数据表了，多了一个名字 newuserinfo 的数据表，执行效果如图 3.24 所示。

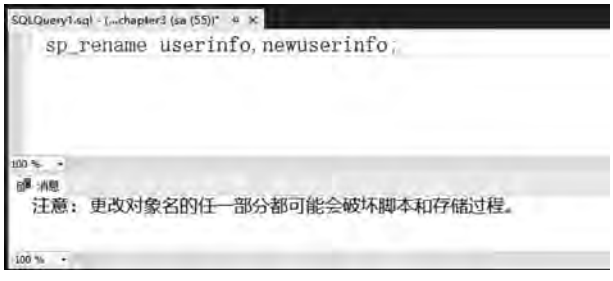

图 3.24 将表 userinfo 更改成 newuserinfo

3.3.5 使用 SSMS 修改表

有了在 SSMS 中创建表的基础，使用 SSMS 修改数据表就很容易了。下面就用例 3-16 演示如何在 SSMS 中修改数据表。

例 3-16 在 SSMS 中，按照如下要求修改用户信息表 userinfo。

(1) 向用户信息表添加备注列(remark)，数据类型是 varchar(50)。

(2) 将备注列(remark)的数据类型修改成 varchar(100)。

(3) 将备注列(remark)重命名成 remarks。

(4) 删除备注列(remarks)。

根据题目要求，这 4 个小题都将在 SSMS 中用户信息表(userinfo)的表设计界面完成。在 SSMS 中的“对象资源管理器”中，展开 chapter3 数据库节点，在其中的“表”节点下右击

userinfo 表，在弹出的快捷菜单中选择“设计”选项，如图 3.25 所示。

下面在图 3.25 所示的界面中，按照题目的顺序演示如何修改数据表。

(1) 添加列操作。在图 3.25 所示的界面中，tel 所在行的下一个空行，单击“列名”所对应的单元格，并录入列名 remark、数据类型 varchar(50)即可，录入后的效果如图 3.26 所示。

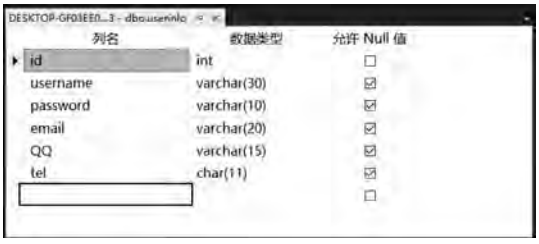

图 3.25　userinfo 表的设计界面

图 3.26　添加字段 remark 的效果

添加列后，记得要保存表的信息。

(2) 修改 remark 字段的数据类型在图 3.26 的界面中操作就可以了，直接将 remark 列的数据类型 varchar(50)改成 varchar(100)，效果如图 3.27 所示。

同样，也要将图 3.27 修改后的结果保存后才完成修改操作！

(3) 重命名 remark 列，与修改 remark 列的数据类型相似，在图 3.27 所示的界面中操作就可以了。操作方法是直接单击 remark 所在的单元格，然后将其改成 remarks 即可，效果如图 3.28 所示。

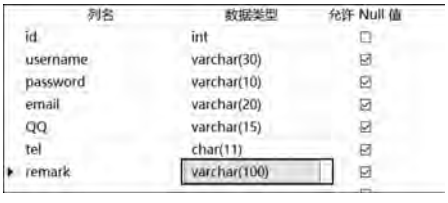

图 3.27　修改 remark 字段的数据类型

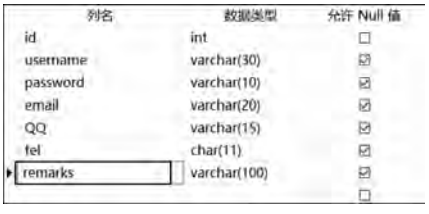

图 3.28　重命名 remark 字段

修改 remark 列的名称，保存修改后的内容即可。

(4) 删除 remarks 字段在图 3.28 的界面中操作就可以，右击 remarks 字段所在的行，在弹出的快捷菜单中选择“删除列”选项，即可将该列删除。删除后不要忘记保存表信息。

说明：在例 3-16 中讲解了 SSMS 中修改表的一些操作，除了这些操作外，还可以进行将表中的列删除，把列设置成标识列等操作。

视频讲解

3.4　删除数据表

当不再需要某一个数据表时，也可以通过 SQL 语句或者 SSMS 将其删除。但是删除后的数据表很难恢复。因此，在删除数据表前一定要先备份数据表，以免带来不必要的损失。

3.4.1 删除数据表的语法

删除数据表的语法比前面介绍的创建、修改数据表的语法简单，记住 DROP 关键字。创建和修改数据表时，每次只能创建或修改一张数据表，但是删除的时候可以一次删除多张数据表，删除数据表的语法如下：

```
DROP TABLE database_name.table_name1, database_name.table_name2,...
```

- database_name：数据库名。如果已经把表所在的数据库打开了，那么，数据库名就可以省略了。但是，要删除其他数据库的表，就要加上数据库名。
- table_name：表名。

3.4.2 使用 DROP 语句删除多余的表

通过学习 DROP TABLE 删除数据表的语句，读者已经知道如何删除数据表。下面通过几个例子使用 DROP TABLE 语句。

例 3-17 删除 chapter3 数据库中的 userinfo。

根据题目要求，具体的语法如下：

```
USE chapter3
DROP TABLE userinfo;
```

执行上面的语法，userinfo 表从数据库 chapter3 中移除了，执行效果如图 3.29 所示。

该例不仅可以使用上面的语句删除表 userinfo，也可以使用 DROP TABLE chapter3.userinfo 语句完成。

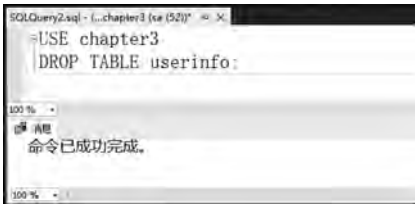

图 3.29 删除表 userinfo

例 3-18 同时删除 chapter3 数据库中的 userinfo2 和 userinfo3。

根据题目要求，一次要删除 2 张数据表，具体的语法如下：

```
USE chapter3;
DROP TABLE userinfo2, userinfo3;
```

执行上面的语法，将表 userinfo2 和 userinfo3 从数据库 chapter3 中移除了，执行效果如图 3.30 所示。

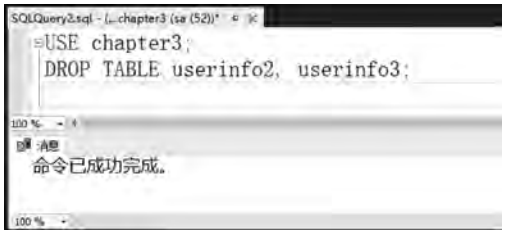

图 3.30 删除表 userinfo2 和 userinfo3

注意：如果要删除的数据表在数据库中不存在，在执行删除语句后，会出现如图 3.31 所示的错误提示。

图 3.31　删除不存在的表

3.4.3　使用 SSMS 删除数据表

现在就来讲解在 SSMS 中最简单的一个表操作，那就是删除数据表。在 SSMS 中删除数据表时不需要使用表设计器，直接使用鼠标就可以操作。下面通过例 3-20 演示删除操作。

例 3-19　在 SSMS 中删除 userinfo 表。

在“对象资源管理器”中，右击要删除的数据表 userinfo，在弹出的快捷菜单中选择“删除”选项，如图 3.32 所示，弹出如图 3.33 所示对话框。

图 3.32　选中待删除表并右击鼠标

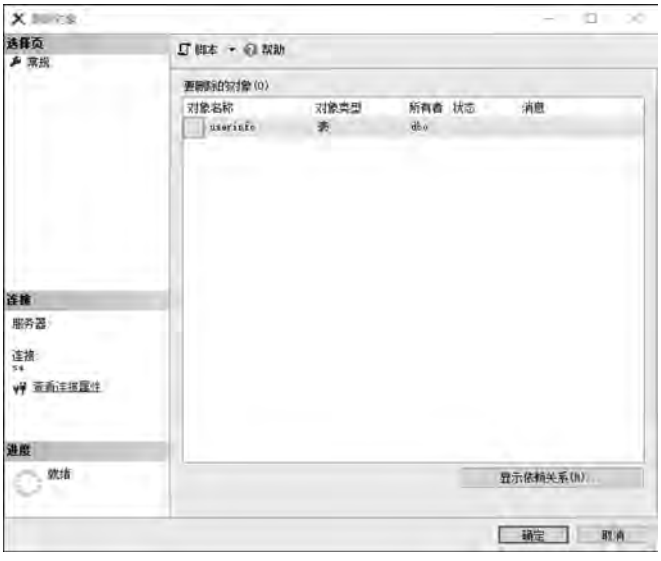

图 3.33　删除表对话框

在图 3.33 所示对话框中单击“确定”按钮，即可将 userinfo 表删除了。

3.5 本章小结

本章主要讲解了 SQL Server 里数据表中列的数据类型，使用 SQL 语句和使用 SSMS 创建、修改及删除数据表。其中，在讲解数据类型时着重讲解了整型、浮点型以及字符串类型的使用。在创建数据表部分除了讲解数据表的创建，还讲解如何使用存储过程 sp_help、系统表 sys.objects 及系统视图 information_schema.columns 查看数据表的信息。如果想成为一个优秀的数据库管理器员，还要记牢 SQL 语句，不能全靠 SSMS 操作。

3.6 本章习题

一、填空题

1. sp_help 的作用是_____。
2. 为表重命名使用的存储过程是_____。
3. 临时表是以_____为前缀的。

二、选择题

1. 下面对数据表描述正确的是(　　)。
   A. 在 SQL Server 中，一个数据库中可以有重名的表
   B. 在 SQL Server 中，一个数据库中表名是唯一的
   C. 数据表通常都以数字来命名
   D. 以上都不对
2. 下面对创建表的描述正确的是(　　)。
   A. 可以使用 CREATE 语句创建不带列的空表
   B. 在创建表时，就可以为表设置标识列
   C. 在创建表时，列名可以重复
   D. 以上都对
3. 下面对修改数据表的描述正确的是(　　)。
   A. 可以修改表中列的数据类型
   B. 可以删除表中的列
   C. 可以将表重命名
   D. 以上都对

三、操作题

创建名为 test 的数据表(自定义表结构)，并对表做如下操作。

(1) 将表中的第 1 个列删除。

(2) 给表中的第 2 列改名。

(3) 将表 test 的名字更改成 testone。