第 3 章　操作存储数据的单元

在第 2 章中已经讲解了数据库的一些基本操作，那么，数据库中的数据是如何存放的呢？数据库就相当于是一个文件夹，在一个文件夹中可以存放多个文件。数据库中的文件被称为数据表，也就是用来存储数据的容器。一个数据库由若干张数据表组成，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。

本章的主要知识点如下：

- 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　认识表中能存放什么样的数据

读者可以思考一下，当我们在网站上注册一个用户信息的时候，都会输入哪些数据呢？通常会有用户名、密码、邮箱、年龄和联系方式等等。只要是注册时填入的数据，最终都将提交到数据库中存放。想想这些数据都包含什么呢？输入注册信息的时候会有汉字、数字、字母以及特殊符号等。既然这些数据都够存到数据库中，也就是说数据表中应该能够存放这些类型的数据。在本节中将详细讲解 SQL Server 数据表中使用的数据类型。

3.1.1　整型和浮点型

整型和浮点型实际上都属于数值类型，也就是用来存放数字的一种类型。这个类型在日常生活中用得是比较多的，读者想一想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等。这样看来整数和浮点数很重要喽，那就让我们看看在 SQL Server 数据库中究竟整数和浮点数用什么数据类型名表示的吧！首先，来学习一下表 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	$0 \sim 2^8-1$	表示小整数，占 1 个字节
smallint	$-2^{15} \sim 2^{15}-1$	表示短整数，占 2 个字节
int	$-2^{31} \sim 2^{31}-1$	表示一般整数，占用 4 个字节
bigint	$-2^{63} \sim 2^{63}-1$	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。在实际的应用中，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。这就像你在超市结账时，根据选择物品的多少来购买购物袋一样。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。

接下来让我们再认识一下表 3-2 所示的浮点型吧。

表 3-2　浮点型

数据类型	取值范围	说　　明
numeric(m,n)	$-10^{38}+1$~$10^{38}-1$	表示$-10^{38}+1$~$10^{38}-1$ 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)表示长度为 7 的数，并保留 2 位小数
decimal(m,n)	$-10^{38}+1$~$10^{38}-1$	与 numeric(m,n)的用法相同
real	-3.40E+38~3.40E+38	占用 4 个字节
float	-1.79E+308~1.79E+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。总之是要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来表示呢？实际上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来存储。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	说　　明
char(n)	1～8000 个字符	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的长度，系统会自动用空格填充到该长度
varchar(n)	1～8000 个字符	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)	1～2^{31}–1 个字符	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 字节
text	1～2^{31}–1 个字符	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 个字符	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长度，系统会自动用空格填充到该长度
nvarchar(n)	1～4000 个字符	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)	1～2^{31}–1 个字符	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 字节
ntext	1～2^{31}–1 个字符	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

续表

数据类型	取值范围	说　　明
binary(n)	1～8000 个字符	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 个字符	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	1～2^{31}–1 个字符	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上分为三类，一类是 1 个字符占用 1 个字节的字符串类型（char、varchar 和 text），一类是 1 个字符占用 2 个字节的字符串类型（nchar、nvarchar 和 ntext），一类是存放二进制数据的字符串类型（binary、varbinary 和 image）。在每一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使用可变长度的类型，这样可以节省数据的存储空间。

说明：在字符串类型中的 varchar(max)和 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 中还是准备了一套数据类型来专门用于表示日期时间。通过日期时间类型可以将日期时间表示得更加准确，在 SQL Server 中表示日期时间的数据类型主要有 datetime 和 smallldatetime 两种。具体的表示方法如表 3-4 所示。

表 3-4　日期时间型

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，但还要清楚的是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+分隔符的格式以及数字格式。下面就分别用这 3 种形式来表示 2012 年 5 月 1 日。

```
May 1 2012                    --英文+数字格式
2012-5-1                      --数字+分隔符格式
2012.5.1                      --数字+分隔符格式
2012/5/1                      --数字+分隔符格式
20120501                      --数字格式
120501                        --数字格式
```

看了上面的例子，你对日期类型的表示有所了解了吧。其实，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、日月年的顺序来表示日期类型的数据。例如：5-1-2012、1-5-2012 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格式都是按照“小时：分钟：秒.毫秒”来存储的。例如：表示上午的 9 点 10 分 20 秒，就可以用 9:10:20 来表示。时间的表示可以分为 24 小时和 12 小时两种格式，如果用的是 12 小时的格式，用 am 表示上午，用 pm 表示下午。例如：表示晚上的 10 点 40 分 10 秒，就可以用 10:40:10 pm 表示。

在存储日期时间数据时通常将日期和时间一起存储，这时就需要在日期格式后面加上一个空格然后加上时间格式来表示。例如：表示 2012 年 5 月 25 日下午 5 点 25 分 10 秒，就可以写成 2012-5-25 5:25:10pm。存储日期时间类型时，读者要注意的就是格式问题，另外，还要提醒读者在一个数据表中存储的日期时间格式要统一，否则在查询数据时就会给你造成一些不必要的麻烦的哦！

3.1.4　其他数据类型

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。比如：timestamp 类型、xml 类型和 cursor 类型等。timestamp 类型是时间戳类型，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；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，具体的语句如下：

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

其中：

- 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。定义自定义数据类型表示的数据类型，除了指定数据类型外，还可以指定该类型是否为空值。

【示例 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，可以选择的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。具体的语句如下：

```
CREATE TYPE usertype1
FROM varchar(20) not null;
```

通过上面的语句就可以为数据库添加一个数据类型 usertype1，在使用该类型时直接用 usertype1 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 1 中定义的数据类型 usertype1，删除的语句如下：

```
DROP TYPE usertype1;
```

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。请读者在下一节中认真

体会自定义数据类型的优势吧！

3.2　创建数据表

数据表在数据库中的地位，就好像是人的器官一样重要。如果人没有器官，那么人就是一个空架子毫无意义。如果数据库中一张数据表都不存在，那么数据库也就没有了存在的意义。既然数据表如此的重要，就让我们先学习一下数据表是如何创建的吧。

3.2.1　创建数据表的语句

创建数据表的语法是非常复杂的，语句也非常多。但是不用怕，咱们由浅入深慢慢来学。在本小节中先学习一下创建数据表的基本语法格式，如下：

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

其中：

- 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。
- column_name1：字段名。表中的字段名也是不能重复的。
- datatype：数据类型。它可以是系统的数据类型，也可以是用户自定义的数据类型。

3.2.2　试用 CREATE 语句创建简单数据表

有了在上一小节中讲解的创建数据表的语法，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，只有字段名和数据类型，没有其他的内容。不管是多么简单的一张表，都要先弄清楚表中的字段名和数据类型。假设要完成一张用户信息表的创建，表的字段名和数据类型用发表 3-5 所示。

表 3-5　用户信息表（userinfo）

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
4	email	varchar(20)	邮箱
5	QQ	varchar(15)	QQ 号码
6	tel	varchar(15)	电话号码

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。编号用整数来表示，可以给其设置成自动增长的，可以避免用户编号重复。那么，读者会问了，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地设置了。下面就用示例 2 来演示如何创建用户信息表。

【示例 2】 根据表 3-5 的字段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 数据库中来创建数据表。如果没有 chapter3 数据库，请读者自行创建一个名为 chapter3 的数据库。本章的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      varchar(15)
);
```

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

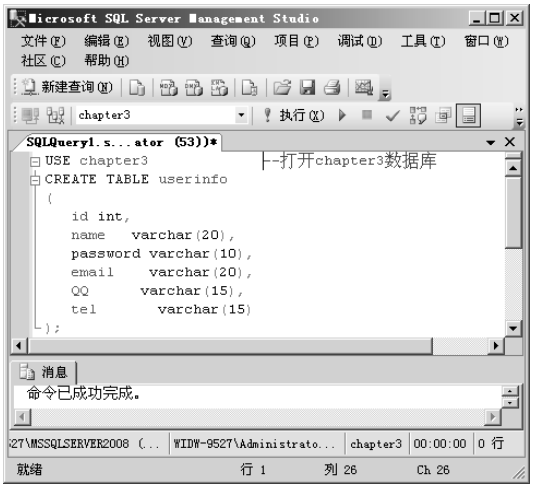

图 3.1　创建表 userinfo

3.2.3　创建带自动增长字段的数据表

所谓自动增长字段就是让字段按照某一个规律增加，这样就可以做到该列的值是唯一的。在 SQL Server 数据库中，设置带自动增长字段的前提是该字段是一个整数类型的数据。另外在设置自动增长字段时，还需要指定最小值以及每次增加多少个参数。具体的设置方式如下：

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

其中：

- 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。

如果要采用默认的从 1 开始每次增加 1 的自动增长方式，直接使用 IDENTITY 关键字设置即可，不需要再添加参数了。有了自动增长字段的设置方式，那么该语句应该写在什么位置呢？请读者先在示例 3 中自己找一找。

【示例 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。

根据题目要求，具体的创建表语句如下所示。仍然将该表创建在 chapter3 数据库中。由于该数据库中已经存在了 userinfo 的数据表，因此，将该表的名字定义成 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      varchar(15)
);
```

执行上面的语句，就可以在 chapter3 中创建表 userinfo1 了。执行效果如图 3.2 所示。

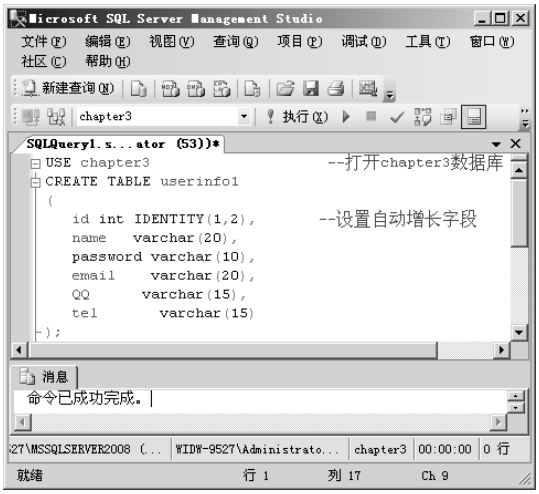

图 3.2　创建表 userinfo1

通过上面的例子，相信读者已经知道了 IDENTITY 这个语句放在什么位置了吧。没错，就是放在需要设置成自动增长列数据类型后面。

3.2.4　创建带自定义数据类型的数据表

读者可以思考一下，如果要在表 3-5 所示的字段信息中使用自定义数据类型，应该将哪些字段设置成自定义数据类型呢？在表 3-5 中有两个字段使用了 varchar(15)，有两个字段使用了 varchar(20)，可以分别将 varchar(15)的定义成一个数据类型，将 varchar(20)的定义成一个数据类型。这样，不仅在这个表中，在整个数据库里如果再需要这两种数据类型

时也可以直接使用自定义的数据类型。实际上，经常会将一个表或一个数据库中经常出现的数据类型定义成自定义数据类型。

【示例 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(15)。

根据题目的要求，先创建自定义数据类型 usertype1，语句如下：

```
USE chapter3
CREATE TYPE usertype1
FROM varchar(15);
```

不要忘记了，将该数据类型也创建到数据库 chapter3 中。执行上面的语句，在 chapter3 中就创建了一个名为 usertype1 的数据类型。

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

```
USE chapter3                                    --打开 chapter3 数据库
CREATE TABLE userinfo2
(
   id int,
   name   varchar(20),
   password varchar(10),
   email    varchar(20),
   QQ     usertype1,
   tel      usertype1
);
```

执行上面的语句，就可以在数据库 chapter3 中创建表 userinfo2 了。执行效果如图 3.3 所示。

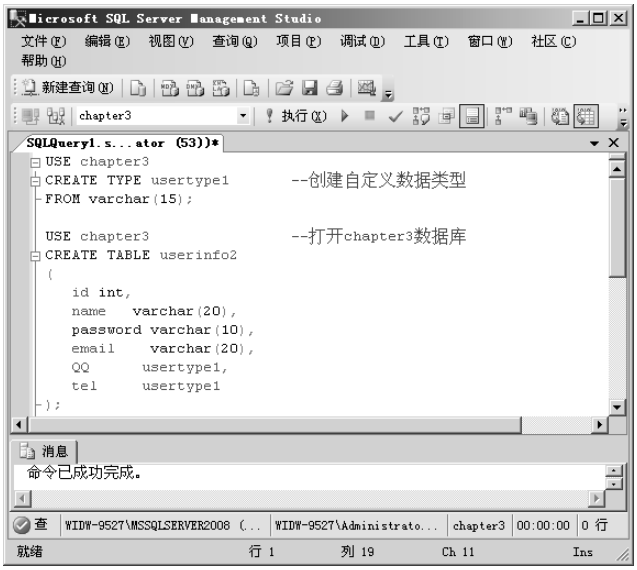

图 3.3　创建表 userinfo2

3.2.5　在其他文件组上创建数据表

前面的示例 2～示例 4 所创建的数据表全部都存放在了 chapter3 数据库中的主文件组

中。在第 2 章学习数据库的操作时就提到过，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到主文件组中，但是也可以指定文件存放到其他文件组中。不仅是数据文件，数据表也是可以指定其存放的文件组的。具体的语句如下：

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

这里，filegroup_name 就是文件组的名字。

【示例 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 数据库中的 chapterfilegroup 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 chapterfilegroup。具体的语句如下：

```
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 chapterfilegroup;
```

执行上面的语句，就可以在 chapter3 数据库中的 chapterfilegroup 文件组里创建 userinfo3 数据表。执行效果如图 3.4 所示。

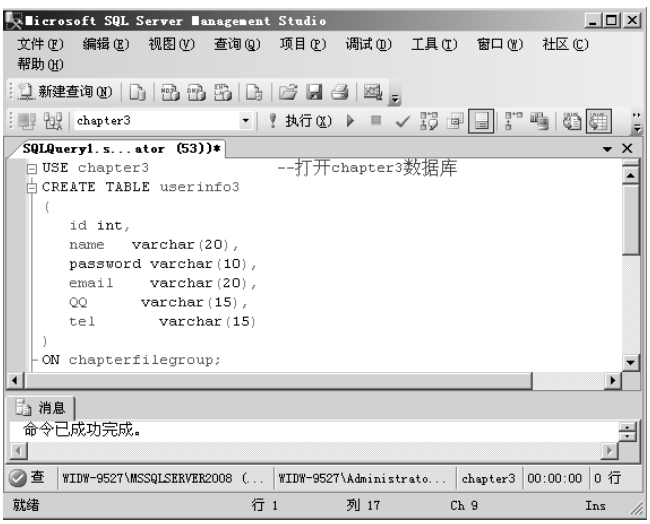

图 3.4　创建表 userinfo3

3.2.6　见识一下临时表

所谓临时表，就不是数据库中永久存在的表。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。临时表就好像是超市的购物车，当购物的时候需要，结账后就不再需要了。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。另外，就是临时表的名字都要以“#”或“##”作为前缀。

【示例 6】 创建一个临时表（#usertemp1），表中的字段信息如表 3-6 所示。

表 3-6　用户信息临时表（#usertemp1）

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 usertemp1 的语句如下：

```
CREATE TABLE  #usertemp1
(
   id int,
   name      varchar(20),
   password  varchar(10)
)
```

执行上面的语句后，就可以在 tempdb 数据库中创建一个名为#usertemp1 的临时表。执行效果如图 3.5 所示。

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

图 3.5　创建临时表#usertemp1

3.2.7　使用企业管理器轻松创建数据表

创建数据表需要记住这么多的语法真是麻烦啊，相信读者也已经厌烦了吧？打起精神吧，现在告诉你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，这个方法就是使用企业管理器。显示企业管理器威力的时间到了，把使用 SQL 语句创建用户信息表的过程在企业管理器中通过示例 7 演练演练。

【示例 7】 在企业管理器中，根据表 3-5 所示的用户信息表的字段信息分别按如下两个要求创建用户信息表 userinfo4 和 userinfo5。完成如下的设置。

（1）设置用户信息表中的编号列为自动增长字段。

（2）使用用户自定义的数据类型 usertype。

根据题目的要求，要在企业管理器中创建数据表，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来完成。下面就先来见识一下表的设计页面。非常简单哦，只需要在 SQL Server 企业管理器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，在弹出的右键菜单中选择“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 所示的界面就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要在该界面中完成。下面就使用该界面分别完成本题的两个小题。

1．自动增长字段

根据题目的要求，要先录入用户信息表的基本内容，然后将表中的编号列为自动增长字段。完成这个要求要分为如下 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，其中数据类型是可以通过下拉列表选择的。录入后效果如图 3.7 所示。

图 3.7　录入用户信息表信息后的效果

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？没错，正如字面的意思就是设置该列是否允许不输入值的，默认情况下，都会将其选中即可以不输入值的。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）列为自动增长

设置某一个列为自动增长列时，前提是该列的数据类型是整数。在企业管理器中设置自动增列是很容易的，只需要在列的属性界面中完成即可。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 3.8　id 的列属性界面

在图 3.8 所示界面中，标识规范选项就是用来设置自动增长的，将其改成“是”，即可将该列设置成自动增长的。如果不加其他的设置，设置的自动增长列就是从 1 开始每次增加 1 的效果。如果要设置自动增长的起始值以及增量，如在本例中将其起始值设置成 1，增量设置成 10，效果如图 3.9 所示。

在图 3.9 中，“标识增量”选项就是要设置的增量，“标识种子”选项就是初始值。读者会发现，在将 id 设置成自增长的列后，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。没错，设置了自动增长值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自动增长列设置后，一定不要忘记保存表的信息哦！保存表的信息方法有很多，这里介绍几个常用的方法吧。最简单的方法是像保存文件一样用 Ctrl+S 组合键，不愿意用键盘还可以使用工具栏上的按钮来保存表信息。除了上面的两种方法外，还可以使用文件菜单下的“保存”选项来完成保存的操作。不论使用哪种方法保存表的信息，都会弹出如图 3.10 所示的界面。

图 3.9　设置自动增长列

图 3.10　保存表信息界面

在图 3.10 所示界面中，输入表的名称，按照本题的要求输入“userinfo4”，单击“保存”按钮，即可完成 userinfo4 的创建操作。还有一点需要注意哦，那就是名字不要与 chaper3 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。

2．用户自定义的数据类型

在设置自定义数据类型之前，不要忘记先创建自定义数据类型。在本题中就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。

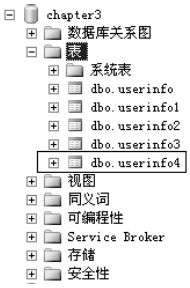

图 3.11　表节点下的 userinfo4

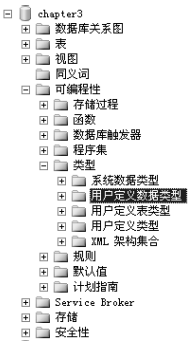

图 3.12 用户定义数据类型的位置

在图 3.12 所示界面中，右击“用户定义数据类型”选项，在弹出的右键菜单中选择“新建用户定义数据类型”选项，弹出如图 3.13 所示界面。

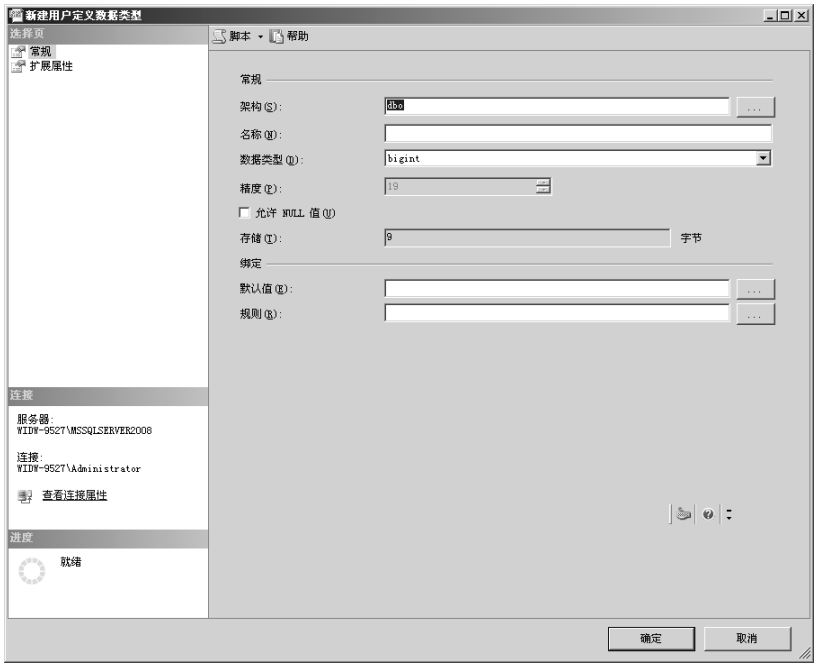

图 3.13　新建用户定义数据类型界面

在图 3.13 所示的界面中，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。

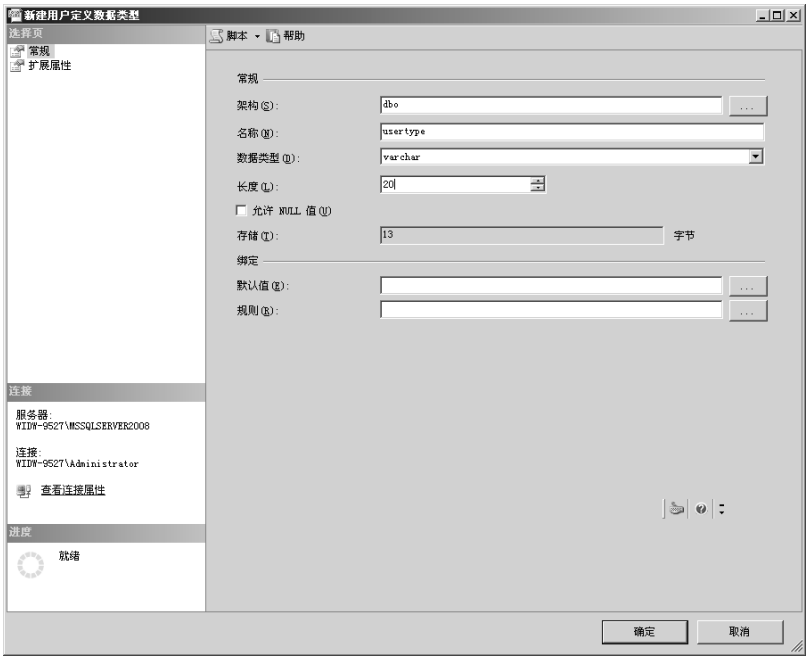

图 3.14　输入自定义值后的效果

在图 3.14 所示界面中，单击“确定”按钮，即可完成自定义数据类型的添加操作。

（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，在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，先将用户信息表的字段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，请读者自己尝试模仿一下吧。这里，为了确保读者能够找到用户自定义数据类型，将表设计器的数据类型显示出来，如图 3.15 所示。

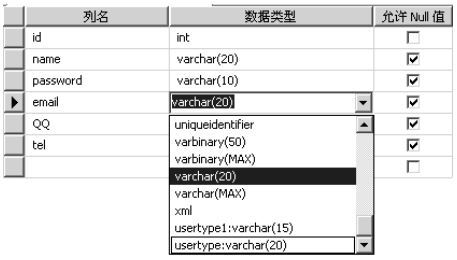

图 3.15　自定义数据类型显示的位置

从图 3.15 可以看出，自定义的数据类型会显示在数据类型列表的最后面。

通过示例 7 的讲解，相信读者已经发现了使用企业管器创建数据表还是很方便的吧！但是，读者也不要放松对 SQL 语句的学习哦！

3.2.8　使用 SP_HELP 看看表的骨架

数据表创建好后，如何查看一下数据表呢？当然，用企业管理器就可以看容易地查看数据了。如果不使用企业管理器如何查看数据表的信息呢？有很多方法可以查看数据表的信息，这里先介绍使用 SP_HELP 存储过程来查看数据表的信息。查看的语句如下：

```
SP_HELP table_name;
```

这里，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哦。

【示例 8】 使用存储过程 SP_HELP 查看用户信息表（userinfo）的表信息。

根据题目的要求，查看的语句如下：

```
SP_HELP userinfo;
```

执行上面的语句，效果如图 3.16 所示。

在图 3.16 中查询结果划分成了 5 个部分，下面分别说明这 5 部分显示的信息都是什么。

- 第 1 部分：显示表创建时的基本信息，包括数据表的名称、类型和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：显示表中自动增长列的信息。由于在表 userinfo 中没有自动增长列，因此在本图中没有自动增长列。
- 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标识符列。在表 userinfo 中也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。

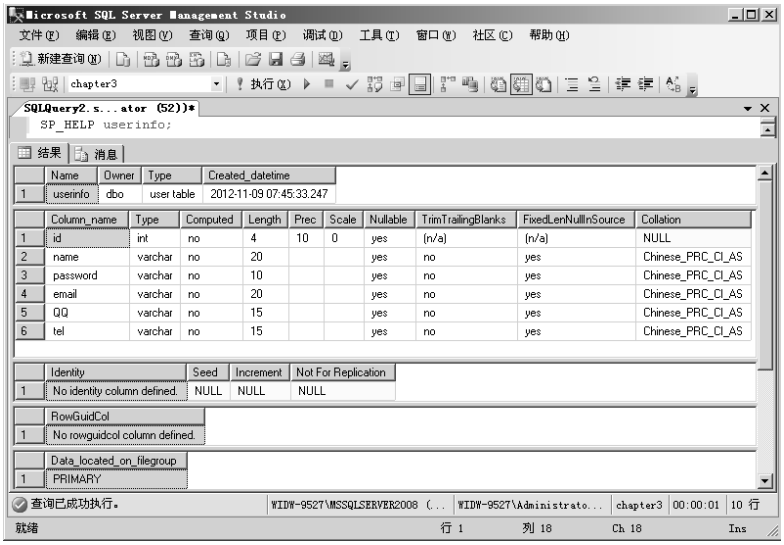

图 3.16　userinfo 表的信息

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。

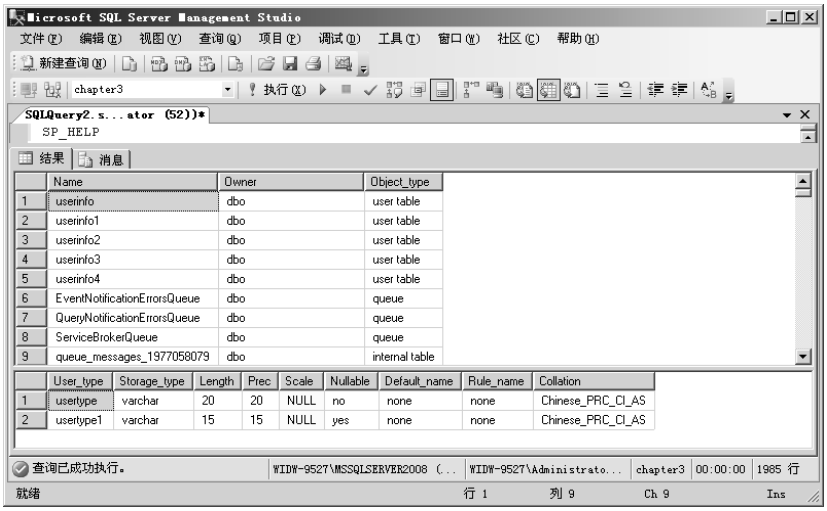

图 3.17　SP_HELP 查询的效果

在图 3.17 的查询结果中有两个部分组成，一部分用于显示数据库 chapter3 中所有的数据对象信息，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 查看表的信息

看了存储过程 SP_HELP 查看的结果，读者觉得有点混乱吧。如果你只想知道数据表的创建信息，现在告诉你一个相对简单点的方法，那就是使用系统表 sys.objects 来查看。

虽然查看的结果显示清晰，但是使用的 SQL 语句就要复杂一些了。下面就使用示例 9 来演示如何使用 sys.objects 来查看表的信息。

【示例 9】 使用系统表 sys.objects 查看 userinfo 表的信息。

根据题目的要求，查看 userinfo 表的信息语句如下：

```
select * from sys.objects where name='userinfo' ;
```

执行上面的语句，就可以将 userinfo 表的创建信息显示出来。效果如图 3.18 所示。

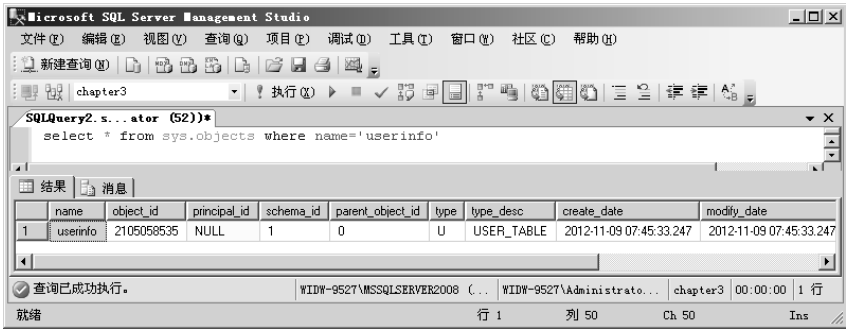

图 3.18　使用 sys.objects 查看 userinfo 表信息

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，相信读者已经看出，在 sys.objects 中的 name 列就是要查找的数据对象名称，也就是说，要查询哪个数据对象就将 name 后的对象名改成要查找的对象即可。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对象。

3.2.10　使用 Information_schema.columns 查看表的信息

在前面的两个小节中分别使用了存储过程和系统表来查看表的信息，除了这两种方法外，还可以使用系统视图 Information_schema.columns 来查看表的信息。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。具体的查看方法使用示例 10 来告诉读者。

【示例 10】 使用 Information_schema.columns 查看 userinfo 表的信息。

根据题目要求，查看 userinfo 表的信息语句如下：

```
select * from Information_schema.columns where table_name='userinfo';
```

执行上面的语句，效果如图 3.19 所示。

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 和 Information_schema.columns 这 3 种查询表信息的方式，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。知道了它们的用途，读者可以根据自己的需要自行选择了吧！

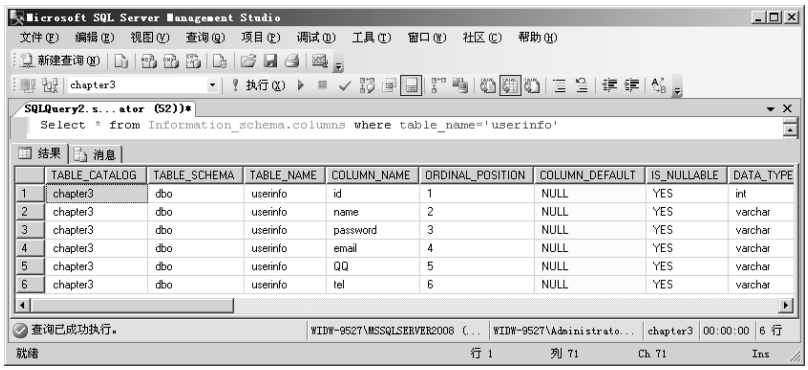

图 3.19　使用 Information_schema.columns 查看 userinfo 表

3.3　修改数据表

任何事务都不可避免地会遇到修改的问题，数据表也不例外。创建好的数据表都可以修改它的哪些内容呢？告诉你吧，其实几乎所有的内容都是可以修改的，包括修改字段的数据类型、添加或减少表中的字段、修改表中字段名字以及给表改名等。另外，还有一个好工具可以帮助我们修改数据表，那就是企业管理器。在本节中，将为读者详细讲述如何修改数据表。

3.3.1　改一改表中的数据类型

表中的数据类型，可以说是最常修改的一项内容了。之所以说它是最常修改的，其实多半是由于数据表的设计人员在设计表时设计的长度不够或者是创建表时写错了。修改数据类型是一件很容易的事情哦，请看下面的语法吧。

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

其中：

- 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，可以先查看一下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字段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。

【示例 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，只是将数据类型的长度更改了一下，具体的语句如下：

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

执行上面的语句后，就可以将用户信息表（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。

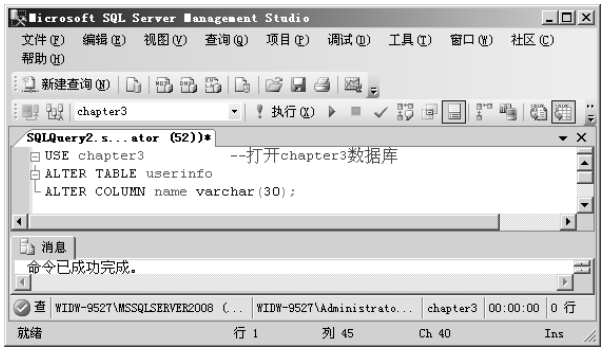

图 3.20　更改 name 列的长度

3.3.2　更改表中字段的数目

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字段，这时就可以用到修改表中字段数目的语句了。修改表中字段数目的语句比较简单，但是也得读者认真记忆，以免混淆。

（1）向表中添加字段

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

其中：

- 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一下表中现有列的信息。
- datatype：数据类型。

（2）删除表中的字段信息

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

其中：

- table_name：表名。
- column_name：列名。删除后的字段可不能恢复了，在删除前可要考虑清楚了。

【示例 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。具体的语句如下：

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

执行上面的语句后，就在用户信息表 userinfo 中就增加了备注列（remark）。执行效果

如图 3.21 所示。

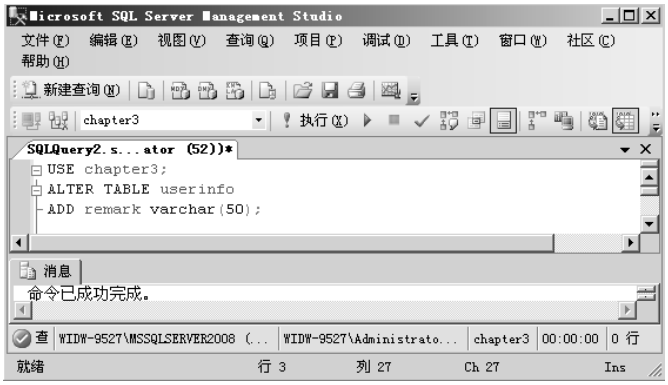

图 3.21　添加 remark 列

【示例 13】 删除在示例 12 中为用户信息表（userinfo）添加的备注列（remark）。

根据题目的要求，删除字段的语句如下：

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

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

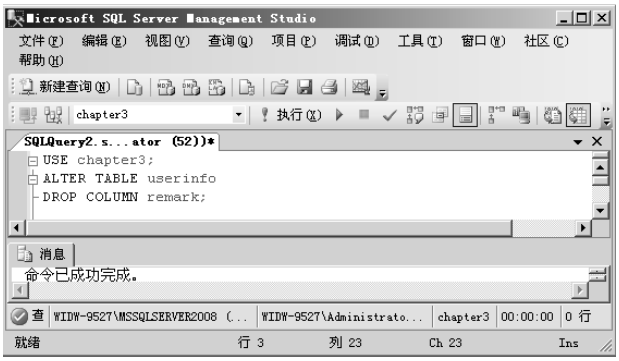

图 3.22　删除 remark 列

3.3.3　给表中的字段改名

要想给表中的字段改名，用 ALTER 语句可就不行喽。那用什么语句来给表中的字段改名呢？如果读者已经学习过了第 2 章，在第 2 章中修改数据库的名字用什么来着？没错，就是用存储过程 sp_rename 来完成的。这里，也可以通过 sp_rename 来修改字段的名字。具体的语法如下：

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

其中：

- tablename.columnname：原来表中的字段名。记住，表中的字段名要加上单引号。