第3章 表的创建和操作

在创建数据库之后，下一步就需要建立数据库表。表是数据库中最基本的数据对象，用于存放数据库中的数据。对表中数据的操作包括添加、修改、删除、查询等。

3.1 表结构和数据类型

3.1.1 表和表结构

每个数据库都包含若干个表。表是 SQL Server 中最主要的数据库对象，它是用来存储数据的一种逻辑结构。表由行和列组成，因此也称为二维表。表是在日常工作和生活中经常使用的一种表示数据及其关系的形式，如表 3.1 所示就是一个“学生”表。

表 3.1 “学生”表

学　号	姓　名	性　别	出生时间	专　业	总学分	备　注
191301	王林	男	1990-2-10	计算机	50	
191302	程明	男	1991-2-1	计算机	50	
191303	王燕	女	1989-10-6	计算机	50	
191304	韦严平	男	1990-8-26	计算机	50	
191306	李方方	男	1990-11-20	计算机	50	
191307	李明	男	1990-5-1	计算机	54	提前修完“数据结构”课程
191308	林一帆	男	1989-8-5	计算机	52	班长
……						

每个表都有一个名字，以标识该表。如表 3.1 的名字是“学生”，它共有 7 列，每列都有一个名字，即列名（一般就用标题作为列名），它描述了学生某一方面的属性。每个表由若干行组成，表的第一行为各列标题，其余各行都是数据。

下面简单介绍与表有关的几个概念。

（1）表结构。组成表的各列名称及数据类型，统称为表结构。

（2）记录。每个表包含若干行数据，它们是表的“值”，表中的一行称为一个记录。因此，表是记录的有限集合。

（3）字段。每个记录由若干个数据项构成，将构成记录的每个数据项称为字段。如表 3.1 的表结构为（学号，姓名，性别，出生时间，专业，总学分，备注）7 个字段，由 5 个记录组成。

（4）空值。空值（NULL）通常表示未知、不可用或将在以后添加的数据。若一个列允许为空值，则向表中输入记录值时可不对该列给出具体值；而一个列若不允许为空值，则在输入时必须给出具体值。

（5）关键字。若表中记录的某一字段或字段组合能唯一标识记录，则称该字段或字段组合为候选关键字（Candidate key）。若一个表有多个候选关键字，则选定其中一个为主关键字（Primary key），也称为主键。当一个表仅有唯一的一个候选关键字时，该候选关键字就是主关键字。

例如，在“学生”表中，两个及其以上记录的“姓名”“性别”“出生时间”“专业”“总学分”“备注”这 6 个字段的值有可能相同，但是“学号”字段的值对表中所有记录来说一定不同，即通过“学号”字段可以将表中的不同记录区分开来。因此，“学号”字段是唯一的候选关键字，“学号”就是主关键字。

“学生成绩”表记录的候选关键字是（学号，课程号）字段组合，它也是唯一的候选关键字。

注意：

表的关键字不允许为空值。空值不能与数值数据 0 或字符类型的空字符混为一谈。任意两个空值都不相等。

3.1.2 数据类型

设计数据库表结构，除了表属性外，主要就是设计列属性。在表中创建列时，必须为其指定数据类型，列的数据类型决定了数据的取值、范围和存储格式。

列的数据类型可以是 SQL Server 提供的系统数据类型，也可以是用户定义的数据类型。SQL Server 提供的系统数据类型如表 3.2 所示。

表 3.2 系统数据类型

数据类型	符号标识
整数型	int, smallint, tiny, bigint
精确数值型	decimal, numeric
浮点型	real,float
货币型	money, smallmoney
位型	bit
字符型	char, varchar, varchar(MAX)
Unicode 字符型	nchar, nvarchar, nvarchar(MAX)
文本型	text, ntext
二进制型	binary, varbinary, varbinary(MAX)
图像型	Image
日期时间型	date, datetime, smalldatetime, datetime2, datetimeoffset, time
时间戳型	timestamp
平面和地理空间数据类型	geometry, geography
其他数据类型	sql_variant, uniqueidentifier, xml, hierarchyid

在讨论数据类型时，使用了精度、小数位数是针对数值型数据的，它们的含义如下。

- 精度：指数值数据中所存储的十进制数据的总位数。
- 小数位数：指数值数据中小数点右边可以有的数字位数的最大值。例如，数值数据 3890.587 的精度是 7，小数位数是 3。

下面分类说明系统数据类型。

1．整数型：int，smallint，tiny，bigint

整数型包括 4 种类型，从标识符的含义就可以看出，它们表示的数范围逐渐缩小。

类型	名称	数范围	精度	存储字节
int	整数	-2^{31}～$2^{31}-1$	10	4
smallint	短整数	-2^{15}～$2^{15}-1$	5	2
tinyint	微短整数	0～255	3	1
bigint	大整数	-2^{63}～$2^{63}-1$	19	8

2．精确数值型：decimal，numeric

精确数值型数据由整数和小数部分构成，其所有的数字都是有效位，能够以完整的精度存储十进制数。decimal 和 numeric 在功能上完全等价。

格式：numeric | decimal(*p*[,*s*])，其中 *p* 为精度，*s* 为小数位数，*s*<*p*, 默认值为 0。

存储$-10^{38}+1$～$10^{38}-1$ 的固定精度和小数位的数字数据。

3．浮点型：real，float

浮点型不能精确表示数据的精度，用于处理取值范围非常大且对精确度要求不太高的数值量。

类型	数范围	定义长度(*n*)	精度	字节
real	–3.40E+38～3.40E+38	1～24	7	4
float	–1.79E+308～1.79E+308	25～53	15	8

4．货币型：money，smallmoney

用十进制数表示货币值。

类型	数范围	小数位数	精度	字节
money	-2^{63}～$2^{63}-1$	4	19	8
smallmoney	-2^{31}～$2^{31}-1$	4	10	4

说明：

（1）当向表中插入 money 或 smallmoney 类型的值时，必须在数据前面加上货币表示符号（$），并且数据中间不能有逗号(,)；若货币值为负数，则需要在符号$的后面加上负号(-)。例如，$15 000.32，$680，$-20 000.9088 都是正确的货币数据表示形式。

（2）money 的数范围与 bigint 相同，不同的只是 money 有 4 位小数。smallmoney 与 int 的关系就如同 money 与 bigint 的关系。

5．位型：bit

它只存储 0 和 1。当为 bit 类型数据赋 0 时，其值为 0，而赋非 0 时，其值为 1。字符串值 TRUE 转换为 1，FALSE 转换为 0。

6．字符型、Unicode 字符型和文本型：char/nchar，varchar/nvarchar，text/ntext varchar（MAX）/nvarchar（MAX）

字符型数据用于存储字符串，字符串中可包括字母、数字和其他特殊符号（如#、@、&等）。在输入字符串时，需将串中的符号用单引号或双引号括起来，如'abc'、"C 语言"。

（1）char：定长字符数据类型，其中 *n* 定义字符型数据的长度，*n* 在 1～8000 之间。默认 *n*=1。若实际存储的串长度不足 *n* 时，则在字符串的尾部添加空格以达到长度 *n*。若输入的字符个数超出了 *n*，则超出的部分被截断。

（2）varchar：变长字符数据类型，*n*（1～8000）表示的是字符串可达到的最大长度。实际长度为输入字符串的实际字符个数，而不一定是 *n*。当列中的字符数据值长度接近一致时，如姓名，此时可

使用 char；当列中的数据值长度显著不同时，使用 varchar 较为恰当，可以节省存储空间。

（3）text：可以表示最大长度为 $2^{31}-1$ 个字符，其数据的存储长度为实际字符个数。

nchar、nvarchar、ntext 使用 UNICODE UCS-2 字符集，该字符集 1 个字符用 2 个字节表示，*n*=1～4000，占用 2*n* 字节空间。

（4）varchar(MAX)、nvarchar(MAX)：最多可存放 $2^{31}-1$ 个字节的数据，可以用来替换 text、ntext 数据类型。

7．二进制型和图像型：binary，varbinary，varbinary(MAX)，image

二进制数据类型表示的是位数据流，包括 binary（固定长度）和 varbinary（可变长度）两种。

（1）binary：固定长度的 *n* 个字节二进制数据。*n* 的取值范围为 1～8000，默认为 1。binary 数据的存储长度为 *n*+4 个字节。若输入的数据长度小于 *n*，则不足部分用 0 填充；若输入的数据长度大于 *n*，则多余部分被截断。

（2）varbinary：*n* 个字节变长二进制数据。

（3）image（图像数据型）：用于存储图片、照片等。实际存储的是可变长度二进制数据，介于 0 与 $2^{31}-1$ 字节之间。该类型是为了向下兼容而保留的数据类型。

（4）varbinary(MAX)：最多可存放 $2^{31}-1$ 个字节的数据，推荐用户使用 varbinary(MAX)数据类型来替代 image 类型。

8．日期时间型：date，datetime，smalldatetime，datetime2，datetimeoffset，time

日期时间类型数据用于存储日期和时间信息，用户以字符串形式输入日期时间类型数据，系统也以字符串形式输出日期时间类型数据。

数据类型	日期范围	精确度	说明
date	1.1.1～9999.12.31		日期
datetime	1753.1.1～9999.12.31	3.33ms	日期和时间分别给出
smalldatetime	1900.1.1～2079.6.6	分	日期和时间分别给出
datetime2	1.1.1～9999.12.31	hh:mm:ss[.nnnnnnn]	datetime(*n*)表示 *n*（=1～7）位微秒
datetimeoffset	YYYY-MM-DD	hh:mm:ss[.nnnnnnn] [{+|–}hh:mm]	带时区偏移量
time		hh:mm:ss[.nnnnnnn]	time(*n*)表示 *n*（=1～7）位微秒

（1）日期部分的表示形式常用的格式如下。

年 月 日	2001 Jan 20、2001 January 20
年 日 月	2001 20 Jan
月 日[,]年	Jan 20 2001、Jan 20,2001、Jan 20,01
月 年 日	Jan 2001 20
日 月[,]年	20 Jan 2001、20 Jan,2001
日 年 月	20 2001 Jan
年（4 位数）	2001 表示 2001 年 1 月 1 日
年月日	20010120、010120
月/日/年	01/20/01、1/20/01、01/20/2001、1/20/2001
月-日-年	01-20-01、1-20-01、01-20-2001、1-20-2001
月.日.年	01.20.01、1.20.01、01.20.2001、1.20.2001

说明：年可用 4 位或 2 位表示，月和日可用 1 位或 2 位表示。

（2）时间部分常用的表示格式如下。

时:分	10:20、08:05
时:分:秒	20:15:18、20:15:18.2
时:分:秒:毫秒	20:15:18:200
时:分 AM|PM	10:10AM、10:10PM

9．时间戳型：timestamp

该类型反映系统对该记录修改的相对（相对于其他记录）顺序，它实际上是二进制格式数据，其长度为 8 字节。每当对该表加入新行或修改已有行时，都由系统自动将一个计数器值加到该列，即将原来的时间戳值加上一个增量。一个表只能有一个 timestamp 列。

10．平面和地理空间数据类型：geometry，geography

（1）geometry（平面空间数据类型）：它作为.NET 公共语言运行时（CLR）数据类型实现，表示欧几里得（平面）坐标系中的数据。

（2）geography（地理空间数据类型）：它作为.NET 公共语言运行时（CLR）数据类型实现，表示圆形地球坐标系中的数据。SQL Server 支持 geography 数据类型用于存储 GPS 纬度和经度坐标之类的椭球体（圆形地球）数据。

SQL Server 支持包括开放地理空间信息联盟（OGC）标准和对该标准的一组 Microsoft 扩展所定义的方法。

11．其他数据类型：sql_variant，uniqueidentifier，xml，hierarchyid

（1）sql_variant：一种存储 SQL Server 支持的各种数据类型（除 text、ntext、image、timestamp 和 sql_variant 外）值的数据类型。sql_variant 的最大长度可达 8016 字节。

（2）uniqueidentifier：唯一标识符类型。系统将为这种类型的数据产生唯一标识值，它是一个 16 字节长的二进制数据。

（3）xml：用来在数据库中保存 xml 文档和片段的一种类型，但是此种类型的文件大小不能超过 2 GB。

（4）hierarchyid：可表示层次结构中的位置。

3.1.3　表结构设计

创建表的实质就是定义表结构，设置表和列的属性。在创建表之前，先要确定表的名字、表的属性，同时确定表所包含的列名、列的数据类型、长度、是否可为空值、约束条件、默认值设置、规则及所需索引、哪些列是主键、哪些列是外键等，这些属性构成表结构。

例如，学生管理系统包括学生表（xsb）、课程表（kcb）和成绩表（cjb），为了便于理解，基础部分使用中文名来表示列名（在实际开发中，使用英文字母来表示列名编程会更方便）。

学生表包含的属性有学号、姓名、性别、出生时间、专业、总学分、备注。其中，

- “学号”列：只有学号列能唯一标识一个学生，将学号列设为该表主键。学号值有一定的意义，例如，“191301”中“19”表示所属班级，“13”表示学生的年级，“01”表示学生在班级中的序号，所以学号列的可以是 6 位的定长字符型数据，数据类型为 char(6)。
- “姓名”列：姓名一般不超过 4 个中文字符，可以采用 8 位定长字符型数据，数据类型为 char(8)。
- “性别”列：只有“男”“女”两种值，所以可以使用 bit 型数据，值 1 表示“男”，值 0 表示“女”，默认是 1。
- “出生时间”列：该列可能进行日期运算，存放日期时间类型数据，列类型为 date。
- “专业”列：假定专业名最多为 6 个汉字，可定为 12 位定长字符型数据 char(12)。

- “总学分”列：是整数型数据，值在 0～160 之间，列类型为 int，默认是 0。
- “备注”列：需要存放学生的备注信息，备注信息的内容在 0～500 个字之间，所以应该使用 varchar 类型。

最后设计的 xsb（学生表）的表结构如表 3.3 所示。

表 3.3 xsb（学生表）的表结构

列 名	数据类型	长 度	是否可空	默认值	说 明
学号	定长字符型（char）	6	×	无	主键，前 2 位表示班级，中间 2 位为年级号，后 2 位为序号
姓名	定长字符型（char）	8	×	无	
性别	位型（bit）	默认值	√	1	1 表示男；0 表示女
出生时间	日期型（date）	默认值	√	无	
专业	定长字符型（char）	12	√	无	
总学分	整数型（int）	默认值	√	0	
备注	不定长字符型（varchar）	500	√	无	

当然，如果要包含学生的“照片”列，可以使用 image 或 varbinary(MAX)数据类型；要包含学生的“联系方式”列，可以使用 xml 数据类型。

参照 xsb 表结构的设计方法，同样可以设计出其他两个表的结构，如表 3.4 所示为 kcb（课程表）的表结构，如表 3.5 所示为 cjb（成绩表）的表结构。

表 3.4 kcb（课程表）的表结构

列 名	数据类型	长 度	是否可空	默认值	说 明
课程号	定长字符型（char）	3	×	无	主键
课程名	定长字符型（char）	16	×	无	
开课学期	整数型（tinyint）	1	√	1	范围为 1～8
学时	整数型（tinyint）	1	√	0	
学分	整数型（tinyint）	1	×	0	范围为 1～6

表 3.5 cjb（成绩表）的表结构

列 名	数据类型	长 度	是否可空	默认值	说 明
学号	定长字符型（char）	6	×	无	主键
课程号	定长字符型（char）	3	×	无	主键
成绩	整数型（int）	默认值	√	0	范围为 0～100

表结构设计完后就可以在数据库中创建表了，本书所用到的学生管理系统表都在 xscj 数据库中创建。创建和操作数据库中的表既可以通过“SSMS”中的界面方式进行，又可以通过 T-SQL 命令方式进行。

3.2　以命令方式创建表

通过 SSMS 图形界面方式创建表简单方便，但从学习的角度，建议使用 T-SQL 命令创建表。当然，命令方式和界面方式可相互配合，以界面方式创建表可以通过命令方式修改其属性，以命令方式创建表可以通过界面方式查看和操作。

界面方式操作表将在附录 A 中系统介绍，此后不再说明。

3.2.1　创建表


1．创建表命令

创建表命令的主要格式如下。

```
CREATE TABLE　表名
(
    { <列定义>　|　<计算列定义>　|　<列集>}
    [ <表约束> ] [ ,...n ]
 )
 [ ; ]
```

说明：

（1）表名的完整写法是：[数据库名.[架构名]. | 架构名.] 表名，但前面部分一般不写。后面不再说明。

（2）创建表包括定义表和定义组成表的各列。<表约束>用于保证表中数据的完整性。

列的定义可以是下列 3 种。

1）列

<列定义> ::=

```
列名 <数据类型>                                  /*指定列名、列的数据类型*/
[ NULL | NOT NULL ]                              /*指定是否为空*/
[
    [ CONSTRAINT 约束名 ]
    DEFAULT 常量表达式                           /*指定默认值*/
]
| [ IDENTITY [ (初值, 增量) ]                    /*指定列为标识列*/
[ ROWGUIDCOL ]                                   /*指定列为全局标识符列*/
[ <列约束> … ]                                   /*指定列的约束*/
```

<数据类型> ::=

```
类型名 [ ( 精度 [ , 小数位 ] | max ]
```

说明：

（1）NULL | NOT NULL：NULL 表示列可取空值，NOT NULL 表示列不可取空值。

（2）DEFAULT 常量表达式：为所在列指定默认值，默认值“常量表达式”必须是一个常量值、标量函数或 NULL 值。DEFAULT 定义可适用于除定义为 timestamp 或带 identity 属性列以外的任何列。

（3）IDENTITY：指出该列为标识符列，为该列提供一个唯一的、递增的值。“初值”是标识字段的起始值，默认值为 1，“增量”是标识增量，默认值为 1。

（4）ROWGUIDCOL：表示列是行的全局唯一标识符列，ROWGUIDCOL 属性只能指派给 uniqueidentifier 列。该属性并不强制列中所存储值的唯一性，也不会为插入到表中的新行自动生成值。

（5）<列约束>：列的完整性约束。指定该列为主键则使用 PRIMARY KEY 关键字。

2）计算列

计算列中的值是通过其他列计算出来的，该列实际并不存放值。

<计算列定义> ::=

```
列名 AS 计算列表达式
[ PERSISTED [ NOT NULL ] ]
```

注意：

有些函数，如 getdate()，每次调用时都输出不同的结果，不能用于计算表达式的定义。

3）列集

列集用于 XML 列。

【例 3.1】 设已经创建了数据库学生成绩，现在该数据库中需创建学生情况表 xsb，该表的结构（见表 3.3）。

单击“新建查询”按钮，在查询编辑器中输入下列 T-SQL 命令：

```
USE xscj
GO
CREATE TABLE xsb
(
    学号        char(6)      NOT NULL PRIMARY KEY,
    姓名        char(8)      NOT NULL,
    性别        bit          NULL DEFAULT 1,
    出生时间    date         NULL,
    专业        char(12)     NULL DEFAULT   '计算机',
    总学分      int          NULL DEFAULT 0,
    备注        varchar(500)  NULL,
    年龄        AS   2015-year(出生时间)   PERSISTED
)
GO
```

说明：

（1）首先使用 USE xscj 语句将数据库 xscj 指定为当前数据库，然后使用 CREATE TABLE 语句在该数据库中创建表 xsb。如果在“SSMS”中当前的数据库为“xscj”，如图 3.1 所示，则不需要使用 USE xscj 语句。

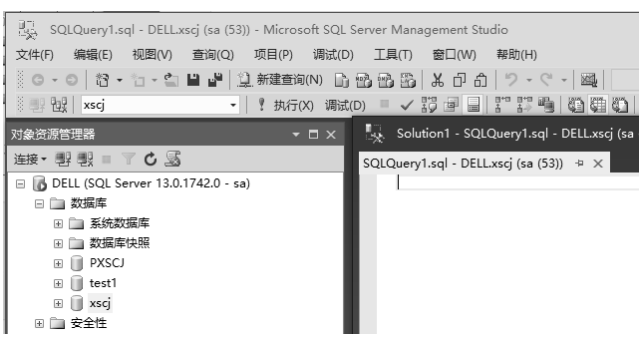

图 3.1　当前的数据库

（2）年龄字段是由出生时间字段计算得到的，是计算类型字段。不能用 getdate()函数代替 2015，因为 getdate()函数在一次操作中会有不同的值。

（3）执行命令后，在 xscj 数据库中就创建了 xsb 表，如图 3.2 所示。

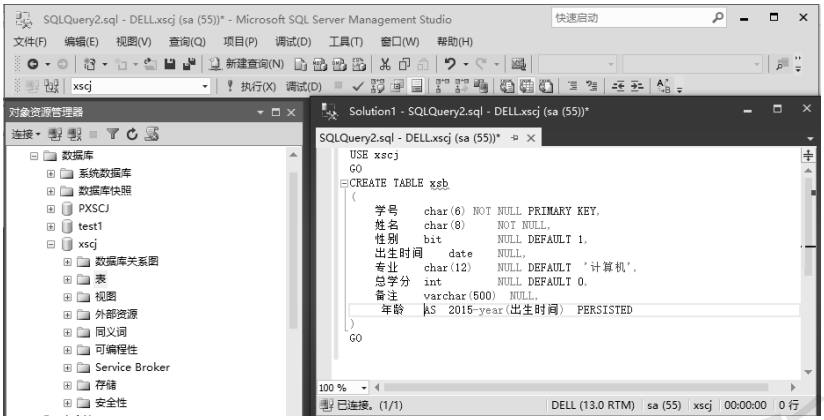

图 3.2　以命令方式创建了 xsb 表

2．创建临时表

在 SQL Server 中创建的表通常称为持久表。在数据库中，持久表一旦创建，则将一直存在，多个用户或者多个应用程序可以同时使用持久表。有时需要临时存放数据，例如，临时存储复杂的 SELECT 语句的结果，此后，可能要重复地使用这个结果，但这个结果又不需要永久保存。这时，就可以使用临时表。用户可以像操作持久表一样操作临时表，只不过临时表的生命周期较短，当断开与该数据库的连接时，服务器会自动删除它们。

在表名称前添加“#”或“##”符号，创建的表就是临时表。添加“#”符号表示创建的是本地临时表，只能由创建者使用；添加“##”符号表示创建的是全局临时表，可以供所有的用户使用。

3．利用已有表创建表

```
SELECT  *  INTO 新表名
    FROM  已有表名  WHERE  条件
```

如果仅仅拷贝老表的结构，则命令“条件”为假。

【例 3.2】 利用 xscj 数据库中的 xsb 表创建表 xsb1 表。

```
SELECT  *  INTO  xsb1  FROM  xsb
```

3.2.2　修改表结构

1．修改表结构命令

修改表结构语法格式如下：

```
ALTER TABLE  表名
{
    ALTER COLUMN 列名{, …}                          /*修改列属性*/
    | ADD                                           /*添加列*/
    {
        <列的定义>
    }   [ , ... ]  <表约束>
    | DROP                                          /*删除列*/
    {
```

```
        [ CONSTRAINT ] 约束名                                    /*删除约束*/
        |COLUMN 列名
    } [ , ... ]
}
```

1）命令主体

ALTER TABLE 命令主体结构说明如下。

（1）表名：要修改的表名。

（2）ALTER COLUMN 子句：修改表中指定列的属性，“列名”给出要修改的列。

若表中该列所存数据的数据类型与将要修改的列类型冲突，则发生错误。例如，原来 char 类型的列要修改成 int 类型，而原列值包含非数字字符，则无法修改。

（3）ADD 子句：向表中增加新列，新列的定义方法与 CREATE TABLE 命令中定义列的方法相同。一次还可以添加多个列，中间用逗号隔开。

（4）DROP 子句：从表中删除列或约束。

> **注意：**
> 在删除一个列以前，必须先删除基于该列的所有索引和约束。

2）ALTER COLUMN 子句

该子句的内容格式为：

```
ALTER COLUMN 列名
{
    类型名 [ ( 精度[ , 小数位] ) ]
        [ COLLATE 排序名 ]
        [ NULL | NOT NULL ]
    ……
}
```

（1）类型名：为被修改列的新数据类型。当要修改成数值类型时，可以使用“(精度[, 小数位])”分别指定数值的精度和小数位数。

（2）[NULL | NOT NULL]：表示将列设置为是否可为空，设置成 NOT NULL 时要注意表中该列是否有空数据。

2．修改表结构命令举例

【例 3.3】 修改 xscj 数据库 xsb1 表结构。

（1）增加列：在 xsb1 表中增加“入学时间”列。

在查询编辑器中输入下列 T-SQL 命令：

```
ALTER TABLE xsb1
    ADD 入学时间 date
```

如果原表中已经存在与添加列同名的列，则语句运行将出错。

（2）修改列：修改表 xsb1 中已有列的属性。将名为“姓名”的列长度由原来的 8 改为 10；将名为“入学时间”列的数据类型由原来的 date 改为 small datetime。

在查询编辑器中输入下列 T-SQL 命令：

```
ALTER TABLE xsb1
    ALTER COLUMN 姓名 char(10)
ALTER TABLE xsb1
    ALTER COLUMN 入学时间 smalldatetime
```

说明：在 ALTER TABLE 语句中，一次只能包含 ALTER COLUMN、ADD、DROP 子句中的一条。而且使用 ALTER COLUMN 子句时一次只能修改一个列的属性，所以这里需要使用两条 ALTER

TABLE 语句。

（3）删除列：删除入学时间和年龄列。

在查询编辑器中输入下列 T-SQL 命令：

```
ALTER TABLE xsb1
    DROP COLUMN  入学时间, 年龄
```

3.2.3　删除表

语法格式如下：

```
DROP TABLE   表名 [ , ... ]
```

其中，“表名”是要被删除的表名称。

3.3　以命令方式操作表数据

对表数据的插入、修改和删除还可以通过 T-SQL 语句来进行，与界面操作表数据相比，通过 T-SQL 语句操作表数据更为灵活，功能更为强大。

3.3.1　插入记录

1．插入记录命令

插入记录使用 INSERT 语句，语法格式如下：

```
INSERT  [ TOP ( 表达式 ) [ PERCENT ] ]
[ INTO ]  表名 | 视图名
[ ( 列表 ) ]
VALUES (   DEFAULT |   NULL | 表达式…)          /*指定列值*/
|  DEFAULT VALUES                               /*强制新行包含为每个列定义的默认值*/
|  SELECT 命令
```

说明：

（1）表名和视图名：被操作的表名称和视图名称。

（2）列表：只给表的部分列插入数据时，需要用“列表”指出这些列。没有在“列表”中指出的列，它们的值确定原则如下。

- 具有 IDENTITY 属性的列，其值由系统根据初值和增量值自动计算得到。
- 具有默认值的列，其值为默认值。
- 没有默认值的列，若允许为空值，则其值为空值；若不允许为空值，则出错。
- 类型为 timestamp 的列，系统自动赋值。
- 如果是计算列，则使用计算值。

（3）VALUES 子句：包含各列需要插入的数据，数据的顺序要与列的顺序相对应。若省略“列表”，则 VALUES 子句给出每一列（除 IDENTITY 属性和 timestamp 类型以外）的值。VALUES 子句中的值可有以下三种情况。

- DEFAULT：指定为该列的默认值。这要求定义表时必须指定该列的默认值。
- NULL：指定该列为空值。
- 表达式：可以是一个常量、变量或一个表达式，其值的数据类型要与列的数据类型一致。例如，列的数据类型为 int，若插入的数据是'aaa'就会出错。当数据为字符型时要用单引号括起来。

（4）DEFAULT VALUES：该关键字说明向当前表中所有列均插入其默认值。此时，要求所有列均定义了默认值。

（5）SELECT 命令：数据由 SELECT 查询结果产生。

2．插入记录命令举例

【例 3.4】　向表 xsb1 中插入记录。

（1）插入一行。

231301，王一平, 1, 1990-02-10，计算机, 50 , NULL

T-SQL 命令如下：

```
INSERT INTO    xsb1
      VALUES('231301', '王一平' , 1, '1990-02-10', '管理工程',50, NULL)
```

插入上例数据也可以使用以下命令：

```
INSERT INTO xsb1 (学号，姓名，出生时间,专业,总学分)
      VALUES('191301', '王林', '1990-02-10', '管理工程' 50)
```

或者：

```
INSERT INTO xsb1
      VALUES('191301', '王林', DEFAULT, '1990-02-10', '管理工程',50, NULL);
```

说明：若插入数据行中含有与原有行中关键字相同的列值，则 INSERT 语句无法插入此行。例如，如果重复执行上述命令，系统显示错误信息。

（2）一次向表中插入 2 条记录。

'201301', '王海', 1, '1996-05-10', '软件工程', 50, NULL

'201302', '李娜', 0, '1996-04-12', '软件工程', 52, NULL

T-SQL 命令如下：

```
INSERT INTO xsb1 VALUES('201301', '王海', 1, '1991-05-10', '软件工程', 50, NULL),
      ('201302', '李娜', 0, '1991-04-12', '软件工程', 52, NULL)
```

（3）从其他表的记录插入其中：从 xscj 数据库表 xsb1 中生成软件工程专业的学生表（xsb2）。

- 用界面方式或者 CREATE TABLE 语句建立表 xsb2，表结构与 xsb1 相同。
- 用 INSERT 语句向 xsb2 表中插入数据。

```
INSERT INTO xsb2
      SELECT    *
            FROM xsb1
            WHERE  专业= '软件工程'
```

- 使用 SELECT 语句查询 xb1 和 xsb2 表记录。

```
SELECT    *
      FROM    xsb1                                    /*显示 xsb1 表记录*/
GO
SELECT    *
      FROM    xsb2                                    /*显示 xsb2 表记录*/
GO
```

执行结果如图 3.3 所示。

说明：在执行 INSERT 语句时，如果插入的数据与约束或规则的要求产生冲突或值的数据类型与列的数据类型不匹配，那么 INSERT 执行失败。

图 3.3　从其他表的记录插入

3.3.2　修改记录

在 T-SQL 中，UPDATE 语句可以用来修改表中的数据行。

语法格式如下：

```
UPDATE [ TOP ( 表达式 ) [ PERCENT ] ]
{   表名 | 视图名      }
SET { 列名=表达式, …   }                                    /*赋予新值*/
[ FROM   <表源> … ]
[ WHERE <查找条件>  | …   ]                                  /*指定条件*/
……
```

其中，

（1）SET 子句：用于指定要修改的列或变量名及其新值。

（2）FROM 子句：指定用表来为更新操作提供数据。

（3）WHERE 子句：WHERE 子句中的<查找条件>指明只对满足该条件的行进行修改，若省略该子句，则对表中的所有行进行修改。

【例 3.5】　将 xsb1 表中学号为“231301”的学生备注值改为“外校互认学分课程”，同时将总学分+3 分。

```
UPDATE xsb1
      SET  备注 ='外校互认学分课程',
           总学分 = 总学分+3
      WHERE  学号='231301'
```

执行完上述语句后，xsb1 表学号为“231301”的学生记录如图 3.4 所示。

图 3.4　备注值改变

说明：对于学生成绩管理数据库，一般来说，学生表（xsb）总学分字段数据增加是由于在成绩表（cjb）中增加了记录（学生完成一门课，成绩合格），这两个表的记录修改命令是一起完成的。否

则数据库数据是不完整的。

3.3.3 删除记录

在 T-SQL 语言中，删除数据可以使用 DELETE 语句或 TRANCATE TABLE 语句来实现，分别用于删除表中符合条件记录和所有记录。

1．删除符合条件记录

语法格式如下：

```
DELETE [ TOP ( 表达式 ) [ PERCENT ] ]
[ FROM  表名 | 视图名 |  <表源> ]
[ WHERE  <查找条件>  |  …  ]                    /*指定条件*/
```

说明：

（1）[TOP (表达式) [PERCENT]]：指定将要删除的任意行数或任意行的百分比。

（2）FROM 子句：说明从何处删除数据。可以从以下三种类型的对象中删除数据。

①表：由“表名”指定要从其中删除数据的表，关键字 WITH 指定目标表所允许的一个或多个表提示，一般情况下不需要使用 WITH 关键字。

②视图：由“视图名”指定要从其中删除数据的视图，注意该视图必须可以更新，并且正确引用了一个基本表。

③表源：将在介绍 SELECT 语句时详细讨论。

（3）WHERE 子句：删除操作指定条件。若省略 WHERE 子句，则 DELETE 语句将删除所有数据。

【例 3.6】 在 xsb1 表中删除软件工程专业学生。

```
DELETE  FROM  xsb1  WHERE 专业 = '软件工程'
```

执行完上述语句后，xsb1 表记录如图 3.5 所示。

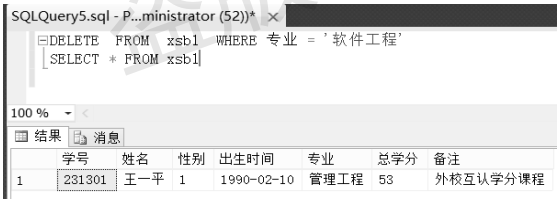

图 3.5 删除软件工程专业学生

2．删除表所有记录

使用 TRUNCATE TABLE 语句将删除指定表中的所有数据，因此也称为清除表数据语句。

语法格式如下：

```
TRUNCATE TABLE  表名
```

说明：

（1）使用 TRUNCATE TABLE 语句删除了指定表中的所有行，但表结构及其列、约束、索引等保持不变，而新行标识所用的计数值重置为该列的初始值。如果要保留标识计数值，则要使用 DELETE 语句。

（2）“TRUNCATE TABLE 表名”与“DELETE 表名”二者均可删除表中的全部行。但 TRUNCATE TABLE 比 DELETE 速度快，且使用的系统和事务日志资源少。DELETE 语句每次删除一行，并会在事务日志中为所删除的每一行记录一项。而 TRUNCATE TABLE 则通过释放存储表数据所用的数据页来删除数据，并且只在事务日志中的记录页释放。

对于由外键（FOREIGN KEY）约束引用的表，不能使用 TRUNCATE TABLE 语句删除数据，而应使用不带 WHERE 子句的 DELETE 语句。另外，TRUNCATE TABLE 语句也不能用于参与索引视图的表。

（3）表记录删除后不能恢复。

（4）如果要在删除表记录的同时删除表结构，则使用“DROP TABLE 表名”命令。

【例 3.7】　删除 xscj 数据库的 xsb1 和 xsb2 表中的所有行。

T-SQL 命令如下：

```
USE xscj
DELETE   xsb1
TRUNCATE TABLE   xsb2
GO
SELECT   * FROM   xsb1                          /*显示没有记录*/
SELECT   * FROM   xsb2                          /*显示没有记录*/
GO
DROP TABLE xsb2
SELECT   * FROM   xsb2                          /*显示错误信息，因为 xsb2 已经没有了*/
GO
```

执行结果如图 3.6 所示。

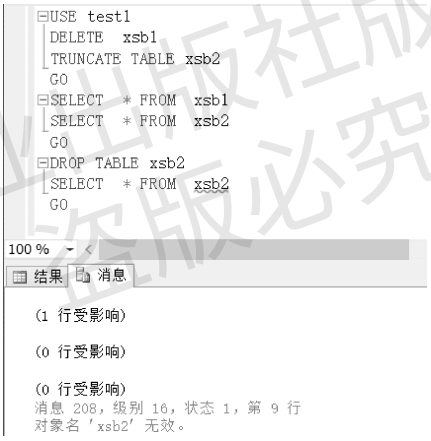

图 3.6　执行结果

另外，INSERT、UPDATE、DELETE 和 MERGE 语句可以包含 OPTION 子句、 OUTPUT 子句等。

OPTION 子句指定应在整个查询中使用所指定的查询提示。

OUTPUT 子句返回基于受这些语句影响的各行表达式。 这些结果可以返回到处理应用程序，以供在确认消息、存档及其他类似的应用程序要求中使用。 也可以将这些结果插入表或表变量。 另外，可以捕获嵌入语句中 OUTPUT 子句的结果，然后将这些结果插入目标表或视图。

3.4　为查询准备数据

为了数据库查询方便，需要通过命令或者界面方式，除了学生成绩数据库（xscj）中学生表（xsb）的表结构已经创建，还需要根据附录 A，创建课程表（kcb）和成绩表（cjb）表结构，然后向学生表（xsb）、课程表（kcb）和成绩表（cjb）录入所有样本记录。