项目4

操作数据

项目3学习了如何通过SQL Server Management Studio和T-SQL语句在创建好的用户数据库中创建表、设计约束以及对表中的列进行添加、修改和删除等维护操作。本项目将开始学习使用T-SQL语句进行数据的输入、修改、删除以及各种查询的操作方法。

【学习目标】

- 了解SQL Server 2008中的逻辑表达式、函数和运算符。
- 掌握向表中插入数据。
- 掌握更新表中的数据。
- 掌握删除表中的数据。
- 理解查询的机制。
- 掌握使用SELECT语句进行基本查询。
- 掌握多表联合查询。
- 掌握子查询。

【知识重点】

- 对记录的添加、修改、删除和查询操作。
- SQL Server 2008中函数的应用。

【知识难点】

- 表中数据的高级查询：多表联合查询和子查询。
- 函数的使用。

4.1 知识准备

与C语言等编程一样，SQL中的表达式也是符号和运算符的组合，并且可以对其求值，得到单个数据值。简单表达式可以是一个常数、变量、列或标量函数，也可以用运算符把两个或多个表达式组合成更复杂的表达式。

4.1.1 条件表达式

SQL中的表达式可以包含以下一个或多个参数。

- 常量：表示指定单个数据值的符号。一个常量由一个或多个字母(a～z，A～Z)、数字字符(0～9)或符号(!、@、#等)组成。字符和datetime数据需要用引号引用，而二进制字符串和数字常量则不需要。
- 列名：表中列的名称，表达式中允许使用列的名称。
- {一元运算符}：仅有一个操作数的运算符，其中“+”表示正数，“-”表示负数，“～”表示补数运算符。
- {二元运算符}：将两个操作数组合执行操作的运算符。二元运算符可以是算数运算符、赋值运算符(=)、位运算符、比较运算符、逻辑运算符和字符串串联(或者联接)运算符(+)。其中比较运算符及其含义如表4-1所示。

表4-1　比较运算符及其含义

运算符	含　义
=	等于
>	大于
<	小于
>=	大于或等于
<=	小于或等于
<>或者！=	不等于
!	非

可以用这些运算符组合成条件表达式。例如，可以编写以下代码：

```
Price>100
Name LIKE '李 %'
Grade<>3
```

在某些T-SQL语句中还可以使用如表4-2所示的通配符运算符。

表 4-2　通配符

通 配 符	解　　释	示　　例
_	一个字符	A LIKE 'C_'
%	任意长度的字符串	B LIKE 'CO_%'
[ ]	括号中所指定范围内的一个字符	C LIKE '9W0[1-2]'
[^]	不在括号中所指定范围内的一个字符	D LIKE '%[A-D][^1-2]'

通常通配符与 LIKE 关键字一起配合使用，完成对表的一些特殊约束。例如，要求表中的电话号码列输入的格式是 11 位手机号，可以编写以下约束：

```
TelPhone  LIKE '1[3,5][5-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'
```

4.1.2　逻辑表达式

T-SQL 支持逻辑运算符 AND、OR 和 NOT。AND 和 OR 是连接条件，NOT 是否定条件。AND 连接条件，当且仅当两个条件都为真时才返回 TRUE；OR 也连接两个条件，但只要其中一个为真就返回 TRUE。逻辑运算符的运算规则如表 4-3 所示。

当一个语句使用了多个逻辑运算符时，应该首先计算 NOT 表达式的值，然后计算 AND 表达式的值，最后再计算 OR 表达式的值。

表 4-3　逻辑运算符的运算规则

A	B	A　AND　B	A　OR　B	NOT A	NOT B
TRUE	TRUE	TRUE	TRUE	FALSE	FALSE
FALSE	TRUE	FALSE	TRUE	TRUE	FALSE
TRUE	FALSE	FALSE	TRUE	FALSE	TRUE
FALSE	FALSE	FALSE	FALSE	TRUE	TRUE

4.2　项目实施

4.2.1　任务 1：使用 SQL Server Management Studio 管理表中数据

子任务 1：查看和添加记录

在对象资源管理器中选择 AdventureWorks 数据库中的 Production. Product 表，右击，在弹出的快捷菜单中选择“编辑前 200 行”选项，在右侧打开的表编辑窗格中即可查看表中数据或者编辑某个字段值，也可以在最下边全部为 NULL 的一行输入数据，单击工具栏中的“执行”按钮或者按 F5 键，修改或添加记录到数据表中，如图 4-1 所示。

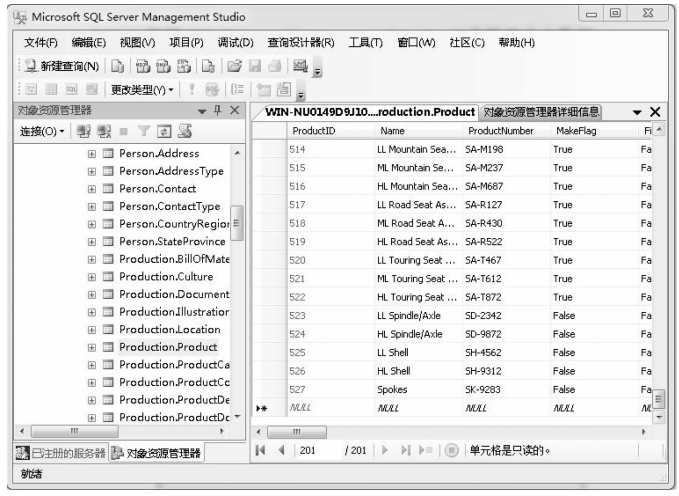

图 4-1　添加记录

子任务 2：删除记录

按照子任务 1 介绍的方法打开表，在如图 4-1 所示的窗口中单击某条记录左侧的灰色选择块，选中整条记录，右击，在弹出的快捷菜单中选择“删除”命令，弹出确认删除对话框，如图 4-2所示。单击“是”按钮即可删除记录。如果该表有关联表，被删除的记录所在表的主键字段在外键表中有对应记录的话，会弹出约束冲突提示对话框，如图 4-3 所示。

图 4-2　确认删除对话框

图 4-3　删除数据时关联约束冲突提示对话框

通过以上方式可以方便地对数据表中的数据进行增加、修改、删除及查找操作，但这些操作不能很好地与应用程序进行数据交互，要想把应用程序中获取的数据添加到数据表中，或者把数据表中的数据提供给应用程序，还需要使用 T-SQL 语句对数据表进行相应操作。

4.2.2　任务 2：使用 T-SQL 语句管理表中数据

子任务 1：使用 T-SQL 语句插入数据

在查询编辑器中，可以使用 T-SQL 语句向表中添加数据行，也可以将现有表中的数据添加到新建的表中。

1)使用 INSERT 语句插入数据行

使用 INSERT 语句一行一行地插入数据是最常见的一种方式,其语法格式如下:

INSERT [INTO] <表名> [列名] VALUES(值列表)

其中的一些选项说明如下:

- [INTO]是可选的,可以省略;
- 表名是必需的,表的列名是可选的,如果省略将依次插入。

提示:多个列名和多个值列表之间用逗号分隔。

在表中插入数据之前,需要先了解下列重要事项:

(1)哪些列必须有值;

(2)哪些列有数据完整性约束;

(3)哪些列要由数据库通过函数来管理;

(4)哪些列有默认值或允许空值;

(5)目标列的数据类型是什么。

例如,向学生信息表 Students 中添加一行数据:

```
INSERT INTO Students (sname,ssex,sbirthday,semail,saddress)
VALUES ('刘婧','女','89-01-01','liuqing@126.com','河南许昌')
```

SQL 语句的执行一般在查询编辑器中进行,以上语句的执行结果如图 4-4 所示。

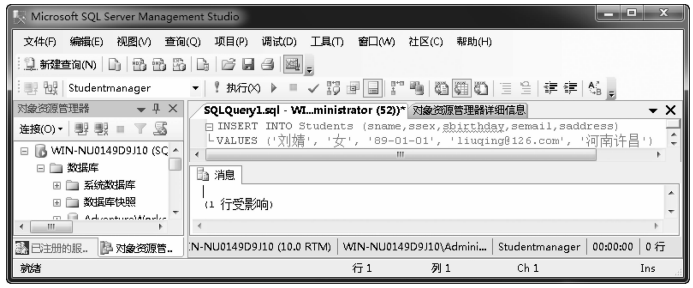

图 4-4 在查询编辑器中执行语句

欲检查 SQL 语句是否正确执行,可以在 SQL Server Management Studio 中查看数据项是否正确添加。

在插入数据时,应注意以下事项。

(1)每次插入一行数据,不可能只插入半行或者几列数据,如果违反字段的非空约束,那么插入操作会失败。

(2)数据值的数目必须与列数相同,而且每个数据值的数据类型、精度和小数位数必须与相应列对应。

(3)INSERT 语句不能为标识列指定值,因为标识列的数字是自动增长的。

(4)对字符类型的列,当插入数据时,最好用单引号将其引起来,因为字符中包含数字时较容易出错。

(5)尽管可以不指定列名,但最好明确指定插入的列和对应的值。

(6)如果在设计表时就指定了某列不允许为空,则必须插入数据,否则将报告出错。

(7)插入的数据项要求符合检查约束的要求。例如,在前面设置的 semail 字段必须包含一个字符“@”,如果插入语句修改为:

```
INSERT INTO Students (sname,ssex,sBirthday,sEmail,sAddress)
VALUES ('刘红刚',1,'89/01/01','LHG6688126.com','河南开封')
```

执行该语句后，将提示与表的约束冲突，插入失败，如图 4-5 所示。

图 4-5　执行插入语句时与检查约束发生冲突

具有默认值的列，可以使用 DEFAULT(缺省)关键字来代替插入的数值。

还有一个问题就是如果指定了列名，如何为具有默认值的列插入数据？例如，以上学生信息表中的地址信息就是具有默认值的。

这时可以使用 DEFAULT(缺省)关键字来代替插入的数值，插入语句如下：

```
INSERT INTO Students (sname,ssex,sBirthday,sEmail,sAddress)
VALUES ('赵伶俐',0,'90/01/01','ZHAOLINGLI@126.com',DEFAULT)
```

2)一次插入多行数据

一次插入多行数据的方法有如下 3 种。

(1)通过 INSERT…SELECT…语句将现有表中的数据添加到新表。例如，创建一张新表来存储本班的通讯录信息，并从学生表中提取相关的数据，SQL 语句如下：

```
INSERT  INTO  xinxi('姓名','地址','电子邮件')
SELECT  sName,sAddress,sEmail
FROM  Students
```

上面的 SQL 语句用来把学生信息表中已经存在的姓名、地址和 E-mail 信息插入到新的 xinxi 表中，避免了输入大量重复的数据项。

需要注意的是，被插入数据的 xinxi 表必须先创建好结构；查询得到的数据的个数、顺序、数据类型等必须与插入的数据项保持一致。

(2)通过 SELECT…INTO…语句将现有表中的数据添加到新表。与上面的 INSERT…SELECT…语句类似，SELECT…INTO…语句也是从一个表中选择一些数据插入到新表中，所不同的是这个新表是执行查询语句时创建的，不能够预先存在。例如：

```
SELECT Students.sName,Students.sAddress,Students.sEmail
INTO  newTable1
FROM  Students
```

上面的语句是将创建 newTable 表，把 Students 表中的 sName、sAddress、sEmail 作为 newTable 表的新列，并把查询到的数据全部添加到新表中。

在向一个新表中添加数据时，又会涉及一个新的问题，即如何插入新的标识列？标识列的数据是不能指定的，因此可以创建一个新的标识列，语法如下：

```
SELECT  IDENTITY(data_type,seed,increment) AS 列名
INTO 新表
FROM 原始表
```

其中，IDENTITY(data_type,seed,increment)是一个函数，它的作用是在带有 INTO 子句的 SELECT 语句中将标识列插入到新表中。

- data_type:标识列的数据类型。标识列的有效数据类型既可以是任何整数数据类型(bit 数据类型除外),也可以是 decimal 数据类型。
- seed:分配给表中第一行的整数值。为每一个后续行分配下一个标识值,该值等于上一个 seed 值加上 increment 值。如果既没有指定 seed,也没有指定 increment,那么它们都默认为 1。
- increment:要加到表中后续行的 seed 值上的整数值。

上面的语句修改如下:

```
SELECT Students. sName, Students. sAddress, Students. sEmail, IDENTITY( int, 1, 1)
As StudentID
    INTO  newTable2
    FROM Students
```

(3)通过 UNION 关键字合并数据进行插入。UNION 语句用于将两个不同的数据或查询结果组合成一个新的结果集。

当然,不同的数据或查询结果,也要求数据个数、顺序和数据类型都一致,因此,当向表中多次重复插入数据时,可以使用 SELECT…UNION 语句来简化操作。语法格式如下:

```
INSERT INTO <表名>(列名)
SELECT <列名> UNION
SELECT <列名> UNION
```

例如:

```
INSERT  STUDENTS (sName,sSex)
SELECT '测试女生 1',0 UNION
SELECT '测试女生 2',0 UNION
SELECT '测试女生 3',0 UNION
SELECT '测试女生 4',0 UNION
SELECT '测试男生 1',1 UNION
SELECT '测试男生 2',1 UNION
SELECT '测试男生 3',1 UNION
SELECT '测试男生 4',1 UNION
SELECT '测试男生 5',1
```

这样的效果其实与使用 INSERT…SELECT…的效果是一样的,只不过多行数据是手写的,然后用 UNION 合并组成多行,再把这多行数据一起插入。

子任务 2:使用 T-SQL 更新数据

数据更新是数据库经常要进行的操作,使用 T-SQL 语句就可以进行数据更新。

使用 UPDATE 语句更新表中某行数据的语法格式是:

```
UPDATE <表名> SET <列名 1=更新值 1>[,<列名 2=更新值 2>]
[WHERE <更新条件>]
```

其中的一些选项说明如下:

- SET 后面可以紧随多个数据列的更新值,数据列表达式之间用逗号间隔开;
- WHERE 字句是可选的,用来限制条件,如果不限制,整个表的所有数据行都将被更新。

需要注意的是，使用 UPDATE 语句可以更新一行，也可以更新多行，甚至可能一行也不更新。

例如，将学生信息表中所有学生的性别都改为 0(女性)的语句如下：

```
UPDATE Students SET sSEX=0
```

按原来表的定义，学生的地址如果不输入则为默认值“河南郑州”，如果需要修改此默认值，则需要按照条件进行更新，更新语句如下：

```
UPDATE Students
SET sAddress='河北石家庄'
WHERE sAddress='河南郑州'
```

前面已经提到，在 T-SQL 语句表达式中，可以使用列名和数值。如果学生在考试时题目难度过大，需要整体增加分数以提高及格率，就需要在成绩表中更新成绩。例如，所有低于 95 分的都在原来的基础上增加 5 分，更新的 SQL 语句如下：

```
UPDATE Score
SET Score=Score+5
WHERE Score<95
```

提示：在更新数据时，一般都有条件限制，不要忘了写 WHERE 条件语句，否则将更新表中所有数据行的数据，这就可能导致有效数据丢失。

子任务 3：使用 T-SQL 删除数据

如果经常需要删除表中的数据，使用 T-SQL 语句操作相对比较简单。

1)使用 DELETE 语句删除数据

(1)删除若干符合条件的记录。使用 T-SQL 语句删除数据表中的数据，语法格式为：

```
DELETE FROM <表名> [WHERE <删除条件>]
```

例如，在学生信息表中删除姓名为“张三”的记录，删除语句如下：

```
DELETE FROM Students
WHERE sName='张三'
```

另外一种情况是，如果要删除的行的主键被其他表引用，如分数表中的 sID 引用了学生表的 sID 字段，那么删除被引用的行时，SQL Server 2008 将报告与约束冲突的错误信息。

例如：

```
DELETE FROM Students
WHERE sID='0010012'
```

提示：DELETE FROM… 的作用是删除整条记录，不会只删除单个字段，所以在 DELETE后不能出现字段名。

例如：

```
DELETE sAddress FROM Students
```

将报告错误信息。

(2)删除全部记录。如果删除记录语句中缺少 WHERE 子句，则其功能是删除全部记录。例如：

```
DELETE  FROM  students
```

此语句将删除 students 表中的所有记录，但表的结构、列和约束索引等还将存在。

2)使用 TRUNCATE TABLE 语句删除数据

TRUNCATE TABLE 语句用来删除表中的所有行,功能上类似于没有 WHERE 子句的 DELETE 语句。例如,删除学生信息表中所有记录行的语句如下:

```
TRUNCATE  TABLE  Students
```

但 TRUNCATE TABLE 语句比 DELETE 语句执行速度快,而且使用的系统资源和事务日志资源更少。

提示: TRUNCATE TABLE 语句将删除表中的所有行,但表的结构、列和约束索引等不会被改动。TRUNCATE TABLE 语句不能用于有外键约束的表,这种情况下需要使用 DELETE 语句。

4.2.3 任务 3:使用 T-SQL 查询数据

数据表在接受查询请求时,可以简单地理解为,它将逐行选取、判断是否符合查询的条件。如果符合就提取出来,然后把所有被选择的行组织在一起,形成另外一个“类似于表的结构”。这便是查询的结果,通常称为记录集(recordset)。

由于记录集的结构实际上和表的结构是相同的,都是由多行组成的,因此,在记录集上依然可以再进行查询。

子任务 1:使用 SELECT 语句进行查询

使用 SELECT 语句进行查询,最简单的语法格式可以表示为:

```
SELECT     <列名>
FROM       <表名>
[WHERE     <查询条件表达式>]
[ORDER BY <排序的列名> [ASC 或 DESC]]
```

其中一些选项的说明如下。

- WHERE 及条件是可选的,如果不限制,则查询返回所有行的数据项。
- ORDER BY 是用来进行排序的,数据表中的记录是无序的,并不按照一定的次序存储。例如,要按照学生的考试成绩排序以看到高分的情况,则需要按照分数列的值进行排序。

提示:在查询中还可以用到很多其他关键字,实现其他特殊的要求。有关 SELECT 语句的详细语法请参考 SQL Server 2008 联机丛书。

查询语句一般都在查询编辑器中进行调试和运行,下面分别举例说明最基本的查询的不同情况。

1)查询全部行和列

可以使用通配符“*”来表示所有列,把数据表中的所有行和列都列举出来,语句如下:

```
SELECT  *  FROM Students
```

2)查询部分行和列,即条件查询和筛选列

查询部分列,需要列举部分列名,而查询部分行需要使用 WHERE 子句进行条件限制。例如:

```
SELECT sID,sName,sAddress FROM Students
WHERE sAddress='河北石家庄'
```

以上的查询将只查询地址为“河北石家庄”的学生，并且只显示编号、姓名和地址列。同理，以下的查询结果为：只要地址不是“河北石家庄”的学生都被查询出来。

```
SELECT sID,sName,sAddress FROM Students
WHERE sAddress<>'河北石家庄'
```

3）在查询中使用列别名

使用 AS 子句可以改变结果集列的名称，也可以为组合或者计算出来的列指定名称，让标题列的信息更易懂。例如，把 sID 列名查询后显示为“学生编号”。在 T-SQL 语句中重新命名列也可以使用 AS 子句。例如：

```
SELECT sID AS 学生编号,sName AS 学生姓名,sAddress AS 学生地址
FROM Students
WHERE sAddress <> '河南新乡'
```

在查询编辑器中执行的结果如图 4-6 所示。

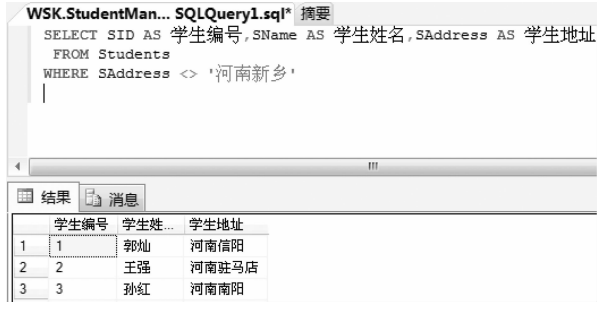

图 4-6　在查询中重新命名列

还有一种情况是计算、合并得到新列的列名。例如，在查询 NorthWind 数据库的 Employees 表中的数据时，需要把 FirstName 和 LastName 字段合并成一个叫“姓名”的字段，可以执行以下查询：

```
SELECT FirstName+ '.' +LastName AS '姓名'
FROM Employees
```

重新命名列也可以用“'列名' =”的方法来命名。例如：

```
SELECT '姓名' =FirstName+ '.'+LastName  FROM Employees
```

4）查询空行

在 SQL 语句中可以采用 IS NULL 或者 IS NOT NULL 来判断是否为空行。因此要查询学生信息表中没有填写 SEmail 信息的学生，可以使用以下查询语句：

```
SELECT sName FROM Students WHERE sEmail IS NULL
```

5）使用常量列

有的时候，一些常量的缺省信息需要添加到查询输出列中，以方便统计和计算。例如，要查询计算机专业的学生信息时，查询输出的语句是：

```
SELECT 姓名=sName,地址=sAddress,'计算机专业' AS 专业
FROM Students
```

查询结果多出一列“专业”，该列的所有数据都是“计算机专业”。

6）查询返回限制的行数

一些查询需要返回所限制的行数。例如，在测试时，如果数据库中有成千上万条记录，

而只要检查前面 10 条记录数据是否有效，就没有必要查询输出全部数据，为提高查询速度，就要用到返回限制的行数查询。

在 T-SQL 语句中，限制行数使用 TOP 关键字来约束。例如，要查询返回 5 位女生的姓名和地址信息的语句如下：

```
SELECT TOP 5 SName,sAddress
FROM Students WHERE sSex=0
```

如果需要从表中按一定的百分比提取记录，则还需要用到 PERCENT 关键字来限制。例如，提取一半的女生信息的相关语句如下：

```
SELECT TOP 50 PERCENT sName,sAddress
FROM Students WHERE sSex=0
```

子任务 2：查询排序

如果需要按照一定的顺序排列查询语句选中的行，需要使用 ORDER BY 子句，排序可以指定是升序(ASC)或者降序(DESC)。如果不指定 ASC 或者 DESC，则记录集默认按 ASC 升序排列。

上面介绍过的 SQL 语句，都可以在其后面再加上 ORDER BY 子句来进行排序。

【实例 4-1】 查询学生成绩时，把所有人的成绩都降低 10%后加 5 分，再按照及格成绩由低到高排列。

```
SELECT sID AS 学员编号,(Score*0.9+5) AS 综合成绩
FROM Score
WHERE (Score*0.9+5)>60
ORDER BY Score
```

在查询编辑器中的执行结果如图 4-7 所示。

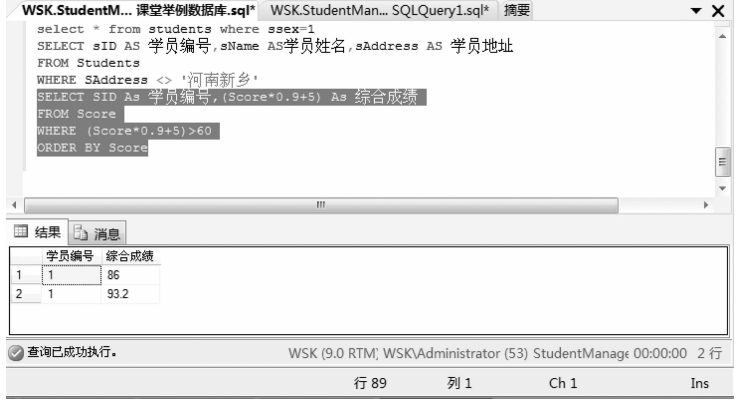

图 4-7 查询结果按照升序排列

【实例 4-2】 查询 Pubs 数据库中的作者表和雇员表，合并查到的所有姓名信息，然后按照姓名降序排列。

```
SELECT Au_Lname + '.' +Au_fName AS EMP
FROM Authors Union
SELECT fName + '.' + LName AS EMP
FROM Employee
```

```
ORDER BY EMP DESC
```

在查询编辑器中的执行结果如图 4-8 所示。

图 4-8　查询结果按照降序排列

查询语句选中的行还可以按照多个字段进行排序。例如，在学生成绩表的基础上再按照学员 ID 进行排序，语句如下：

```
SELECT SID AS 学员编号,Score AS 成绩
FROM Score
WHERE Score>60
ORDER BY Score,SID
```

在查询编辑器中的执行结果如图 4-9 所示。

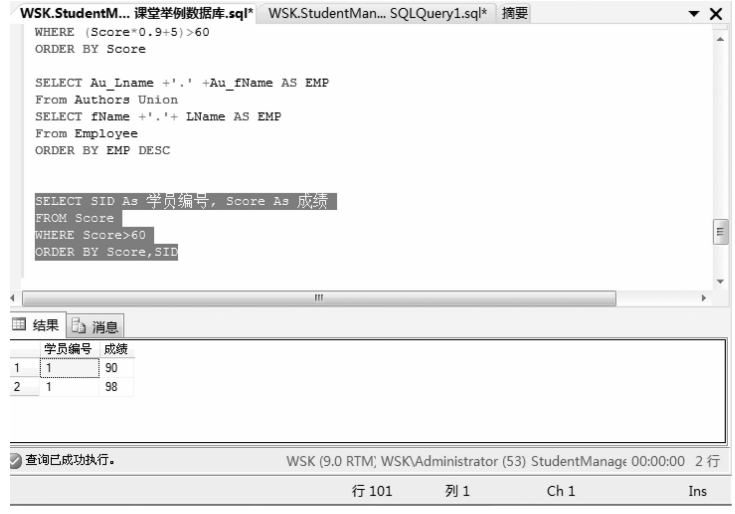

图 4-9　查询结果按照多个列来排序

子任务 3：在查询中使用 T-SQL 函数

与 C 语言类似，T-SQL 也提供了一些内部函数，不同类别的函数可以和 T-SQL 的 SELECT、UPDATE 和 INSERT 语句一起使用。

一般把函数分为字符串函数、日期函数、数学函数和系统函数 4 类。

1）字符串函数

字符串函数用于控制返回给用户的字符串。例如，返回指定字符串长度的语句 SELECT len('软件学院软件技术专业')的结果是 10。

2）日期函数

日期函数用于操作日期值，不能直接对日期运用数学函数。例如，执行“当前日期＋1”语句，T-SQL 无法理解要增加的是一日、一月还是一年。

日期函数用于提取日期值中的日、月及年，以便分别操作它们。其格式如下所示。

- SELECT GETDATE( )　返回当前日期和时间。
- SELECT DATEADD(dd,－1,GETDATE( ))　返回系统日期前一天的日期。
- SELECT DATEADD(dd,1,GETDATE( ))　返回系统日期后一天的日期。

3）数学函数

数学函数用于对数值进行代数运算，表 4-4 所示为 T-SQL 常用的数学函数。

表 4-4　T-SQL 常用的数学函数

函 数 名	描　　述	实　　例
ABS	返回数值表达式的绝对值	SELECT　ABS(－43) 返回：43
CEILING	返回大于或等于所给数字表达式的最小整数	SELECT　CEILING(43.5) 返回：44
FLOOR	返回小于或等于指定表达式的最大整数	SELECT　FLOOR(43.5) 返回：43
POWER	返回数值表达式的幂值	SELECT　POWER(5,2) 返回：25
ROUND	返回数值表达式并四舍五入为指定长度或精度	SELECT　ROUND(43.543,1) 返回：43.5
SIGN	返回给定表达式的正(＋1)、负(－1)号或零(0)	SELECT　SIGN(－43) 返回：－1
SQRT	返回给定表达式的平方根	SELECT　SQRT(9) 返回：3

4）系统函数

系统函数用来获取有关 T-SQL 中对象和设置的系统信息，表 4-5 所示为部分常用的系统函数。

表 4-5　部分常用的系统函数

函数名	描　述	实　例
CONVERT	用来转变数据类型	SELECT CONVERT(VARCHAR(5), 12345)返回字符串 12345
CURRENT_USER	返回当前用户的名字	SELECT CURRENT_USER 返回登录的用户名
DATALENGTH	返回任何表达式所占用的字节数	SELECT DATALENGTH('中国 A 盟')返回 7
HOST_NAME	返回当前用户所登录的计算机名字	SELECT HOST_NAME( )返回所登录的计算机的名字
SYSTEM_USER	返回当前所登录的用户名	SELECT SYSTEM_USER 返回当前所登录的用户名
USER_NAME	返回给定标识号的用户数据库用户名	SELECT USER_NAME(1)返回指定 ID 的用户数据库的用户名

子任务 4：模糊查询

1)使用 LIKE 进行模糊查询

在前面已经介绍过如何使用 LIKE 子句来编写约束，LIKE 运算符用于匹配字符串或者字符串的一部分(称为子串)，仅与 char 和 varchar 数据类型联合使用。

在数据更新、删除或者查询时，可以使用 LIKE 关键字来进行匹配查找。例如：

```
SELECT * FROM Students WHERE SName LIKE '张%'
```

查询不是 8 月份发行的 A 卡或者 C 卡：

```
SELECT * FROM Card WHERE 编号 LIKE '00[^8]%[A,C]%'
```

2)使用 BETWEEN 在某个范围内查询

使用 BETWEEN 可以查找那些介于两个已知值之间的一组未知值。要实现这种查找，必须知道开始要查找的初始值和终值，最大值和最小值之间用单词 AND 分开。例如：

```
SELECT SID,Score FROM SCore
WHERE Score
BETWEEN 60 AND 80
```

BETWEEN 在查询日期范围时用得比较多。例如，查询 Pubs 数据库中不在 1992 年 8 月 1 日到 1993 年 8 月 1 日之间订购的图书列表，语句如下：

```
SELECT * FROM Sales
WHERE ord_date NOT
BETWEEN '1992-8-1' AND '1993-8-1'
```

提示：使用 NOT 来对限制的条件进行“取反”操作。

3)使用 IN 在列举范围内进行查询

若查询的值是指定的某些值之一，可以使用带列举值的 IN 关键字来进行查询。列举值

放在圆括号里，用逗号分开，例如：

```
SELECT SName AS 学生姓名,SAddress As 地址 FROM Students WHERE SAddress IN ('北京','广州','上海')  ORDER BY  SAddress
```

4.2.4 任务 4：使用聚合函数

在查询中还会经常遇到要求查取某些列的最大值、最小值以及平均值等信息，有时候还需要查询有多少行数据项，这时查询的“统计数据”是用户比较关心的。T-SQL 提供了以下聚合函数能够基于列进行计算，并返回单个值。

1. SUM 函数

SUM 函数返回表达式中所有数值的总和，但只能用于数字类型的列，不能够汇总字符、日期等其他数值类型。

例如，在 Pubs 数据库中，要得到商务付款的总数，可以执行下述查询语句：

```
SELECT SUM(ytd_sales) as 总额  FROM titles
WHERE (type='business')
```

得到的查询结果如图 4-10 所示。

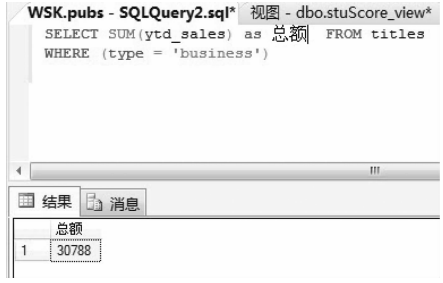

图 4-10　使用 SUM 聚合函数进行汇总

注意：这种查询只返回一个数值，因此，不能够直接与返回多行的列一起使用。

例如，以下语句将报告错误信息，但在一个查询中可以使用多个聚合函数。

```
SELECT SUM(ytd_sales),Price
FROM titles WHERE type='business'
```

2. AVG 函数

AVG 函数返回表达式中所有数值的平均值，只能用于数字型的数据列。例如，学生成绩表中存在的数据项，如图 4-11 所示。

要查询学号为 3 的学生的平均成绩，可以执行以下查询语句：

```
SELECT AVG(Score) AS 平均成绩
FROM Score WHERE sID=3
```

执行的查询结果如图 4-12 所示。

	scoreID	sID	c...	score
1	1	1	1	89
2	2	1	2	98
3	3	1	3	60
4	4	2	1	74
5	5	2	2	90
6	6	2	3	76
7	7	3	1	88
8	8	3	2	80
9	9	3	3	88
10	10	4	1	90
11	11	4	2	80
12	12	4	3	62
13	13	38	1	86
14	14	38	2	66
15	15	38	3	80

图 4-11　学生成绩表中的数据

结果　消息

	平均成绩
1	85.3333333333333

图 4-12　使用 AVG 聚合函数求平均值

3. MAX 函数和 MIN 函数

MAX 函数返回表达式中所有值的最大值，MIN 函数返回表达式中所有值的最小值，它们均可以用于数字型、字符型以及日期时间类型的列。

例如，查询课程代号为 3 的课程平均成绩、最高分、最低分，语句如下：

```
SELECT AVG(Score) AS 平均成绩,MIN(Score) AS 最低分,MAX(Score) AS 最高分
FROM Score WHERE cID=3
```

得到的查询结果如图 4-13 所示。

结果　消息

	平均成绩	最低...	最高分
1	65.2	44	88

图 4-13　使用 MAX 和 MIN 聚合函数求最大值和最小值

4. COUNT 函数

COUNT 函数返回表达式中所有非空值的计数，可以用于数字和字符类型的列。另外，也可以使用通配符“*”作为 COUNT 函数的表达式，使用通配符可以不必指定特定的列而计算所有的行数。

例如，查询课程代号为 3 的课程的及格人数，查询语句如下：

```
SELECT COUNT(*)  AS 及格人数
FROM Score
WHERE Score>60 AND cID=3
```

得到的查询结果如图 4-14 所示。

结果　消息

	及格人数
1	2

图 4-14　使用 COUNT 聚合函数得到行数

4.2.5 任务5:使用分组查询和多表连接查询

子任务1:使用分组查询

1)分组统计

学生成绩表中存放了所有课程的成绩。在这种情况下,可能需要统计不同课程的平均成绩。如果按照前面介绍的方法,则有多少门课程,就需要统计多少次。实际上使用分组统计的方法可以很容易地解决此问题,即首先需要对不同的成绩按照课程来进行分组,分组以后再进行聚合计算,最终得到累计信息。

要统计不同课程的平均分,首先把相同的课程号 cID 都分组,然后对每个组对应的分数值再使用前面介绍的聚合函数去求平均值,如图 4-15 所示。

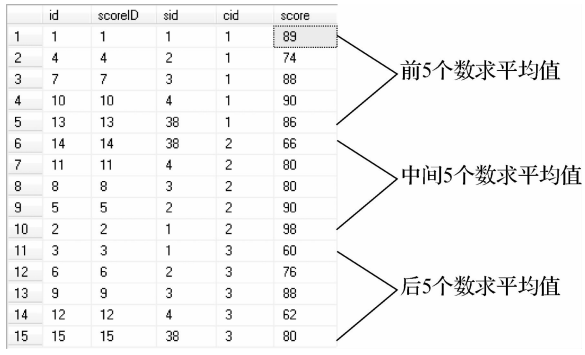

图 4-15 在分组的基础上分别统计

在编写 SQL 语句之前,要明确想要的输出结果。首先是不同的课程,并且课程不能重复存在,因为重复就不符合"分组"的原则了,其次是这些不重复课程的平均分。那么,还能在查询中输出显示这张表中的学生编号的信息吗?显然是不能了,因为学生和课程再也不是一对一的对应关系了,课程已经被分组,分组后的数量减少为 3 组,而学生没有被分组,依然保持有 15 个。

以上这种类型的查询,在 T-SQL 中又称分组查询。分组查询采用 GROUP BY 子句来实现。

完成分组查询的 SQL 语句如下:

```
SELECT cID,AVG(Score) AS 课程平均成绩
FROM Score
GROUP BY cID
```

得到的查询结果如图 4-16 所示。

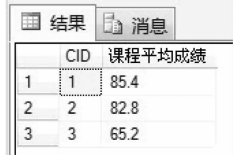

图 4-16 分组查询的输出结果

另外，分组查询可能还要按照多个列进行分组。例如，分组统计中如果写成如下形式：

```
SELECT sID,cID,AVG(Score) AS 课程平均成绩
FROM Score
GROUP BY cID
```

则会提示选择列表中的列'Score. sID' 无效的错误信息，因为该列没有包含在聚合函数或GROUP BY 子句中。

不难理解，在使用 GROUP BY 关键字时，SELECT 语句中指定的项目是有限制的，其中仅允许以下两项：

- 被分组的列；
- 为每个分组返回一个值的表达式。

2)分组统计使用 HAVING 子句进行筛选

继续考虑上面的分组查询，如果查询每门课程中平均分超过 80 分的课程信息，则此时使用 WHERE 子句是不能满足查询要求的，因为 WHERE 子句只能对没有分组统计前的数据行进行筛选(如成绩必须大于 60 分等)。对分组后的条件筛选必须使用 HAVING 子句。

完成上述查询的 SQL 语句如下：

```
SELECT cID AS 课程编号,AVG(Score) AS 平均成绩
FROM Score
GROUP BY cID
HAVING AVG(Score)>80
```

得到的查询结果如图 4-17 所示。

结果　消息

	课程编号	平均成绩
1	1	85.4
2	2	82.8

图 4-17　查询平均成绩超过 80 分的课程

WHERE、HAVING 和 GROUP BY 子句可以用在同一个 SELECT 语句中，使用的顺序如图 4-18 所示。

图 4-18　WHERE 、GROUP BY 和 HAVING 子句的使用顺序

在 SELECT 语句中，WHERE、GROUP BY、HAVING 子句的执行次序如下。

- WHERE 子句从数据源中去掉不符合其搜索条件的数据。
- GROUP BY 子句搜集数据行到各个组中，统计函数为各个组计算统计值。
- HAVING 子句去掉不符合搜索条件的数据行。

例如，按照课程编号分组统计，查询考试不及格学生的科目及不及格的人数，语句如下：

```
SELECT cID AS 课程编号,count( * ) AS 人数
FROM score
WHERE score<60
GROUP BY cID
```

```
HAVING COUNT( * )>1
```

子任务 2：多表连接查询

前面讲述的所有查询都是基于单个数据库表的查询，本子任务将涉及多个表的数据查询。

1)多表连接查询的分类

上面介绍的学生成绩的查询，每次显示的都是学生的编号信息和课程的代号信息，因为该表中只存储了这两种信息。实际上查询学生的考试情况，最好能显示学生的姓名以及每门课程的成绩，而姓名存储在学生的信息表中，课程名称存储在课程表中，像这样需要从多个表中选择或者比较数据项的情况，就需要用到多表连接查询。

多表连接查询实际上是通过各个表之间共同列的关联性来查询数据的，它是关系数据库最主要的查询。

(1)内连接。内连接查询是最典型、最常用的连接查询，它根据表中共同的列来进行匹配，特别是两个表之间存在主—外键关系时通常会用到内连接查询。

内连接查询通常会使用“=”或“!=”等比较运算符来判断两列是否相等，上面所说的根据学生编号信息来判断出学生姓名的连接就是一种内连接。

内连接使用 INNER JOIN 关键字来实现表之间的关联。

(2)外连接。外连接可以是左外连接、右外连接和完全外连接。

- 左外连接：LEFT JOIN 或者 LEFT OUTER JOIN。左外连接的结果集包括 LEFT OUTER 指定的左表的所有行，而不仅仅是连接列所匹配的行。如果左表中的行在右表中没有匹配的行，则在相关联的结果集行中右表的所有选择列将均为空值。
- 右外连接：RIGHT JOIN 或者 RIGHT OUTER JOIN。右外连接是左外连接的反向连接，将返回右表的所有行，如果右表的某行在左表中没有匹配的行，则将为左表返回空值。
- 完全外连接：FULL JOIN 或者 FULL OUTER JOIN。完全外连接返回左表和右表中的所有行。当某行在另外一个表中没有匹配的行时，另外一个表的选择列包含空值，如果表之间有匹配行，则整个结果集将包含基表的数据值。

(3)交叉连接(CROSS JOIN)。交叉连接返回左表中的所有行，左表中所有行再与右表中的所有行一一组合，相当于两个表“相乘”。

2)内连接查询

内连接查询可以通过以下两种方式来实现。

(1)在 WHERE 子句中指定连接条件。例如，查询学生姓名和成绩的语句如下：

```
SELECT Students.sName,Score.CID,Score.Score
FROM   Students,Score
WHERE  Students.sID=Score.sID
```

上面这种形式的查询，相当于 FROM 后面紧跟了两个表名，然后在字段列表中用“表名.列名”的格式来区分列，再在 WHERE 条件子句中加判断条件，即要求学生编号信息相等。当然 FROM 子句中表名也可以使用别名，表中字段也可以使用表的别名来标识。例如，查询学生姓名和成绩的 SQL 语句可以改为：

```
SELECT s.sName,sc.cID,sc.Score
```

```
FROM   Students AS s,Score AS sc
WHERE   s.sID=sc.sID
```

(2)在 FROM 子句中使用 JOIN…ON。上面的查询也可以通过一条 JOIN…ON 子句来实现：

```
SELECT s.sName,sc.cID,sc.Score
FROM   Students AS s INNER JOIN Score AS sc ON   s.sID=sc.sID
```

上述语句是利用 ON 子句指明两个表的连接条件。在上面的内连接查询中，使用关键字 AS 来指定表的“别名”是为了提高查询语句的可读性。如果查询到的列名在用到的两个或者多个表中不重复，则对这一列的引用不必用表名来限定。

在 Northwind 数据库中，以下的 SQL 语句：

```
SELECT ProductID,Suppliers.SupplierID,CompanyName FROM Suppliers   INNER JOIN
Products   ON (Suppliers.SupplierID=Products.SupplierID)
WHERE UnitPrice> $ 10 AND CompanyName LIKE 'F %'
```

将返回某公司提供的一组产品和供应商信息，该公司名以 F 字母开头，并且产品价格在 10 美元以上。

注意：SQL Server 2008 执行连接查询的速度与执行“在 WHERE 子句中指定连接条件”的查询的速度是不一样的。另外，内连接 INNER JOIN 也可以直接写成 JOIN。

内连接查询通常不仅仅连接两个表，有时候还会涉及 3 个或者更多个表。例如，除了学生信息表、学生成绩表之外，还存在课程名称表。上面的查询不仅要显示学生的姓名、成绩，而且还要通过课程编号来显示课程名称表中的课程名称，可以使用以下三表连接查询的 SQL 语句来实现：

```
SELECT S.sName AS 姓名,C.cName AS 课程,SC.Score AS 成绩
FROM Students AS S
JOIN Score AS SC ON (S.sID=SC.sID)
JOIN Course AS C ON (SC.cID=C.cID)
```

该 SQL 语句也可以改为：

```
SELECT S.sName AS 姓名,C.cName AS 课程,SC.Score AS 成绩
FROM Students   S,Score   SC,Course   C
WHERE S.sID=SC.sID AND SC.cID=C.cID
```

在查询编辑器中执行以上语句，查询结果如图 4-19 所示。

3)外连接查询

通过上面的例子可以看出：内连接的查询结果是从两个或两个以上的表的组合中挑选出符合连接条件的数据，如果数据无法满足连接条件则将其丢弃。在内连接中，参与连接的表的地位是平等的。

与内连接相对应的连接方式是外连接，在外连接中参与连接的表有主从之分，是以主表的每行数据去匹配从表的数据列，符合连接条件的数据将直接返回到结果集中；对于那些不符合连接条件的列，将被填上 NULL(空值)后，再返回到结果集中。

(1)左外连接查询。例如，要统计所有学生的考试情况，要求显示所有参加考试的学生每次考试的分数，没有参加考试的学生也要显示出来。这时要以学生信息表为主表，学生成绩表为从表，左外连接查询语句如下：

```
SELECT   S.sName,SC.cID,SC.Score
FROM     Students AS S LEFT JOIN  Score AS SC
ON       SC.SID=S.SID
```

查询的结果可能有一部分学生的成绩没有出现在成绩表上，对应的科目和成绩以NULL(空值)填充，查询的结果如图 4-20 所示。

结果 消息

	姓名	课程	成绩
1	郭灿	JAVA 程序设计	89
2	郭灿	C#	98
3	郭灿	SQL	44
4	王强	JAVA 程序设计	74
5	王强	C#	90
6	王强	SQL	76
7	孙红	JAVA 程序设计	88
8	孙红	C#	80
9	孙红	SQL	88
10	赵春平	JAVA 程序设计	90
11	赵春平	C#	80
12	赵春平	SQL	62
13	张山	JAVA 程序设计	86
14	张山	C#	66
15	张山	SQL	56

图 4-19　三表连接查询

结果 消息

	SName	CID	Score
1	郭灿	1	89
2	郭灿	2	98
3	郭灿	3	44
4	王强	1	74
5	王强	2	90
6	王强	3	76
7	孙红	1	88
8	孙红	2	80
9	孙红	3	88
10	赵春平	1	90
11	赵春平	2	80
12	赵春平	3	62
13	刘星	NULL	NULL
14	夏雪	NULL	NULL
15	夏雨	NULL	NULL
16	许林	NULL	NULL
17	王飒飒	NULL	NULL
18	王春花	NULL	NULL

图 4-20　左外连接查询结果

由以上左外连接查询的结果，不难找出没有参加考试的学生的姓名，查询语句如下：

```
SELECT   S.sName
FROM   Students AS S   LEFT JOIN   Score AS SC
ON   S.sID=SC.sID AND score IS NULL
```

(2)右外连接查询。右外连接查询与左外连接查询的方法类似，只是要包含右表中所有匹配的行。如果右表中有的项在左表中没有对应的项，则以空值来填充。例如，在 Pubs 数据库中，表 Titles 和 Publishers 之间的右外连接将包括所有的出版商，而在 Titles 表中即使没有与 Publishers 表中对应的书名，也将被列出来。

```
SELECT Titles.Title_id,Titles.Title,Publishers.Pub_name
FROM titles
RIGHT OUTER JOIN Publishers
ON Titles.Pub_id=Publishers.Pub_id
```

4.2.6　任务 6:简单子查询

1. 简单子查询

究竟什么是子查询？子查询有什么用？带着这些疑问，不妨先解决一个问题。

学生信息表(Students 表)和学生成绩表(Mark 表)的数据如图 4-21 所示。