第5章 数据查询

视频讲解

建立数据库的目的是查询数据，因此，可以说数据查询是数据库的核心操作，也是 SQL 数据操作的主要内容。查询语句的功能是从数据库中检索满足条件的数据。查询的数据源可以是一张表(或视图)，也可以是多张表(或视图)，查询的结果是由 0 行(没有满足条件的数据)或多行记录组成的一个记录集合(结果表)，并允许选择一列或多列作为输出结果的目标列。SELECT 语句还可以对查询的结果进行排序、汇总等。

5.1 基本查询

5.1.1 SELECT 语句的基本格式

SQL 提供了 SELECT 语句进行数据库的查询，该语句具有灵活的使用方法和丰富的功能。其一般格式为：

```
SELECT  [ALL|DISTINCT] [TOP n[PERCENT]]  <目标列表达式> [,<目标列表达式>,…]
[INTO  <新表名>]
FROM       <表或视图名> [,<表或视图名>]…
[WHERE     <条件表达式>]
[GROUP BY  <列名 1 > [HAVING <条件表达式>]]
[ORDER BY  <列名 2 >[ASC | DESC]]
```

在上述结构中，SELECT 子句用于指定输出字段，FROM 子句用于指定数据的来源，WHERE 子句用于指定数据的选择条件，GROUP BY 子句用于对检索到的结果进行分组，HAVING 子句用于指定组的筛选条件，ORDER BY 子句用于对查询的结果进行排序。在这些子句中，SELECT 和 FROM 两个子句是必需的，其他子句是可选的，各子句的顺序必须和上面格式中的一样。

SELECT 语句既可以完成简单的单表查询，也可以完成复杂的连接查询和嵌套查询。下面以 TMS 数据库为例说明 SELECT 语句的各种用法。

5.1.2 简单查询

简单查询指查询过程中只涉及一个表的查询语句。简单查询是最基本的查询语句。

1. 从表中选择所有的数据内容

查询表的所有数据内容就是将表的所有行和所有列对应的数据全部列出来。将表中的

所有属性列选出来可以有两种方法。一种方法是在 SELECT 关键字后面按照所要显示的列的顺序列出所有列名，如果列的显示顺序与其在基本表中的顺序相同，也可以简单地将<目标列表达式>指定为 * 。

【例 5.1】　查询全体学生的详细信息。

```
SELECT * FROM S
```

也可以写成：

```
SELECT SNO, SNAME, SEX, BIRTHYEAR, SDEPT, SCLASS
FROM S
```

执行结果如图 5.1 所示。

结果　消息

	SNO	SNAME	SEX	BIRTHYEAR	SDEPT	SCLASS
1	20140123	李融	男	1996	软件	软工141
2	20152114	杨宏宇	男	1997	机械	机械151
3	20152221	孙亚彬	女	1997	机械	机械152
4	20180101	袁野	男	2000	软件	软工181
5	20180102	刘明明	男	2000	软件	软工181
6	20180103	王睿	男	2000	软件	软工181
7	20180104	刘平	女	2000	软件	软工181
8	20180201	王珊	女	1999	软件	软工182
9	20180202	张坤	男	2000	软件	软工182
10	20181103	姜鹏飞	男	1998	交通	交通181

图 5.1　例 5.1 执行结果

2. 显示所有的列，有限行

要显示所有的列，但要求结果表只包括有限的行，可在 SELECT 语句中引入 TOP 子句实现。

SELECT 语句中使用 TOP n 关键字输出查询结果集的前 n 行，使用 TOP n PERCENT 输出查询结果集的前面一部分，其中 n 为输出元组总数占结果集总元组数的百分比。

【例 5.2】　要查看 TMS 数据库中，表 S 的数据组成情况，可通过列出该表前 5 行的所有列来了解。

```
SELECT TOP 5 * FROM S
```

执行上述语句可以显示表 S 中的所有列，但对返回结果表的行数进行了限制(TOP 子句)，即只返回前 5 行。查询结果如图 5.2 所示。

TOP 子句在和 ORDER BY 子句一起使用来返回排序后的前 n 个记录时是非常有用的。例如，要得到年龄最大的学生信息，可用下列语句实现：

```
SELECT TOP 1 *
FROM S
ORDER BY BIRTHYEAR ASC
```

查询结果如图 5.3 所示。

结果　消息

	SNO	SNAME	SEX	BIRTHYEAR	SDEPT	SCLASS
1	20140123	李融	男	1996	软件	软工141
2	20152114	杨宏宇	男	1997	机械	机械151
3	20152221	孙亚彬	女	1997	机械	机械152
4	20180101	袁野	男	2000	软件	软工181
5	20180102	刘明明	男	2000	软件	软工181

图 5.2　例 5.2 查询结果(一)

结果　消息

	SNO	SNAME	SEX	BIRTHYEAR	SDEPT	SCLASS
1	20140123	李融	男	1996	软件	软工141

图 5.3　例 5.2 查询结果(二)

注意：上面的方法在有多人出生在相同年份时，仅能查询出自然顺序在第一行的元组，更精确的查询则需要使用到聚合函数，将在后面进行讨论。

【例 5.3】 查询学生表中前面 20%的学生信息。

```
SELECT TOP 20 PERCENT *
FROM S
```

在此查询中输出(前面总人数的 20%)个元组。

关于对结果集进行排序的更多内容将在后文中讨论。

3. 选择表中特定的列

实际上,人们很少选择一个表中的所有信息,大多数情况下只对其中的某些行或列的信息感兴趣。可以通过在 SELECT 子句的<目标列表达式>中指定要查询的属性列名,有选择地列出感兴趣的列。

【例 5.4】 要查看全体学生的学号、姓名和出生年份,可用下列语句实现:

```
SELECT SNO,SNAME,BIRTHYEAR
FROM S
```

查询结果如图 5.4 所示。

注意:SELECT 列名列表中的列名必须用逗号分开,逗号前、后有没有空格都可以。

4. 对结果表进行排序

通过 ORDER BY 子句,可对返回的结果表的行按照在 ORDER BY 子句中指定的一列或多列进行升序(ASC)或降序(DESC)排列,默认为升序排列。通常,ORDER BY 子句中的列名必须在结果表的列名之中。

【例 5.5】 如果按出生年份降序排列例 5.4 的查询结果,可用下列语句实现:

```
SELECT SNO,SNAME,BIRTHYEAR
FROM S
ORDER BY BIRTHYEAR DESC
```

查询结果如图 5.5 所示。

结果 消息

	SNO	SNAME	BIRTHYEAR
1	20140123	李融	1996
2	20152114	杨宏宇	1997
3	20152221	孙亚彬	1997
4	20180101	袁野	2000
5	20180102	刘明明	2000
6	20180103	王睿	2000
7	20180104	刘平	2000
8	20180201	王珊	1999
9	20180202	张坤	2000
10	20181103	姜鹏飞	1998

图 5.4 例 5.4 查询结果

结果 消息

	SNO	SNAME	BIRTHYEAR
1	20180101	袁野	2000
2	20180102	刘明明	2000
3	20180103	王睿	2000
4	20180104	刘平	2000
5	20180202	张坤	2000
6	20180201	王珊	1999
7	20181103	姜鹏飞	1998
8	20152114	杨宏宇	1997
9	20152221	孙亚彬	1997
10	20140123	李融	1996

图 5.5 例 5.5 查询结果

如果在 ORDER BY 子句中有多列,则它们排序的优先顺序是它们在 ORDER BY 子句中从左到右出现的顺序。ORDER BY 子句中的第一列决定了各行排列的主次序,后面的列再对其进行更细致的排列。第一列的值相同的行,其顺序由 ORDER BY 子句中的第二列

决定，以此类推。ASC 或 DESC 可以对每列分别进行设置。

【例 5.6】 如果先按出生年份降序、再按学号升序排列例 5.5 的查询结果，可用下列语句实现：

```
SELECT SNO,SNAME,BIRTHYEAR
FROM S
ORDER BY BIRTHYEAR DESC, SNO ASC
```

查询结果如图 5.6 所示。

ORDER BY 子句中的列可以是列名，也可以是一个整数，该数表示相应的列在 SELECT 子句目标列中的位置，如例 5.6 语句中的 ORDER BY 子句可以有下列几种等价形式：

（1）ORDER BY 3 DESC,1 ASC。

（2）ORDER BY BIRHYEAR,1 ASC。

（3）ORDER BY 3 DESC,SNO ASC。

在一个查询语句中只能有一个 ORDER BY 子句。在 SELECT、FROM、WHERE、ORDER BY 构成的查询语句中，ORDER BY 子句要放在查询语句的最后面。

5. 查询经过计算的值

SELECT 子句的目标列表达式为表达式，即其不仅可以是目标列，还可以是算术表达式、字符串常量、函数、列别名等。

【例 5.7】 查询全体学生的姓名及其年龄。

```
SELECT SNAME, '年龄：',YEAR(GETDATE()) - BIRTHYEAR
FROM S
```

查询结果如图 5.7 所示。

结果　消息

	SNO	SNAME	BIRTHYEAR
1	20180101	袁野	2000
2	20180102	刘明明	2000
3	20180103	王睿	2000
4	20180104	刘平	2000
5	20180202	张坤	2000
6	20180201	王珊	1999
7	20181103	姜鹏飞	1998
8	20152114	杨宏宇	1997
9	20152221	孙亚彬	1997
10	20140123	李融	1996

图 5.6　例 5.6 查询结果

结果　消息

	SNAME	(无列名)	(无列名)
1	李融	年龄：	23
2	杨宏宇	年龄：	22
3	孙亚彬	年龄：	22
4	袁野	年龄：	19
5	刘明明	年龄：	19
6	王睿	年龄：	19
7	刘平	年龄：	19
8	王珊	年龄：	20
9	张坤	年龄：	19
10	姜鹏飞	年龄：	21

图 5.7　例 5.7 查询结果

6. 使用列别名

从例 5.7 查询结果可以看到，如果目标列表达式是经过处理的列，如字符串常量、算数表达式、函数等，查询结果集中该列显示为“无列名”。此时，可以使用列别名代替原列名。

在 SELECT 子句中可以通过以下 4 种方式来定义列别名。

(1) 使用 AS 关键字,如 SELECT SNO AS 学号 FROM S。

(2) 带单引号的列别名,如 SELECT SNO '学号' FROM S。

(3) 带双引号的列别名,如 SELECT SNO "学号" FROM S。

(4) 不带引号的列别名,如 SELECT SNO 学号 FROM S。

如果列别名包含空格、特殊符号等,那么必须将列别名放在双引号或者单引号内。

在下列 4 种情况下通常会使用列别名。

(1) 字段为英文,为方便查看,可以使用中文列别名代替英文字段。

(2) 多表查询时出现相同的列名。如果对多个数据表进行查询,查询结果中可能会出现相同的列名,很容易出现误解,这时候应采用列别名来解决上述问题。

(3) 在查询结果中添加列,在表中出现计算产生新的列时,可以使用列别名。

(4) 统计结果中出现的列,使用聚集函数语句对数据查询时,需要对产生的统计字段使用列别名。

因此,例 5.7 的查询也可以用下面语句完成:

```
SELECT SNAME   NAME,
        '年龄: '  'AGE OF STUDENT',
        YEAR(GETDATE()) - BIRTHYEAR AS AGE
FROM S
```

查询结果如图 5.8 所示。

需要注意的是,字段别名可以使用在 ORDER BY 子句中,但不能使用在 WHERE、GROUP BY 或 HAVING 语句中。

7. 禁止结果表返回重复行

在对一个表进行查询时,得到的结果中可能会有相同的元组,SELECT 语句提供了 DISTINCT 关键字可以保证结果表中行唯一,即 DISTINCT 可以删除结果表中的重复行。该关键字必须紧跟在关键字 SELECT 之后,在一个 SELECT 语句范畴内只可使用一次,且只对返回行有效(对 SELECT 子句指定的列集合中相同的行进行删除)。

【例 5.8】 列出学生表中所有的系名。

```
SELECT SDEPT
FROM S;
```

查询结果如图 5.9 所示。

结果 消息

	NAME	AGE OF STUDENT	AGE
1	李融	年龄:	23
2	杨宏宇	年龄:	22
3	孙亚彬	年龄:	22
4	袁野	年龄:	19
5	刘明明	年龄:	19
6	王睿	年龄:	19
7	刘平	年龄:	19
8	王珊	年龄:	20
9	张坤	年龄:	19
10	姜鹏飞	年龄:	21

图 5.8 例 5.7 查询结果(列别名方法)

结果 消息

	SDEPT
1	软件
2	机械
3	机械
4	软件
5	软件
6	软件
7	软件
8	软件
9	软件
10	交通

图 5.9 例 5.8 查询结果

要去掉重复的元组，需要用以下语句：

```
SELECT DISTINCT SDEPT
FROM S
```

该语句的执行结果如图 5.10 所示。

图 5.10　例 5.8 查询结果（去掉重复元组）

8. 查询满足给定条件的元组

可以通过 WHERE 子句指定返回的结果表中各行需要满足的条件。WHERE 子句指定的条件表达式可能含有一个或多个的谓词或选择条件，它们由各种运算符组合而成，常用的运算符如表 5.1 所示。在 WHERE 子句中使用的列并不一定要求出现在 SELECT 子句的输出列表中。

表 5.1　常用的运算符

查 询 条 件	谓　　词
比较运算符	=，>，<，>=，<=，<>
逻辑运算符	AND，OR，NOT
谓词	IN，BETWEEN AND，LIKE
空值	IS NULL

1）比较条件

比较条件最为简单，同 Java、C 等高级语言一样，用比较运算符比较两个常量或变量的大小。用比较运算符构成的条件形式是：

```
列名　运算符　常量
列名　运算符　列名
常量　运算符　列名
```

如果比较字符型的列，该值就必须用单引号引起来。而且，对字符的比较是大小写敏感的。

【例 5.9】　查询软件学院全体学生的名单。

```
SELECT SNAME
FROM S
WHERE SDEPT = '软件'
```

查询结果如图 5.11 所示。

对于数值型的列，如 INTEGER、SMALLINT 或 DECIMAL，被比较的值不能放在单引号之内。

【例 5.10】　查询在 1999 年以前出生（不包括 1999）的学生姓名及出生年份。

```
SELECT SNAME, BIRTHYEAR
FROM S
WHERE BIRTHYEAR < 1999  (或 NOT BIRTHYEAR >= 1999)
```

查询结果如图 5.12 所示。

结果 消息

	SNAME
1	李融
2	袁野
3	刘明明
4	王睿
5	刘平
6	王珊
7	张坤

图 5.11 例 5.9 查询结果

结果 消息

	SNAME	BIRTHYEAR
1	李融	1996
2	杨宏宇	1997
3	孙亚彬	1997
4	姜鹏飞	1998

图 5.12 例 5.10 查询结果

2) 复合查询条件

可以用 AND、OR、NOT 等逻辑运算符构造更复杂的查询条件。如图 5.13 所示,AND 的运算规律是同真时,结果为真; OR 的运算规律是只要有一个为真,结果即为真。

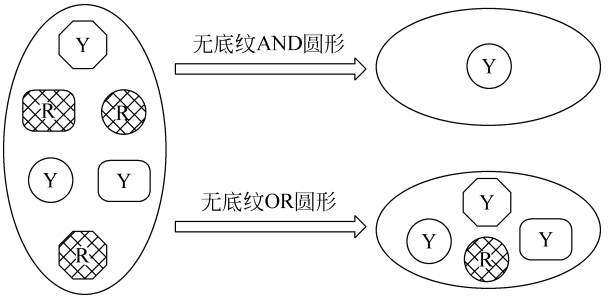


图 5.13 AND 运算和 OR 运算

【例 5.11】 查询出生在 1998—2000 年的学生姓名、所在系和出生年份。

```
SELECT SNAME, SDEPT, BIRTHYEAR
FROM S
WHERE BIRTHYEAR >= 1998 AND BIRTHYEAR <= 2000
```

该语句的查询结果如图 5.14 所示。

【例 5.12】 查询软件学院、机械学院和交通学院的学生姓名和所在学院名。

```
SELECT SNAME, SDEPT
FROM S
WHERE SDEPT = '软件' OR SDEPT = '机械' OR SDEPT = '交通';
```

该语句的查询结果如图 5.15 所示。

有时,需要用小括号()和逻辑运算符来构建复合查询条件。例如,要查询软件学院和机械学院的女生姓名、性别和所在学院,可以用下列语句实现:

```
SELECT SNAME, SEX, SDEPT
FROM S
WHERE (SDEPT = '软件' OR SDEPT = '机械') AND SEX = '女'
```

结果　消息

	SNAME	SDEPT	BIRTHYEAR
1	袁野	软件	2000
2	刘明明	软件	2000
3	王睿	软件	2000
4	刘平	软件	2000
5	王珊	软件	1999
6	张坤	软件	2000
7	姜鹏飞	交通	1998

图 5.14　例 5.11 查询结果

结果　消息

	SNAME	SDEPT
1	李融	软件
2	杨宏宇	机械
3	孙亚彬	机械
4	袁野	软件
5	刘明明	软件
6	王睿	软件
7	刘平	软件
8	王珊	软件
9	张坤	软件
10	姜鹏飞	交通

图 5.15　例 5.12 查询结果

3）谓词

SQL 中的谓词指的是返回值是逻辑值的函数。对于函数而言，返回值可以是数字、字符串或者日期等，但谓词的返回值全部是逻辑值（TRUE/FALSE/UNKNOW），谓词是一种特殊的函数。

（1）IN 谓词。

IN 谓词可用来确定一个值是否属于一个集合。这个值可以是数字、字符、日期或时间。字符、日期或时间必须用单引号引起来。

【例 5.13】 在例 5.12 结果中查询软件学院、机械学院和交通学院的学生姓名和所在学院名，也可以用下列语句实现。

```
SELECT SNAME, SDEPT
FROM S
WHERE SDEPT IN('软件','机械,'交通')
```

若查询既不是软件学院，也不是机械学院和交通学院的学生信息，则 WHERE 条件可以写为：

```
WHERE SDEPT NOT IN('软件','机械,'交通')
```

（2）BETWEEN 谓词。

在 WHERE 子句中用 BETWEEN 谓词可以判断一个值是否在按升序给定的两个值之间（包括和这两个值相等）。较低的值紧跟 BETWEEN 谓词书写，较高或相等的值写在 AND 后。下列 WHERE 子句：

```
WHERE AGE BETWEEN 12 AND 15
```

等价于

```
WHERE AGE >= 12 AND AGE <= 15
```

字符和数字混合的数据也可以用 BETWEEN 谓词。如：

```
WHERE SCLASS BETWEEN '机械 151' AND '软工 181'
```

【例 5.14】 在例 5.10 结果中查询出生在 1998—2000 年的学生姓名、所在系和出生年份,也可以用下列语句实现。

```
SELECT SNAME, SDEPT, BIRTHYEAR
FROM S
WHERE BIRTHYEAR BETWEEN 1998 AND 2000
```

(3) LIKE 谓词。

用 LIKE 谓词可以实现字符匹配。一般语法格式如下:

```
属性列 [NOT] LIKE '<匹配串>' [ESCAPE '<换码字符>']
```

其含义是查询指定的属性列值与<匹配串>相匹配的元组。<匹配串>可以是一个完整的字符串,也可以含有通配符%和_。

① %(百分号):代表任意长度(可以为 0)的字符串。例如,a%b 表示以 a 开头、以 b 结尾的任意长度的字符串。ab、afb、acdeb 等都满足该匹配串。

② _(下画线):代表任意单个字符或汉字。例如,a_b 表示以 a 开头、以 b 结尾的长度为 3 的任意字符串。afb、amb 等都满足该匹配串。

【例 5.15】 查询所有姓王的同学的姓名和所在学院。

```
SELECT SNAME,SDEPT
FROM S
WHERE SNAME LIKE '王%'
```

图 5.16 例 5.15 查询结果

查询结果如图 5.16 所示。

LIKE 谓词可以和逻辑运算符 NOT 组合使用,例如,查询不是 2018 级的学生学号和姓名,可以用下列语句实现:

```
SELECT SNO, SNAME
FROM S
WHERE SNO NOT LIKE '2018%'
```

由于%和_在 LIKE 谓词中做通配符,因此,如果要查询的内容含有%或_,则需要使用 ESCAPE'<换码字符>'短语对通配符进行转义,所用的转义字符必须要用 ESCAPE 子句指定,并且该转义字符在一个 WHERE 子句中可使用多次。

【例 5.16】 查询以 DB_开头,且倒数第三个字符为 i 的课程的详细情况。

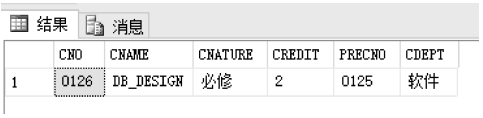
```
SELECT *
FROM C
WHERE CNAME LIKE 'DB\_%i__' ESCAPE '\'
```

在此 SELECT 语句中,反斜杠为转义字符,跟在它后面的下画线不再是通配符,而是下画线本身。

查询结果如图 5.17 所示。

4) 空值(NULL)应用

NULL 只和 IS 或 IS NOT 进行逻辑运算,它既不是数字 0,也不是空串。所以,下列的



结果　消息

	CNO	CNAME	CNATURE	CREDIT	PRECNO	CDEPT
1	0126	DB_DESIGN	必修	2	0125	软件

图 5.17　例 5.16 查询结果

SELECT 语句是错误的：

```
SELECT * FROM SC WHERE GRADE = NULL
```

当执行该语句时，数据库管理系统会提示错误信息。

【例 5.17】　查询成绩为空的学生的学号和相应的课程号。

```
SELECT SNO, CNO
FROM SC
WHERE GRADE IS NULL
```

5.1.3　聚合函数与分组

聚合函数对一组值执行计算，并返回单个值。除 COUNT 外，聚合函数都会忽略 Null 值。聚合函数经常与 SELECT 语句的 GROUP BY 子句一起使用。

1. SQL 中的聚合函数及应用

SQL 中的聚合函数及其应用见表 5.2。

表 5.2　SQL 中的聚合函数及应用

聚 合 函 数	应　　用
SUM（列表达式）	求和
AVG（列表达式）	求平均值
MIN（列表达式）	求最小值
MAX（列表达式）	求最大值
COUNT（*）	返回满足条件的记录数
COUNT（<列名>）	返回满足条件的列中非空值的个数
COUNT（DISTINCT <列名>）	返回满足条件的列中不重复的非空值的个数

SUM，AVG，MIN，MAX，COUNT(*)，COUNT(DISTINCT...) 等聚合函数的参数可以是一列中的一行或多行，而且 SUM 和 AVG 两个函数的参数必须是数值型的。

【例 5.18】　聚合函数 COUNT()几种使用方式的比较。

```
SELECT COUNT(*)                  'COUNT(*)',
       COUNT(GRADE)              'COUNT(GRADE)',
       COUNT(DISTINCT GRADE)     'COUNT(DISTINCT GRADE)'
FROM   SC
```

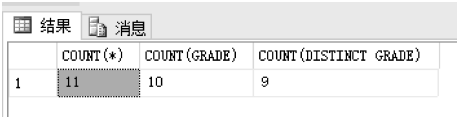
查询结果如图 5.18 所示。

该语句返回 SC 表中的行数、成绩的数目以及不重复的成绩数目。

【例 5.19】 查询女生的人数。

```
SELECT COUNT( * )
FROM S
WHERE SEX = '女'
```

通常,函数 COUNT(*)的自变量是行集合,其结果是集合中的记录个数。其查询结果如图 5.19 所示。



	COUNT(*)	COUNT(GRADE)	COUNT(DISTINCT GRADE)
1	11	10	9

图 5.18 例 5.18 查询结果

	(无列名)
1	3

图 5.19 例 5.19 查询结果

再看下面的例子。

【例 5.20】 查询 0211 号课程的选课人数、平均成绩、最高成绩、最低成绩。

```
SELECT COUNT(SNO) AS 选课人数,
       AVG(GRADE) AS 平均成绩,
       MAX(GRADE) AS 最高成绩,
       MIN(GRADE) AS 最低成绩
FROM  SC
WHERE CNO = '0211'
```

该语句的查询结果如图 5.20 所示。

2. 分组-GROUP BY 子句

1) 简单分组查询

GROUP BY 子句将查询结果表的各行按一列或多列取值相等的原则进行分组。使用 GROUP BY 子句时,一个聚合数对每组生成一个值。对查询结果分组是为了细化聚合函数的作用对象。如果未对查询结果分组,聚合函数将作用于整个查询结果。在实际应用中,经常需要将查询结果进行分组,然后再对每个分组进行统计。

【例 5.21】 查询学生表中男、女生各多少人。

```
SELECT SEX AS 性别,COUNT(SNO) AS 人数
FROM S
GROUP BY SEX
```

查询结果如图 5.21 所示。

	选课人数	平均成绩	最高成绩	最低成绩
1	3	65.5	75	56

图 5.20 例 5.20 查询结果

	性别	人数
1	男	7
2	女	3

图 5.21 例 5.21 查询结果

【例 5.22】　查询每门课程的选课人数，列出课程号及相应的人数。

```
SELECT CNO AS 课程号,COUNT(SNO) AS 选课人数
FROM SC
GROUP BY CNO
```

查询结果如图 5.22 所示。

【例 5.23】　查询每个学院的男、女生各多少人。

```
SELECT SDEPT, SEX, COUNT(SNO)
FROM S
GROUP BY SDEPT, SEX
ORDER BY SDEPT
```

查询结果如图 5.23 所示。

结果　消息

	课程号	选课人数
1	0117	2
2	0121	1
3	0125	2
4	0127	1
5	0211	3
6	0305	2

图 5.22　例 5.22 查询结果

结果　消息

	SDEPT	SEX	(无列名)
1	机械	男	1
2	机械	女	1
3	交通	男	1
4	软件	男	5
5	软件	女	2

图 5.23　例 5.23 查询结果

例 5.23 中，GROUP BY 子句含有两列，即两列的值都相等的为一组。在结果表中的前两行有相同的 SDEPT 列的值('机械')，但是 SEX 列的值不同。GROUP BY 子句的结果保证该子句指定的列组合的唯一。

注意：① 用 AS 子句命名的列不能用在 GROUP BY 子句中。

② GROUP BY 子句不能保证对结果表进行排序。要对结果表排序，需要用 ORDER BY 子句。

2) 带 HAVING 子句的分组查询

如果分组后还要求按一定的条件对这些组进行筛选，最终只输出满足指定条件的组，则可以使用 HAVING 短语指定筛选条件。

【例 5.24】　查询选修了两门以上课程的学生学号。

```
SELECT SNO
FROM SC
GROUP BY SNO
HAVING COUNT(CNO)>2
```

WHERE 子句与 HAVING 子句的根本区别在于处理的对象不同。WHERE 子句作用于基本表或视图，从中选择满足条件的元组。HAVING 子句作用于组，从中选择满足条件的组。

注意：如果一个 SELECT 子句中有聚合函数(如 MIN、MAX、SUM、AVG、COUNT(*)、COUNT(DISTINCT 列名))，并且还有其他列(即没有用到聚合函数的列)存在，那么，SELECT 子句所有没有用到聚合函数的列都必须包含在 GROUP BY 子句中(如例 5.23)。相反，GROUP BY 子句中的列不一定要出现在 SELECT 列表中(如例 5.24)。聚合函数也

可以用在 HAVING 子句中。在 HAVING 子句中可以在表中任何一列上使用聚合函数,该列也可以不出现在 SELECT 子句中。此外,聚合函数不能嵌套使用。

5.1.4 输出结果选项

SELECT 语句中的 INTO 子句用于把查询结果存放到一个新建的表中,新建表名由<新表名>给出,新建表的列由 SELECT 子句中指定的列构成。

【例 5.25】 将所有女生的学号、姓名和所在院系存入表 S_FEMALE 中。

```
SELECT SNO,SNAME,SDEPT
INTO S_FEMALE
FROM S
WHERE SEX = '女'
```

上例查询执行完成后即得到一个名为 S_FEMALE 的新表,表中数据如图 5.24 所示。

结果 消息

	SNO	SNAME	SDEPT
1	20152221	孙亚彬	机械
2	20180104	刘平	软件
3	20180201	王珊	软件

图 5.24 例 5.25 查询结果

5.1.5 SELECT 语句完整的语法

SELECT 语句完整的语法结构如下。

```
SELECT    <目标列表达式> [,<目标列表达式>]
[INTO      <新表名>]
FROM       <表或视图名> [,<表或视图名>]
[WHERE     <条件表达式>]
[GROUP BY  <列名 1 > [HAVING <条件表达式>]]
[ORDER BY  <列名 2 >[ASC | DESC]]
```

SQL 不同于其他编程语言的最明显特征是处理代码的顺序。在多数编程语言中,代码按编码顺序被处理,但是在 SQL 中,第一个被处理的子句是 FROM 子句,尽管 SELECT 语句第一个出现,但是几乎总是最后被处理。

SQL 中的每个步骤都会产生一个虚拟表,该虚拟表被用作下一个步骤的输入。这些虚拟表对调用者(客户端应用程序或者外部查询)不可用。只有最后一步生成的表才会被返回给调用者。如果没有在查询中指定某一子句,则将跳过相应的步骤。

整个语句的执行过程如下。

(1) 读取 FROM 子句中基本表和视图的数据,若为多个基本表或视图,则对它们执行笛卡儿积。

(2) 选取满足 WHERE 子句中给出的条件表达式的元组。

(3) 按 GROUP BY 子句中指定列的值分组,同时提取满足 HAVING 子句中条件表达式的那些组。

(4) 按 SELECT 子句中给出的列名或表达式求值输出。

(5) ORDER BY 子句对输出的目标列进行排序,按 ASC 升序排列,或按 DESC 降序排列。

(6) 创建新表,将查询到的输入插入新表中。

【例 5.26】 下列语句的执行过程如图 5.25 所示。

```
SELECT   SNO,AVG(GRADE) 'AVE OF SCORE'
FROM     SC
```

```
WHERE     SNO<>'20180201'
GROUP BY SNO
HAVING    AVG(GRADE)>70
ORDER BY 2
```

FROM

SNO	CNO	GRADE
20140123	0211	56
20152114	0305	91
20152221	0305	82
20180101	0117	90
20180101	0211	75
20180102	0211	NULL
20180103	0117	60
20180201	0121	80
20180201	0125	56
20180201	0127	77
20180202	0125	42

WHERE

SNO	CNO	GRADE
20140123	0211	56
20152114	0305	91
20152221	0305	82
20180101	0117	90
20180101	0211	75
20180102	0211	NULL
20180103	0117	60
20180202	0125	42

GROUP BY

SNO	CNO	GRADE
20140123	0211	56
20152114	0305	91
20152221	0305	82
20180101	0117	90
20180101	0211	75
20180102	0211	NULL
20180103	0117	60
20180202	0125	42

HAVING

SNO	CNO	GRADE
20152114	0305	91
20152221	0305	82
20180101	0117	90
20180101	0211	75

SELECT

SNO	AVG OF SCORE
20152114	91
20152221	82
20180101	825

ORDER BY

SNO	AVG OF SCORE
20152114	91
20180101	825
20152221	82

图 5.25　SELECT 语句的执行过程

5.2　多表查询

5.2.1　连接查询

查询过程中只涉及一个表的查询称为简单查询。但通常需要借助一个数据库的多个表之间的联系，以从这些表中获得更多的信息。如图 5.26 所示，S 表中的每个学号在 SC 表中都有其相应的一行或多行来表示该学生的选课情况。观察两个表的数据可以发现，在 S 表中的任意一行，可以通过学号(SNO)与 SC 表中的某一行相联系，正是这种联系使人们有可能只使用一个查询语句，从多个表中获取更多的信息。这种一个查询同时涉及两个或两个以上的表称为连接查询。连接查询是关系数据库中最主要的查询，主要包括交叉连接、内连接、外连接等。

S

SNO	SNAME	SEX	BIRTHYEAR	SDEPT	SCLASS
20140123	李融	男	1996	软件	软工141
20152114	杨宏宇	男	1997	机械	机械151
20152221	孙亚彬	女	1997	机械	机械152
⋮	⋮	⋮	⋮	⋮	⋮

SC

SNO	CNO	GRADE
20140123	0211	56
20152114	0305	91
20152221	0305	82
20180101	0117	90
⋮	⋮	⋮

图 5.26　连接查询的实现原理

1. 交叉连接

交叉连接也称为笛卡儿积,它是没有连接条件下的两个表的连接,包含了所连接的两个表中所有元组的全部组合。该连接方式在实际应用中很少使用,语法格式如下:

```
SELECT <目标列表达式> [,...n]
FROM <表 1> CROSS JOIN <表 2>
```

【例 5.27】 查询所有学生可能的选课情况。

```
SELECT SNAME,CNAME
FROM S CROSS JOIN C
```

2. 内连接

内连接指从两个表的笛卡儿积中选出符合连接条件的元组。它使用 INNER JOIN 连接运算符,并使用 ON 关键字指定连接条件。内连接是一种常用的连接方式,如果在 JOIN 关键字前面没有指定连接类型,那么默认的连接类型就是内连接。内连接的语法格式如下:

```
SELECT <目标列表达式> [,...n]
FROM <表 1> INNER JOIN <表 2>
ON <连接条件表达式> [,...n]
```

注意:连接条件涉及的列的数据类型不必相同,但这些数据类型必须相容。计算连接条件的方式与计算其他搜索条件的方式相同,并且使用相同的比较规则。

当不指定连接条件时,则返回 FROM 子句中列出的表中行的所有组合,即使这些行可能完全不相关,这就是前面介绍的交叉连接,交叉连接查询的结果称为表的交叉积。

内连接只保留交叉积中满足指定连接条件的行。如果某行在一个表中存在,但在另一个表中不存在,则结果表中不包括该信息。

从概念上讲,DBMS 执行连接操作的过程是:首先取表 1 的第一个元组,然后从头开始扫描表 2,逐一查找满足连接条件的元组,找到后就将表 1 中的第 1 个元组与该元组拼接起来,形成结果表中的一个元组。表 2 完全查找完毕后,再取表 1 中的第 2 个元组,然后再从头开始扫描表 2,逐一查找满足条件的元组,找到后就将表 1 中的第 2 个元组与该元组拼接起来,形成结果表中的另一个元组。重复这个过程,直到表 1 中的全部元组都处理完毕为止。

【例 5.28】 在 TMS 数据库中,查找所有学生的学号、姓名和他们所选的课程号及成绩。

```
SELECT S.SNO,SNAME,CNO,GRADE
FROM S JOIN SC
ON S.SNO = SC.SNO
```

该语句的查询结果如图 5.27 所示。


结果 消息

	SNO	SNAME	CNO	GRADE
1	20140123	李融	0211	56
2	20152114	杨宏宇	0305	91
3	20152221	孙亚彬	0305	82
4	20180101	袁野	0117	90
5	20180101	袁野	0211	75
6	20180102	刘明明	0211	NULL
7	20180103	王睿	0117	60
8	20180201	王珊	0121	80
9	20180201	王珊	0125	56
10	20180201	王珊	0127	77
11	20180202	张坤	0125	42

图 5.27 例 5.28 查询结果

因为列名 SNO 同时出现在 S 和 SC 两个表中,因此需要对其加以限制,即指明是哪个表里的 SNO。将 SELECT 子句中出现的来自多个表的相同列名加以限定,可以避免可能出

现的潜在错误。

通常情况下，连接两个表至少需要一个连接条件，要连接三个表，则至少需要两个连接条件。为保证不出现无连接的表，连接条件的最小数目通常是需要连接的表的数目减 1。其他条件可以通过 AND 或 OR 操作符进行添加。虽然可以用任意列来构造连接条件，但经常用一个表的主键和另一个表的外键进行连接操作。

【例 5.29】　查询选修了 0211 号课程的学生姓名和该门课程的成绩。

```
SELECT SNAME,GRADE
FROM S JOIN SC
ON S.SNO = SC.SNO
WHERE CNO = '0211'
```

查询结果如图 5.28 所示。

在上例中，JOIN 没有指定类型，则系统默认为内连接。选修了 0211 号课程为选择条件，可写在 WHERE 子句中，也可写在 ON 子句中用逻辑运算符与连接条件相连，不同情况下执行效率有差别。

【例 5.30】　查询选修了 0211 号课程的学生姓名和该门课程的课程名和成绩。

```
SELECT SNAME, CNAME, GRADE
FROM S JOIN SC ON S.SNO = SC.SNO
        JOIN C ON SC.CNO = C.CNO
WHERE C.CNO = '0211'
```

查询结果如图 5.29 所示。

结果　消息

	SNAME	GRADE
1	李融	56
2	袁野	75
3	刘明明	NULL

图 5.28　例 5.29 查询结果

结果　消息

	SNAME	CNAME	GRADE
1	李融	高等数学	56
2	袁野	高等数学	75
3	刘明明	高等数学	NULL

图 5.29　例 5.30 查询结果

上例也可以用下列语句实现：

```
SELECT SNAME, CNAME, GRADE
FROM S JOIN SC ON S.SNO = SC.SNO AND CNO = '0211'
        JOIN C ON SC.CNO = C.CNO
```

连接操作不仅可以在两个表之间进行，也可以是一个表与其自身进行连接，这种连接称为表的自身连接。

【例 5.31】　查询至少选修了课程号为 0121 和 0125 两门课的学生学号。

```
SELECT a.SNO
FROM SC a JOIN SC b
ON a.SNO = b.SNO
AND a.CNO = '0121'
AND b.CNO = '0125'
```

注意：上述语句的 FROM 子句中为参加连接的表定义了别名，这样，就可以在 SELECT 子句和 WHERE 子句中的属性名前分别用这些别名加以区分。而表 SC 有两个相关名，是因为该表在 FROM 子句中用到了两次。

3. 外连接

外连接是内连接和左表/右表中不在内连接中的那些行的并置。外连接中不仅包含那些满足连接条件的元组，而且某些表中不满足条件的元组也会出现在结果集中。也就是说，外连接只限制其中一个表的元组，而不限制另外一个表的元组。

外连接只能用于对两个表执行连接时。当对两个表执行外连接时，可任意将一个表指定为左表而将另一个表指定为右表，外连接有三种类型：左外连接、右外连接和全外连接。

(1) 左外连接。

左外连接包括内连接和左表中未包括在内连接中的那些行，对连接条件左边的表不加限制。语法格式如下：

```
SELECT <目标列表达式> [,...n]
FROM <表 1> LEFT [OUTER] JOIN <表 2>
ON <连接条件表达式>
```

【例 5.32】 查询全体学生信息及他们的选课情况。

```
SELECT S. *, SC. *
FROM S LEFT OUTER JOIN SC
ON S.SNO = SC.SNO
```

查询结果如图 5.30 所示。

结果 消息

	SNO	SNAME	SEX	BIRTHYEAR	SDEPT	SCLASS	SNO	CNO	GRADE
1	20140123	李融	男	1996	软件	软工141	20140123	0211	56
2	20152114	杨宏宇	男	1997	机械	机械151	20152114	0305	91
3	20152221	孙亚彬	女	1997	机械	机械152	20152221	0305	82
4	20180101	袁野	男	2000	软件	软工181	20180101	0117	90
5	20180101	袁野	男	2000	软件	软工181	20180101	0211	75
6	20180102	刘明明	男	2000	软件	软工181	20180102	0211	NULL
7	20180103	王睿	男	2000	软件	软工181	20180103	0117	60
8	20180104	刘平	女	2000	软件	软工181	NULL	NULL	NULL
9	20180201	王珊	女	1999	软件	软工182	20180201	0121	80
10	20180201	王珊	女	1999	软件	软工182	20180201	0125	56
11	20180201	王珊	女	1999	软件	软工182	20180201	0127	77
12	20180202	张坤	男	2000	软件	软工182	20180202	0125	42
13	20181103	姜鹏飞	男	1998	交通	交通181	NULL	NULL	NULL

图 5.30 例 5.32 查询结果

在上例中，若只进行内连接，则没有选课的学生信息就不会出现在结果表中(因不满足连接条件)。使用左外连接，则左表(即 S 表)中不符合连接条件的学生信息也会体现在结果表中。

(2) 右外连接。

包括内连接和右表中未包括在内连接中的那些行。语法格式如下：

```
SELECT <目标列表达式> [,...n]
FROM <表 1> RIGHT [OUTER] JOIN <表 2>
ON <连接条件表达式>
```

(3) 全外连接。

包括内连接以及左表和右表中未包括在内连接中的那些行。语法格式如下：

```
SELECT <目标列表达式> [,...n]
FROM <表 1> FULL [OUTER] JOIN <表 2>
ON <连接条件表达式>
```

5.2.2　子查询

为了提高 SQL 语句的查询功能，通常需要将一个查询语句嵌入另一个 SQL 查询语句的 WHERE 子句或 HAVING 子句的条件中，这种查询语句称为嵌套查询或子查询。

假设要查询和李融同学同一个学院的学生信息，如果不用子查询，则需要首先求出李融同学所在的学院名，此处需要用到第一个 SELECT 语句；然后用这个学院名去构造第二个 SELECT 语句的 WHERE 条件，并通过执行第二个 SELECT 语句得到需要的结果。这个过程需要写两个 SELECT 语句，并分别执行。

这个问题也可以用一个包含子查询的 SELECT 语句完成。将第一个语句放在第二个语句的 WHERE 子句中需要学院名称的位置。在这里，第二个 SELECT 就作为一个外部(父)查询，内部的查询就被称为子查询。当子查询用在 WHERE 或 HAVING 子句中时，必须用小括号括起来。如下所示：

```
SELECT *
FROM S
WHERE SDEPT = (SELECT SDEPT
                FROM S
                WHERE SNAME = '李融');
```

上例中，子查询先运行，其结果用于构造主查询的 WHERE 条件，这种子查询称为无关子查询。

1. 无关子查询

无关子查询的执行不依赖父查询。它的执行过程是首先执行子查询语句，将得到的子查询结果集传递给父查询语句使用。无关子查询中对父查询没有任何引用。

(1) 带有比较运算符的子查询。

带有比较运算符的子查询指主查询与子查询之间用比较运算符进行连接。当用户能确切知道内层查询返回的是单值时，可以用＞、＜、＝、＞＝、＜＝和＜＞等比较运算符。

【例 5.33】　查询成绩最高的学生学号、该成绩的课程号和成绩。

```
SELECT SNO, CNO, GRADE
FROM SC
WHERE GRADE = (SELECT MAX (GRADE)
                FROM SC)
```

	SNO	CNO	GRADE
1	20152114	0305	91

图 5.31　例 5.33 查询结果

子查询首先要确定最高的成绩是多少，并用它和每个学生的成绩进行比较。如果某个学生的成绩和最高成绩相等，则该判定条件为真，相应的行就被返回。查询结果如图 5.31 所示。

注意：带有比较运算符的子查询不能返回一个以上的值。

(2) 带有集合比较运算符的子查询。

子查询返回单值时可以用比较运算符，但返回多值时要用到集合比较运算符。集合比较运算符及其含义如表 5.3 所示。

表 5.3　集合比较运算符及其含义

运算符	含　义
ALL	如果一系列的比较都为 TRUE，则为 TRUE
ANY	如果一系列的比较中任何一个为 TRUE，则为 TRUE
BETWEEN	如果操作数在某个范围之内，则为 TRUE
EXISTS	如果子查询结果包含一些行(结果不空)，则为 TRUE
IN	如果操作数等于表达式列表中的一个，则为 TRUE
NOT	对任何布尔运算的值取反
SOME	如果在一系列比较中有些为 TRUE，则为 TRUE

带有 IN 谓词的子查询指主查询与子查询之间用 IN 进行连接，判断某个属性列值是否在子查询的结果中。由于在嵌套查询中，子查询的结果往往是一个集合，所以谓词 IN 是嵌套查询中最常用到的谓词。用 IN 谓词的嵌套查询可以返回零个、一个或多个 IN 谓词左边的列(或列组合)的值。

【例 5.34】　查询选修了 0211 号课程的学生姓名和所在学院。

```
SELECT SNAME, SDEPT
FROM S
WHERE SNO IN (SELECT SNO
                FROM SC
                WHERE CNO = '0211' )
```

查询结果如图 5.32 所示。

【例 5.35】　查询没有选修任何课程的学生信息。

```
SELECT *
FROM S
WHERE SNO NOT IN (SELECT SNO
                    FROM SC )
```

查询结果如图 5.33 所示。

	SNAME	SDEPT
1	李融	软件
2	袁野	软件
3	刘明明	软件

图 5.32　例 5.34 查询结果

	SNO	SNAME	SEX	BIRTHYEAR	SDEPT	SCLASS
1	20180104	刘平	女	2000	软件	软工181
2	20181103	姜鹏飞	男	1998	交通	交通181

图 5.33　例 5.35 查询结果

该例中子查询首先查找出 SC 表中的学生学号(即选修了课程的学生学号),主查询则列出学号不在子查询结果中的学生信息,即没有选修任何课程的学生信息。

对 IN 谓词,如果任何子查询返回的任意一个非空值都和主查询所要查找的列的非空值相匹配,那么,数据表中包含该值的行都会出现在最终的结果表中。

子查询结果中的空值将不和主查询中的空值匹配。当在 IN 前使用逻辑运算符 NOT,如例 5.35 中所示,若子查询只返回非空值,那么,不和子查询相匹配的主查询的行都会出现在结果表中。如果 NOT IN 子查询中返回一个空值,那么主查询将总是返回一个空的结果表。因此,需要对 NOT IN 子查询中的空值加以注意。

【例 5.36】　查询年龄最小的学生的姓名和出生年份。

```
SELECT SNAME, BIRTHYEAR
FROM   S
WHERE BIRTHYEAR >= ALL(SELECT BIRTHYEAR
                       FROM   S )
```

查询结果如图 5.34 所示。

(3) HAVING 子句中的子查询。

除了 WHERE 子句外,子查询还可以用在分组数据的 HAVING 子句中。

【例 5.37】　查询所选课程的平均成绩高于所有学生成绩平均值的学生学号和平均成绩。

```
SELECT SNO,AVG(GRADE)
FROM SC
GROUP BY SNO
HAVING AVG(GRADE) > (SELECT AVG(GRADE)
                     FROM SC)
```

该语句的子查询结果为 70.9,即 HAVING 子句筛选出平均成绩大于 70.9 分的分组。

最后的查询结果如图 5.35 所示。

结果　消息

	SNAME	BIRTHYEAR
1	袁野	2000
2	刘明明	2000
3	王睿	2000
4	刘平	2000
5	张坤	2000

图 5.34　例 5.36 查询结果

结果　消息

	SNO	(无列名)
1	20152114	91
2	20152221	82
3	20180101	82.5
4	20180201	71

图 5.35　例 5.37 查询结果

注意:除非另外指定了 TOP 语句,否则在子查询中不能含有 ORDER BY 子句,因为其执行结果是传递给外部主查询的,对用户不可见。子查询返回值的个数必须和外部 SELECT 的操作符相匹配;返回的项目数也必须和与之比较的项目数相同。

2. 相关子查询

相关子查询是引用了外部查询列的子查询。逻辑上讲,子查询会为外部查询的每行计算一次。而在物理上,它是一个动态的过程,会随情况的变化有所不同,有不止一种物理方

法来处理相关子查询。在相关子查询中,子查询的执行依赖父查询,多数情况下子查询的WHERE子句中引用了父查询的表。相关子查询的执行过程与无关子查询不同,无关子查询中子查询只执行一次,而相关子查询中的子查询需要重复地执行。

相关子查询执行过程如下:

① 从父查询中取出一个元组,将子查询中引用的元组相关列的值传给内层子查询;

② 执行子查询,得到子查询操作的结果或结果集;

③ 父查询根据子查询返回的结果或结果集得到满足条件的行;

④ 外层父查询依次取出下一个元组,重复步骤①~③,直到外层的元组全部处理完毕。

(1) 带有比较运算符的相关子查询。

与无关子查询类似,相关子查询中同样可以使用比较运算符及集合比较运算符。

【例 5.38】 查询每个学生比其自身平均成绩高的所有成绩,并给出该学生的学号、满足条件的课程号和成绩。

```
SELECT SNO,CNO,GRADE
FROM SC a
WHERE GRADE >(SELECT AVG(GRADE)
                FROM SC b
                WHERE b.SNO = a.SNO)
```

查询结果如图 5.36 所示。

结果 消息

	SNO	CNO	GRADE
1	20180101	0117	90
2	20180201	0121	80
3	20180201	0127	77

图 5.36 例 5.38 查询结果

在此查询语句中,子查询用于计算一个学生所有选修课程的平均成绩,此学生即为外层父查询所扫描到的学生(b.SNO=a.SNO),若学生的平均成绩满足父查询 WHERE 条件,则返回其相关信息。由于父子两层查询都引用了表 SC,为加以区别,分别起别名为 a 和 b。因为子查询计算哪个学生成绩是与父查询相关的,所以这种子查询称为相关子查询。SQL Server 的一种执行过程如下。

① 父查询顺序扫描表 SC,取出第一行元组,将该元组的 SNO 值(a.SNO='20180201')传递给子查询。

② 执行子查询,此时的子查询即相当于执行语句:

```
SELECT AVG(GRADE)
FROM SC b
WHERE b.SNO = '20180201'
```

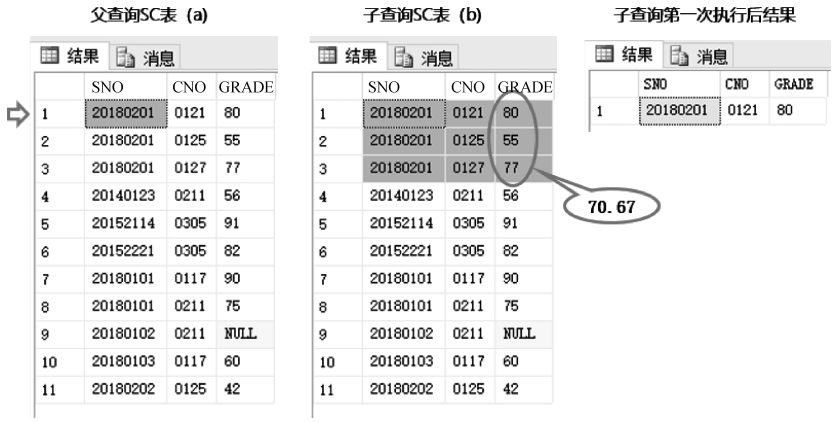
即计算学号为 20180201 的学生所选课程的平均成绩,得到值为 70.67。

③ 将子查询得到的该学生平均成绩 70.67 返回给父查询,则此时父查询 WHERE 语句即为:

```
WHERE GRADE > 70.67
```

若当前元组满足 WHERE 语句条件,则将其保留在父查询的查询结果集中。详细过程如图 5.37 所示:

④ 父查询依次取下一行元组,重复上述步骤,直到 SC 表所有元组扫描完成。



	SNO	CNO	GRADE
1	20180201	0121	80
2	20180201	0125	55
3	20180201	0127	77
4	20140123	0211	56
5	20152114	0305	91
6	20152221	0305	82
7	20180101	0117	90
8	20180101	0211	75
9	20180102	0211	NULL
10	20180103	0117	60
11	20180202	0125	42

	SNO	CNO	GRADE
1	20180201	0121	80

图 5.37　相关子查询执行过程示例

(2) 带有谓词 EXISTS 的相关子查询。

EXISTS 是一个非常强大的谓词，它允许高效地检查指定查询是否产生某些行。EXISTS 的输入是一个子查询，它通常会关联到外部查询，但不是必须的，根据子查询是否返回行，该谓词返回 TRUE 或 FALSE。不同于其他谓词和逻辑表达式，无论输入子查询是否返回行，EXISTS 都不返回 UNKNOWN。

在使用谓词 EXISTS 的子查询中，只要子查询返回非空结果，则父查询的 WHERE 子句将返回逻辑真，否则返回逻辑假。至于返回结果集是什么数据对于子查询是无关紧要的，所以在子查询中的目标列表达式都用符号 *（即使给出字段名也没有实际意义）。

【例 5.39】　查询选修了 0211 号课程的学生姓名。

```
SELECT SNAME
FROM S
WHERE EXISTS (SELECT *
                FROM SC
                WHERE SNO = S.SNO AND CNO = '0211')
```

查询结果如图 5.38 所示。

SQL Server 执行该子查询的一种执行过程如下。

① 父查询顺序扫描表 S，取出第一行元组，将该元组的 SNO 值（即 S.SNO='20140123 '）传递给子查询并执行子查询，则此时子查询即为执行语句：

```
SELECT *
FROM SC
WHERE SNO = '20140123 ' AND CNO = '0211'
```

查询结果如图 5.39 所示。

	SNAME
1	李融
2	袁野
3	刘明明

图 5.38　例 5.39 查询结果

	SNO	CNO	GRADE
1	20140123	0211	56

图 5.39　子查询执行结果

② 子查询返回非空结果集,则父查询的 WHERE 子句返回逻辑真,即输出当前学生的姓名;若子查询返回空集,则父查询的 WHERE 子句返回逻辑假,不输出当前学生的姓名。

③ 父查询依次扫描 S 表,取出下一行元组,重复上述步骤,直到 S 表所有元组扫描完成。

【例 5.40】 查询选修了全部课程的学生姓名。

```
SELECT SNAME
FROM S
WHERE NOT EXISTS (SELECT *
                  FROM C
                  WHERE NOT EXISTS (SELECT *
                                    FROM SC
                                    WHERE SNO = S.SNO AND CNO = C.CNO)
```

一般情况下,有些带 EXSITS 或 NOT EXISTS 的子查询不能被其他形式的子查询等价替换,但所有带 IN、比较运算符、ANY 和 ALL 的子查询都能用带 EXISTS 的子查询等价替换。由于带 EXISTS 的相关子查询只关心内层查询是否有返回值,并不需要查询具体值,有时也是一种高效的方法。

5.2.3 联合查询

SELECT 语句返回的结果是若干条记录的集合。集合有其固有的一些运算,如并、交、差等。从集合运算的角度看,可以将每个 SELECT 语句当作一个集合,于是,可以对任意两个 SELECT 语句进行集合运算。SQL 中提供了并(UNION)、交(INTERSECT)和差(EXCEPT)等几种集合运算。下面分别介绍这几种运算。

1. 集合并(UNION)运算

两个查询的并(UNION)指将两个查询的返回结果集合并到一起,同时去掉重复的记录。显然,并运算的前提是两个查询返回的结果集在结构上要一致,即结果集的字段个数要相等以及字段的数据类型要相容。

【例 5.41】 查询软件学院的所有学生名及开设的所有课程名。

```
SELECT SNAME
FROM S
WHERE SDEPT = '软件'
UNION
SELECT CNAME
FROM C
WHERE CDEPT = '软件'
```

查询结果如图 5.40 所示。

结果 消息

	SNAME
1	离散数学
2	李融
3	刘明明
4	刘平
5	软件测试
6	软件工程
7	数据库原理与应用
8	王睿
9	王珊
10	袁野
11	张坤

图 5.40 例 5.41 查询结果

2. 集合交(INTERSECT)运算

两个查询的交(INTERSECT)指将两个查询的返回结果集中相同的元组组合起来。同样地,交运算也要求两个查询返回的结果集在结

构上要一致。

【例 5.42】　查询软件学院选修了 0211 号课程的学生学号。

```
SELECT SNO
FROM S
WHERE SDEPT = '软件'
INTERSECT
SELECT SNO
FROM SC
WHERE CNO = '0211'
```

3. 集合差(EXCEPT)运算

两个查询的差(EXCEPT)指将属于左查询结果集但不属于右查询结果集的元组组合起来。差运算同样要求两个查询返回的结果集在结构上要一致。

【例 5.43】　查询选修了 0121 号课程但没选修 0127 号课程的学生学号。

```
SELECT SNO
FROM SC
WHERE CNO = '0121'
EXCEPT
SELECT SNO
FROM SC
WHERE CNO = '0127'
```

总之,SELECT 语句既可以完成简单的单表查询,也可以完成复杂的连接查询和嵌套查询。现将其各子句功能总结如下:

(1) SELECT 子句指定查询结果集的目标列。目标列可以直接从数据源中投影得到列名、与列名相关的表达式或数据统计的函数表达式,如算术表达式、标量函数、聚合函数等。目标列也可以是常量,如文字(数字或文本)等。如果目标列中使用了两个基本表(或视图)中相同的列名,要在列名前加表名限定,即使用"<表名>.<列名>"形式表示。

(2) FROM 子句用于指明查询的数据源。查询操作需要的数据源指基本表(或视图)。如果在查询中需要一表多用,则每次使用都需要一个表的别名标识,并在各自使用中用不同的表别名表示。

(3) WHERE 子句通过条件表达式描述关系中元组的选择条件。DBMS 处理语句时,以元组为单位,逐个考察每个元组是否满足条件,将不满足条件的元组筛选掉。

(4) GROUP BY 子句的作用是按分组列的值对结果集分组。分组可以使同组的元组集中在一起,使数据能够分组统计。当 SELECT 子句后的目标列中有统计函数时,如果查询语句中有分组子句,则统计为分组统计,否则为对整个结果集统计。GROUP BY 子句后可以接 HAVING 子句表达式选择条件,组选择条件为带有函数的条件表达式,它决定着整个组记录的取舍条件。

(5) ORDER BY 子句的作用是对结果集进行排序。查询结果集可以按多个排序列进行排序,每个排序列后都可以跟一个排序要求:当排序要求为 ASC 时,结果集的元组按排序列值的升序排序;当排序要求为 DESC 时,结果集的元组按排序列值的降序排列。

5.3 本章小结

SQL是关系数据库的标准语言,其核心部分和关系代数是等价的,但它还有一些重要的功能已经超越了关系代数的表达能力。借助于SQL,人们可以实现数据操纵、数据定义和数据控制等功能。

SQL的数据操纵功能包括数据查询和数据更新两部分。SQL的数据查询用SELECT语句实现,既可以进行简单查询,也可进行多表连接查询和嵌套查询(子查询)。在查询中可以运用标量函数和聚合函数实现相关计算,并可按查询结果的列进行分组和排序。SQL应用是数据库课程学习的重点,通过本章学习,读者应能够熟练掌握SQL的语法,并能在实践中熟练运用SQL实现各种查询要求。

由于SQL是非过程化的语言,要实现SQL对数据库查询等操作的过程控制,可将SQL嵌入其他高级语言中。需要注意的是,各数据库厂商支持的SQL在遵循标准的基础上,也常常会进行不同的扩充或修改。

习题5

应用题

设有一个SPJ数据库,包括S,P,J,SPJ 4个关系模式,具体描述如下文及表5.4～表5.7所示。

设备供应商　S(SNO,SNAME,STATUS,CITY)
零件　P(PNO,PNAME,COLOR,WEIGHT)
工程项目　J(JNO,JNAME,CITY)
供应情况　SPJ(SNO,PNO,JNO,QTY)

表5.4　S(设备供应商)

字段名	SNO	SNAME	STATUS	CITY
数据类型	char	char	char	char
长度	10	12	2	8
描述	供应商编号	供应商名称	供应状态	所在城市

表5.5　P(零件)

字段名	PNO	PNAME	COLOR	WEIGHT
数据类型	char	char	char	Integer
长度	10	12	5	
描述	零件编号	零件名称	零件颜色	零件重量

表 5.6　J(工程项目)

字 段 名	JNO	JNAME	CITY
数据类型	char	char	char
长度	10	10	8
描述	工程编号	工程名	所在城市

表 5.7　SPJ(供应情况)

字 段 名	SNO	PNO	JNO	QTY
数据类型	char	char	char	Integer
长度	10	10	10	
描述	供应商编号	零件编号	工程编号	供应数量

用 SQL 语句完成如下查询：

(1) 查询所有供应商的姓名和所在城市；

(2) 查询重量大于 50g 的所有红色零件的名称；

(3) 查询工程编号以“J210_”开头的所有工程的工程号和所在城市；

(4) 查询没有给任何工程供应零件的设备供应商号；

(5) 查询使用供应商 S1(供应商号)所供应零件的工程号码；

(6) 查询工程 J2(工程编号)使用的各种零件的名称及其数量，并按数量降序排序；

(7) 查询上海厂商供应的所有零件编号(要求使用子查询完成)；

(8) 查询使用上海产的零件的工程名称；

(9) 查询上海供应商的数量；

(10) 查询每个城市的工程数量，给出城市名和数量；

(11) 查询每项工程使用的零件总数，给出工程名和零件的总数；

(12) 查询使用 3 种以上零件的工程编号。