第5章 数据查询与视图管理

学习要点：查询是数据库系统中最常用，也是最重要的功能，它为用户快速、方便地使用数据库中的数据提供了一种有效的方法。视图是根据用户的需求而定义的从基本表导出的虚表。要求掌握单表查询，灵活运用单表查询、多表连接查询，掌握分组与排序的使用方法，理解子查询和联合查询的使用规则，掌握视图的创建和管理。

5.1 简单查询

简单查询是按照一定的条件在单一的表上进行数据查询，还包括查询结果的排序与利用查询结构生成新表。

使用 SQL 中的 SELECT 子句来实现对数据库的查询。SELECT 语句的作用是让服务器从数据库中按用户要求检索数据，并将结果以表格的形式返回给用户。

5.1.1 SELECT 语句结构

完整的 SELECT 语句非常复杂，为了更加清楚地理解 SELECT 语句，从下面几个成分来描述。数据查询 SELECT 语句的语法格式如下。

```
SELECT <子句 1>
    FROM <子句 2>
        [WHERE <表达式 1>]
        [GROUP BY <子句 3>]
        [HAVING <表达式 2>]
        [ORDER BY <子句 4>]
        [LIMIT <子句 5>]
        [UNION <运算符>];
```

说明：

(1) SELECT 子句指定查询结果中需要返回的值。

(2) FROM 子句指定从其中检索行的表或视图。

(3) WHERE 表达式指定查询的搜索条件。

(4) GROUP BY 子句指定查询结果的分组条件。

(5) HAVING 表达式指定分组或集合的查询条件。

(6) ORDER BY 子句指定查询结果的排序方法。

(7) LIMIT 子句可以被用于限制被 SELECT 语句返回的行数。

(8) UNION 操作符将多个 SELECT 语句查询结果组合为一个结果集,该结果集包含联合查询中的所有查询的全部行。

5.1.2 SELECT 子语句

SELECT 子句的语法格式如下。

```
SELECT [ALL|DISTINCT] <目标表达式>[,<目标表达式>][,…]
    FROM  <表或视图名>[,<表或视图名>][,…] [LIMIT n1[,n2]];
```

说明:

(1) ALL 指定表示结果集的所有行,可以显示重复行,ALL 是默认选项。

(2) DISTINCT 指定在结果集中显示唯一行,空值被认为相等,用于消除取值重复的行。ALL 与 DISTINCT 不能同时使用。

(3) LIMIT n1 表示返回最前面的 n1 行数据,n1 表示返回的行数。

(4) LIMIT n1,n2 表示从 n1 行开始,返回 n2 行数据。初始行为 0(从 0 行开始)。n1,n2 必须是非负的整型常量。

(5) 目标表达式为结果集选择的要查询的特定表中的列,可以是星号(*)、表达式、列表、变量等。其中,星号(*)用于返回表或视图的所有列,列表用“表名.列名”来表示,如 student.学号,若只有一个表或多个表中没有相同的列时,表名可以省略。

例 5.1 在数据库 D_sample 中查询 student 表中学生的学号、姓名和性别。SQL 语句如下。

```
use D_sample;
select 学号,姓名,性别 from student;
```

查询结果如图 5.1 所示。

学号	姓名	性别
201907001	张文静	女
201907002	刘海燕	女
201907003	宋志强	男
201907004	马媛	女
201907005	李立波	男
201907006	高峰	男
201907007	梁雅婷	女
201907008	包晓娅	女
201907009	黄岩松	男
201907010	王丹丹	女
201907011	孙倩	女
201907012	乔雨	女

12 rows in set (0.01 sec)

图 5.1 对 student 表的投影查询

例 5.2 在数据库 D_sample 中查询 student 表中学生的全部信息。SQL 语句如下。

```
select * from student;
```

查询结果如图 5.2 所示。

```
mysql> select * from student;
+-----------+--------+------+------------+--------+--------------+
| 学号      | 姓名   | 性别 | 出生日期   | 民族   | 政治面貌     |
+-----------+--------+------+------------+--------+--------------+
| 201907001 | 张文静 | 女   | 1999-02-01 | 汉族   | 共青团员     |
| 201907002 | 刘海燕 | 女   | 2000-10-10 | 汉族   | 共青团员     |
| 201907003 | 宋志强 | 男   | 2000-05-23 | 汉族   | 中共党员     |
| 201907004 | 马媛   | 女   | 2001-04-06 | 回族   | 共青团员     |
| 201907005 | 李立波 | 男   | 2000-11-06 | 汉族   | 共青团员     |
| 201907006 | 高峰   | 男   | 2001-01-12 | 汉族   | 共青团员     |
| 201907007 | 梁雅婷 | 女   | 2001-12-28 | 汉族   | 共青团员     |
| 201907008 | 包晓娅 | 女   | 2000-06-17 | 蒙古族 | 共青团员     |
| 201907009 | 黄岩松 | 男   | 2000-09-23 | 汉族   | 中共党员     |
| 201907010 | 王丹丹 | 女   | 2001-11-25 | 汉族   | 共青团员     |
| 201907011 | 孙倩   | 女   | 2001-03-02 | 满族   | 共青团员     |
| 201907012 | 乔雨   | 女   | 2000-07-23 | 汉族   | 共青团员     |
+-----------+--------+------+------------+--------+--------------+
12 rows in set (0.00 sec)
```

图 5.2　对 student 表全部信息查询

例 5.3　在数据库 D_sample 中查询 student 表前 6 行数据。SQL 语句如下。

```
select * from student limit 6;
```

查询结果如图 5.3 所示。

```
mysql> select * from student limit 6;
+-----------+--------+------+------------+--------+--------------+
| 学号      | 姓名   | 性别 | 出生日期   | 民族   | 政治面貌     |
+-----------+--------+------+------------+--------+--------------+
| 201907001 | 张文静 | 女   | 1999-02-01 | 汉族   | 共青团员     |
| 201907002 | 刘海燕 | 女   | 2000-10-10 | 汉族   | 共青团员     |
| 201907003 | 宋志强 | 男   | 2000-05-23 | 汉族   | 中共党员     |
| 201907004 | 马媛   | 女   | 2001-04-06 | 回族   | 共青团员     |
| 201907005 | 李立波 | 男   | 2000-11-06 | 汉族   | 共青团员     |
| 201907006 | 高峰   | 男   | 2001-01-12 | 汉族   | 共青团员     |
+-----------+--------+------+------------+--------+--------------+
6 rows in set (0.00 sec)
```

图 5.3　对 student 表前 6 行查询

例 5.4　在数据库 D_sample 中查询 student 表中从第 4 行开始的 6 行数据。SQL 语句如下。

```
select * from student limit 3,6;
```

查询结果如图 5.4 所示。

```
mysql> select * from student limit 3,6;
+-----------+--------+------+------------+--------+--------------+
| 学号      | 姓名   | 性别 | 出生日期   | 民族   | 政治面貌     |
+-----------+--------+------+------------+--------+--------------+
| 201907004 | 马媛   | 女   | 2001-04-06 | 回族   | 共青团员     |
| 201907005 | 李立波 | 男   | 2000-11-06 | 汉族   | 共青团员     |
| 201907006 | 高峰   | 男   | 2001-01-12 | 汉族   | 共青团员     |
| 201907007 | 梁雅婷 | 女   | 2001-12-28 | 汉族   | 共青团员     |
| 201907008 | 包晓娅 | 女   | 2000-06-17 | 蒙古族 | 共青团员     |
| 201907009 | 黄岩松 | 男   | 2000-09-23 | 汉族   | 中共党员     |
+-----------+--------+------+------------+--------+--------------+
6 rows in set (0.00 sec)
```

图 5.4　对 student 表中从第 4 行开始的 6 行数据的查询

例 5.5　在数据库 D_sample 中查询 student 表所有学生的民族信息，要求输出的信息不重复。SQL 语句如下。

```
select distinct 民族 from student;
```

查询结果如图 5.5 所示。


```
mysql> select distinct 民族 from student;
+--------+
| 民族   |
+--------+
| 汉族   |
| 回族   |
| 蒙古族 |
| 满族   |
+--------+
4 rows in set (0.01 sec)
```

图 5.5　对 student 表不重复数据的查询

5.1.3　WHERE 子语句

使用 SELECT 进行查询时,如果用户希望设置查询条件来限制返回的数据行,可以通过在 SELECT 语句后使用 WHERE 子句来实现。

WHERE 子句的语法格式如下。

```
WHERE <表达式>;
```

使用 WHERE 子句可以限制查询的范围,提高查询的效率。使用时,WHERE 子句必须紧跟在 FROM 子句之后。WHERE 子句中的查询条件或限定条件可以是比较运算符、模式匹配、范围说明、是否为空值、逻辑运算符。

1. 比较查询

比较查询条件由两个表达式和比较运算符(如表 5.1 所示)组成,系统将根据该查询条件的真假来决定某一条记录是否满足该查询条件,只有满足该查询条件的记录才会出现在最终结果集中。

比较查询条件的格式如下。

```
表达式 1　比较运算符　表达式 2
```

表 5.1　比较运算符

运　算　符	描　　述	表　达　式	运　算　符	描　　述	表　达　式
=	相等	x=y	<=	小于等于	x<=y
<>	不相等	x<>y	>=	大于等于	x>=y
>	大于	x>y	!=	不等于	x!=y
<	小于	x<y	<=>	相等或都等于空	x<=>y

例 5.6　在数据库 D_sample 中查询 student 表中姓名为李立波的学号、姓名和性别的列信息。SQL 语句如下。

```
select 学号,姓名,性别 from student
    where 姓名 = '李立波';
```

查询结果如图 5.6 所示。

```
mysql> select 学号,姓名,性别 from student
    -> where 姓名='李立波';
+----------+--------+--------+
| 学号     | 姓名   | 性别   |
+----------+--------+--------+
| 201907005 | 李立波 | 男     |
+----------+--------+--------+
1 row in set (0.00 sec)
```

图 5.6　对 student 表的比较查询结果

2. 模式匹配

模式匹配常用来返回某种匹配格式的所有记录,通常使用 LIKE 或 REGEXP 关键字来指定模式匹配条件。

1) LIKE 运算符

LIKE 运算符使用通配符来表示字符串需要匹配的模式,通配符及其含义见表 5.2。

表 5.2　常用通配符及其含义

通　配　符	名　　称	描　　述
%	百分号	匹配 0 个或多个任意字符
_	下画线	匹配单个的任意字符

模式匹配条件的格式如下。

```
表达式　[NOT] LIKE　模式表达式
```

例 5.7　在数据库 D_sample 中查询 student 表中姓张的学生信息。SQL 语句如下。

```
select * from student
    where 姓名 like '张%';
```

查询结果如图 5.7 所示。

```
mysql> select * from student
    -> where 姓名 like '张%';
+----------+--------+--------+--------------+--------+--------------+
| 学号     | 姓名   | 性别   | 出生日期     | 民族   | 政治面貌     |
+----------+--------+--------+--------------+--------+--------------+
| 201907001 | 张文静 | 女     | 2000-02-01   | 汉族   | 共青团员     |
+----------+--------+--------+--------------+--------+--------------+
1 row in set (0.00 sec)
```

图 5.7　对 student 表的 LIKE 查询

例 5.8　在数据库 D_sample 中查询 student 表中少数民族的学生信息。SQL 语句如下。

```
select * from student
    where 民族 not like '汉%';
```

查询结果如图 5.8 所示。

```
mysql> select * from student
    -> where 民族 not like '汉%';
+-----------+--------+--------+------------+--------+--------------+
| 学号      | 姓名   | 性别   | 出生日期   | 民族   | 政治面貌     |
+-----------+--------+--------+------------+--------+--------------+
| 201907004 | 马媛   | 女     | 2001-04-06 | 回族   | 共青团员     |
| 201907008 | 包晓娅 | 女     | 2000-06-17 | 蒙古族 | 共青团员     |
| 201907011 | 孙倩   | 女     | 2001-03-02 | 满族   | 共青团员     |
+-----------+--------+--------+------------+--------+--------------+
3 rows in set (0.00 sec)
```

图 5.8　对 student 表的 NOT LIKE 查询

2) REGEXP 运算符

REGEXP 运算符使用通配符来表示字符串需要匹配的模式,通配符及其含义见表 5.3。

表 5.3　常用通配符及其含义

通　配　符	名　　称	描　　述
^	插入号	匹配字符串的开始部分
$	美元	匹配字符串的结束部分
.	句号	匹配字符串(包括回车和新行)
*	乘号	匹配 0 个或多个任意字符
+	加号	匹配单个或多个任意字符
?	问号	匹配 0 个或单个任意字符
()	括号	匹配括号里的内容
{n}	大括号	匹配括号前的内容出现 n 次的序列

模式匹配条件的格式如下。

```
表达式　[NOT] REGEXP　模式表达式
```

例 5.9　在数据库 D_sample 中查询 student 表中姓张的学生信息。SQL 语句如下。

```
select * from student
    where 姓名 regexp '^张';
```

或者

```
select * from student
    where 姓名 regexp '张 + ';
```

查询结果如图 5.9 所示。

```
mysql> select * from student
    -> where 姓名 regexp '^张';
+-----------+--------+--------+------------+--------+--------------+
| 学号      | 姓名   | 性别   | 出生日期   | 民族   | 政治面貌     |
+-----------+--------+--------+------------+--------+--------------+
| 201907001 | 张文静 | 女     | 2000-02-01 | 汉族   | 共青团员     |
+-----------+--------+--------+------------+--------+--------------+
1 row in set (0.04 sec)
```

图 5.9　对 student 表的 REGEXP 查询

3. 范围查询

如果需要返回某一字段的值介于两个指定值之间的所有记录，那么可以使用范围查询条件进行检索。范围检索条件主要有以下两种情况。

(1) 使用 BETWEEN…AND…语句指定内含范围条件。

要求返回记录某个字段的值在两个指定值范围以内，同时包括这两个指定的值，通常使用 BETWEEN…AND…语句来指定内含范围条件。

内含范围条件的格式如下。

```
表达式 BETWEEN 表达式 1 AND 表达式 2
```

例 5.10 在数据库 D_sample 中查询 sc 表中成绩为 80～100 分的学生的学号和成绩信息。SQL 语句如下。

```
select 学号,成绩 from sc
    where 成绩 between 80 and 100;
```

查询结果如图 5.10 所示。

```
mysql> select 学号,成绩 from sc
    -> where 成绩 between 80 and 100;
+-----------+--------+
| 学号      | 成绩   |
+-----------+--------+
| 201907001 |   89.0 |
| 201907002 |   92.0 |
| 201907003 |   81.0 |
| 201907003 |   85.0 |
| 201907006 |   91.0 |
+-----------+--------+
5 rows in set (0.00 sec)
```

图 5.10 对 sc 表的范围查询

(2) 使用 IN 语句指定列表查询条件。

包含列表查询条件的查询将返回所有与列表中的任意一个值匹配的记录，通常使用 IN 语句指定列表查询条件。对于查询条件表达式中出现多个条件相同的情况，也可以用 IN 语句来简化。

列表查询条件的格式如下。

```
表达式  IN(表达式[,…])
```

例 5.11 在数据库 D_sample 中查询 course 表中开课学期为第 1 学期和第 2 学期的课程信息。SQL 语句如下。

```
select * from course
    where 开课学期 in('1','2');
```

查询结果如图 5.11 所示。

4. 空值判断查询条件

空值判断查询条件主要用来搜索某一字段为空值的记录，可以使用 IS NULL 或 IS

```
mysql> select * from course
    -> where 开课学期 in('1','2');
+----------+--------------------+--------------------+------+------+--------------+
| 课程号   | 课程名称           | 课程简介           | 课时 | 学分 | 开课学期     |
+----------+--------------------+--------------------+------+------+--------------+
| 07001    | 计算机应用基础     | 掌握计算机基本操作 |    4 |    4 | 1            |
| 07002    | 计算机网络技术基础 | 掌握计算机网络应用 |    4 |    4 | 1            |
| 07003    | 数据库技术基础     | 掌握数据库系统设计 |    4 |    4 | 2            |
| 07004    | 程序设计基础       | 掌握编程思想与方法 |    4 |    4 | 2            |
+----------+--------------------+--------------------+------+------+--------------+
4 rows in set (0.01 sec)
```

图 5.11　对 course 表的范围查询

NOT NULL 关键字来指定查询条件。

注意：IS NULL 不能用“=NULL”代替。

例 5.12　在数据库 D_sample 中查询 course 表中所有课程简介为空的课程信息。SQL 语句如下。

```
select * from course
    where 课程简介 is null;
```

查询结果如图 5.12 所示。

```
mysql> select * from course
    -> where 课程简介 is null;
+----------+--------------+--------------+------+------+--------------+
| 课程号   | 课程名称     | 课程简介     | 课时 | 学分 | 开课学期     |
+----------+--------------+--------------+------+------+--------------+
| 07007    | JAVA程序设计 | NULL         |    4 |    4 | 4            |
+----------+--------------+--------------+------+------+--------------+
1 row in set (0.00 sec)
```

图 5.12　对 course 表的空值判断查询

5. 使用逻辑运算符查询

前面介绍的查询条件还可以通过逻辑运算符组成更为复杂的查询条件，逻辑运算符有 4 个，分别是 NOT、AND、OR 和 XOR。其中，NOT 表示对条件的否定；AND 用于连接两个条件，当两个条件都满足时才返回 TRUE，否则返回 FALSE；OR 也用于连接两个条件，只要有一个条件满足时就返回 TRUE；XOR 同样也用于连接两个条件，只有一个条件满足时才返回 TRUE，当两个条件都满足或都不满足时返回 FALSE。

说明：

(1) 4 种运算的优先级按从高到低的顺序是 NOT、AND、OR 和 XOR，但可以通过括号改变其优先级关系。

(2) 在 MySQL 中，逻辑表达式共有 3 种可能的结果值，分别是 1(TRUE)、0(FALSE)和 NULL。

例 5.13　在数据库 D_sample 中查询 sc 表中成绩为 80～100 分的学号和成绩信息。SQL 语句如下。

```
select 学号,成绩 from sc
    where 成绩>=80 and 成绩<=100;
```

查询结果如图 5.13 所示。

```
mysql> select 学号,成绩 from sc
    -> where 成绩>=80 and 成绩<=100;
+-----------+--------+
| 学号      | 成绩   |
+-----------+--------+
| 201907001 |   89.0 |
| 201907002 |   92.0 |
| 201907003 |   81.0 |
| 201907003 |   85.0 |
| 201907006 |   91.0 |
+-----------+--------+
5 rows in set (0.00 sec)
```

图 5.13　对 sc 表的逻辑运算符查询

5.1.4　ORDER BY 子语句

当使用 SELECT 语句查询时，如果希望查询结果能够按照其中的一个或多个字段进行排序，这时可以通过在 SELECT 语句后跟一个 ORDER BY 子句来实现。排序有两种方式：一种是升序，使用 ASC 关键字来指定；一种是降序，使用 DESC 关键字来指定。如果没有指定顺序，系统将默认使用升序。

ORDER BY 子句的语法格式如下。

```
ORDER BY <字段名> [ASC|DESC][,…];
```

例 5.14　在数据库 D_sample 中查询 course 表中开课学期按照升序排列的课程信息。SQL 语句如下。

```
select * from course
    order by 开课学期;
```

查询结果如图 5.14 所示。

```
mysql> select * from course
    -> order by 开课学期;
+--------+--------------------+--------------------+------+------+----------+
| 课程号 | 课程名称           | 课程简介           | 课时 | 学分 | 开课学期 |
+--------+--------------------+--------------------+------+------+----------+
| 07001  | 计算机应用基础     | 掌握计算机基本操作 |    4 |    4 | 1        |
| 07002  | 计算机网络技术基础 | 掌握计算机网络应用 |    4 |    4 | 1        |
| 07003  | 数据库技术基础     | 掌握数据库系统设计 |    4 |    4 | 2        |
| 07004  | 程序设计基础       | 掌握编程思想与方法 |    4 |    4 | 2        |
| 07006  | 网页设计           | 掌握DIV+CSS网页布  |    4 |    4 | 3        |
| 07005  | 数据结构           | 掌握基本概念算法描 |    4 |    4 | 4        |
| 07007  | JAVA程序设计       | NULL               |    4 |    4 | 4        |
+--------+--------------------+--------------------+------+------+----------+
7 rows in set (0.00 sec)
```

图 5.14　对 course 表的排序

例 5.15　在数据库 D_sample 中查询 sc 表中选修了“07003”课程的学生成绩信息，成绩按降序进行排序。SQL 语句如下。

```
select * from sc
    where 课程号 = '07003'
        order by 成绩 desc;
```

查询结果如图 5.15 所示。

```
mysql> select * from sc
    -> where 课程号='07003'
    -> order by 成绩 desc;
+-----------+-----------+-------+
| 学号      | 课程号    | 成绩  |
+-----------+-----------+-------+
| 201907002 | 07003     |  92.0 |
| 201907001 | 07003     |  78.0 |
+-----------+-----------+-------+
2 rows in set (0.00 sec)
```

图 5.15　对 sc 表的排序

5.1.5　GROUP BY 子语句

使用 SELECT 进行查询时,如果用户希望将数据记录依据设置的条件分成多个组,可以通过在 SELECT 语句后使用 GROUP BY 子句来实现。如果 SELECT 子句<目标表达式>中包含聚合函数,则 GROUP BY 将计算每组的汇总值。指定 GROUP BY 时,选择列表中任意非聚合表达式内的所有列都应包含在 GROUP BY 列表中,或者 GROUP BY 表达式必须与选择列表的表达式完全匹配。GROUP BY 子句可以将查询结果按字段或字段组合在行的方向上进行分组,每组在字段或字段组合上具有相同的聚合值。如果聚合函数没有使用 GROUP BY 子句,则只为 SELECT 语句报告一个聚合值。常用的聚合函数见表 5.4。

表 5.4　常用的聚合函数

函　数　名	功　　能
sum()	返回一个数值列或计算列的总和
avg()	返回一个数值列或计算列的平均值
min()	返回一个数值列或计算列的最小值
max()	返回一个数值列或计算列的最大值
count()	返回满足 SELECT 语句中指定条件的记录数
count(*)	返回找到的行数

GROUP BY 子句的语法格式如下。

```
GROUP BY {字段名|表达式}[ASC|DESC][,…]
    [WITH ROLLUP];
```

说明:

(1) 与 ORDER BY 子句中的 ASC 或 DESC 关键字相同。ASC 关键字用来指定升序,DESC 关键字用来指定降序。

(2) ROLLUP 指定在结果集内不仅包含由 GROUP BY 提供的行,还包含汇总行。汇总行在结果集中显示为 NULL,用于表示所有值。按层次结构顺序,从组内的最低级别到最高级别汇总组。组的层次结构取分组时指定使用的顺序。更改列分级的顺序会影响在结果集内生成的行数。

例 5.16　在数据库 D_sample 中统计 student 表中学生的男女人数。SQL 语句如下。

```
select 性别,count(性别) as 人数 from student
    group by 性别;
```

查询结果如图 5.16 所示。

```
mysql> select 性别,count(性别) as 人数 from student
    -> group by 性别;
+--------+--------+
| 性别   | 人数   |
+--------+--------+
| 女     |      8 |
| 男     |      4 |
+--------+--------+
2 rows in set (0.01 sec)
```

图 5.16　对 student 表的分组查询

例 5.17　在数据库 D_sample 中统计 student 表中每个民族的男女人数、总人数，以及学生总人数。SQL 语句如下。

```
select 民族,性别,count( * ) as 人数 from student
    group by 民族,性别
        with rollup;
```

查询结果如图 5.17 所示。

```
mysql> select 民族,性别,count(*) as 人数 from student
    -> group by 民族,性别
    -> with rollup;
+----------+--------+--------+
| 民族     | 性别   | 人数   |
+----------+--------+--------+
| 汉族     | 男     |      4 |
| 汉族     | 女     |      5 |
| 汉族     | NULL   |      9 |
| 回族     | 女     |      1 |
| 回族     | NULL   |      1 |
| 满族     | 女     |      1 |
| 满族     | NULL   |      1 |
| 蒙古族   | 女     |      1 |
| 蒙古族   | NULL   |      1 |
| NULL     | NULL   |     12 |
+----------+--------+--------+
10 rows in set (0.00 sec)
```

图 5.17　对 student 表使用 ROLLUP 分组的查询

5.1.6　HAVING 子语句

当完成数据结果的查询和统计后，若希望对查询和计算后的结果进行进一步的筛选，可以通过在 SELECT 语句后使用 GROUP BY 子句配合 HAVING 子句来实现。

HAVING 子句的语法格式如下。

```
HAVING <表达式>;
```

可以在包含 GROUP BY 子句的查询中使用 WHERE 子句。WHERE 与 HAVING 子句的根本区别在于作用对象不同，WHERE 子句作用于基本表或视图，从中选择满足条件的记录，HAVING 子句作用于组，选择满足条件的组，必须用于 GROUP BY 子句之后，但

GROUP BY 子句可以没有 HAVING 子句。HAVING 与 WHERE 语法类似,但 HAVING 可以包含聚合函数。

例 5.18 在数据库 D_sample 中查询 sc 表中平均成绩在 85 分以上的课程号。SQL 语句如下。

```
select 课程号, avg(成绩) as 平均成绩 from sc
    group by 课程号
        having  avg(成绩)>=85;
```

查询结果如图 5.18 所示。

```
mysql> select 课程号, avg(成绩) as 平均成绩 from sc
    -> group by 课程号
    -> having  avg(成绩)>=85;
+----------+--------------+
| 课程号   | 平均成绩     |
+----------+--------------+
| 07001    |     89.00000 |
| 07003    |     85.00000 |
| 07004    |     91.00000 |
| 07005    |     85.00000 |
+----------+--------------+
4 rows in set (0.00 sec)
```

图 5.18 对 sc 表的限定查询

课堂实践 6:简单查询的应用

(1) 在教务管理系统数据库 D_eams 中,查询学生信息表 T_student 中前 8 条记录。SQL 语句如下。

```
use D_eams;
select * from T_student limit 8;
```

查询结果如图 5.19 所示。

```
mysql> select * from T_student limit 8;
+-----------+--------+------+------------+----------+--------------+
| 学号      | 姓名   | 性别 | 出生日期   | 民族     | 政治面貌     |
+-----------+--------+------+------------+----------+--------------+
| 201907001 | 张文静 | 女   | 2000-02-01 | 汉族     | 共青团员     |
| 201907002 | 刘海燕 | 女   | 2000-10-10 | 汉族     | 共青团员     |
| 201907003 | 宋志强 | 男   | 2000-05-23 | 汉族     | 中共党员     |
| 201907004 | 马媛   | 女   | 2001-04-06 | 回族     | 共青团员     |
| 201907005 | 李立波 | 男   | 2000-11-06 | 汉族     | 共青团员     |
| 201907006 | 高峰   | 男   | 2001-01-12 | 汉族     | 共青团员     |
| 201907007 | 梁雅婷 | 女   | 2001-12-28 | 汉族     | 共青团员     |
| 201907008 | 包晓娅 | 女   | 2000-06-17 | 蒙古族   | 共青团员     |
+-----------+--------+------+------------+----------+--------------+
8 rows in set (0.00 sec)
```

图 5.19 对 T_student 表的前 8 条记录查询

(2) 查询课程信息表 T_course 中的课程名称。SQL 语句如下。

```
select distinct 课程名称 from T_course;
```

查询结果如图 5.20 所示。

```
mysql> select distinct 课程名称 from T_course;
+--------------------------+
| 课程名称                 |
+--------------------------+
| 计算机应用基础           |
| 计算机网络技术基础       |
| 数据库技术基础           |
| 程序设计基础             |
| 数据结构                 |
| 网页设计                 |
+--------------------------+
6 rows in set (0.00 sec)
```

图 5.20　对 T_course 表的不重复记录查询

(3) 查询学生信息表 T_student 中姓“李”的男生的学生信息。SQL 语句如下。

```
select * from T_student
    where 姓名 like '李%' and 性别 = '男';
```

查询结果如图 5.21 所示。

```
mysql> select * from T_student
    -> where 姓名 like '李%'  and 性别='男';
+-----------+--------+--------+-------------+--------+--------------+
| 学号      | 姓名   | 性别   | 出生日期    | 民族   | 政治面貌     |
+-----------+--------+--------+-------------+--------+--------------+
| 201907005 | 李立波 | 男     | 2000-11-06  | 汉族   | 共青团员     |
+-----------+--------+--------+-------------+--------+--------------+
1 row in set (0.01 sec)
```

图 5.21　对 T_student 表的选择查询

(4) 查询学生信息表 T_student 中年龄为 20～25 岁的学生信息。SQL 语句如下。

```
select * from T_student
    where year(now()) - year(出生日期) between 20 and 25;
```

查询结果如图 5.22 所示。

```
mysql> select * from T_student
    -> where year(now())-year(出生日期) between 20 and 25;
+-----------+--------+--------+-------------+----------+--------------+
| 学号      | 姓名   | 性别   | 出生日期    | 民族     | 政治面貌     |
+-----------+--------+--------+-------------+----------+--------------+
| 201907001 | 张文静 | 女     | 2000-02-01  | 汉族     | 共青团员     |
| 201907002 | 刘海燕 | 女     | 2000-10-10  | 汉族     | 共青团员     |
| 201907003 | 宋志强 | 男     | 2000-05-23  | 汉族     | 中共党员     |
| 201907005 | 李立波 | 男     | 2000-11-06  | 汉族     | 共青团员     |
| 201907008 | 包晓娅 | 女     | 2000-06-17  | 蒙古族   | 共青团员     |
| 201907009 | 黄岩松 | 男     | 2000-09-23  | 汉族     | 中共党员     |
| 201907012 | 乔雨   | 女     | 2000-07-23  | 汉族     | 共青团员     |
+-----------+--------+--------+-------------+----------+--------------+
7 rows in set (0.00 sec)
```

图 5.22　对 T_student 表的范围查询

(5) 查询成绩表 T_sc 中考试成绩为 85 分以下的学生的学号。SQL 语句如下。

```
select distinct 学号 from T_sc
    where 成绩<=85;
```

查询结果如图 5.23 所示。

```
mysql> select distinct 学号 from T_sc
    -> where 成绩<=85;
+-----------+
| 学号      |
+-----------+
| 201907001 |
| 201907003 |
+-----------+
2 rows in set (0.00 sec)
```

图 5.23　对 T_sc 表的条件查询

(6) 查询成绩表 T_sc 中有成绩的学生学号和课程号。SQL 语句如下。

```
select 学号,课程号 from T_sc
    where 成绩 is not null;
```

查询结果如图 5.24 所示。

```
mysql> select 学号,课程号 from T_sc
    -> where 成绩 is not null;
+-----------+-----------+
| 学号      | 课程号    |
+-----------+-----------+
| 201907001 | 07001     |
| 201907001 | 07003     |
| 201907002 | 07003     |
| 201907003 | 07002     |
| 201907003 | 07005     |
| 201907006 | 07004     |
+-----------+-----------+
6 rows in set (0.00 sec)
```

图 5.24　对 T_sc 表的空值判断查询

(7) 统计成绩表 T_sc 中选修了课程的学生人数。SQL 语句如下。

```
select count(distinct 学号) as 人数
    from T_sc;
```

查询结果如图 5.25 所示。

```
mysql> select count(distinct 学号) as 人数
    -> from T_sc;
+--------+
| 人数   |
+--------+
|      4 |
+--------+
1 row in set (0.00 sec)
```

图 5.25　对 T_sc 表的统计人数查询

(8) 计算成绩表 T_sc 中 07003 号课程的学生平均成绩。SQL 语句如下。

```
select avg(成绩) as 平均成绩 from T_sc
    where 课程号 = '07003';
```

查询结果如图 5.26 所示。

```
mysql> select avg(成绩) as 平均成绩 from T_sc
    -> where 课程号='07003';
+--------------+
| 平均成绩     |
+--------------+
|     85.00000 |
+--------------+
1 row in set (0.00 sec)
```

图 5.26　对 T_sc 表的统计平均值查询

(9) 统计党团员的人数。SQL 语句如下。

```
select 政治面貌,count(政治面貌) as 人数 from T_student
    group by 政治面貌;
```

查询结果如图 5.27 所示。

```
mysql> select 政治面貌,count(政治面貌) as 人数 from T_student
    -> group by 政治面貌;
+--------------+--------+
| 政治面貌     | 人数   |
+--------------+--------+
| 共青团员     |     10 |
| 中共党员     |      2 |
+--------------+--------+
2 rows in set (0.01 sec)
```

图 5.27　对 T_sc 表的分组统计查询

(10) 统计党团员的男女人数。SQL 语句如下。

```
select 政治面貌,性别,count( * ) as 人数 from T_student
    group by 政治面貌,性别;
```

查询结果如图 5.28 所示。

```
mysql> select 政治面貌,性别,count(*) as 人数 from T_student
    -> group by 政治面貌,性别;
+--------------+--------+--------+
| 政治面貌     | 性别   | 人数   |
+--------------+--------+--------+
| 共青团员     | 女     |      8 |
| 中共党员     | 男     |      2 |
| 共青团员     | 男     |      2 |
+--------------+--------+--------+
3 rows in set (0.00 sec)
```

图 5.28　对 T_student 表的分组统计查询

(11) 查询选修了两门以上课程的学生学号。SQL 语句如下。

```
select 学号 from T_sc
    group by 学号
        having count(课程号)>=2;
```

查询结果如图 5.29 所示。

(12) 查询选修了 07003 号课程的学生学号及其成绩,要求成绩按照由高到低的顺序排列。SQL 语句如下。

```
mysql> select 学号 from T_sc
    -> group by 学号
    -> having count(课程号)>=2;
+-----------+
| 学号      |
+-----------+
| 201907001 |
| 201907003 |
+-----------+
2 rows in set (0.00 sec)
```

图 5.29　对 T_sc 表的分组统计限定查询

```
select 学号,成绩 from T_sc
    where 课程号 = '07003'
        order by 成绩 desc;
```

查询结果如图 5.30 所示。

```
mysql>  select 学号,成绩 from T_sc
    -> where 课程号='07003'
    -> order by 成绩 desc;
+-----------+--------+
| 学号      | 成绩   |
+-----------+--------+
| 201907002 |   92.0 |
| 201907001 |   78.0 |
+-----------+--------+
2 rows in set (0.00 sec)
```

图 5.30　对 T_sc 表的排序查询

5.2　连 接 查 询

连接查询是关系数据库中最主要的查询,主要包括内连接、外连接和交叉连接等。通过连接运算符可以实现多个表查询。当检索数据时,通过连接操作查询出存放在多个表中的不同实体的信息。连接操作给用户带来很大的灵活性,可以在任何时候增加新的数据类型。为不同实体创建新的表,然后通过连接进行查询。

5.2.1　内连接

内连接的连接查询结果集中仅包含满足条件的行,内连接是 MySQL 默认的连接方式,可以把 INNER JOIN 简写成 JOIN,根据所使用的比较方式不同,内连接又分为等值连接、自然连接和不等连接三种。

内连接命令的语法格式如下。

```
FROM <表名 1> [别名 1], <表名 2> [别名 2] [,…]
    WHERE <连接条件表达式> [AND <条件表达式>];
```

或者

```
FROM <表名 1> [别名 1] INNER JOIN <表名 2> [别名 2] ON <连接条件表达式>
    [WHERE <条件表达式>];
```

其中,第一种命令格式的连接类型在 WHERE 子句中指定,第二种命令格式的连接类型在 FROM 子句中指定。

另外,连接条件是指在连接查询中连接两个表的条件。连接条件表达式的一般格式如下。

```
[<表名 1>]<别名 1.列名> <比较运算符> [<表名 2>]<别名 2.列名>
```

比较运算符可以使用等号“=”,此时称作等值连接;也可以使用不等比较运算符,包括>、<、>=、<=、!=、<>,等等,此时为不等值连接。

说明:

(1) FROM 后可跟多个表名,表名与别名之间用空格间隔。

(2) 当连接类型在 WHERE 子句中指定时,WHERE 后一定要有连接条件表达式,即两个表的公共字段相等。

(3) 若不定义别名,表的别名默认为表名,定义别名后使用定义的别名。

(4) 若在输出列或条件表达式中出现两个表的公共字段,则在公共字段名前必须加别名。

例 5.19 在数据库 D_sample 中查询每个学生及其选修课的情况。

学生的基本情况存放在 student 表中,选课情况存放在 sc 表中,所以查询过程涉及上述两个表。这两个表是通过公共字段学号实现内连接的。SQL 语句如下。

```
use D_sample;
select a.*,b.*
    from student a,sc b
        where a.学号 = b.学号;
```

或者

```
select a.*,b.*
    from student a inner join sc b on a.学号 = b.学号;
```

查询结果如图 5.31 所示。

```
mysql> select a.*,b.*
    -> from student a,sc b
    -> where a.学号=b.学号;
+-----------+--------+------+------------+------+--------+-----------+----------+--------+
| 学号      | 姓名   | 性别 | 出生日期   | 民族 | 政治面貌 | 学号      | 课程号   | 成绩   |
+-----------+--------+------+------------+------+--------+-----------+----------+--------+
| 201907001 | 张文静 | 女   | 2000-02-01 | 汉族 | 共青团员 | 201907001 | 07001    |   89.0 |
| 201907001 | 张文静 | 女   | 2000-02-01 | 汉族 | 共青团员 | 201907001 | 07003    |   78.0 |
| 201907002 | 刘海燕 | 女   | 2000-10-10 | 汉族 | 共青团员 | 201907002 | 07003    |   92.0 |
| 201907003 | 宋志强 | 男   | 2000-05-23 | 汉族 | 中共党员 | 201907003 | 07002    |   81.0 |
| 201907003 | 宋志强 | 男   | 2000-05-23 | 汉族 | 中共党员 | 201907003 | 07005    |   85.0 |
| 201907006 | 高峰   | 男   | 2001-01-12 | 汉族 | 共青团员 | 201907006 | 07004    |   91.0 |
+-----------+--------+------+------------+------+--------+-----------+----------+--------+
6 rows in set (0.00 sec)
```

图 5.31 对 student 表和 sc 表的等值连接

若在等值连接中把目标列中的重复字段去掉,则称为自然连接。

例 5.20 用自然连接完成例 5.19 的查询。SQL 语句如下。

```
select student.学号,姓名,性别,出生日期,课程号,成绩
    from student,sc
        where student.学号 = sc.学号;
```


查询结果如图 5.32 所示。

```
mysql> select student.学号,姓名,性别,出生日期,课程号,成绩
    -> from student,sc
    -> where student.学号=sc.学号;
+-----------+--------+------+------------+----------+--------+
| 学号      | 姓名   | 性别 | 出生日期   | 课程号   | 成绩   |
+-----------+--------+------+------------+----------+--------+
| 201907001 | 张文静 | 女   | 2000-02-01 | 07001    |   89.0 |
| 201907001 | 张文静 | 女   | 2000-02-01 | 07003    |   78.0 |
| 201907002 | 刘海燕 | 女   | 2000-10-10 | 07003    |   92.0 |
| 201907003 | 宋志强 | 男   | 2000-05-23 | 07002    |   81.0 |
| 201907003 | 宋志强 | 男   | 2000-05-23 | 07005    |   85.0 |
| 201907006 | 高峰   | 男   | 2001-01-12 | 07004    |   91.0 |
+-----------+--------+------+------------+----------+--------+
6 rows in set (0.00 sec)
```

图 5.32　对 student 表和 sc 表的自然连接

注意：在学号前的表名不能省略，因为学号是 student 和 sc 共有的属性，所以必须加上表名前缀。

例 5.21　查询所有女生的学号、姓名、课程号及成绩信息。SQL 语句如下。

```
select a.学号,姓名,课程号,成绩
    from student a,sc b
        where a.学号 = b.学号 and 性别 = '女';
```

或者

```
select a.学号,姓名,课程号,成绩
    from student a inner join sc b on a.学号 = b.学号
        where 性别 = '女';
```

查询结果如图 5.33 所示。

```
mysql> select a.学号,姓名,课程号,成绩
    -> from student a,sc b
    -> where a.学号=b.学号 and 性别='女';
+-----------+--------+----------+--------+
| 学号      | 姓名   | 课程号   | 成绩   |
+-----------+--------+----------+--------+
| 201907001 | 张文静 | 07001    |   89.0 |
| 201907001 | 张文静 | 07003    |   78.0 |
| 201907002 | 刘海燕 | 07003    |   92.0 |
+-----------+--------+----------+--------+
3 rows in set (0.00 sec)
```

图 5.33　对 student 表和 sc 表的内连接

例 5.22　查询学生的姓名、课程名称和成绩信息。SQL 语句如下。

```
select 姓名,课程名称,成绩
    from student a,course b,sc c
        where a.学号 = c.学号 and b.课程号 = c.课程号;
```

其中，3 个表进行两两连接，a.学号＝c.学号和 b.课程号＝c.课程号是两个连接条件。若 n 个表连接，需要 n－1 个连接条件。

另一种方法为：

```
select a.姓名,b.课程名称,c.成绩
    from student a inner join sc c on a.学号 = c.学号
                inner join course b on b.课程号 = c.课程号;
```

查询结果如图 5.34 所示。

```
mysql> select 姓名,课程名称,成绩
    -> from student a,course b,sc c
    -> where a.学号=c.学号 and b.课程号=c.课程号;
+--------+--------------------------+--------+
| 姓名   | 课程名称                 | 成绩   |
+--------+--------------------------+--------+
| 张文静 | 计算机应用基础           |   89.0 |
| 张文静 | 数据库技术基础           |   78.0 |
| 刘海燕 | 数据库技术基础           |   92.0 |
| 宋志强 | 计算机网络技术基础       |   81.0 |
| 宋志强 | 数据结构                 |   85.0 |
| 高峰   | 程序设计基础             |   91.0 |
+--------+--------------------------+--------+
6 rows in set (0.00 sec)
```

图 5.34　对 student 表、course 表和 sc 表的内连接

5.2.2　外连接

外连接的连接查询结果集中既包含那些满足条件的行，还包含其中某个表的全部行，有 3 种形式的外连接：左外连接、右外连接、全外连接。

左外连接是对连接条件中左边的表不加限制，即在结果集中保留连接表达式左表中的非匹配记录；右外连接是对右边的表不加限制，即在结果集中保留连接表达式右表中的非匹配记录；全外连接对两个表都不加限制，所有两个表中的行都会包括在结果集中。

外连接命令的语法格式如下。

```
FROM <表名 1> LEFT| RIGHT| FULL [OUTER]JOIN <表名 2>
    ON <表名 1.列 1> = <表名 2.列 2>
```

例 5.23　在数据库 D_sample 中查询所有学生信息及其选修的课程号，如果学生未选修任何课程，也要包括其基本信息。SQL 语句如下。

```
select student.*,课程号
    from student left join sc
        on student.学号 = sc.学号;
```

执行该查询时，若学生未选修任何课程，则结果表中相应行的课程号字段值为 NULL。查询结果如图 5.35 所示。

```
mysql> select student.*,课程号
    -> from student left join sc
    -> on student.学号=sc.学号;
+-----------+--------+--------+------------+--------+--------------+----------+
| 学号      | 姓名   | 性别   | 出生日期   | 民族   | 政治面貌     | 课程号   |
+-----------+--------+--------+------------+--------+--------------+----------+
| 201907001 | 张文静 | 女     | 2000-02-01 | 汉族   | 共青团员     | 07001    |
| 201907001 | 张文静 | 女     | 2000-02-01 | 汉族   | 共青团员     | 07003    |
| 201907002 | 刘海燕 | 女     | 2000-10-10 | 汉族   | 共青团员     | 07003    |
| 201907003 | 宋志强 | 男     | 2000-05-23 | 汉族   | 中共党员     | 07002    |
| 201907003 | 宋志强 | 男     | 2000-05-23 | 汉族   | 中共党员     | 07005    |
| 201907004 | 马媛   | 女     | 2001-04-06 | 回族   | 共青团员     | NULL     |
| 201907005 | 李立波 | 男     | 2000-11-06 | 汉族   | 共青团员     | NULL     |
| 201907006 | 高峰   | 男     | 2001-01-12 | 汉族   | 共青团员     | 07004    |
| 201907007 | 梁雅婷 | 女     | 2001-12-28 | 汉族   | 共青团员     | NULL     |
| 201907008 | 包晓娅 | 女     | 2000-06-17 | 蒙古族 | 共青团员     | NULL     |
| 201907009 | 黄岩松 | 男     | 2000-09-23 | 汉族   | 中共党员     | NULL     |
| 201907010 | 王丹丹 | 女     | 2001-11-25 | 汉族   | 共青团员     | NULL     |
| 201907011 | 孙倩   | 女     | 2001-03-02 | 满族   | 共青团员     | NULL     |
| 201907012 | 乔雨   | 女     | 2000-07-23 | 汉族   | 共青团员     | NULL     |
+-----------+--------+--------+------------+--------+--------------+----------+
14 rows in set (0.00 sec)
```

图 5.35　对 student 表和 sc 表的左外连接

例 5.24　查询被选修的成绩信息和所有的课程名称。SQL 语句如下。

```
select sc.*,课程名称
    from sc right join course
        on course.课程号 = sc.课程号;
```

该查询执行时,若某课程未被选修,则结果表中相应行的学号、课程号和成绩字段值均为 NULL。查询结果如图 5.36 所示。

```
mysql> select sc.*,课程名称
    -> from sc right join course
    -> on course.课程号=sc.课程号;
+-----------+----------+--------+--------------------------+
| 学号      | 课程号   | 成绩   | 课程名称                 |
+-----------+----------+--------+--------------------------+
| 201907001 | 07001    |   89.0 | 计算机应用基础           |
| 201907003 | 07002    |   81.0 | 计算机网络技术基础       |
| 201907001 | 07003    |   78.0 | 数据库技术基础           |
| 201907002 | 07003    |   92.0 | 数据库技术基础           |
| 201907006 | 07004    |   91.0 | 程序设计基础             |
| 201907003 | 07005    |   85.0 | 数据结构                 |
| NULL      | NULL     |   NULL | 网页设计                 |
| NULL      | NULL     |   NULL | JAVA程序设计             |
+-----------+----------+--------+--------------------------+
8 rows in set (0.00 sec)
```

图 5.36　对 course 表和 sc 表的右外连接

5.2.3　交叉连接

交叉连接又称为笛卡儿连接,是指两个表之间作笛卡儿积操作,得到结果集的行数是两个表的行数的乘积。

交叉连接命令的语法格式如下。

```
FROM <表名 1>[别名 1],<表名 2>[别名 2]
```

需要连接查询的表名在 FROM 子句中指定,表名之间用英文逗号隔开。

例 5.25 在数据库 D_sample 中 sc 表和 course 表进行交叉连接。SQL 语句如下。

```
select a.*,b.*
    from course a,sc b;
```

此处为了简化表名,分别给两个表指定了别名。一旦表名指定了别名,在该命令中,都必须用别名代替表名。查询结果如图 5.37 所示。

```
mysql> select a.*,b.*
    -> from course a,sc b;
```

课程号	课程名称	课程简介	课时	学分	开课学期	学号	课程号	成绩
07001	计算机应用基础	掌握计算机基本操作	4	4	1	201907001	07001	89.0
07001	计算机应用基础	掌握计算机基本操作	4	4	1	201907001	07003	78.0
07001	计算机应用基础	掌握计算机基本操作	4	4	1	201907002	07003	92.0
07001	计算机应用基础	掌握计算机基本操作	4	4	1	201907003	07002	81.0
07001	计算机应用基础	掌握计算机基本操作	4	4	1	201907003	07005	85.0
07001	计算机应用基础	掌握计算机基本操作	4	4	1	201907006	07004	91.0
07002	计算机网络技术基础	掌握计算机网络应用	4	4	1	201907001	07001	89.0
07002	计算机网络技术基础	掌握计算机网络应用	4	4	1	201907001	07003	78.0
07002	计算机网络技术基础	掌握计算机网络应用	4	4	1	201907002	07003	92.0
07002	计算机网络技术基础	掌握计算机网络应用	4	4	1	201907003	07002	81.0
07002	计算机网络技术基础	掌握计算机网络应用	4	4	1	201907003	07005	85.0
07002	计算机网络技术基础	掌握计算机网络应用	4	4	1	201907006	07004	91.0
07003	数据库技术基础	掌握数据库系统设计	4	4	2	201907001	07001	89.0
07003	数据库技术基础	掌握数据库系统设计	4	4	2	201907001	07003	78.0
07003	数据库技术基础	掌握数据库系统设计	4	4	2	201907002	07003	92.0
07003	数据库技术基础	掌握数据库系统设计	4	4	2	201907003	07002	81.0
07003	数据库技术基础	掌握数据库系统设计	4	4	2	201907003	07005	85.0
07003	数据库技术基础	掌握数据库系统设计	4	4	2	201907006	07004	91.0
07004	程序设计基础	掌握编程思想与方法	4	4	2	201907001	07001	89.0
07004	程序设计基础	掌握编程思想与方法	4	4	2	201907001	07003	78.0
07004	程序设计基础	掌握编程思想与方法	4	4	2	201907002	07003	92.0
07004	程序设计基础	掌握编程思想与方法	4	4	2	201907003	07002	81.0
07004	程序设计基础	掌握编程思想与方法	4	4	2	201907003	07005	85.0
07004	程序设计基础	掌握编程思想与方法	4	4	2	201907006	07004	91.0
07005	数据结构	掌握基本概念算法描	4	4	4	201907001	07001	89.0
07005	数据结构	掌握基本概念算法描	4	4	4	201907001	07003	78.0
07005	数据结构	掌握基本概念算法描	4	4	4	201907002	07003	92.0
07005	数据结构	掌握基本概念算法描	4	4	4	201907003	07002	81.0
07005	数据结构	掌握基本概念算法描	4	4	4	201907003	07005	85.0
07005	数据结构	掌握基本概念算法描	4	4	4	201907006	07004	91.0
07006	网页设计	掌握DIV+CSS网页布	4	4	3	201907001	07001	89.0
07006	网页设计	掌握DIV+CSS网页布	4	4	3	201907001	07003	78.0
07006	网页设计	掌握DIV+CSS网页布	4	4	3	201907002	07003	92.0
07006	网页设计	掌握DIV+CSS网页布	4	4	3	201907003	07002	81.0
07006	网页设计	掌握DIV+CSS网页布	4	4	3	201907003	07005	85.0
07006	网页设计	掌握DIV+CSS网页布	4	4	3	201907006	07004	91.0
07007	JAVA程序设计	NULL	4	4	4	201907001	07001	89.0

图 5.37 对 course 表和 sc 表的交叉连接

5.2.4 自连接

连接操作不只是在不同的表之间进行,一张表内还可以进行自身连接操作,即将同一个表的不同行连接起来。自连接可以看作一张表的两个副本之间的连接。在自连接中,必须为表指定两个别名,使之在逻辑上成为两张表。

自连接命令的语法格式如下。

```
FROM <表名 1> [别名 1],<表名 1> [别名 2][,…]
    WHERE <连接条件表达式> [AND <条件表达式>];
```

例 5.26 在数据库 D_sample 中查询同时选修了 07001 和 07003 课程的学生学号。SQL 语句如下。

```
select a.学号
    from sc a,sc b
        where a.学号 = b.学号
```

```
        and a.课程号 = '07001'
        and b.课程号 = '07003';
```


查询结果如图 5.38 所示。

```
mysql> select a.学号
    -> from sc a,sc b
    -> where a.学号=b.学号
    -> and a.课程号='07001'
    -> and b.课程号='07003';
+-----------+
| 学号      |
+-----------+
| 201907001 |
+-----------+
1 row in set (0.01 sec)
```

图 5.38　对 sc 表的自连接

例 5.27　查询选修相同课程的学生学号、课程号和成绩。SQL 语句如下。

```
select distinct a.学号,a.课程号,a.成绩
    from sc a,sc b
        where a.课程号 = b.课程号 and a.学号<> b.学号
```

查询结果如图 5.39 所示。

```
mysql> select distinct a.学号,a.课程号,a.成绩
    -> from sc a,sc b
    -> where a.课程号=b.课程号 and a.学号<>b.学号;
+-----------+-----------+--------+
| 学号      | 课程号    | 成绩   |
+-----------+-----------+--------+
| 201907001 | 07003     |   78.0 |
| 201907002 | 07003     |   92.0 |
+-----------+-----------+--------+
2 rows in set (0.00 sec)
```

图 5.39　对 sc 表的自连接

5.2.5　多表连接

在进行内连接时,有时候出于某种特殊需要,可能涉及三张表甚至更多表进行连接。三张表甚至更多表进行连接和两张表连接的方法基本是相同的,先把两张表连接成一个大表,再将其和第三张表进行连接,以此类推。

课堂实践 7:连接查询的应用

(1) 在教务管理系统数据库 D_eams 中,查询“宋志强”同学所选课程的成绩。SQL 语句如下。

```
use D_eams;
select 成绩
    from T_student a,T_sc b
        where a.学号 = b.学号 and a.姓名 = '宋志强';
```

查询结果如图 5.40 所示。

```
mysql> select 成绩
    -> from T_student a,T_sc b
    -> where a.学号=b.学号 and a.姓名='宋志强';
+--------+
| 成绩   |
+--------+
|   81.0 |
|   85.0 |
+--------+
2 rows in set (0.00 sec)
```

图 5.40　对 T_student 和 T_sc 表的连接查询

(2) 查询至少选修一门课程的女学生姓名。SQL 语句如下。

```
select distinct 姓名
    from T_student a,T_sc b
        where a.学号 = b.学号 and 性别 = '女';
```

查询结果如图 5.41 所示。

```
mysql> select distinct 姓名
    -> from T_student a,T_sc b
    -> where a.学号=b.学号 and 性别='女';
+-----------+
| 姓名      |
+-----------+
| 张文静    |
| 刘海燕    |
+-----------+
2 rows in set (0.01 sec)
```

图 5.41　对 T_student 和 T_sc 表的连接查询

(3) 查询姓宋的学生选修的课程名称。SQL 语句如下。

```
select 课程名称
    from T_student a,T_course b,T_sc c
        where a.学号 = c.学号 and b.课程号 = c.课程号 and 姓名 like '宋%';
```

查询结果如图 5.42 所示。

```
mysql> select 课程名称
    -> from T_student a,T_course b,T_sc c
    -> where a.学号=c.学号 and b.课程号=c.课程号 and 姓名 like '宋%';
+----------------------------+
| 课程名称                   |
+----------------------------+
| 计算机网络技术基础         |
| 数据结构                   |
+----------------------------+
2 rows in set (0.00 sec)
```

图 5.42　对三个表的连接查询

(4) 查询选修了课程号为 07003 的课程且成绩在 80 分以上的学生姓名及成绩。SQL 语句如下。

```
select 姓名,成绩
    from T_student a,T_sc b
        where a.学号 = b.学号 and 课程号 = '07003' and 成绩>=80;
```

查询结果如图 5.43 所示。

```
mysql> select 姓名,成绩
    -> from T_student a,T_sc b
    -> where a.学号=b.学号 and 课程号='07003' and 成绩>=80;
+----------+--------+
| 姓名     | 成绩   |
+----------+--------+
| 刘海燕   |   92.0 |
+----------+--------+
1 row in set (0.00 sec)
```

图 5.43　对两个表连接的选择查询

(5) 查询选修了数据库技术基础课程且成绩在 80 分以上的学生学号、姓名、课程名称及成绩。SQL 语句如下。

```
select a.学号,姓名,课程名称,成绩
    from T_student a,T_course b,T_sc c
        where a.学号 = c.学号 and b.课程号 = c.课程号
            and 课程名称 = '数据库技术基础' and 成绩> = 80;
```

查询结果如图 5.44 所示。

```
mysql> select a.学号,姓名,课程名称,成绩
    -> from T_student a,T_course b,T_sc c
    -> where a.学号=c.学号 and b.课程号=c.课程号
    -> and 课程名称='数据库技术基础' and 成绩>=80;
+-----------+----------+----------------------+--------+
| 学号      | 姓名     | 课程名称             | 成绩   |
+-----------+----------+----------------------+--------+
| 201907002 | 刘海燕   | 数据库技术基础       |   92.0 |
+-----------+----------+----------------------+--------+
1 row in set (0.00 sec)
```

图 5.44　对三个表连接的选择查询

(6) 查询在第 2 学期所开课程的课程名称及成绩。SQL 语句如下。

```
select 课程名称,成绩
    from T_course a,T_sc b
        where a.课程号 = b.课程号 and 开课学期 = '2';
```

查询结果如图 5.45 所示。

```
mysql> select 课程名称,成绩
    -> from T_course a,T_sc b
    -> where a.课程号=b.课程号 and 开课学期='2';
+----------------------+--------+
| 课程名称             | 成绩   |
+----------------------+--------+
| 数据库技术基础       |   78.0 |
| 数据库技术基础       |   92.0 |
| 程序设计基础         |   91.0 |
+----------------------+--------+
3 rows in set (0.00 sec)
```

图 5.45　对两个表连接的选择查询

(7) 查询选修课程名称为数据库技术基础的学生学号和姓名。SQL 语句如下。

```
select a.学号,姓名
    from T_student a,T_course b,T_sc c
        where a.学号 = c.学号 and b.课程号 = c.课程号
            and 课程名称 = '数据库技术基础';
```

查询结果如图 5.46 所示。

```
mysql> select a.学号,姓名
    -> from T_student a,T_course b,T_sc c
    -> where a.学号=c.学号 and b.课程号=c.课程号
    -> and 课程名称='数据库技术基础';
+-----------+-----------+
| 学号      | 姓名      |
+-----------+-----------+
| 201907001 | 张文静    |
| 201907002 | 刘海燕    |
+-----------+-----------+
2 rows in set (0.00 sec)
```

图 5.46　对三个表连接的选择查询

(8) 查询课程成绩及格的女同学的学生信息及课程号与成绩。SQL 语句如下。

```
select a.*,b.课程号,成绩
    from T_student a,T_sc b
        where a.学号 = c.学号
            and 成绩>=60 and 性别 = '女';
```

查询结果如图 5.47 所示。

```
mysql> select a.*,b.课程号,成绩
    -> from T_student a,T_sc b
    -> where a.学号=b.学号
    -> and 成绩>=60 and 性别='女';
+-----------+----------+--------+------------+--------+--------------+----------+--------+
| 学号      | 姓名     | 性别   | 出生日期   | 民族   | 政治面貌     | 课程号   | 成绩   |
+-----------+----------+--------+------------+--------+--------------+----------+--------+
| 201907001 | 张文静   | 女     | 2000-02-01 | 汉族   | 共青团员     | 07001    |   89.0 |
| 201907001 | 张文静   | 女     | 2000-02-01 | 汉族   | 共青团员     | 07003    |   78.0 |
| 201907002 | 刘海燕   | 女     | 2000-10-10 | 汉族   | 共青团员     | 07003    |   92.0 |
+-----------+----------+--------+------------+--------+--------------+----------+--------+
3 rows in set (0.00 sec)
```

图 5.47　对两个表连接的选择查询

(9) 查询高峰的所有选修课的成绩。SQL 语句如下。

```
select 成绩
    from T_student a,T_sc b
        where a.学号 = c.学号
            and 姓名 = '高峰';
```

查询结果如图 5.48 所示。

(10) 查询选修了课程号为 07005 的学生的姓名和成绩。SQL 语句如下。

```
mysql> select 成绩
    -> from T_student a,T_sc b
    -> where a.学号=b.学号
    -> and 姓名='高峰';
+--------+
| 成绩   |
+--------+
|   91.0 |
+--------+
1 row in set (0.01 sec)
```

图 5.48　对两个表连接的选择查询

```
select 姓名,成绩
    from T_student a,T_sc b
        where a.学号 = b.学号 and 课程号 = '07005';
```

查询结果如图 5.49 所示。

```
mysql> select 姓名,成绩
    -> from T_student a,T_sc b
    -> where a.学号=b.学号 and 课程号='07005';
+----------+--------+
| 姓名     | 成绩   |
+----------+--------+
| 宋志强   |   85.0 |
+----------+--------+
1 row in set (0.00 sec)
```

图 5.49　对两个表连接的选择查询

(11) 查询选修了程序设计基础课程的学生的姓名和课程成绩,并按成绩降序排列。SQL 语句如下。

```
select 姓名,成绩
    from T_student a,T_course b,T_sc c
        where a.学号 = c.学号 and b.课程号 = c.课程号
            and 课程名称 = '程序设计基础'
                order by 成绩 desc;
```

查询结果如图 5.50 所示。

```
mysql> select 姓名,成绩
    -> from T_student a,T_course b,T_sc c
    -> where a.学号=c.学号 and b.课程号=c.课程号
    -> and 课程名称='程序设计基础'
    -> order by 成绩 desc;
+--------+--------+
| 姓名   | 成绩   |
+--------+--------+
| 高峰   |   91.0 |
+--------+--------+
1 row in set (0.00 sec)
```

图 5.50　对三个表连接的选择查询

(12) 查询选修 07003 课程的学生的平均年龄。SQL 语句如下。

```
select avg(year(now()) - year(出生日期)) as 平均年龄
    from T_student a,T_sc b
        where a.学号 = b.学号 and 课程号 = '07003';
```

查询结果如图 5.51 所示。

```
mysql> select avg(year(now())-year(出生日期)) as 平均年龄
    -> from T_student a,T_sc b
    -> where a.学号=b.学号 and 课程号='07003';
+--------------+
| 平均年龄     |
+--------------+
|      20.0000 |
+--------------+
1 row in set (0.01 sec)
```

图 5.51　对两个表连接的选择查询

5.3　子　查　询

子查询指在一个 SELECT 查询语句的 WHERE 子句中包含另一个 SELCET 查询语句，或者将一个 SELECT 查询语句嵌入在另一个语句中成为其一部分。在外层的 SELECT 查询语句称为主查询，WHERE 子句中的 SELECT 查询语句被称为子查询。

子查询可描述复杂的查询条件，也称为嵌套查询。嵌套查询一般会涉及两个以上的表，所做的查询有的也可以采用连接查询或者用几个查询语句完成。

在子查询中可以使用 IN 关键字、EXISTS 关键字和比较操作符（ALL 与 ANY）等来连接表数据信息。

5.3.1　IN 子查询

IN 子查询可以用来确定指定的值是否与子查询或列表中的值相匹配。通过 IN（或 NOT IN）引入的子查询结果是一列值。子查询返回结果之后，外部查询将利用这些结果。

IN 子查询的语法格式如下。

```
<字段名> [NOT]IN(子查询)
```

例 5.28　在数据库 D_sample 中查询没有选修计算机网络技术基础的学生学号和姓名。SQL 语句如下。

```
use D_sample;
select 学号,姓名 from student where 学号 not in
    (select 学号 from sc where 课程号 in
        (select 课程号 from course where 课程名称 = '计算机网络技术基础'));
```

查询结果如图 5.52 所示。

例 5.29　查询所有成绩大于 80 分的学生的学号和姓名。SQL 语句如下。

```
select 学号,姓名 from student
    where 学号 in
        (select 学号 from sc where 成绩> 80);
```

查询结果如图 5.53 所示。

```
mysql> select 学号,姓名 from student where 学号 not in
    -> (select 学号 from sc where 课程号 in
    -> (select 课程号 from course where 课程名称='计算机网络技术基础'));
+-----------+-----------+
| 学号      | 姓名      |
+-----------+-----------+
| 201907001 | 张文静    |
| 201907002 | 刘海燕    |
| 201907004 | 马媛      |
| 201907005 | 李立波    |
| 201907006 | 高峰      |
| 201907007 | 梁雅婷    |
| 201907008 | 包晓娅    |
| 201907009 | 黄岩松    |
| 201907010 | 王丹丹    |
| 201907011 | 孙倩      |
| 201907012 | 乔雨      |
+-----------+-----------+
11 rows in set (0.01 sec)
```

图 5.52　IN 子查询

```
mysql> select 学号,姓名 from student
    -> where 学号 in
    -> (select 学号 from sc where 成绩>80);
+-----------+-----------+
| 学号      | 姓名      |
+-----------+-----------+
| 201907001 | 张文静    |
| 201907002 | 刘海燕    |
| 201907003 | 宋志强    |
| 201907006 | 高峰      |
+-----------+-----------+
4 rows in set (0.00 sec)
```

图 5.53　IN 子查询

5.3.2　比较运算符子查询

带有比较运算符的子查询是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>、<、=、>=、<=、!=或<>等比较运算符。

比较运算符子查询的语法格式如下。

```
<字段名> <比较运算符> <子查询>
```

例 5.30　在数据库 D_sample 中查询超过平均年龄的学生的信息。SQL 语句如下。

```
select * from student
    where year(now()) - year(出生日期)>
        (select avg(year(now()) - year(出生日期)) from student);
```

查询结果如图 5.54 所示。

例 5.31　查询选修 07003 号课程,并且分数超过课程平均成绩的学号。SQL 语句如下。

```
select 学号 from sc
    where 课程号 = '07003' and 成绩> =
        (select avg(成绩) from sc);
```

```
mysql> select * from student
    -> where year(now())-year(出生日期)>
    -> (select avg(year(now())-year(出生日期)) from student);
+----------+--------+------+------------+----------+--------------+
| 学号     | 姓名   | 性别 | 出生日期   | 民族     | 政治面貌     |
+----------+--------+------+------------+----------+--------------+
| 201907001 | 张文静 | 女 | 2000-02-01 | 汉族 | 共青团员 |
| 201907002 | 刘海燕 | 女 | 2000-10-10 | 汉族 | 共青团员 |
| 201907003 | 宋志强 | 男 | 2000-05-23 | 汉族 | 中共党员 |
| 201907005 | 李立波 | 男 | 2000-11-06 | 汉族 | 共青团员 |
| 201907008 | 包晓娅 | 女 | 2000-06-17 | 蒙古族 | 共青团员 |
| 201907009 | 黄岩松 | 男 | 2000-09-23 | 汉族 | 中共党员 |
| 201907012 | 乔雨 | 女 | 2000-07-23 | 汉族 | 共青团员 |
+----------+--------+------+------------+----------+--------------+
7 rows in set (0.01 sec)
```

图 5.54　比较运算符子查询

查询结果如图 5.55 所示。

```
mysql> select 学号 from sc
    -> where 课程号='07003' and 成绩>=
    -> (select avg(成绩) from sc);
+----------+
| 学号     |
+----------+
| 201907002 |
+----------+
1 row in set (0.00 sec)
```

图 5.55　比较运算符子查询

5.3.3　ANY 或 ALL 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 ANY 或 ALL 谓词修饰符。而使用 ANY 或 ALL 谓词时，必须同时使用比较运算符。子查询由一个比较运算符引入，后面跟 ANY 或 ALL 的比较运算符，ANY 和 ALL 用于一个值与一组值的比较，以“>”为例，ANY 表示大于一组值中的任意一个，ALL 表示大于一组值中的每一个。

ANY 或 ALL 与比较运算符一起使用的语义见表 5.5。

表 5.5　ANY 和 ALL 的用法和具体含义

用　法	含　义
> ANY	大于子查询结果中的某个值
> ALL	大于子查询结果中的所有值
< ANY	小于子查询结果中的某个值
< ALL	小于子查询结果中的所有值
>= ANY	大于等于子查询结果中的某个值
>= ALL	大于等于子查询结果中的所有值
<= ANY	小于等于子查询结果中的某个值
<= ALL	小于等于子查询结果中的所有值
= ANY	等于子查询结果中的某个值
= ALL	等于子查询结果中的所有值(通常没有实际意义)
!= ANY 或<> ANY	不等于子查询结果中的某个值
!= ALL 或<> ALL	不等于子查询结果中的任何一个值

ANY 或 ALL 子查询的语法格式如下。

```
<字段名> <比较运算符>[ANY|ALL] <子查询>
```

例 5.32 在数据库 D_sample 中查询成绩最高的学号和成绩。SQL 语句如下。

```
select 学号,成绩 from sc
    where 成绩> = all(select 成绩 from sc);
```

查询结果如图 5.56 所示。

```
mysql> select 学号,成绩 from sc
    -> where 成绩>=all(select 成绩 from sc);
+-----------+--------+
| 学号      | 成绩   |
+-----------+--------+
| 201907002 |   92.0 |
+-----------+--------+
1 row in set (0.00 sec)
```

图 5.56 ALL 子查询

例 5.33 查询选修 07002 课程号的成绩高于 07003 课程号的成绩的学生的学号。SQL 语句如下。

```
select 学号  from sc
    where  课程号 = '07002' and 成绩> any
        (select 成绩 from sc
            where 课程号 = '07003');
```

查询结果如图 5.57 所示。

```
mysql> select 学号  from sc
    -> where  课程号='07002' and 成绩>any
    -> (select 成绩 from sc
    ->  where 课程号='07003');
+-----------+
| 学号      |
+-----------+
| 201907003 |
+-----------+
1 row in set (0.00 sec)
```

图 5.57 ANY 子查询

5.3.4 EXISTS 子查询

带有 EXISTS 的子查询不需要返回任何实际数据,而只需要返回一个逻辑真值 TRUE 或逻辑假值 FALSE。也就是说,它的作用是在 WHERE 子句中测试子查询返回的行是否存在。如果存在则返回真值;如果不存在则返回假值。

EXISTS 子查询的语法格式如下。

```
<字段名> [NOT] EXISTS(子查询)
```

例 5.34 在数据库 D_sample 中查询选修了 07003 课程的学生姓名。SQL 语句如下。

```
select 姓名 from student
    where exists
        (select * from sc
            where student.学号 = sc.学号 and 课程号 = '07003');
```

查询结果如图 5.58 所示。

```
mysql> select 姓名 from student
    -> where exists
    -> (select * from sc
    -> where student.学号=sc.学号 and 课程号='07003');
+----------+
| 姓名     |
+----------+
| 张文静   |
| 刘海燕   |
+----------+
2 rows in set (0.00 sec)
```

图 5.58　EXISTS 子查询

例 5.35　查询没有选修 07003 课程的学生姓名。SQL 语句如下。

```
select 姓名 from student
    where not exists
        (select * from sc
            where student.学号 = sc.学号 and 课程号 = '07003');
```

查询结果如图 5.59 所示。

```
mysql> select 姓名 from student
    -> where not exists
    -> (select * from sc
    -> where student.学号=sc.学号 and 课程号='07003');
+----------+
| 姓名     |
+----------+
| 宋志强   |
| 马媛     |
| 李立波   |
| 高峰     |
| 梁雅婷   |
| 包晓娅   |
| 黄岩松   |
| 王丹丹   |
| 孙倩     |
| 乔雨     |
+----------+
10 rows in set (0.01 sec)
```

图 5.59　NOT EXISTS 子查询

课堂实践 8：子查询的应用

(1) 在教务管理系统数据库 D_eams 中，查询选修了课程的学生的学号、姓名，并按学号升序排序。SQL 语句如下。

```
use D_eams;
select 学号,姓名 from T_student
    where 学号 in
```

```
        (select 学号 from T_sc)
            order by 学号 asc;
```


查询结果如图 5.60 所示。

```
mysql> select 学号,姓名 from T_student
    -> where 学号 in
    -> (select 学号 from T_sc)
    -> order by 学号 asc;
+-----------+-----------+
| 学号      | 姓名      |
+-----------+-----------+
| 201907001 | 张文静    |
| 201907002 | 刘海燕    |
| 201907003 | 宋志强    |
| 201907006 | 高峰      |
+-----------+-----------+
4 rows in set (0.00 sec)
```

图 5.60　IN 子查询

(2) 查询未选修任何课程的学生的学号、姓名,并按学号升序排序。SQL 语句如下。

```
select 学号,姓名 from T_student
    where 学号 not in
        (select 学号 from T_sc)
            order by 学号 asc;
```

查询结果如图 5.61 所示。

```
mysql> select 学号,姓名 from T_student
    -> where 学号 not in
    -> (select 学号 from T_sc)
    -> order by 学号 asc;
+-----------+-----------+
| 学号      | 姓名      |
+-----------+-----------+
| 201907004 | 马媛      |
| 201907005 | 李立波    |
| 201907007 | 梁雅婷    |
| 201907008 | 包晓娅    |
| 201907009 | 黄岩松    |
| 201907010 | 王丹丹    |
| 201907011 | 孙倩      |
| 201907012 | 乔雨      |
+-----------+-----------+
8 rows in set (0.00 sec)
```

图 5.61　NOT IN 子查询

(3) 查询所有选修了 07005 课程的学生的姓名。SQL 语句如下。

```
select 姓名 from T_student
    where exists
        (select * from T_sc
            where T_student.学号 = T_sc.学号 and 课程号 = '07005');
```

查询结果如图 5.62 所示。

(4) 用 NOT EXISTS 子查询改写查询未选修任何课程的学生的学号、姓名,并按学号升序排序。SQL 语句如下。

```
mysql> select 姓名 from T_student
    -> where exists
    -> (select * from T_sc
    -> where T_student.学号= T_sc.学号 and 课程号='07005');
+----------+
| 姓名     |
+----------+
| 宋志强   |
+----------+
1 row in set (0.00 sec)
```

图 5.62　EXISTS 子查询

```
select 学号,姓名 from T_student
    where not exists
        (select 学号 from T_sc where T_student.学号 = T_sc.学号)
            order by 学号 asc;
```

查询结果如图 5.63 所示。

```
mysql> select 学号,姓名 from T_student
    -> where not exists
    -> (select 学号 from T_sc where T_student.学号= T_sc.学号)
    -> order by 学号 asc;
+----------+----------+
| 学号     | 姓名     |
+----------+----------+
| 201907004 | 马媛    |
| 201907005 | 李立波  |
| 201907007 | 梁雅婷  |
| 201907008 | 包晓娅  |
| 201907009 | 黄岩松  |
| 201907010 | 王丹丹  |
| 201907011 | 孙倩    |
| 201907012 | 乔雨    |
+----------+----------+
8 rows in set (0.00 sec)
```

图 5.63　NOT EXISTS 子查询

(5) 查询出生日期大于所有男同学出生日期的女同学的姓名。SQL 语句如下。

```
select 姓名 from T_student
    where 出生日期> all
        (select 出生日期 from T_student
            where 性别 = '男');
```

查询结果如图 5.64 所示。

```
mysql> select 姓名 from T_student
    -> where 出生日期>all
    -> (select 出生日期 from T_student
    -> where 性别='男');
+----------+
| 姓名     |
+----------+
| 马媛     |
| 梁雅婷   |
| 王丹丹   |
| 孙倩     |
+----------+
4 rows in set (0.01 sec)
```

图 5.64　ALL 子查询

(6) 查询选修了课程“计算机网络技术基础”的学生的学号和成绩。SQL 语句如下。

```
select 学号,成绩 from T_sc
    where 课程号 =
        (select 课程号 from T_course
            where 课程名称 = '计算机网络技术基础');
```

查询结果如图 5.65 所示。

```
mysql> select 学号,成绩 from T_sc
    -> where 课程号=
    -> (select 课程号 from T_course
    -> where 课程名称='计算机网络技术基础');
+-----------+--------+
| 学号      | 成绩   |
+-----------+--------+
| 201907003 |   81.0 |
+-----------+--------+
1 row in set (0.00 sec)
```

图 5.65 比较运算符子查询

(7) 查询成绩比该课程平均成绩高的学生的学号及成绩。SQL 语句如下。

```
select 学号,成绩 from T_sc
    where 成绩>=
        (select avg(成绩) from T_sc);
```

查询结果如图 5.66 所示。

```
mysql> select 学号,成绩 from T_sc
    -> where 成绩>=
    -> (select avg(成绩) from T_sc);
+-----------+--------+
| 学号      | 成绩   |
+-----------+--------+
| 201907001 |   89.0 |
| 201907002 |   92.0 |
| 201907006 |   91.0 |
+-----------+--------+
3 rows in set (0.00 sec)
```

图 5.66 比较运算符子查询

(8) 查询选修课考试不及格的学生的学号和姓名。SQL 语句如下。

```
select 学号,姓名 from T_student
    where 学号 in
        (select 学号 from T_sc
            where 成绩< 60);
```

查询结果如图 5.67 所示。

```
mysql> select 学号,姓名 from T_student
    -> where 学号 in
    -> (select 学号 from T_sc
    -> where 成绩<60);
Empty set (0.00 sec)
```

图 5.67 IN 子查询

(9) 查询年龄比包晓娅大的学生的学号和姓名。SQL 语句如下。

```
select 学号,姓名 from T_student
    where 出生日期<
        (select 出生日期 from T_student
            where 姓名 = '包晓娅');
```

查询结果如图 5.68 所示。

```
mysql> select 学号,姓名 from T_student
    -> where 出生日期<
    -> (select 出生日期 from T_student
    -> where 姓名='包晓娅');
+-----------+-----------+
| 学号      | 姓名      |
+-----------+-----------+
| 201907001 | 张文静    |
| 201907003 | 宋志强    |
+-----------+-----------+
2 rows in set (0.00 sec)
```

图 5.68　比较运算符子查询

(10) 查询所有与张文静选修了至少一门相同课程的学号、课程号和成绩。SQL 语句如下。

```
select * from T_sc
    where 课程号 in
        (select 课程号 from T_sc
            where 学号 =
                (select 学号 from T_student
                    where 姓名 = '张文静'));
```

查询结果如图 5.69 所示。

```
mysql> select * from T_sc
    -> where 课程号 in
    -> (select 课程号 from T_sc
    -> where 学号=
    -> (select 学号 from T_student
    -> where 姓名='张文静'));
+-----------+----------+--------+
| 学号      | 课程号   | 成绩   |
+-----------+----------+--------+
| 201907001 | 07001    |   89.0 |
| 201907001 | 07003    |   78.0 |
| 201907002 | 07003    |   92.0 |
+-----------+----------+--------+
3 rows in set (0.00 sec)
```

图 5.69　IN 子查询

5.4　联 合 查 询

对于不同的查询操作会生成不同的查询结果集，但在实际应用中会希望这些查询结果集连接到一起，从而组成符合实际需要的数据，此时就可以使用联合查询。使用联合查询可以将两个或更多的结果集组合到一个结果集中，新结果集则包含所有查询结果集中的全部

数据。

SELECT 的查询结果是元组的集合,所以可以对 SELECT 的结果进行集合操作。SQL 提供的集合操作主要包括 3 个:UNION(并操作)、INTERSECT(交操作)、MINUS(差操作)。下面对并操作举一个实例,另外两个集合操作的方法类似。

5.4.1 UNION 操作符

查询的并操作被称为“并集运算”,又称为“合并查询”,是将两个或两个以上的查询结果合并,形成一个具有综合信息的查询结果。使用 UNION 语句可以把两个或两个以上的查询结果集合并为一个结果集。

联合查询的语法格式如下。

```
SELECT <子句 1> UNION [ALL] SELECT <子句 2>;
```

说明:ALL 关键字为可选项。如果在 UNION 子句中使用 ALL 关键字,则返回全部满足匹配的结果;如果不使用 ALL 关键字,则返回结果中删除满足匹配的重复行。在进行联合查询时,查询结果的列标题为第一个查询语句的列标题。因此,必须在第一个查询语句中定义列标题。

例 5.36 在数据库 D_sample 中查询年龄不大于 21 岁的学生和女生的信息。SQL 语句如下。

```
use D_sample;
select * from student
    where year(now()) - year(出生日期)<= 21
union
select * from student
    where 性别 = '女';
```

查询结果如图 5.70 所示。

```
mysql> select * from student
    -> where year(now())-year(出生日期)<=21
    -> union
    -> select * from student
    -> where 性别='女';
+-----------+--------+------+------------+--------+--------------+
| 学号      | 姓名   | 性别 | 出生日期   | 民族   | 政治面貌     |
+-----------+--------+------+------------+--------+--------------+
| 201907001 | 张文静 | 女   | 2000-02-01 | 汉族   | 共青团员     |
| 201907002 | 刘海燕 | 女   | 2000-10-10 | 汉族   | 共青团员     |
| 201907003 | 宋志强 | 男   | 2000-05-23 | 汉族   | 中共党员     |
| 201907004 | 马媛   | 女   | 2001-04-06 | 回族   | 共青团员     |
| 201907005 | 李立波 | 男   | 2000-11-06 | 汉族   | 共青团员     |
| 201907006 | 高峰   | 男   | 2001-01-12 | 汉族   | 共青团员     |
| 201907007 | 梁雅婷 | 女   | 2001-12-28 | 汉族   | 共青团员     |
| 201907008 | 包晓娅 | 女   | 2000-06-17 | 蒙古族 | 共青团员     |
| 201907009 | 黄岩松 | 男   | 2000-09-23 | 汉族   | 中共党员     |
| 201907010 | 王丹丹 | 女   | 2001-11-25 | 汉族   | 共青团员     |
| 201907011 | 孙倩   | 女   | 2001-03-02 | 满族   | 共青团员     |
| 201907012 | 乔雨   | 女   | 2000-07-23 | 汉族   | 共青团员     |
+-----------+--------+------+------------+--------+--------------+
12 rows in set (0.01 sec)
```

图 5.70 联合查询

说明：

(1) 两个查询结果表必须是兼容的，即列的数目相同且对应列的数据类型相同。

(2) 组合查询最终结果表中的列名来自第一个 SELECT 语句。

(3) 如果希望组合查询最终结果表中的行以特定的顺序出现，则可在最后一个 SELECT 语句之后使用 ORDER BY 子句来排序。

例 5.37 在数据库 D_sample 中查询选修课程 07002 或者 07003 的成绩信息。SQL 语句如下。

```
select * from sc
    where 课程号 = '07002'
union
select * from sc
    where 课程号 = '07003';
```

查询结果如图 5.71 所示。

```
mysql> select * from sc
    -> where 课程号='07002'
    -> union
    -> select * from sc
    -> where 课程号='07003';
+-----------+-----------+-------+
| 学号      | 课程号    | 成绩  |
+-----------+-----------+-------+
| 201907003 | 07002     |  81.0 |
| 201907001 | 07003     |  78.0 |
| 201907002 | 07003     |  92.0 |
+-----------+-----------+-------+
3 rows in set (0.01 sec)
```

图 5.71 联合查询

5.4.2 UNION 操作符和 JOIN 操作符的区别与联系

UNION 操作符和 JOIN 操作符都可以将两个表或多个数据表连接在一起，但是，UNION 操作符通常用于连接两个或多个 SELECT 查询语句，而 JOIN 操作符则是在一个 SELECT 查询语句中将两个或多个表连接在一起。

在有些情况下，同一个操作任务，可使用 UNION 或者 JOIN 两种不同的查询方法。

例 5.38 在数据库 D_sample 中使用 UNION 或者 JOIN 查询选修了课程 07002 或者选修成绩在 85 分以上的成绩信息。SQL 语句如下。

```
select distinct a. *
    from sc a join sc b on a.学号 = b.学号
        where (a.课程号 = '07002')
            or (a.成绩>= 85);
```

或者

```
select * from sc
    where 课程号 = '07002'
union
```

```
select * from sc
    where 成绩>=85;
```


查询结果如图 5.72 所示。

```
mysql> select distinct a.*
    -> from sc a join sc b on a.学号=b.学号
    -> where (a.课程号='07002')
    -> or (a.成绩>=85);
+-----------+-----------+-------+
| 学号      | 课程号    | 成绩  |
+-----------+-----------+-------+
| 201907001 | 07001     |  89.0 |
| 201907002 | 07003     |  92.0 |
| 201907003 | 07002     |  81.0 |
| 201907003 | 07005     |  85.0 |
| 201907006 | 07004     |  91.0 |
+-----------+-----------+-------+
5 rows in set (0.00 sec)
```

图 5.72　UNION 与 JOIN 不同的查询方法相同的结果

5.5　视图管理

视图是由一个或多个数据表或视图导出的虚拟表或查询表组成的,是关系数据库系统提供给用户以多种角度观察数据库中数据的重要机制。

5.5.1　视图概述

视图是从一个或者几个基本表或者视图中导出的虚拟表,是从现有基表中抽取若干子集组成用户的“专用表”,这种构造方式必须使用 SQL 中的 SELECT 语句来实现。在定义一个视图时,只是把其定义存放在数据库中,并不直接存储视图对应的数据,直到用户使用视图时才去查找对应的数据。

使用视图具有如下优点。

(1) 简化对数据的操作。视图可以简化用户操作数据的方式。可将经常使用的连接、投影、联合查询和选择查询定义为视图,这样在每次执行相同的查询时,不必重写这些复杂的语句,只要一条简单的查询视图语句即可。视图可向用户隐藏表与表之间复杂的连接操作。

(2) 自定义数据。视图能够让不同用户以不同方式看到不同或相同的数据集,即使不同水平的用户共用同一数据库时也是如此。

(3) 数据集中显示。视图使用户着重于其感兴趣的某些特定数据或所负责的特定任务,可以提高数据操作效率,同时增强了数据的安全性,因为用户只能看到视图中所定义的数据,而不是基本表中的数据。

(4) 导入和导出数据。可以使用视图将数据导入或导出。

(5) 合并分割数据。在某些情况下,由于表中数据量太大,在表的设计过程中可能需要经常对表进行水平分割或垂直分割,然而,这样表结构的变化会对应用程序产生不良的影响。使用视图就可以重新保持原有的结构关系,从而使外模式保持不变,原有的应用程序仍可以通过视图来重载数据。

(6) 安全机制。视图可以作为一种安全机制。通过视图,用户只能查看和修改他们能看到的数据。其他数据库或表既不可见也不可访问。

5.5.2 创建视图

在 SQL 中,使用 CREATE VIEW 语句创建视图。其语法格式如下。

```
CREATE [OR REPLACE] VIEW <视图名> [(字段名[, …])]
    AS SELECT 语句
    [WITH CHECK OPTION];
```

说明:

(1) OR REPLACE 允许在同名的视图中,用新语句替换掉旧语句。

(2) SELECT 语句定义视图的 SELECT 命令。

(3) WITH CHECK OPTION 强制所有通过视图修改的数据满足 SELECT 语句中指定的选择条件。

(4) 视图中的 SELECT 命令不能包含 FROM 子句中的子查询,不能引用系统变量或局部变量。

(5) 在视图定义中命名的表必须已存在,不能引用 TEMPORARY 表,不能创建 TEMPORARY 视图,不能将触发程序与视图关联在一起。

例 5.39 在数据库 D_sample 中定义视图查询学生的姓名、课程名称和成绩。SQL 语句如下。

```
use D_sample;
create view v1
    as
    select 姓名,课程名称,成绩
        from student a,course b,sc c
            where a.学号 = c.学号 and b.课程号 = c.课程号;
```

视图定义后,可以像基本表一样进行查询。例如,若要查询以上定义的视图 v1,可以使用如下 SQL 语句。

```
select * from v1;
```

在安装系统和创建数据库之后,只有系统管理员 sa 和数据库所有者 DBO 具有创建视图的权限,此后他们可以使用 GRANT CREATE VIEW 命令将这个权限授予其他用户。此外,视图创建者必须具有在视图查询中包括的每一列的访问权。

5.5.3 更新视图

在 SQL 语句中,使用 ALTER VIEW 语句修改视图。其语法格式如下。

```
ALTER VIEW <视图名> [(字段名[, …])]
    AS SELECT 语句
    [WITH CHECK OPTION];
```

说明：如果在创建视图时使用了 WITH CHECK OPTION 选项，则在使用 ALTER VIEW 命令时，也必须包括这些选项。

例 5.40 修改例 5.39 中的视图 v1。

```
alter view v1
    as
    select 学号,姓名 from student;
```

5.5.4 删除视图

在 SQL 中，使用 DROP VIEW 语句删除视图。其语法格式如下。

```
DROP VIEW {视图名}[,…];
```

DROP VIEW 语句可以删除多个视图，各视图名之间用逗号分隔。

例 5.41 删除视图 v1。

```
drop view v1;
```

说明：

(1) 删除视图时，将从系统目录中删除视图的定义和有关视图的其他信息，还将删除视图的所有权限。

(2) 使用 DROP TABLE 删除的表上的任何视图都必须用 DROP VIEW 语句删除。

课堂实践 9：教务管理系统中视图管理的应用

(1) 在 D_eams 数据库中创建视图 V_sc 查询成绩大于 90 分的所有学生选修的成绩信息。SQL 语句如下。

```
use D_eams;
create view V_sc
    as
    select * from T_sc
        where 成绩> 90;
```

(2) 创建视图 V_course 查询选修课程号 07005 的所有学生的学号和姓名。SQL 语句如下。

```
create view V_course
    as
    select a.学号,姓名 from T_student a,T_sc b
        where a.学号 = b.学号 and 课程号 = '07005';
```

(3) 创建视图 V_student 查询学生姓名、课程名称、成绩等信息的视图。SQL 语句如下。

```
create view V_student
    as
    select 姓名,课程名称,成绩 from T_student a,T_sc b,T_course c
        where a.学号 = b.学号 and b.课程号 = c.课程号;
```

(4) 修改视图 V_sc,查询成绩大于 90 分且开课学期为第 3 学期的所有学生选修成绩信息。SQL 语句如下。

```
alter view V_sc
    as
    select 成绩 from T_sc a,T_course b
        where a.课程号 = b.课程号 and 成绩> 90 and 开课学期 = '3';
```

(5) 将视图 V_student 删除。SQL 语句如下。

```
drop view V_student;
```

小　　结

本章详细介绍了简单查询、连接查询、子查询、联合查询和视图的创建与管理。数据查询是数据库系统中最常用,也是最重要的功能,它为用户快速、方便地使用数据库中的数据提供了一种有效的方法。视图是根据用户的需求而定义的从基本表导出的虚表。通过本章的学习,重点掌握数据查询的各种方法(简单查询、连接查询、子查询、联合查询),同时掌握视图的创建和管理。应注重培养良好的学习习惯:勤于思考,善于学习,勇于实践,敢于创新。

思考与实践

1. 选择题

(1) 要查询学生信息表中学生姓"张"的学生情况,可用(　　)命令。

A. select * from 学生信息表 where 姓名 like '张%'

B. select * from 学生信息表 where 姓名 like '张_'

C. select * from 学生信息表 where 姓名 like '%张%'

D. select * from 学生信息表 where 姓名='张'

(2) 使用关键字(　　)可以把查询结果中的重复行屏蔽。

A. DISTINCT　　B. UNION　　C. ALL　　D. TOP

(3) WHERE 子句的条件表达式中,可以匹配 0 个到多个字符的通配符是(　　)。

A. _　　B. %　　C. +　　D. $

(4) 模式查找 like '_a%',下面哪个结果是可能的?(　　)

A. aiss　　B. eai　　C. hba　　D. dda

(5) 在 SQL 中,SELECT 语句的"SELECT DISTINCT"表示查询结果中(　　)。

A. 属性名都不相同　　B. 去掉了重复的列

C. 行都不相同　　D. 属性值都不相同

(6) 与 where g between 60 and 80 语句等价的子句是(　　)。

A. where g > 60 and g < 80　　B. where g >= 60 and g < 80

C. where g > 60 and g <= 80　　D. where g >= 60 and g <= 80

(7) 当两个子查询的结果(　　)时,可以执行并、交、差操作。

A. 结构完全不一致　　B. 结构完全一致

C. 结构部分一致　　D. 主键一致

(8) MySQL 中表查询的命令是(　　)。

A. USE　　B. SELECT　　C. UPDATE　　D. DROP

(9) 表示职称为副教授同时性别为男的表达式为(　　)。

A. 职称='副教授' OR 性别='男'　　B. 职称='副教授' AND 性别='男'

C. BETWEEN '副教授' AND '男'　　D. IN ('副教授','男')

(10) 查询员工工资信息时,结果按工资降序排列,正确的是(　　)。

A. ORDER BY 工资　　B. ORDER BY 工资 desc

C. ORDER BY 工资 asc　　D. ORDER BY 工资 dictinct

(11) 使用 SQL 创建视图时,不能使用的关键字是(　　)。

A. ORDER BY　　B. WHERE

C. COMPUTE　　D. WITH CHECK OPTION

(12) 在 SQL 中,CREATE VIEW 语句用于建立视图。如果要求对视图更新时必须满足于查询中的表达式,应当在该语句中使用(　　)短语。

A. WITH UPDATE　　B. WITH INSERT

C. WITH DELETE　　D. WITH CHECK OPTION

(13) 数据库中只存放视图的(　　)。

A. 操作　　B. 对应的数据　　C. 定义　　D. 限制

(14) SQL 的视图是从(　　)中导出的。

A. 基本表　　B. 视图　　C. 基本表或视图　　D. 数据库

(15) 以下关于视图的描述,错误的是(　　)。

A. 视图是从一个或几个基本表或视图导出的虚表

B. 视图并不实际存储数据,只在数据字典中保存其逻辑定义

C. 视图里面的任何数据不可以进行修改

D. SQL 中的 SELECT 语句可以像对基表一样,对视图进行查询

(16) 在视图上不能完成的操作是(　　)。

A. 更新视图　　B. 查询

C. 在视图上定义新的基本表　　D. 在视图上定义新视图

(17) 在 SQL 中,删除一个视图的命令是(　　)。

A. DELETE　　B. DROP　　C. CLEAR　　D. REMOVE

2. 填空题

(1) 用 SELECT 进行模式匹配时,可以使用 like 或 not like 匹配符,但要在条件值中使

用(　　)或(　　)等通配符来配合查询。并且模式匹配只能针对(　　)类型字段查询。

(2) 检索姓名字段中含有'娟'的表达式为：姓名 like (　　)。

(3) HAVING 子句与 WHERE 子句很相似,其区别在于：WHERE 子句作用的对象是(　　),HAVING 子句作用的对象是(　　)。

(4) 视图是从(　　)中导出的表,数据库中实际存放的是视图的(　　),而不是(　　)。

(5) 当对视图进行 UPDATE、INSERT 和 DELETE 操作时,为了保证被操作的行满足视图定义中子查询语句的谓词条件,应在视图定义语句中使用可选择项(　　)。

(6) 如果在视图中删除或修改一条记录,则相应的(　　)也随着视图更新。

3. 实践题

(1) 创建信息表 info(编号,姓名,出生日期,职业)并添加数据。查询信息表 info 中姓钟且年龄大于 80 岁的人员信息。

(2) 创建中国城市信息表 city(编号,城市名称,分值)并添加数据。查询该表,确定中国十大最美城市的排名。

(3) 创建工匠信息表 craftsman(工号,姓名,性别,职务/职称,单位)、行业信息表 industry(编号,行业名称,行业描述,岗位)和工匠与行业关联表 ci(工号,编号,贡献),并为三个表添加数据。根据某位大国工匠的姓名,查询其岗位和所做贡献的信息。

(4) 根据第(3)题中的数据表,查询在制造业工作且岗位为焊工的大国工匠的姓名和所做贡献的信息。

(5) 计算学号为 201907003 的学生的总分和平均分。

(6) 查询课程号为 07003 的课程成绩低于 60 分的学生的学号信息。

(7) 用两种方法查询既没有选修课程号为 07002 的课程,也没有选修课程号为 07004 的课程的学生的学号、课程号和成绩信息。

(8) 查询有两门以上的课程成绩在 80 分以上的学生的姓名信息。