

2021

数据库

SHUJUKU

微信公众号：灰灰考研

第五章 数据查询

- 1、简单查询
- 2、统计查询
- 3、子查询
- 4、多表连接、联合、嵌套查询

一、select语句的基本语法

SELECT [ALL|DISTINCT] 要查询的内容
FROM 表名列表
[WHERE 条件表达式
[GROUP BY 字段名列表 [HAVING 逻辑表达式]
[ORDER BY 字段名 [ASC|DESC]]
[LIMIT [OFFSET,] n];

注意:

- 在上述语法结构中, SELECT 查询语句共有7个子句, 其中 SELECT 和 FROM 子句为必选子句, 而 WHERE、GROUP BY、ORDER BY 和 LIMIT 子句为可选子句, HAVING 子句与 GROUP BY 子句联合使用, 不能单独使用。
- SELECT 子句既可以实现数据的简单查询、结果集的统计查询, 也可以实现多表查询。

二、简单查询

(1) 基本查询

SELECT [ALL|DISTINCT]要查询的内容
FROM 表名列表;

【例5-1】查询“学生选课”数据库的studentinfo表，输出所有学生的详细信息。

注：查询结果要输出表或视图的特定字段时，要明确指出字段名，多个字段名之间用逗号分开。

对应的sql语句：SELECT sno,sname,sgender,sbirth,sclass
FROM studentinfo;

```
mysql> SELECT sno, sname, sgender, sbirth, sclass  
-> FROM studentinfo;
```

sno	sname	sgender	sbirth	sclass
10101001	张永峰	男	1993-08-01 00:00:00	电子商务101
10101002	何小丽	女	1992-11-03 00:00:00	电子商务101
10101003	张宇	男	1992-08-21 00:00:00	电子商务101
10102001	王斌	男	1991-07-14 00:00:00	网络技术101
10102002	包玉明	男	1993-11-15 00:00:00	网络技术101
10102003	孙平平	女	1992-02-27 00:00:00	网络技术101
10102004	翁静静	女	1992-05-09 00:00:00	网络技术101
11101001	刘淑芳	女	1994-06-10 00:00:00	电子商务111
11101002	王亚旭	男	1993-03-18 00:00:00	电子商务111
11101003	高磊	男	1993-05-11 00:00:00	电子商务111

10 rows in set (0.03 sec)

微信公众号：灰灰考研

二、简单查询

(1) 基本查询

在SELECT子句的查询字段列表中，字段的顺序是可以改变的，无需按照表中定义的顺序排列。例如，上述语句可以写为：

SELECT sname,sno,sgender,sbirth,sclass

FROM studentinfo;

当要查询的内容是数据表中所有列的集合时，可以用符号 "*" 来代表所有字段名的集合。例如，上述语句可以写为

SELECT * FROM studentinfo;

```
mysql> SELECT sname, sno, sgender, sbirth, sclass  
-> FROM studentinfo;
```

sname	sno	sgender	sbirth	sclass
张永峰	10101001	男	1993-08-01	电子商务101
何小丽	10101002	女	1992-01-01	电子商务101
张宇	10101003	男	1992-08-21	电子商务101
王斌	10102001	男	1991-07-14	网络技术101
包玉明	10102002	男	1993-11-15	网络技术101
孙平平	10102003	女	1992-02-27	网络技术101
翁静静	10102004	女	1992-05-09	网络技术101
刘淑芳	11101001	女	1994-06-10	电子商务111
王亚旭	11101002	男	1993-03-18	电子商务111
高磊	11101003	男	1993-05-11	电子商务111

10 rows in set (0.00 sec)

二、简单查询

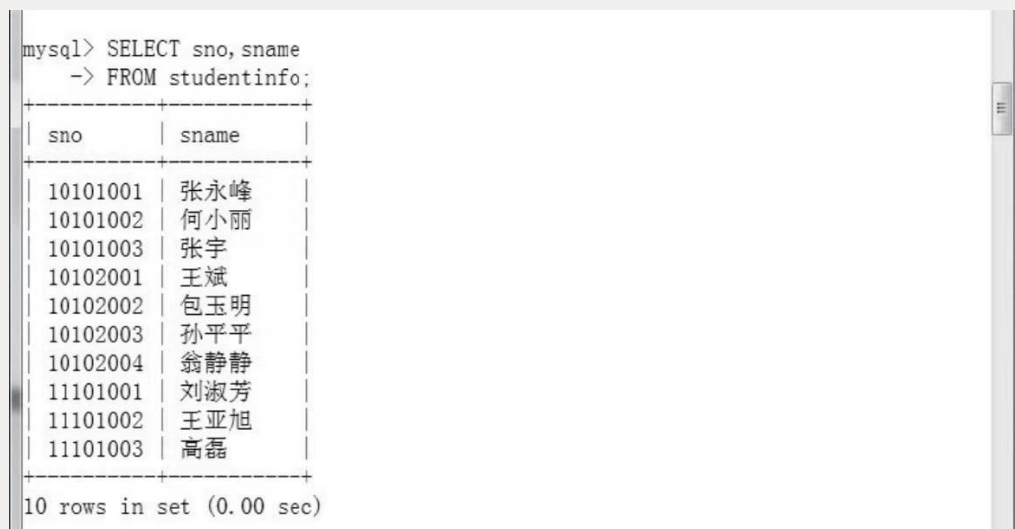
(1) 基本查询

例：查询“学生选课”数据库的studentinfo表，输出所有学生的学号和姓名

对应的SQL语句如下：

```
SELECT sno,sname  
FROM studentinfo;
```

执行结果如图



```
mysql> SELECT sno,sname  
-> FROM studentinfo;
```

sno	sname
10101001	张永峰
10101002	何小丽
10101003	张宇
10102001	王斌
10102002	包玉明
10102003	孙平平
10102004	翁静静
11101001	刘淑芳
11101002	王亚旭
11101003	高磊

10 rows in set (0.00 sec)

二、简单查询

(1) 基本查询

例：查询“学生选课”数据库的studentinfo表，输出所有学生的学号、姓名，以及此次查询的日期和时间。

对应的SQL语句如下：

```
SELECT sno,sname,now()  
FROM studentinfo;
```

执行结果如图：

```
mysql> SELECT sno,sname,now()  
-> FROM studentinfo;
```

sno	sname	now()
10101001	张永峰	2017-08-01 12:01:59
10101002	何小丽	2017-08-01 12:01:59
10101003	张宇	2017-08-01 12:01:59
10102001	王斌	2017-08-01 12:01:59
10102002	包玉明	2017-08-01 12:01:59
10102003	孙平平	2017-08-01 12:01:59
10102004	翁静静	2017-08-01 12:01:59
11101001	刘淑芳	2017-08-01 12:01:59
11101002	王亚旭	2017-08-01 12:01:59
11101003	高磊	2017-08-01 12:01:59

10 rows in set (0.02 sec)

二、简单查询

(1) 基本查询

例：查询“学生选课”数据库的studentinfo表，输出所有学生的学号、姓名、以及此次查询的日期和时间，并分别使用“学生学号”“学生姓名”“查询日期”作为别名。

对应的SQL语句如下：

SELECT sno学生学号,sname AS学生姓名, now() AS查询日期
FROM studentinfo;

1. 别名 AS 别名
2. —
3. —

执行结果如图：

```
mysql> SELECT sno 学生学号,sname AS 学生姓名, now() AS 查询日期
-> FROM studentinfo;
```

学生学号	学生姓名	查询日期
10101001	张永峰	2017-08-01 12:03:06
10101002	何小丽	2017-08-01 12:03:06
10101003	张宇	2017-08-01 12:03:06
10102001	王斌	2017-08-01 12:03:06
10102002	包玉明	2017-08-01 12:03:06
10102003	孙平平	2017-08-01 12:03:06
10102004	翁静静	2017-08-01 12:03:06
11101001	刘淑芳	2017-08-01 12:03:06
11101002	王亚旭	2017-08-01 12:03:06
11101003	高磊	2017-08-01 12:03:06

10 rows in set (0.00 sec)

二、简单查询

(1) 基本查询

例：查询“学生选课”数据库的studentinfo表，输出学生所在的班级，每个班级只输出一次。

对应的SQL语句如下：

```
SELECT DISTINCT sclass  
FROM studentinfo;
```

执行结果如图：

```
mysql> SELECT DISTINCT sclass  
-> FROM studentinfo;  
+-----+  
| sclass |  
+-----+  
| 电子商务101 |  
| 网络技术101 |  
| 电子商务111 |  
+-----+  
3 rows in set (0.00 sec)
```

二、简单查询

(2) 使用WHERE子句

WHERE子句可以指定查询条件，用以从数据表中筛选出满足条件的数据行。其语法格式如下：

SELECT [ALL|DISTINCT]要查询的内容

FROM表名列表

WHERE条件表达式;

1+ B

WHERE子句的条件表达式可以使用的运算符如表

运算符分类	运算符	说明
<u>比较运算符</u>	<u>>、>=、=、<、<=、<>、!=、!>、!<</u>	比较字段值的大小
<u>范围运算符</u>	<u>BETWEEN ... AND、NOT BETWEEN ... AND</u>	判断字段值是否在指定范围内
<u>列表运算符</u>	<u>IN、NOT IN</u>	判断字段值是否在指定的列表中
<u>模式匹配运算符</u>	<u>LIKE、NOT LIKE</u>	判断字段值是否和指定的模式字符串匹配
<u>空值判断运算符</u>	<u>IS NULL、IS NOT NULL</u>	判断字段值是否为空
<u>逻辑运算符</u>	<u>AND、OR、NOT</u>	用于多个条件表达式的逻辑连接

二、简单查询

(2) 使用WHERE子句

比较运算符的使用

例：查询“学生选课”数据库的studentinfo表，输出“网络技术101”班学生的详细信息。

对应的SQL语句如下：

```
SELECT *  
FROM studentinfo  
WHERE sclass= '网络技术101';
```

执行结果如图：

```
mysql> SELECT *  
-> FROM studentinfo  
-> WHERE sclass= '网络技术101';
```

sno	sname	sgender	sbirth	sclass
10102001	王斌	男	1991-07-14 00:00:00	网络技术101
10102002	包玉明	男	1993-11-15 00:00:00	网络技术101
10102003	孙平平	女	1992-02-27 00:00:00	网络技术101
10102004	翁静静	女	1992-05-09 00:00:00	网络技术101

4 rows in set (0.00 sec)

微信公众号：灰灰考研

二、简单查询

(2) 使用WHERE子句

范围运算符的使用

例：查询“学生选课”数据库的studentinfo表，输出1992年出生的学生的详细信息

对应的SQL语句如下：

```
SELECT *  
FROM studentinfo  
WHERE sbirth BETWEEN '1992-1-1' AND '1992-12-31';
```

执行结果如图

```
mysql> SELECT *  
-> FROM studentinfo  
-> WHERE sbirth BETWEEN '1992-1-1' AND '1992-12-31';
```

sno	sname	sgender	sbirth	sclass
10101002	何小丽	女	1992-11-03 00:00:00	电子商务101
10101003	张宇	男	1992-08-21 00:00:00	电子商务101
10102003	孙平平	女	1992-02-27 00:00:00	网络技术101
10102004	翁静静	女	1992-05-09 00:00:00	网络技术101

4 rows in set (0.00 sec)

微信公众号：灰灰考研

二、简单查询

(2) 使用WHERE子句

列表运算符的使用

例：查询“学生选课”数据库的studentinfo表，输出学号为10101001、10102001、11101001的学生的详细信息。

对应的SQL语句如下：

```
SELECT *  
FROM studentinfo  
WHERE sno IN ('10101001', '10102001', '11101001');
```

执行结果如图

```
mysql> SELECT *  
-> FROM studentinfo  
-> WHERE sno IN ('10101001', '10102001', '11101001');
```

sno	sname	sgender	sbirth	sclass
10101001	张永峰	男	1993-08-01 00:00:00	电子商务101
10102001	王斌	男	1991-07-14 00:00:00	网络技术101
11101001	刘淑芳	女	1994-06-10 00:00:00	电子商务111

3 rows in set (0.00 sec)

微信公众号：灰灰考研

二、简单查询

(2) 使用WHERE子句

模式匹配运算符的使用

在指定的条件不是很明确的情况下，可以使用LIKE运算符与模式字符串进行匹配运算。其语法格式如下：

字段名 [NOT] LIKE '模式字符串'

通配符

通配符	含义
%	匹配任意长度（0个或多个）的字符串
_	匹配任意单个字符

通配符和字符串必须括在单引号中。

如果要查找的字符串本身就包括通配符，可以用符号“\”将通配符转义为普通字符

二、简单查询

(2) 使用WHERE子句

例：查询“学生选课”数据库的studentinfo表，输出姓“张”的学生的详细信息。
对应的SQL语句如下：

```
SELECT *  
FROM studentinfo  
WHERE sname LIKE '张%';
```

执行结果如图

```
mysql> SELECT *  
-> FROM studentinfo  
-> WHERE sname LIKE '张%';
```

sno	sname	sgender	sbirth	sclass
10101001	张永峰	男	1993-08-01 00:00:00	电子商务101
10101003	张宇	男	1992-08-21 00:00:00	电子商务101

2 rows in set (0.00 sec)

二、简单查询

(2) 使用WHERE子句

空值判断运算符的使用

IS [NOT] NULL运算符用于判断指定字段的值是否为空值。对于空值判断，不能使用比较运算符或模式匹配运算符。

对应的SQL语句如下：

```
SELECT *  
FROM elective  
WHERE score IS NULL;
```

执行结果如图

```
mysql> SELECT *  
-> FROM elective  
-> WHERE score IS NULL;  
+-----+-----+-----+  
| sno   | cno   | score |  
+-----+-----+-----+  
| 10102003 | c004 | NULL |  
| 11101002 | c002 | NULL |  
+-----+-----+-----+  
2 rows in set (0.00 sec)
```

微信公众号：灰灰考研

二、简单查询

(2) 使用WHERE子句

逻辑运算符的使用

查询条件可以是一个条件表达式，也可以是多个条件表达式的组合。逻辑运算符能够连接多个条件表达式，构成一个复杂的查询条件。逻辑运算符包括：AND（逻辑与）、OR（逻辑或）、NOT（逻辑非）。

例：查询“学生选课”数据库的studentinfo表，输出姓“王” 且是“电子商务111”班的学生的信息。

对应的SQL语句如下：

```
SELECT *  
FROM studentinfo  
WHERE sname LIKE '王%' AND sclass = '电子商务111';
```

二、简单查询

(2) 使用WHERE子句

执行结果如图

```
mysql> SELECT *  
-> FROM studentinfo  
-> WHERE sname LIKE '王%' AND sclass = '电子商务111';
```

sno	sname	sgender	sbirth	sclass
11101002	王亚旭	男	1993-03-18 00:00:00	电子商务111

1 row in set (0.00 sec)

例：查询“学生选课”数据库的studentinfo表，输出姓“王”或者是“电子商务111”班的学生信息。

对应的SQL语句如下：

```
SELECT *  
FROM studentinfo  
WHERE sname LIKE '王%' OR sclass = '电子商务111';
```

二、简单查询

(2) 使用WHERE子句

执行结果如图

```
mysql> SELECT *  
      -> FROM studentinfo  
      -> WHERE sname LIKE '王%' OR sclass = '电子商务111';
```

sno	sname	sgender	sbirth	sclass
10102001	王斌	男	1991-07-14 00:00:00	网络技术101
11101001	刘淑芳	女	1994-06-10 00:00:00	电子商务111
11101002	王亚旭	男	1993-03-18 00:00:00	电子商务111
11101003	高磊	男	1993-05-11 00:00:00	电子商务111

4 rows in set (0.02 sec)

例：查询“学生选课”数据库的studentinfo表，输出不是1992年出生的学生的信息。

二、简单查询

(2) 使用WHERE子句

对应的SQL语句如下:

```
SELECT *  
FROM studentinfo  
WHERE NOT (YEAR(sbirth)=1992);
```

执行结果如图

```
mysql> SELECT *  
-> FROM studentinfo  
-> WHERE NOT (YEAR(sbirth)=1992);
```

sno	sname	sgender	sbirth	sclass
10101001	张永峰	男	1993-08-01 00:00:00	电子商务101
10102001	王斌	男	1991-07-14 00:00:00	网络技术101
10102002	包玉明	男	1993-11-15 00:00:00	网络技术101
11101001	刘淑芳	女	1994-06-10 00:00:00	电子商务111
11101002	王亚旭	男	1993-03-18 00:00:00	电子商务111
11101003	高磊	男	1993-05-11 00:00:00	电子商务111

6 rows in set (0.00 sec)

微信公众号: 灰灰考研

二、简单查询

(3) 使用ORDER BY子句

排序

在查询结果集中，数据行是按照它们在表中的顺序进行排列的。我们可以使用ORDER BY子句对查询结果集中的数据行依照指定字段的值重新排列。其语法格式如下：

SELECT [ALL|DISTINCT]要查询的内容
FROM表名列表
[WHERE条件表达式]
ORDER BY字段名[ASC|DESC] ;

例：查询“学生选课”数据库的elective表，输出选修了c001号课程的学生信息，并将查询结果按成绩的降序排序。

对应的SQL语句如下：

```
SELECT *  
FROM elective  
WHERE cno= 'c001'  
ORDER BY score DESC;
```

微信公众号：灰灰考研

二、简单查询

(3) 使用ORDER BY子句

执行结果如图

```
mysql> SELECT * FROM elective WHERE cno= 'c001' ORDER BY score DESC;
```

sno	cno	score
10102003	c001	85
10101002	c001	78
10101001	c001	73
11101002	c001	67
10102001	c001	50
11101001	c001	49

6 rows in set (0.00 sec)

二、简单查询

(3) 使用LIMIT子句

使用LIMIT子句可以指定查询结果从哪一条记录开始，一共查询多少条记录。其语法格式如下：

```
SELECT  [ALL|DISTINCT]要查询的内容  
FROM表名列表  
[WHERE条件表达式]  
[ORDER BY字段名[ASC|DESC]]  
LIMIT [OFFSET,] n;
```

LIMIT子句接受一个或两个整数参数。其中OFFSET代表从第几行记录开始检索，n代表检索多少行记录。需要注意的是，OFFSET可以省略不写，默认取值为0，代表从第一行记录开始检索。

二、简单查询

(3) 使用LIMIT子句

例：查询“学生选课”数据库的studentinfo表，输出前三条学生记录的信息。

```
SELECT *  
FROM studentinfo  
LIMIT 3;
```

执行结果

```
mysql> SELECT *  
-> FROM studentinfo  
-> LIMIT 3;
```

sno	sname	sgender	sbirth	sclass
10101001	张永峰	男	1993-08-01 00:00:00	电子商务101
10101002	何小丽	女	1992-11-03 00:00:00	电子商务101
10101003	张宇	男	1992-08-21 00:00:00	电子商务101

3 rows in set (0.00 sec)

二、简单查询

(3) 使用LIMIT子句

例：查询“学生选课”数据库的studentinfo表，输出表中第五行学生记录的信息。
对应的SQL语句如下：

```
SELECT *  
FROM studentinfo  
LIMIT 4, 1;
```

执行结果如图

```
mysql> SELECT *  
-> FROM studentinfo  
-> LIMIT 4, 1;
```

sno	sname	sgender	sbirth	sclass
10102002	包玉明	男	1993-11-15 00:00:00	网络技术101

```
1 row in set (0.00 sec)
```

三、统计查询

(1) 集合函数

SELECT语句可以通过集合函数和GROUP BY子句、HAVING子句的组合对查询结果集进行求和、求平均值、求最大值、求最小值、分组等统计查询。

集合函数用于对查询结果集中的指定字段进行统计，并输出统计值。常用的集合函数如所示。

集 合 函 数	功 能 描 述
COUNT([DISTINCT ALL]字段*)	计算指定字段中值的个数。COUNT(*)返回满足条件的行数，包括含有空值的行，不能与DISTINCT一起使用
SUM([DISTINCT ALL]字段)	计算指定字段中数据的总和（此字段为数值类型）
AVG([DISTINCT ALL]字段)	计算指定字段中数据的平均值（此字段为数值类型）
MAX([DISTINCT ALL]字段)	计算指定字段中数据的最大值
MIN([DISTINCT ALL]字段)	计算指定字段中数据的最小值

三、统计查询

(1) 集合函数

例：查询“学生选课”数据库的studentinfo表，统计学生总人数。
对应的SQL语句如下：

```
SELECT COUNT(*) AS 学生总人数  
FROM studentinfo;
```

执行结果如图

```
mysql> SELECT COUNT(*) AS 学生总人数  
-> FROM studentinfo;  
+-----+  
| 学生总人数 |  
+-----+  
|          10 |  
+-----+  
1 row in set (0.00 sec)
```

三、统计查询

(1) 集合函数

例：查询“学生选课”数据库的elective表，统计选修了c003号课程的学生人数、总成绩、平均分、最高分和最低分。

对应的SQL语句如下：

```
SELECT COUNT(*) AS学生人数,SUM(score) AS总成绩,  
AVG(score) 平均分,MAX(score) 最高分,MIN(score) 最低分  
FROM elective  
WHERE cno='c003';
```

执行结果如图

```
mysql> SELECT COUNT(*) AS 学生人数,SUM(score) AS 总成绩,  
-> AVG(score) 平均分,MAX(score) 最高分,MIN(score) 最低分  
-> FROM elective  
-> WHERE cno='c003';
```

学生人数	总成绩	平均分	最高分	最低分
5	402	80.4000	95	67

1 row in set (0.00 sec)

三、统计查询

(2) 使用GROUP BY子句

上述示例所进行的查询都是针对整个查询结果集进行的，而GROUP BY子句用于对查询结果集按指定字段的值进行分组，字段值相同的放在一组。集合函数和GROUP BY子句配合使用，可以对查询结果集进行分组统计。其语法格式如下：

```
SELECT  [ALL|DISTINCT]要查询的内容  
FROM表名列表  
[WHERE条件表达式]  
GROUP  BY字段名列表[HAVING条件表达式];
```

三、统计查询

(2) 使用GROUP BY子句

例：查询studentinfo表，分别统计男女生人数。
对应的SQL语句如下：

```
SELECT sgender, COUNT(*) AS 人数
FROM studentinfo
GROUP BY sgender;
```

执行结果如图

```
mysql> SELECT sgender, COUNT(*) AS 人数
-> FROM studentinfo
-> GROUP BY sgender;
```

sgender	人数
女	4
男	6

2 rows in set (0.00 sec)

三、统计查询

(2) 使用GROUP BY子句

例：查询elective表，统计并输出每个学生所选课程数目及平均分。
对应的SQL语句如下：

```
SELECT sno, COUNT(cno) AS选修课程数目, AVG(score) AS平均分
FROM elective
GROUP BY sno;
```

执行结果如图

```
mysql> SELECT sno, COUNT(cno) AS 选修课程数目, AVG(score) AS 平均分
-> FROM elective
-> GROUP BY sno;
```

sno	选修课程数目	平均分
10101001	3	68.3333
10101002	1	78.0000
10101003	1	69.0000
10102001	1	50.0000
10102002	3	79.3333
10102003	4	76.6667
11101001	3	59.3333
11101002	2	67.0000
11101003	3	86.6667

9 rows in set (0.00 sec)

微信公众号：灰灰考研

三、统计查询

(2) 使用GROUP BY子句

例：查询elective表，统计并输出每门课程所选学生人数及最高分。
对应的SQL语句如下：

```
SELECT cno, COUNT(sno) AS所选学生人数, MAX(score) AS最高分
FROM elective
GROUP BY cno;
```

执行结果如图

```
mysql> SELECT cno, COUNT(sno) AS 所选学生人数, MAX(score) AS 最高分
-> FROM elective
-> GROUP BY cno;
```

cno	所选学生人数	最高分
c001	6	85
c002	5	88
c003	5	95
c004	5	82

4 rows in set (0.01 sec)

三、统计查询

(2) 使用GROUP BY子句

注意：

GROUP BY子句常和HAVING子句配合使用。HAVING子句用于对分组后的结果进行条件筛选HAVING子句只能出现在GROUP BY子句后。

WHERE子句和HAVING子句的区别如下。

(1) WHERE子句设置的查询筛选条件在GROUP BY子句之前发生作用，并且条件中不能使用集合函数。

(2) HAVING子句设置的查询筛选条件在GROUP BY子句之后发生作用，并且条件中允许使用集合函数。

当一个语句中同时出现了WHERE子句、GROUP BY子句和HAVING子句，执行顺序如下。

(1) 执行WHERE子句，从数据表中选取满足条件的数据行。

(2) 由GROUP BY子句对选取的数据行进行分组。

(3) 执行集合函数。

(4) 执行HAVING子句，选取满足条件的分组。

三、统计查询

(2) 使用GROUP BY子句

例：查询elective表中每门课成绩都在70~90分的学生的学号。
对应的SQL语句如下：

```
SELECT sno AS每门成绩都在70—90之间的学生
FROM elective
GROUP BY sno
HAVING MIN(score)>=70 AND MAX(score)<=90;
```

结果：

```
mysql> SELECT sno AS 每门成绩都在70—90之间的学生
-> FROM elective
-> GROUP BY sno
-> HAVING MIN(score)>=70 AND MAX(score)<=90;
```

每门成绩都在70—90之间的学生
10101002
11101003

2 rows in set (0.00 sec)

微信公众号：灰灰考研

三、统计查询

(2) 使用GROUP BY子句

例：查询至少选修了三门课程的学生学号。
对应的SQL语句如下：

```
SELECT sno, count(*) 选修课程数
FROM elective
GROUP BY sno
HAVING count(*) >= 3;
```

结果：

```
mysql> SELECT sno, count(*) 选修课程数
-> FROM elective
-> GROUP BY sno
-> HAVING count(*) >= 3;
```

sno	选修课程数
10101001	3
10102002	3
10102003	4
11101001	3
11101003	3

5 rows in set (0.17 sec)

三、统计查询

(2) 使用GROUP BY子句

注意：

GROUP BY子句常和HAVING子句配合使用。HAVING子句用于对分组后的结果进行条件筛选HAVING子句只能出现在GROUP BY子句后。

WHERE子句和HAVING子句的区别如下。

(1) WHERE子句设置的查询筛选条件在GROUP BY子句之前发生作用，并且条件中不能使用集合函数。

(2) HAVING子句设置的查询筛选条件在GROUP BY子句之后发生作用，并且条件中允许使用集合函数。

四、多表查询

在实际查询中，很多情况下用户需要的数据并不全在一个表中，而是存在于多个不同的表中，这时就要使用多表查询。多表查询是通过各个表之间的共同列的相关性来查询数据的。多表查询首先要在这些表中建立连接，再在连接生成的结果集基础上进行筛选。

```
SELECT [表名.]目标字段表达式[AS别名], ...  
FROM 左表名[AS别名]连接类型 右表名[AS别名]  
ON 连接条件  
[WHERE 条件表达式];
```

四、多表查询

其中，连接类型以及运算符有以下几种。

- (1) CROSS JOIN: 交叉连接
- (2) INNER JOIN或JOIN: 内连接
- (3) LEFT JOIN或LEFT OUTER JOIN: 左外连接
- (4) RIGHT JOIN或RIGHT OUTER JOIN: 右外连接
- (5) FULL JOIN或FULL OUTER JOIN: 完全连接

为了便于理解各种类型的连接运算，现假设有两个表R和S，R和S中存储的数据如图

R		
A	B	C
1	2	3
4	5	6

S	
A	D
1	2
3	4
5	6

四、多表查询

(1) 交叉连接

交叉连接就是将要连接的两个表的所有行进行组合，也就是将第一个表的所有行分别与第二个表的每个行连接形成一个新的行。连接后生成的结果集的行数等于两个表的行数的乘积，字段个数等于两个表的字段个数的和。其语法格式如下所示：



```
SELECT 字段名列表  
FROM 表名1 CROSS JOIN 表名2;
```



四、多表查询

(1) 交叉连接

表R和表S进行交叉连接的结果集如图

R CROSS JOIN S				
A	B	C	A	D
1	2	3	1	2
1	2	3	3	4
1	2	3	5	6
4	5	6	1	2
4	5	6	3	4
4	5	6	5	6

微信公众号：灰灰考研

四、多表查询

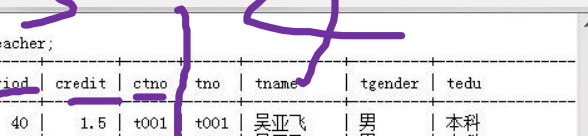
(1) 交叉连接

例：对course表和teacher表进行交叉连接，观察交叉连接后的结果集。

对应的SQL语句如下：

```
SELECT *  
FROM course CROSS JOIN teacher;
```

执行结果如图



cno	cname	cperiod	credit	ctno	tno	tname	tgender	tedu
c001	文学欣赏	40	1.5	t001	t001	吴亚飞	男	本科
c002	中国历史文化	60	2.0	t003	t001	吴亚飞	男	本科
c003	视频编辑	70	2.5	t002	t001	吴亚飞	男	本科
c004	音乐欣赏	40	1.5	t005	t001	吴亚飞	男	本科
c001	文学欣赏	40	1.5	t001	t002	李琦	男	硕士研究生
c002	中国历史文化	60	2.0	t003	t002	李琦	男	硕士研究生
c003	视频编辑	70	2.5	t002	t002	李琦	男	硕士研究生
c004	音乐欣赏	40	1.5	t005	t002	李琦	男	硕士研究生
c001	文学欣赏	40	1.5	t001	t003	王艳红	女	硕士研究生
c002	中国历史文化	60	2.0	t003	t003	王艳红	女	硕士研究生
c003	视频编辑	70	2.5	t002	t003	王艳红	女	硕士研究生
c004	音乐欣赏	40	1.5	t005	t003	王艳红	女	硕士研究生
c001	文学欣赏	40	1.5	t001	t004	马志超	男	博士研究生
c002	中国历史文化	60	2.0	t003	t004	马志超	男	博士研究生
c003	视频编辑	70	2.5	t002	t004	马志超	男	博士研究生
c004	音乐欣赏	40	1.5	t005	t004	马志超	男	博士研究生
c001	文学欣赏	40	1.5	t001	t005	万丽	女	硕士研究生
c002	中国历史文化	60	2.0	t003	t005	万丽	女	硕士研究生
c003	视频编辑	70	2.5	t002	t005	万丽	女	硕士研究生
c004	音乐欣赏	40	1.5	t005	t005	万丽	女	硕士研究生

20 rows in set (0.00 sec)

微信公众号：灰灰考研



5行 x 4行

= 20

四、多表查询

(2) 内连接

内连接是指用比较运算符设置连接条件，只返回满足连接条件的数据行，是将交叉连接生成的结果集按照连接条件进行筛选后形成的。

内连接有以下两种语法格式：

```
SELECT 字段名列表  
FROM 表名1 [INNER] JOIN 表名2  
ON 表名1. 字段名 比较运算符 表名2. 字段名;
```

或者

```
SELECT 字段名列表  
FROM 表名1, 表名2  
WHERE 表名1. 字段名 比较运算符 表名2. 字段名;
```

四、多表查询

(2) 内连接

内连接包括3种类型：等值连接、非等值连接和自然连接。

(1) 等值连接：在连接条件中使用等号(=)比较运算符来比较连接字段的值，其查询结果中包含被连接表的所有字段，包括重复字段。在等值连接中，两个表的连接条件通常采用“表1.主键字段=表2.外键字段”的形式。

(2) 非等值连接：在连接条件中使用了除等号之外的比较运算符(>、<、>=、<=、!=)来比较连接字段的值。

(3) 自然连接：与等值连接相同，都是在连接条件中使用比较运算符，但结果集中不包括重复字段。

四、多表查询

(2) 内连接

表R和表S进行等值连接、非等值连接和自然连接的结果集如图

R JOIN S (R.A=S.A)

R.A	B	C	S.A	D
1	2	3	1	2

(等值连接)

R JOIN S (R.A>S.A)

R.A	B	C	S.A	D
4	5	6	1	2
4	5	6	3	4

(非等值连接)

R JOIN S

R.A	B	C	D
1	2	3	2

(自然连接)

在上述语法格式中，如果要输出的字段是表1和表2都有的字段，则必须在输出的字段名前加上表名进行区分，用“表名.字段名”来表示。如果表名太长，可以给表名定义一个简短的别名，这样在SELECT语句的输出字段名和连接条件中，用到表名的地方都可以用别名来代替。

为连接表

四、多表查询

(2) 内连接

例：查询“学生选课”数据库，输出考试成绩不及格学生的学号、姓名、课程号、成绩。

提示：需要输出四个字段作为查询结果，在“学生选课”数据库中，没有一个表包含这四个字段，因此需要多表连接查询。多表连接查询首先确定需要哪几个表进行连接查询。进行连接查询的表要能够包含输出的所有字段，并且保证用到表的数量最少。进行连接的表之间要有含义相同的字段。

对应的SQL语句如下：

```
SELECT s.sno, sname, cno, score
FROM studentinfo AS s JOIN elective AS e ON s.sno=e.sno
WHERE score<60;
```

或者：

```
SELECT s.sno, sname, cno, score
FROM studentinfo AS s, elective AS e
WHERE s.sno=e.sno AND score<60;
```

四、多表查询

(2) 内连接

例：查询“学生选课”数据库，输出考试成绩不及格学生的学号、姓名、课程名、成绩。

对应的SQL语句如下：

```
SELECT s.sno, sname, cname, score
FROM studentinfo AS s JOIN elective AS e ON s.sno=e.sno
JOIN course AS c ON c.cno=e.cno
WHERE score<60;
```

或者：

```
SELECT s.sno, sname, cname, score
FROM studentinfo AS s, elective AS e, course AS c
WHERE s.sno=e.sno AND e.cno=c.cno AND score<60;
```

四、多表查询

(3) 外连接

外连接与内连接不同，有主从表之分。使用外连接时，以主表中每行数据去匹配从表中的数据行，如果符合连接条件则返回到结果集中；如果没有找到匹配的数据行，则在结果集中仍然保留主表的数据行，相对应的从表中的字段则被填上 NULL 值。

外连接的语法格式如下所示

```
SELECT 字段名列表  
FROM 表名1 LEFT|RIGHT JOIN 表名2  
ON 表名1. 字段名 比较运算符 表名2. 字段名;
```

四、多表查询

(3) 外连接

外连接包括3种类型：左外连接、右外连接和全外连接。

(1) 左外连接：即左表为主表，连接关键字为LEFT JOIN。将左表中的所有数据行与右表中的每行按连接条件进行匹配，结果集中包括左表中所有的数据行。左表中与右表没有相匹配记录的行，在结果集中对应的右表字段都以NULL来填充。BIT类型不允许为NULL，就以0填充。

(2) 右外连接：即右表为主表，连接关键字为RIGHT JOIN。将右表中的所有数据行与左表中的每行按连接条件进行匹配，结果集中包括右表中所有的数据行。右表中与左表没有相匹配记录的行，在结果集中对应的左表字段都以NULL来填充。

(3) 全外连接：连接关键字为FULL JOIN。查询结果集中包括两个连接表的所有的数据行，若左表中每一行在右表中有匹配数据，则结果集中对应的右表的字段填入相应数据，否则填充为NULL；若右表中某一行在左表中没有匹配数据，则结果集对应的左表字段填充为NULL

注意：外连接查询只适用于两个表。

四、多表查询

(3) 外连接

表R和表S进行外连接的结果集如图

R LEFT JOIN S (R.A=S.A)

<u>R.A</u>	B	C	<u>S.A</u>	D
1	2	3	1	2
4	5	6	NULL	NULL

R RIGHT JOIN S (R.A=S.A)

<u>R.A</u>	B	C	<u>S.A</u>	D
1	2	3	1	2
NULL	NULL	NULL	3	4
NULL	NULL	NULL	5	6

R FULL JOIN S (R.A=S.A)

<u>R.A</u>	B	C	<u>S.A</u>	D
1	2	3	1	2
4	5	6	NULL	NULL
NULL	NULL	NULL	3	4
NULL	NULL	NULL	5	6

四、多表查询

(3) 外连接

例：查询“学生选课”数据库，输出所有教师教授的课程信息，没有教授课程的教师也要列出。

对应的SQL语句如下：

```
SELECT *  
FROM teacher AS t LEFT JOIN course AS c ON t.tno=c.ctno;
```

执行结果如图

```
mysql> SELECT *  
-> FROM teacher AS t LEFT JOIN course AS c  
-> ON t.tno=c.ctno;
```

tno	tname	tgender	tedu	tpro	cno	cname	cperiod	credit	ctno
t001	吴亚飞	男	本科	讲师	c001	文学欣赏	40	1.5	t001
t003	王艳红	女	硕士研究生	讲师	c002	中国历史文化	60	2.0	t003
t002	李琦	男	硕士研究生	副教授	c003	视频编辑	70	2.5	t002
t005	万丽	女	硕士研究生	助理讲师	c004	音乐欣赏	40	1.5	t005
t004	马志超	男	博士研究生	教授	NULL	NULL	NULL	NULL	NULL

5 rows in set (0.00 sec)

四、多表查询

(4) 自连接

自连接就是一个表的两个副本之间的内连接，即同一个表名在FROM子句中出现两次，故为了区别，必须对表指定不同的别名，字段名前也要加上表的别名进行限定。

例：查询和学号为11101002的学生在同一个班级的学生的学号和姓名。

对应的SQL语句如下：

```
SELECT s2.sno, s2.sname
FROM studentinfo AS s1 JOIN studentinfo AS s2
ON s1.sclass=s2.sclass
WHERE s1.sno='11101001' AND s2.sno!= '11101001';
```

执行结果如图

```
mysql> SELECT s2.sno, s2.sname
-> FROM studentinfo AS s1 JOIN studentinfo AS s2
-> ON s1.sclass=s2.sclass
-> WHERE s1.sno='11101001' AND s2.sno!= '11101001';
```

sno	sname
11101002	王亚旭
11101003	高磊

2 rows in set (0.00 sec)

微信公众号：灰灰考研

五、子查询

子查询是将一个SELECT语句嵌套在另一个SELECT语句的WHERE子句中的查询。包含子查询的SELECT语句称为父查询或外部查询。子查询可以多层嵌套，执行时由内向外，即每一个子查询在其上一级父查询之前被处理，其查询结果回送给父查询。

子查询也可以嵌套在INSERT、UPDATE或DELETE语句中。使用子查询时，应注意以下几点。

(1) 使用圆括号将子查询的SELECT语句括起来。

(2) 当子查询的返回值为单个值时，子查询可以应用到任何表达式中。子查询有几种形式，分别是比较子查询、IN子查询、批量比较子查询和EXISTS子查询。

五、子查询

(1) 比较子查询

比较子查询是指在父查询与子查询之间用比较运算符进行连接的查询。在这种类型的子查询中，子查询返回的值最多只能有一个。

例：查询“学生选课”数据库，输出选修了“音乐欣赏”这门课的所有学生的学号和成绩。

对应的SQL语句如下：

```
SELECT sno, score AS音乐欣赏的成绩
FROM elective
WHERE cno=(SELECT cno FROM course WHERE cname='音乐欣赏');
```

执行结果如图

```
mysql> SELECT sno, score AS 音乐欣赏的成绩
-> FROM elective
-> WHERE cno=(SELECT cno FROM course WHERE cname='音乐欣赏');
```

sno	音乐欣赏的成绩
10101001	51
10102002	75
10102003	NULL
11101001	62
11101003	82

5 rows in set (0.00 sec)

微信公众号：灰灰考研

五、子查询

(1) 比较子查询

例：查询“学生选课”数据库，输出年龄最大的学生的姓名。
对应的SQL语句如下：

```
SELECT sname AS年龄最大的学生  
FROM studentinfo  
WHERE sbirth=(SELECT MIN(sbirth) FROM studentinfo);
```

执行结果如图

Value

```
mysql> SELECT sname AS 年龄最大的学生  
-> FROM studentinfo  
-> WHERE sbirth=(SELECT MIN(sbirth) FROM studentinfo);  
+-----+  
| 年龄最大的学生 |  
+-----+  
| 王斌           |  
+-----+  
1 row in set (0.01 sec)
```

五、子查询

(1) 比较子查询

例：查询“学生选课”数据库，输出“音乐欣赏”这门课不及格的学生的姓名。
对应的SQL语句如下：

```
SELECT sname AS音乐欣赏不及格的学生
FROM studentinfo
WHERE sno=(SELECT sno FROM elective
WHERE score<60 AND cno=(SELECT cno FROM course WHERE cname='音乐
欣赏'));
```

执行结果如图

```
mysql> SELECT sname AS 音乐欣赏不及格的学生
-> FROM studentinfo
-> WHERE sno=(SELECT sno FROM elective
-> WHERE score<60 AND cno=(SELECT cno FROM course WHERE cname='音乐欣赏'));
+-----+
| 音乐欣赏不及格的学生 |
+-----+
| 张永峰                |
+-----+
1 row in set (0.00 sec)
```

五、子查询

(2) IN子查询

IN子查询是指父查询与子查询之间用IN或NOT IN进行连接并判断某个字段的值是否在子查询查找到的集合中。

例：查询“学生选课”数据库，输出考试不及格的学生的姓名。
对应的SQL语句如下：

```
SELECT sname AS考试不及格的学生
FROM studentinfo
WHERE sno IN (SELECT sno FROM elective WHERE score<60);
```

执行结果如图

```
mysql> SELECT sname AS 考试不及格的学生
-> FROM studentinfo
-> WHERE sno IN (SELECT sno FROM elective WHERE score<60);
```

考试不及格的学生
张永峰
王斌
刘淑芳

3 rows in set (0.00 sec)

微信公众号：灰灰考研

五、子查询

(3) 批量比较子查询

批量比较子查询是指子查询的结果不止一个，父查询和子查询之间需要用比较运算符进行连接。这时候，就需要在子查询前面加上谓词ALL或ANY。。

1. 使用ANY谓词在子查询前面使用ANY谓词时，会使用指定的比较运算符将一个表达式的值或字段的值与每一个子查询返回值进行比较，只要有一次比较的结果为TRUE，则整个表达式的值为TRUE，否则为FALSE。

例：查询“学生选课”数据库，输出需要补考的学生姓名

```
SELECT sname AS补考学生
FROM studentinfo
WHERE sno = ANY(SELECT sno FROM elective WHERE score<60);
```

```
mysql> SELECT sname AS 补考学生
-> FROM studentinfo
-> WHERE sno = ANY(SELECT sno FROM elective WHERE score<60);
```

补考学生
张永峰
王斌
刘淑芳

3 rows in set (0.00 sec)

五、子查询

(3) 批量比较子查询

2. 使用ALL谓词在子查询前面使用ALL谓词时，会使用指定的比较运算符将一个表达式的值或字段的值与每一个子查询返回值进行比较，只有当所有比较的结果都为TRUE时，整个表达式的值才为TRUE，否则为FALSE。

例：查询“学生选课”数据库，输出不需要补考的学生的姓名。

```
SELECT sname AS不需补考的学生  
FROM studentinfo  
WHERE sno != ALL(SELECT sno FROM elective WHERE score<60);
```

执行结果如图

```
mysql> SELECT sname AS 不需补考的学生  
-> FROM studentinfo  
-> WHERE sno != ALL(SELECT sno FROM elective WHERE score<60);
```

不需补考的学生
何小丽
张宇
包玉明
孙平平
翁静静
王亚旭
高磊

```
7 rows in set (0.00 sec)
```

微信公众号：灰灰考研

五、子查询

(4) EXISTS子查询

EXISTS子查询是指在子查询前面加上EXISTS运算符或NOT EXISTS运算符，构成EXISTS表达式。如果子查询查找到了满足条件的数据行，那么EXISTS表达式的返回值为TRUE，否则为FALSE。

例：查看“学生选课”数据库的teacher表,若不存在具有教授职称的教师，则显示所有教师的姓名和职称。

```
SELECT tname, tpro  
FROM teacher  
WHERE NOT EXISTS (SELECT * FROM teacher WHERE tpro='教授');
```

执行结果如图

mysql> SELECT tname, tpro
-> FROM teacher
-> WHERE NOT EXISTS (SELECT * FROM teacher WHERE tpro='教授');
Empty set (0.00 sec)

五、子查询

(4) EXISTS子查询

例：查询“学生选课”数据库的elective表，如果有需要补考的，就显示所有学生的成绩信息。如果没有需要补考的，就不输出任何信息。

对应的SQL语句如下：

```
SELECT *  
FROM elective  
WHERE EXISTS (SELECT * FROM elective WHERE score<60);
```

执行结果如图

```
mysql> SELECT *  
-> FROM elective  
-> WHERE EXISTS (SELECT * FROM elective WHERE score<60);
```

sno	cno	score
10101001	c001	73
10101001	c003	81
10101001	c004	51
10101002	c001	78
10101003	c003	69
10102001	c001	50
10102002	c002	68
10102002	c003	95
10102002	c004	75
10102003	c001	85
10102003	c002	78
10102003	c003	67
10102003	c004	NULL
11101001	c001	49
11101001	c002	67
11101001	c004	62
11101002	c001	67
11101002	c002	NULL
11101003	c002	88
11101003	c003	90
11101003	c004	82

21 rows in set (0.00 sec)

五、子查询

(4) EXISTS子查询

(1) 对于例5-28查询和学号为11101002的学生在同一个班级的学生学号和姓名的问题，可以用子查询来实现，对应的SQL语句如下：

```
SELECT sno, sname  
FROM studentinfo  
WHERE sno != '11101002' AND sclass = (SELECT sclass FROM studentinfo  
WHERE sno = '11101002');
```

(2) 对于例5-32查询“学生选课”数据库输出需要补考的学生姓名的问题，也可以用连接查询来实现，对应的SQL语句如下：

```
SELECT sname  
FROM studentinfo AS s JOIN elective AS e  
ON s.sno = e.sno  
WHERE score < 60;
```

五、子查询

(4) EXISTS子查询

至于什么时候使用连接查询，什么时候使用子查询，可以参考以下原则。

- (1) 如果查询语句要输出的字段来自多个表时，用连接查询。
- (2) 如果查询语句要输出的字段来自一个表，但其WHERE子句涉及另一个表时，常用子查询。
- (3) 如果查询语句要输出的字段和WHERE子句都只涉及一个表，但是WHERE子句的查询条件涉及应用集合函数进行数值比较时，一般用子查询。

五、子查询

(5) 在INSERT、UPDATE、DELETE语句中使用子查询

1. 在INSERT语句中使用子查询使用INSERT...SELECT语句可以将SELECT语句的查询结果添加到表中，一次可以添加多行。语法格式如下：

```
INSERT表1[(字段名列表1)]  
SELECT字段名列表2 FROM表2 [WHERE条件表达式]
```

注意使用本语句时，表1已经存在，且“字段名列表1”中字段的个数、字段的顺序、字段的数据类型必须和“字段名列表2”中对应的字段信息一样或兼容。

五、子查询

(5) 在INSERT、UPDATE、DELETE语句中使用子查询

例：建立一个电子商务专业学生的信息表studs，表里有学号、姓名、所在班级等字段，把“学生选课”数据库中的studentinfo表中查询到的电子商务专业的学生的相关信息添加到本表中。

(1) 建立studs表。对应的SQL语句如下：

```
CREATE TABLE studs  
(sno CHAR(8),  
sname VARCHAR(10),  
sclass VARCHAR(20));
```

结果如图

```
mysql> CREATE TABLE studs  
-> (sno CHAR(8),  
-> sname VARCHAR(10),  
-> sclass VARCHAR(20));  
Query OK, 0 rows affected (0.08 sec)
```

微信公众号：灰灰考研

五、子查询

(5) 在INSERT、UPDATE、DELETE语句中使用子查询

(2) 将studentinfo表中电子商务专业的学生信息插入studs表中。对应的SQL语句如下：

```
INSERT INTO studs(sno, sname, sclass)
SELECT sno, sname, sclass FROM studentinfo WHERE sclass LIKE '电子商务%';
```

结果如图

```
mysql> INSERT INTO studs(sno, sname, sclass)
-> SELECT sno, sname, sclass FROM studentinfo WHERE sclass LIKE '电子商务%';
Query OK, 6 rows affected (0.03 sec)
Records: 6  Duplicates: 0  Warnings: 0
```

五、子查询

(5) 在INSERT、UPDATE、DELETE语句中使用子查询

2. 在UPDATE语句中使用子查询

使用UPDATE语句时，可以在WHERE子句中使用子查询。

例：修改“学生选课”数据库的course表，把职称为“副教授”的老师教授课程的学时减少6个。对应的SQL语句如下：

```
UPDATE course SET cperiod=cperiod-6  
WHERE ctno IN (SELECT tno FROM teacher WHERE tpro='副教授');
```

结果如图

```
mysql> UPDATE course SET cperiod=cperiod-6  
-> WHERE ctno IN (SELECT tno FROM teacher WHERE tpro='副教授');  
Query OK, 1 row affected (0.03 sec)  
Rows matched: 1  Changed: 1  Warnings: 0
```

微信公众号：灰灰考研

五、子查询

(5) 在INSERT、UPDATE、DELETE语句中使用子查询

3. 在DELETE语句中使用子查询使用DELETE语句时，可以在WHERE子句中使用子查询。

例：将elective表中王斌的选课信息删除。对应的SQL语句如下：

```
DELETE FROM elective  
WHERE sno=(SELECT sno FROM studentinfo WHERE sname='王斌');
```

执行结果如图

```
mysql> DELETE FROM elective  
-> WHERE sno=(SELECT sno FROM studentinfo WHERE sname='王斌');  
Query OK, 1 row affected (0.03 sec)
```

六、合并结果集

合并结果集是指对多个SELECT语句查询的结果集进行合并操作，组合成一个结果集。合并结果集使用的运算符是UNION。使用UNION时，需要注意以下几点。

- (1) 所有SELECT语句中的字段个数必须相同。
- (2) 所有SELECT语句中对应的字段的数据类型必须相同或兼容。
- (3) 合并后的结果集中的字段名是第一个SELECT语句中各字段的字段名。如果要为返回的字段指定别名，则必须在第一个SELECT语句中指定。
- (4) 使用UNION运算符合并结果集时，每一个SELECT语句本身不能包含ORDERBY子句，只能在最后使用一个ORDER BY子句对整个结果集进行排序，且在该ORDER BY子句中必须使用第一个SELECT语句中的字段名。

六、合并结果集

例：对“学生选课”数据库进行查询，输出所有学生和教师的编号和姓名。对应的SQL语句如下：

```
SELECT sno AS编号, sname AS姓名FROM studentinfo
UNION
SELECT tno AS编号, tname AS姓名FROM teacher;
```

执行结果如图

```
mysql> SELECT sno AS 编号, sname AS 姓名 FROM studentinfo
-> UNION
-> SELECT tno AS 编号, tname AS 姓名 FROM teacher;
```

编号	姓名
10101001	张永峰
10101002	何小丽
10101003	张宇
10102001	王斌
10102002	包玉明
10102003	孙平平
10102004	翁静静
11101001	刘淑芳
11101002	王亚旭
11101003	高磊
t001	吴亚飞
t002	李琦
t003	王艳红
t004	马志超
t005	万丽

15 rows in set (0.00 sec)

微信公众号：灰灰考研

本章小结

- (1) 查询是数据库最常用的操作，MySQL使用SELECT语句进行数据查询。
- (2) SELECT语句最基本的格式包括SELECT和FROM两部分，格式如下：

```
SELECT 要查询的内容  
FROM 表名;
```

其中，“要查询的内容”指出查询结果要输出的字段名列表，“表名”指出要从哪张数据表中进行查询。

- (3) SELECT语句中可以包含WHERE子句，用于指明查询条件。
- (4) SELECT语句中可以包含ORDER BY子句，用于对查询结果进行排序。
- (5) SELECT语句中可以包含LIMIT子句，用于对查询结果的输出数量进行限制。

本章小结

(6) SELECT语句中可以包含集合函数，用于对查询结果进行统计。SELECT语句还可以包含GROUP BY子句，用于对查询结果进行分组。GROUP BY子句常和集合函数配合使用。如果SELECT语句中没有包含GROUP BY子句，则集合函数对整个查询结果进行统计。如果SELECT语句中包含GROUP BY子句，则查询结果按分组进行统计。GROUP BY子句后可加HAVING子句，用于对分组设置筛选条件。

(7) 连接查询和子查询是实现多表查询的常用方式。

1. 数据查询语句SELECT由多个子句构成，（ ）子句能够将查询结果按照指定字段的值进行分组。

- A. ORDER BY B. LIMIT C. GROUP BY D. DISTINCT

2. WHERE子句用于指定（ ）。

- A. 查询结果的分组条件
B. 查询结果的统计方式
C. 查询结果的排序条件
D. 查询结果的搜索条件

3. 要在“网上书店”数据库的“图书”表中查找图书名称包含“中国”两字的图书信息，可使用（ ）。

- A. SELECT*FROM图书WHERE图书名称LIKE '中国%'
B. SELECT*FROM图书WHERE图书名称LIKE '%中国%'
C. SELECT*FROM图书WHERE图书名称LIKE '%中国'
D. SELECT*FROM图书WHERE图书名称LIKE '_中国%'

4. 在子查询语句中，下面哪一个子句用于将查询结果存储在另一张表中？（ ）

- A. GROUP BY子句 B. INSERT子句 C. WHERE子句 D. DISTINCT子句

5. 集合函数（ ）可对指定字段求平均值。A. SUM B. AVG C. MIN D. MAX

6. 对于“网上书店”数据库，以下SELECT语句的含义是（ ）。

SELECT 会员昵称 FROM会员

WHERE 会员编号 NOT IN (SELECT会员编号FROM订购)

- A. 查询输出没有订购图书的会员昵称 B. 查询输出订购图书的会员昵称
C. 查询输出所有会员昵称 D. 查询输出没有编号的会员昵称

7. 子查询的结果不止一个值时, 可以使用的运算符是 ()。

- ~~A. IN~~ B. LIKE ~~C. =~~ ~~D. >~~

8. EXISTS子查询的返回值是 ()。 A. 数值类型 B. 字符串类型 C. 日期和时间类型 D. 逻辑类型

9. 执行以下SQL语句:

SELECT 学号, 姓名 FROM 学生

LIMIT 2, 2;

查询结果返回了哪几行数据? ()

- A. 返回了两行数据, 分别是第1行和第2行数据
B. 返回了两行数据, 分别是第2行和第3行数据
C. 返回了两行数据, 分别是第3行和第4行数据
D. 返回了两行数据, 分别是第4行和第5行数据

10. 输出“学生选课”数据库中学生的成绩, 在输出时把每个学生每门课程成绩都提高10%, 使用的SQL语句是 ()。

- ~~A. SELECT sno, cno, score*10 FROM elective~~
~~B. SELECT sno, cno, score+10 FROM elective~~
~~C. SELECT sno, cno, score*0.1 FROM elective~~
D. SELECT sno, cno, score*1.1 FROM elective



THANKS

感谢观看

微信公众号：灰灰考研

