---

typora-copy-images-to: images

1.1 今日目标

1. 理解查询五子句的顺序关系；
2. 掌握两张表的联合查询方法；
3. 理解连接查询的原理；
4. 掌握子查询的使用方式；
5. 掌握预处理的实现步骤；
6. 理解事务的基本工作原理；
7. 掌握事务的四个特点；
8. 理解视图的概念和作用；

1.2 多表查询

1.2.1 内连接

规则：返回两个表的公共记录

语法：

```
-- 语法一
select * from 表1 inner join 表2 on 表1.公共字段=表2.公共字段
-- 语法二
select * from 表1, 表2 where 表1.公共字段=表2.公共字段
```

例题

```
-- inner join
mysql> select * from stuinfo inner join stumarks on
stuinfo.stuno=stumarks.stuno;
```

```
+--------+----------+--------+--------+---------+-----------+---------+----
----+-------------+---------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress | examNo  |
stuNo  | writtenExam | labExam |
+--------+----------+--------+--------+---------+-----------+---------+----
----+-------------+---------+
| s25303 | 李斯文         | 女      |     22 |       2 | 北京           |
s271811 | s25303 |          80 |      58 |
| s25302 | 李文才         | 男       |     31 |       3 | 上海           |
s271813 | s25302 |          50 |      90 |
| s25304 | 欧阳俊雄         | 男       |     28 |       4 | 天津           |
s271815 | s25304 |          65 |      50 |
| s25301 | 张秋丽          | 男       |     18 |       1 | 北京           |
s271816 | s25301 |          77 |      82 |
| s25318 | 争青小子         | 男       |     26 |       6 | 天津           |
s271819 | s25318 |          56 |      48 |
+--------+----------+--------+--------+---------+-----------+---------+----
----+-------------+---------+
5 rows in set (0.00 sec)

-- 相同的字段只显示一次
mysql> select s.stuno,stuname,stusex,writtenexam,labexam from stuinfo s
inner join stumarks m on s.stuno=m.stuno;
+--------+----------+--------+-------------+---------+
| stuno  | stuname  | stusex | writtenexam | labexam |
+--------+----------+--------+-------------+---------+
| s25303 | 李斯文         | 女      |          80 |      58 |
| s25302 | 李文才         | 男       |          50 |      90 |
| s25304 | 欧阳俊雄         | 男       |          65 |      50 |
| s25301 | 张秋丽          | 男       |          77 |      82 |
| s25318 | 争青小子         | 男       |          56 |      48 |
+--------+----------+--------+-------------+---------+
5 rows in set (0.00 sec)

-- 使用where
mysql> select * from stuinfo,stumarks where stuinfo.stuno=stumarks.stuno;
+--------+----------+--------+--------+---------+-----------+---------+----
----+-------------+---------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress | examNo  |
stuNo  | writtenExam | labExam |
+--------+----------+--------+--------+---------+-----------+---------+----
----+-------------+---------+
| s25303 | 李斯文         | 女      |     22 |       2 | 北京           |
s271811 | s25303 |          80 |      58 |
| s25302 | 李文才         | 男       |     31 |       3 | 上海           |
s271813 | s25302 |          50 |      90 |
| s25304 | 欧阳俊雄         | 男       |     28 |       4 | 天津           |
s271815 | s25304 |          65 |      50 |
| s25301 | 张秋丽          | 男       |     18 |       1 | 北京           |
s271816 | s25301 |          77 |      82 |
| s25318 | 争青小子         | 男       |     26 |       6 | 天津           |
s271819 | s25318 |          56 |      48 |
+--------+----------+--------+--------+---------+-----------+---------+----
----+-------------+---------+
5 rows in set (0.00 sec)
```

多学一招：

```
-- 1、内连接中inner可以省略
select * from 表1  join 表2 on 表1.公共字段=表2.公共字段

mysql> select * from stuinfo join stumarks on stuinfo.stuno=stumarks.stuno;
+--------+----------+--------+--------+---------+-----------+---------+----
----+-------------+---------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress | examNo  |
stuNo  | writtenExam | labExam |
+--------+----------+--------+--------+---------+-----------+---------+----
----+-------------+---------+
| s25303 | 李斯文         | 女      |     22 |       2 | 北京           |
s271811 | s25303 |          80 |      58 |
| s25302 | 李文才         | 男       |     31 |       3 | 上海           |
s271813 | s25302 |          50 |      90 |
| s25304 | 欧阳俊雄         | 男       |     28 |       4 | 天津           |
s271815 | s25304 |          65 |      50 |
| s25301 | 张秋丽          | 男       |     18 |       1 | 北京           |
s271816 | s25301 |          77 |      82 |
| s25318 | 争青小子         | 男       |     26 |       6 | 天津           |
s271819 | s25318 |          56 |      48 |
+--------+----------+--------+--------+---------+-----------+---------+----
----+-------------+---------+
5 rows in set (0.00 sec)

-- 如何实现三表查询
select * from 表1 inner join 表2 on 表1.公共字段=表2.公共字段 inner join 表3 on
表2.公共字段=表3.公共字段

-- 表连接越多，效率越低
```

思考：

```
select * from 表1 inner join 表2 on 表1.公共字段=表2.公共字段
和
select * from 表2 inner join 表1 on 表1.公共字段=表2.公共字段    一样吗?

答：一样的
```

1.2.2 左外连接

规则：以左边的表为准，右边如果没有对应的记录用null显示

语法：

```
select * from 表1 left join 表2 on 表1.公共字段=表2.公共字段
```

例题：

```
mysql> select stuname,writtenexam,labexam from stuinfo left join stumarks on
stuinfo.stuno=stumarks.stuno;
+---------+-------------+---------+
| stuname  | writtenexam | labexam |
+---------+-------------+---------+
| 张秋丽          |          77 |      82 |
| 李文才         |          50 |      90 |
| 李斯文         |          80 |      58 |
| 欧阳俊雄          |          65 |      50 |
| 诸葛丽丽          |        NULL |    NULL |
| 争青小子          |          56 |      48 |
| 梅超风         |        NULL |    NULL |
+---------+-------------+---------+
7 rows in set (0.01 sec)
```

思考：

```
select * from 表1 left join 表2 on 表1.公共字段=表2.公共字段
和
select * from 表2 left join 表1 on 表1.公共字段=表2.公共字段    一样吗?

答：不一样，第一个SQL以表1为准，第二个SQL以表2为准。
```

1.2.3 右外连接

规则：以右边的表为准，左边如果没有对应的记录用null显示

语法：

```
select * from 表1 right join 表2 on 表1.公共字段=表2.公共字段
```

例题：

```
mysql> select stuname,writtenexam,labexam from stuinfo right join stumarks
on stuinfo.stuno=stumarks.stuno;
+---------+-------------+---------+
| stuname  | writtenexam | labexam |
+---------+-------------+---------+
| 李斯文         |          80 |      58 |
| 李文才         |          50 |      90 |
| 欧阳俊雄          |          65 |      50 |
```

```
| 张秋丽          |          77 |      82 |
| 争青小子        |          56 |      48 |
| NULL     |          66 |      77 |
+----------+-------------+---------+
6 rows in set (0.00 sec)
```

思考

```
select * from 表1 left join 表2 on 表1.公共字段=表2.公共字段
和
select * from 表2 right join 表1 on 表1.公共字段=表2.公共字段    一样吗?

答: 一样
```

1.2.4 交叉连接

语法，返回笛卡尔积

```
select * from 表1 cross join 表2
```

例题

```
-- 交叉连接
mysql> select * from stuinfo cross join stumarks;

-- 交叉连接有连接表达式与内连接是一样的
mysql> select * from stuinfo cross join stumarks on
stuinfo.stuno=stumarks.stuno;
+--------+----------+--------+--------+---------+-----------+---------+----
----+-------------+---------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress | examNo  |
stuNo  | writtenExam | labExam |
+--------+----------+--------+--------+---------+-----------+---------+----
----+-------------+---------+
| s25303 | 李斯文        | 女      |     22 |       2 | 北京           |
s271811 | s25303 |          80 |      58 |
| s25302 | 李文才        | 男       |     31 |       3 | 上海           |
s271813 | s25302 |          50 |      90 |
| s25304 | 欧阳俊雄        | 男       |     28 |       4 | 天津           |
s271815 | s25304 |          65 |      50 |
| s25301 | 张秋丽         | 男       |     18 |       1 | 北京           |
s271816 | s25301 |          77 |      82 |
| s25318 | 争青小子        | 男       |     26 |       6 | 天津           |
s271819 | s25318 |          56 |      48 |
+--------+----------+--------+--------+---------+-----------+---------+----
```

```
----+------------+---------+
5 rows in set (0.00 sec)
```

小结

1、交叉连接如果没有连接条件返回笛卡尔积

2、如果有连接条件和内连接是一样的。

1.2.5 自然连接

自动判断条件连接，判断的条件是依据同名字段

1、自然内连接（natural join）

```
mysql> select * from stuinfo natural join stumarks;
+--------+----------+--------+--------+---------+------------+---------+----
---------+---------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress | examNo  |
writtenExam | labExam |
+--------+----------+--------+--------+---------+------------+---------+----
---------+---------+
| s25303 | 李斯文         | 女      |     22 |       2 | 北京           |
s271811 |          80 |      58 |
| s25302 | 李文才         | 男       |     31 |       3 | 上海           |
s271813 |          50 |      90 |
| s25304 | 欧阳俊雄         | 男       |     28 |       4 | 天津           |
s271815 |          65 |      50 |
| s25301 | 张秋丽          | 男       |     18 |       1 | 北京           |
s271816 |          77 |      82 |
| s25318 | 争青小子         | 男       |     26 |       6 | 天津           |
s271819 |          56 |      48 |
+--------+----------+--------+--------+---------+------------+---------+----
---------+---------+
5 rows in set (0.00 sec)
```

2、自然左外连接（natural left join）

```
mysql> select * from stuinfo natural left join stumarks;
+--------+----------+--------+--------+---------+------------+---------+----
---------+---------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress | examNo  |
writtenExam | labExam |
+--------+----------+--------+--------+---------+------------+---------+----
---------+---------+
| s25301 | 张秋丽          | 男       |     18 |       1 | 北京           |
s271816 |          77 |      82 |
| s25302 | 李文才         | 男       |     31 |       3 | 上海           |
```

```
s271813 |          50 |      90 |
| s25303 | 李斯文         | 女       |     22 |       2 | 北京            |
s271811 |          80 |      58 |
| s25304 | 欧阳俊雄         | 男       |     28 |       4 | 天津            |
s271815 |          65 |      50 |
| s25305 | 诸葛丽丽          | 女       |     23 |       7 | 河南            |
NULL    |        NULL |    NULL |
| s25318 | 争青小子         | 男       |     26 |       6 | 天津            |
s271819 |          56 |      48 |
| s25319 | 梅超风          | 女       |     23 |       5 | 河北            | NULL
|        NULL |    NULL |
+--------+----------+--------+--------+---------+-----------+---------+----
---------+---------+
7 rows in set (0.00 sec)
```

3、自然右外连接（natural right join）

```
mysql> select * from stuinfo natural right join stumarks;
+--------+---------+-------------+---------+----------+--------+--------+---
------+------------+
| stuNo  | examNo  | writtenExam | labExam | stuName  | stuSex | stuAge |
stuSeat | stuAddress |
+--------+---------+-------------+---------+----------+--------+--------+---
------+------------+
| s25303 | s271811 |          80 |      58 | 李斯文         | 女       |     22
|       2 | 北京           |
| s25302 | s271813 |          50 |      90 | 李文才         | 男       |
31 |       3 | 上海          |
| s25304 | s271815 |          65 |      50 | 欧阳俊雄         | 男       |
28 |       4 | 天津           |
| s25301 | s271816 |          77 |      82 | 张秋丽          | 男       |
18 |       1 | 北京           |
| s25318 | s271819 |          56 |      48 | 争青小子         | 男       |
26 |       6 | 天津           |
| s25320 | s271820 |          66 |      77 | NULL     | NULL   |   NULL |
NULL | NULL       |
+--------+---------+-------------+---------+----------+--------+--------+---
------+------------+
6 rows in set (0.00 sec)
```

小结：

1、表连接是通过同名字段来连接的

2、如果没有同名字段就返回笛卡尔积

3、同名的连接字段只显示一个，并且将该字段放在最前面

1.2.6 using

using用来指定连接字段

```
mysql> select * from stuinfo inner join stumarks using(stuno);
+--------+----------+--------+--------+---------+-----------+---------+----
---------+---------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress | examNo  |
writtenExam | labExam |
+--------+----------+--------+--------+---------+-----------+---------+----
---------+---------+
| s25303 | 李斯文         | 女      |     22 |       2 | 北京           |
s271811 |          80 |      58 |
| s25302 | 李文才         | 男       |     31 |       3 | 上海           |
s271813 |          50 |      90 |
| s25304 | 欧阳俊雄         | 男       |     28 |       4 | 天津           |
s271815 |          65 |      50 |
| s25301 | 张秋丽          | 男       |     18 |       1 | 北京           |
s271816 |          77 |      82 |
| s25318 | 争青小子         | 男       |     26 |       6 | 天津           |
s271819 |          56 |      48 |
+--------+----------+--------+--------+---------+-----------+---------+----
---------+---------+
5 rows in set (0.00 sec)
```

using的结果也会对公共字段进行优化，优化的规则和自然连接是一样的；

1.2.7 练习

1、显示地区及每个地区参加笔试的人数，并按人数降序排列

```
-- 第一步：  显示地区及每个地区参加笔试的人数
mysql> select stuaddress,count(writtenexam) from stuinfo left join stumarks
using(stuno) group by stuaddress;
+-----------+--------------------+
| stuaddress | count(writtenexam) |
+-----------+--------------------+
| 上海          |                  1 |
| 北京           |                  2 |
| 天津           |                  2 |
| 河北          |                  0 |
| 河南           |                  0 |
+-----------+--------------------+
5 rows in set (0.00 sec)

-- 第二步：将结果降序排列
mysql> select stuaddress,count(writtenexam) c from stuinfo left join
stumarks using(stuno) group by stuaddress order by c desc;
+-----------+---+
| stuaddress | c |
+-----------+---+
| 北京           | 2 |
| 天津           | 2 |
```

```
| 上海          | 1 |
| 河北          | 0 |
| 河南           | 0 |
+------------+---+
5 rows in set (0.00 sec)
```

2、显示有学生参加考试的地区

```
-- having筛选
mysql> select stuaddress,count(writtenexam) c from stuinfo left join
stumarks using(stuno) group by stuaddress having c>0;
+------------+---+
| stuaddress | c |
+------------+---+
| 上海          | 1 |
| 北京           | 2 |
| 天津           | 2 |
+------------+---+
3 rows in set (0.00 sec)

-- 表连接实现
-- 第一步：右连接获取有成绩的地区
mysql> select stuaddress from stuinfo right join stumarks using(stuno);
+------------+
| stuaddress |
+------------+
| 北京           |
| 上海          |
| 天津           |
| 北京           |
| 天津           |
| NULL       |
+------------+
6 rows in set (0.00 sec)
-- 第二步：去重复
mysql> select distinct stuaddress from stuinfo right join stumarks
using(stuno);
+------------+
| stuaddress |
+------------+
| 北京           |
| 上海          |
| 天津           |
| NULL       |
+------------+
4 rows in set (0.00 sec)

-- 去除null
mysql> select distinct stuaddress from stuinfo right join stumarks
using(stuno) having stuaddress is not null;
+------------+
| stuaddress |
+------------+
| 北京           |
```

```
| 上海          |
| 天津          |
+-------------+
3 rows in set (0.00 sec)
```

3、显示男生和女生的人数

```
-- 方法一：  分组查询
mysql> select stusex,count(*) from stuinfo group by stusex;
+--------+----------+
| stusex | count(*) |
+--------+----------+
| 女     |        3 |
| 男     |        4 |
+--------+----------+
2 rows in set (0.00 sec)

-- 方法二：  union
mysql> select stusex,count(*) from stuinfo where stusex='男' union select
stusex,count(*) from stuinfo where stusex='女';
+--------+----------+
| stusex | count(*) |
+--------+----------+
| 男     |        4 |
| 女     |        3 |
+--------+----------+
2 rows in set (0.00 sec)

-- 方法三：直接写条件

mysql> select sum(stusex='男') 男,sum(stusex='女') 女 from stuinfo;
+------+------+
| 男   | 女   |
+------+------+
|    4 |    3 |
+------+------+
1 row in set (0.00 sec)
```

4、显示每个地区男生、女生、总人数

```
mysql> select stuaddress,count(*) 总人数,sum(stusex='男') 男,sum(stusex='女')
女 from stuinfo group by stuaddress;
+-----------+--------+------+------+
| stuaddress | 总人数      | 男   | 女   |
+-----------+--------+------+------+
| 上海          |      1 |    1 |    0 |
| 北京          |      2 |    1 |    1 |
| 天津          |      2 |    2 |    0 |
| 河北          |      1 |    0 |    1 |
| 河南          |      1 |    0 |    1 |
```

```
+-----------+--------+------+------+
5 rows in set (0.00 sec)
```

1.3 子查询

语法：select * from 表1 where （子查询）

外面的查询称为父查询

子查询为父查询提供查询条件

1.3.1 标量子查询

特点：子查询返回的值是一个

```
-- 查找笔试成绩是80的学生
mysql> select * from stuinfo where stuno=(select stuno from stumarks where
writtenexam=80);
+--------+---------+--------+--------+---------+-----------+
| stuNo  | stuName | stuSex | stuAge | stuSeat | stuAddress |
+--------+---------+--------+--------+---------+-----------+
| s25303 | 李斯文       | 女      |     22 |       2 | 北京          |
+--------+---------+--------+--------+---------+-----------+
1 row in set (0.00 sec)


-- 查找最高分的学生
-- 方法一
mysql> select * from stuinfo where stuno=(select stuno from stumarks order
by writtenexam desc limit 1);
+--------+---------+--------+--------+---------+-----------+
| stuNo  | stuName | stuSex | stuAge | stuSeat | stuAddress |
+--------+---------+--------+--------+---------+-----------+
| s25303 | 李斯文       | 女      |     22 |       2 | 北京          |
+--------+---------+--------+--------+---------+-----------+
1 row in set (0.00 sec)

-- 方法二：
mysql> select * from stuinfo where stuno=(select stuno from stumarks where
writtenexam=(select max(writtenexam) from stumarks))
+--------+---------+--------+--------+---------+-----------+
| stuNo  | stuName | stuSex | stuAge | stuSeat | stuAddress |
+--------+---------+--------+--------+---------+-----------+
| s25303 | 李斯文       | 女      |     22 |       2 | 北京          |
+--------+---------+--------+--------+---------+-----------+
1 row in set (0.00 sec)
```

1.3.2 列子查询

特点： 子查询返回的结果是一列

如果子查询的结果返回多条记录，不能使用等于，用in或not in

```
-- 查找及格的同学
mysql> select * from stuinfo where stuno in (select stuno from stumarks
where writtenexam>=60);
+--------+----------+--------+--------+---------+------------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress |
+--------+----------+--------+--------+---------+------------+
| s25301 | 张秋丽         | 男       |     18 |       1 | 北京           |
| s25303 | 李斯文         | 女       |     22 |       2 | 北京           |
| s25304 | 欧阳俊雄         | 男       |     28 |       4 | 天津           |
+--------+----------+--------+--------+---------+------------+
3 rows in set (0.00 sec)

-- 查询不及格的同学
mysql> select * from stuinfo where stuno in (select stuno from stumarks
where writtenexam<60);
+--------+----------+--------+--------+---------+------------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress |
+--------+----------+--------+--------+---------+------------+
| s25302 | 李文才         | 男       |     31 |       3 | 上海           |
| s25318 | 争青小子         | 男       |     26 |       6 | 天津           |
+--------+----------+--------+--------+---------+------------+
2 rows in set (0.00 sec)

-- 查询需要补考的学生
mysql> select * from stuinfo where stuno not in (select stuno from stumarks
where writtenexam>=60);
+--------+----------+--------+--------+---------+------------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress |
+--------+----------+--------+--------+---------+------------+
| s25302 | 李文才         | 男       |     31 |       3 | 上海           |
| s25305 | 诸葛丽丽         | 女       |     23 |       7 | 河南           |
| s25318 | 争青小子         | 男       |     26 |       6 | 天津           |
| s25319 | 梅超风         | 女       |     23 |       5 | 河北           |
+--------+----------+--------+--------+---------+------------+
4 rows in set (0.00 sec)
```

1.3.3 行子查询

特点：子查询返回的结果是多个字段组成

```
-- 查找语文成绩最高的男生和女生

mysql> select * from stu where(stusex,ch) in (select stusex,max(ch) from stu
group by stusex);
```

```
+--------+----------+--------+--------+---------+------------+------+------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress | ch   | math |
+--------+----------+--------+--------+---------+------------+------+------+
| s25318 | 争青小子         | 男       |     26 |       6 | 天津           |
86 |   92 |
| s25321 | Tabm     | 女       |     23 |       9 | 河北           |   88 |
77 |
+--------+----------+--------+--------+---------+------------+------+------+
2 rows in set (0.00 sec)
```

1.3.4 表子查询

特点：将子查询的结果作为表

```
-- 查找语文成绩最高的男生和女生

mysql> select * from (select * from stu order by ch desc) t group by stusex;
+--------+----------+--------+--------+---------+------------+------+------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress | ch   | math |
+--------+----------+--------+--------+---------+------------+------+------+
| s25321 | Tabm     | 女       |     23 |       9 | 河北           |   88 |
77 |
| s25318 | 争青小子         | 男       |     26 |       6 | 天津           |
86 |   92 |
+--------+----------+--------+--------+---------+------------+------+------+
2 rows in set (0.00 sec)
```

注意：from后面跟的是数据源，如果将子查询当成表来看， 必须给结果集取别名。

1.3.5 exists子查询

```
-- 如果笔试成绩有人超过80人，就显示所有学生信息
mysql> select * from stuinfo where exists (select * from stumarks where
writtenexam>=80);

-- 没有超过80的学生，就显示所有学生信息
mysql> select * from stuinfo where not exists (select * from stumarks where
writtenexam>=80);
Empty set (0.00 sec)
```

作用：提高查询效率

1.4 视图

1.4.1 概述

1、视图是一张虚拟表，它表示一张表的部分数据或多张表的综合数据，其结构和数据是建立在对表的查询基础上

2、视图中并不存放数据，而是存放在视图所引用的原始表（基表）中

3、同一张原始表，根据不同用户的不同需求，可以创建不同的视图

1.4.2 作用

1、筛选表中的行

2、防止未经许可的用户访问敏感数据

3、隐藏数据表的结构

4、降低数据表的复杂程度

1.4.3 创建视图

语法：

```
-- 创建视图
create view 视图名
as
        select 语句;

-- 查询视图
select 列名 from 视图
```

例题

```
-- 创建视图
mysql> create view view1
    -> as
    -> select * from stu where ch>=60 and math>=60;
Query OK, 0 rows affected (0.00 sec)

-- 查询视图
mysql> select * from view1;
+--------+----------+--------+--------+---------+------------+------+------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress | ch   | math |
+--------+----------+--------+--------+---------+------------+------+------+
| s25302 | 李文才        | 男       |     31 |       3 | 上海         |   77 |
|   76 |
```

```
| s25318 | 争青小子         | 男       |     26 |       6 | 天津           |
86 |   92 |
| s25319 | 梅超风         | 女       |     23 |       5 | 河北           |   74 |
67 |
| s25320 | Tom       | 男       |     24 |       8 | 北京           |   65 |
67 |
| s25321 | Tabm      | 女       |     23 |       9 | 河北           |   88 |
77 |
+--------+----------+--------+--------+---------+------------+------+------+
5 rows in set (0.02 sec)

-- 视图可以使得降低SQL语句的复杂度
mysql> create view view2
    -> as
    -> select stuno,stusex,writtenexam,labexam from stuinfo natural join
stumarks;
Query OK, 0 rows affected (0.01 sec)
```

1.4.4 修改视图

语法

```
alter view 视图名
as
        select 语句
```

例题:

```
mysql> alter view view2
    -> as
    -> select stuname from stuinfo;
Query OK, 0 rows affected (0.00 sec)
```

1.4.5 删除视图

语法

```
drop view [if exists ] 视图1, 视图,...
```

例题

```
mysql> drop view view2;
```

```
Query OK, 0 rows affected (0.00 sec)
```

1.4.6 查看视图信息

```
-- 方法一;
mysql> show tables;      -- 显示所有的表和视图

-- 方法二：精确查找视图（视图信息存储在information_schema下的views表中）
mysql> select table_name from information_schema.views;
+------------+
| table_name |
+------------+
| view1      |
+------------+
1 row in set (0.05 sec)

-- 方法三：通过表的comment属性查询视图
mysql> show table status\G;              -- 查询所有表和视图的详细状态信息
mysql> show table status where comment='view'\G    -- 只查找视图信息
```

查询视图的结构

```
mysql> desc view1;
```

查询创建视图的语法

```
mysql> show create view view1\G
```

1.4.7 视图算法

场景：找出语文成绩最高的男生和女生

方法一：

```
mysql> select * from (select * from stu order by ch desc) t group by stusex;
+--------+----------+--------+--------+---------+------------+------+------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress | ch   | math |
+--------+----------+--------+--------+---------+------------+------+------+
| s25321 | Tabm     | 女     |     23 |       9 | 河北       |   88 |
77 |
| s25318 | 争青小子 | 男     |     26 |       6 | 天津       |
86 |   92 |
```

```
+--------+----------+--------+--------+---------+-----------+------+------+
2 rows in set (0.00 sec)
```

方法二：

```
mysql> create view view3
    -> as
    -> select * from stu order by ch desc;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from view3 group by stusex;
+--------+---------+--------+--------+---------+-----------+------+------+
| stuNo  | stuName | stuSex | stuAge | stuSeat | stuAddress | ch   | math |
+--------+---------+--------+--------+---------+-----------+------+------+
| s25301 | 张秋丽        | 男       |     18 |       1 | 北京           |   80
| NULL |
| s25303 | 李斯文       | 女      |     22 |       2 | 北京           |   55 |
82 |
+--------+---------+--------+--------+---------+-----------+------+------+
2 rows in set (0.00 sec)
```

结论：方法一和方法二的结果不一样，这是因为视图的算法造成的。

视图的算法有：

1、merge：合并算法（将视图语句和外层语句合并后再执行）

2、temptable:临时表算法（将视图作为一个临时表来执行）

3、undefined：未定义算法（用哪种算法有MySQL决定，这是默认算法，视图一般会选merge算法）

重新通过视图实现

```
-- 创建视图，指定算法为临时表算法
mysql> create or replace algorithm=temptable view view3
    -> as
    -> select * from stu order by ch desc;
Query OK, 0 rows affected (0.00 sec)

mysql> select * from view3 group by stusex;
+--------+----------+--------+--------+---------+-----------+------+------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress | ch   | math |
+--------+----------+--------+--------+---------+-----------+------+------+
| s25321 | Tabm     | 女      |     23 |       9 | 河北          |   88 |
77 |
| s25318 | 争青小子        | 男       |     26 |       6 | 天津           |
86 |   92 |
```

```
+--------+----------+--------+--------+---------+-----------+------+------+
2 rows in set (0.00 sec)
```

结论：和子查询结果一致。

1.5 事务

1.5.1 概述

事务(TRANSACTION)是一个整体，要么一起执行，要么一起不执行

1.5.2 事务特性

事务必须具备以下四个属性，简称ACID 属性：

原子性（Atomicity）：事务是一个完整的操作。事务的各步操作是不可分的（原子的）；要么都执行，要么都不执行

一致性（Consistency）：当事务完成时，数据必须处于一致状态

隔离性（Isolation）：对数据进行修改的所有并发事务是彼此隔离的。

永久性（Durability）：事务完成后，它对数据库的修改被永久保持。

1.5.3 事务处理

开启事务

```
start transaction  或 begin [work]
```

提交事务

```
commit
```

回滚事务

```
rollback
```

例题：

```
-- 插入测试数据
mysql> create table bank(
    -> card char(4) primary key comment '卡号',
    -> money decimal(10,2) not null
    -> )engine=innodb charset=utf8;
Query OK, 0 rows affected (0.05 sec)

mysql> insert into bank values ('1001',1000),('1002',1);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0

-- 开启事务
mysql> begin;
Query OK, 0 rows affected (0.00 sec)

mysql> delimiter //   -- 更改定界符
mysql> update bank set money=money-100 where card='1001';
    -> update bank set money=money+100 where card='1002' //
Query OK, 1 row affected (0.04 sec)
Rows matched: 1  Changed: 1  Warnings: 0

-- 回滚事务
mysql> rollback //
Query OK, 0 rows affected (0.00 sec)

mysql> select * from bank  //
+------+---------+
| card | money   |
+------+---------+
| 1001 | 1000.00 |
| 1002 |    1.00 |
+------+---------+
2 rows in set (0.00 sec)

------------------------------------------------------------------
-- 开启事务
mysql> start transaction  //
Query OK, 0 rows affected (0.00 sec)

mysql> update bank set money=money-100 where card='1001';
    -> update bank set money=money+100 where card='1002' //
Query OK, 1 row affected (0.00 sec)
Rows matched: 1  Changed: 1  Warnings: 0

-- 提交事务
mysql> commit //
Query OK, 0 rows affected (0.00 sec)

mysql> select * from bank //
+------+--------+
| card | money  |
+------+--------+
| 1001 | 900.00 |
```

```
| 1002 | 101.00 |
+------+--------+
```

设置事务的回滚点

```
-- 开启事务
mysql> begin  //
Query OK, 0 rows affected (0.00 sec)

mysql> insert into bank values ('1003',500) //
Query OK, 1 row affected (0.00 sec)

-- 记录事务的回滚点
mysql> savepoint a1  //
Query OK, 0 rows affected (0.00 sec)

mysql> insert into bank values ('1004',200) //
Query OK, 1 row affected (0.00 sec)

-- 回滚到回滚点
mysql> rollback  to a1  //
Query OK, 0 rows affected (0.00 sec)
-- 查询
mysql> select * from bank //
+------+--------+
| card | money  |
+------+--------+
| 1001 | 900.00 |
| 1002 | 101.00 |
| 1003 | 500.00 |
+------+--------+
3 rows in set (0.00 sec)
```

自动提交事务

```
每一个SQL语句都是一个独立的事务
```

小结：

1、事务是事务开启的时候开始

2、提交事务、回滚事务后事务都结束

3、只有innodb支持事务

4、一个SQL语句就是一个独立的事务，开启事务是将多个SQL语句放到一个事务中执行

1.6 索引

1.6.1 概述

优点

> 加快查询速度

缺点:

> 带索引的表在数据库中需要更多的存储空间
> 增、删、改命令需要更长的处理时间，因为它们需要对索引进行更新

1.6.2 创建索引的指导原则

适合创建索引的列

> 1、该列用于频繁搜索
> 2、该列用于对数据进行排序
> 3、在WHERE子句中出现的列，在join子句中出现的列。

请不要使用下面的列创建索引:

> 1、列中仅包含几个不同的值。
> 2、表中仅包含几行。为小型表创建索引可能不太划算，因为MySQL在索引中搜索数据所花的时间比在表中逐行搜索所花的时间更长

1.6.3 创建索引

1、主键索引：主要创建了主键就会自动的创建主键索引

2、唯一索引：创建唯一键就创建了唯一索引

```
-- 创建表的时候添加唯一索引
create table t5(
    id int primary key,
    name varchar(20),
```

```
    unique ix_name(name)          -- 添加唯一索引
);

-- 给表添加唯一索引
mysql> create table t5(
    -> name varchar(20),
    -> addr varchar(50)
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql> create unique index ix_name on t5(name);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

-- 通过更改表的方式创建唯一索引
mysql> alter table t5 add unique ix_addr (addr);
Query OK, 0 rows affected (0.00 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

普通索引

```
-- 创建表的时候添加普通索引
mysql> create table t6(
    ->         id int primary key,
    ->         name varchar(20),
    ->         index ix_name(name)
    -> );
Query OK, 0 rows affected (0.02 sec)

-- 给表添加普通索引
mysql> create table t7(
    -> name varchar(20),
    -> addr varchar(50)
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql> create index ix_name on t7(name) ;
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

-- 通过更改表的方式创建索引
mysql> alter table t7 add index ix_addr(addr);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

小结：

1、创建主键就会创建主键索引

2、创建唯一键就会创建唯一索引

3、创建唯一键的语法

```
--语法一
create unique [index] 索引名 on 表名 (字段名)
-- 方法二
alter table 表名 add uniqe [index] 索引名(字段名)
```

4、创建普通索引

```
-- 语法一
create index 索引名 on 表名 (字段名)
-- 语法二
alter table 表名 add index 索引名(字段名)
```

5、索引创建后，数据库根据查询语句自动选择索引

1.6.4 删除索引

语法：drop index 索引名 on 表名

```
mysql> drop index ix_name on t7;
Query OK, 0 rows affected (0.01 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

1.7 函数

1.7.1 数字类

```
-- 获取随机数
mysql> select rand();
+------------------+
| rand()           |
+------------------+
| 0.25443412666622 |
+------------------+
1 row in set (0.00 sec)

-- 随机排序
mysql> select * from stuinfo order by rand();

-- 随机获取一条记录
```

```
mysql> select * from stuinfo order by rand() limit 1;

-- 四舍五入，向上取整，向下取整
mysql> select round(3.1415926,3) '四舍五入',truncate(3.14159,3) '截取数
据',ceil(3.1) '向上取整',floor(3.9) '向下取整';
+----------+----------+----------+----------+
| 四舍五入  | 截取数据  | 向上取整   | 向下取整        |
+----------+----------+----------+----------+
|    3.142 |    3.141 |        4 |        3 |
+----------+----------+----------+----------+
1 row in set (0.04 sec)
注意：  截取数据直接截取，不四舍五入
```

1.7.2 字符串类

```
-- 大小写转换
mysql> select ucase('i name is tom') '转成大写',lcase('My Name IS TOM') '转成
小写';
+--------------+----------------+
| 转成大写        | 转成小写          |
+--------------+----------------+
| I NAME IS TOM | my name is tom |
+--------------+----------------+
1 row in set (0.00 sec)

-- 截取字符串
mysql> select left('abcdef',3) '从左边截取',right('abcdef',3) '从右边截
取',substring('abcdef',2,3) '字符串';
+-----------+-----------+--------+
| 从左边截取  | 从右边截取    | 字符串      |
+-----------+-----------+--------+
| abc       | def       | bcd    |
+-----------+-----------+--------+
1 row in set (0.00 sec)

-- 字符串相连
mysql> select concat('中国','北京','顺义') '地址';
+--------------+
| 地址           |
+--------------+
| 中国北京顺义          |
+--------------+
1 row in set (0.00 sec)

mysql> select concat(stuname,'-',stusex) 信息 from stuinfo;
+-------------+
| 信息          |
+-------------+
| 张秋丽-男          |
| 李文才-男         |
| 李斯文-女         |
| 欧阳俊雄-男          |
| 诸葛丽丽-女          |
| 争青小子-男          |
```

```
| 梅超风-女          |
+--------------+
7 rows in set (0.00 sec)

-- coalesce(str1,str2)  :str1有值显示str1,如果str1为空就显示str2
-- 将成绩为空的显示为缺考
mysql> select stuname,coalesce(writtenexam,'缺考'),coalesce(labexam,'缺考')
from stuinfo natural left join stumarks;
+----------+-------------------------------+-------------------------+
| stuname  | coalesce(writtenexam,'缺考')    | coalesce(labexam,'缺考')    |
+----------+-------------------------------+-------------------------+
| 张秋丽          | 77                              | 82                          |
| 李文才         | 50                              | 90                          |
| 李斯文         | 80                              | 58                          |
| 欧阳俊雄         | 65                              | 50                           |
| 诸葛丽丽          | 缺考                              | 缺考
|
| 争青小子         | 56                              | 48                           |
| 梅超风         | 缺考                              | 缺考
|
+----------+-------------------------------+-------------------------+
7 rows in set (0.02 sec)

-- length () :字节长度, char_length(): 字符长度
mysql> select length('锄禾日当午') 字节,char_length('锄禾日当午') 字符;
+------+------+
| 字节  | 字符    |
+------+------+
|   10 |    5 |
+------+------+
1 row in set (0.00 sec)
```

1.7.3 时间类

```
-- 时间戳
mysql> select unix_timestamp();
+------------------+
| unix_timestamp() |
+------------------+
|       1560330458 |
+------------------+
1 row in set (0.00 sec)

-- 格式化时间戳
mysql> select from_unixtime(unix_timestamp());
+---------------------------------+
| from_unixtime(unix_timestamp()) |
+---------------------------------+
| 2019-06-12 17:08:18             |
+---------------------------------+
1 row in set (0.05 sec)

-- 获取当前格式化时间
mysql> select now();
```

```
+---------------------+
| now()               |
+---------------------+
| 2019-06-12 17:08:50 |
+---------------------+
1 row in set (0.00 sec)

-- 获取年，月，日，小时，分钟，秒
mysql> select year(now()) 年,month(now()) 月,day(now()) 日,hour(now()) 小
时,minute(now()) 分钟,second(now())秒;
+------+------+------+------+------+------+
| 年   | 月   | 日   | 小时 | 分钟 | 秒   |
+------+------+------+------+------+------+
| 2019 |    6 |   12 |   17 |   10 |   48 |
+------+------+------+------+------+------+
1 row in set (0.00 sec)

-- 星期，本年第几天;
mysql> select dayname(now()) 星期,dayofyear(now()) 本年第几天;
+-----------+------------+
| 星期          | 本年第几天          |
+-----------+------------+
| Wednesday |        163 |
+-----------+------------+
1 row in set (0.00 sec)

-- 日期相减
mysql> select datediff(now(),'2010-08-08') 相距天数;
+----------+
| 相距天数         |
+----------+
|     3230 |
+----------+
1 row in set (0.00 sec)
```

1.7.4 加密函数

1、md5()

2、sha()

```
mysql> select md5('aa');
+----------------------------------+
| md5('aa')                        |
+----------------------------------+
| 4124bc0a9335c27f086f24ba207a4912 |
+----------------------------------+
1 row in set (0.00 sec)

mysql> select sha('aa');
+------------------------------------------+
| sha('aa')                                |
+------------------------------------------+
```

```
| e0c9035898dd52fc65c41454cec9c4d2611bfb37 |
+------------------------------------------+
1 row in set (0.00 sec)
```

1.8 预处理

每个代码的段的执行都要经历：词法分析——语法分析——编译——执行

预编译一次，可以多次执行。用来解决一条SQL语句频繁执行的问题。

```
预处理语句：prepare 预处理名字 from ‘sql语句’
执行预处理：execute 预处理名字 [using 变量]
```

例题：不带参数的预处理

```
-- 创建预处理
mysql> prepare stmt from 'select * from stuinfo';
Query OK, 0 rows affected (0.06 sec)
Statement prepared

-- 执行预处理
mysql> execute stmt;
+--------+----------+--------+--------+---------+------------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress |
+--------+----------+--------+--------+---------+------------+
| s25301 | 张秋丽         | 男       |     18 |       1 | 北京           |
| s25302 | 李文才        | 男       |     31 |       3 | 上海          |
| s25303 | 李斯文        | 女       |     22 |       2 | 北京          |
| s25304 | 欧阳俊雄         | 男       |     28 |       4 | 天津           |
| s25305 | 诸葛丽丽          | 女       |     23 |       7 | 河南           |
| s25318 | 争青小子         | 男       |     26 |       6 | 天津           |
| s25319 | 梅超风        | 女       |     23 |       5 | 河北          |
+--------+----------+--------+--------+---------+------------+
7 rows in set (0.00 sec)
```

例题：带一个参数的预处理

```
-- 创建带有位置占位符的预处理语句
mysql> prepare stmt from 'select * from stuinfo where stuno=?' ;
Query OK, 0 rows affected (0.00 sec)
Statement prepared

-- 调用预处理，并传参数
mysql> delimiter //
```

```
mysql> set @id='s25301';
    -> execute stmt using @id //
Query OK, 0 rows affected (0.00 sec)

+--------+---------+--------+--------+---------+-----------+
| stuNo  | stuName | stuSex | stuAge | stuSeat | stuAddress |
+--------+---------+--------+--------+---------+-----------+
| s25301 | 张秋丽        | 男       |     18 |       1 | 北京          |
+--------+---------+--------+--------+---------+-----------+
1 row in set (0.00 sec)
```

例题：传递多个参数

```
mysql> prepare stmt from 'select * from stuinfo where stuage>? and stusex=?'
//
Query OK, 0 rows affected (0.00 sec)
Statement prepared

mysql> set @age=20;
    -> set @sex='男';
    -> execute stmt using @age,@sex //
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

+--------+----------+--------+--------+---------+-----------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress |
+--------+----------+--------+--------+---------+-----------+
| s25302 | 李文才         | 男       |     31 |       3 | 上海          |
| s25304 | 欧阳俊雄         | 男       |     28 |       4 | 天津          |
| s25318 | 争青小子         | 男       |     26 |       6 | 天津          |
+--------+----------+--------+--------+---------+-----------+
3 rows in set (0.00 sec)
```

小结：

1、MySQL中变量以@开头

2、通过set给变量赋值

3、？是位置占位符