1.1 今日目标

1. 理解多表查询
2. 理解子查询
3. 能够创建视图
4. 能够删除视图
5. 能够查看创建视图的SQL语句
6. 能够理解事务的作用
7. 能够操作事务
8. 理解索引的作用
9. 能够创建索引
10. 能够删除索引
11. 知道常用的函数
12. 了解预处理语句的作用
13. 能够使用预处理语句
14. 了解存储过程的作用
15. 能够创建存储过程
16. 能够调用存储过程

1.2 多表查询分类

将多个表的数据横向的联合起来。

1、 内连接
2、 外连接
   a) 左外连接
   b) 右外连接
3、 交叉连接
4、 自然连接

1.2.1 内连接【inner join】

```
语法一：select 列名 from 表1 inner join 表2 on 表1.公共字段=表2.公共字段

语法二：select 列名 from 表1,表2 where 表1.公共字段=表2.公共字段
```

例题

```
方法一：
mysql> select stuname,stusex,writtenexam,labexam from stuinfo inner join stumarks
on stuinfo.stuno=stumarks.stuno;
+----------+--------+-------------+---------+
| stuname  | stusex | writtenexam | labexam |
+----------+--------+-------------+---------+
| 李斯文         | 女       |          80 |       58 |
| 李文才         | 男       |          50 |       90 |
| 欧阳俊雄         | 男       |          65 |       50 |
```

```
| 张秋丽          | 男       |          77 |       82 |
| 争青小子        | 男       |          56 |       48 |
+----------+--------+-------------+---------+
```

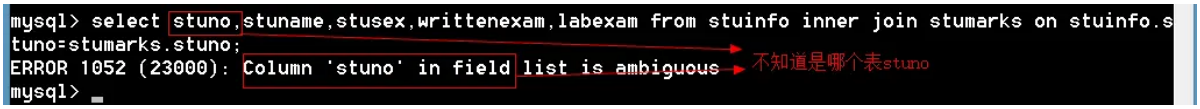
方法二：

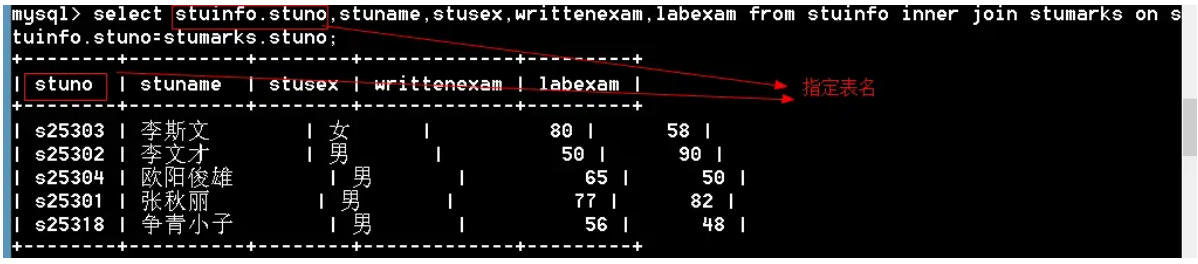
```
mysql> select stuinfo.stuno,stuname,stusex,writtenexam,labexam from
stuinfo,stumarks where stuinfo.stuno=stumarks.stuno;
+--------+----------+--------+-------------+---------+
| stuno  | stuname  | stusex | writtenexam | labexam |
+--------+----------+--------+-------------+---------+
| s25303 | 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 |
+--------+----------+--------+-------------+---------+
```

可以给表取别名

```
mysql> select i.stuno,stuname,stusex,writtenexam,labexam from stuinfo i,stumarks
s where i.stuno=s.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)
```

脚下留心：显示公共字段需要指定表名





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   结果是否一样？
```

答：一样的，因为内连接获取的是两个表的公共部分

多学一招：三个表的内连接如何实现？

```
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
```

1.2.2 左外连接【left join】

以左边的表为标准，如果右边的表没有对应的记录，用NULL填充。

```
语法：select 列名 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 |
+----------+-------------+---------+
```

```
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    是否一样？
答：不一样，左连接一左边的表为准。
```

1.2.3 右外连接【right join】

以右边的表为标准，如果左边的表没有对应的记录，用NULL填充。

```
语法：select 列名 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  是否一样？

答：一样的
```

1.2.4 交叉连接【cross join】

插入测试数据

```
mysql> create table t1(
    -> id int,
    -> name varchar(10)
    -> );
Query OK, 0 rows affected (0.06 sec)

mysql> insert into t1 values (1,'tom'),(2,'berry');
Query OK, 2 rows affected (0.00 sec)

mysql> create table t2(
    -> id int,
    -> score int);
Query OK, 0 rows affected (0.02 sec)

mysql> insert into t2 values (1,88),(2,99);
```

1、如果没有连接表达式返回的是笛卡尔积

```
mysql> select * from t1 cross join t2;   # 返回笛卡尔积
+------+-------+------+-------+
| id   | name  | id   | score |
+------+-------+------+-------+
|    1 | tom   |    1 |    88 |
|    2 | berry |    1 |    88 |
|    1 | tom   |    2 |    99 |
|    2 | berry |    2 |    99 |
+------+-------+------+-------+
```

2、如果有连接表达式等价于内连接

```
mysql> select * from t1 cross join t2 where t1.id=t2.id;
+------+-------+------+-------+
| id   | name  | id   | score |
+------+-------+------+-------+
|    1 | tom   |    1 |    88 |
|    2 | berry |    2 |    99 |
+------+-------+------+-------+
```

1.2.5 自然连接【natural】

> 自动的判断连接条件，它是过同名字段来判断的

自然连接又分为：

1. 自然内连接　　natural join
2. 自然左外连接　　natural left join
3. 自然右外连接　　natural right join

例题：

```
# 自然内连接
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)

# 自然左外连接

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 |
ULL |
+--------+----------+--------+--------+---------+------------+---------+---------
----+---------+
7 rows in set (0.00 sec)

# 自然右外连接
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)
```

自然连接结论：

1. 表连接通过同名的字段来连接的
2. 如果没有同名的字段返回笛卡尔积
3. 会对结果进行整理，整理的规则如下

a) 连接字段保留一个

b) 连接字段放在最前面

c) 左外连接左边在前，右外连接右表在前

1.2.6 using()

1. 用来指定连接字段。
2. using()也会对连接字段进行整理，整理方式和自然连接是一样的。

```
mysql> select * from stuinfo inner join stumarks using(stuno);   # using指定字段
+--------+---------+--------+--------+---------+------------+---------+---------
----+---------+
| 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)
```

1.3 子查询

语法

```
语法：select 语句 where 条件 (select … from 表)
```

1. 外面的查询称为父查询，括号中的查询称为子查询
2. 子查询为父查询提供查询条件

1.3.1 例题

1、查找笔试80分的学生

```
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           |
+--------+---------+--------+--------+---------+------------+
```

2、查找笔试最高分的学生

```
# 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)
```

脚下留心：上面的例题，子查询只能返回一个值。如果子查询返回多个值就不能用“=”了,需要用 in

1.3.2 in|not in子查询

用于子查询的返回结果多个值。

1、查找笔试成绩及格的同学

```
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)
```

2、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           |
+--------+----------+--------+--------+---------+------------+
```

3、查询没有通过的同学（不及格，缺考）

```
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 exists和not exists

1、 如果有人笔试超过80分就显示所有的学生

```
mysql> select * from stuinfo where exists (select * from stumarks where
writtenexam>=80);
+--------+----------+--------+--------+---------+------------+
| 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           |
+--------+----------+--------+--------+---------+------------+
```

2、 如果没有人超过80分就显示所有的学生

```
mysql> select * from stuinfo where not exists (select * from stumarks where
writtenexam>=80);
Empty set (0.02 sec)
```

1.3.4 子查询分类

1、标量子查询：子查询返回的结果就一个

2、列子查询：子查询返回的结果是一个列表

3、行子查询：子查询返回的结果是一行

例题：查询成绩最高的男生和女生

```
mysql> select stuname,stusex,ch from stu where (stusex,ch) in (select
stusex,max(ch) from stu group by stusex);
+----------+--------+------+
| stuname  | stusex | ch   |
+----------+--------+------+
| 争青小子         | 男       |   86 |
| Tabm     | 女       |   88 |
+----------+--------+------+
```

4、表子查询：子查询返回的结果当成一个表

例题：查询成绩最高的男生和女生

```
mysql> select stuname,stusex,ch from (select * from stu order by ch desc) as t
group by stusex;
+----------+--------+------+
| stuname  | stusex | ch   |
+----------+--------+------+
| Tabm     | 女       |   88 |
| 争青小子         | 男       |   86 |
+----------+--------+------+
```

脚下留心：from后面是一个表，如果子查询的结果当成表来看，必须将子查询的结果取别名。

1.4 视图【view】

1、 视图是一张虚拟表，它表示一张表的部分或多张表的综合的结构。

2、 视图仅仅是表结构，没有表数据。视图的结构和数据建立在表的基础上。

1.4.1 创建视图

语法

```
create [or replace] view 视图的名称
as
    select语句
```

例题：

```
mysql> create view vw_stu
    -> as
    -> select stuname,stusex,writtenexam,labexam from stuinfo inner join stumarks
using(stuno);
Query OK, 0 rows affected (0.00 sec)
```

多学一招：因为视图是一个表结构，所以创建视图后，会在数据库文件夹中多一个与视图名同名的.frm文件

1.4.2 使用视图

视图是一张虚拟表，视图的用法和表的用法一样

```
mysql> select * from vw_stu;
+----------+--------+-------------+---------+
| stuname  | stusex | writtenexam | labexam |
+----------+--------+-------------+---------+
| 李斯文         | 女       |          80 |      58 |
| 李文才         | 男       |          50 |      90 |
| 欧阳俊雄         | 男       |          65 |      50 |
| 张秋丽          | 男       |          77 |      82 |
| 争青小子         | 男       |          56 |      48 |
+----------+--------+-------------+---------+

mysql> update vw_stu set writtenexam=88 where stuname='李斯文';
Query OK, 1 row affected (0.05 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

1.4.3 查看视图的结构

语法:

```
desc 视图名
```

例题

```
mysql> desc vw_stu;
+-------------+-------------+------+-----+---------+-------+
| Field       | Type        | Null | Key | Default | Extra |
+-------------+-------------+------+-----+---------+-------+
| stuname     | varchar(10) | NO   |     | NULL    |       |
| stusex      | char(2)     | NO   |     | NULL    |       |
| writtenexam | int(11)     | YES  |     | NULL    |       |
| labexam     | int(11)     | YES  |     | NULL    |       |
+-------------+-------------+------+-----+---------+-------+
```

1.4.4 查看创建视图的语法

语法:

```
show create view 视图名
```

例题

```
mysql> show create view vw_stu\G
*************************** 1. row ***************************
                View: vw_stu
         Create View: CREATE ALGORITHM=UNDEFINED DEFINER=`root`@`localhost` SQL SECURITY DEFINER VIE
W `vw_stu` AS select `stuinfo`.`stuName` AS `stuname`,`stuinfo`.`stuSex` AS `stusex`,`stumarks`.`wri
ttenExam` AS `writtenexam`,`stumarks`.`labExam` AS `labexam` from (`stuinfo` join `stumarks` on((`st
uinfo`.`stuNo` = `stumarks`.`stuNo`)))
character_set_client: gbk
collation_connection: gbk_chinese_ci
1 row in set (0.00 sec)
```

1.4.5 显示所有视图

```
 #方法一：
mysql> show tables;
+------------------+
| Tables_in_itcast |
+------------------+
| stu              |
| stuinfo          |
| stumarks         |
| t1               |
| t2               |
| vw_stu           |

# 方法二
mysql> select table_name from information_schema.views;
+------------+
| table_name |
+------------+
| vw_stu     |
+------------+
1 row in set (0.05 sec)
+------------------+

#方法三
mysql> show table status where comment='view' \G
*************************** 1. row ***************************
           Name: vw_stu
         Engine: NULL
        Version: NULL
     Row_format: NULL
           Rows: NULL
 Avg_row_length: NULL
    Data_length: NULL
Max_data_length: NULL
   Index_length: NULL
      Data_free: NULL
 Auto_increment: NULL
    Create_time: NULL
    Update_time: NULL
     Check_time: NULL
      Collation: NULL
       Checksum: NULL
 Create_options: NULL
        Comment: VIEW
1 row in set (0.00 sec)
```

1.4.6 更改视图

语法：

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

例题：

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

1.4.7 删除视图

语法：

```
drop view [if exists] 视图1,视图2,…
```

例题

```
mysql> drop view vw_stu;
Query OK, 0 rows affected (0.00 sec)
```

1.4.8 视图的作用

1. 筛选数据，防止未经许可访问敏感数据
2. 隐藏表结构
3. 降低SQL语句的复杂度

1.4.9 视图的算法

场景：找出语文成绩最高的男生和女生

```
mysql> select * from (select * from stu order by ch desc) as 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 |
+--------+----------+--------+--------+---------+------------+------+------+
```

我们可以将子查询封装到视图中

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

可以将上面的子查询更改成视图，但是，结果和上面不一样

```
mysql> select * from vw_stu 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
|
+--------+---------+--------+--------+---------+------------+------+------+
```

原因：这是因为视图的算法造成的

1. merge：合并算法，将视图的语句和外层的语句合并后在执行。
2. temptable：临时表算法，将视图生成一个临时表，再执行外层语句
3. undefined：未定义，MySQL到底用merge还是用temptable由MySQL决定，这是一个默认的算法，一般视图都会选择merge算法，因为merge效率高。

解决：在创建视图的时候指定视图的算法

```
create algorithm=temptable view 视图名
as
    select 语句
```

指定算法创建视图

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

mysql> select * from vw_stu group by stusex;   # 结果是一致的
+--------+----------+--------+--------+---------+------------+------+------+
| 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 |
+--------+----------+--------+--------+---------+------------+------+------+
```

1.5 事务【transaction】

1. 事务是一个不可分割的执行单元
2. 事务作为一个整体要么一起执行，要么一起回滚

插入测试数据

```
mysql> create table bank(
    -> cardid char(4) primary key,
    -> money int
    -> );
Query OK, 0 rows affected (0.00 sec)

mysql> insert into bank values ('1001',1000),('1002',100);
Query OK, 2 rows affected (0.00 sec)
Records: 2  Duplicates: 0  Warnings: 0
```

1.5.1 事务操作

```
开启事务：start transaction或begin [work]
提交事务：commit
回滚事务：rollback
```

例题：

```
mysql> delimiter //            # 更改定界符

mysql> start transaction;           # 开启事务
    -> update bank set money=money-100 where cardid='1001';
    -> update bank set money=money+100 where cardid='1002'  //
Query OK, 0 rows affected (0.00 sec)

mysql> commit //   # 提交事务

mysql> rollback //  # 回滚事务
```

```
思考：事务什么时候产生？什么时候结束？
答：开启的时候产生，提交事务或回滚事务都结束

脚下留心：只有innodb和BDB才支持事务，myisam不支持事务。
```

1.5.2 设置事务的回滚点

语法：

```
设置回滚点：  savepoint 回滚点名
回滚到回滚点：  rollback to 回滚点
```

例题：

```
mysql>  start transaction;
Query OK, 0 rows affected (0.00 sec)

mysql> insert into bank values ('1003',1000);
Query OK, 1 row affected (0.00 sec)

mysql> savepoint aa;    # 设置回滚点 aa
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> insert into bank values ('1004',500);
Query OK, 1 row affected (0.00 sec)

mysql> savepoint bb;   # 设置回滚点bb
Query OK, 0 rows affected (0.00 sec)

mysql> rollback to aa;    # 回滚到aa点
Query OK, 0 rows affected (0.00 sec)

mysql> commit;    # 提交事务

mysql> select * from bank ;
+--------+-------+
| cardid | money |
+--------+-------+
| 1001   |   800 |
| 1002   |   200 |
| 1003   |  1000 |
+--------+-------+
```

1.5.3 事务的特性（ACID）

1. 原子性（Atomicity）：事务是一个整体，不可以再分，要么一起执行，要么一起不执行。
2. 一致性（Consistency）：事务完成时，数据必须处于一致的状态。
3. 隔离性（Isolation）：每个事务都是相互隔离的
4. 永久性（Durability）：事务完成后，对数据的修改是永久性的。

1.6 索引【index】

索引的优点：查询速度快

索引的缺点：

1. 增、删、改（数据操作语句）效率低了
2. 索引占用空间

1.6.1 索引的类型

1. 普通索引
2. 唯一索引（唯一键）
3. 主键索引：只要主键就自动创建主键索引，不需要手动创建。
4. 全文索引，搜索引擎使用，MySQL不支持中文的全文索引，我们通过sphinx去解决中文的全文索引。

1.6.2 创建普通索引【create index】

语法：

```
create index [索引名] on 表名 （字段名）
alter table 表名 add index [索引的名称] （列名）
```

例题：

```
# 创建索引方法一
mysql> create index ix_stuname on stuinfo(stuname);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

# 创建索引方法二
mysql> alter table stuinfo add index ix_address (stuaddress);
Query OK, 0 rows affected (0.08 sec)
Records: 0  Duplicates: 0  Warnings: 0

# 创建表的时候就添加索引
mysql> create table emp(
    -> id int,
    -> name varchar(10),
    -> index ix_name (name)   # 创建索引
    -> );
Query OK, 0 rows affected (0.00 sec)
```

1.6.3 创建唯一索引

```
语法一：create unique index 索引名 on 表名 （字段名）
语法二：alter table 表名 add unqiue [index] [索引的名称] （列名）
语法三：创建表的时候添加唯一索引，和创建唯一键是一样的。
```

例题

```
# 方法一：
mysql> create unique index UQ_stuname on stu(stuname);
Query OK, 0 rows affected (0.06 sec)
Records: 0  Duplicates: 0  Warnings: 0

# 方法二：
mysql> alter table stu add unique UQ_address (stuaddress);
Query OK, 0 rows affected (0.02 sec)
Records: 0  Duplicates: 0  Warnings: 0

# 方法三
mysql> create table stu2(
    -> id int,
    -> name varchar(20),
    -> unique UQ_name(name)
    -> );
Query OK, 0 rows affected (0.01 sec)
```

1.6.4 删除索引

语法

```
drop index 索引名 on 表名
```

例题

```
mysql> drop index ix_stuname on stuinfo;
Query OK, 0 rows affected (0.03 sec)
Records: 0  Duplicates: 0  Warnings: 0
```

1.6.5 创建索引的指导原则

1. 该列用于频繁搜索
2. 改列用于排序
3. 公共字段要创建索引
4. 如果表中的数据很少，不需要创建索引。MySQL搜索索引的时间比逐条搜索数据的时间要长。
5. 如果一个字段上的数据只有几个不同的值，改字段不适合做索引，比如性别。

1.7 函数

1.7.1 数字类

```
mysql> select rand();           # 生成随机数
+--------------------+
| rand()             |
+--------------------+
| 0.18474003969201822 |
+--------------------+
1 row in set (0.00 sec)

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

mysql> select * from stuinfo order by rand() limit 2;    # 随机抽两个学生
+--------+----------+--------+--------+---------+------------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress |
+--------+----------+--------+--------+---------+------------+
| s25305 | 诸葛丽丽          | 女       |     23 |       7 | 河南            |
| s25304 | 欧阳俊雄         | 男       |     28 |       4 | 天津            |
+--------+----------+--------+--------+---------+------------+
2 rows in set (0.00 sec)

mysql> select round(3.5);     #四舍五入
+------------+
| round(3.5) |
+------------+
|          4 |
+------------+
1 row in set (0.00 sec)

mysql> select ceil(3.1);    # 向上取整
+-----------+
| ceil(3.1) |
+-----------+
|         4 |
+-----------+
1 row in set (0.00 sec)

mysql> select floor(3.9);   # 向下取整
+------------+
```

```
| floor(3.9) |
+------------+
|          3 |
+------------+
1 row in set (0.00 sec)

mysql> select truncate(3.1415926,3);    # 截取数字
+-----------------------+
| truncate(3.1415926,3) |
+-----------------------+
|                 3.141 |
+-----------------------+
1 row in set (0.00 sec)
```

1.7.2 字符串类

```
mysql> select ucase('i am a boy!');     # 转成大写
+----------------------+
| ucase('i am a boy!') |
+----------------------+
| I AM A BOY!          |
+----------------------+
1 row in set (0.00 sec)

mysql> select lcase('I Am A Boy!');     #转成小写
+----------------------+
| lcase('I Am A Boy!') |
+----------------------+
| i am a boy!          |
+----------------------+
1 row in set (0.00 sec)

mysql> select left('abcde',3);      # 从左边开始截取，截取3个
+-----------------+
| left('abcde',3) |
+-----------------+
| abc             |
+-----------------+
1 row in set (0.00 sec)

mysql> select right('abcde',3);     # 从右边开始截取，截取3个
+------------------+
| right('abcde',3) |
+------------------+
| cde              |
+------------------+
1 row in set (0.00 sec)

mysql> select substring('abcde',2,3);   #从第2个位置开始截取，截取3个【位置从1开始】
+------------------------+
| substring('abcde',2,3) |
+------------------------+
| bcd                    |
+------------------------+
1 row in set (0.00 sec)
```

```
mysql> select concat('中国','上海');    # 字符串相连
+----------------------+
| concat('中国','上海')       |
+----------------------+
| 中国上海                     |
+----------------------+
1 row in set (0.00 sec)

mysql> select concat(stuname,'-',stusex) from stuinfo;  # 将表中的姓名和性别连接起来
+---------------------------+
| concat(stuname,'-',stusex) |
+---------------------------+
| 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(字段1，字段2)  如果字段1不为空就显示字段1，否则，显示字段2
mysql> select stuname,coalesce(writtenexam,'缺考'),coalesce(labexam,'缺考') from
stuinfo natural left join stumarks;   # 将考试成绩为空的显示为缺考
+----------+-----------------------------+--------------------------+
| stuname  | coalesce(writtenexam,'缺考')    | coalesce(labexam,'缺考')    |
+----------+-----------------------------+--------------------------+
| 张秋丽         | 77                           | 82                       |
| 李文才         | 50                           | 90                       |
| 李斯文         | 88                           | 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                       |
+----------+-----------------------------+--------------------------+

mysql> select length('锄禾日当午');      # 字节长度
+----------------------+
| length('锄禾日当午')          |
+----------------------+
|                   10 |
+----------------------+
1 row in set (0.00 sec)

mysql> select char_length('锄禾日当午');     # 字符个数
+---------------------------+
| char_length('锄禾日当午')          |
+---------------------------+
|                         5 |
+---------------------------+
1 row in set (0.00 sec)
```

1.7.3 时间类

```
mysql> select unix_timestamp(); #获取时间戳
+------------------+
| unix_timestamp() |
+------------------+
|       1537084508 |
+------------------+
1 row in set (0.00 sec)

mysql> select from_unixtime(unix_timestamp());  # 将时间戳转成年-月-日 小时:分钟:秒的
格式
+---------------------------------+
| from_unixtime(unix_timestamp()) |
+---------------------------------+
| 2018-09-16 15:55:56             |
+---------------------------------+
1 row in set (0.00 sec)

mysql> select now();        # 获取当前日期时间
+---------------------+
| now()               |
+---------------------+
| 2018-09-16 15:57:04 |
+---------------------+
1 row in set (0.00 sec)

mysql> select year(now()) 年,month(now()) 月, day(now()) 日,hour(now())
小,minute(now()) 分钟,second(now()) 秒;
+------+------+------+------+------+------+
| 年     | 月     | 日     | 小时   | 分钟     | 秒     |
+------+------+------+------+------+------+
| 2018 |    9 |   16 |   15 |   59 |   14 |
+------+------+------+------+------+------+
1 row in set (0.00 sec)

mysql> select dayname(now()) 星期,monthname(now()),dayofyear(now()) 本年的第几天;
+--------+-----------------+-------------+
| 星期       | monthname(now()) | 本年的第几天           |
+--------+-----------------+-------------+
| Sunday | September       |          259 |
+--------+-----------------+-------------+
1 row in set (0.00 sec)

mysql> select datediff(now(),'2008-8-8');   # 日期相减
+----------------------------+
| datediff(now(),'2008-8-8') |
+----------------------------+
|                       3691 |
+----------------------------+
1 row in set (0.00 sec)

mysql> select convert(now(),date),convert(now(),time);  # 将now()转成日期和时间
+--------------------+--------------------+
| convert(now(),date) | convert(now(),time) |
```

```
+--------------------+--------------------+
| 2018-09-16         | 16:07:24           |
+--------------------+--------------------+

mysql> select cast(now() as date),cast(now() as time);   # 将now()转成日期和时间
+--------------------+--------------------+
| cast(now() as date) | cast(now() as time) |
+--------------------+--------------------+
| 2018-09-16         | 16:08:03           |
+--------------------+--------------------+
1 row in set (0.00 sec)
```

1.7.4 加密函数

```
+----------------------------------+------------------------------------------+
| md5('root')                      | sha('root')                              |
+----------------------------------+------------------------------------------+
| 63a9f0ea7bb98050796b649e85481845 | dc76e9f0c0006e8f919e0c515c66dbba3982f785 |
+----------------------------------+------------------------------------------+
1 row in set (0.00 sec)
```

1.7.5 判断函数

语法

```
if(表达式,值1,值2)
```

例题:

```
mysql> select if(10%2=0,'偶数','奇数');
+--------------------------+
| if(10%2=0,'偶数','奇数')        |
+--------------------------+
| 偶数                         |
+--------------------------+
1 row in set (0.00 sec)

# 语文和数学都超过60分才通过
mysql> select stuname,ch,math,if(ch>=60 && math>=60,'通过','不通过') '是否通过' from
stu;
+----------+------+------+----------+
| stuname  | ch   | math | 是否通过       |
+----------+------+------+----------+
| 张秋丽         |   80 | NULL | 不通过        |
| 李文才         |   77 |   76 | 通过         |
| 李斯文         |   55 |   82 | 不通过        |
| 欧阳俊雄        | NULL |   74 | 不通过        |
| 诸葛丽丽         |   72 |   56 | 不通过        |
| 争青小子         |   86 |   92 | 通过         |
| 梅超风         |   74 |   67 | 通过         |
| Tom      |   65 |   67 | 通过         |
| Tabm     |   88 |   77 | 通过         |
+----------+------+------+----------+
```

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

1.8 预处理

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]
```

例题一：

```
mysql> prepare stmt from 'select * from stuinfo';   # 创建预处理
Query OK, 0 rows affected (0.00 sec)
Statement prepared

mysql> execute stmt;    # 执行预处理
+--------+----------+--------+--------+---------+------------+
| 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)
```

例题二：传递参数

```
mysql> delimiter //
mysql> prepare stmt from 'select * from stuinfo where stuno=?' // -- ?是位置占位符
Query OK, 0 rows affected (0.00 sec)
Statement prepared

mysql> set @id='s25301';        -- 变量以@开头，通过set给变量赋值
    -> execute stmt using @id //  -- 执行预处理，传递参数
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)
```

```
脚下留心：
1、?是位置占位符
2、变量以@开头
3、通过set给变量赋值
```

例题三：传递多个参数

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

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

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)
```

1.9 存储过程【procedure】

1.7.1 存储过程的优点

1. 存储过程可以减少网络流量
2. 允许模块化设计
3. 支持事务

1.7.2 创建存储过程

语法:

```
create procedure 存储过程名(参数)
begin
    //sql语句
end;
```

脚下留心：由于过程中有很多SQL语句，每个语句的结束都要用（；）结束。默认情况下，分号既表示语句结束，又表示向服务器发送SQL语句。我们希望分号仅表示语句的结束，不要将SQL语句发送到服务器执行，通过delimiter来更改结束符。

例题

```
mysql> delimiter //
mysql> create procedure proc()      -- 创建存储过程
    -> begin
    -> select * from stuinfo;
    -> end //
Query OK, 0 rows affected (0.00 sec)
```

1.7.3 调用存储过程

语法:

```
call 存储过程名()
```

例题:

```
mysql> call proc() //     -- 调用存储过程
+--------+----------+--------+--------+---------+------------+
| 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)
```

1.7.4 删除存储过程

语法

```
drop procedure [if exists] 存储过程名
```

例题:

```
mysql> drop procedure proc //    -- 删除存储过程
Query OK, 0 rows affected (0.00 sec)
```

1.7.5 查看存储过程的信息

```
show create procedure 存储过程名\G
```

例题

```
mysql> show create procedure proc \G
*************************** 1. row ***************************
           Procedure: proc
            sql_mode:
STRICT_TRANS_TABLES,NO_AUTO_CREATE_USER,NO_ENGINE_SUBSTITUTION
    Create Procedure: CREATE DEFINER=`root`@`localhost` PROCEDURE `proc`()
begin
select * from stuinfo;
end
character_set_client: gbk
collation_connection: gbk_chinese_ci
  Database Collation: utf8_general_ci
1 row in set (0.00 sec)
```

1.7.6 显示所有的存储过程

```
mysql> show procedure status \G
```

1.7.7　　存储过程的参数

存储过程的参数分为：输入参数（in）【默认】，输出参数（out），输入输出参数（inout）

存储过程不能使用return返回值，要返回值只能通过"输出参数"来向外传递值。

例题一：传递学号，获取对应的信息

```
mysql> create procedure proc(in param varchar(10))   -- 输入参数
    -> select * from stuinfo where stuno=param //
Query OK, 0 rows affected (0.00 sec)

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

例题二：查找同桌

```
mysql> create procedure proc(name varchar(10))
    -> begin
    -> declare seat tinyint;   -- 声明局部变量
    -> select stuseat into seat from stuinfo where stuname=name;  -- 将座位号保存到
变量中
    -> select * from stuinfo where stuseat=seat+1 or stuseat=seat-1;  -- 查找同桌
    -> end //
Query OK, 0 rows affected (0.00 sec)

mysql> call proc('李文才') //
+--------+----------+--------+--------+---------+------------+
| stuNo  | stuName  | stuSex | stuAge | stuSeat | stuAddress |
+--------+----------+--------+--------+---------+------------+
| s25303 | 李斯文        | 女       |     22 |       2 | 北京           |
| s25304 | 欧阳俊雄        | 男       |     28 |       4 | 天津           |
+--------+----------+--------+--------+---------+------------+
2 rows in set (0.00 sec)
```

强调

```
1、通过declare关键字声明局部变量；全局变量@开头就可以了
2、给变量赋值有两种方法
    方法一：set 变量名=值
    方法二：select 字段 into 变量 from 表 where 条件
3、声明的变量不能与列名同名
```

例题三：输出参数

```
mysql> create procedure proc(num int, out result int)  //out 表示输出参数
    -> begin
    -> set result=num*num;
    -> end //
Query OK, 0 rows affected (0.00 sec)

mysql> call proc(10,@result) //
Query OK, 0 rows affected (0.00 sec)

mysql> select @result //
+---------+
| @result |
+---------+
|     100 |
+---------+
1 row in set (0.00 sec)
```

例题四：输入输出参数

```
mysql> create procedure proc(inout num int)  #  inout 表示是输入输出参数
    -> begin
    -> set num=num*num;
    -> end //
Query OK, 0 rows affected (0.00 sec)

mysql> set @num=10;
    -> call proc(@num);
    -> select @num //
Query OK, 0 rows affected (0.00 sec)

Query OK, 0 rows affected (0.00 sec)

+------+
| @num |
+------+
|  100 |
+------+
1 row in set (0.00 sec)
```