Mysql 数据查询

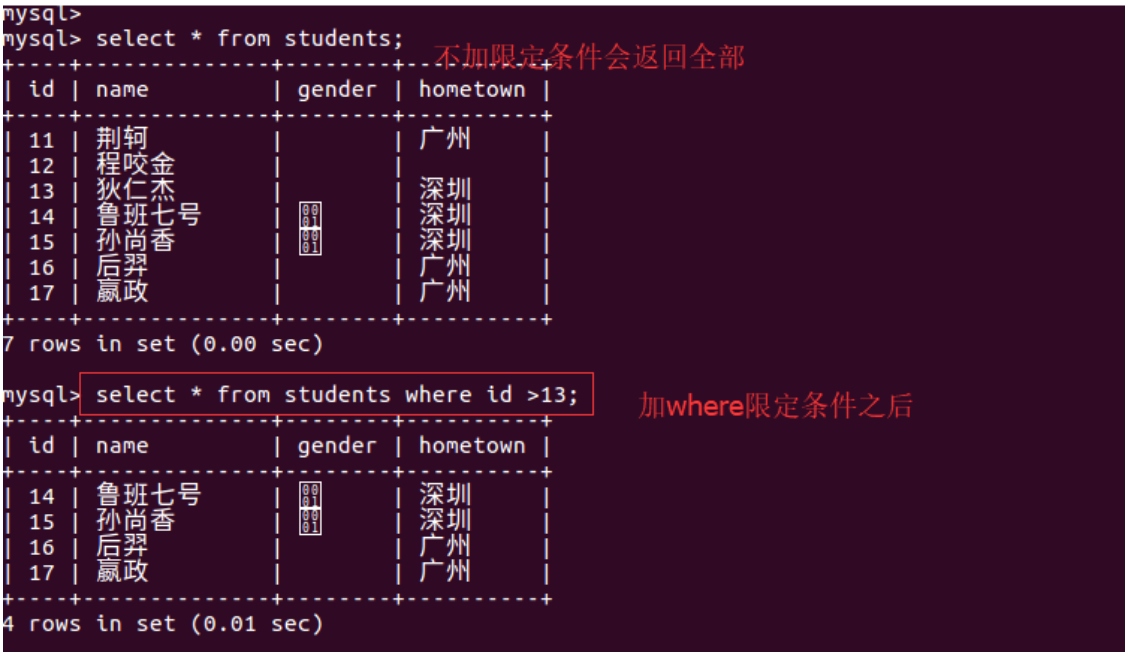
1、条件查询

1.1、条件查询

刚才讲修改与删除的时候提到过 where，使用 where 限定语句，查询集只返回条件为 True 的内容。

例如

`select * from students where id>13;` #这个查询语句将会返回 `id` 大于 `10` 的数据，不会返回小于 `13` 的



条件运算符

1、比较运算符

比较运算符	描述
=	等于
>	大于
<	小于
>=	大于等于
<=	小于等于
!=或者<>	不等于

案例：

查询名字叫后羿的同学信息

```
mysql> select * from students where name='后羿';
+----+--------+--------+----------+
| id | name   | gender | hometown |
+----+--------+--------+----------+
| 16 | 后羿   |        | 广州     |
+----+--------+--------+----------+
1 row in set (0.00 sec)
```

查询 id 小于 15 的同学的信息

```
mysql> select * from students where id<15;
+----+--------------+--------+----------+
| id | name         | gender | hometown |
+----+--------------+--------+----------+
| 11 | 荆轲         |        | 广州     |
| 12 | 程咬金       |        |          |
| 13 | 狄仁杰       |        | 深圳     |
| 14 | 鲁班七号     | 01     | 深圳     |
+----+--------------+--------+----------+
4 rows in set (0.00 sec)
```

查询家乡不在广州的同学的信息

```
mysql> select * from students where hometown !='广州';
+----+--------------+--------+----------+
| id | name         | gender | hometown |
+----+--------------+--------+----------+
| 12 | 程咬金       |        |          |
| 13 | 狄仁杰       |        | 深圳     |
| 14 | 鲁班七号     | 01     | 深圳     |
| 15 | 孙尚香       | 01     | 深圳     |
+----+--------------+--------+----------+
4 rows in set (0.00 sec)
```

逻辑运算符

逻辑运算符	描述
and	逻辑与
or	逻辑或
not	逻辑非

案例：

查询家在深圳的男同学

```
select * from students where hometown='深圳' and gender=0;
```

```
mysql> select * from students where hometown='深圳' and gender=0;
+----+-----------+--------+----------+
| id | name      | gender | hometown |
+----+-----------+--------+----------+
| 13 | 狄仁杰    |        | 深圳     |
+----+-----------+--------+----------+
1 row in set (0.01 sec)
```

查询性别为女生或者家住广州的

```
select * from students where hometown='广州' or gender=1;
```

```
mysql> select * from students where hometown='广州' or gender=1;
+----+--------------+--------+----------+
| id | name         | gender | hometown |
+----+--------------+--------+----------+
| 11 | 荆轲         |        | 广州     |
| 14 | 鲁班七号     | [illegible]      | 深圳     |
| 15 | 孙尚香       | [illegible]      | 深圳     |
| 16 | 后羿         |        | 广州     |
| 17 | 嬴政         |        | 广州     |
+----+--------------+--------+----------+
5 rows in set (0.00 sec)
```

查询除了 id=2 的同学信息

```
select * from students where not id=2;
```

```
mysql>
mysql> select * from students where not id=2;
+----+--------------+--------+----------+
| id | name         | gender | hometown |
+----+--------------+--------+----------+
| 11 | 荆轲         |        | 广州     |
| 12 | 程咬金       |        |          |
| 13 | 狄仁杰       |        | 深圳     |
| 14 | 鲁班七号     |        | 深圳     |
| 15 | 孙尚香       |        | 深圳     |
| 16 | 后羿         |        | 广州     |
| 17 | 嬴政         |        | 广州     |
+----+--------------+--------+----------+
7 rows in set (0.00 sec)
```

模糊查询

like 表示模糊查询
% 表示任意多个字符
_ 表示一个字符
rlike 可以匹配正则
in 包含在里面的

如果需要匹配%本身，那么则需要使用%%

案例

```
mysql> select * from students where hometown like '%州';
+----+--------+--------+----------+
| id | name   | gender | hometown |
+----+--------+--------+----------+
| 11 | 荆轲   |        | 广州     |
| 16 | 后羿   |        | 广州     |
| 17 | 嬴政   |        | 广州     |
+----+--------+--------+----------+
3 rows in set (0.00 sec)

mysql> select * from students where hometown rlike '.*州';
+----+--------+--------+----------+
| id | name   | gender | hometown |
+----+--------+--------+----------+
| 11 | 荆轲   |        | 广州     |
| 16 | 后羿   |        | 广州     |
| 17 | 嬴政   |        | 广州     |
+----+--------+--------+----------+
3 rows in set (0.00 sec)
```

like模糊匹配

rlike 使用可使用正则

1.2、in 用法匹配括号里面的，符合就返回结果集中

```
mysql> select * from students where id in (13,16,14);
+----+--------------+--------+----------+
| id | name         | gender | hometown |
+----+--------------+--------+----------+
| 13 | 狄仁杰       |        | 深圳     |
| 14 | 鲁班七号     |        | 深圳     |
| 16 | 后羿         |        | 广州     |
+----+--------------+--------+----------+
3 rows in set (0.00 sec)
```

1.3、between and 表示一段区间

查询 id13 到 16

```
select * from students where id between 13 and 16;
```

```
mysql> select * from students where id between 13 and 16;
+----+--------------+--------+----------+
| id | name         | gender | hometown |
+----+--------------+--------+----------+
| 13 | 狄仁杰       |        | 深圳     |
| 14 | 鲁班七号     |        | 深圳     |
| 15 | 孙尚香       |        | 深圳     |
| 16 | 后羿         |        | 广州     |
+----+--------------+--------+----------+
4 rows in set (0.01 sec)
```

1.4、判断为 null 值

先插入几条数据 hometown 为 null 值的。

```
mysql> select * from students;
+----+--------------+--------+----------+
| id | name         | gender | hometown |
+----+--------------+--------+----------+
| 11 | 荆轲         |        | 广州     |
| 12 | 程咬金       |        |          |
| 13 | 狄仁杰       |        | 深圳     |
| 14 | 鲁班七号     |        | 深圳     |
| 15 | 孙尚香       |        | 深圳     |
| 16 | 后羿         |        | 广州     |
| 17 | 嬴政         |        | 广州     |
| 18 | 白起         |        | NULL     |
| 19 | 安妮         |        | NULL     |
+----+--------------+--------+----------+
9 rows in set (0.00 sec)

mysql> select * from students where hometown=null;
Empty set (0.00 sec)
```

直接用 hometown=null 并不能得到结果，在 mysql 中 null 表示空。 如果要查询为 null 值的应该是使用 is

```
select * from students where hometown is null;
```

```
mysql> select * from students where hometown is null;
+----+--------+--------+----------+
| id | name   | gender | hometown |
+----+--------+--------+----------+
| 18 | 白起   |        | NULL     |
| 19 | 安妮   |        | NULL     |
+----+--------+--------+----------+
2 rows in set (0.00 sec)
```

1.5、排序

order by 字段 [desc/asc]

desc 表示降序（从大到小排序）

asc 默认排序规则，表示升序（从小到大排序）

```
select * from students order by id desc; #按照 id 从大到小排序
```

```
mysql> select * from students order by id desc;
+----+--------------+--------+----------+
| id | name         | gender | hometown |
+----+--------------+--------+----------+
| 19 | 安妮         |        | NULL     |
| 18 | 白起         |        | NULL     |
| 17 | 嬴政         |        | 广州     |
| 16 | 后羿         |        | 广州     |
| 15 | 孙尚香       |        | 深圳     |
| 14 | 鲁班七号     |        | 深圳     |
| 13 | 狄仁杰       |        | 深圳     |
| 12 | 程咬金       |        |          |
| 11 | 荆轲         |        | 广州     |
+----+--------------+--------+----------+
9 rows in set (0.00 sec)
```

2、聚合函数

1. 2.1、聚合函数主要是为了快速得到结果，经常使用的几个聚合函数

count 统计行数
max 计算最大值
min 计算最小值
sum 求和
avg 求平均数
round 保留几位小数

2.2、count 函数

查询学生表中一共有多少人

```
select count(*) from students;
```

```
mysql> select count(*)  from students;
+----------+
| count(*) |
+----------+
|        9 |
+----------+
1 row in set (0.00 sec)
```

2.3、max(列字段) 表示查询这一列中的最大值

查询学生表中 id 最大的

```
mysql> select max(id) from students;
+---------+
| max(id) |
+---------+
|      19 |
+---------+
1 row in set (0.05 sec)
```

2.4、min(列字段) 表示查询这一列中的最小值

查询 students 中 id 最小值

```
mysql> select min(id) from students;
+---------+
| min(id) |
+---------+
|      11 |
+---------+
1 row in set (0.00 sec)
```

2.5、avg 求平均数

查询 students 表中的 id 的平均数

```
mysql>
mysql> select avg(id) from students;
+---------+
| avg(id) |
+---------+
| 15.0000 |
+---------+
1 row in set (0.05 sec)
```

求平均数之后默认保留了 4 位小数，如果要自己决定保留小数位！

2.6、round 函数

round(decimal,num) #decimal 小数，num 保留位数

```
mysql> select round(avg(id),2) from students;
+------------------+
| round(avg(id),2) |
+------------------+
|            15.00 |
+------------------+
1 row in set (0.00 sec)
```

3、分组与分页

3.1、分组

group by 字段 # 以 xx 字段作为分组依据分组

注意：分组后分组依据会显示在结果集，其他列不会出现

统计男生，女生分别有多少人

```
select gender,count(*) from students group by gender;
```

```
mysql> select gender,count(*) from students group by gender;
+--------+----------+
| gender | count(*) |
+--------+----------+
|        |        7 |
| 男     |        2 |
+--------+----------+
2 rows in set (0.00 sec)
```

在统计人数的时候结果集中显示的是 count(*)这个有时候我们并不知道他代表的是什么，如何改成有语义的命名。

3.2、as 取别名

```
mysql> select gender,count(*) as '人数' from students group by gender;
+--------+--------+
| gender | 人数   |
+--------+--------+
|        |      7 |
| 男     |      2 |
+--------+--------+
2 rows in set (0.00 sec)
```

3.3、分组后条件筛选

分组后不能使用 where 做条件过滤，需要一个使用新的 having 函数

```
mysql> select gender,count(*) as '人数' from students group by gender where gender=0
;
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that corr
esponds to your MySQL server version for the right syntax to use near 'where gender=
0' at line 1
mysql>
mysql> select gender,count(*) as '人数' from students group by gender having gender=
0;
+--------+--------+
| gender | 人数   |
+--------+--------+
|        |      7 |
+--------+--------+
1 row in set (0.00 sec)
```

分组后使用where报错

使用having正确过滤数据

3.4、where 与 having 的区别

where 用户 from 之后的条件过滤

having 用在分组之后的条件过滤，两个功能是一样的，只是作用的位置不一样。

3.5、limit 分页

如果数据量很大的话，一次性将所有数据查询出来，不仅不方便查看而且耗费传输带宽。

那么就使用到了分页功能，一次只查询一页的数据

`select * from students limit start,count;` #start 从第几条数据开始，count 表示获取几条数据

`select * from students limit 0,3;` #查询前 3 名同学信息

```
mysql> select * from students limit 0,3;
+----+-----------+--------+----------+
| id | name      | gender | hometown |
+----+-----------+--------+----------+
| 11 | 荆轲      |        | 广州     |
| 12 | 程咬金    |        |          |
| 13 | 狄仁杰    |        | 深圳     |
+----+-----------+--------+----------+
3 rows in set (0.00 sec)
```

实例：每页显示 3 条数据，要求获取第 3 页的数据

```
第几页:   start     每页显示数量
1         0,        3
2         3,        3
3         6,        3
4         9,        3
5         12        3
n         (3*n-1),  3
结论:
每一页显示m条数据，求第n页的数据
(m*n-1),m

select * from students limit (m*n-1),m;

```

4、连接查询

在讲连接查询前我们先给 students 表中的学生分下班。

新加一个 class_id 字段，保存班级 id，之前已经有一个 class 表已经保存了班级信息了。

students 表添加 class_id 字段

```
alter table students add class_id int default null;
```

id 小于 15 分到 1 班，大于等于分到 2 班

```
update students set class_id = 1 where id < 15;
update students set class_id = 2 where id >=15;
```

```
mysql> select * from class;
+----+---------+--------+
| id | name    | delete |
+----+---------+--------+
|  1 | python1 |        |
|  2 | python2 |        |
+----+---------+--------+
2 rows in set (0.00 sec)

mysql> select * from students;
+----+-------------+--------+----------+----------+
| id | name        | gender | hometown | class_id |
+----+-------------+--------+----------+----------+
| 11 | 荆轲        |        | 广州     |        1 |
| 12 | 程咬金      |        |          |        1 |
| 13 | 狄仁杰      |        | 深圳     |        1 |
| 14 | 鲁班七号    | 0      | 深圳     |        1 |
| 15 | 孙尚香      | 0      | 深圳     |        2 |
| 16 | 后羿        |        | 广州     |        2 |
| 17 | 嬴政        |        | 广州     |        2 |
| 18 | 白起        |        | NULL     |        2 |
| 19 | 安妮        |        | NULL     |        2 |
+----+-------------+--------+----------+----------+
9 rows in set (0.00 sec)
```

4.1、mysql 中有三种连接查询方式

- 内连接查询：查询的结果为两个表匹配到的数据，两个表都能匹配上的数据将返回给结果集

  select * from 表 1 inner join 表 2 on 表 1.列=表 2.列；

使用内连接查询学生表和班级表

```
mysql>
mysql> select * from students inner join class on students.class_id = class.id;
+----+--------------+--------+----------+----------+----+---------+--------+
| id | name         | gender | hometown | class_id | id | name    | delete |
+----+--------------+--------+----------+----------+----+---------+--------+
| 11 | 荆轲         |        | 广州     |        1 |  1 | python1 |        |
| 12 | 程咬金       |        |          |        1 |  1 | python1 |        |
| 13 | 狄仁杰       |        | 深圳     |        1 |  1 | python1 |        |
| 14 | 鲁班七号     |        | 深圳     |        1 |  1 | python1 |        |
| 15 | 孙尚香       |        | 深圳     |        2 |  2 | python2 |        |
| 16 | 后羿         |        | 广州     |        2 |  2 | python2 |        |
| 17 | 嬴政         |        | 广州     |        2 |  2 | python2 |        |
| 18 | 白起         |        | NULL     |        2 |  2 | python2 |        |
| 19 | 安妮         |        | NULL     |        2 |  2 | python2 |        |
+----+--------------+--------+----------+----------+----+---------+--------+
9 rows in set (0.04 sec)
```

在连接查询的时候需要多次用到表名，如果表名过长可以使用 as 给表取别名。

- 右连接查询：查询的结果为两个表匹配到的数据，右表特有的数据，对于左表中不存在的数据使用 null 填充

  select * from 表 1 right join 表 2 on 表 1.列=表 2.列；

使右连接查询学生表和班级表

```
mysql> select * from students as s  right join class as c  on s.class_id=c.id;
+------+--------------+--------+----------+----------+----+---------+--------+
| id   | name         | gender | hometown | class_id | id | name    | delete |
+------+--------------+--------+----------+----------+----+---------+--------+
|   11 | 荆轲         |        | 广州     |        1 |  1 | python1 |        |
|   12 | 程咬金       |        |          |        1 |  1 | python1 |        |
|   13 | 狄仁杰       |        | 深圳     |        1 |  1 | python1 |        |
|   14 | 鲁班七号     |        | 深圳     |        1 |  1 | python1 |        |
|   15 | 孙尚香       |        | 深圳     |        2 |  2 | python2 |        |
|   16 | 后羿         |        | 广州     |        2 |  2 | python2 |        |
|   17 | 嬴政         |        | 广州     |        2 |  2 | python2 |        |
|   18 | 白起         |        | NULL     |        2 |  2 | python2 |        |
|   19 | 安妮         |        | NULL     |        2 |  2 | python2 |        |
| NULL | NULL         | NULL   | NULL     |     NULL |  3 | python3 |        |
+------+--------------+--------+----------+----------+----+---------+--------+
10 rows in set (0.00 sec)
```

- 左连接查询：查询的结果为两个表匹配到的数据，左表特有的数据，对于右表中不存在的数据使用 null 填充

  select * from 表 1 left join 表 2 on 表 1.列=表 2.列；

```
mysql>
mysql> select * from students as s  left join class as c  on s.class_id=c.id;
+----+--------------+--------+----------+----------+------+---------+--------+
| id | name         | gender | hometown | class_id | id   | name    | delete |
+----+--------------+--------+----------+----------+------+---------+--------+
| 11 | 荆轲         |        | 广州     |        1 |    1 | python1 |        |
| 12 | 程咬金       |        |          |        1 |    1 | python1 |        |
| 13 | 狄仁杰       |        | 深圳     |        1 |    1 | python1 |        |
| 14 | 鲁班七号     |        | 深圳     |        1 |    1 | python1 |        |
| 15 | 孙尚香       |        | 深圳     |        2 |    2 | python2 |        |
| 16 | 后羿         |        | 广州     |        2 |    2 | python2 |        |
| 17 | 嬴政         |        | 广州     |        2 |    2 | python2 |        |
| 18 | 白起         |        | NULL     |        2 |    2 | python2 |        |
| 19 | 安妮         |        | NULL     |        2 |    2 | python2 |        |
| 20 | 赵信         |        | NULL     |     NULL | NULL | NULL    | NULL   |
+----+--------------+--------+----------+----------+------+---------+--------+
10 rows in set (0.00 sec)
```

4.2、案例：

查询学生姓名以及对应班级名称

```
select s.name,c.name from students as s  inner join class as c  on s.class_id=c.id;
```

```
mysql> select s.name,c.name from students as s  inner join class as c  on s.class_id
=c.id;
+--------------+---------+
| name         | name    |
+--------------+---------+
| 荆轲         | python1 |
| 程咬金       | python1 |
| 狄仁杰       | python1 |
| 鲁班七号     | python1 |
| 孙尚香       | python2 |
| 后羿         | python2 |
| 嬴政         | python2 |
| 白起         | python2 |
| 安妮         | python2 |
+--------------+---------+
```

students 和 class 表中都有 name 字段，在显示结果集的时候需要指定表。

查询 python1 班所有学生的个人信息和班级信息

```
select * from students as s  inner join class as c  on s.class_id=c.id where class_id=1;
```

```
mysql> select * from students as s  inner join class as c  on s.class_id=c.id where
class_id=1;
+----+--------------+--------+----------+----------+----+---------+--------+
| id | name         | gender | hometown | class_id | id | name    | delete |
+----+--------------+--------+----------+----------+----+---------+--------+
| 11 | 荆轲         |        | 广州     |        1 |  1 | python1 |        |
| 12 | 程咬金       |        |          |        1 |  1 | python1 |        |
| 13 | 狄仁杰       |        | 深圳     |        1 |  1 | python1 |        |
| 14 | 鲁班七号     | 01     | 深圳     |        1 |  1 | python1 |        |
+----+--------------+--------+----------+----------+----+---------+--------+
4 rows in set (0.00 sec)
```

5、子查询

在一个 select 语句中嵌入了另外一个 select 语句，嵌入的这个 select 语句就是子查询语句。

子查询是辅助主查询的，充当数据源，或者充当条件。子查询是一条独立的语句，即使单独拿出子查询也是可以正常执行的。

students 表中再加一列年龄 age，将学生的年龄补充完整，方便下面演示

5.1、子查询有四种类型：

- 第一种子查询返回一行一列的数据，称之为标量子查询

  –查询学生年龄小于平均年龄的学生信息

  –子查询语句先查出平均年龄，`select avg(age) from students;`

```
select * from students where age < (select avg(age) from students);
```

```
mysql> select avg(age) from students;
+----------+
| avg(age) |
+----------+
|  17.4000 |
+----------+
1 row in set (0.00 sec)
```

子查询语句

将子查询嵌套到主查询

```
mysql> select * from students where age < (select avg(age) from students);
+----+--------------+--------+----------+----------+------+
| id | name         | gender | hometown | class_id | age  |
+----+--------------+--------+----------+----------+------+
| 14 | 鲁班七号     | 01     | 深圳     |        1 |   15 |
| 15 | 孙尚香       | 01     | 深圳     |        2 |   12 |
| 17 | 嬴政         |        | 广州     |        2 |   14 |
| 20 | 赵信         |        | NULL     |     NULL |   16 |
+----+--------------+--------+----------+----------+------+
4 rows in set (0.00 sec)
```

- 第二种子查询返回的是一列多行的数据，称之为列级子查询

  –查询 class 表中已经安排学生的班级信息

```
select * from class where id in (select class_id from students);
```

```
mysql> select * from class where id in (select class_id from students);
+----+---------+--------+
| id | name    | delete |
+----+---------+--------+
|  1 | python1 |        |
|  2 | python2 |        |
+----+---------+--------+
2 rows in set (0.00 sec)
```

注意:上面这里使用 in 做条件判断

in 符合列子查询里面一个

any | some 任意一个　主查询 where 列 = all(列子查询) :

all 格式: 主查询 where 列 = all(列子查询) : 等于里面所有，主查询 where 列 <> all(列子查询) : 不等于子查询

- 第三种子查询返回的是一行多列，称之为行级子查询

  –查询一班同学中年龄最大的同学信息，单独使用子查询这条语句查的结果可以看出结果集是一行多列。嵌套到主查询后将查出一班同学中年龄最大的同学信息。

```
mysql> select class_id,max(age) from students where class_id=1
    -> ;
+----------+----------+
| class_id | max(age) |
+----------+----------+
|        1 |       20 |
+----------+----------+
1 row in set (0.00 sec)

mysql> select * from students where (class_id,age)=(select class_id,max(age) from students where class_id=1);
+----+-----------+--------+----------+----------+------+
| id | name      | gender | hometown | class_id | age  |
+----+-----------+--------+----------+----------+------+
| 12 | 程咬金    |        |          |        1 |   20 |
+----+-----------+--------+----------+----------+------+
1 row in set (0.00 sec)
```

- 第四种子查询返回多行多列，称之为表级子查询。

  –查询学生信息对应班级名称，子查询返回的数据充当数据源，再进行过滤。

```
select t1.name,t1.class_name from (
  select s.*,c.name as class_name from students as s inner join class as c  on s.
class_id =c.id) as t1;
```

```
mysql> select t1.name,t1.class_name from (select s.*,c.name as class_name from students
 as s inner join class as c  on s.class_id =c.id) as t1;
+--------------+------------+
| name         | class_name |
+--------------+------------+
| 荆轲         | python1    |
| 程咬金       | python1    |
| 狄仁杰       | python1    |
| 鲁班七号     | python1    |
| 孙尚香       | python2    |
| 后羿         | python2    |
| 嬴政         | python2    |
| 白起         | python2    |
| 安妮         | python2    |
+--------------+------------+
9 rows in set (0.00 sec)
```

6、保存查询结果

语句格式：

```
insert into 表名 （列1，列2） select .....  #这个方法可以将查询的结果直接保存到表里。
```

新建一个表用来保存查询结果，学生 id，名字，班级，年龄

```
create table info (
id int unsigned auto_increment primary key not null,
name varchar(10) not null,
class_name varchar(10) not null,
age int(100) unsigned
);
```

全列插入，将查询结果插入到 info 表。

```
insert into info
select s.id,s.name,c.name as class_name,age from
students as s  inner join class as c  on s.class_id=c.id;
```

```
mysql> insert into info
    -> select s.id,s.name,c.name as class_name,age from
    -> students as s  inner join class as c  on s.class_id=c.id;
Query OK, 9 rows affected (0.01 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql> select * from info;
+----+--------------+------------+------+
| id | name         | class_name | age  |
+----+--------------+------------+------+
| 11 | 荆轲         | python1    |   18 |
| 12 | 程咬金       | python1    |   20 |
| 13 | 狄仁杰       | python1    |   18 |
| 14 | 鲁班七号     | python1    |   15 |
| 15 | 孙尚香       | python2    |   12 |
| 16 | 后羿         | python2    |   18 |
| 17 | 嬴政         | python2    |   14 |
| 18 | 白起         | python2    |   22 |
| 19 | 安妮         | python2    |   21 |
+----+--------------+------------+------+
9 rows in set (0.00 sec)
```

指定列插入，约束条件为 not null 的必须插入，其他可用为 null 的列自动使用 null 填充。

```
insert into info (name,class_name)
select s.name,c.name as class_name from
students as s  inner join class as c  on s.class_id=c.id;
```

```
mysql> insert into info (name,class_name)
    -> select s.name,c.name as class_name from
    -> students as s  inner join class as c  on s.class_id=c.id;
Query OK, 9 rows affected (0.00 sec)
Records: 9  Duplicates: 0  Warnings: 0

mysql>
mysql> select * from info;
+----+--------------+------------+------+
| id | name         | class_name | age  |
+----+--------------+------------+------+
| 11 | 荆轲         | python1    |   18 |
| 12 | 程咬金       | python1    |   20 |
| 13 | 狄仁杰       | python1    |   18 |
| 14 | 鲁班七号     | python1    |   15 |
| 15 | 孙尚香       | python2    |   12 |
| 16 | 后羿         | python2    |   18 |
| 17 | 嬴政         | python2    |   14 |
| 18 | 白起         | python2    |   22 |
| 19 | 安妮         | python2    |   21 |
| 20 | 荆轲         | python1    | NULL |
| 21 | 程咬金       | python1    | NULL |
| 22 | 狄仁杰       | python1    | NULL |
| 23 | 鲁班七号     | python1    | NULL |
| 24 | 孙尚香       | python2    | NULL |
| 25 | 后羿         | python2    | NULL |
| 26 | 嬴政         | python2    | NULL |
| 27 | 白起         | python2    | NULL |
| 28 | 安妮         | python2    | NULL |
+----+--------------+------------+------+
18 rows in set (0.00 sec)

mysql>
```

6.1、union 和 union all

- union all 将两次查询的结果集合并到一起显示
- union 将两个查询的结果集先去重后合并到一起显示，

6.1.1、union all

```
mysql> select * from students union all  select * from students;
+----+--------------+--------+----------+----------+------+
| id | name         | gender | hometown | class_id | age  |
+----+--------------+--------+----------+----------+------+
| 11 | 荆轲         |        | 广州     |        1 |   18 |
| 12 | 程咬金       |        |          |        1 |   20 |
| 13 | 狄仁杰       |        | 深圳     |        1 |   18 |
| 14 | 鲁班七号     | 0      | 深圳     |        1 |   15 |
| 15 | 孙尚香       | 0      | 深圳     |        2 |   12 |
| 16 | 后羿         |        | 广州     |        2 |   18 |
| 17 | 嬴政         |        | 广州     |        2 |   14 |
| 18 | 白起         |        | NULL     |        2 |   22 |
| 19 | 安妮         |        | NULL     |        2 |   21 |
| 11 | 荆轲         |        | 广州     |        1 |   18 |
| 12 | 程咬金       |        |          |        1 |   20 |
| 13 | 狄仁杰       |        | 深圳     |        1 |   18 |
| 14 | 鲁班七号     | 0      | 深圳     |        1 |   15 |
| 15 | 孙尚香       | 0      | 深圳     |        2 |   12 |
| 16 | 后羿         |        | 广州     |        2 |   18 |
| 17 | 嬴政         |        | 广州     |        2 |   14 |
| 18 | 白起         |        | NULL     |        2 |   22 |
| 19 | 安妮         |        | NULL     |        2 |   21 |
+----+--------------+--------+----------+----------+------+
18 rows in set (0.00 sec)
```

6.1.2、union 去重后合并

```
mysql>  mysql> select * from students union select * from students;
+----+--------------+--------+----------+----------+------+
| id | name         | gender | hometown | class_id | age  |
+----+--------------+--------+----------+----------+------+
| 11 | 荆轲         |        | 广州     |        1 |   18 |
| 12 | 程咬金       |        |          |        1 |   20 |
| 13 | 狄仁杰       |        | 深圳     |        1 |   18 |
| 14 | 鲁班七号     | 0      | 深圳     |        1 |   15 |
| 15 | 孙尚香       | 0      | 深圳     |        2 |   12 |
| 16 | 后羿         |        | 广州     |        2 |   18 |
| 17 | 嬴政         |        | 广州     |        2 |   14 |
| 18 | 白起         |        | NULL     |        2 |   22 |
| 19 | 安妮         |        | NULL     |        2 |   21 |
+----+--------------+--------+----------+----------+------+
9 rows in set (0.00 sec)
```