DQL-MySQL数据查询SQL

语法格式：

```
select 字段列表|*  from 表名
[where 搜索条件]
[group by 分组字段 [having 分组条件]]
[order by 排序字段 排序规则]
[limit 分页参数]
```

基础查询

```
# 查询表中所有列 所有数据
select * from users;

# 指定字段列表进行查询
select id,name,phone from users;
```

Where 条件查询

- 可以在where子句中指定任何条件
- 可以使用 and 或者 or 指定一个或多个条件
- where条件也可以运用在update和delete语句的后面
- where子句类似程序语言中if条件，根据mysql表中的字段值来进行数据的过滤

示例：

```
-- 查询users表中 age > 22的数据
select * from users where age > 22;

-- 查询 users 表中 name=某个条件值 的数据
select * from users where name = '王五';

-- 查询 users 表中 年龄在22到25之间的数据
select * from users where age >= 22 and age <= 25;
select * from users where age between 22 and 25;

-- 查询 users 表中 年龄不在22到25之间的数据
select * from users where age < 22 or age > 25;
select * from users where age not between 22 and 25;

-- 查询 users 表中 年龄在22到25之间的女生信息
select * from users where age >= 22 and age <= 25 and sex = '女';
```

and和or 使用时注意

假设要求 查询 users 表中 年龄为22或者25 的女生信息

select * from users where age=22 or age = 25 and sex = '女';

思考上面的语句能否返回符合条件的数据?

实际查询结果并不符合要求?

```
select * from users where age=22 or age = 25 and sex = '女';
+------+--------+------+-------+-------+------+------+
| id   | name   | age  | phone | email | sex  | mm   |
+------+--------+------+-------+-------+------+------+
|    1 | 章三   |   22 |       | NULL  | 男   |    0 |
| 1002 | cc     |   25 | 123   | NULL  | 女   | NULL |
+------+--------+------+-------+-------+------+------+
2 rows in set (0.00 sec)

-- 上面的查询结果并不符合 查询条件的要求。
-- 问题出在 sql 计算的顺序上，sql会优先处理and条件，所以上面的sql语句就变成了
-- 查询变成了为年龄22的不管性别，或者年龄为 25的女生

-- 如何改造sql符合我们的查询条件呢?
-- 使用小括号来关联相同的条件
select * from users where (age=22 or age = 25) and sex = '女';
+------+------+------+-------+-------+------+------+
| id   | name | age  | phone | email | sex  | mm   |
+------+------+------+-------+-------+------+------+
| 1002 | cc   |   25 | 123   | NULL  | 女   | NULL |
+------+------+------+-------+-------+------+------+
1 row in set (0.00 sec)
```

Like 子句

> 我们可以在where条件中使用=,<,> 等符合进行条件的过滤，但是当想查询某个字段是否包含时如何过滤?
>
> 可以使用like语句进行某个字段的模糊搜索，
>
> 例如： 查询 name字段中包含五的数据

```
-- like 语句  like某个确定的值 和。where name = '王五' 是一样
select * from users where name like '王五';
+----+--------+------+-------+-----------+------+------+
| id | name   | age  | phone | email     | sex  | mm   |
+----+--------+------+-------+-----------+------+------+
|  5 | 王五   |   24 | 10011 | ww@qq.com | 男   |    0 |
+----+--------+------+-------+-----------+------+------+
1 row in set (0.00 sec)

-- 使用 % 模糊搜索。%代表任意个任意字符
  -- 查询name字段中包含五的
  select * from users where name like '%五%';
```

```
  -- 查询name字段中最后一个字符 为 五的
  select * from users where name like '%五';

  -- 查询name字段中第一个字符 为 王 的
  select * from users where name like '王%';

-- 使用 _ 单个的下划线。表示一个任意字符，使用和%类似

  -- 查询表中 name 字段为两个字符的数据
  select * from users where name like '__';

  -- 查询 name 字段最后为五，的两个字符的数据
  select * from users where name like '_五';
```

注意：where子句中的like在使用%或者_进行模糊搜索时，效率不高，使用时注意：

- 尽可能的不去使用%或者_
- 如果需要使用，也尽可能不要把通配符放在开头处

Mysql中的统计函数（聚合函数）

max(),min(),count(),sum(),avg()

```
# 计算 users 表中 最大年龄，最小年龄，年龄和及平均年龄
select max(age),min(age),sum(age),avg(age) from users;
+----------+----------+----------+----------+
| max(age) | min(age) | sum(age) | avg(age) |
+----------+----------+----------+----------+
|       28 |       20 |      202 |  22.4444 |
+----------+----------+----------+----------+

-- 上面数据中的列都是在查询时使用的函数名，不方便阅读和后期的调用，可以通过别名方式 美化
select max(age) as max_age,
min(age) min_age,sum(age) as sum_age,
avg(age) as avg_age
from users;
+---------+---------+---------+---------+
| max_age | min_age | sum_age | avg_age |
+---------+---------+---------+---------+
|      28 |      20 |     202 | 22.4444 |
+---------+---------+---------+---------+

-- 统计 users 表中的数据量
select count(*) from users;
+----------+
| count(*) |
+----------+
|        9 |
+----------+
select count(id) from users;
+-----------+
| count(id) |
+-----------+
```

```
|         9 |
+-----------+

-- 上面的两个统计，分别使用了 count(*) 和 count(id),结果目前都一样，有什么区别?
-- count(*) 是按照 users表中所有的列进行数据的统计，只要其中一列上有数据，就可以计算
-- count(id) 是按照指定的 id 字段进行统计，也可以使用别的字段进行统计，
-- 但是注意，如果指定的列上出现了NULL值，那么为NULL的这个数据不会被统计
-- 假设有下面这样的一张表需要统计
+------+-----------+------+--------+-----------+------+------+
| id   | name      | age  | phone  | email     | sex  | mm   |
+------+-----------+------+--------+-----------+------+------+
|    1 | 章三      |   22 |        | NULL      | 男   |    0 |
|    2 | 李四      |   20 |        | NULL      | 女   |    0 |
|    5 | 王五      |   24 | 10011  | ww@qq.com | 男   |    0 |
| 1000 | aa        |   20 | 123    | NULL      | 女   | NULL |
| 1001 | bb        |   20 | 123456 | NULL      | 女   | NULL |
| 1002 | cc        |   25 | 123    | NULL      | 女   | NULL |
| 1003 | dd        |   20 | 456    | NULL      | 女   | NULL |
| 1004 | ff        |   28 | 789    | NULL      | 男   | NULL |
| 1005 | 王五六    |   23 | 890    | NULL      | NULL | NULL |
+------+-----------+------+--------+-----------+------+------+
9 rows in set (0.00 sec)

-- 如果按照sex这一列进行统计，结果就是8个而不是9个，因为sex这一列中有NULL值存在
mysql> select count(sex) from users;
+------------+
| count(sex) |
+------------+
|          8 |
+------------+
```

聚合函数除了以上简单的使用意外，通常情况下都是配合着分组进行数据的统计和计算

Group BY 分组

> group by 语句根据一个或多个列对结果集进行分组
>
> 一般情况下，是用与数据的统计或计算，配合聚合函数使用

```
-- 统计 users 表中 男女生人数，
-- 很明显按照上面的需要，可以写出两个语句进行分别统计
select count(*) from users where sex = '女';
select count(*) from users where sex = '男';
-- 可以使用分组进行统计，更方便
select sex,count(*) from users group by sex;
+------+----------+
| sex  | count(*) |
+------+----------+
| 男   |        4 |
| 女   |        5 |
+------+----------+
```

```
-- 统计1班和2班的人数
 select classid,count(*) from users group by classid;
+---------+----------+
| classid | count(*) |
+---------+----------+
|       1 |        5 |
|       2 |        4 |
+---------+----------+

-- 分别统计每个班级的男女生人数
select classid,sex,count(*) as num from users group by classid,sex;
+---------+------+-----+
| classid | sex  | num |
+---------+------+-----+
|       1 | 男   |   2 |
|       1 | 女   |   3 |
|       2 | 男   |   2 |
|       2 | 女   |   2 |
+---------+------+-----+

# 注意，在使用。group by分组时，一般除了聚合函数，其它在select后面出现的字段列都需要出现在grouop by 后面
```

Having 子句

> having时在分组聚合计算后，对结果再一次进行过滤，类似于where，
>
> where过滤的是行数据，having过滤的是分组数据

```
-- 要统计班级人数
select classid,count(*) from users group by classid;

-- 统计班级人数，并且要人数达到5人及以上
select classid,count(*) as num from users group by classid having num >=5;
```

Order by 排序

> 我们在mysql中使用select的语句查询的数据结果是根据数据在底层文件的结构来排序的，
>
> 首先不要依赖默认的排序，另外在需要排序时要使用orderby对返回的结果进行排序
>
> Asc 升序，默认
>
> desc降序

```sql
-- 按照年龄对结果进行排序，从大到小
select * from users order by age desc;

-- 从小到大排序 asc 默认就是。可以不写
select * from users order by age;

-- 也可以按照多个字段进行排序
select * from users order by age,id; # 先按照age进行排序，age相同情况下，按照id进行排序
select * from users order by age,id desc;
```

Limit 数据分页

- limit n　　提取n条数据，
- limit m,n 跳过m跳数据，提取n条数据

```sql
-- 查询users表中的数据，只要3条
select * from users limit 3;

-- 跳过前4条数据，再取3条数据
select * from users limit 4,3;

-- limit一般应用在数据分页上面
-- 例如每页显示10条数据，第三页的 limit应该怎么写？ 思考
第一页  limit 0,10
第二页  limit 10,10
第三页  limit 20,10
第四页  limit 30,10

-- 提取 user表中 年龄最大的三个用户数据 怎么查询?
select * from users order by age desc limit 3;
```

课后练习题

```sql
-- 1. 统计班级 classid为2的男女生人数?

-- 2. 获取每个班级的 平均年龄，并按照平均年龄从大到小排序

-- 3. 统计每个班级的人数，按照从大到小排序

-- 4. 获取班级人数最多的 班级id信息
```

总结:

> mysql中的查询语句比较灵活多样，所以需要多加练习，
>
> 并且在使用查询语句时，一定要注意sql的正确性和顺序

子句	说明	是否必须
select	要返回的列或表达式，字段列表| *	是
from	查询的数据表	需要在表中查询时
Where	数据行的过滤	否
group by	分组	仅在分组聚合计算时
having	分组后的数据过滤	否
order by	输出排序	否
limit	要提取的结果行数	否