今日内容：MySQL单表查询

授课老师：习知_前吉

联系方式：QQ：3264935908

准备工作

```
create table product(
    pid int primary key auto_increment, -- 自增长
    pname varchar(40),
    price double,
    num int
);

insert into product values(null,'苹果电脑',18000.0,10);
insert into product values(null,'华为5G手机',30000,20);
insert into product values(null,'小米手机',1800,30);
insert into product values(null,'iPhonex',8000,10);
insert into product values(null,'iPhone7',6000,200);
insert into product values(null,'iPhone6s',4000,1000);
insert into product values(null,'iPhone6',3500,100);
insert into product values(null,'iPhone5s',3000,100);
```

单表查询

查询某张表特定列的记录

```
select 列名,列名,列名... from 表
```

去重查询distinct

```
SELECT DISTINCT 字段名 FROM 表名; # 要数据一模一样才能去重
```

> 注意点: 去重针对某列, distinct前面不能先出现列名

别名查询

```
select 列名 as 别名 ,列名 from 表  #列别名 as可以不写
select 别名.* from 表 as 别名 #表别名
```

运算查询(+,-,*,/,%等)

```
select pname ,price+10 as price from product;
select name,chinese+math+english as total from student
```

> 运算查询字段,字段之间是可以的
>
> 字符串等类型可以做运算查询，但结果没有意义

条件查询

```
select ... from 表 where 条件  # 取出表中的每条数据，满足条件的记录就返回，不满足条件的记录不返回
```

运算符

- 比较运算符

```
# 大于: >
select * from product where price > 3000;

# 小于: <
select * from product where price < 3000;

# 大于等于: >=
select * from product where price >= 3000;

# 小于等于: <=
select * from product where price <= 3000;

# 等于: = 不能用于null判断
select * from product where pid = 1;

# 不等于: != 或 <>
select * from product where pid <> 1;

# 安全等于: <=> 可以用于null值判断
select * from product where pid <=> 1;
```

- 逻辑运算符（建议用单词，可读性来说）

```
# 逻辑与: && 或 and
select * from product where price > 3000 and num > 20;

# 逻辑或: || 或 or
select * from product where price > 3000 or num > 20;

# 逻辑非: ! 或 not
select * from product where price not 3000;
```

- 范围

```
# 区间范围: between ... and ...
select * from product where price between 3000 and 6000;

# 区间范围: not between ... and ...
select * from product where price not between 3000 and 6000;

# 集合范围: in (x,x,x)
select * from product where id in (1,5,7,15);

# 集合范围: not in (x,x,x)
select * from product where id not in (1,5,7,15);
```

- 模糊查询

```
# like 'xxx' 模糊查询是处理字符串的时候进行部分匹配 如果想要表示0~n个字符，用%
select * from product where pname like 'iPho%';
select * from product where pname like '%iPho';
select * from product where pname like '%iPho%';
```

- 特殊的null值处理

```
xx is null
xx is not null
xx <=> null
```

约束

- 即规则,规矩 限制;
- 作用：保证用户插入的数据保存到数据库中是符合规范的

约束	约束关键字	
主键	primary key	非空且唯一，并且一张表只能有一个主键
唯一	unique	唯一，当前列不能出现相同的数据
非空	not null	非空，当前列不能为null
默认	default	如果当前列没有数据，则指定默认数据

- 约束种类
  - not null: 非空 ; eg: username varchar(40) not null username这个列不能有null值
  - unique:唯一约束, 后面的数据不能和前面重复; eg: cardNo char(18) unique; cardNo 列里面不可以有重复数据
  - primary key；主键约束(非空+唯一); 一般用在表的id列上面. 一张表基本上都有id列的, id列作为唯一标识的
    - auto_increment: 自动增长,必须是设置了primary key之后,才可以使用auto_increment
  - id int primary key auto_increment; id不需要我们自己维护了, 插入数据的时候直接插入null, 自动的增长进行填充进去, 避免重复了.

> 1. 先设置了primary key 再能设置auto_increment
> 2. 只有当设置了auto_increment 才可以插入null , 否则插入null会报错
> 3. id列: 设置为int类型, 添加主键约束, 自动增长，或者给id设置为字符串类型,添加主键约束, 不能设置自动增长

排序查询

排序是写在查询的后面，代表把数据查询出来之后再排序

- 单列排序

```
# ASC: 升序，默认值; DESC: 降序
SELECT 字段名 FROM 表名 [WHERE 条件] ORDER BY 字段名 [ASC|DESC];
```

- 组合排序

同时对多个字段进行排序，如果第1个字段相等，则按第2个字段排序，依次类推

```
SELECT 字段名 FROM 表名 WHERE 字段=值 ORDER BY 字段名1 [ASC|DESC], 字段名2 [ASC|DESC];
```

聚合函数

聚合函数通常会和分组查询一起使用，用于统计每组的数据

常用聚合函数列表

聚合函数	作用
max(列名)	求这一列的最大值
min(列名)	求这一列的最小值
avg(列名)	求这一列的平均值
count(列名)	统计这一列有多少条记录
sum(列名)	对这一列求总和

语法

```
SELECT 聚合函数(列名) FROM 表名 [where 条件];
```

> 注意: 聚合函数会忽略空值NULL

我们发现对于NULL的记录不会统计，建议如果统计个数则不要使用有可能为null的列，但如果需要把NULL也统计进去呢？我们可以通过 IFNULL(列名，默认值) 函数来解决这个问题. 如果列不为空，返回这列的值。如果为NULL，则返回默认值

```
SELECT AVG(IFNULL(score,0)) FROM student;
```

分组查询

GROUP BY将分组字段结果中相同内容作为一组，并且返回每组的第一条数据，所以单独分组没什么用处。分组的目的就是为了统计，一般分组会跟聚合函数一起使用

```
SELECT 聚合函数(列名) FROM 表名 [where 条件] GROUP BY 列;
```

- 多字段分组

```
SELECT 字段1,字段2... FROM 表名 [where 条件] GROUP BY 列1,列2;
```

- 分组后筛选 having

```
SELECT 字段1,字段2... FROM 表名 [where 条件] GROUP BY 列 [HAVING 条件];
```

where和having的区别【面试】

子名	作用
where 子句	1) 查询结果分组前，将不符合where条件的行去掉，即在分组前过滤数据，先过滤再分组。 2) where后面不可以使用聚合函数
having 子句	1) having 子句的作用是筛选满足条件的组，即在分组之后过滤数据，即先分组再过滤。 2) having后

分页查询

```
# a 表示的是跳过的数据条数,b 表示的是要查询的数据条数
select ... from .... limit a ,b
select ... from .... limit a
```

示例

```
-- 分页查询
-- limit 关键字是使用在查询的后边，如果有排序的话则使用在排序的后边
-- limit的语法: limit offset,length 其中offset表示跳过多少条数据，length表示查询多少条数据
SELECT * FROM product LIMIT 0,3 -- 查询product表中的前三条数据(0表示跳过0条，3表示查询3条)
```

外键约束

- 概念

在正常项目中很多时候我们必须要进行拆表，将数据分别存放在多张表中，以减少冗余数据。但是拆分出来的表与表之间是有着关联关系的，我们必须得通过一种约束来约定表与表之间的关系，这种约束就是外键约束

- 作用

外键约束是保证一个或两个表之间的参照完整性,外键是构建于一个表的两个字段或是两个表的两个字段之间的参照关系。

是不是听不懂，白话就是a表的主键，在b表里面也存在，并且要保证必须一致，这就是外键约束。