一、条件查询

> 使用 where 子句对表中的数据筛选，符合条件的数据会出现在结果集中

1.语法格式

```
select 字段1,字段2... from 表名 where 条件;
```

2. 条件构成

where 后面支持多种运算符，进行条件的处理

- 比较运算
- 逻辑运算
- 模糊查询
- 范围查询
- 空判断

1.1 比较运算符

- 等于: =
- 大于: >
- 大于等于: >=
- 小于: <
- 小于等于: <=
- 不等于: != 或 <>

例1：查询小乔的年龄

```
select age from students where name='小乔'
```

例2：查询20岁以下的学生

```
select * from students where age<20
```

例3：查询家乡不在北京的学生

```
select * from students where hometown!='北京'
```

练习：

```
1、查询学号是'007'的学生的身份证号
2、查询'1班'以外的学生信息
3、查询年龄大于20的学生的姓名和性别
```

1.2 逻辑运算符

- and

- or
- not

例1：查询年龄小于20的女同学

```
select * from students where age<20 and sex='女'
```

例2：查询女学生或'1班'的学生

```
select * from students where sex='女' or class='1班'
```

例3：查询非天津的学生

```
select * from students where not hometown='天津'
```

练习：

```
1、查询河南或河北的学生
2、查询'1班'的'上海'的学生
3、查询非20岁的学生
```

1.3 模糊查询

- like
- %表示任意多个任意字符
- _表示一个任意字符

例1：查询姓孙的学生

```
select * from students where name like '孙%'
```

例2：查询姓孙且名字是一个字的学生

```
select * from students where name like '孙_'
```

例3：查询姓名以乔结尾的学生

```
select * from students where name like '%乔'
```

例4：查询姓名含白的学生

```
select * from students where name like '%白%'
```

练习：

```
1、查询姓名为两个字的学生
2、查询姓百且年龄大于20的学生
3、查询学号以1结尾的学生
```

1.4 范围查询

- in表示在一个非连续的范围内，格式为 in (...,...)

例1：查询家乡是北京或上海或广东的学生

```
select * from students where hometown in('北京','上海','广东')
```

- between ... and ...表示在一个连续的范围内

例2：查询年龄为18至20的学生

```
select * from students where age between 18 and 20
```

练习：

```
1、查询年龄在18或19或22的女生
2、查询年龄在20到25以外的学生
```

1.5 空判断

- 判空is null

例1：查询没有填写身份证的学生

```
select * from students where card is null
```

- 判非空is not null

例2：查询填写了身份证的学生

```
select * from students where card is not null
```

> 注意：null与' '是不同的

二、排序

> 为了方便查看数据，可以对数据进行排序

1.1 语法格式

```
select * from 表名
order by 列1 asc|desc,列2 asc|desc,...
```

- 默认按照列值从小到大排列
- asc从小到大排列，即升序
- desc从大到小排序，即降序

> 将行数据按照列1进行排序，如果某些行列1的值相同时，则按照列2排序，以此类推

1.2 举例说明

例1：查询所有学生信息，按年龄从小到大排序

```
select * from students order by age
```

例2：查询所有学生信息，按年龄从大到小排序，年龄相同时，再按学号从小到大排序

```
select * from students order by age desc,studentNo
```

练习：

```
1、查询所有学生信息，按班级从小到大排序，班级相同时，再按学号再按学号从小到大排序
```

三、聚合函数

1. 聚合函数

> 对于一组数据进行计算返回单个结果的实现过程

- 使用聚合函数方便进行数据统计
- 聚合函数不能在 where 子句中使用

2.常用聚合函数

- count(): 查询总记录数
- max(): 查询最大值
- min(): 查询最小值
- sum(): 求和
- avg(): 求平均值

2.1 查询总记录数

count(*)表示计算总行数，括号中也可以使用字段名

示例：查询学生总数

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

2.2 查询最大值

max(列)表示求此列的最大值

示例：查询女生的最大年龄

```
select max(age) from students where sex='女';
```

2.3 查询最小值

min(列)表示求此列的最小值

示例：查询1班的最小年龄

```
select min(age) from students;
```

2.4 求和

sum(列)表示求此列的和

示例：查询北京学生的年龄总和

```
select sum(age) from students where hometown='北京';
```

2.5 求平均值

avg(列)表示求此列的平均值

示例：查询女生的平均年龄

```
select avg(age) from students where sex='女'
```

1.6 练习

```
1、查询所有学生的最大年龄、最小年龄、平均年龄
2、一班共有多少个学生
3、查询3班年龄小于18岁的同学有几个
```

四、分组

> 按照字段分组，此字段相同的数据会被放到一个组中
>
> 分组的目的是对每一组的数据进行统计(使用聚合函数)

1. 语法格式

```
select 字段1,字段2,聚合... from 表名 group by 字段1,字段2...
```

2.举例说明

例1：查询各种性别的人数

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

例2：查询每个班级中各种性别的人数

```
select class,sex,count(*) from students group by class,sex
```

练习

```
1、查询各个班级学生的平均年龄、最大年龄、最小年龄
```

3. 分组后的数据筛选

3.1 语法格式

```
select 字段1,字段2,聚合... from 表名
group by 字段1,字段2,字段3...
having 字段1,...聚合...
```

- having后面的条件运算符与where的相同

例1：查询男生总人数

```
方案一
select count(*) from students where sex='男'
-----------------------------------
方案二：
select sex,count(*) from students group by sex having sex='男'
```

练习

```
1、查询1班除外其他班级学生的平均年龄、最大年龄、最小年龄
```

3.2 where 与 having

- where 是对 from 后面指定的表进行数据筛选，属于对原始数据的筛选
- having 是对 group by 的结果进行筛选
- having 后面的条件中可以用聚合函数，where 后面不可以

```
1.查询班级平均年龄大于22岁的班级有哪些
select class from students group by class having avg(age)>22;
```

五、分页

> 当数据量过大时，在一页中查看数据是一件非常麻烦的事情

1. 语法格式

```
select * from 表名 limit start,count
```

- 从start开始，获取count条数据
- start索引从0开始

2. 举例说明

例1：查询前3行学生信息

```
select * from students limit 0,3
```

练习：

```
1、查询第4到第6行学生信息
```

3. 分页格式

limit典型的应用场景就是实现分页查询

已知：每页显示m条数据，求：显示第n页的数据

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

练习：

```
1、每页显示5条数据，显示每一页的数据
```

> 扩展：已知总记录数和每页显示条数，求总页数?

六、连接查询

> 当查询结果的列来源于多张表时，需要将多张表连接成一个大的数据集，再选择合适的列返回

1. 连接方式

- **内连接**：查询的结果为两个表匹配到的数据

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

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

2. 内连接

2.1语法格式

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

2.2举例说明

例1：查询学生信息及学生的成绩

```
select
    *
from
    students stu
inner join scores sc on stu.studentNo = sc.studentNo
```

扩展：内连接的另一种语法

```
select * from 表1,表2 where 表1.列=表2.列

例1：查询学生信息及学生的成绩

select
    *
from
    students stu,
    scores sc
where
    stu.studentNo = sc.studentNo
```

例2：查询课程信息及课程的成绩

```
select
    *
from
    courses cs
inner join scores sc on cs.courseNo = sc.courseNo
```

例3：查询学生信息及学生的课程对应的成绩

```
select
    *
from
    students stu
inner join scores sc on stu.studentNo = sc.studentNo
inner join courses cs on cs.courseNo = sc.courseNo
```

例4：查询王昭君的成绩，要求显示姓名、课程号、成绩

```
select
    stu.name,
    sc.courseNo,
    sc.score
from
    students stu
inner join scores sc on stu.studentNo = sc.studentNo
where
    stu.name = '王昭君'
```

例5：查询王昭君的数据库成绩，要求显示姓名、课程名、成绩

```
select
    stu.name,
    cs.name,
    sc.score
from
    students stu
inner join scores sc on stu.studentNo = sc.studentNo
inner join courses cs on sc.courseNo = cs.courseNo
where
    stu.name = '王昭君' and cs.name = '数据库'
```

例6：查询所有学生的数据库成绩，要求显示姓名、课程名、成绩

```
select
    stu.name,
    cs.name,
    sc.score
from
    students stu
inner join scores sc on stu.studentNo = sc.studentNo
inner join courses cs on sc.courseNo = cs.courseNo
where
    cs.name = '数据库'
```

例7：查询男生中最高成绩，要求显示姓名、课程名、成绩

```
select
    stu.name,
    cs.name,
    sc.score
from
    students stu
inner join scores sc on stu.studentNo = sc.studentNo
inner join courses cs on sc.courseNo = cs.courseNo
where
    stu.sex = '男'
order by
    sc.score desc
limit 1
```

3. 左连接

3.1语法格式

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

3.2举例说明

例1：查询所有学生的成绩，包括没有成绩的学生

```
select
    *
from
    students stu
left join scores sc on stu.studentNo = sc.studentNo
```

例2：查询所有学生的成绩，包括没有成绩的学生，需要显示课程名

```
select
    *
from
    students stu
left join scores sc on stu.studentNo = sc.studentNo
left join courses cs on cs.courseNo = sc.courseNo
```

4. 右连接

4.1语法格式

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

4.2举例说明

例1：查询所有学生的成绩，包括没有成绩的学生

```
select
    *
from
    scores sc
right join students stu on stu.studentNo = sc.studentNo
```

例2：查询所有学生的成绩，包括没有成绩的学生，需要显示课程名

```
select
    *
from
    scores sc
right join students stu on stu.studentNo = sc.studentNo
left join courses cs on cs.courseNo = sc.courseNo
```

七、自关联

1. 案例

- 设计省信息的表结构provinces
    - id
    - ptitle
- 设计市信息的表结构citys
    - id

    - ctitle
    - proid
- citys表的proid表示城市所属的省，对应着provinces表的id值
- 问题：能不能将两个表合成一张表呢?
- 思考：观察两张表发现，citys表比provinces表多一个列proid，其它列的类型都是一样的
- 意义：存储的都是地区信息，而且每种信息的数据量有限，没必要增加一个新表，或者将来还要存储区、乡镇信息，都增加新表的开销太大
- 答案：定义表areas，结构如下
    - id
    - atitle
    - pid
- 因为省没有所属的省份，所以可以填写为null
- 城市所属的省份pid，填写省所对应的编号id
- 这就是自关联，表中的某一列，关联了这个表中的另外一列，但是它们的业务逻辑含义是不一样的，城市信息的pid引用的是省信息的id
- 在这个表中，结构不变，可以添加区县、乡镇街道、村社区等信息

准备数据：

```
drop table if exists areas;
create table areas(
    aid int primary key,
    atitle varchar(20),
    pid int
);

insert into areas values
('130000', '河北省', NULL),
('130100', '石家庄市', '130000'),
('130400', '邯郸市', '130000'),
('130600', '保定市', '130000'),
('130700', '张家口市', '130000'),
('130800', '承德市', '130000'),
('410000', '河南省', NULL),
('410100', '郑州市', '410000'),
('410300', '洛阳市', '410000'),
('410500', '安阳市', '410000'),
('410700', '新乡市', '410000'),
('410800', '焦作市', '410000'),
('410101', '中原区', '410100'),
('410102', '二七区', '410100'),
('410301', '洛龙区', '410300');
```

2. 自关联

自关联：表中的某一列，关联了这个表中的另外一列，但是它们的业务逻辑含义是不一样的。（自己关联自己）

例1：查询河南省所有的市

```
select
    *
from
    areas as a1
inner join areas as a2 on a1.aid=a2.pid
where
    a1.atitle='河南省';
```

例2：查询郑州市的所有的区

```
select
    *
from
    areas as a1
inner join areas as a2 on a1.aid=a2.pid
where
    a1.atitle='郑州市';
```

例3：查询河南省的所有的市区

```
select
    *
from
    areas as a1
inner join areas as a2 on a1.aid=a2.pid
inner join areas as a3 on a2.aid=a3.pid
where
    a1.atitle='河南省'
```

八、子查询

> 子查询：在一个 select 语句中，嵌入了另外一个 select 语句，那么嵌入的 select 语句称之为子查询语句
>
> 主查询：外层的 select 语句称之为主查询语

1. 主查询和子查询的关系

- 子查询是嵌入到主查询中的
- 子查询是辅助主查询的，要么充当条件，要么充当数据源
- 子查询是可以独立使用的语句，是一条完整的 select 语句

2. 子查找充当条件

例1：查询大于平均年龄的学生

```
查询班级学生平均年龄
select avg(age) from students

查询大于平均年龄的学生
select * from students where age > 21.4167

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

例2：查询王昭君的成绩，要求显示成绩

```
学生表中查询王昭君的学号
select studentNo from students where name = '王昭君'

成绩表中根据学号查询成
select * from scores where studentNo = '001'

select * from scores
where studentNo = (select studentNo from students where name = '王昭君')
```

例1和例2中：子查询返回的结果只有一个值(一行一列)，这种称之为标量子查询

例3：查询18岁的学生的成绩，要求显示成绩

```
学生表中查询18岁的学生的学号
select studentNo from students where age=18

成绩表中根据学号查询成绩
select * from scores where studentNo in ('002','006')

select * from scores
where studentNo in (select studentNo from students where age=18)
```

例3中：子查询返回的结果是一列数据(一列多行)，这种称之为列子查询

例4：查询和王昭君同班、同龄的学生信息

```
select class,age from students where name='王昭君'

select * from students where class='1班' and age=20

select * from students where (class,age)=('1班',20)

select * from students
where (class,age)=(select class,age from students where name='王昭君')
```

例4中：子查询返回的结果是一行(一行多列)，这种称之为行子查询

3. 子查询充当数据源

例5：查询数据库和系统测试的课程成绩

```
select
    *
from
    scores s
inner join
    (select * from courses where name in ('数据库','系统测试')) c
on s.courseNo = c.courseNo
```

例5中：子查询返回的结果是多行多列(相当于一个表)，这种称之为表级子查询