数据库高级

1、多表查询

- 多表查询是什么：
  - 当查询的结果来源于多张表时，需要将多张表连接成一个大的数据集，在选择合适的数据返回和显示
- 为什么要做多表查询：
  - 如果将所有数据放在一张表中，会导致数据冗余，浪费过多的存储空间
- 表与表之间的关系：
  - 一对一：
    - 一张表对应另外一张表
  - 一对多：
    - 一张表对应多张表
  - 多对多：
    - 多张表对应多张表
- 笛卡尔积：
  - 当多张表联合查询是，数据会相互组合匹配，数据量等于两个表的数据量的乘积
- 去笛卡尔积：
  - 通过条件筛选合适的数据返回
- 多表查询格式：
  - 格式： select 字段 from 表1 ,表2 where 表1.字段 = 表2.字段
  - 举例：dept 和 person表

```
select * from person ,dept where person.dept_id = dept.did

取别名：
select * from person p ,dept d where p.dept_id = d.did
```

  - 三表查询:

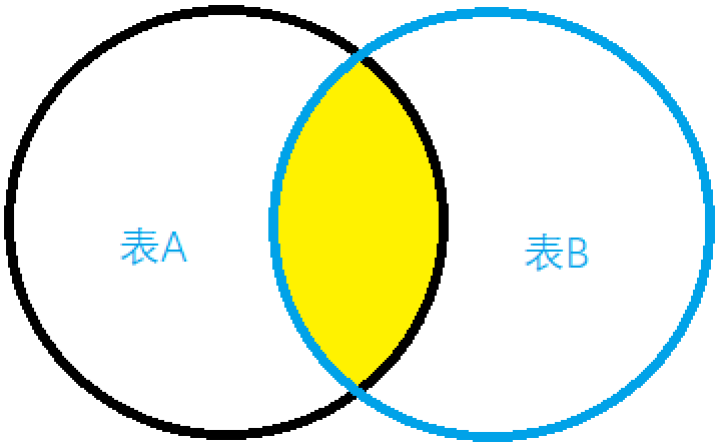
```
查询诸葛亮的数据库成绩（运行英雄表的数据）

SELECT
    students_yx. NAME,
    courses_yx. NAME,
    scores_yx.score
FROM
    students_yx,
    courses_yx,
    scores_yx
WHERE
    students_yx.studentNo = scores_yx.studentno
AND courses_yx.courseNo = scores_yx.courseNo
AND students_yx. NAME = '诸葛亮'
AND courses_yx. NAME = '数据库'
```

2、内连接

按照条件匹配的数据展示

数据展示效果和多表查询基本一样

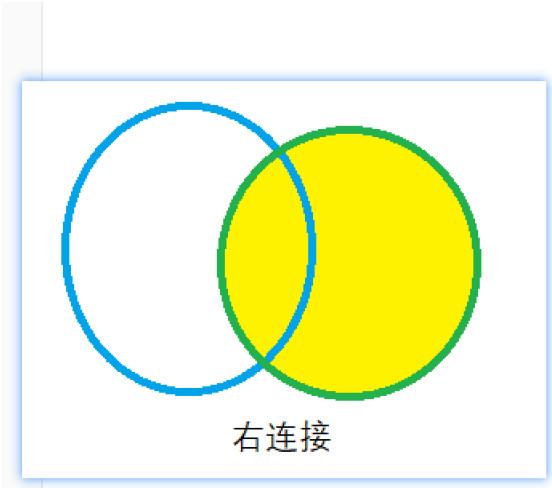


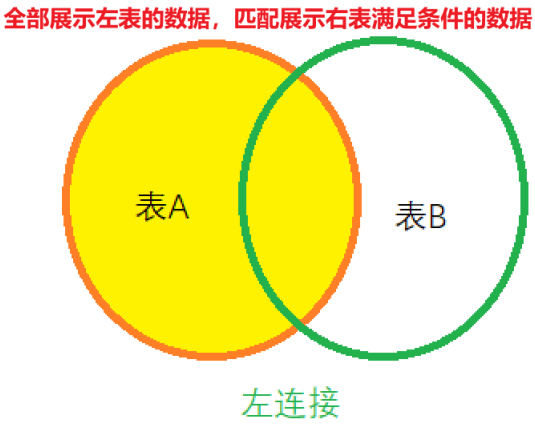
举例：

```
语法格式：
    select * from  表1  INNER JOIN  表2  on  表1.字段=表2.字段

dept 和 person用内连接查询
select * from  dept  INNER JOIN  person  on  dept.did=person.dept_id
```

3、左/右连接





- 举例：

```
select * from 表1 RIGHT JOIN 表2 on 表1.字段=表2.字段    #右连接
select * from 表1 left JOIN 表2 on 表1.字段=表2.字段    #左连接

查询没有部门的人
select * from dept RIGHT JOIN person on  dept.did=person.dept_id  where
dept.did is null

查询没有人的部门
select * from dept left JOIN person on  dept.did=person.dept_id where
person.id is null


查询没有成绩的学生 （英雄表）
SELECT * from students_yx LEFT JOIN scores_yx on students_yx.studentNo =
scores_yx.studentno
WHERE  scores_yx.score is null
```

4、全连接

显示两张表满足条件和未满足条件的数据匹配展示

mysql没有全连接，但是可以使用union来将左右连接合并

orlcal ： 是有全连接

- union :
  - 联合显示的字段必须相同
  - 自动去除重复的数据
- 举例:

```
将dept表和person表的左右连接联合展示

select * from dept RIGHT JOIN person on  dept.did=person.dept_id
union
select * from dept left JOIN person on  dept.did=person.dept_id
```

5、自关联

- 什么是自关联:
  - 同一个表中的一列关联了另外的一列
- 自关联有什么用:
  - 表与表之间存在上下级关系，下级可能是没有尽头的，如果每一级关系都产生一张表的话，那么将会产生无数的表，不便于维护

```
查询郑州市下的所有区
select * from areas_b a , areas_b b where a.aid=b.pid  and a.atitle='郑州市'

查询河南省下的所有城市
select * from  areas_b a , areas_b  b WHERE a.aid=b.pid and a.atitle='河南省'
```

```
查询河南省下的所有城市和地区，展示省，市，区

SELECT
    *
FROM
(areas_b a RIGHT  JOIN areas_b b ON b.aid = a.pid )
RIGHT  JOIN   areas_b c  ON  b.pid = c.aid   where  c.atitle='河南省'
```

6、子查询

主查询：主要查询对象，第一条select 语句

子查询：在第一条select语句中，嵌入了另外一条select语句，嵌入的select语句叫做子查询

子查询可以放置的位置：

- 在where的条件中
- 在having的条件中
- 在select后面，如。 select (1)-(5)

子查询的结果是一个值：

- 子查询返回的结果只有一个值

```
查询大于平均年龄的人
select avg(age),age from dep         #查询平均年龄
select * from dep where age>(select avg(age)  from dep)  #做条件比较
```

  - 练习：

```
查询比 广东技术人员平均工资 大的人    dep

select avg(salary) from dep where job='技术' and address='广东'
SELECT * from dep where salary > (select avg(salary) from dep where
job='技术' and address='广东')



查询平均工资 大于技术平均工资的其他岗位    dep

select avg(salary) from dep where job='技术'    ---技术平均工资
select job,avg(salary) from dep GROUP BY job having avg(salary)>(select
avg(salary) from dep where job='技术')
```

查询结果为一列数据：

子查询的结果为一列数据，至少包含2个数据

any / some： 关键字放在子查询的前面，将比较结果用or进行连接。符号：> < >= <= !<>

all： 关键字放在子查询的前面，将比较结果用and进行连接

```
查找出所有经理的老乡
找出经理的家乡
select address  from dep where job ='经理'

匹配数据：
select * from dep where address = any(select address  from dep where job ='经理')
and  job <> '经理'
等效于
select * from dep where (address = '广东' or address='江西') and  job <> '经理'



查询比湖南人年龄都大的人

查询湖南人的年龄：
    select age from dep where address='湖南'
匹配结果：
    select * from dep  where age > all (select age from dep where address='湖南')
    等效于
    select * from dep  where age > 23 and age>30 and age>33
```

子查询结果为一个表

```
查询所有的经理中，年龄最大的人的信息：

查询出所有的经理：
select * from dep where job ='经理'

求最大的年龄数
select max(age) from (select * from dep where job ='经理') a

求对应岁数的人
SELECT * FROM ( SELECT * FROM dep WHERE job = '经理' ) s
WHERE age = ( SELECT max(age) FROM ( SELECT * FROM dep WHERE job = '经理' ) a )
```

7、存储过程

存储过程可以保存多条sql语句，然后通过调用存储过程来实现重复执行

使用存错过程的场景：测试时需要制造大量的测试数据的时候

存错过程其实就是一组为了完成特定功能等的sql语句集，经过编译后存在数据库中。可以由应用程序调用来执行。

创建存储过程

```
在查询页面敲语句：

create procedure 存储过程名（）          #创建存储过程
comment '存储过程的注释'        #存储过程的注释
BEGIN            #sql语句开始写的位置

sql语句;

end;         #sql结束的位置


在命令界面输入语句：

mysql> delimiter//       #修改语句结束符为//
mysql> create procedure 2242_1()
    -> comment '哈哈'
    -> begin
    -> select * from dep where name like '王%';       #每一个sql语句结尾用;结束
    -> end//     #使用//来运行整个存储过程
Query OK, 0 rows affected            #运行成功

mysql> delimiter;        #执行完整个存储过程后将结束符改回到原来的样子
mysql> call 2242_1();
```

调用：

```
call 存储过程名
```

删除：

```
drop procedure 存储过程名
```

查看存储过程：

```
show procedure status
```

参数

存储过程在调用的时候可以传入参数，以实现不同的效果

- 形式参数：在创建存储过程的时候设定的参数
- 实际参数：在调用存储过程的时候传入的实际参数，传入的参数的数量和位置要和设计的一致

```
create PROCEDURE 2242_2(参数1 数据类型（长度），参数2 数据类型（长度）)
begin
sql语句
end;
```

举例：

```
    create PROCEDURE 2242_4(dname varchar(5),djob  char(10))
    BEGIN
    select * from dep where name like dname  and job = djob  ;
    end;

    call 2242_4('王%','销售')
```

变量

可以在存储过程里面声明变量，然后提供给sql语句使用

书写格式： declare 变量名1 ， 变量名2 数据类型（长度）

```
create PROCEDURE 2242_5()
BEGIN
declare age_1 int(6) ;  #声明一个变量age_1
set  age_1 = 5;     #给变量赋值
select  age_1;      #将变量打印出来

end;
```

- **变量的3中赋值方式**
  - 通过set修改变量的值：

    ```
    格式： set 变量名=值
    ```

  - 给变量一个默认值：

    ```
    格式： declare 变量名1  数据类型（长度） default 默认值


    声明两个变量，变量1为dname  默认值为'王玉'
    变量2 为 age1  给age1赋值为100

    create PROCEDURE 2242_6()
    BEGIN
    declare dname  varchar(5) default '王玉';
    declare age1  int(5);
    set age1 = 100 ;
    select dname ,age1;
    end;
    ```

  - 将查询的结果赋值变量， 结果一个变量必须对应一个值

```
格式：   select 字段1，字段2 into 变量名1,变量名2   from 表  wherer 条件


create PROCEDURE 2242_7()
BEGIN
declare sala , ag int;
select salary,age into sala , ag  from dep where name='王玉';
select sala ,ag ;
end;

call 2242_7
```

8、流程控制

if判断：

```
if 条件  then
    sql语句;
elseif 条件  THEN
    sql语句;
ELSE
    sql语句;
end if ;


举例：

创建一个存储过程，输入员工名称查询他的工资，如果员工表中没有这个人，则返回：查无此人！
create PROCEDURE 2242_9(inname varchar(5))
BEGIN

if (select id from dep where name = inname) THEN
    select salary from dep  where name = inname;
else
    select '查无此人！';
end if ;

end;


call 2242_9('王小玉')



练习2：
编写一个存错过程，可以输入：高工资，中工资和低工资来查询对应的人员
高工资：大于8000
中工资：5000-8000 （包含5000和8000）
低工资：小于5000

create PROCEDURE 2242_11(sal varchar(5))
BEGIN
if  sal = '高工资' THEN
    select * from dep where salary >8000;
```

```
elseif sal = '中工资' THEN
    select * from dep where salary between  5000 and 8000;
elseif sal = '低工资' THEN
    select * from dep where salary < 5000;
ELSE
    select '要输入正确的字符';
end if;
end;

call 2242_11('中工资')
```

9、循环

while循环

当条件满足时，执行循环体

```
循环往dep表里面插入10条数据
create PROCEDURE 2242_12()
BEGIN
declare i int default 0 ;
while i<10 do
    insert into dep values(0,'王小玉',i,'女','湖南','销售',4000);
    set i = i + 1;
end while;
end;

call 2242_12
```

练习：

```
创建一个存储过程，插入100条数据，但是需要跳过第10,20,30,40,50,60,70,80,90条数据
插入语句：
    insert into dep values(0,'王小玉',i,'女','湖南','销售',4000);

create PROCEDURE 2242_14()
BEGIN
declare s int default 1 ;
while s<=100 DO
    if s%10!=0 THEN
        insert into dep values(0,'王小玉',s,'女','湖南','销售',4000);
    end if;
    set s=s+1;
end while;
end;

call 2242_14



创建一个存储过程，插入数据，根据输入的参数插入对应的数据，
```

```
比如调用存储过程是传入3000，则
插入3000条数据。
插入语句：insert into dep values(0,'王小玉',i,'女','湖南','销售',4000);

create PROCEDURE 2242_13(num int)
BEGIN
declare s int default 0 ;
while  s<=num DO
    insert into dep values(0,'王小玉',s,'女','湖南','销售',4000);
    set s=s+1;
end while;
end;

call 2242_13(3000)
```

10、索引

索引是添加在字段上的规则，索引可以提高数据的查询、分组和排序的效率，但是会降低插入，修改和删除数据的效率并且会占用磁盘空间

创建索引

```
create index 索引名 on 表名（字段名）
```

删除索引

```
drop index 索引名 on 表名
```

查询索引

```
show index from 表名
```

11、视图

视图可以保存一条sql查询语句，可以对表的数据进行设置

- 视图上可以再创建视图
- 可以通过视图修改原表数据，但是只能修改视图内的数据
- 可以通过视图向原表插入数据
- 若视图被引用的字段发生了改变，视图会变成无效，如果又改回来视图有重新生效

创建视图 create view 视图名 as select查询语句

删除视图： drop view 视图名

查看视图： show TABLEs ;