目录

MySQL

一、结构说明:

员工信息在 myemployees 表中 演员在 girls 中

二、登录

固定 mysql -h 登录的主机 -P 端口号 -u 用户名 -p 密码(不能空格) mysql -h locahost -P 3306 -u root -pQwer1234

如果连接本机+默认端口可简写 mysql -u root -pQwer1234

三、环境变量

直接复制 mysql 的目录下的 bin 添加到 path 里面

四、基础语句

```
show database                                          //查询所有数据库
use mysql;                                             //进入表
show table;                                            //进入数据库
show table from mysql;                                 //在其他地方查看 my
sql 中的表
select database();                                     //查看当前在哪个表
中
create table stuinfo(id int,name varchar(20));         //创建一个 stuinfo
的表,有 2 个字段,id 为 int 类型,name 为 varchar 类型
desc stuinfo;                                          //查看 stuinfo 中表
的结构
select * from stuinfo                                  //查看表中所有数据
insert into stuinfo (id,name) values(1,"rose");        //添加一条数据
update stuinfo set name="李磊" where id=1;             //修改 id 为 1 的 nam
e 为李磊
delete from stuinfo where id=1;                        //删除 id 为 1 的字段
```

```
select version();                                    //查看 mysql 的版本
号
mysql -V //查看 mysql 版本号 13.exit                 //退出 mysql
```

五、语法规则

不区分大小写,但建议关键字大写,表名列名小写

每条命令最好用分号结尾

每天命令根据需要,非常长,可以换行,以分号结尾

关键字一行,其他一行

注释 单行注释:#注释文字 多行注释:/* */

六、别名(as)

1.使用 as 关键字

```
select 100%98 as 结果
```

2.使用空格

```
select 100%98 结果
```

3.特殊情况,别名中有空格,使用""

```
select last_name "xi ng",first_name "mi ng"
from employees;
```

七.+号的说明

1.如果两个都为数值 就运行加法运算

```
select 90+10 结果为 100
```

2.如果一个是字符,一个是数值,就会试图将字符转换为数值再做加法运算

```
select "10"+90 结果为 100
```

3.如果一个是字符,一个是数值,就会试图将字符转换为数值再做加法运算,如果转换不成功则为 0,再运算

```
select "aaa"+90 结果为 90
```

4.如果一方为 null,不管一方是什么结果都为 null

```
select null+10 结果为 null
```

八.安全等于<=>

相当于=号,但是可以用于所有类型

例:查询工资为 12000 的员工信息

```
select last_name,salary
from employees
where salary <=>12000
```

九、查询

(一)、语法:

```
select 查询的列表
from 来自于(表名)
```

(二)、特点:

查询列表可以是:表中的字段,常亮值,表达式,函数

查询结果是一个虚拟的表格

(三) 、普通查询:

1、询表中单个字段,select 后面跟字段名 例:查询 myemployees 中的姓名字段

```
select last_name
from employees;
```

2、询表中的多个字段,字段一逗号隔开 例:employees 中的姓名,薪资,邮箱

```
select last_name,salary,email
from employees;
```

3、询表中所有的字段

```
select *
from employees;
```

4、去重:在 select 后面加 distinct 关键字 例:查询 employees 中部门编号

```
select distinct department_id
from employees;
```

5、连接字段:使用 concat 函数(str,str,str[]) 例:employees 中姓和名连在一起

```
select concat(last_name,first_name)
from employees;
```

6、判断是否为 null IFNULL(字段,返回值) 例:将 employees 中的奖金率为 null 的替换为 0

```
select ifnull(commission_pct,0) as 奖金率
from employees
```

(四) 、条件查询

1、语法:

```
select 查询列表 from 表名 where 筛选条件;
```

2、分类:

按照条件表达式筛选:

> < = !=(<>) >= <=

按逻辑表达式筛选:

and or not && || !

作用:用来连接条件表达式的

&& and:两个条件都是 true,反之为 false

|| or:只要有一个为 true,结果为 true

! not:如果一个为 true,结果为 false

模糊查询:

like between and in is null

3、条件表达式筛选

案例一:查询工资大于 1200 员工的信息

```
select *
from employees
where salary>1200;
```

案例二:查询部门编号不等于 90 的员工名和部门编号

```
select last_name,department_id
from employees
where department_id != 90;
```

4、按照逻辑表达式筛选

案例一:查询工资在 10000 和 20000 之间的员工名,工资,奖金

```
select last_name,salary,commssion_pct
from employees
where salary >= 10000 || salary<=20000;
```

案例二:查询部门编号不是在 90 和 110 之间的,或者工资高于 15000 的

```
SELECT department_id,salary
FROM employees
WHERE department_id <90 OR department_id >=110 || salary>15000
```

(五) 、模糊查询

1、like 关键字

配通配符使用 %(0 个或多个) _任意单个字符

案例一:查询员工名中包含字符 a 的员工信息 %

```
select *
from employees
where last_name
like '%a%';
```

案例二:查询员工名中第三个字符为 e,第五个字符为 a 的员工名和工资

```
select last_name,salary
from employees
where last_name lik '___e_a%';
```

案例三:特殊:查询员工名中第二个字符为*的员工名 *转义字符* 也可以用**escape "*来说明_为普通字符

```
select last_name
from employees
where last_name like '_\_%'
```

2、between and 关键字

字段 between 几 and 到几

案例一:查询员工编号在 100 到 120 之间的员工信息

以前:

```
select *
from employees
where employee_id>=100
and employees <=120;
```

使用 between band:

```
select *
from employees
where employee_id
between 100 and 120
```

3、in 关键字

字段 in('匹配的值')

案例一:查询员工的工种编号是 IT*PROG.AD*VP.AD_PRES 中的一个员工名和工种编号

原来:

```
select last_name,job_id
from employees
where job_id=IT_PROG or job_id=AD_VP or job_id=AD_PRES
```

使用 in 关键字:

```
select last_name,job_id
from employees
where job_id
in('IT_PROG','AD_VP','AD_PRES');
```

4、is null 关键字

是否为 null 值

案例一:查询奖金率为 null 的值

```
SELECT *
FROM employees
WHERE commission_pct IS NULL;
```

5、is not null 关键字

查询不为 null 值

案例一:查询奖金率不为 null 的值

```
select *
from employees
where commission_prt is not null;
```

(六)、排序查询

1、语法:

order by 排序列表 [升序|降序]

```
select 查询列表 from 表 where 筛选条件 order by 排序列表 [asc|desc]
```

2 、案例

案例一.查询员工信息,要求工资从高到低排序 正序可省略

```
select *
from employees
order by salary desc;
```

案例二:查询部门编号>=90 的员工信息,按入职时间进行排序[加入筛选]

```
SELECT *
FROM employees
WHERE department_id >=90
ORDER BY hiredate;
```

案例三:按年薪的高低显示员工的信息和年薪[按表达式排序] 三种都能达到效果

```
SELECT *,salary*12*(1+IFNULL(commission_pct,0)) AS 年薪
FROM employees ORDER BY salary;

SELECT *,salary AS 年薪
FROM employees
ORDER BY salary*12*(1+IFNULL(commission_pct,0));

SELECT *,salary*12*(1+IFNULL(commission_pct,0)) AS 年薪
FROM employees
ORDER BY 年薪;
```

案例四:按姓名的长度显示员工的姓名和工资 使用 length(字段)返回长度

```
SELECT *
FROM employees
ORDER BY LENGTH(last_name);
```

案例五:查询员工信息,要求先按工资升序,再按员工编号降序

```
select *
from employees
order by salary,employee_id desc;
```

(七)、分组查询

1、语法:

```
select 分组函数,列(要求出现在group by后面)) from 表 [where 筛选条件]
 group by 分组列表 [order by排序]
```

2、按筛选条件分组

案例一:查询每个工种的最高工资

```
SELECT MAX(salary),job_id
FROM employees
GROUP BY job_id;
```

案例二:查询每个位置上的部分个数

```
SELECT COUNT(`department_id`),`location_id`
FROM departments
GROUP BY `location_id`
```

案例三:添加筛选条件 例:查询邮箱中包含 a 字符的,每个部门的平均工资

```
SELECT department_id,AVG(salary)
FROM employees
WHERE email LIKE '%a%'
GROUP BY department_id;
```

案例四:查询有奖金的每个领导手下员工的最高工资

```
SELECT manager_id,MAX(salary)
FROM employees
WHERE `commission_pct` IS NOT NULL
GROUP BY manager_id
```

3、having 的使用

案例:查询哪个部门的员工个数大于

按照使用到 having():将 having 前的的结果进行筛选

```
    SELECT COUNT(*),department_id,employee_id
    FROM employees
    GROUP BY department_id HAVING COUNT(*)>2
```

4、按照表达式或函数分组

案例:按员工姓名的长度分组,查询每一组的员工个数,筛选员工个数>5 的有哪些

```
SELECT LENGTH(last_name) AS las_name,COUNT(*) AS coun
FROM employees
GROUP BY las_name HAVING coun>5
```

5、按多个字段分组

案例:查询每个部门每个工种的平均工资

```
SELECT AVG(salary),department_id,job_id
FROM employees
GROUP BY department_id,job_id
```

6、添加排序

案例:查询每个部门每个工种的平均工资,按平均工资排序

```
SELECT AVG(salary),department_id,job_id
FROM employees
GROUP BY department_id,job_id order by avg(salary)
```

(八)、连接查询

1、含义:

又称多表查询,当查询的字段来自于多个表时,就会用到连接查询

2 、笛卡尔乘积现象:

表 1 有 m 行 表 2 有 n 行 结果=m*n 行

3 、分类

按年代分类:

sql92 标准:仅支持内连接

sql99 标准:支持内连接+外连接(左外+右外)+交叉连接

按功能分类:

内连接: 等值连接:就两个表中相同的字段

非等值连接:拿第一个表和第二个表的字段比大小

自连接:把 1 张表当成 2 张表使用

外连接:应用于查询一个表有,一个表没有的记录,拿第一个表去匹配第二个表,成功显示,不成功表 2 返回 null

右外连接:left 左边的表为主表 左外连接:right 右边的为主表

左外和右外只是表换一下顺序,left 改为 right

全外连接 交叉连接:使用 sql99 方式实现的笛卡尔乘积

3、注意:

当 2 个表中都有相同的字段,需要加表名.字段 但表名一般都很长,所有一般情况下给表起别名 注意的是如果给表起了别名,原始的查询就不能使用原来的表名

4 、sql92 标准

(1)、等值连接

1.1 、普通的等值连接

案例一.查询女演员对应的男演员

```
SELECT NAME,boyName
FROM beauty,boys
WHERE beauty.boyfriend_id = boys.id
```

案例二.查询员工名和对应的部门名

```
SELECT last_name,department_name
FROM employees,departments
WHERE employees.`department_id` = departments.`department_id`;
```

案例三:查询员工名,工种名,工种号

```
SELECT last_name,job_title,employees.`job_id`
FROM employees,jobs
WHERE employees.`job_id` = jobs.`job_id`
```

2.2 、添加筛选

案例四:查询有奖金的员工名,部门名

```
SELECT last_name,department_name
FROM employees AS e,departments AS d
WHERE e.`department_id` = d.`department_id`
AND commission_pct IS NOT NULL;
```

案例五:查询城市名中第二个字符为 o 的部门名和城市名

```
SELECT city,department_name
FROM departments AS d,locations AS l
WHERE d.`location_id` = l.`location_id`
AND city LIKE '_o%'
```

3.3 、添加分组

案例一.查询每个城市的部门个数

```
SELECT COUNT(department_name),city
FROM departments d,locations l
where d.location_id = l.location_id GROUP BY city
```

案例二:查询有奖金的每个部门的部门名的领导编号和该部门的最低工资

```
SELECT d.department_name,d.`manager_id`,MIN(salary)
FROM employees AS e,departments AS d
WHERE e.`department_id` = d.`department_id`
AND e.`commission_pct` IS NOT NULL
GROUP BY d.department_id
```

4.4 、添加排序

案例一:查询每个工种的工种名和员工的个数,并且按员工个数降序

```
SELECT j.job_title,COUNT(employee_id)
FROM employees AS e,jobs AS j
WHERE e.`job_id` = j.`job_id`
GROUP BY job_title
ORDER BY COUNT(employee_id) DESC
```

5.5、三表连接

案例一:查询员工名.部门名.所在城市

```
SELECT last_name,department_name,city
FROM employees AS e,departments AS d,locations AS l
WHERE e.`department_id` = d.`department_id`
AND d.`location_id` = l.`location_id`
```

(2)、非等值连接

案例一:查询员工的工资和工资级别

```
SELECT salary,grade_level
FROM employees AS e ,job_grades AS g
WHERE salary BETWEEN g.`lowest_sal`
AND g.`highest_sal`
```

(3)、自连接

一张表当 2 张表用

案例一:查询员工名和上级的名称

```
SELECT a1.`employee_id`,a1.`last_name`,a1.`manager_id`,a2.`last_n
ame`
FROM employees AS a1,employees AS a2
WHERE a1.`manager_id` = a2.`employee_id`
```

5、sql92 标准

语法:

```
select 查询列表 from 表 1 别名 [连接类型] join 表 2 别名 on 连接条件 wh
ere 筛选条件 连接类型:内链(inner) 左外(left) 右外(right) 全外(full)
交叉(cross)
```

(1) 、等值连接

案例一:查询员工名,部门名(普通连接)

```
SELECT d.department_name,last_name
FROM employees AS e
INNER JOIN departments AS d
ON e.`department_id` = d.department_id
```

案例二:查询名字包含 e 的员工名和工种名(添加筛选)

```
SELECT last_name,job_title
FROM employees e
INNER JOIN jobs AS j ON e.`job_id` = j.`job_id`
WHERE last_name LIKE '%e%'
```

案例三:查询部门个数>3 的城市名和部门个数(添加分组)

```
SELECT l.`city`,COUNT(*)
FROM locations l
INNER JOIN departments d
ON d.`location_id` = l.`location_id`
GROUP BY l.city HAVING COUNT(*)>3
```

案例四:查询哪个部门员工个数>3 的员工部门名和员工个数并按个数降序(添加排序)

```
SELECT COUNT(*),department_name
FROM employees e
INNER JOIN departments d
ON d.`department_id` = e.`department_id`
GROUP BY d.`department_id` HAVING COUNT(*)>3 ORDER BY COUNT(*) DES
C
```

案例五:查询员工名,部门名,工种名,并按部门名降序(三表查询)

```
SELECT last_name,d.`department_name`,job_title
FROM employees e
INNER JOIN departments d
ON e.`department_id` = d.`department_id`
INNER JOIN jobs j ON j.`job_id` = e.`job_id`
ORDER BY d.`department_name` DESC
```

(2) 、非等值连接

案例一:查询员工的工资级别

```
SELECT e.`salary`,j.`grade_level`
FROM employees e
JOIN job_grades j ON e.`salary`
BETWEEN j.`lowest_sal` AND j.`highest_sal`
```

案例二:查询员工工资级别,每个工资级别的个数,并且按工资级别降序

```
SELECT COUNT(*),j.`grade_level`
FROM employees e
JOIN job_grades j ON e.`salary`
BETWEEN j.`lowest_sal` AND j.`highest_sal`
GROUP BY j.`grade_level` HAVING COUNT(*)>20 ORDER BY j.`grade_leve
l` DESC
```

(3) 、自连接

案例一:查询员工的名字、上级的名字

```
SELECT e.`last_name` '员工姓名',m.`manager_id` '上级 id',m.`last_na
me` '上级名字'
FROM employees e
JOIN employees m ON e.`manager_id` = m.`employee_id`
```

案例二:查询员工字符包含 k 的员工名字、上级的名字

```
SELECT e.`last_name` '员工姓名',m.`manager_id` '上级 id',m.`last_na
me` '上级名字'
FROM employees e
JOIN employees m ON e.`manager_id` = m.`employee_id`
WHERE e.`last_name` LIKE '%k%'
```

(4) 、左外连接

案例一:查询没有男朋友的女演员

```
SELECT b.name
FROM beauty b
LEFT OUTER JOIN boys bo ON b.`boyfriend_id` = bo.`id`
WHERE bo.`id` IS NULL
```

(5) 、右外连接

案例一:查询没有男朋友的女演员

```
SELECT b.name
FROM boys bo right
OUTER JOIN beauty b ON b.`boyfriend_id` = bo.`id`
WHERE bo.`id` IS NULL
```

案例二:哪个部门没有员工

```
SELECT d.department_name
FROM departments d
LEFT OUTER JOIN employees e ON d.`department_id` = e.`department_i
d`
WHERE e.`last_name` IS NULL;
```

(6) 、交叉连接

案例一:笛卡尔乘积

```
select b.*,bo.* from beauty b cross join boys bo
```

(九)、子查询

1、含义:

出现在其他语句中的 select 语句,称为子查询或内查询 外部的查询语句,成为主查询或外查询

2 、分类:

按子查询出现的位置:

select 后面 仅仅支持标量子查询 from 后面 仅支持表子查询

where 或 having 后面 支持标量子查询,列子查询,行子查询

exists 后面 表子查询 按结果集的行列数不通

- 标量字查询(一行一列)
- 列子查询(一列多行)
- 行子查询(一行多列)
- 表子查询(多列多行)

3、where 和 having 后面

特点:

①子查询放在小括号内

②子查询一般放在条件右侧

③标量子查询一般配合单行操作符使用 > < >= <= <> =

④列子查询:一般搭配多行操作符使用 in any all in:in(列表)

**是否等于列表中的其中一个值 any:any(列表)

** 是否大于列表中中的某一个值 all:all(列表)

** 是否大于列表中的全部值

3.1、标量子查询(单行子查询)

案例一:谁的工资比 Abel 高

```
SELECT *
FROM employees e
WHERE salary>( SELECT salary FROM employees WHERE last_name = "Abe
l");
```

案例二:返回 job*id 与 141 员工相同,salary 比 143 号员工多的员工名字,job*id,工资

```
SELECT last_name,job_id,salary
FROM employees
WHERE job_id = (
      SELECT job_id
      FROM employees
      WHERE employee_id = 141)
AND salary>(
       SELECT salary
       FROM employees
       WHERE employee_id = 143
)
```

案例三:查询公司工资最少的员工的 last*name,job*id 和 salary

```
SELECT last_name,job_id,salary
FROM employees
WHERE salary = (
   SELECT MIN(salary) FROM employees
)
```

3.2、列子查询(多行子查询)

案例一:查询 locaion_id 为 1400 或 1700 的部门中所有员工姓名

```
SELECT last_name
FROM employees
WHERE department_id
```

```
IN(
	SELECT department_id
	FROM departments
	WHERE location_id IN(1400,1700)
)
```

案例二:查询其他员工 salary 小于 job*id* *为 IT*PROG 员工的员工名,job_id,salary

```
SELECT last_name,job_id,salary
FROM employees
WHERE salary < ANY (
	SELECT salary
	FROM employees
	WHERE job_id = 'IT_PROG'
)
AND job_id <>'IT_PROG'
```

案例三:查询其他工种比 job*id* *为 IT*PROG'工种部门所有工资最低的员工的员工号,姓名,job_id,salary

```
SELECT employee_id,last_name,job_id,salary
FROM employees
WHERE salary<ALL (
	SELECT salary FROM employees WHERE job_id = 'IT_PROG'
)
```

4、放在 having 后面

4.1、标量子查询

案例一:查询最低工资大于 50 号部门最低工资的部门 id 和其他最低工资

```
SELECT department_id,salary
FROM employees GROUP BY department_id
HAVING MIN(salary)>( SELECT MIN(salary)
FROM employees
WHERE department_id = 50 )
```

4.2、行子查询(一行多列) 要求 2 个值都等于

案例一:查询员工编号最小并且工资最高的员工信息

```
SELECT *
FROM employees
WHERE (employee_id,salary)=( SELECT MIN(employee_id),MAX(salary)
FROM employees )
```

5、放在 select 后面

案例一.查询每个部门的员工个数

```
SELECT d.*,(
        SELECT COUNT(*)
        FROM employees e
       WHERE e.department_id = d.`department_id` )
FROM departments d
```

案例二:查询员工号为 102 的部门名

```
SELECT (
       SELECT d.`department_name`
       FROM employees e JOIN departments d
       WHERE e.department_id = d.`department_id`
       AND e.employee_id = 102 )
```

6、放在 from 后面

把查询后的结果当成一张表,必须起别名

```
SELECT ag.par,grade_level
FROM(
       SELECT department_id AS par,AVG(salary) AS sal
       FROM employees e
       GROUP BY department_id )AS ag
JOIN job_grades j ON ag.sal
BETWEEN lowest_sal AND j.`highest_sal`
```

(十) 、exists 简单使用

案例一:查询 employees 表中有没有 employee_id 字段

```
SELECT EXISTS(SELECT employee_id FROM employees)
```

案例二:查询有员工的部门名

```
SELECT department_name
FROM departments d
```

```
WHERE EXISTS( SELECT * FROM employees e WHERE e.`department_id` =
d.`department_id` )
```

(十一) 、分页查询

场景:当要显示的数据,一页显示补全,需要分页提交 sql 请求

语法: select 查询列表 from 表 join 表 2 on 连接条件 where 筛选条件 group by 分组 having 分组后的筛选条件 order by 排序字段 limit offset,size; offset(起始索引,从 0 开始) size(要显示的条目个数)

案例一:查询前五条的员工信息

```
SELECT *
FROM employees
LIMIT 0,5
```

案例二:查询第 11 条到 25 条

```
SELECT *
FROM employees
LIMIT 10,15
```

案例三:有奖金的员工信息,并且工资较高的前 10 名显示出来

```
SELECT *
FROM employees
WHERE commission_pct IS NOT NULL
ORDER BY salary DESC LIMIT 10
```

(十二) 、联合查询

说明:

又名联合、合并 将多条查询语句的结果合并成一个结果

应用场景:

查询的结果来自多个表,但是多个表中没有直接的关系

注意点:

多个表查询的字段必须相同,查询结果顺序推荐一致 比如说: 姓名 性别 姓名 性别 姓名 性别 性别 姓名

联合查询自动去重,如果不想去重,union 后面追加 all

(1)简单使用

案例一:查询部门编号>90 或邮箱包含 a 的员工信息

以前写法:

```
select *
from employees
where email lik "%a%"
and department_id>90
```

联合查询:

```
SELECT * FROM employees
WHERE email LIKE '%a%'
union
   select *
   from employees
   where department_id >90
```

十、函数

(一)、概念:类似于 java 的方法

(二) 、字符函数

1、连接字段

concat(字段 1,字段二):

例:employees 中姓和名连在一起

```
select concat(last_name,first_name)
from employees;
```

2、替换为空

null IFNULL(字段,返回值):

判断是否为 0,如果为 0 返回返回值

例:将 employees 中的奖金率为 null 的替换为 0

```
select ifnull(commission_pct,0) as 奖金率 from employees
```

3、长度

length(字段):

返回字节长度 例:返回 join 的字节数

```
select length("join")
```

4、大写

upper()

转换为大写 例:拼接姓和名,姓为大写,名为小写

```
SELECT CONCAT(UPPER(first_name),LOWER(last_name)) FROM employees
```

5、小写

lower()

转换为小写 例:将 ABC 转换为小写

```
select lower("ABC");
```

6、截取字符串

substr()

截取字符串

2 个重载方法(pos)从哪里开始

substr(str,pos) pos(从哪里开始) 索引从 1 开始 包括 5 select substr ("今天天气好晴朗",5); substr(str,start,end) 从哪里开始,从哪里结束 包前包后 select substr("今天天气好晴朗",1,3);

案例:姓名中首字母大写,其他字符小写,然后用_拼接,显示出来

```
SELECT CONCAT(UPPER(SUBSTR(last_name,1,1)),'_',LOWER(SUBSTR(last_
name,2))) FROM employees
```

7、返回字符串位置

instr(str1,str2)

返回 str2 在 str1 中第一次的索引,没有则返回 0,

```
select instr("今天天气好晴朗","好晴朗");
```

8、去除空格

trim()

去除空格,不显示空格

```
select trim(" 好晴朗 ");
```

9、去除字符串

trim(str1 from str2)

在 str2 中去除 str1

```
select trim(a from "aaaaaaa 今天天气好晴朗 aaaa");
```

10、截取前面

lpad(str,int,"str2")

指定一个字符串长度为 int,如果不够用 str2 在左边填充,如果够截取 str1 最前面的

```
select lpad("好天气",9,"a");
```

11、截取后面

rpad(str,int,str2)

指定一个字符串长度为 int,如果不够用 str2 在右边边填充,如果够截取 str1 最钱面的

```
select rpad("好天气",9,"a");
```

12、替换

replace(str,str1,str2)

将 str 中的 str1 替换为 str2,不管有多少个都更改

```
select replace("今天天气好晴朗","好天气","坏天气");
```

(三)、数学函数

1、四舍五入

round()

```
select round(-1.55) //返回-2 select round(1.567,2) //返回 1.57 保留 2 位
```

2、向上取整

ceil()

```
select ceil(1.02) //返回 2 -1.02 返回-1
```

3、向下取整

floor()

```
select floor(9.9) //返回 9 -9.9 返回-10
```

4、保留几位小数

truncate()

```
select truncate(1.65,1); //返回 1.6 不考虑四舍五入
```

5、取模

mod()

```
select mod(10,3) //返回 1
```

(四)、日期函数

1、当前系统日期+时间

now()

```
select now();
```

2、当前系统日期,不包含时间

curdate()

```
select curdate();
```

3、返回当前的时间,不包括日期

curtime()

```
select curtime();
```

4、截取时间中的年

year(now())

```
select year(now());
```

5、截取时间中的月

month()

```
select monthname(now());
```

也可以获取年,月,日,小时,分钟,秒 select+英文

6、将日期转换为一个指定格式的日期

str*to*date()

```
%Y 年 %y2 为的年份 %m 月(01,01...) %c 月份(1,2,3) %d 日 %H 小时(24 小时)
 %h(12 小时) %i 分钟 %s 秒
```

例子:如果不一样的日期如,4-3 1993 的话转换为正常的

```
select str_to_date("4-3 1992",%c-%d %y)
```

7、将日期转换为字符

date_format()

```
select date_format(上述转换的日期)
```

8、日期的加减,日期 1-日期 2

datediff(日期 1,日期 2)

```
select datediff('2017-10-5','2017-10-3');
```

(五) 、其他函数

1、版本号

Version()

```
select version();
```

2、查看当前所有数据库

database()

```
select database();
```

3、查看当前用户

user()

```
select user();
```

(六)、流程控制函数

1、if(条件,obj1,obj2):

如果为 true 返回 obj1,如果 false 返回 obj2

```
select if(10>5,10,5);
```

例子:获取没有奖金的用户

```
select last_name,commission_pct,if(commission_pct is null,"哈哈","
嘎嘎") from employees
```

2、case():

使用一:

格式:

```
    case 条件
    when 常量 1 then 要显示的内容;
    when 常量 2 then 要显示的内容;
    ...
    else 默认值
    end
```

案例:查询员工的工资,要求部门号为 30 工资乘以 1.1 40 1.2 50 1.3 其他源工资

```
        SELECT salary,department_id,
                CASE department_id
                WHEN 30 THEN salary*1.1
                WHEN 40 THEN salary*1.2
                WHEN 50 THEN salary*1.3
                ELSE salary
                END
        FROM employees;
```

使用二:

格式:

```
       case
       when 条件 1 then 显示的值 1 //条件 1 为 true 的情况下
       when 条件 2 then 显示的值 2
```

```
...
else 默认值
end
```

案例:查询员工的工资>20000 显示 A >15000 显示 B >10000 显示 C 其他为 c

```
SELECT salary
    CASE
    WHEN salary>20000 THEN 'A'
    WHEN salary>15000 THEN 'B'
    WHEN salary>10000 THEN 'C'
    ELSE 'D'
    END
FROM employees
```

(七) 、分组函数

功能:做统计使用,又称为统计函数又称为聚合函数或统计函数或组函数

分类:

sum(求和) avg(平均值) max(最大值) min(最小值) count(计算个数)

1、求和,忽略 null 值

sum(int 集合)

案例:统计工资之和

```
select sum(slary)
from employees;
```

2、平均值,忽略 null 值

avg(int 集合)

案例:统计工资平均值

```
select avg(slary)
from employees;
```

3、最大值,忽略 null 值

max(obj 集合)

案例:统计工资最大值

```
select max(slary)
from employees;
```

4、最小值,忽略 null 值

min(obj 集合)

案例:统计工资最小值

```
select min(salary)
from employees;
```

5、个数,忽略 null 值

count()

count(*)和 count(1)比 count(字段)效率高

方式一:count(字段) 案例:统计工资个数

```
   select count(slary) from employees;
```

方式二:count(*) 案例:所有行数

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

方式三:count(1) 案例:统计多少行数

```
   select count(1) from employees;
```

可以和 distinct(去重)使用

案例:计算去重后的工资和

```
select sum(distinct(salary)),sum(salary)
from employees;
```

十一、插入

(一)、语法一:

优点:可以插入多行,添加子查询

```
insert into 表明(列名,...) values (值 1,...)
```

案例一:在 beauty 表中插入一条数据(普通插入)

```
INSERT INTO beauty (id,NAME,sex,borndate,phone,photo,boyfriend_i
d)
VALUES (13,'玛吉斯','男','1999-09-09','18595926383',NULL,6)
```

案例二:插入多行数据

```
INSERT INTO beauty (id,NAME,sex,borndate,phone,photo,boyfriend_i
d)
VALUES (16,'玛吉斯','男','1999-09-09','18595926383',NULL,6),
(17,'玛吉斯','男','1999-09-09','18595926383',NULL,6),
(18,'玛吉斯','男','1999-09-09','18595926383',NULL,6);
```

案例三:支持子查询,将查到的内容添加到表中

```
INSERT INTO beauty(id,NAME,phone)
SELECT 19,'111','111'
```

(二)、语法二:

```
insert into 表明 set 列名=值,列名=值
```

案例一:在 beauty 表中插入一条数据

```
INSERT INTO beauty SET id=14,NAME='娃哈哈',sex='男',phone='123'
```

说明:

列名的个数和值必须相等 不为 null 的必须添加数据 列名可以省略,默认为所有列,顺序也不能更改

十二、修改

(一) 、修改单表记录

```
updata 表明 set 列=新值,列=新值,列=新值 where 筛选条件;
```

案例一:修改 beauty 中 name 娃哈哈为嘻嘻嘻

```
UPDATE beauty SET NAME='嘻嘻嘻'
WHERE NAME='娃哈哈'
```

案例二:修改 beauty 中 name 为 111 的改为 222,电话号 111 的改为 222

```
UPDATE beauty SET NAME='222',phone='222'
WHERE NAME='111' AND phone='111'
```

(二) 、修改多表记录

92 语法

```
    update 表 1 别名,表 2 别名 set 列=值 where 连接条件 and 筛选条件
```

99 语法

```
    update 表 1 别名 join 表 2 别名 on 连接条件 set 列=值 where 筛选条
件
```

案例一:修改张无忌的女朋友的手机号为 114

```
UPDATE boys AS bo JOIN beauty AS b ON bo.`id`=b.`boyfriend_id` SET
 phone='114' WHERE bo.`boyName`='张无忌'
```

十三、删除

```
delete from 表 where 筛选条件
```

(一) 、delete 关键字删除

1、单表的删除

案例一:删除 beauty 中手机编号最后为 9 的信息

```
DELETE FROM beauty WHERE phone LIKE '%9'
```

2、多表删除(支持 92,99 语法)

案例一:删除张无忌女朋友的信息(删除 beauty 中的数据)

```
DELETE b FROM beauty b JOIN boys bo ON bo.`id`=b.`boyfriend_id` WH
EREbo.`boyName`='张无忌'
```

案例二:删除黄晓明的信息以及他女朋友的信息

```
DELETE b,bo FROM beauty b JOIN boys bo ON bo.`id`=b.`boyfriend_id`
 WHERE bo.`boyName`='黄晓明'
```

(二) 、truncate 关键字删除

删除 boys 表

```
truncate table boys;
```

十四、库和表的管理

(一)、库的管理

1、创建

```
创建数据库
create database 库名
有则创建,无则不动
create database if not exists books
```

案例一:创建一个 books 库

```
create database books
```

案例二:创建一个 books 库 如果没则创建,如果有则不动

```
create database if not exists books
```

2、修改库

修改库的字符集

```
alter database books character set gbk;
```

3、删除库

```
drop database if exists books; (有则删除)
```

(二) 、表的管理

1、创建

```
create table 表明(
        列名 列的类型[(长度)约束],
        列名 列的类型[(长度)约束],
        .....
        列名 列的类型[(长度)约束]
);
```

案例一:创建 book 表

```
CREATE TABLE book( id INT,#编号 bName VARCHAR(20),#图书名 price DOU
BLE,#价格 authorID int,#作者编号 publishDate DATETIME#出版日期 )
```

3、删除表

```
drop table 表名
```

2、修改

2.1、修改列名

```
alter table 表名 changge column 旧列名 新列名 新列名 类型
```

案例一:将 publishdate 修改为 pubDate

```
alter table book change column publishdate pubdate datetime
```

2.2、修改列的类型

```
alter table 表名 change column 列名 新类型
```

案例一:修改 pubdate 的类型为 timestamp

```
alter table book change column pubdate timestamp
```

2.3、添加新列

```
alter table 表名 add column 添加的列名 类型
```

案例一:在 book 表中添加年薪

```
alter table book ad column annual double
```

2.4、删除列

```
alter table 表名 drop column 列名
```

案例一:删除 book 中的 annual 列

```
alter table book drop column annual
```

2.4、修改表名

```
alter table 表名 rename to 新表名
```

案例一:将 book 表名修改为 bookss

```
alter table book rename to bookss
```

4、表的复制

4.1、仅仅复制表结构

```
create table 新表名 like 旧表名
```

案例一:复制 boos 表,表名为 copy

```
create table copy like boos
```

4.2、复制某些字段

案例一:仅复制 id 和 bNmae 字段 create table copy2 select id,bName from boos where 0

4.3、复制表结构+全部数据

```
create table 新表名 select * from 旧表名 (可加筛选条件复制单个数据)
```

案例一:复制 boos 表和它的数据

```
create table copy3 select * from boos
```

4.4、复制表结构+某些数据

案例一:仅复制 id 为 1 的数据 create table copy2 select * from boos where id = 1

十五、表的数据类型

(一) 、数值型

1、整型

```
Tinyint -128~127
Smallint -32768~32767
Mediumint -8388608~8388607
int、integer 大
Bigint 更大
```

特点:

1.创建表设置类型的时候默认是有符号的(有符号支持负数) 如果想无符号,在类型后面加 unsigned create table book(id int unsigned)

2.如果插入的数据超出范围,会报 out of range 异常, 然后在表中插入临界值

3.如果不设置长度,会有默认的长度,长度代表了显示的最大宽度,如果不够会用 0 填充,但是需要在创建表的时候在类型后面 添加 zerofill 关键字

2、小数

M 和 D 的意思:

M:小数点前+小数点后一共多少位

D:小数点后面保留几位小数

M 和 D 都可以省略,如果省略的话 float 和 double 都没有限制,但是 dec 默认的是(10,0)

精度较高使用 dec,货币

2.1、定点数

dec(m,d) 和 double 的范围相同

2.2、浮点数

float(m,d) double(m,d)

(二) 、字符型

1、短的文本

说明: m 最大的字符数,char(m)可以省略,默认为 1 varchar(m) m 不能省略

char(m):固定长度的字符 耗费空间 效率高

varchar(m):可变长度字符 节省空间 效率低 一个字符的使用 char NEUM(a,b) 该类型的字段只能插入 a 或者 b set(a,b) 该类型可以插入多个值 a b a,b

2、较长的文本

test blob(大的二进制)

(三)、日期型

date:保存日期,不带时间 1999-9-9

datetime:带时间 1999-9-9 00:00:00

timestamp:跟 datetime 差不多,时间戳 time:只有时间 year:只有年

十六、约束

(一) 、含义:

一种限制,用于限制表中的数据,为了保证表中数据的准确和可靠性

(二) 、语法:

create table 表名(字段名 字段类型 约束)

(三) 、说明

1.非空和默认写在字段后面

2.外键写在表约束中

3.一个字段可以添加多个约束

(四) 、六大约束:

not null:非空,用于保证该字段的值不能为空

default:默认,用于该字段有默认值

primary key:主键,用于保证字段的唯一性,并且费控

unique:唯一,用于保证该字段的值具有唯一性,可以为空

foreign key:外键:在从表中使用,外键用来引用主表的值

check:检查,mysql 不支持

(五) 、添加约束的时间

1.创建表时

2.修改表时

(六)、约束的添加分类

列级约束:在列后面添加的,六大约束都支持,但外键约束没有效果

表级约束:字段写完后在所有字段写完后写,除了非空和默认都支持

(七) 、创建表时添加列级约束

1、创建并进入测试数据库

create database student; use student;

1.1、创建一个外键需要连接的表

```
CREATE TABLE major(
   id INT PRIMARY KEY,
   majorNmae VARCHAR(20) )
```

1.2、开始使用约束

```
CREATE TABLE stuinfo(
   id INT PRIMARY KEY,#主键约束
   stuName VARCHAR(20) NOT NULL,#非空约束
   sex CHAR(1) CHECK(sex='男' OR sex='女'),#检查约束(没效果)
   seat INT UNIQUE,#唯一约束
   age INT DEFAULT 18,#默认约束
   majorId INT REFERENCES major(id) #外键约束(没效果)
)
```

2、创建表时添加表时约束

2.1、添加列级约束

```
alter table 表名 modify column 字段名 类型 约束
```

2.2、添加表级约束

```
alter table 表名 add [constraint 约束名] 约束类型(字段名)
```

constraint 可以省略,默认字段名

```
CREATE TABLE stuinfos(
   id INT, stuName VARCHAR(20),
   sex CHAR(1), seat INT,
   age INT,
   majorId INT,
   CONSTRAINT pk PRIMARY KEY(id),#主键
   CONSTRAINT uq UNIQUE(seat),#唯一键
   CONSTRAINT ck CHECK(sex='男' OR sex ='女'),#检查
```

```
    CONSTRAINT fk_stuinfos_major FOREIGN KEY(majorid) REFERENCES m
ajor(id)#外键 )

2.3、SHOW INDEX FROM stuinfos;      //查看键的信息
```

3、修改表时添加约束

```
创建一个表,什么都不约束
    CREATE TABLE students(
        id INT,
        stuName VARCHAR(20),
        sex CHAR(1),
        seat INT, age INT,
        majorId INT
    )
直接在类型后面添加约束,添加非空字段
    ALTER TABLE students MODIFY COLUMN stuName VARCHAR(20) NOT NULL
添加默认约束
    alter table students modify column age int default 18

添加主键
    alter table students modify column id int primary key
添加唯一键
    alter table students modify column seat int unique
添加外键
    alter table stuinfo add foreign key(majorId) references major(i
d)
```

(八) 、删除约束

1、语法:

```
直接字段类型后面什么都不添加,除了键除外
```

2、删除非空约束

```
alter table stuinfo modify column stuName varchar (20);
```

3、删除默认约束

```
alter table stuinfo modify column age int;
```

4、删除主键

```
alter table stuinfo modify column id int drop primary key;
```

5、删除唯一键

```
alter table stuinfo drop index seat;
```

6、删除外键

```
alter table stuinfo drop foreign key 外键名
```

(九) 、创建表时设置标识列(自增)

说明:

标识列必须和键约一起使用

一个表最多一个自增长列

每次添加数据自增长 1

设置默认值 set auto_increment_increment=3 设置默认值后自增长 3

如果想从 10 开始,就手动添加第一条数据 id 为 10 就从 10 开始了 6.自增长必须为 int 类型

案例一:创建一个 stu*izeng 表,id**字段为自增*

```
create table stu*ziezeng(
    id int primary key auto_increment, name varchar(20)
)
```

(十) 、修改表时设置标识列(自增)

```
alter table stu_zizeng modify column id primary key auto_increment
```

(十一) 、修改表时删除标识列(自增)

```
alter table stu_zizeng modify column id primary key
```

十七、事务

(一) 、含义:

将一多个 sql 语句组合合起来,一条失败则全部失败,要么全成功,要么全失败

(二) 、案例讲解:

张三丰有 1500,郭襄 500,张三丰向郭襄转 1000

```
    update table set salary=500
    where name=张三丰
    update table set salary=500
    where name=郭襄
```

如果第一条语句执行完出现未知错误第二个语句因此执行不成 功,张三丰的变为 500,而郭襄的还是 500,所有使用事务

(三) 、特点:

1.原子性:不可分割的工作单位,要么全执行,要么什么都不发生

2.一致性:发生前的数据和发生后的数据要一样

3.隔离性:一个事务是独立的,并发各个事务之间不能互相干扰

4.持久性:事务执行结束不能撤销

(四) 、savepoint 的使用(保存点)

ma1 先重置 1000 然后马 1 向马 2 转 500 块钱,在转钱的过程中出现了错误,就要回滚到充值后的状态,我们就可以使用 savepoint 记录一下保存点,相当于虚拟机的快照

重置 1000 块

```
update student set salary=1000 where name='ma1';
```

创建保存点 a

```
savepoint a;
UPDATE student SET salary=500 WHERE NAME='ma1';
```

发生错误,进行回滚

```
rollback to a;
UPDATE student SET salary=1500 WHERE NAME='ma2';
```

(五) 、事务的创建

1、隐式事务

隐式事务没有明显开启和结束的标记 以前的 update insert delete 都是隐式事务,可以单个提交

2、显示事务

如果 2 个 update 语句,这就是 2 个事务,需要设置禁用自动提交

禁用自动提交的功能,只在本次生效,重启恢复

```
set autocommit=0;
```

语法:

```
set autocommit=0;
start transaction; (可写可不写)
sql 语句:
    select update delete insert
    .... 2 种结束方式
        commit; 提交事务
        rollback; 回滚事务
```

案例一:创建 student 表,设置 name 和 salary 字段,添加 ma1,1000 和 ma2,1000,然后马 1 向马 2 转 500 块钱

```
创建表
    CREATE TABLE student( NAME VARCHAR(20), salary INT )
开启事务 SET autocommit=0;
sql 语句
    INSERT INTO student (NAME,salary)
    VALUES ('ma1',1000);
    INSERT INTO student (NAME,salary)
    VALUES ('ma2',1000);
    UPDATE student SET salary=500
    WHERE NAME='ma1';
    UPDATE student SET salary=1500
    WHERE NAME='ma2';
结束事务 COMMIT;
```

(六) 、隔离级别

1、设置当前 mysql 连接的隔离级别

```
set transaction isolation level read committed;
```

2、设置数据库系统全局隔离级别

```
set global transaction isolation level read committed
```

3、查看隔离级别

```
select @@tx_isolation;
```

4、事务的隔离级别

脏读:对于 2 个事务 T1,T2,T1 正在修改数据,此时 T2 读取的就是修改过的数据,如果 T1 回滚读取到的数据无效

不可重复读:对于 2 个事务 T1,T2,T1 读取一个字段,然后 T2 修改一个字段,2 次读取同一字段,值不一样

幻读:对于 2 个事务 T1,T2,T1 读取一个字段,然后 T2 插入一个新行,T1 再读取就会多出几行

脏读 幻读 不可重复读

read uncommitted √ √ √

read committed × √ √

repeatable read × √ ×

serializable × × ×

十八、视图

(一)、说明:

查询非常复杂而查询结果频繁使用的时候,就可以给查询结果起一个名字,把查询结果当作一个临时的表使用,称之为视图,就像 java 包装类

(二) 、视图的创建

create view 视图名 as 查询语句

案例一:查询邮箱中包含 a 字符的员工名,部门名和工种名

1、创建视图

```
CREATE VIEW myv1 AS
SELECT last_name,department_name,job_title,email
FROM employees e JOIN departments d
ON e.department_id = d.department_id JOIN jobs j ON e.job_id =
j.job_id
```

2、使用视图

```
SELECT * FROM myv1 WHERE email LIKE '%a%'
```

案例二:查询每个部门平均工资的级别

1、创建视图

```
CREATE VIEW myv2 AS
SELECT AVG(salary),department_id
FROM employees GROUP BY department_id
```

2、使用视图

```
SELECT myv2.`avg(salary)`,grade_level FROM myv2
JOIN job_grades j
ON myv2.`avg(salary)`
BETWEEN j.lowest_sal AND j.highest_sal
```

案例三:查询平均工资最低的部门信息

使用上方的 myv2 select * from myv2 order by ag limit 1;

案例四:查询平均工资最低的部门和工资

```
SELECT *
FROM myv2
JOIN departments d
ON myv2.`department_id` = d.`department_id`
ORDER BY myv2.`avg(salary)` LIMIT 1;
```

(三) 、视图的修改

1、没则创建,有则修改

```
create or replace view 视图名 as 查询语句
```

2、创建

```
alter view 视图名 as 查询语句
```

(四) 、删除视图

需要用户有一定的权限 drop view 视图 1,视图 2,视图 3....

(五) 、查看视图

```
desc 视图名
```

(六) 、视图的更新

1、准备一个视图 myv3

```
CREATE VIEW myv3 AS SELECT last_name,email FROM employees
```

2、插入数据

```
insert into myv3 values('张飞','299@qq.com')
```

3、修改数据

```
UPDATE myv3 SET last_name='小马' WHERE last_name='张飞'
```

4、删除数据

```
delete from myv3 where last_name='张无忌'
```

(七) 、视图的权限

具备一下特点的视图不允许更新

包含分组函数和关键字: distinct、group by、having、union、union all

常量视图(常量,有确定赋值的值) create view myv1 select name='aa'

select 中包含子查询 select (select .... from employees)

用上 join,说白了就是连接查询都不能用

视图中包含视图 create view myv1 select * from myv2 5.where 子查询用到的表和主查询用到的表相同时

十九、变量

(一) 、系统变量

系统变量由系统提供,不是用户定义,属于服务器层面

(二) 、查看变量

查看全局变量 show global variables

查看局部变量 show variables

查看包含什么什么的变量 show global|variables like "%***%"

查看指定的某个系统变量的值 select @@global|session.变量名 5.为某个系统变量复制 set global|session 变量名=值

(三) 、全局变量

1.服务器每次启动将为所有全局变量赋值默认值

(四) 、会话变量

仅仅针对于当前会话有用

(五) 、自定义变量

1、用户变量

变量是用户自定义的 2.步骤:声明,赋值,使用

1.1、作用域

针对于当前会话(连接)有效

1.2、使用

如果没有就是声明,如果有就是赋值和 java 一样 set @变量名:=值

另一种赋值,将查询的内容赋值给变量,只能一个字段 select 字段 into 变量名 from 表名

查看变量的值 select @用户变量名

2、局部变量

2.1、作用域

仅仅在定义它的 begin ned 中

2.2 使用

声明 declare 变量名 类型 default 值;

赋值查看与用户变量一样

二十、存储过程

非常重要:使用 in 的时候必须输入 set names gbk$

(一) 、说明:

类似于 java 的方法

(二) 、含义:

一组预先准备好的 sql 集合

(三) 、存储过程格式

参数列表分为三部分

1.参数模式(in out inout)

2.参数名

3.参数类型

in:需要外面传过来值

out:返回值 return

inout:又输入,又输出

如果方法体只有一句话,begin end 可以省略

方法体中 sql 语句以分号(;)结尾

存储过程结尾可以使用 delimiter 设置 delimiter % 设置全部结束符号

```
create procedure 方法名(参数列表)
begin 方法体 end
```

(四) 、使用存储过程

需要在控制台使用 call 方法名(实参列表)

(五) 、案例

案例一:插入 admin 表(girls 库)中五条数据 无参的方法

```
DELIMITER $ CREATE PROCEDURE myp1()
BEGIN INSERT INTO admin(username,PASSWORD)
VALUES ('a1','1'), ('a2','2'), ('a3','3'), ('a4','4');
END $ CALL myp1()$
```

案例二:创建存储过程实现根据女神名查询男神信息(in 参数)

```
DELIMITER $ CREATE PROCEDURE myp2(IN bName VARCHAR(20))
BEGIN
   SELECT bo.*
   FROM boys bo
   RIGHT JOIN beauty b
   ON bo.id = b.boyfriend_id
   WHERE bName = b.name;
END $

CALL myp2()$
```

案例三:判断用户是否登录成功(双 in 参数)

```
DELIMITER $ CREATE PROCEDURE myp3(IN username VARCHAR(20),IN PASSW
ORD VARCHAR(20))
BEGIN DECLARE result VARCHAR(20) DEFAULT '';

SELECT COUNT(*) INTO result
FROM admin
WHERE admin.`username` = username AND admin.`password` = PASSWORD;
 SELECT IF(result >0,'成功','失败');
END $

CALL myp3('a2','2') $
```

案例四:根据女神名,返回对应的男神名(out 的使用)

```
DELIMITER $
CREATE PROCEDURE myp4(IN beautyname VARCHAR(20),
OUT boname VARCHAR(20))
BEGIN S
ELECT boyName
FROM boys bo
JOIN beauty b ON bo.`id` = b.`boyfriend_id`
WHERE beautyname = b.name;
END $ SET @bname:='';
```

```
CALL myp4('小昭',@bname) $ SELECT @bname $
```

案例五:根据女神名返回对应男神名和魅力值(双 out 使用)

```
DELIMITER $ CREATE PROCEDURE myp4(IN beautyname VARCHAR(20),OUT bo
name VARCHAR(20))
BEGIN SELECT boyName,
FROM boys bo JOIN beauty b
ON bo.`id` = b.`boyfriend_id`
WHERE beautyname = b.name;
END $ SET @bname:='';

CALL myp4('小昭',@bname) $ SELECT @bname $
```

案例六:传入 a 和 b 两个值,最终 a 和 b 都翻倍返回过来

```
DELIMITER $ CREATE PROCEDURE myp6(INOUT a INT,INOUT b INT)
BEGIN SET a=a*2; SET b=b*2;
END $ SET @a:=2; SET @b:=3;

CALL myp6(@a,@b) $ SELECT @a,@b $
```

(六) 、删除存储过程

```
drop procedure 方法名
```

(七) 、查看存储过程的信心

```
show create procedure 存储名
```

二十一、函数

(一) 、说明:

与存储过程差不多

(二) 、区别:

存储过程可以多个返回,函数只能有一个

(三) 、创建语法

参数列表包括参数名,参数类型

函数体中必须有 return,函数体写完返回

函数体只有一句,可以省略 begin end

和存储过程一样需要用 delimiter 设置结束标记

```
create founction 函数名(参数列表)
returns 返回类型
begin 函数体 end
```

(四)、调用语法

```
select 函数名(参数列表)
```

(五)、案例:

案例一.返回公司员工个数(无参)

结束语

```
   delimiter$
```

返回变量

```
   declare c int default 0;
```

创建函数

```
   CREATE FUNCTION myf1() RETURNS INT BEGIN DECLARE c INT DEFAULT
0; SELECT COUNT(*) INTO @c FROM employees; RETURN c; END $
```

使用

```
   select myf1()$
```

案例二:根据员工名返回他的工资(有参)

```
DELIMITER$
CREATE FUNCTION myf3(yname VARCHAR(20))
RETURNS DOUBLE BEGIN SET @d=0; SELECT salary
INTO @d FROM employees
WHERE yname = last_name;
RETURN @d;
END $

SELECT myf3('Kochhar')$
```

案例三:根据部门名,返回该部门平均工资(有参)

```
DELIMITER$
CREATE FUNCTION myf4(yname VARCHAR(20))
RETURNS DOUBLE BEGIN SET @d=0;
```

```
SELECT AVG(salary) INTO @d
FROM employees e
JOIN departments d
ON e.department_id = d.department_id
WHERE yname = d.`department_name`;
RETURN @d;
END $
SELECT myf4('IT')$
```

(六) 、查看函数

```
show create function 函数名;
```

(七) 、删除函数

```
drop function 函数名;
```

二十二、流程控制结构

顺序结构:从上往下

分支结构:选择性执行

循环结构:for while 循环

(一) 、分支结构

1、if 结构

```
if(表达式 1,表达式 2,表达式 3) 当表达式 1 成立返回表达式 2,不成立返回表达式 3
```

案例:传入成绩,返回等级

```
DELIMITER$
CREATE FUNCTION myq1(cheng INT) RETURNS CHAR BEGIN
IF cheng>=90 THEN RETURN 'A';
IF cheng>=80 THEN RETURN 'B';
IF cheng>=60 THEN RETURN 'C';
ELSE RETURN 'D';
END IF;
END $

CALL myq1(90)$
```

2、case 结构

可以嵌套在任何地方,上面有案例

可以单独为表达式使用,但只能在 begin end 中

情况一:

```
case 变量|表达式|字段
when 要判断的值 then 返回的值 1
when 要判断的值 then 返回的值 2
....
else 要返回值的 n
end
```

情况二:

```
case
when 要判断的条件一 then 返回的值 1
when 要判断条件二 then 返回的值 2
....
else 要返回值的 n
end
```

case 放在 begin end 中案例

语法改变:

```
case
when 要判断条件一 then select 值 1;
when 要判断条件二 then select 值 2;
....
else 要返回值的 n;
end case;
```

案例:传入成绩,判断等级

```
DELIMITER$
CREATE PROCEDURE myq1(IN cheng INT) BEGIN
CASE
WHEN cheng>=90 THEN SELECT 'A';
WHEN cheng>=80 THEN SELECT 'B';
```

```
WHEN cheng>=60 THEN SELECT 'C';
ELSE SELECT 'D';
END CASE;
END $

CALL myq1(80)$
```

(二)、循环结构

1、分类

while、loop、repeat

2、循环控制:

iterate:结束本次循环,继续下一次

leave:跳出循环