权限管理:设置所有的ip均可访问

grant all privileges on \*.\* to root@'%' identified by 'root' with grant option;

刷新

flush privileges;

# 启动停止与登陆

时区问题:

set global time\_zone="+8:00";

1.启动和停止(管理员启动DOS窗口)

net stop/start MySQL

2.登录:

(1)使用mysql client直接输入密码(仅限于root用户)

(2)使用cmd命令窗口:mysql –h localhost –P 3306 –u root –p

h代表:host地址 P代表post端口号 u 用户 p代表密码

简写:mysql –u root -p

3.退出 ctrl+c(结束进程) 或者 exit

4.乱码 set names gbk;

二、常用操作命令

1.显示数据库:

show databases;

2.显示表:

show tables;

3.查看处于哪一个数据库

select database();//查看当前处于哪个库

4.在当前库查询其他库的表

show tables from mysql;不换库的情况下查询其他数据库的表

5.查看某个表的表结构

desc tableName;查看某个表的结构

6.查看mysql当前版本

(1)select version();

(2)在cmd界面输入 mysql --version;或者,mysql –V

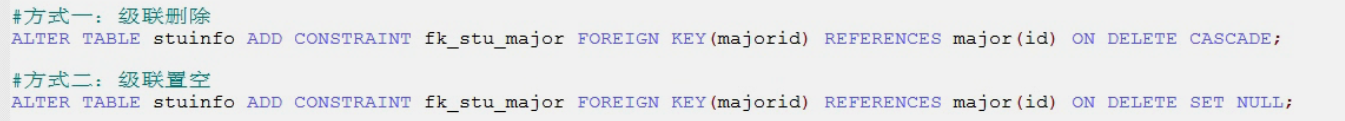
` `这个着重号 :用来区分一个字段还是一个关键字 可读性比较好(键盘1左边的键)

查看索引

Show index from 表名;

查看存储引擎

Show engines;



# 二、查询

## select 基础查询函数

1. 去重(distinct) 可以和分组函数搭配使用

(1)distinct后面只能加入一个字段 也不能出现 distinct a,distinct b,这样是错误的

select dinstinct department\_id from department;

2.+号的作用仅仅只有一个功能**运算符**

(1)例如:”123”+90 其中一方为字符串则试图将字符串转换成数值型,转换成和就相加,失败则将字符串转换成0继续运算,只要其中一方为null 相加结果一定为null

(2)select last\_name+first\_name as 姓 from employees;这个拼接就会失败

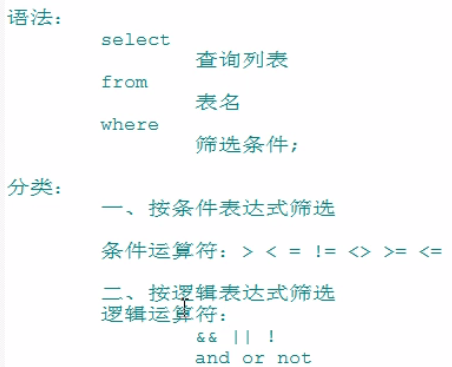
(3)如何拼接字段使用函数concat(..);

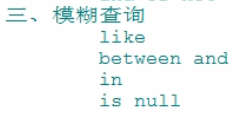
select concat(‘a’,’b’,’c’); 如果其中有null name拼接出来的就为null

null与任何拼接都为null

如果不想显示null,使用 函数 ifnull(arg1,arg2);

## select 条件查询



 is not null

(1)in(arg1,arg2,arg3) 形参不支持通配符 因为其中的形参默认使用=来匹配的

(2)<>或者=不能用于判断null 用is null 或者 is not null

(3)安全等于<=>判断是否=,(可读性比较差)

可以用于判断null也可以判断普通的值

(4)字符型和日期型的常量值必须使用单引号引起来,数值型不需要

(5)mysql中的函数 必须加select 并且 必须有返回值

(6)isnull 判断某字段或表达式是否为null 如果is 返回1,否则返回0;

## select 排序查询 order by 字段名 desc(降序)/asc(升序);

(1)可以使用表达式进行排序

select \*,salary\*12\*(1+ifnull(commission\_pct,0)) 年薪

from employees

order by salary\*12\*(1+ifnull(commission\_pct,0));

(2)可以使用别名进行排序

select \*,salary\*12\*(1+ifnull(commission\_pct,0)) 年薪

from employees

order by年薪;

(3)按函数排序,要求:按last\_name的长度进行排序 length(agrs) 放的是字符类型

select length(last\_name) as 字节长度,last\_name,salary

from employees

order by length(last\_name);

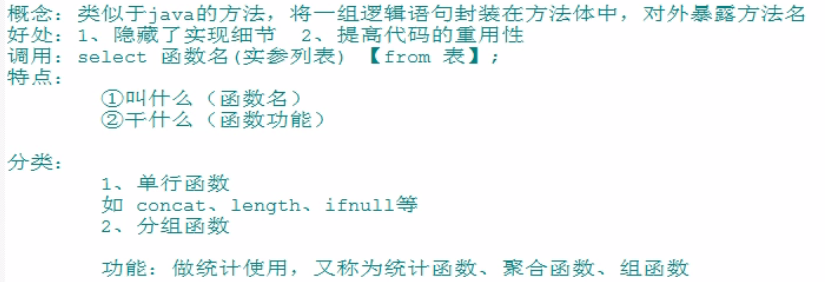
(4)多个字段进行排序放在前面的优先级高

select \* from employees

order by salary asc, employees\_id desc;

## 常见函数

函数调用都需要使用select 函数(形参),函数可以嵌套函数



### 单行函数:

1.length(str);获取str的字节长度 如果是汉字utf8一个汉字3个字节 ,gbk 2个字节

2.concat(…);不定长参数 用于拼接

3.upper(str) 转大写 lower(str) 转小写

4.字符串截取函数注意sql语句中索引是从1开始的

select substr(“str”,pos); 从pos开始截取以后的所有字符

select substr(“str”,pos,len);从第几位开始截取len长度个字符

5.instr(str,subStr) 返会subStr在str中第一次出现的索引,如果没有返回0

6.trim(str); 去除前后空格

trim(str1 from str2); 举例:select trim(‘a’ from ‘aaaaaaaaaaaaa张aa翠aa山aaaaaaaaaa’);

会把开始和结尾的a都去除掉

7.lpad(str,len,padStr);左填充在str的左边填充padStr,直到字符长度为len

select lpad(‘殷素素’,10,’\*’)结果为 \*\*\*\*\*\*\*殷素素;

select lpad(‘殷素素’,2,’\*’);结果为殷素

最终返回的长度一定和len相等

8.rpad和lpad相同的原理

9.replace(str,oldStr,newStr);把str中所有的oldStr替换成newStr和javareplace相同

### 数学函数

1. round四舍五入

select round(1.45); 🡺1 默认不保留小数

select round(1.65,2);可以指定保留多少位小数

1. ceil 向上取整>=该参数的最小整数

select ceil(1.002);🡺2

select ceil(-1.002);🡺-1

1. floor向下取整 返回<=该参数的最大整数

select floor(-9.99);🡺-10

1. truncate 截断函数

truncate(1.6999,1);截断保留一位小数🡺1.6

1. mod 取余 相当于%

select mod(10,-3);

### 日期函数

1. now() 返回当前系统日期+时间

select now();

1. curdate() 返回系统当前日期 不包含时间
2. curtime() 返回系统当前的时间
3. 可以获取指定的部分,年,月,日,小时,分钟,秒

year(date) 传入一个date类型的形参

month(date);

monthname(date) 当前月份的英语名称

举例:select year(now()) 年;

1. str\_to\_date(‘str’,formatStr) 将字符串通过指定的格式转换成默认日期

即使用formatStr格式去解析字符串 转换成默认格式



select \* from employees where hiredate=str\_to\_date(‘4-3 1992’,’%c-%d %Y’)

1. date\_format(date,formatStr);将日期转换成指定格式的字符串
2. 日期相减函数 datediff(arg1,arg2);

可以使用select datediff(now(),’1998-08-10’);得到你活了多少天

### 其他函数

select version();版本

select database();目前所在的库

select user();当前用户

### 加密函数:

md5(str);

password(str);

### 流程控制函数

1. if函数:可以实现if else的效果

select if(boolean,str1,str2) 如果boolean为true 返回str1 否则返回str2

select if(10>5,’大’,’小’);

1. case when then 函数 如果then 后面是一个值 不加;如果是一句话加;

举例:select salary 原工资,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 as 新工资

from employees order by department\_id asc;

1. 多重if else 同样使用case when then函数 case后面不跟值 when后面跟判断语句

举例:SELECT salary,

CASE

WHEN salary>30000 THEN 'C'

WHEN salary>20000 THEN 'B'

WHEN salary>10000 THEN 'A'

ELSE 'D'

END AS 级别

FROM employees;

### 分组函数

功能:用作统计作用,又称聚合函数或统计函数或组函数

分类:

1.sum 求和 avg 求平均值 max 最大值 min最小值 count计算个数(非null个数)

(1)sun avg只能处理数值型

(2)max min count可以处理任何类型

(3)上述分组函数都忽略null值

2.和 dinstinct搭配使用

select count(distinct salary) ,count(salary) form employees;得出的结果不同 说明distinct起作用了

3.count 函数详细介绍

(1)当使用count(id) 时如果为null的字段就不会统计上 效率低

(2)可以使用count(\*) 只要一行中有一个字段不为null就可以统计上 效率高

(3)或者count(1) 会在查询的表中虚拟一列全为1,count(1)就相当于统计全部的个数效率高

综上所述:用count(\*) 效率高并且用的多

1. 和分组函数一同查询的字段有限制

select avg(salary),employee\_id form employees; 语法没错误但是逻辑有错误

和分组函数一同查询的字段要求是group by后的字段

# 三、分组查询

语法:

select 分组函数,列(要求出现在group by的后面)

from 表

[where 筛选条件]

group by 分组的列表

[order by 字句]

注意: 查询列表必须特殊,要求是分组函数和group by后出现的字段

### 简单分组查询:

1.案例1

-- 查询每个工种的最高工资

select max(salary),job\_id from employees group by job\_id;

2.案例2

#查询每个位置上的部门个数

SELECT COUNT(\*),location\_id

FROM departments GROUP BY location\_id;

### 添加筛选条件

1.#案例1:查询邮箱中包含a字符的,每个部门的平均工资

SELECT AVG(salary),department\_id

FROM employees WHERE email LIKE '%a%'

GROUP BY department\_id;

1. 案例2:查询有奖金的每个领导手下员工的最高工资

SELECT MAX(salary),manager\_id FROM employees

WHERE `commission\_pct` IS NOT NULL

GROUP BY manager\_id;

### 添加复杂的筛选条件

分组前的查询使用where 在group by之前 分组查询后的过滤条件使用having 并且放在group by之后

group by和having后面都支持别名但是where 后面不支持别名,但是别名不常用,因为只有mysql支持 oracle等其他一般不支持

1. 案例1:查询部门的员工个数>2的部门(分组后的查询)

SELECT COUNT(\*),department\_id FROM employees

GROUP BY department\_id

HAVING COUNT(\*)>2

1. 案例2:查询每个工种有奖金的员工的最高工资>12000的工种编号和最高工资

SELECT MAX(salary),job\_id FROM employees

where commission\_pct is not null

GROUP BY job\_id

HAVING MAX(salary)>12000;

1. 案例3.查询领导编号>102的每个领导手下的最低工资>5000的领导编号是哪个以及最低工资

SELECT MAX(salary),MIN(salary) 最低,manager\_id

FROM employees

WHERE manager\_id>102

GROUP BY manager\_id

HAVING 最低>5000

ps:做这种题的时候可以分步进行

### 按表达式或函数分组

1.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 COUNT(\*),LENGTH(last\_name)

FROM employees

GROUP BY LENGTH(last\_name)

HAVING COUNT(\*)>5;

### 按多个字段分组

多个字段分组的时候group by 字段1,字段2,…

只有字段1和字段2都相同的两条数据才会被认为是一组,没有顺序要求

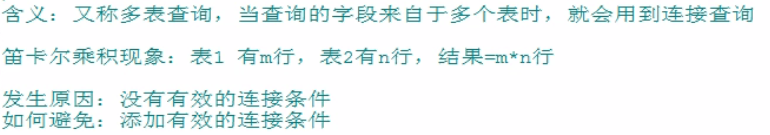
案例:查询每个部门每个工种的员工的平均工资

select avg(salary),department\_id,job\_id

from employees group by department\_id,job\_id

order by job\_id;

# 四、连接查询(多表查询)



## sql92标准

### 1.等值连接

案例1.查询女神名和和对应的cp

select name,boyName from boys,beauty where beauty.boyfriend\_id=boys.id;

案例2.加筛选 查询有奖金的员工名、部门名

select last\_name,department\_name,commission\_pct

from employees e，department d

where e.department\_id=d.department\_id

and e.commission\_pct is not null;

注:

1. 多表等值连接的结果为多表的交集部分
2. n个表连接,至少需要n-1个连接条件
3. 多表的顺序没有要求
4. 一般需要为表起表名
5. 可以搭配前面介绍的所有字句使用,如排序,分组,筛选

### 2.非等值连接

#案例1:查询员工的工资和工资级别

select salary,grade\_level

from employees e,job\_grades j

where salary between lowest\_sal and highest\_sal;

### 3.自连接

案例:查询员工名和上级的名称

SELECT e1.`last\_name`,e1.manager\_id,e2.employee\_id,e2.last\_name

FROM employees e1,employees e2

WHERE e1.`manager\_id`=e2.employee\_id;

## sql99标准

### 内连接

#### 等值连接

语法:select \* from tableName join tableName2 on tableName1.clu=tableName2.clu

join tableName3 on tableName1.clu=tableName3.clu

where tableName\*.\*=?;

group by

order by

limit

#### 非等值连接

语法:

select salary grade\_leave

from employees join job\_grades

on salary between lowest\_sal and highest\_sal

### 外连接

应用场景:用于查询一个表中有,另一个表中没有的查询

有主从表之分,主表一定会全部显示从表不一定

1. 外连接的查询结果为主表中的所有记录

如果从表中有和它匹配的,则显示匹配的值

如果从表中没有和它匹配的则显示null

外连接查询结果=内连接结果+主表中有而从表中没有的记录

1. 左外连接,left join 左边的是主表

右外连接right右边的是主表

左外和右外交换两个表的顺序可以实现相同的效果

案例1:查询没有男朋友的女神

SELECT b.name,bo.\*

FROM beauty b

LEFT JOIN boys bo ON b.`boyfriend\_id`=bo.`id`

WHERE bo.id IS NULL

(where筛选的时候最好选主键因为主键一定不为null 除非是经过外连接筛选出来的结果)

### 全外连接

mysql语法暂时不支持全外连接

全外连接=内连接的结果+表1中有但表2中没有的+表2中有但表1中没有的

### 交叉连接

select b.\*,bo.\* from beauty b cross join boys bo;

相当于笛卡尔乘积

92语法中笛卡尔乘积

select b.\*,bo.\* from beauty b,boys bo;

# 五、子查询

子查询优先于主查询

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,称为主查询或外查询

分类:

按子查询出现的位置:

select后面

from后面

where或having后面 重点☆

标量子查询 √

列子查询 √

行子查询

exists后面(相关子查询)

表子查询

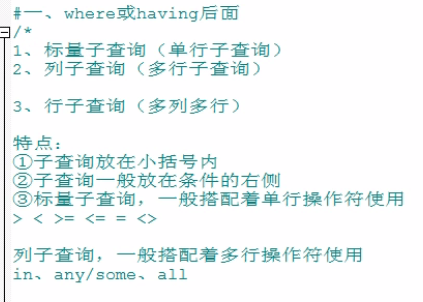
按结果集的行列数不同

标量子查询(结果集只有一行一列)

列子查询(结果集只有一列多行)

行子查询(结果集只有一列)

表子查询(结果集一般为多列多行)



## where 后面

### **标量子查询**

案例1:查询工资比abel高的员工

select salary from employees where(

select salary from employees where last\_name=’Abel’

);

案例2:返回job\_id 与141员工相同,salary比143好员工多的员工姓名,job\_id和工资

支持多个子查询语句

select last\_name,job\_id,salary from employees

where job\_id=(

select job\_id from employees where employees\_id=141

)and salary>(

select salary from employees where employee\_id=143

)

案例3:查询最低工资大于50号部门最低工资的部门id和其最低工资

select min(salary),department\_id from employees group by department\_id

having min(salary)>(

select min(salary) from employees where department\_id=50

);

### **列子查询(一列多行**)

使用多行比较操作符 in/not in any/some(可以使用min/max替换) all(可以使用max.min替换)

案例1:查询location\_id是1400或1700的部门中的所有员工的姓名

select last\_name from employees where department\_id in(

select distinct department\_id from departments where location\_id in(1400,1700)

)

案例2:返回其他工种中比job\_id为`IT\_PROG`部门任一工资低的员工的员工号、姓名、job\_id以及salary

select last\_name,employee\_id,job\_id,salary

from employees where salary<any(

select distinct salary from employees where job\_id=`IR\_PROG`

) and job\_id<>`IT\_PROG`;

返回其他部门中比job\_id为`IT\_PROG`部门所有工资都低的员工 员工号、姓名、job\_id 以及salary

SELECT employee\_id,last\_name,job\_id,salary

FROM employees WHERE salary<ALL(

SELECT salary FROM employees WHERE job\_id='IT\_PROG'

) AND job\_id<>'IT\_PROG';

### 行子查询

应用较少

案例1:查询员工编号最小,并且工资最高的员工

SELECT \* FROM employees WHERE (employee\_id,salary)=(

SELECT MIN(employee\_id) ,MAX(salary) FROM employees

)

## select 后面

案例: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 ) 个数

from departments d;

## from后面

相当于将子查询的结果当做数据表必须起别名

案例1:查询每个部门的平均工资的工资等级

SELECT dep\_avg.\*,j.`grade\_level`

FROM (SELECT AVG(salary) av,department\_id

FROM employees GROUP BY department\_id) dep\_avg

JOIN job\_grades j WHERE dep\_avg.av BETWEEN j.`lowest\_sal` AND j.`highest\_sal`;

## exists后面 (相关子查询)

select exists(select employee\_id from employees) 如果存在返回1不存在返回0

# 六、分页查询

语法:

select 查询列表

from 表

[join type] join 表2

where 筛选条件

group by 分组字段

having 分组后的筛选

order by 排序的子段

limit offset ,size

在这里起始索引是0

公式:select \* from \* limit (page-1)\*size,size

案例1:查询前五条员工信息

select \* from employees limit 0,5

案例2:查询第11条-第25条

select \* from employees limit 10,15

# 七、联合查询

union 联合 合并 将多条查询语句的结果合并成一个结果

特点:

多条查询的列数必须是一致的,

类型和顺序需要保持一致

如果存在重复数据会自动去重,如果不想去除重复数据,加一个关键字 union all

语法:

查询语句 union 查询语句2 union

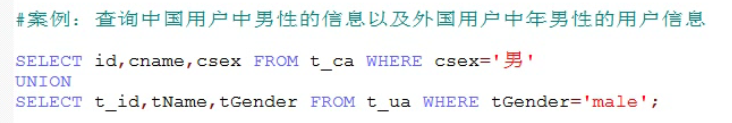
应用场景:

要查询的结果来自于多个表,并且多个表没有直接的连接关心,但查询的信息一致时

案例1:查询部门编号>90或邮箱包含a的员工的信息

select \* from employees where email like %a%

union select \* from employees where department\_id>90;





# 八、DML语言

## 插入语句：

方式一:支持插入多行

insert into 表名(列名,…) values(值1,…);

1.列名和值一一对应 插入的类型必须一致或者兼容

insert into beauty values(13,’唐艺昕’,’女’,’1990-4-230’,’15888888888’,null,2);

1. 不可以为null的必须插入值,可以为null的值如何插入值

插入null 或者 列不写,值也不写都省略

4.可以省略列名,默认所有列,但是值得顺序和表中列的顺序必须一致

方式二:

insert into 表名 set 列名=值,列名=值;

两种方式比较:

1方式一支持多行插入,方式二不支持

Insert into tableName values(值1,值2,…),(值一,值2,…);

1. 方式一支持子查询,方式二不支持

insert into beauty (id,name,phone) select …;

## 修改语句:

1.修改单表的记录

语法:

update 表名 set 列=新值,…where 筛选条件

如果不加筛选条件 就会把所有的都修改

2.修改多表的记录

语法:

sql92语法

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

sql99语法:

update 表1 别名 inner/left/right join 表2 别名 set 列=值 where 连接条件 and 筛选条件

案例:修改张无忌的女朋友的手机号为114

update boys bo inner join beauty b on bo.id=b.boyfriend\_id set b.phone=114 where bo.boyName=’张无忌’

案例:修改没有男朋友的女神的男朋友编号都为2

update boys bo right join beauty b on bo.id=b.boyfriend\_id set b.boyfriend\_id=2 where bo.id isnull

## 删除语句

方式一、

1.单表的删除

delete from 表名 where 筛选条件;

2.多表的删除

sql92语法:

delete 表1别名,表2别名 from 表1 别名,表2 别名 where 连接条件 and 筛选条件

在delete 后面如果只有表1的别名那么就是删除表1中的数据,如果只有表2的别名那么就是只删除表2的数据 如果两个都有 那么就级联删除

sql99语法

delete 表1的别名,表2的别名 from 表1 别名

inner/left/right join表2 别名 on 连接条件 where 筛选条件

在delete 后面如果只有表1的别名那么就是删除表1中的数据,如果只有表2的别名那么就是只删除表2的数据 如果两个都有 那么就级联删除

方式二:清空数据 效率高一些

truncate table 表名；不允许加where

将整个表的数据都删除

比较:

假如要删除的表中有自增,如果用delete删除后,自增长值从断点处开始

如果用truncate删除 自增长值从1开始

truncate没有返回值 delete有返回值

truncate删除后不能回滚 delete删除后可以回滚

# 九、DDL语言 数据定义语言

一、库的管理

创建 修改 删除

二、表的管理

创建 修改 删除

创建: create

删除: drop

修改: alter

## 库的管理

### 1.库的创建

语法:

create database 库名;

create database book;

create database if not exists books;

### 2.库的修改

更改库的字符集

alter database books character set gbk/utf8;

### 3.库的删除

drop databases books;

## 表的管理

### 1.表的创建

语法:create table 表名(

列名 列类型 [(长度)] [约束],

列名 列类型 [(长度)] [约束],

列名 列类型 [(长度)] [约束]

);

### 2.表的修改:

alter table 表名 add/drop/modify/change column 列名[列类型 列约束]

修改表的列名(change)

alter table book change [column] publishdate pubDate datetime;

ALTER TABLE t\_department CHANGE total\_user totalUser INT;

更改列的类型或约束: (modify)

ALTER TABLE t\_department MODIFY total\_User VARCHAR(10);

alter table t\_department modify total\_User int;

添加新列(add)

ALTER TABLE t\_department ADD test VARCHAR(20);

删除列(drop)

alter table t\_department drop test;

修改表名:

alter table t\_department rename to tDepatement;

### 3.表的删除

drop table 表名;

4.表的复制:

仅仅复制表的结构:

drop table if exists t\_copy ;

create table t\_copy like t\_department;

把表的数据一起复制(也可以只复制一部分数据加筛选条件):

create table t\_copy select \* from t\_department [where 筛选条件];

只复制表的某些字段并且不要数据

create table t\_copy select id,name from t\_department where 1=2;

# 十、常见的数据类型

选择的原则:越简单越好,能保存数值的类型越小越好(节省空间)

## 数值型:

### 整数:

类型 字节 范围

Tinyint 1 有符号:-128-127

无符号:0-255

Smallint 2 有符号:-32768-32767

无符号:0-65535

Mediumint 3 不需要记了

int/Integer 4

Bigint 8

1. 如何设置有符号和无符号(默认是有符号的)

举例:

create table tab\_int(

t1 int,#(默认有符号)

t2 int unsigned #(无符号的)

)

desc tab\_int;

2.如果设置的范围超出了临界值,会报异常,默认插入临界值

3.如果不设置长度会有默认的长度 int 有符号11 无符号10

如果设置长度(长度代表显示结果,如果不够会用0填充)需要配合zerofill使用 ,一般不需要设置长度

### 小数:

M和D可以省略

M代表: 小数位数和整数位数加起来不得超过M,否则会插入一个临界值

D代表:小数点后保留位数

#### 定点数

用于保留精确度较高的数值如货币运算

DECIMAL(M,D)可以简写为DEC(M,D)

当省略M,D的时候默认是decimal(10,0);

#### 浮点数

float(M,D) 4字节

double(M,D) 8字节

## 字符型

#### 较短的文本

char,varcahr

#### 较长的文本

text,blob(较长的二进制数据)

## 日期型:

Date

Year

Time

Datetime 和时区无关

Timestamp 和时区有关

# 十一 常见约束

create table 表名(

字段名 字段类型 约束,

……,

字段名 字段类型 约束

)

分类:六大约束

not null:非空 用于保证该字段的值不能为空

default: 默认 用于保证该字段有默认值

primary key:主键,用于保证该字段的值具有唯一性,并且非空

unique: 唯一 保证该字段的值具有唯一性,可以为空

check:检查约束(mysql中不支持但是不报错)

foreign key:外键 用于限制两个表的关系,用于保证该字段的值必须来自主表的关联列的值

约束添加的时机:

创建表时

修改表时

约束的添加分类

列级约束

六大约束语法上都支持,但是外键约束没有效果

表级约束

除了非空,默认,其他都支持

## 列级约束

create table 表名(

字段名 类型 列级约束

字段名 类型 列级约束

表级约束

)

列级约束可以添加多个 只支持 默认 非空 主键 和唯一

列级约束举例:

Create table stuInfo(

Id int primary key.

stuName varchar(20) not null unique,#多个约束 非空和主键

gender char(1) check(gender=’男’ or gender =’女’),#检查约束mysql 不起作用

seat int unique,#唯一

age int default 18,#默认约束

majorId int references major(id),#外键 列级不支持 但是不报错

)

## 表级约束语法

在各个字段最下面

[constraint 约束名]🡨(可以不写 有默认的名称)约束类型(字段名)

#表级约束举例

Create table stuInfo(

Id int,

stuName varchar(20),

gender char(1),#检查约束mysql 不起作用

seat int,#唯一

age int,#默认约束

majorId int,#外键 列级不支持 但是不报错

constraint 主键的名称 primary key(id),#表示为id添加主键

constraint 唯一键名称 unique(seat),#唯一键

constraint ch check(gender=’男’ or gender =’女’),#检查

constraint fk\_stuinfo\_major foreign key(majorId) references major(id), #外键

# constraint 约束名 可以不写

)

主键和唯一的区别

保证唯一 是否为空 一个表可以有几个 组合

主键 √ × 至多一个 √但不推荐

唯一 √ √ 多个 √但不推荐

外键:

1. 要求在从表中设置外键关系
2. 从表的外键列的类型和主表的关联列的类型要求一致或者兼容,名称无要求
3. 主表的关联列必须是一个key(一般是主键或唯一)
4. 插入时先插入主表然后插入从表
5. 删除时先删除从表然后 主表

## 修改表时添加约束

1.添加非空约束

Alter table stuInfo modify column stuName varchar(20) not null;

2.删除非空约束

Alter table stuInfo modify column stuName varchar(20) null;

Alter table stuInfo modify column stuName varchar(20) ;

3.添加默认约束

alter table stuinfo modify column age int default 18;

4.添加主键

①列级约束

Alter table stuInfo modify column id int primary key;

②表级约束

Alter table stuInfo add primary key(id)

5.添加唯一键

①列级约束

Alter table stuInfo modify column seat int unique

②表级约束

Alter table stuInfo add unique(seat)

6.添加外键

Alter table stuInfo add [constraint 约束名] foreign key(majorId) references major(id);

删除约束

1. 删除默认值

Alter table stuInfo modify column age int

1. 删除主键

Alter table stuInfo drop primary key

1. 删除唯一约束

Alter table stuInfo drop index 唯一约束名

1. 删除外键约束

Alter table stuInfo drop foreign key 外键名称

# 十二 标识列

又称为自增长列,

含义:可以不用手动插入值 ,系统提供默认的序列值

特点:

* 1. 标识列必须和主键搭配吗?

否,但要求是一个key

* 1. 一个表中只能有一个标识列
  2. 标识的类型只能是数值型
  3. 标识列可以使用Set auto\_increment\_increment=3;(更改后是全部的表都更改)设置步长
  4. 可以通过手动插入值得方式设置起始值

## 创建表时设置标识列

create table table\_identity(

id int primary key auto\_increment,#自增长

name varchar(20)

)

Show variables like %auto\_increment%;



第一个是每次增加几 第二个是偏移量初始值为几(不支持设置)

可以设置步长每次增加几

Set auto\_increment\_increment=3;(更改后是全部的表都更改)

## 修改表时设置标识列

Alter table tab\_identity modify column id int primary key auto\_increment

删除

Alter table tab\_identity modify column id int;

# 十三 事务

事务的ACID属性

原子性:不可再分割

一致性:事务必须使数据库从一个一致性状态变成另一个一致性状态

隔离性:一个事务的执行不能受其他事务的干扰

持久性:一旦提交,对数据库中的数据的改变是永久的

对于事务 delete可以回滚 但是truncate无法回滚

查看自动提交

Show variables like %autocommit%;

Set autocommit=0;设置自动提交关闭只针对当前会话

步骤:

开启事务

Set autocommit=0;

Start transaction;可选的

编写事务中的sql语句

Commit

或者

Rollback

## 隔离级别

对于同时运行的多个事务,当这些事务访问数据库如果没有采取必要的隔离措施,就会导致各种并发问题

1. 脏读: 对于两个事务t1 t2,t1读取了已经被t2更新但还没有被提交的字段,若t2回滚,t1读取的内容就是无效的
2. 不可重复度:对于t1 t2,t1读取了一个字段,然后t2更新了该字段,之后t1再次读取同一个字段,值就不同了
3. 幻读:对于两个事务t1 t2,t1从表中读取了一个字段,然后t2在表中插入了一些新的行,之后如果t1再次读取同一个表,就会多出几行

Oracle支持两种隔离级别 read commit 和serializable 默认隔离级别为read commit

Mysql支持四种事务隔离级别,默认的事务隔离级别为repeatable read

查看数据库的隔离级别:

Select @@tx\_isolation;

设置数据可隔离级别

Set session transaction isolation level [read uncommitted];设置当前会话连接有效

Set globe transaction isolation level 隔离级别;

四种隔离级别:

1. read uncommited(读未提交事务):允许事务读取被其他事务提交的变更,脏读,不可重复度,幻读都会出现
2. read commit(读已提交事务) :只允许事务读取已经被其他事务提交的变更,可以避免脏读,但不可重复度和幻读问题仍有可能出现
3. repeatable read(可重复读):确保事务可以多次从一个字段中读取相同的值,但这个事务持续期间,禁止其他事务对这个字段进行更新,可以避免脏读和不可重复度,但幻读仍然存在
4. serializable(串行化):确保事务可以从一个表中读取相同的行,在这个事务持续期间,禁止其他事务对该表执行插入,更新和删除操作,所有并发问题都可以解决

脏读 不可重复读 幻读

read uncommitted √ √ √

read committed × √ √

repeatable read × × √

serializable × × ×

## savepoint

set autocommit=0;

start transaction;

delete from 表名 where id=1;

savepoint a;#设置保存点

delete from 表名 where id=2;

rollback to a;#回滚到保存点;

# 十四 视图

含义

虚拟表 和普通表一样使用

Mysql 5.1版本出现的新特性,是通过动态表生成的数据,只保存了sql逻辑,不保存查询结果

应用场景:多个地方用到同样的查询结果或逻辑比较复杂

当表修改后视图也相应会修改

## 语法:

Create view 视图名 as 查询语句语句

案例:

Create view v1 as 查询语句

使用:select \* from v1 where stuName like %张%;

视图的好处:

重用sql语句

简化复杂的sql操作,不必知道他的查询细节

保护数据,提高安全性

## 视图的修改

方式一:

create or replace view 视图名 as 查询语句

方式二:

alter view 视图名 as 查询语句

## 删除视图

可以多个同时删除

drop view 视图名,视图名,视图名

## 查看视图

desc 视图名

show create view 视图名 \G 格式化

## 视图的更新

对视图的操作会影响原始表,因此我们经常会对视图添加权限

具备以下特点的视图是不能更改的



# 十五 系统变量

## 自定义变量

用户变量和局部变量'

## 系统变量:

### 全局变量

全局变量作用域:服务器每次重启都将为所有的全局变量赋初始值,针对所有的会话(连接)有效,但不能夸重启

如果想夸重启需要修改配置文件

说明:变量由系统提供,不是用户自定义的,属于服务器层面

使用语法:

1. 查看所有的系统变量

show [global|session] variables;

1. 查看满足条件的部分系统变量

show [global|session] variables like %char%

1. 查看指定的某个系统变量

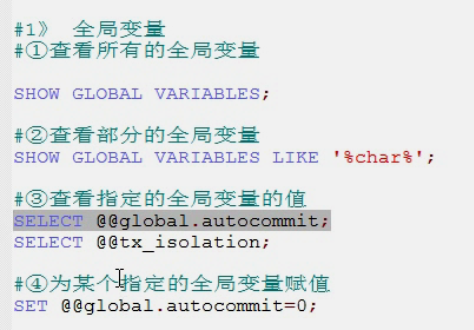
select @@global|session.系统变量名

1. 为某个系统的变零赋值

set global|[session] 系统变量=值

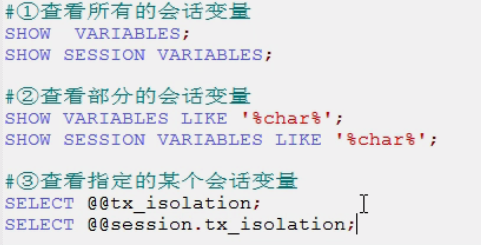
set @@global|[session].系统变零=值

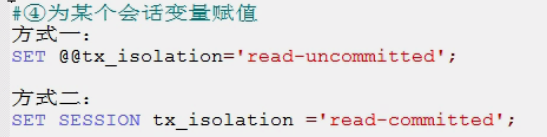
注意:如果是全局级别.需要加global,如果是会话级别,则需要加session,如果不写默认session



### 会话变量:

仅仅针对当前会话(连接)有效





## 自定义变量

说明:变量使用户自定义的,不是系统的

使用步骤:声明 赋值 使用(查看,比较,运算等)

### 用户变量:

针对当前会话有效,等同于会话变量的作用域

#### 声明并初始化:

赋值的操作符有两个: =或:=

set @用户变量名=值;或

set @用户变量名:=值;或

select @用户变量名:=值

#### 赋值(更新用户变量的值)

方式一:

set @用户变量名=值;或

set @用户变量名:=值;或

select @用户变量名:=值

方式二:通过set或select

select 字段 into 变量名 from 表;

#### 使用(查看用户变量的值)

select @用户变量名

### 局部变量

作用域:仅仅在定义它的begin end中

#### 声明

declare 变量名 类型;

declare 变量名 类型 default 值;

#### 赋值

方式一:通过set或select

set局部变量名=值;或

set 局部变量名:=值;或

select 局部变量值:=值;

方式二:通过select 值 into 局部变量名

select 字段 into 局部变量名from 表;

#### 使用

select 局部变量名;

# 十六 存储过程

存储过程和函数

## 好处:

1提高代码的重用性

1. 简化操作
2. 减少了编译次数并且减少了和数据库服务器的连接次数,提高了效率

## 存储过程:

含义:一组预先编译好的sql语句的集合,理解成批处理语句

## 调用语法:

create procedure 存储过程名(参数列表)

begin

存储过程体(一组合法的sql语句)

end

注意:参数列表包含三部分:

参数模式 参数名 参数类型

举例:in stuName varchar(20

参数模式:

in:该参数可以作为输入,也就是该参数需要调用方传入值

out :该参数可以作为输出,也就是该参数值可以作为返回值

inout:该参数既可以作为输入又可以作为输出,也就是该参数即需要传入值.又可以返回值

2.如果存储过程体仅仅只有一句话,begin end 可以省略

存储过程体的每条sql语句的结尾必须加分号.

存储过程的结尾可以使用delimiter重新设置

语法:

delimiter 结束标记

案例:

delimiter $

## 调用语法 :

call 存储过程名(实参列表)

### 空参列表

案例:插入到admin表中五条记录

delimiter $

create procedure myp1()

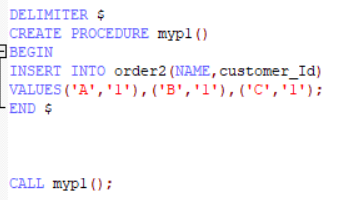
begin

insert into admin(username,password)

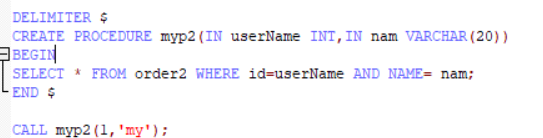
values(“”,””),(‘”,””),(“”,””),(“”,””)

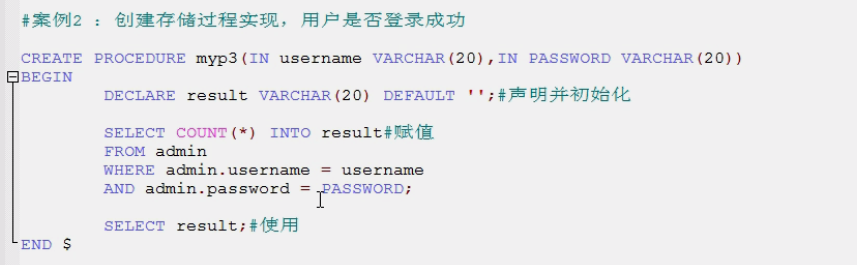
end $

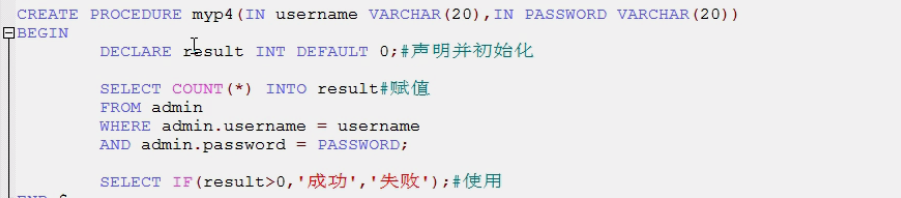
调用;call myp1()$



### 带in模式参数的存储过程







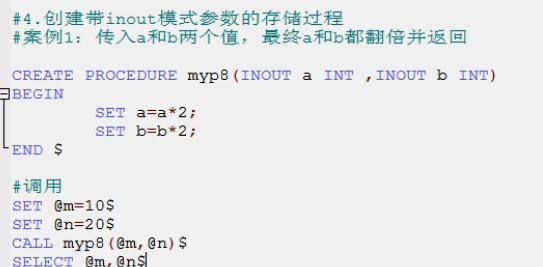
### 带out模式的存储过程



#### 两个out



### 带inout模式参数的存储过程



当使用inout传入值时必须是个变量,因为需要调用输出,如果常量无法调用输出

## 删除存储过程

只能删除单个

drop procedure 存储过程名;

## 查看存储过程信息

show create procedure 存储过程名;

# 十七 函数

含义:一组预先编译好的sql语句的集合,理解成批处理语句

提高代码的重用性

简化操作

减少了编译次数并且减少了和数据库服务器的连接次数,提高了效率

区别

存储过程:可以有0个返回值,也可以有多个返回值,适合做批量插入,批量更新

函数:有且仅有一个返回,适合做数据处理后返回一个结果

## 创建语法:

create function 函数名(参数列表) returns 返回类型

begin

函数体

end

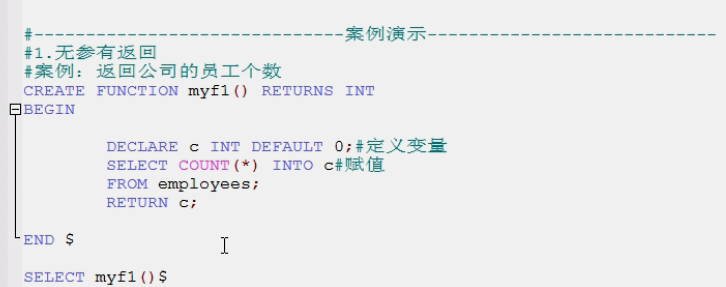
注意:参数列表包含参数名和参数类型两部分

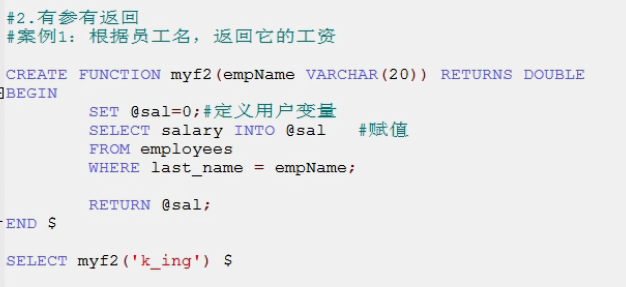
函数体:肯定会有return 语句 如果没有会报错,如果return语句没有放在函数体的最后也不会报错,但是不推荐 return 值;

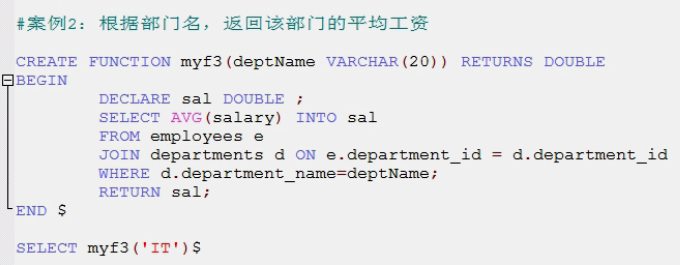
## 调用语法:

select 函数名(参数列表)

## 案例演示





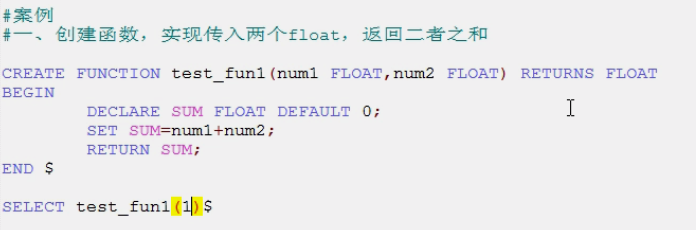


## 查看存储函数

show create function 名称;

## 删除函数

drop function 函数名;

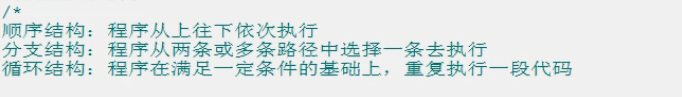


# 流程控制结构

顺序结构

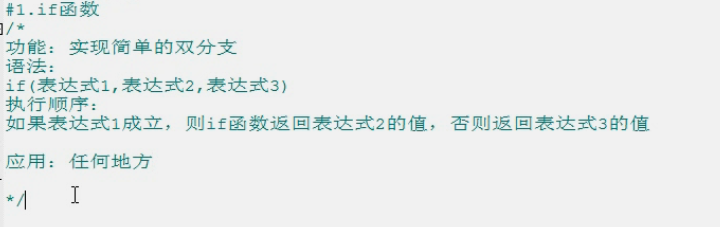
分支结构

循环结构

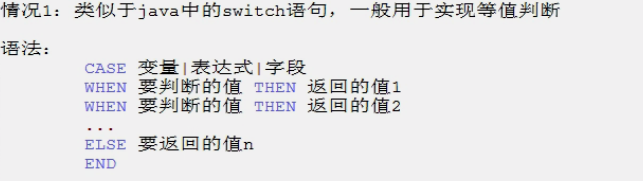


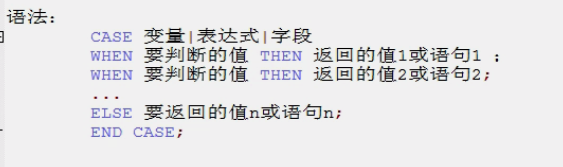
## 分支结构

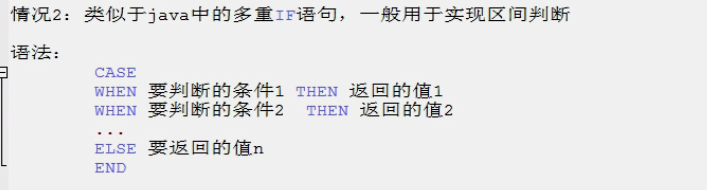
### if函数

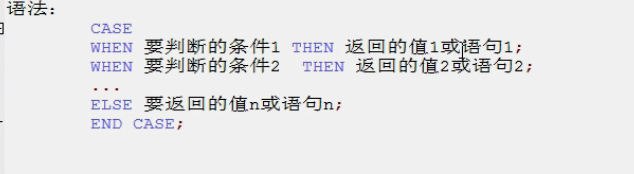


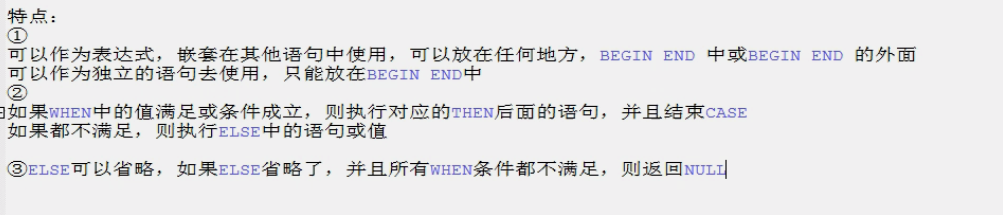
### case结构



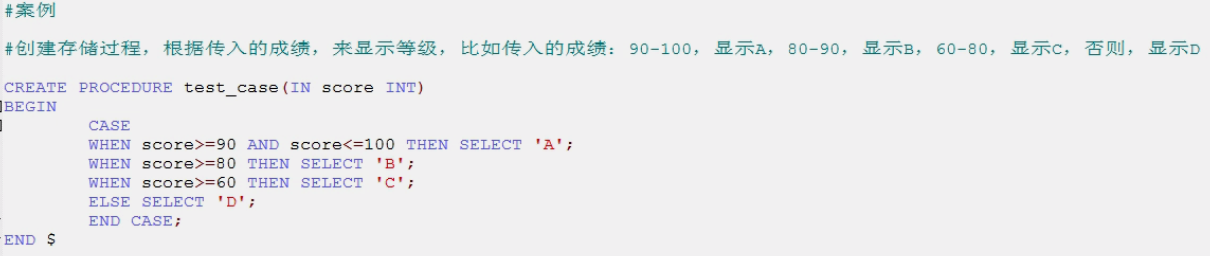




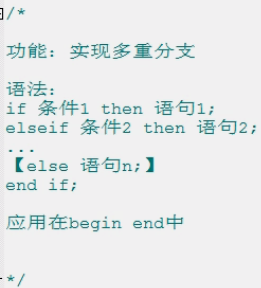




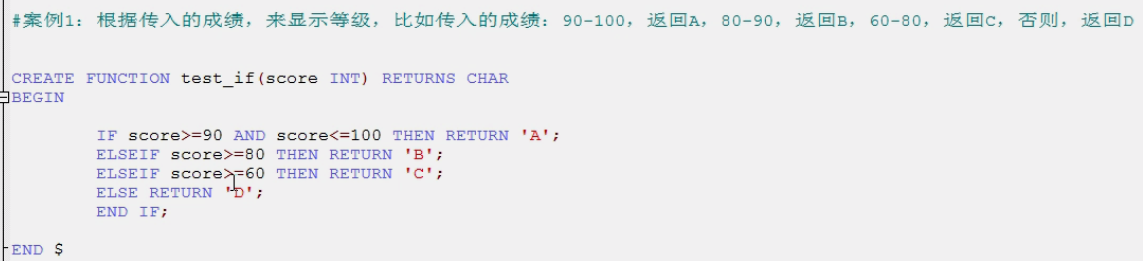
#### 案例



### if结构



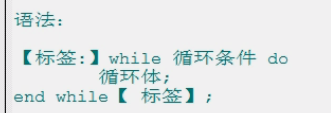
#### 案例



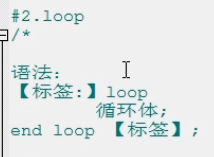
## 循环结构



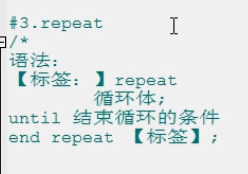
### while



### loop



### repeat



### 案例

## 

