Mysql存储过程及触发器trigger

## 存储过程

### 一个简单的存储过程

1,一个简单的存储过程

delimiter $$

create procedure testa()

begin

Select \* from emp;

Select \* from dept;

End;

$$;

delimiter ;

-- 调用存储过程

call testa();

存储过程的结构组成：

1，创建格式:create procedure 存储过程名

2，包含一个以上代码块，代码块使用begin和end之间

3，在命令行中创建需要定义分隔符 delimiter $$

2,存储过程的特点

1，能完成复杂的判断和运算

2，可编程性强，灵活

3，SQL编程的代码可重复使用

4，执行速度相对快

5，减少网络之间数据传输，节省开销

### 存储过程变量

1，存储过程中的变量

需求：编写存储过程，使用变量取empno=7369的用户名

Delimiter $$;

Create procedure testa();

BEGIN

DECLARE my\_uname varchar(32) default ""; -- 定义变量my\_uname

SET my\_uname='smith'; -- 为变量my\_uname赋值

-- 查询empno=7369的用户名，并将值赋给my\_uname

select ename into my\_uname from emp where empno=7369; -- 为变量赋值

select my\_uname;-- 返回my\_uname的值

END;

$$;

Delimiter ;

**特点：**

1，变量的声明使用declare,一句declare只声明一个变量，变量必须先声明后使用。

2，变量具有数据类型和长度，与mysql的SQL数据类型保持一致，还能指定默认值、字符集和排序规则等。

3，变量可以通过set来赋值，也可以通过select into的方式赋值。

4，变量需要返回，可以使用select语句，如：select 变量名

2，存储过程变量应用示例

**需求：统计表emp、dept的行数和emp表中最早，最晚的入职日期。**

Delimiter $$;

Create procedure stats\_emp();

BEGIN

-- 统计emp和dept表中的记录数

BEGIN

DECLARE emp\_sum int default 0;

DECLARE dept\_sum int default 0;

select count(\*) into emp\_sum from emp;

select count(\*) into dept\_sum from dept;

select emp\_sum,dept\_sum;

END;

-- 统计最早、最晚入职日期

BEGIN

DECLARE max\_time TIMESTAMP ;

DECLARE min\_time TIMESTAMP;

select max(hiredate),min(hiredate) into max\_time,min\_time from emp;

select max\_time,min\_time;

END;

END

$$;

Delimiter ;

### 三、存储过程的参数

**1，存储过程的传入参数IN**

**需求：编写存储过程，传入empno,返回该用户的ename.**

Delimiter $$;

Create procedure test\_param(IN my\_empno int);

--------

BEGIN

DECLARE my\_ename varchar(32) default '';

select ename into my\_ename from emp where empno=my\_empno;

select my\_ename;

END;

$$

Delimiter ;

-- 调用

Call test\_param(7369);

**提示：**

1，传入参数：类型为IN,表示该参数的值必须在调用存储过程时指定，如果不显示指定为IN，那么默认就是IN类型。

2，IN类型参数一般只用于传入，在调用存储过程中一般不作修改和返回。

3，如果调用存储过程中需要修改和返回值，可以使用OUT类型参数。

**2，存储过程的传出参数OUT**

**需求：调用存储过程时，传入empno,返回该用户的ename。**

Delimiter $$;

create procedure test\_param(IN my\_empno int,OUT my\_ename varcahr(32));

--------

BEGIN

select ename into my\_ename from emp where empno=my\_empno;

select my\_ename;

END;

$$

Delimiter ;

-- 调用

Set @uname=’’;

Call test\_param\_out(7369,@uname);

**提示：**

1，传出参数：在调用存储过程中，可以改变其值，并可返回。

2，OUT是传出参数，不能用于传入参数值。

3，调用存储过程时，OUT参数也需要指定，但必须是变量，不能是常量。

4，如果既需要传入，同时又需要传出，则可以使用INOUT类型参数。

**3,存储过程的可变参数INOUT**

需求：调用存储过程时，参数my\_empno和my\_ename,既是传入，也是传出参数。

Delimiter $$;

create procedure test\_param\_inout(INOUT my\_empno int,INOUT my\_ename varchar(32));

BEGIN

set my\_empno=7369;

set my\_ename="smith";

select ename,empno into my\_ename,my\_empno from emp where empno=my\_empno;

END;

$$

Delimiter ;

-- 调用

set @uname:='';

set @empno:=7399;

call test\_param\_inout(@empno,@uname);

select @empno,@uname;

特点：

1，可变变量INOUT,调用时可传入值，在调用过程中，可以修改其值，同时也可以返回值。

2，INOUT 参数集合了IN和OUT类型参数的功能

3，INOUT调用时传入的是变量，而不是常量

### 存储过程条件语句

1，存储过程的条件语句

需求：编写存储过程，如果用户empno是偶数则给出ename,其他情况只返回empno.

Delimiter $$;

create procedure test\_if(IN my\_empno int);

BEGIN

DECLARE my\_ename VARCHAR(32) default '';

if(my\_empno %2=0) then

select ename into my\_ename from emp where empno=my\_empno;

select my\_ename;

else

select my\_empno;

end if;

END;

$$

Delimiter ;

-- 调用

call test\_if(7369);

**特点：**

1，条件语句最基本结构： if() then ... else ... end if;

2,if判断返回逻辑真或者假，表达式可以是任意返回真或假的表达式

2，存储过程的条件语句应用示例

需求：根据用户传入的empno参数判断：

1. 如果用户sal小于2000，则给用户加薪200
2. 如果用户sal小于1000，则给用户加薪500
3. 其他情况加薪100

Delimiter $$;

create procedure test\_if\_else(IN my\_empno int);

BEGIN

DECLARE my\_sal int default 0;

select sal into my\_sal from emp where empno=my\_empno;

if(my\_sal<1000) then

update emp set sal=sal+500 where empno=my\_empno;

ELSEIF(my\_sal<2000)

then update emp set sal=sal+200 where empno=my\_empno;

else update emp set sal=sal+100 where empno=my\_empno;

end if;

END;

$$

Delimiter ;

-- 调用

call test\_if\_else(7369);

特点：

多条件判断结构：

If()

Then

...

Else if()

Then

...

Else

...

End if;

### 五、存储过程循环语句

1，while循环

需求：使用循环语句，向表emp中插入10条empno连续的记录。

Delimiter $$;

create procedure test\_while();

BEGIN

DECLARE i int default 0;

while(i<10) DO

BEGIN

set i=i+1;

insert into acc(id) values(i);

END;

END WHILE;

END;

$$

Delimiter ;

-- 调用

call test\_while();

**特点：**

1，while语句最基本结构： while() do begin ... end end while;;

2,while判断返回逻辑真或者假，表达式可以是任意返回真或假的表达式

2,repeat循环语句

需求：使用repeat循环向表acc插入10条id连续的记录

Delimiter $$;

create procedure test\_repeat();

BEGIN

DECLARE i int default 100;

REPEAT

BEGIN

set i=i+1;

insert into acc(id) values(i);

END;

UNTIL i>=110

END REPEAT;

END;

$$

Delimiter ;

-- 调用

call test\_repeat();

**特点：**

1，repeat语句最基本结构： repeat begin ... end until end repeat;;

2,while判断返回逻辑真或者假，表达式可以是任意返回真或假的表达式

### 六、存储过程游标的使用

1,什么是游标

需求：编写存储过程，使用游标，把uid为偶数的记录逐一更新用户名。

Delimiter $$;

create procedure test\_cursor();

BEGIN

DECLARE stopflag INT DEFAULT 0; -- 游标停止的标记 0：未停止 1：已停止

DECLARE my\_uname VARCHAR(32) default ''; -- 存储查询出的用户名

DECLARE uname\_cursor CURSOR for select uname from acc where uid%2=0; -- 定义游标uname\_cursor，并指定结果集

DECLARE CONTINUE HANDLER for NOT found set stopflag=1; -- 游标结束后stopflag设置为1

open uname\_cursor; -- 打开游标

FETCH uname\_cursor into my\_uname; -- -- 游标向前走一步，取出一条记录放到my\_uname中my\_uname

WHILE(stopflag=0)

DO

BEGIN

update acc set uname=CONCAT(my\_uname,"\_cur") where uname=my\_uname;

FETCH uname\_cursor into my\_uname;

END;

end WHILE;

close uname\_cursor;

END;

$$

Delimiter ;

-- 调用

call test\_repeat();

特点：

declare uname\_cur Cursor for select uname from acc where uid%2=0;

1，游标是保存查询结果的临时内存区域

2，游标变量uname\_cur保存了查询的临时结果，实际上就是查询结果集

Declare continue handler for not found set stopflag=1;

3，当游标变量中保存的结果都查询一遍（遍历），到达结尾，把变量stopflag设为1。

4， FETCH uname\_cursor into my\_uname; -- 游标向前走一步，取出一条记录放到my\_uname中

## Mysql函数

### 一、一个简单函数

需求：编写函数，传入一个uid,返回用户的uname

Delimiter $$;

CREATE FUNCTION f01\_simple(my\_uid int) RETURNS varchar(32) CHARSET utf8

BEGIN

DECLARE my\_uname varchar(32) default '';

select uname into my\_uname from acc where uid=my\_uid;

RETURN my\_uname;

END

$$

Delimiter ;

-- 调用

Select f01\_simple(2);

特点：

1，创建函数使用 create function 函数名(参数) return 返回值

2，函数体放在begin和end之间

3，return 指定函数的返回值

4，函数调用: select 函数名(实参);

### 二、自定义函数综合应用示例

1，自定义函数示例01

需求：输入用户uid,获得accountid,uid,uname组合的uuid值，作为用户的唯一标识。

Delimiter $$;

CREATE FUNCTION test\_uuid(my\_uid int) RETURNS varchar(32) CHARSET utf8

BEGIN

DECLARE uuid varchar(32) default '';

select CONCAT(accountid,"\_",uid,"\_",uname) INTO uuid from acc where uid=my\_uid;

RETURN uuid;

END

$$;

Delimiter ;

-- 调用

Select test\_uuid(2);

2，自定义函数示例02

需求：输入用户uid,计算该uid所在帐号下的所有订单的总价。

## 触发器trigger

场景：通常用 于审计、业务数据完整性

**1，什么是触发器**

需求：处于审计目的，当有人往表users插入一条记录时，把插入的uid,uname和动作及操作时间记录下来。

Delimiter $$;

CREATE TRIGGER `tr\_users\_insert` AFTER INSERT ON `users`

FOR EACH ROW

BEGIN

-- New 插入后当前的用户信息

insert into oplog(uid,uname,action,optime) VALUES(NEW.uid,NEW.uname,'insert',now());

END;

$$;

Delimiter ;

特点：

1，创建触发器使用create trigger 触发器名称。

2，什么时候触发？ after insert on users,处理after还有before ,是在对表操作之前(before)或者之后(after)触发动作的。

3，对什么操作事件触发？After insert on users,操作时间包括insert,update,delete

4,对什么表触发？ after insert on users

5,影响的范围？For each row

触发器：与函数、存储过程一样，触发器是一种对象，它能根据对表的操作事件，触发一些动作，这些动作可以是insert,update,delete等修改操作。

**2，生产环境触发器实例**

需求：处于审计目的，当删除users表时，记录删除前该记录的主要字段值。

Delimiter $$;

CREATE TRIGGER `tr\_user\_delete` BEFORE DELETE ON `users` FOR EACH ROW begin

-- OLD对字段表更新前的数据

insert into oplog(uid,uname,action,optime,old\_value) values(OLD.uid,OLD.uname,"delete",now(),OLD.regtime);

END;

$$;

Delimiter ;