昨日回顾

如何来表示两个表之间的关系。在一个表中新建一个字段，用这个字段存储与之相关的记录的id。

外键是用于强制约束关系。

创建：

foreign key(外键字段) references表名(主键)

级联与置空约束

on update cascade

on delete set null

子查询：

最终得到的数据是来源于一个表。使用一个表的数据，来限制另一个表中能被查询出来的数据。

分类：

子查询的位置：

where子查询

from子查询

子查询得到的数据：

标量子查询 1个字段 = 1个值

列子查询 1个字段 in 多个值

行子查询 (n个字段) = (n个值)

注意：以上有的三个子查询是where子查询，就是放在where表达式中。

表子查询 会作为一个数据源使用。

连接查询：

内连接：

最终的结果是两个表中，连接条件匹配的记录连接为一条。

隐式内连接

select \* from 表A,表B where 连接条件

显式内连接

select \* from 表A 【inner】 join 表B where 连接条件。

外连接：

最终的结果不但包含两个表中，连接条件匹配的记录，而且还包含不匹配的记录。

左外连接

select \* from 表L left join 表R on 连接条件

右外连接

select \* from 表L right join 表R on 连接条件

一、视图view

1、视图简介

View 视图 视窗

视图是MySQL中一种特殊的表。

创建语句：

视图的创建并不是由真正创建表的语句得到的，而是由select语句得到

表结构：

视图是MySQL中的一种特殊的表，视图的结构是由创建视图的select语句所查询出来的字段所创建。

视图的数据：

视图并没有专属于自己的数据，视图的数据来源于创建视图的select语句所查询的表。

2、创建视图

语法：

create 【or replace】 【algorithm=算法】 view 视图名 as select语句

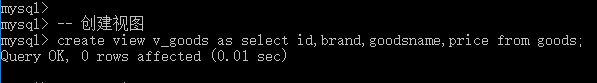
说明：

用于创建一个视图。

【or replace】 用于创建新视图或修改一个已存在的视图

【algorithm=算法】 用于设置视图算法

示例：



3、查看视图

①、查看所有的视图

语法：

show tables;

②、查看视图的创建语句

语法：

show create view 视图名;

示例：

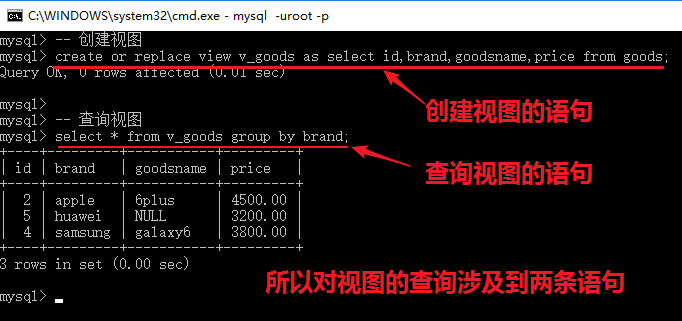


4、视图操作

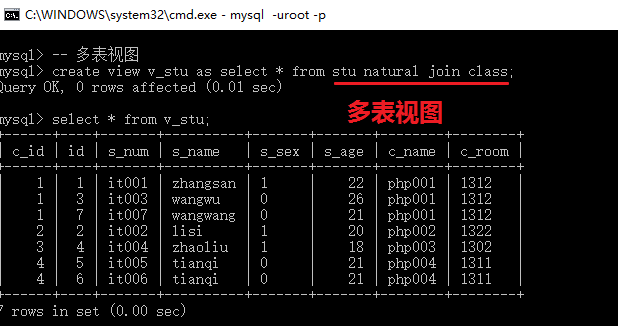
由于视图没有专属于自己的数据，所以对数据的操作绝大多数都是查询操作，也可以进行其他的操作。如果进行其他的操作会有很多的限制条件。

由于视图也是一个表，对表的查询查询，同样适用于视图。

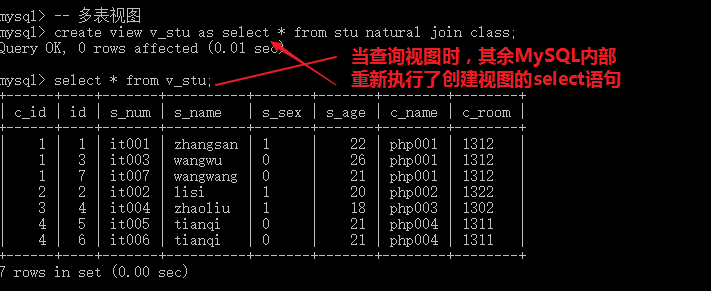
示例：



扩展：



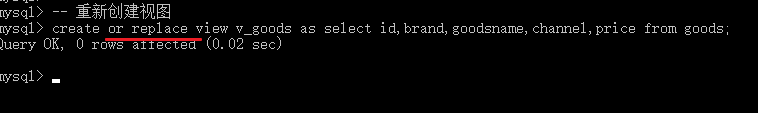
个人的理解视图就是使用一个视图名，保存一条复杂功能强大的select语句。当我们对视图的查询，就触发了那条sql语句的执行。



5、修改视图

修改视图，主要是使用or replace重新定义视图

示例：



6、删除视图

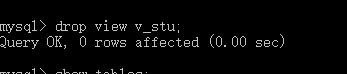
语法：

drop view 视图名;

说明：

查看视图使用是show tables，但是删除视图必须使用drop view。

示例：



7、视图算法

①、首先需要明确

与视图相关的两条select语句，其一是创建视图的select语句，其二是查询视图的select语句。

所谓的视图算法，表示的就是这两条select语句的执行过程。

②、视图算法分类

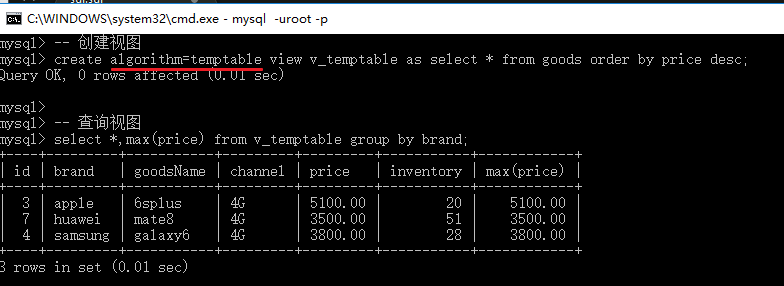
temptable 临时算法

要查询视图时，先执行创建视图的select语句，在内存中会产生一个临时表，之后在这个临时表的基础上再执行查询视图的sql。

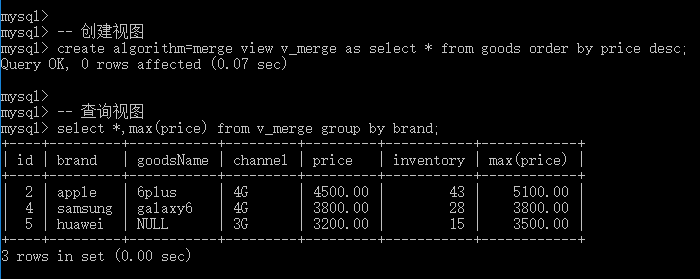
merge 合并算法

要查询视图时，会对创建视图的select语句与查询视图的select语句进行合并为一个select语句，再执行。

示例：temptable算法



示例：merge算法



小节：

视图就是使用一个表存储一个复杂的select语句。

视图算法表示的是两条select语句的执行顺序。主要影响在查询视图的select语句中的5子句，位于创建视图的select语句的之前。

二、触发器

1、触发器的概念

触发器就是由某一种操作，触发执行一段代码。此处的操作包含：insert、update、delete。所要触发执行的代码就是触发器中的代码。

2、触发器的要素

触发事件：insert、update、delete

触发时机：before、after

触发对象：数据表

范围： 每一条记录

3、创建语法

语法：

create trigger 触发器名 时机 事件 on 触发对象 for each row

begin

触发器所要执行的代码

end;

示例：

第1步：更改客户端命令结符

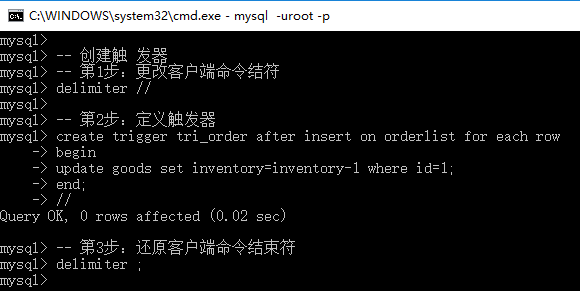
delimiter #

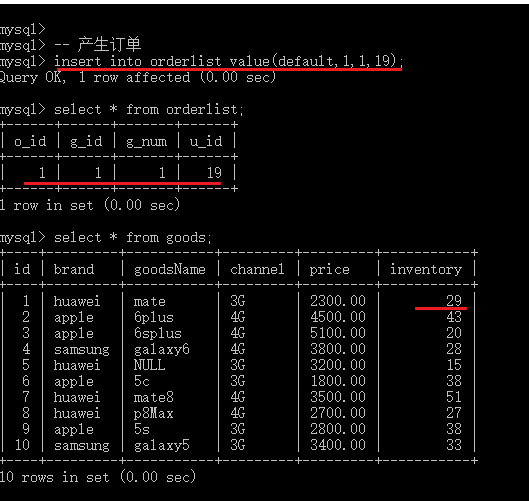
第2步：定义触发器

第3步：还原客户端命令结束符

delimiter ;

示例：



测试：向订单表中插入数据，  


4、触发器记录对象

触发器中的old、new，关键字

分代表触发器所触发的表中所影响的记录。

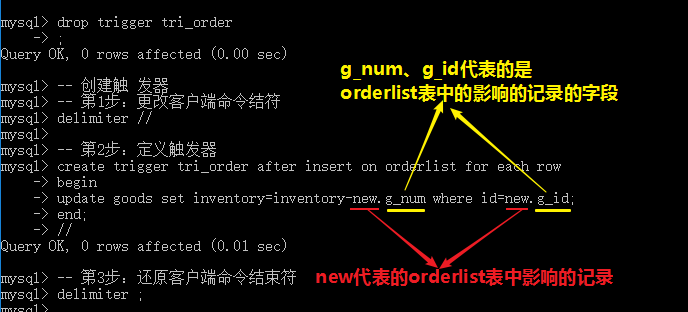
old new

update 原记录 修改之后的记录

delete 原记录 null

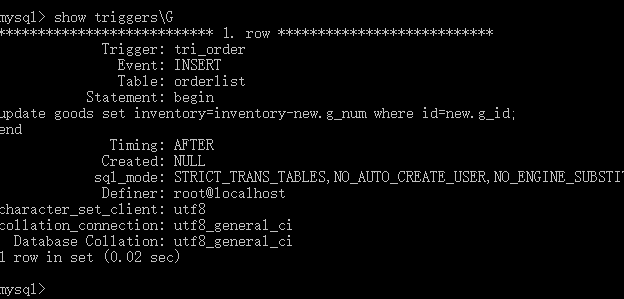
insert null 新增记录

示例：



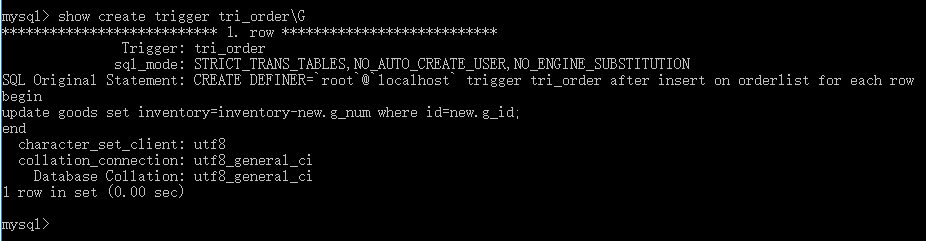
5、查看触发器

①、查看触发器



②、查看触 发器的创建语句

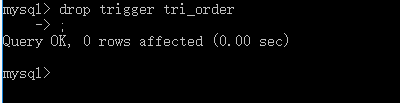
show create trigger 触 发器名



6、删除触发器

语法：

drop trigger 触 发器名



小节：

触发器相关的有两个表：其一是触 发表(orderlist)，其二是触发器操作的表(goods)

触发器的定义是定义在触发表上，触发器操作的表是操作表

old new代表的是触发表上的受影响的记录，与触 发事件有关。

old new

update 原记录 修改之后的记录

delete 原记录 null

insert null 新增记录

三、事务

1、事务

事务是MySQL中的一种机制，主要用于保证在使用多条SQL语句完成，现实中的一件事时，保证结束一致。

现实中的一个场景：

zhangsan向lisi转账100元，只要两种状态，要么成功，在么失败。

在使用MySQL来解决这样的问题：

需要两种SQL

sql1 update bank set balance=balance-100 where id=1;

sql2 update bank set balance =balance+100 where id=2;

会有以下四种情况：

sql1 成功 sql2 失败

sql1 失败 sql2 成功

sql1 成功 sql2 成功

sql1 失败 sql2 失败

2、事务操作

事务操作分两类： 手动事务、自动事务

①、手动事务

第一步：开启事务

start transaction;

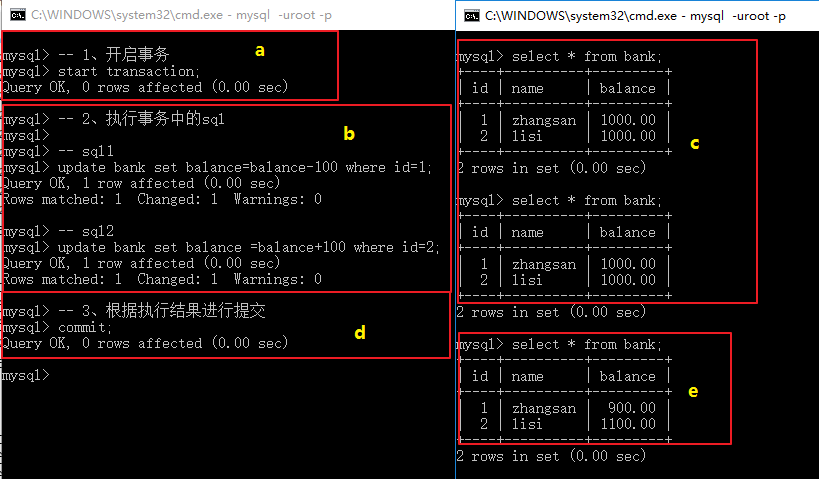
第二步：执行事务中的sql

第三步：根据事务中的sql的执行结束，进行提交或回滚。

commit; 将事务中多条sql的影响写入数据库

rollback; 将事务中多条sql的影响删除。

示例：成功



示例：失败



②、自动事务

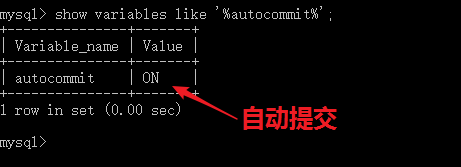
MySQL默认处于自动事务状态(自动提交)，在完成绝大多数的需求时，都是一条SQL语句来解决。

查看事务状态

语法：

show variables like ‘%autocommit%’;

示例：



4、回滚点

语法：

savepoint 回滚点名;

说明：

主要用于在事务的执行过程中，某一条语句定义一个回滚点(还原点)

回滚到指定的回滚点

rollback【to 回滚点名】;

示例：



提示：

手动事务的结束，当执行了commit或rollback 时,手动事务就会结束，而rollback to …并不会结束手动的事务。

5、事务的特性ACID(了解)

Atomicity 原子性

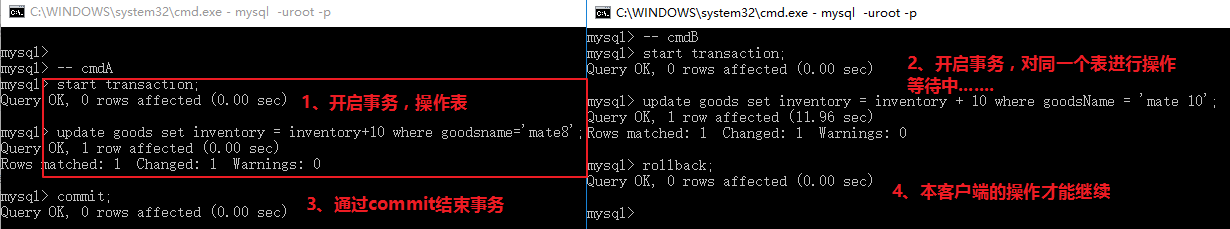
一个事务一个不可分割的工作单位，这个单位中包含多个sql，要么全部成功，要么全部失败。

Consistency 一致性

事务必须使用数据库从一个一致状态变到另一个一致状态。一致性与原子性是密不可分的。

Isolation 隔离性

一件事务的执行不受其他事务的影响。即在一个事务内执行的操作或数据对并发执行的另一个事务是隔离的，相到之交并不会受干扰(在一个事务执行操作的过程中，阻止另一个事务进行操作，而实现不干扰自己——解决方案就是事务锁的)。



Durability 持久性

持久性也就是永久性，一个事务一旦提交，对数据库的更改就是永久的。

四、变量

MySQL是用于存储数据，但是也是一门程序语言，同样存在变量、流程控制、函数、存储过程。

1、MySQL的系统变量

①、查看系统变量

语法：

show variables 【like pattern】;

说明：

使用模糊查询来查看系统变量。



②、修改系统变量的值

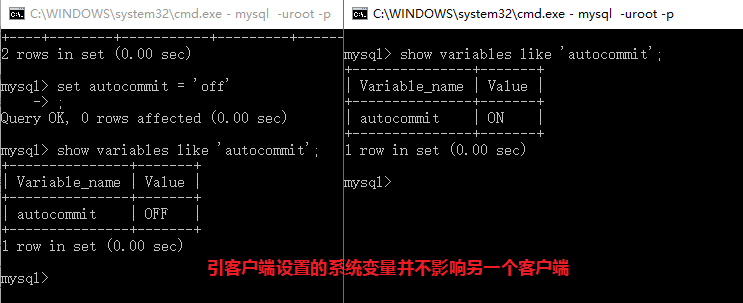
语法：

set 系统变量名=值;

说明：

系统变量的设置仅影响当前客户端。

示例：



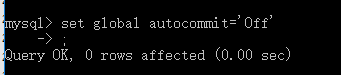
③、全局系统变量的设置

语法：

set global 系统变量名=值;

说明：

用于影响所有的客户端，对已经连接进来的并不受影响。



2、自定义变量

①、定义

语法：

set @变量名=值;

示例：



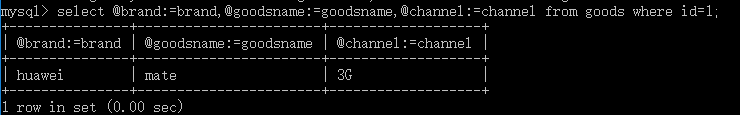
②、查看自定义的变量值

语法：

select @自定义变量名：



③、使用字段为变量赋值



五、函数

1、系统函数

字符串处理函数

LTrim()： 去除左边的空格

RTrim()： 去除右边的空格

Upper()： 返回大写字符。Strtolower.

Lower()： 返回小写字符。strtoupper

Char\_length()： 返回字符的个数。

Left(str,len) ： 返回串左边指定数目的字符.

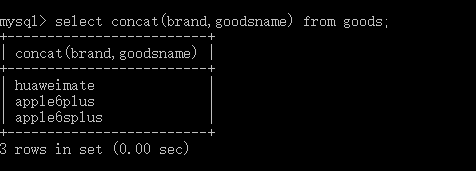
Right(str,len)： 返回串右边指定数目的字符

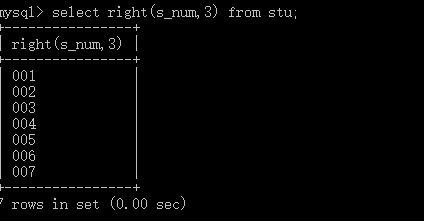
concat(str1,str2)： 连接子字符串

SubString(str,start[,len])：返回子串的字符，mid（）同名函数。(substr)

位置从1开始。

Instr(父串，子串)：第一个出现的位置。位置从1开始。没有找到，返回0.类似于





日期时间函数

* now():获取当前时间日期
* Date() ：返回日期时间的日期部分.当前日期curdate().
* Time() ：返回一个日期的时间部分.当前时间curtime().
* DateDiff() ：求两个日期的差
* Year()：返回一个日期的年份部分
* Date\_add(日期，interval 数字 单位)，向日期添加时间间隔

数学函数

ABS(x)   返回x的绝对值

CEILING(x)   返回大于x的最小整数值  
FLOOR(x)   返回小于x的最大整数值

GREATEST(x1,x2,...,xn)返回集合中最大的值  
LEAST(x1,x2,...,xn)      返回集合中最小的值

MOD(x,y)                 返回x/y的模（余数）   
RAND()返回０到１内的随机值  
ROUND(x,y)返回参数x的四舍五入的有y位小数的值

其他函数

**Md5()**：32位加密函数。主要用于对密码字段进行加密

Version（）：查询版本号。

Datebase():当前选择数据库

2、自定义函数

①、自定义函数

语法：

create function 函数名(变量 类型,变量 类型) returns 返回值的类型

begin

函数体

return 值;

end;

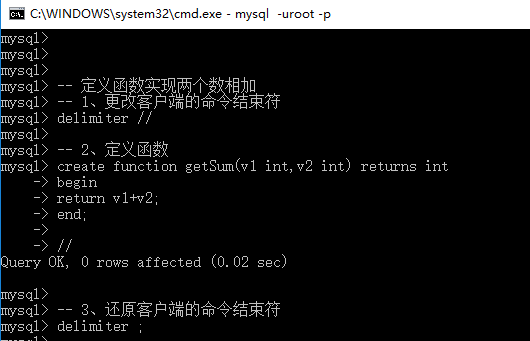
说明：

由于MySQL是一个强数据类型语言，形参必须指定类型

类型就是列类型

函数必须有return,但不能有select

示例：

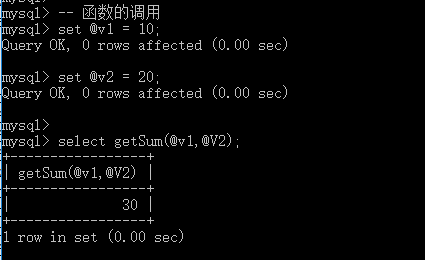


②、函数的调用

语法：

select 函数名(实参)

示例：



③、作用域

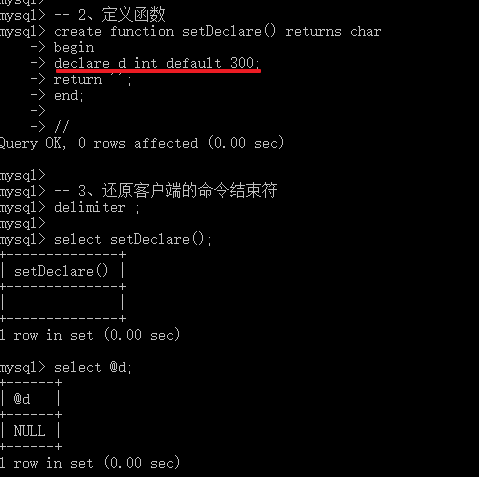
MySQL的作用域分两种，全局，局部

只要使用set定义的变量就是全局变量，其作用域就是全局作用域。

使用declare定义的变量就是局部变量，其作用域局部作用域。

语法：

declare 变量名 类型 【default 值】

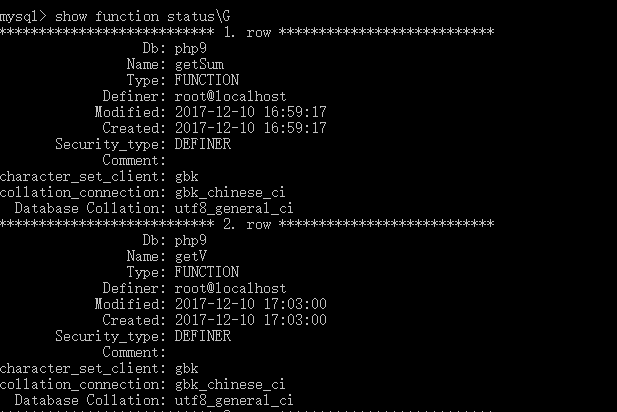


④、查看函数

语法：

show functions;

示例：



3、流程控制

①、if结构

语法：

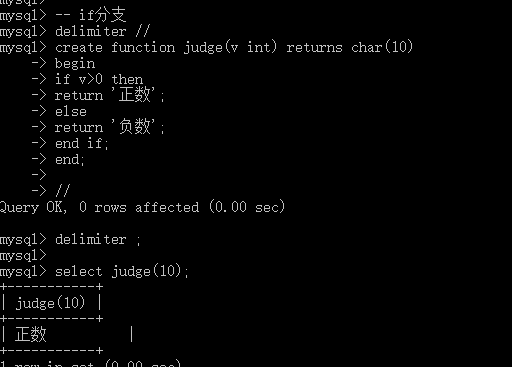
if 表达式 then

begin

else

endif;

示例：



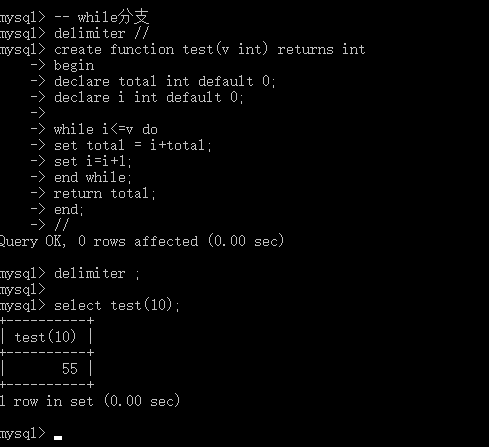
②、while循环结构

语法：  
 while 表达式 do

循环体

end while;

示例：



六、存储过程procedure

1、简介

存储过程是一个特殊的函数。

2、创建存储过程

语法：

create procedure 过程名(方向 变量1 类型,方向 变量2 类型……)

begin

过程体

end;

说明：

不能有return 但，可以有select

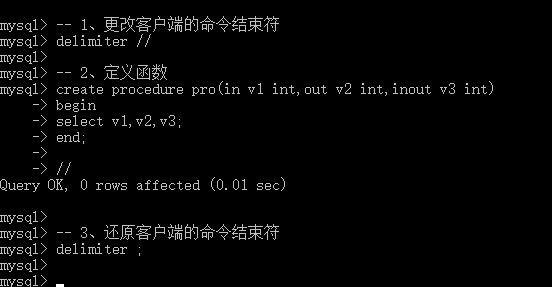
方向包含in、out、inout

in (只能传入，不能传出)只能将变量值传递进去，但过程内对In方向的变量的更改，并不会传出来。

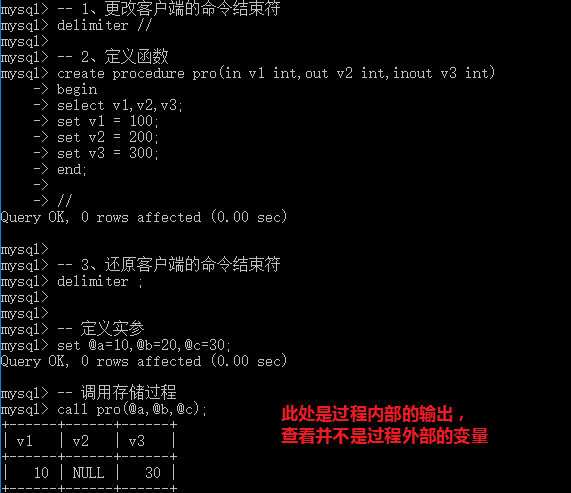
out (只能传出，不能传入)数据不能传递进去，但是过程内对out方向的变量的更改，可以传出来

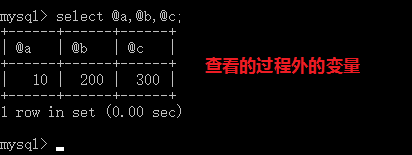
inout 即能传入，又能传出

示例：



示例：



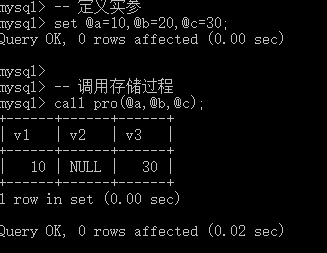


3、调用存储过程

语法：

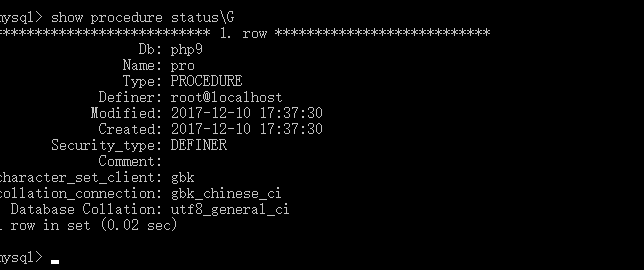
call 过程名(实参)

示例：



4、查看存储过程

show procedure status;



5、删除存储过程

语法：

drop procedure 过程名

