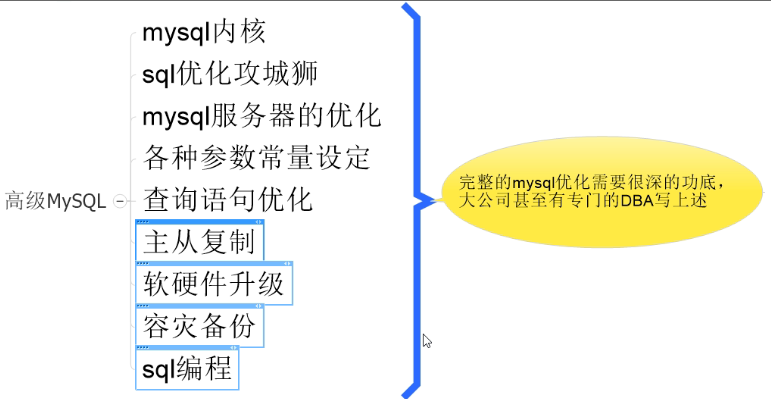
# MySQL高级



目录

[MySQL高级 1](#_Toc463626516)

[一、MySQL视图 3](#_Toc463626517)

[1、什么是MySQL视图 3](#_Toc463626518)

[2、MySQL视图的创建 3](#_Toc463626519)

[3、查看MySQL视图 3](#_Toc463626520)

[4、查看MySQL视图的基本信息 3](#_Toc463626521)

[5、查看MySQL中的所有视图 4](#_Toc463626522)

[6、查看MysQL视图的详细信息（如何创建的视图） 5](#_Toc463626523)

[7、MySQL视图的修改 5](#_Toc463626524)

[8、使用SQL语句访问MySQL视图 5](#_Toc463626525)

[9、删除MySQL视图 5](#_Toc463626526)

[10、视图的应用场景 6](#_Toc463626527)

[11、视图的两种算法（merge与temptable） 8](#_Toc463626528)

[二、触发器trigger 9](#_Toc463626529)

[1、什么是触发器 9](#_Toc463626530)

[2、触发器的四要素 9](#_Toc463626531)

[3、创建触发器的基本语法 9](#_Toc463626532)

[4、触发器的应用场景 10](#_Toc463626533)

[5、触发器中的两个参数(new与old) 11](#_Toc463626534)

[6、查询触发器 12](#_Toc463626535)

[7、删除触发器 13](#_Toc463626536)

[三、MySQL中事务处理 13](#_Toc463626537)

[1、什么是事务处理 13](#_Toc463626538)

[2、事务处理的应用场景（与钱相关的事务） 13](#_Toc463626539)

[3、解决之道，使用事务处理 14](#_Toc463626540)

[4、在MySQL使用事务处理 14](#_Toc463626541)

[5、在PHP中使用事务处理（PDO扩展库） 15](#_Toc463626542)

[6、ThinkPHP中的事务处理 16](#_Toc463626543)

[四、MySQL中的函数 16](#_Toc463626544)

[1、MySQL中函数的类别 16](#_Toc463626545)

[2、MySQL中的系统函数 17](#_Toc463626546)

[3、创建MySQL自定义函数 19](#_Toc463626547)

[4、MySQL自定义函数的参数 19](#_Toc463626548)

[5、编写函数，求一个数的立方。(5 \* 5 \* 5) 20](#_Toc463626549)

[6、自定义函数中的分支结构 20](#_Toc463626550)

[7、自定义函数中的循环结构 21](#_Toc463626551)

[8、循环中的continue与break（iterate与leave） 21](#_Toc463626552)

[9、查看已创建的自定义函数 22](#_Toc463626553)

[10、删除自定义函数 23](#_Toc463626554)

[五、MySQL中的存储过程 23](#_Toc463626555)

[1、什么是存储过程 23](#_Toc463626556)

[2、创建存储过程 23](#_Toc463626557)

[3、存储过程的参数 24](#_Toc463626558)

[4、In/Out/Inout区别 24](#_Toc463626559)

[5、查看存储过程 26](#_Toc463626560)

[6、删除存储过程 26](#_Toc463626561)

# 一、MySQL视图

## 1、什么是MySQL视图

MySQL5中添加了对视图的支持。

视图就是一个存在于数据库中的虚拟表，而不是真实的表。

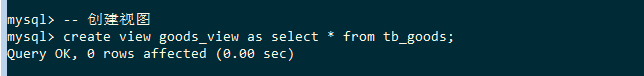
视图本身没有数据，只是通过执行相应的select语句完成获得相应的数据。

## 2、MySQL视图的创建

基本语法：

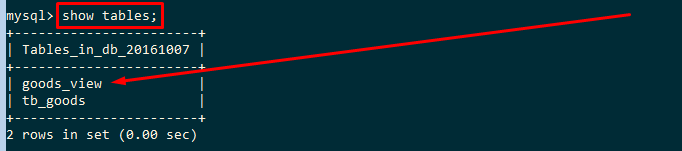
create view 视图名称 as select 语句;

示例代码：



## 3、查看MySQL视图

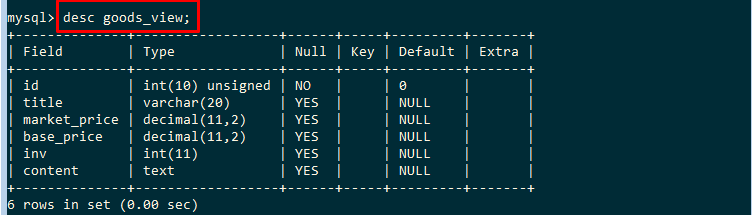
记住：虽然MySQL视图是一张虚拟表，但是其表现形式和一张真实表是一致的，如下图所示：



## 4、查看MySQL视图的基本信息

基本语法：

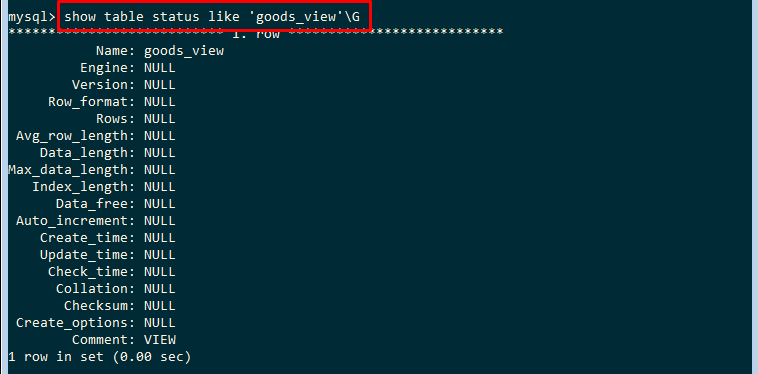
desc 视图名称;



基本语法：

show table status like ‘视图名称’;

示例代码：

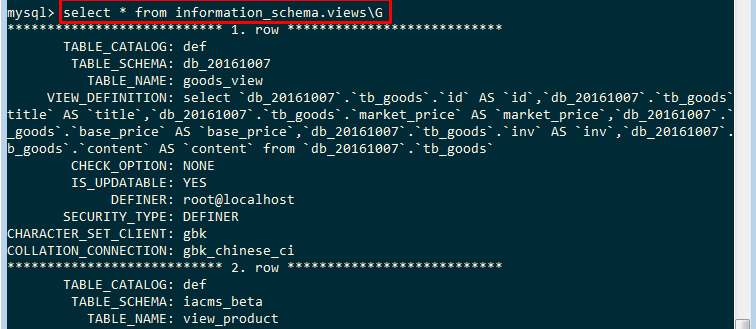


## 5、查看MySQL中的所有视图

基本语法：

select \* from information\_schema.views;

示例代码：

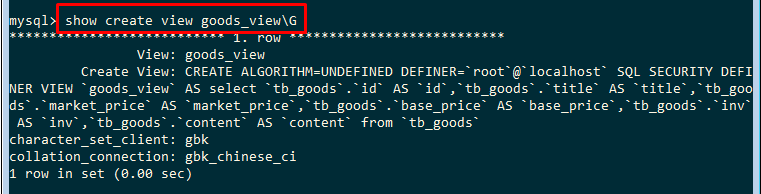


## 6、查看MysQL视图的详细信息（如何创建的视图）

基本语法：

show create view ‘视图名称’;

示例代码：



## 7、MySQL视图的修改

基本语法：

alter view ‘视图名称’ as select 语句;

示例代码：

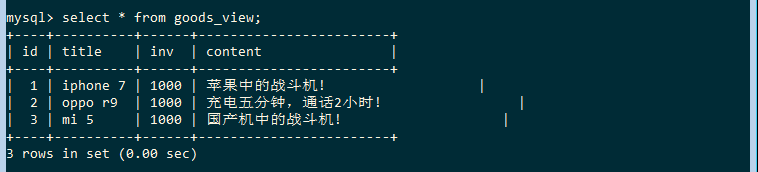


## 8、使用SQL语句访问MySQL视图

基本语法：

select \* from 视图名称;

示例代码：

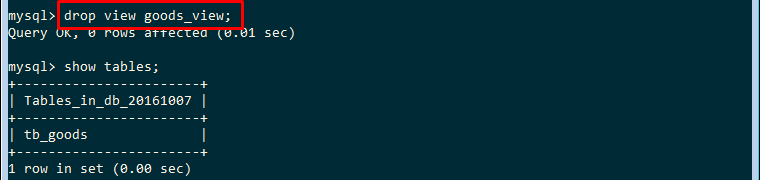


## 9、删除MySQL视图

基本语法：

drop view ‘视图名称’;

示例代码：



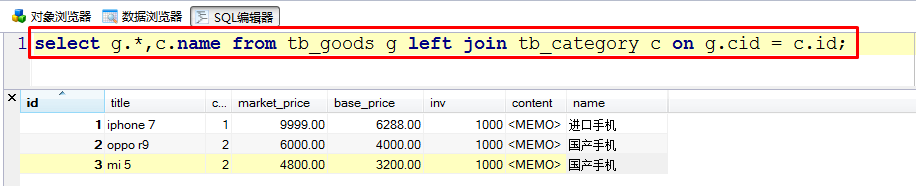
## 10、视图的应用场景

① 重用SQL语句

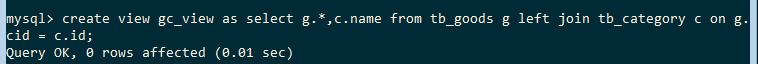
视图就是一个虚拟表，其实际存储的就是一个select语句，所以当下次访问视图时就相当于访问上次封装的SQL语句，提高了SQL语句的重用性!

② 简化复杂的SQL语句

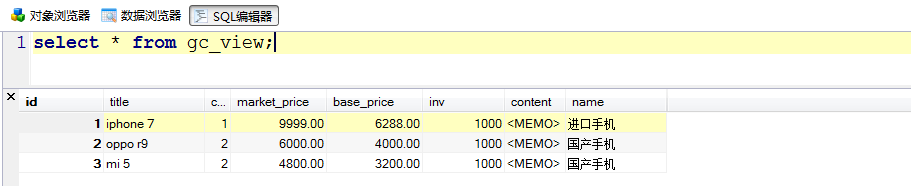
原SQL语句：



封装成视图：

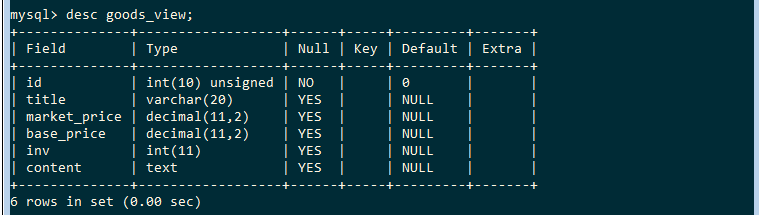


访问视图：



③ 使用表的组成部分而不是整个表

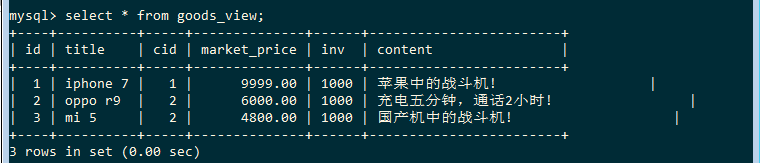




④ 保护数据，授予表的特定部分的访问权限而不是整个表的访问权限



访问效果如下图所示：



⑤ 水平分表垂直分表（大项目）

水平分表就是把一张大表分解为几张一摸一样的小表，然后使用取模算法，把要插入的数据平均分配到每一张小表中。（tb\_goods1,tb\_goods2,tb\_goods3,tb\_goods4）

SQL语句：

select \* from tb\_goods1

union

select \* from tb\_goods2

union

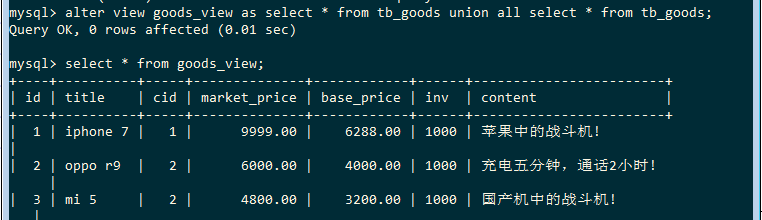
select \* from tb\_goods3

union

select \* from tb\_goods4;

封装成视图：

create view tb\_goods as select \* from tb\_goods1 union select \* from tb\_goods2 union tb\_goods3 …;



垂直分表就是把一张大表中的字段进行拆分（常用的字段放在主表中，不常用的字段放在从表中），这种就是垂直分表。

例：tb\_user表

id、username、password、truename、sex、age、address

垂直分表：tb\_user表

id、username、password

垂直分表：tb\_user\_info表

id、truename、sex、age、address

封装成SQL视图：

create view user\_view as select u.\*,i.truename,i.sex,i.age,i.address from tb\_user u,tb\_user\_info i where u.id = i.id;

## 11、视图的两种算法（merge与temptable）

在MySQL视图中，一共有三种算法，分别为：

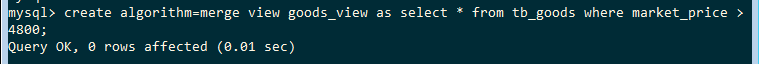
UNDEFINED ：未定义算法，默认算法，让系统自动选择相应算法（严格来说UNDEFINED并不是一个算法，只是一种选择机制）

MERGE ：合并算法，先组装后执行，把要查询的SQL语句与视图的SQL语句进行合并操作

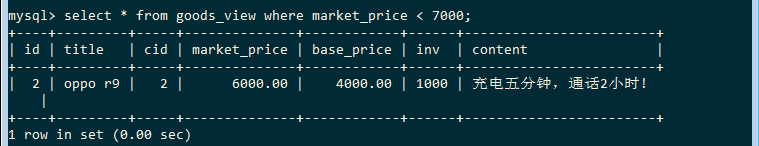
TEMPTABLE ：先执行视图语句，后进行查询查询语句

MERGE算法：合并算法

① 封装一个视图并指定算法为Merge



② 定义一个查询语句



MERGE算法的执行流程：

第一步：先组装

select \* from tb\_goods where market\_price > 4800 and market\_price < 7000;

第二步：执行并获取结果

…

TEMPTABLE算法与MERGE算法的区别：

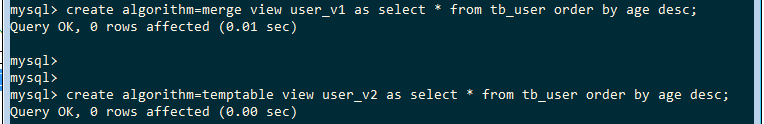
需求：求每个级别下的年龄最大的会员（基本v1，v2，v3）

原生SQL语句：

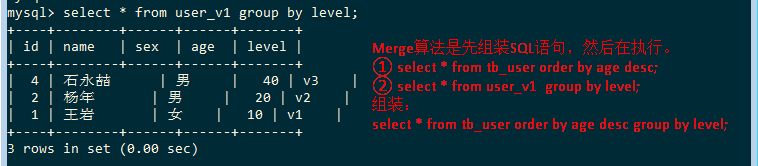
1)先排序select \* from tb\_user order by age desc;

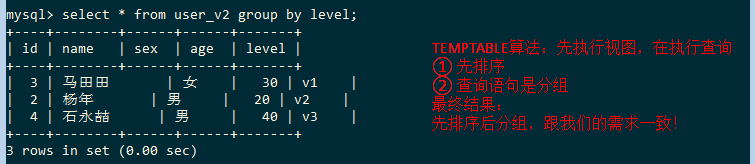
2)后分组select \* from (select \* from tb\_user order by age desc) as user group by level;

① 创建视图



② 对视图进行分组





# 二、触发器trigger

## 1、什么是触发器

触发器TRIGGER是由事件来触发某个操作，这些事件包括INSERT语句、UPDATE语句和DELETE语句。当数据库系统执行这些事件时，就会激活触发器执行相应的操作。

## 2、触发器的四要素

① 触发器的名称

② 触发时间，一般有两种形式：before与after

③ 触发事件，insert、update、delete

④ 触发对象，对应的是哪个表

## 3、创建触发器的基本语法

基本语法：

create trigger 触发器名称 before或after insert/update/delete

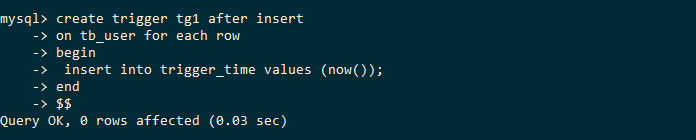
on 表 for each row（固定语法，代表每一行）

begin

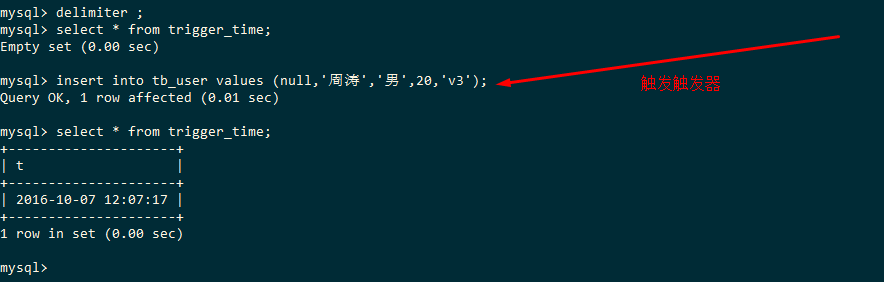
//执行语句;

end

示例代码：



运行结果：



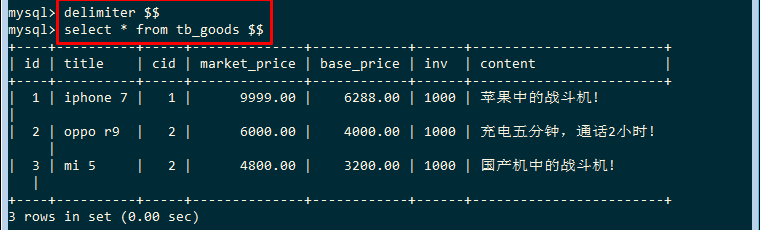
要用到的知识点：在MySQL中，更改SQL语句的结束符，默认为分号；其也可以自定义：

基本语法：

delimiter 结束符($、//、$$)

特别说明：delimiter的生命周期是session会话级别的，当cmd窗口关闭时，其也会随之消失！

示例代码：



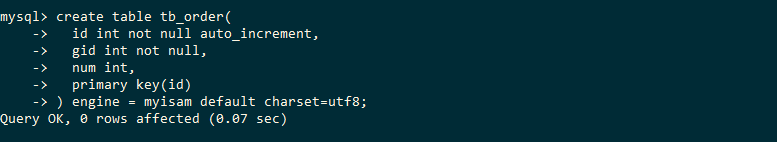
## 4、触发器的应用场景

以电商项目为例：tb\_goods产品表/tb\_order订单表

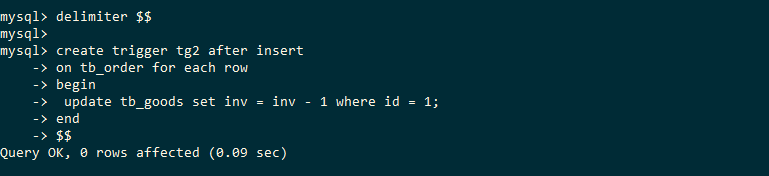
执行流程：① 当用户单击购买按钮时，系统会自动把用户购买的产品插入到订单表中

② 当订单记录生成时，应该自动对产品中的对应产品进行减法操作，减掉订单对应的产品库存。

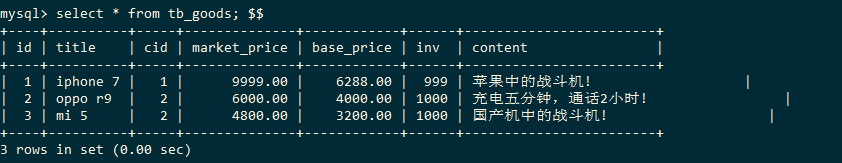
示例代码：创建一个订单表



示例代码：创建触发器



示例代码：触发触发器



## 5、触发器中的两个参数(new与old)

在触发器中，当触发事件发生时，系统会自动产生两个变量（不一定所有事件都有）：

new ：代表新行

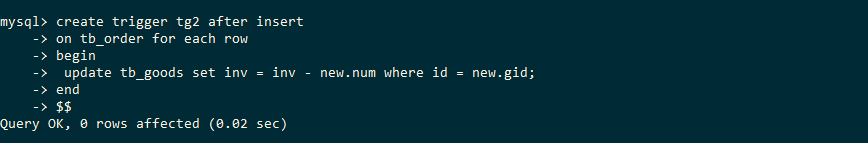
old ：代表旧行

① insert操作，成功之后，其会自动产生一行新的数据，在触发器可以通过new来访问新行，但是insert操作没有old。

② update操作，成功之后，其会对原有的旧行数据进行更新成新行数据，所以在触发器，更新操作既有new也有old。

③ delete操作，成功之后，其会对旧行数据进行删除操作，在触发器中可以通过old来获取删除行的信息，但是delete操作没有new。

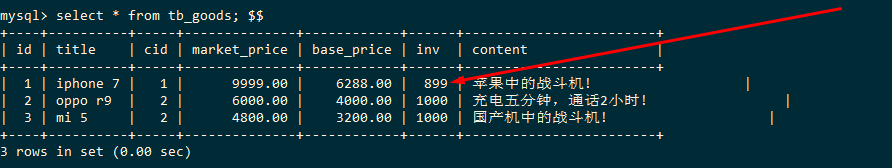
示例代码：使用new来解决订单与产品库存问题



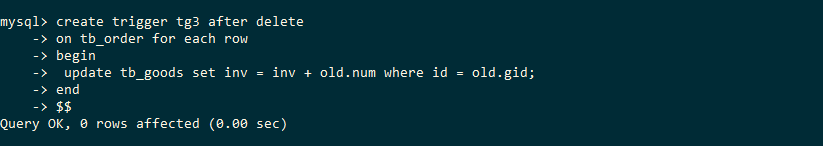
调用触发器：



运行结果：



示例代码：使用old来解决订单与产品库存问题



调用撤单触发器：



运行结果：

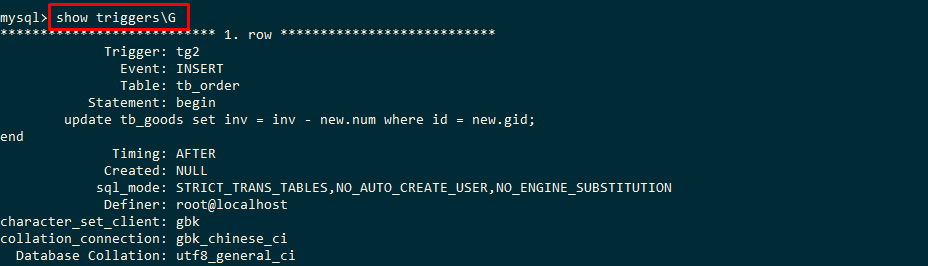


## 6、查询触发器

① 查询当前表对应的所有触发器，基本语法：

show triggers;

示例代码：

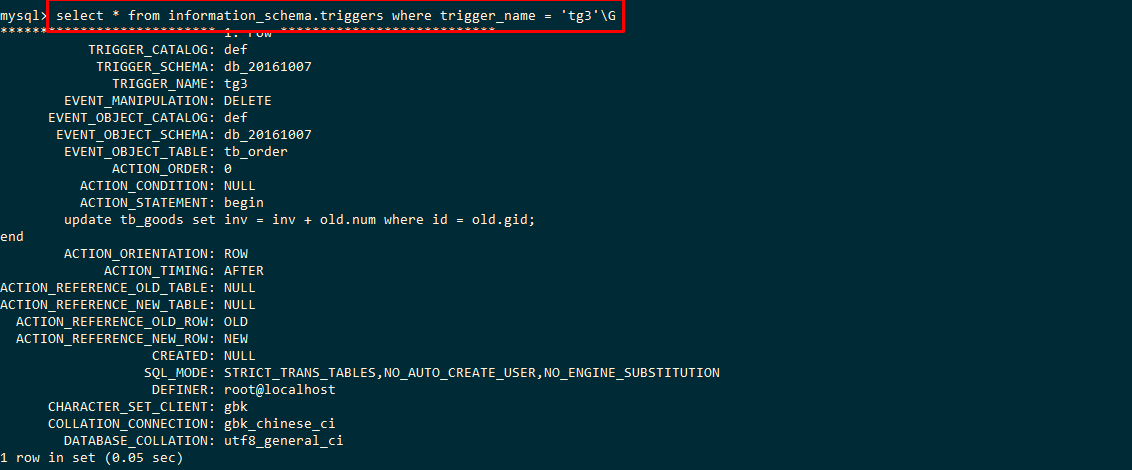


② 查询MySQL中所有的触发器或指定的触发器

基本语法：

SELECT \* FROM information\_schma.triggers where TRIGGER\_NAME=‘触发器名称’;

示例代码：



## 7、删除触发器

基本语法：

drop trigger ‘触发器名称’;

示例代码：



# 三、MySQL中事务处理

## 1、什么是事务处理

在MySQL5以后版本中，其提供了一种保证数据安全性与稳定性的保护机制，叫做事务处理！

## 2、事务处理的应用场景（与钱相关的事务）

涛哥：月初，钱不够，需要借钱，支付宝：0.1

赵刚：刚发工资，有钱，支付宝：2000

① 赵刚借钱给涛哥，共1000元。实行转账机制，理想执行流程如下：

赵刚 🡪 2000 🡪 扣钱1000元 🡪 支付宝还剩1000元 ：update操作

涛哥 🡪 0.1 🡪 加钱1000元 🡪 支付宝还剩1000.1元 ：update操作

② 赵刚借钱给涛哥，共1000元。实行转账机制，不理想执行流程如下：

赵刚 🡪 2000 🡪 扣钱1000元 🡪 支付宝还剩1000元 ：update操作

转账过程 🡪 涛哥 🡪 突然支付宝服务器停电了，问题来了：

赵刚支付宝已被扣款，但是涛哥支付宝并没有增加任何钱，钱哪去了…

## 3、解决之道，使用事务处理

事务处理必须要保证事务的一致性：① 要么成功，要么失败 ② 如果全部成功，则认为交易成功 ③ 如果只要一方失败，则认为整个交易失败。

## 4、在MySQL使用事务处理

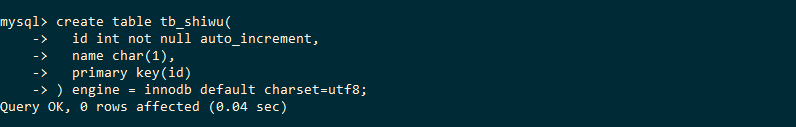
开启事务（start transaction或set autocommit=0）

成功则提交事务（commit）

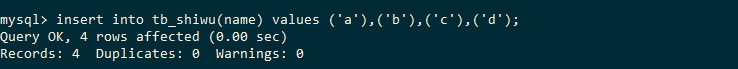
失败则事务回滚（rollback）

特别说明：事务处理只能在InnoDB引擎中使用，Myisam引擎不支持事务处理！

① 创建数据表



② 插入测试数据



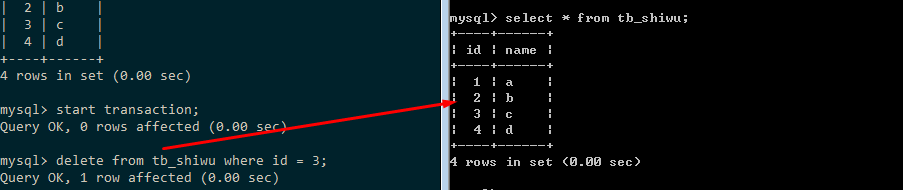
③ 开启事务



④ 删除某条记录

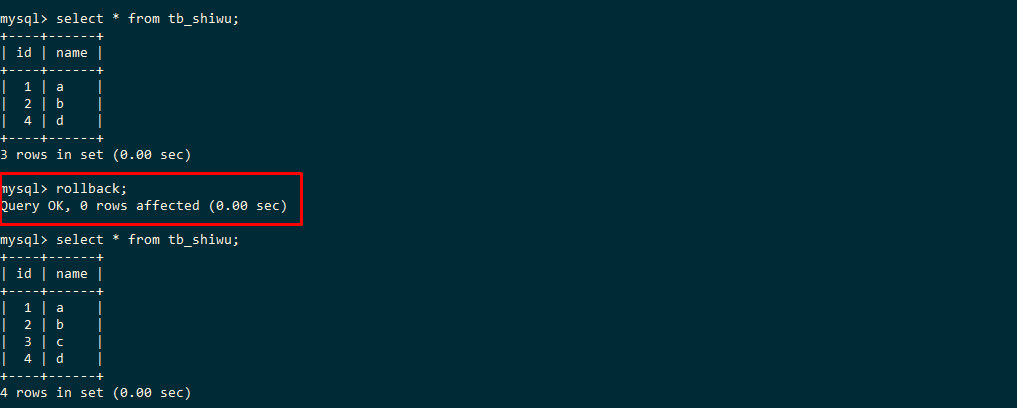


⑤ 打开另外一个cmd窗口，查看tb\_shiwu表，如下图所示：

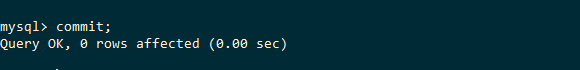


由上图可知，当开启事务后，其所有的MySQL操作都是发生在计算机内存中而并没有反馈到数据库的文件中。

⑥ 在事务处理中，突然发现删除错了数据，怎么办呢？答：使用rollback回滚到最初状态



⑦ 如果确认我们之前的SQL操作，则可以使用commit提交事务，代表整个流程执行完毕！



## 5、在PHP中使用事务处理（PDO扩展库）

要用到的基本语法：

bool PDO::beginTransaction ( void ) ：开启事务

bool PDO::commit ( void ) ：成功，则commit

bool PDO::rollback ( void ) ：失败，则rollback

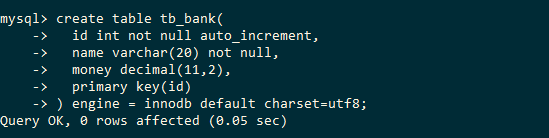
示例代码：

① 在php.ini中开启pdo的扩展库

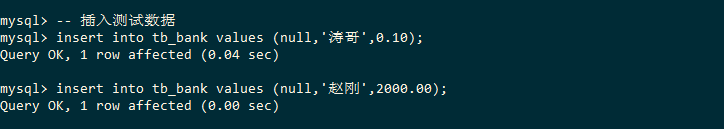


② 一定要重启Apache，否则无法生效

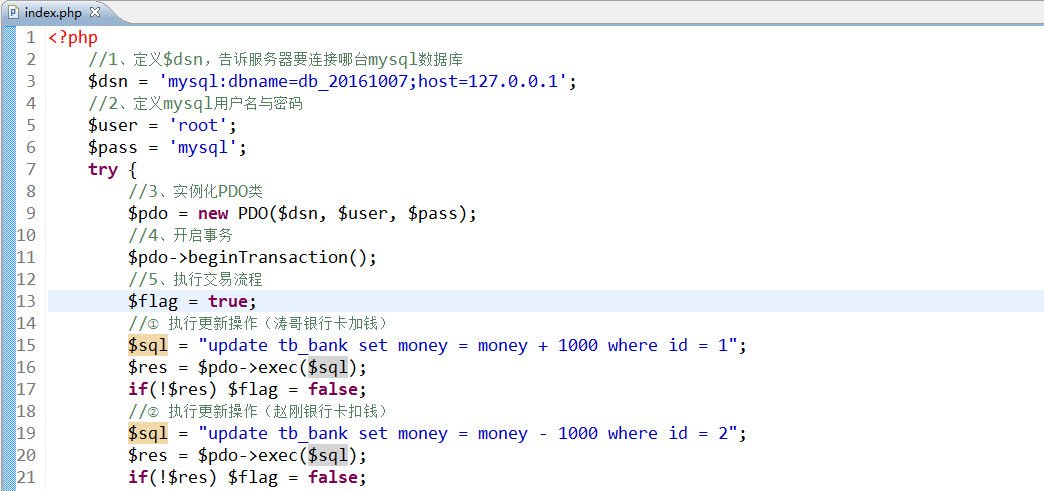
③ 创建数据表

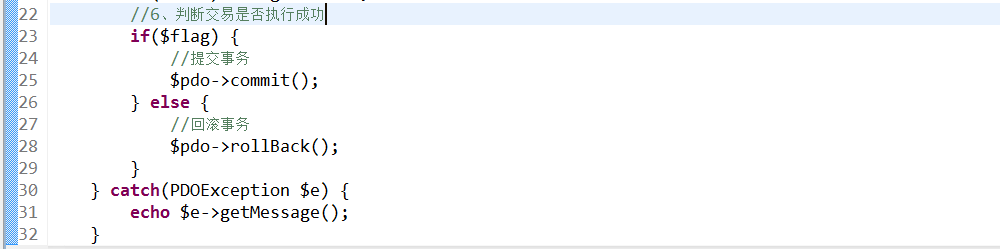


插入测试数据：



④ 编写PHP代码





## 6、ThinkPHP中的事务处理

基本语法：



# 四、MySQL中的函数

## 1、MySQL中函数的类别

在MySQL中，函数一共有两大类：① 系统函数 ② 自定义函数

## 2、MySQL中的系统函数

① 文本处理函数

LTrim()：去除左边的空格

RTrim()：去除右边的空格

Upper()：返回大写字符

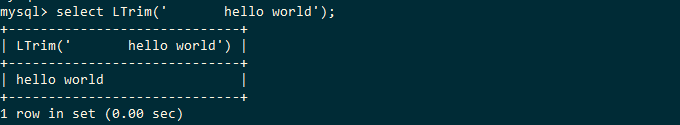
Lower()：返回小写字符

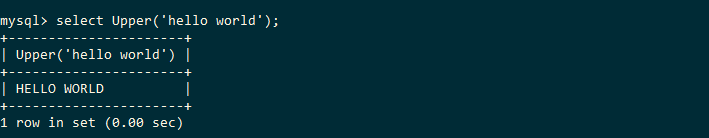
Left() ：返回串左边的字符

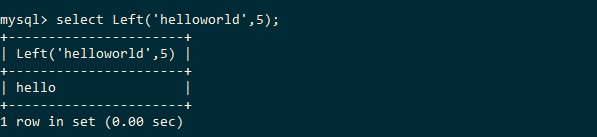
Right()：返回串右边的字符

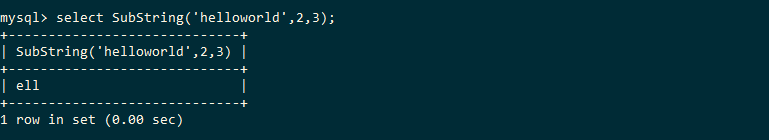
SubString()：返回子串的字符，在MySQL中，索引下标是从1开始的

示例代码：









② 日期与时间函数（2016-10-07 16:16:00）

Date() ：返回日期时间的日期部分（年-月-日）

Time() ：返回一个日期的时间部分（小时:分钟:秒）

DateDiff() ：求两个日期的差

Year()：返回一个日期的年份部分

Day() ：返回一个日期的天数部分

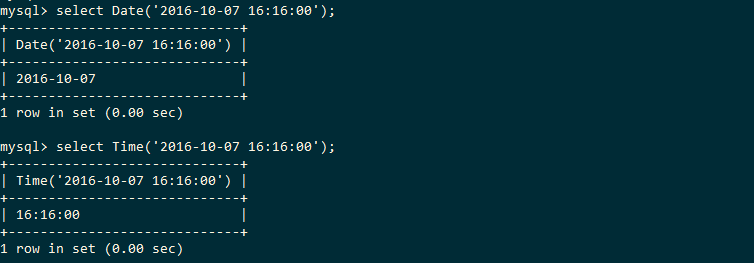
Hour() ：返回一个日期的小时部分

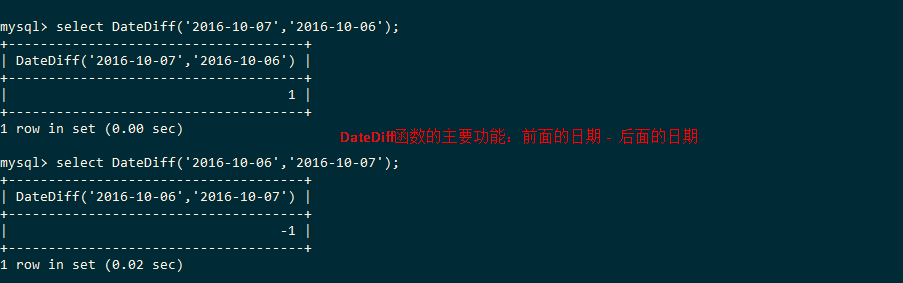
Minute() ：返回一个日期的分钟部分

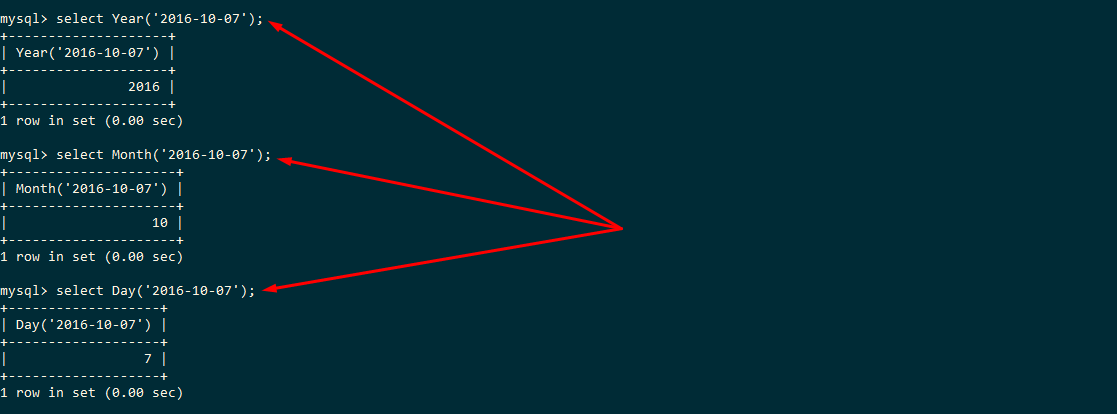
Month() ：返回一个日期的月份部分

Second() ：返回一个日期的秒部分

示例代码：





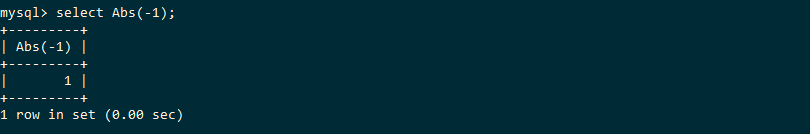


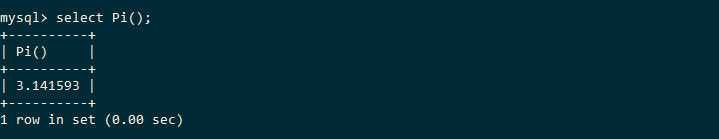
③ 数学函数

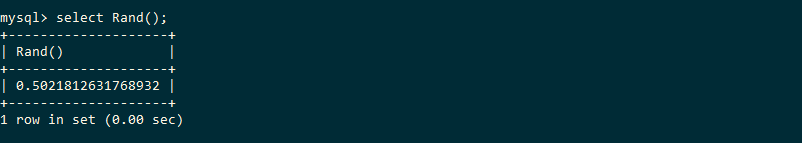
Abs() ：求绝对值，正数的绝对值是它身，负数的绝对值是它的相反数

Pi() ：求圆周率，3.141593…

Rand() ：求随机数，返回0-1之间的随机数（不建议使用，因为效率比较低）







## 3、创建MySQL自定义函数

① MySQL自定义函数要素：

关键词（function）、函数名、函数参数（形参与实参），返回值，函数体

② 创建自定义函数，基本语法：

CREATE FUNCTION 函数名称(参数列表)

RETURNS 返回值类型

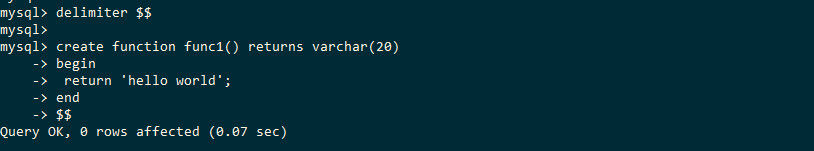
BEGIN

--函数体

--RETURN 返回值

END

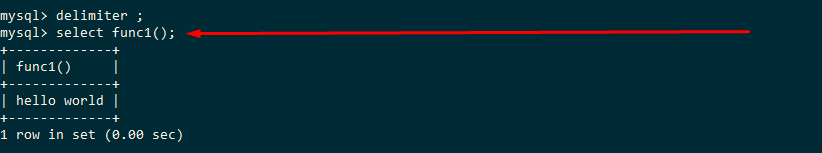
示例代码：定义func1()函数



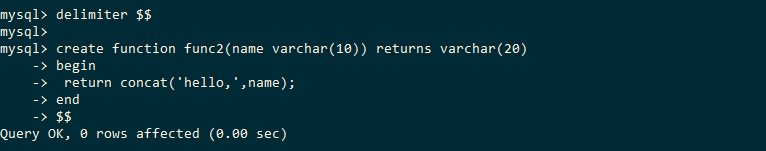
③ 调用自定义函数，基本语法：

select 函数名称(实参);

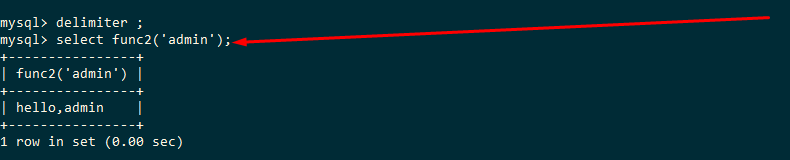
示例代码：



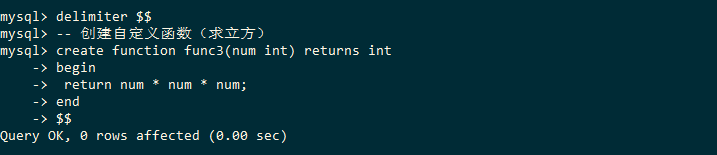
## 4、MySQL自定义函数的参数



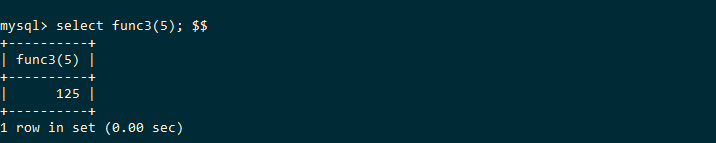
调用带参数的函数：



## 5、编写函数，求一个数的立方。(5 \* 5 \* 5)



调用func3函数：



## 6、自定义函数中的分支结构

基本语法：

if 分支条件 then

-- 相关SQL语句;

else if 分支条件 then

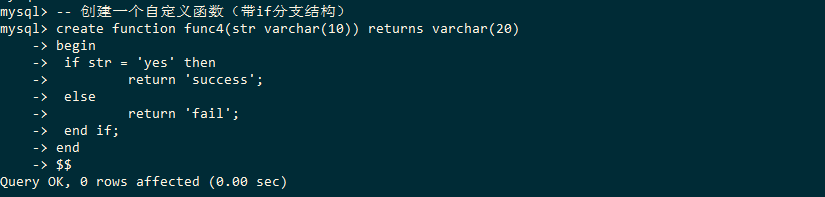
-- 相关SQL语句;

else

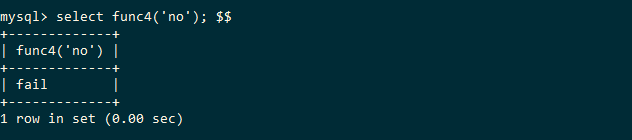
-- 相关SQL语句;

end if;

示例代码：



调用函数：



## 7、自定义函数中的循环结构

基本语法：

[标签名:]while 循环条件 do

-- 定义变量

declare 变量名称 变量类型 default 默认值; --局部变量

set @变量名称 = 变量的值; --全局变量

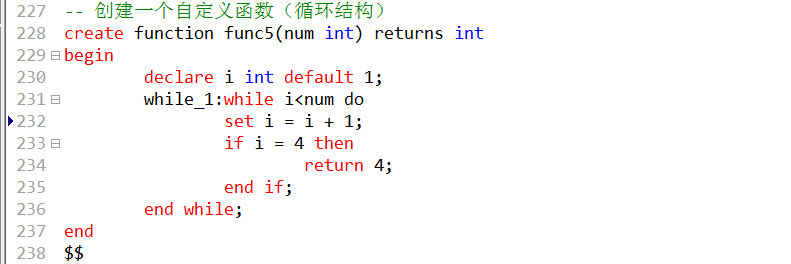
-- 循环体;

iterate 标签名; -- 相当于循环中的continue

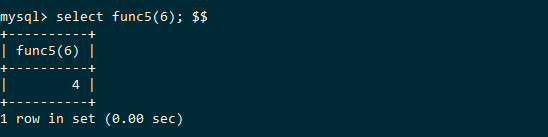
leave 标签名; --相当于循环中的break

end while;

示例代码：

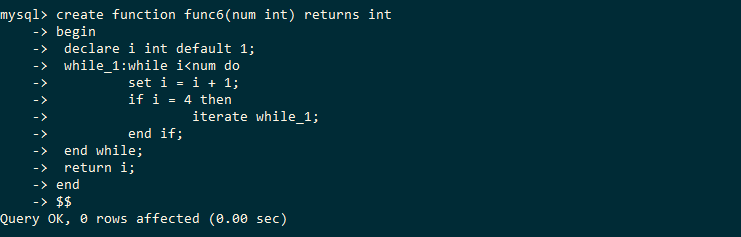


调用函数：

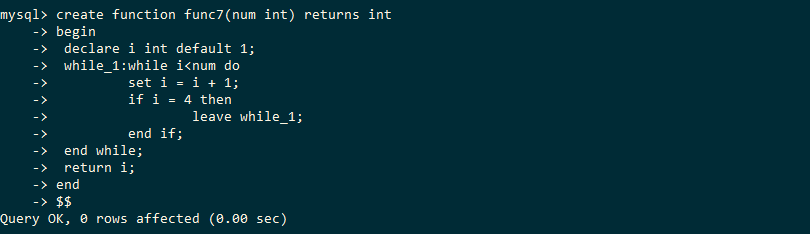


## 8、循环中的continue与break（iterate与leave）

continue：中止当前循环，继续下一次循环



break：中止整个循环



特别说明：如果在循环过程中出现了死循环，可以通过如下方式进行处理

在mysql中，如果出现了死循环，则在cmd窗口中使用net stop mysql中止mysql，再次启动后，找到对应的函数，删除即可。

另外一种解决办法：强制中止循环的执行

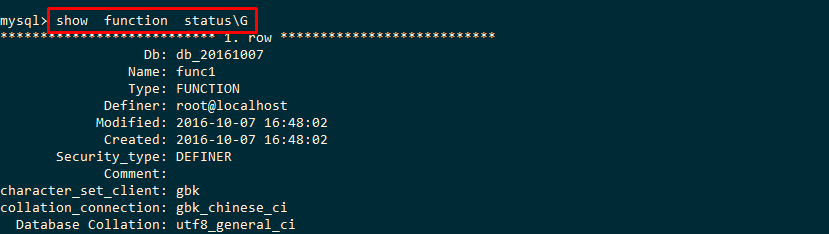
如果执行陷入死循环，并一直卡在cmd窗口，我们可以使用Ctrl + C强制中止循环的继续执行！

## 9、查看已创建的自定义函数

① 显示所有的自定义函数

show function status\G

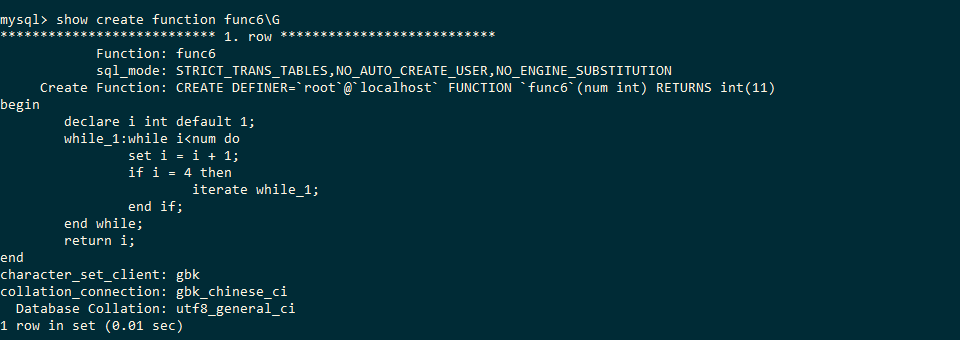
show function status like ‘模糊条件’; 显示指定的函数



② 显示自定义函数的详细信息

SHOW CREATE FUNCTION ‘函数名’;

示例代码：



## 10、删除自定义函数

基本语法：

DROP FUNCTION ‘函数名称’;

示例代码：



# 五、MySQL中的存储过程

## 1、什么是存储过程

存储过程：procedure也叫作过程，是一种没有返回值的函数。

函数是为了解决数据操作，返回结果的话；存储过程就是为了解决数据操作，但是没有结果。

## 2、创建存储过程

① 存储过程的定义

基本语法：

CREATE PROCEDURE 存储过程名称(参数)

BEGIN

//SQL语句

END

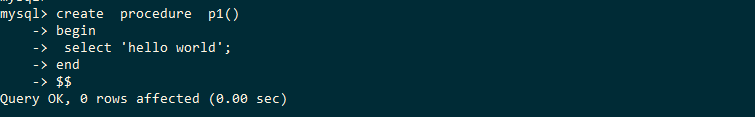
$$

② 调用存储过程

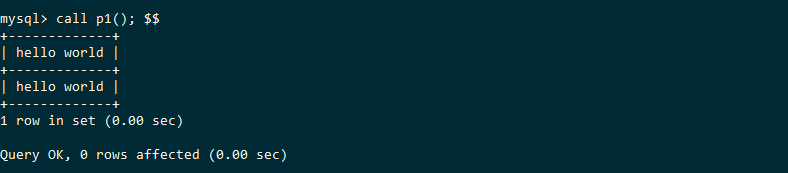
基本语法：

call 存储过程名称(实参);

示例代码：



调用存储过程：



## 3、存储过程的参数

函数中的形参格式：

变量名称 变量类型;

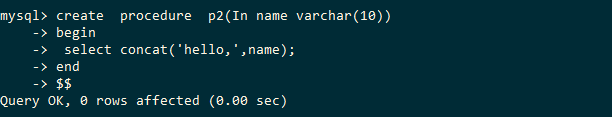
如func1(num int);

存储过程中的形参格式：

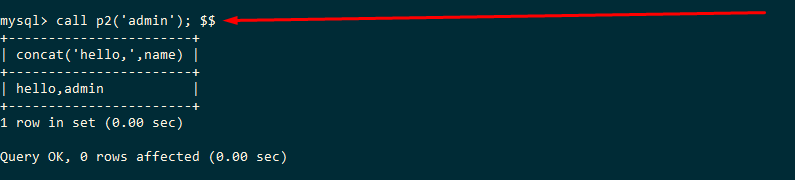
In/Out/Inout 变量名称 变量类型;

如p1(In num int);

示例代码：



调用语法：



## 4、In/Out/Inout区别

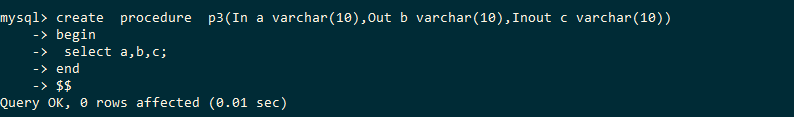
In：传进去，外部（实参）将数据传递给存储过程内部使用（内部可以使用）：但是对外部变量没有影响（值传递）

Out：传出去，外部只需要对内部提供一个变量名，内部会自动实现清空变量=null，然后内部对变量进行操作最后反映到外部的变量身上（会内部事先清空变量引用传递）

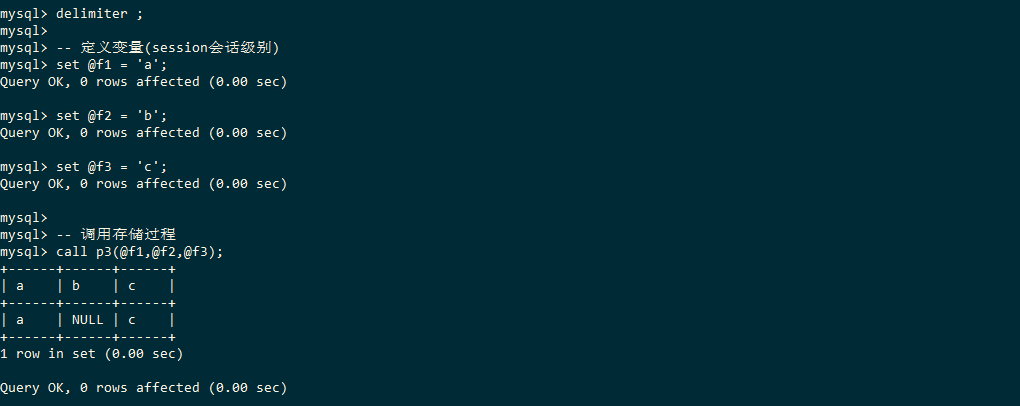
Inout：传进去又可以传出去（典型引用传值）

特别注意：Out类型与Inout类型都是类似引用传值：意味着实参不能使用值常量，必须使用变量。

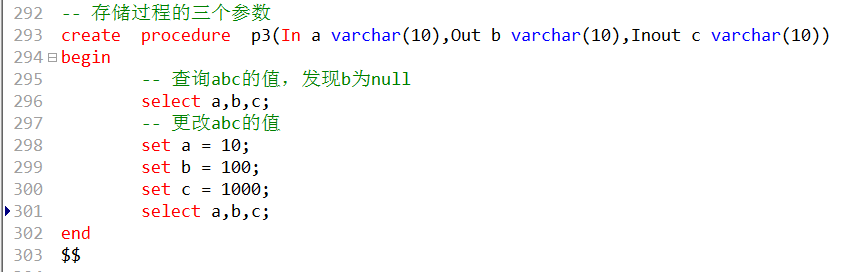
示例代码：



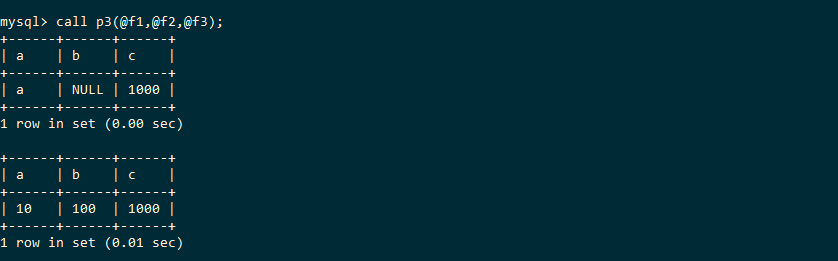
定义变量并调用存储过程



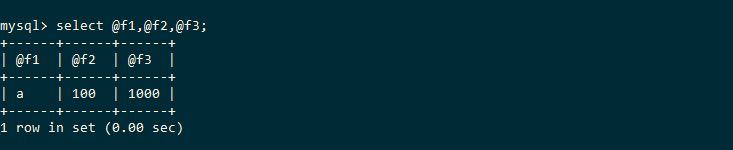
重新定义一个函数p3如下图所示：



定义变量，调用p3



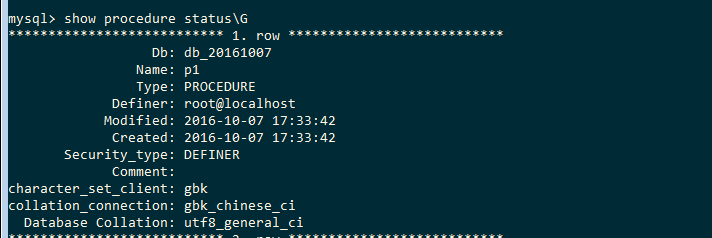
在外部访问@f1，@f2与@f3，查看是否有变化



## 5、查看存储过程

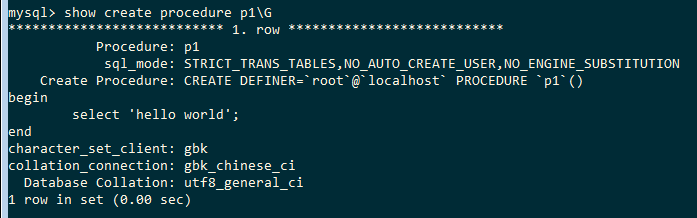
显示存储过程：

SHOW PROCEDURE STATUS;



查看过程的创建语句：

SHOW CREATE PROCEDURE 过程名称;



## 6、删除存储过程

删除存储过程（存储过程只能删除，不能修改）

DROP PROCEDURE 过程名称;

