昨日回顾

联合查询

select 子句

union [选项]

select 子句

① 一些逻辑相冲突,但是数据是一样的时候

② 表的数据非常大的时候,我们拆分开.例如: 电话号码.

交叉连接

select 字段列表 from 表名 cross join 表名;

笛卡尔积,这个笛卡尔积数据没有太多意义

内连接

select \*|字段列表 from 表名 inner join 表名 on(where) 条件;

外连接

分为左外连接和右外连接

左连接: select \*|字段 from 表名 left outer join 右表 on 条件;

右连接: select \*|字段 from 表名 right outer join 右表 on 条件;

子查询

子查询是根据返回的数据来划分

单一数据 – 标量查询

一行

一列

多行多列

exists()

exists()里只会返回一个布尔值,要么就是true要么就false

如果exists的子句 有至少一条记录,就返回true

如果没有记录,就返回false

数据备份和还原

分三种形式:

复制文件(一般很少用)

备份数据

备份: select \*|字段 into outfile “路径” from 表名;

还原: load data infile “路径” into table 表名;

备份sql语句

备份: mysqldump.exe –h –P –u –p 数据库 表 > 路径(路径没有引号,斜杠是反斜杠)

还原: mysql –h –P –u –p 数据库 < 路径

# 第30讲 视图

## 视图介绍

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。

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。

为什么要使用mysql视图?

* 重用SQL语句
* 简化复杂的SQL语句，使用表的组成部分而不是整个表
* 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
* 水平分表垂直分表

## 视图创建

* 视图属于某个数据库，默认在当前数据库中创建

create view 视图名 as

select 子句;

## 

## 访问视图

* 视图是虚拟的数据表。所有操作可以对应数据表的操作。

## 查看视图

### 查看当前数据库中的全部视图

show tables;

### 

### 查看视图的结构

desc 视图名;

### 

### 查看视图的创建语句

show create view 视图名;

## 

## 修改视图

alter view 视图名 as

select 子查询;

## 

## 删除视图

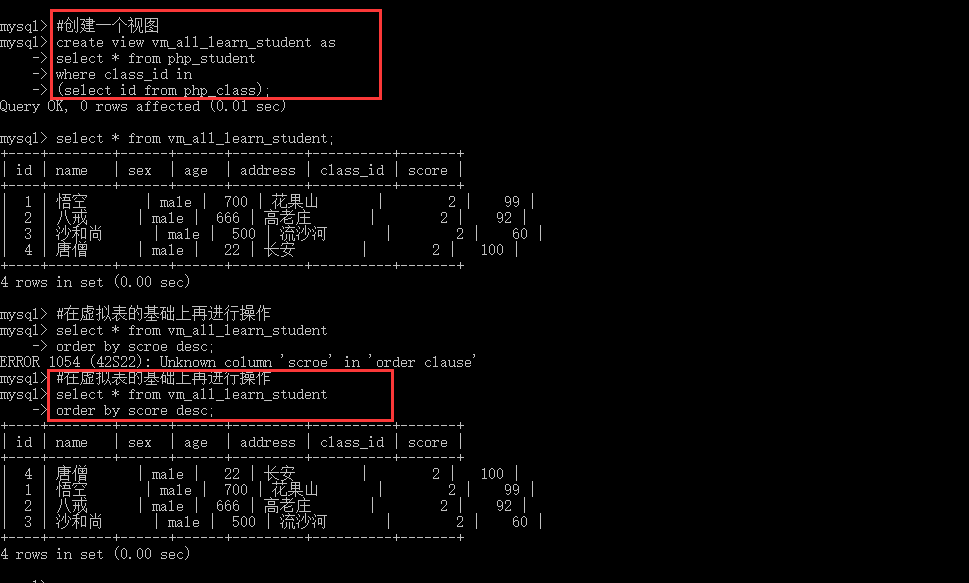
drop view 视图名;



## 视图的作用

### 重用sql语句

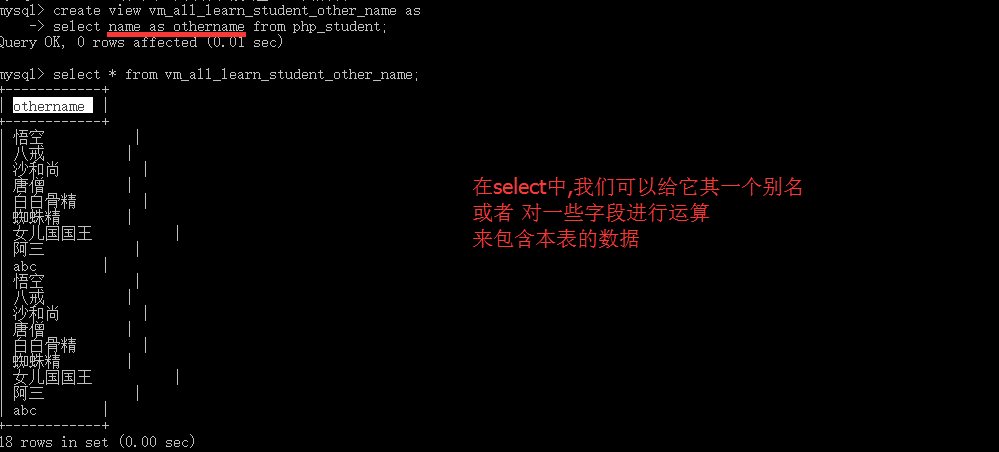
一些比较复杂的数据我们可以创建视图



### 简化复杂的sql语句,使用表的组成部分而不是整个表



### 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



### 水平分表和垂直分表

* 水平分表: 把一个大表分解成为若干个结构相同的小表.需要使用union连接数据
* 垂直分表:把一个表的字段分为两部分,常用的字段放在一个表中,不常用的字段也放在一个表中.需要使用join将表连接起来

## 视图算法(视图的查询模式)

### 基本语法

create algorithm = undifined|merge|temptable view 视图名 as

select 子句;

默认算法是 undefined

### 算法分类

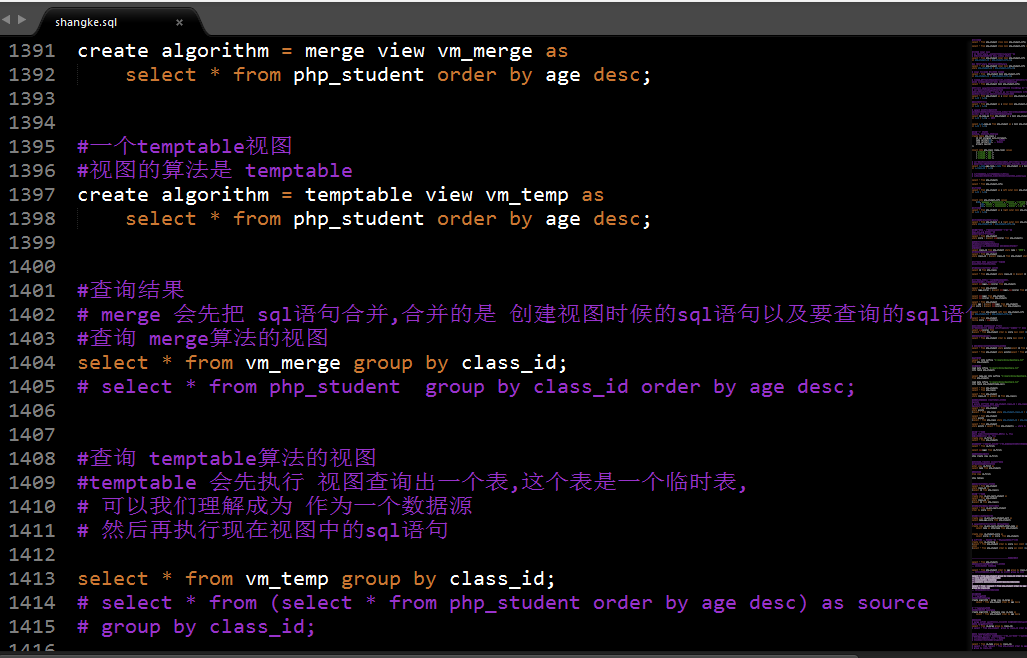
MySQL视图有三种算法：

* 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）。或者指定算法为undifined.
* 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，将其结果放置在临时表中。后在临时表的基础上，执行查询语句

原则就是: merge先合并 ,temptable 先产生临时表(先产生数据源)

### Merge和Temptable的区别

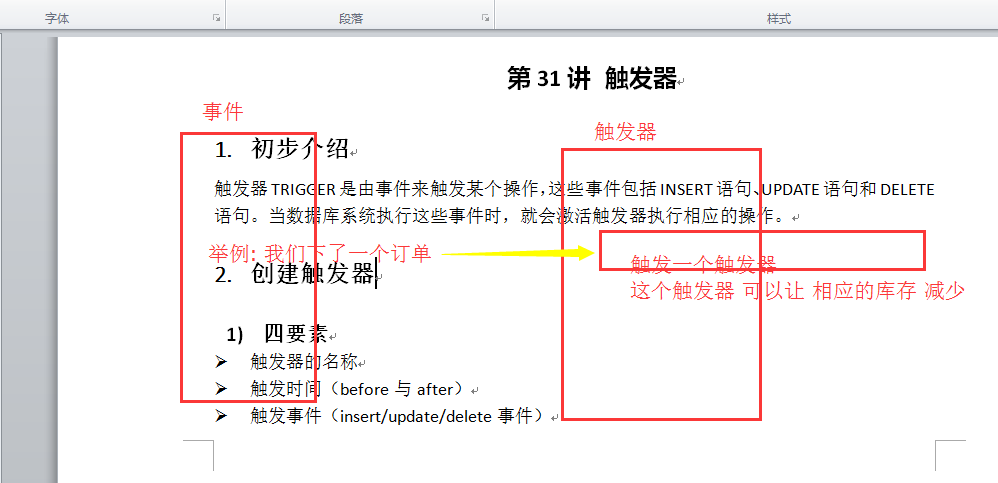
例如:求每个班级中年龄最大的学员信息



# 第31讲 触发器

## 初步介绍

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。



## 创建触发器

### 四要素

* 触发器的名称
* 触发时间（before与after）
* 触发事件（insert/update/delete事件）
* 触发对象（针对哪个表）,这个表要存在

### 创建语法

修改一下sql语句的结束标志

delimiter 新标志

create trigger 触发器的名字

before|after

insert|update|delete

on 表名 for each row

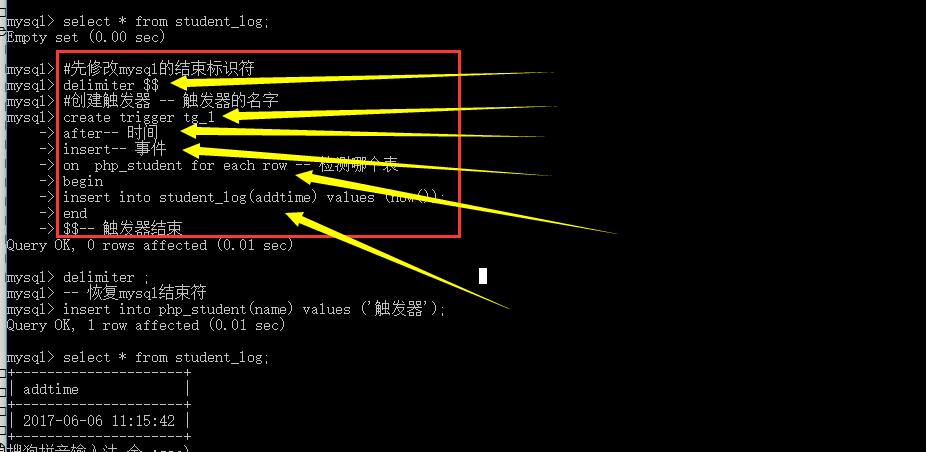
begin

处理的语句; -- 举例 update 表名 set 字段名 = 值 where 语句;

end

结束标志

delimiter ; 恢复sql语句的结束标志



注意事项:

* delimiter 指定定界符
  + 符号可以为除\之外的任意符号
  + 会话模式，退出登录或者关闭窗口即失效
* 触发器是基于行的触发，始终针对的是一行记录。
* 动作语句不能有返回值（结果集）

### 显示触发器

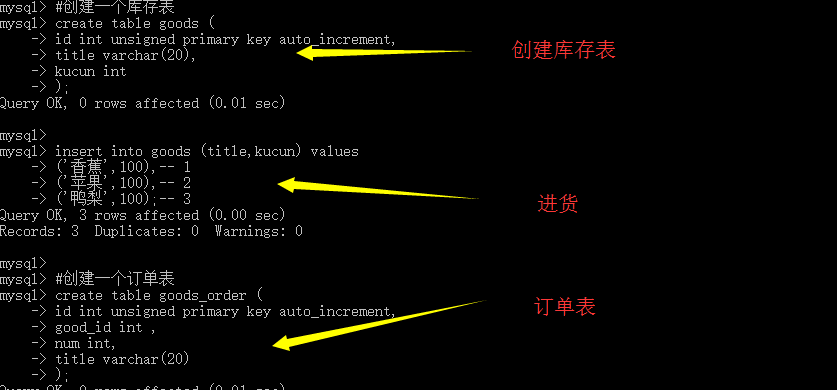
show triggers

### 删除触发器

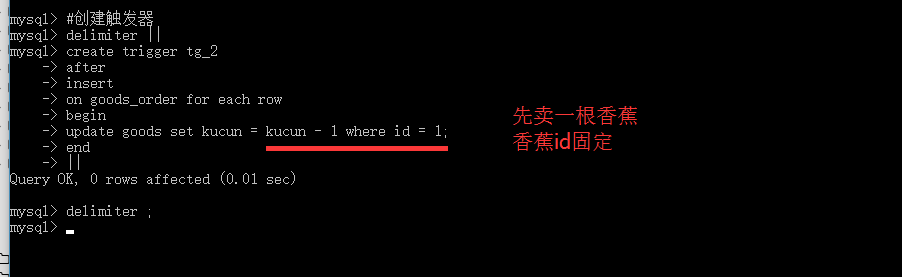
drop trigger 名字

## 

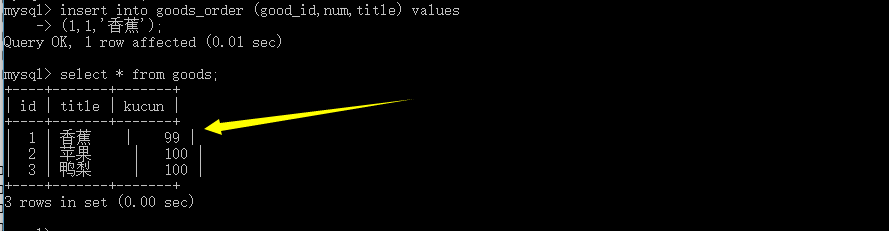
## 触发器使用场景



创建一个触发器



插入数据



## 触发中的两个参数old与new

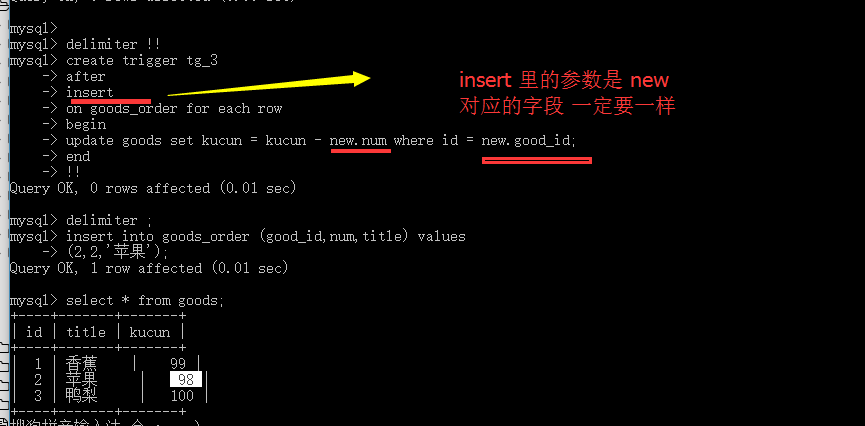
* 当执行insert插入事件时，数据表中会自动生成一行记录，这个时候触发器就会生成一个参数new，我们可以通过new来访问新记录。
* 当执行delete删除事件时，数据表中会自动减少一行记录，这个时候触发器就会生成一个参数old，我们可以通过old来访问已被删除的记录。
* 当执行update更新事件时，数据表中会自动把老行记录更改为新行记录，这个时候触发器就会生成两个参数，一个new、一个old，我们可以通过new来访问新行记录，也可以通过old来访问老行记录。

insert 对应的 参数名就是 new

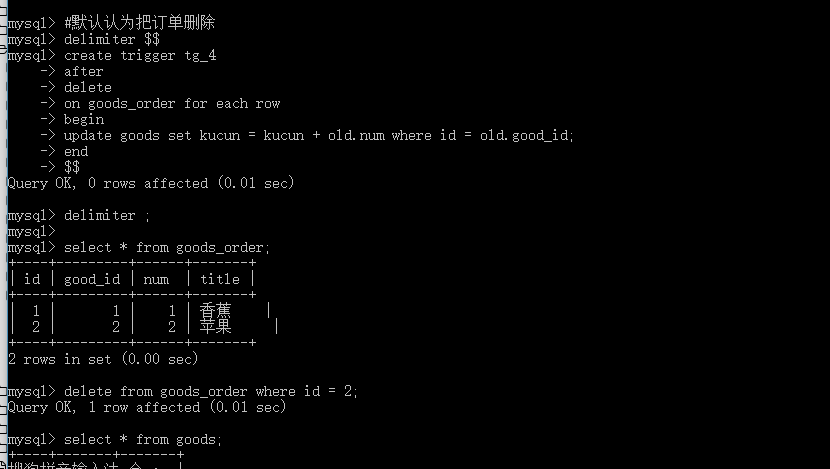
delete 对应的参数就是old

update 对应的参数就是 new 和 old

new|old 表示的就是当前这一行的 记录.我们在使用的时候就是采用 new(old).字段名;



自动退货处理:



# 第32讲 事务

我一分钱都没有,跟午哥借钱

午哥有100万

## 初步介绍

所谓的事务处理是为了保证事务的一致性而引入的概念。

## 实现事务处理功能

① 使用事务处理必须有一个前提：数据表引擎必须为InnoDB引擎

② 实现事务的处理的基本步骤：

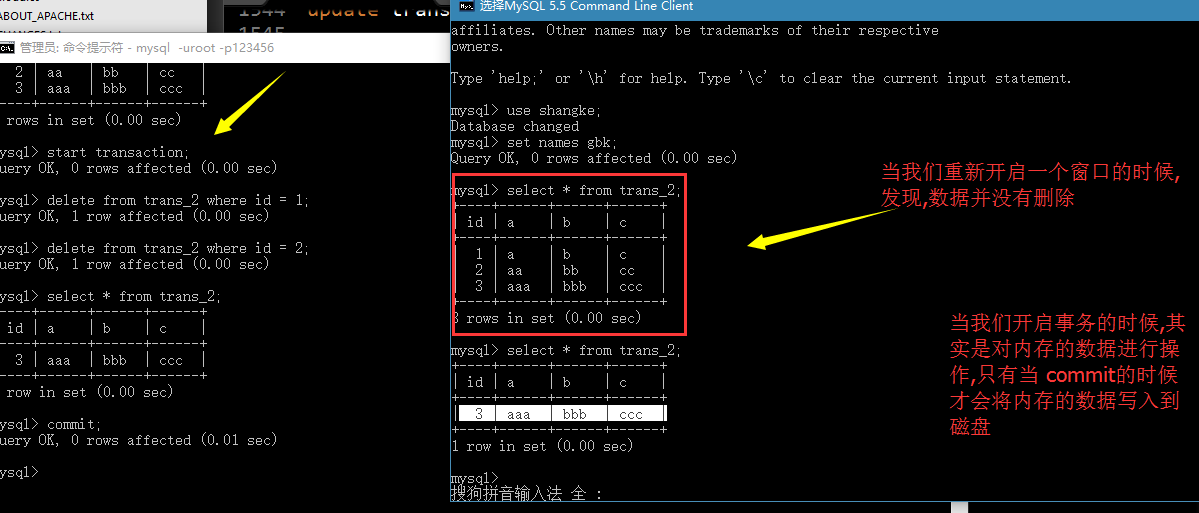
第一步：开启事务start transaction(会话机制)或set autocommit=0(永久的)

第二步：成功，则提交事务commit

第三步：失败，则回滚事务rollback

## 模拟事务处理

事务其实只是将数据在内存中做了修改,当 你进行commit 提交的时候才将 内存中的数据写入到磁盘



# 第33讲 用户权限管理

一个公司基本上会把所有的数据都放到一个数据库上,

如果有很多项目, 不能让你都能看到这些项目,也不能访问所有的数据

默认的使用的都是root用户，超级管理员！拥有最高的所有的权限！

但是，一个大的项目通常是由很多个项目小组共同实现的，也就是说一个数据库服务器上面，可能运行着很多个项目的数据库！

所以，我们应该根据不同的项目小组建立不同的用户，而且分配不同的权限来管理

## 查看用户

用户的基本信息保存在mysql.user中



## 创建用户

create user ‘用户名’@’host’ identified by ‘密码’;

host 可以是localhost – 本机

host 也可以是一个ip地址 例如: 192.168.133.36

host 也可以是一个网段 例如: 192.168.133.%

密码一定要用引号 括起来

## 

## 给用户分配权限

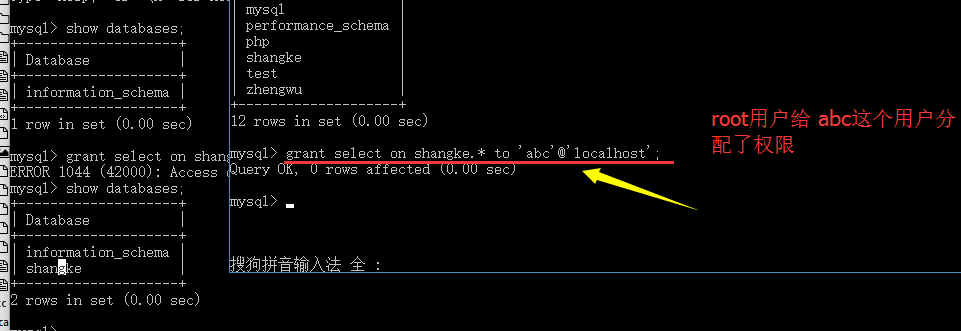
grant 权限 on 数据库.表名 to ‘用户名’@’host’

grant ALL PRIVILEGES on \*.\* to ‘用户名’@’host’;

ALL PRIVILEGES 所有权限

\*.\* 表示 所有数据库的所有表



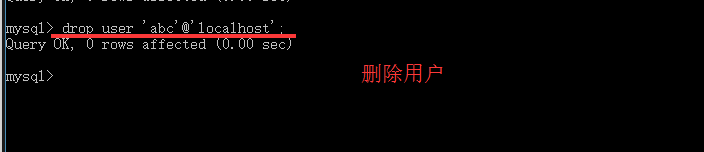


## 回收权限

revoke 权限 on 数据库名.表名 from ‘用户名’@’host’

## 

## 删除用户



# 第34讲 MySQL函数

## 函数分类

* 系统函数
* 自定义函数：存储函数

## 系统函数

函数调用是 select 函数名() 的形式调用;

数据库区分不区分大小写 以系统为准

mysql 的指令不区分大小写,建议大家 大写!!!

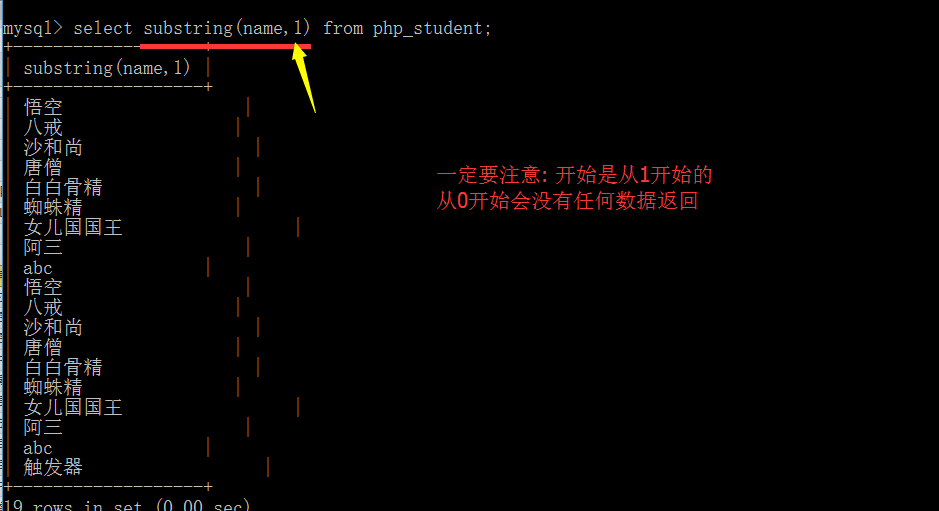
### 字符串处理函数

* 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。区别PHP中的strlen();
* 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



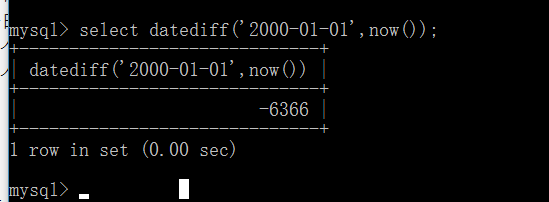
* SubString(str,start[,len])：返回子串的字符，mid（）同名函数。(substr)

位置从1开始。



### 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
* Time() ：返回一个日期的时间部分
* DateDiff() ：求两个日期的差 只是一个天数



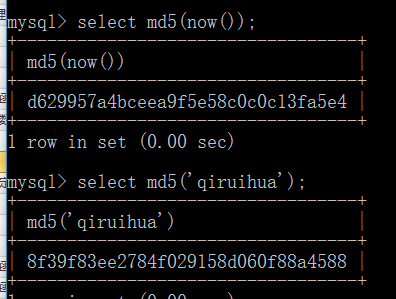
* Year()：返回一个日期的年份部分
* Month() ：返回一个日期的月份部分
* Day() ：返回一个日期的天数部分
* Hour() ：返回一个日期的小时部分
* Minute() ：返回一个日期的分钟部分
* Second() ：返回一个日期的秒部分

### 数学函数

* Abs() ：求绝对值，正数的绝对值是它本身，负数的绝对值是它的相反数。
* Pi() ：求圆周率
* Rand() ：求随机数，效率不高

### 其他函数

* Md5()：32位加密函数。



## MySQL中的自定义函数

### 四要素

* 函数名称
* 参数（形参、实参）
* 返回值
* 函数体

### 创建语法

delimiter 标识符

create function 函数名(参数名 参数类型,…) returns 返回值类型

begin

函数体

return 返回值;

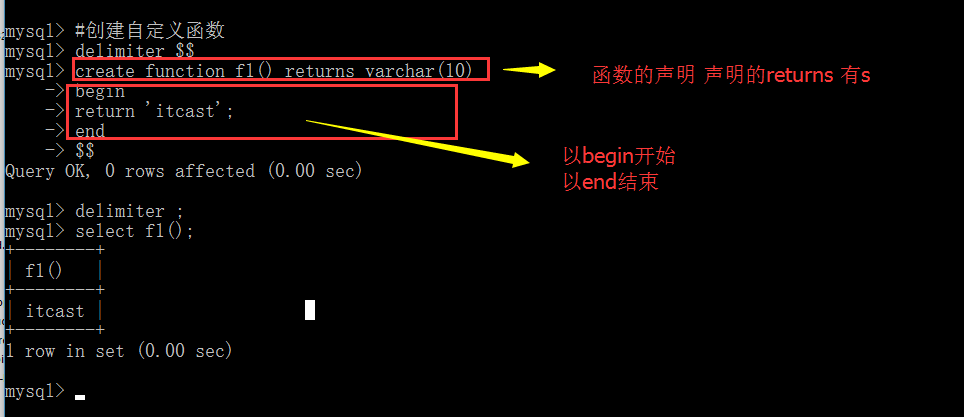
end

标识符

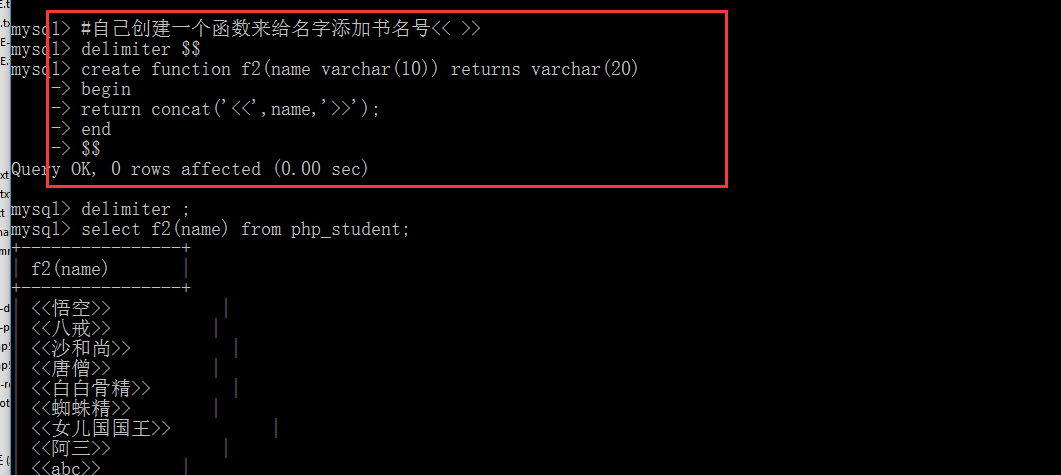
delimiter ;

函数体中必须有 return 因为这个是一个有返回值的函数

### 调用方式

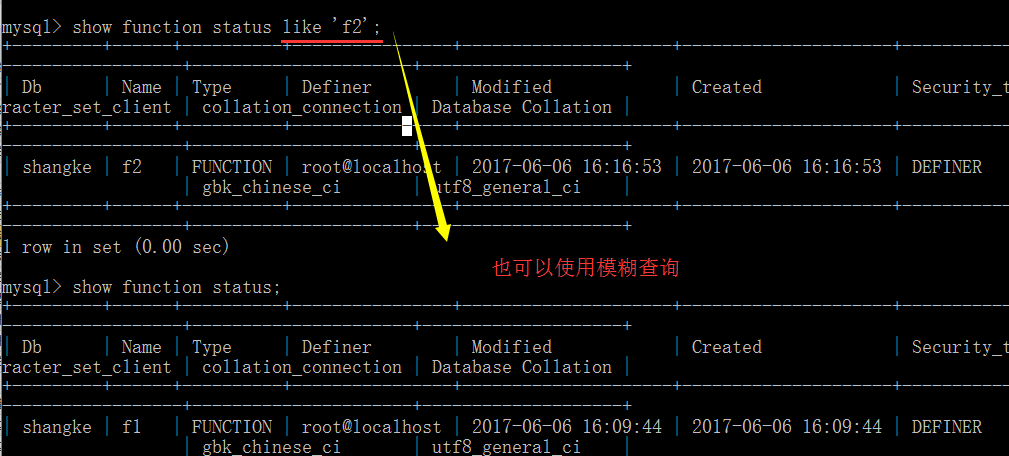


select 函数名();



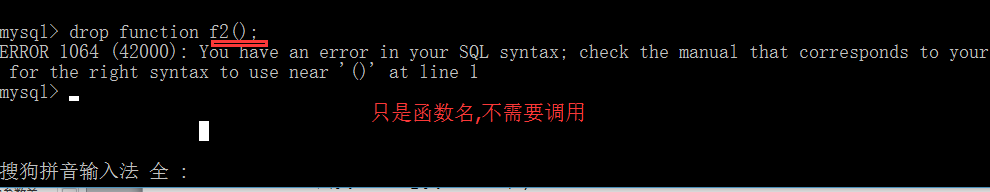
### 查看自定义函数

show function status



### 删除自定义函数

drop function 函数名;



## MySQL中的流程控制

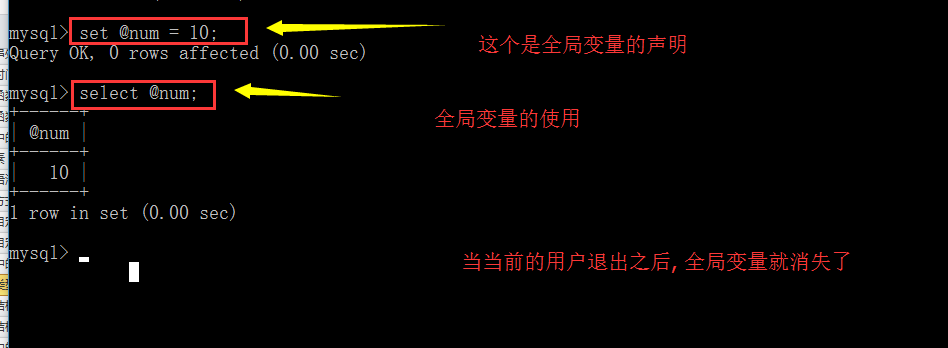
### 变量类型

局部变量只能在函数内部使用

declare 变量名 类型 default 值

全局变量可以在全局使用

set @变量名 = 值;



### 分支结构

if 条件判断 then

执行满足条件的sql语句;

elseif 条件判断 then

执行满足条件的sql语句;

else

执行sql语句;

end if;

### 

### 循环结构

[循环标记]: while 条件判断 do

循环代码

end while;

### 

### 循环中的继续及中止循环

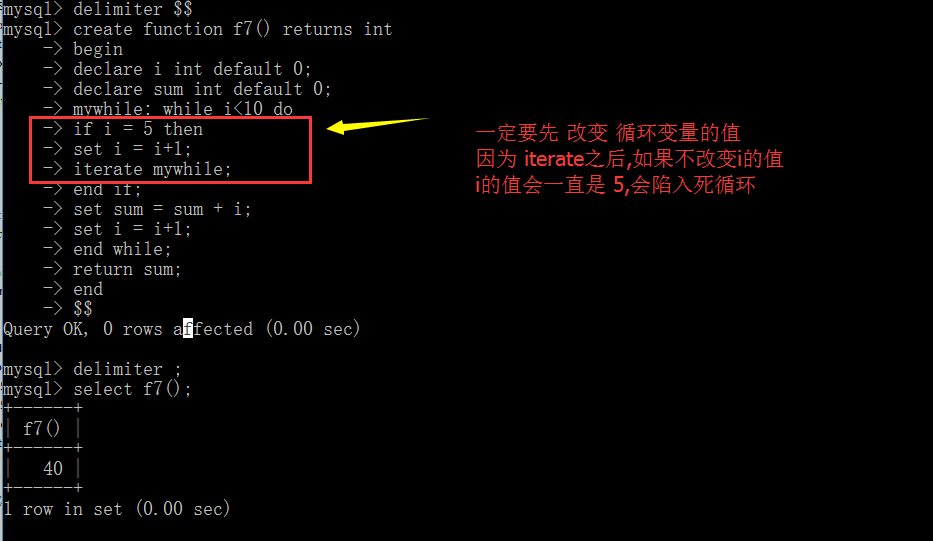
在php循环结构中，存在两个特殊的关键词：continue与break

continue ：中止当前循环，继续下一次循环

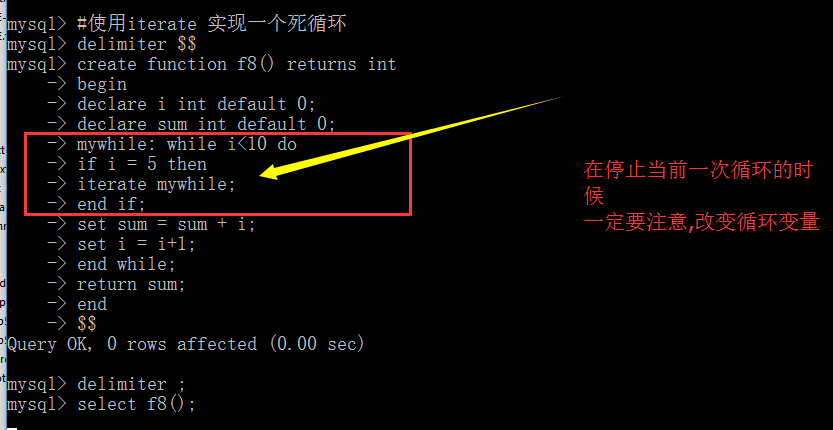
break ：中止整个循环

但是在mysql循环中，其并没有continue与break，而是通过iterate与leave代替continue与break。

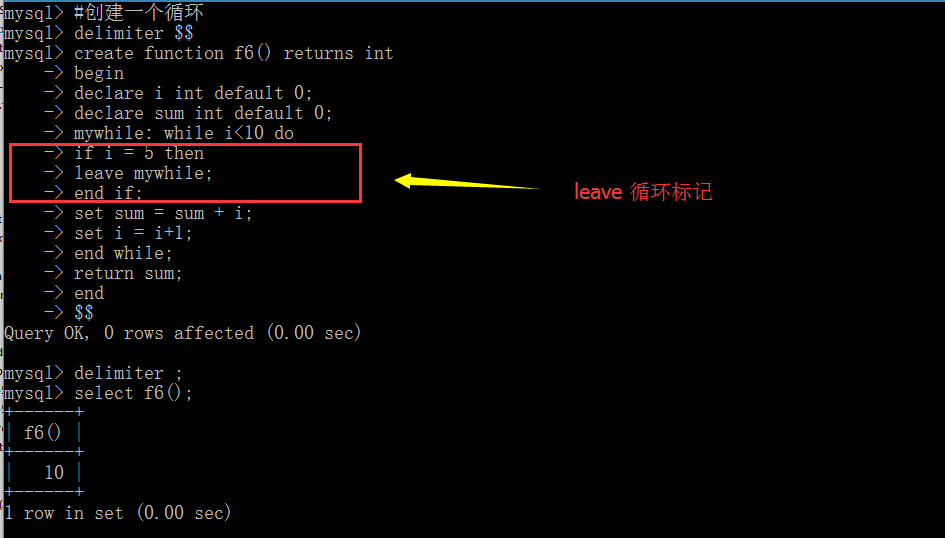
iterate 🡪 continue



以下是错误的实例代码,会造成死循环



leave 🡪 break



# 第35讲 存储过程

之前创建的都是有返回值的,现在我们学习没有返回值

## 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。

## 创建存储过程

delimiter 标识符

create procedure 函数名(参数名 参数类型, …)

begin

函数体

end

标识符

delimiter ;

## 

## 调用存储过程

call 函数名;

## 

## 查询存储过程

show procedure status;

## 

## 删除存储过程

drop procedure 函数名;

## 

## 存储过程的参数关键字

存储过程的参数不单有数据类型，而且有自己的类型

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）

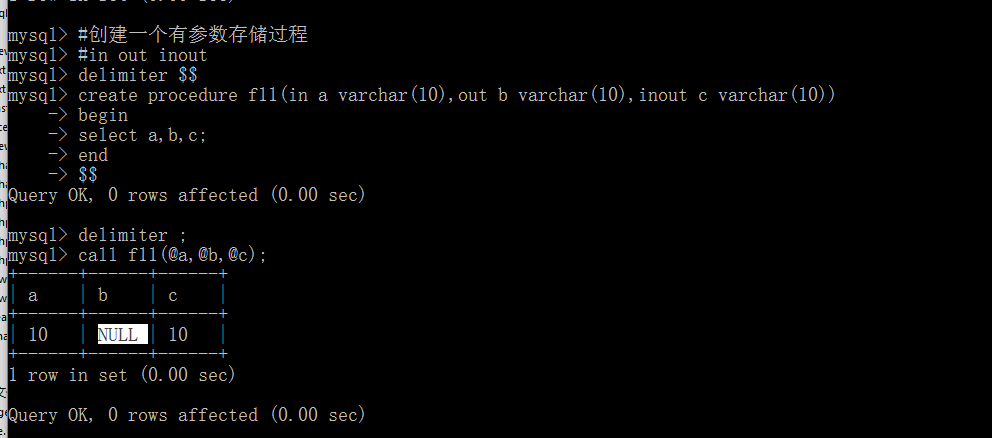
特别注意：Out类型与Inout类型都是类似引用传值：意味着实参不能使用值常量，必须使用变量。

记住：

In类型：代表按值传递，把实参的值创建一个副本赋值给形参，其形参值的改变不会对实参产生任何影响。

Out与Inout类型：代表按引用传递，实参与形参指向了同一块内存空间，形参的值改变会对实参产生影响。

Out与Inout类型，虽然都属于引用类型，但是有很大的区别。Out在执行时首先会清空原有变量的值，设置为null，但是Inout并不会对其值进行重置。



corelated query