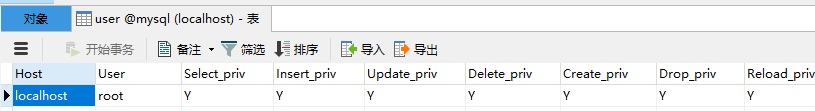
# 数据控制语言

是用于对mysql的用户及其权限进行管理的语句

## 用户管理

Mysql中的所有用户都存储在系统数据库（mysql）中的user表中——不管哪个数据库的用户，都存储在这里。



## 创建用户

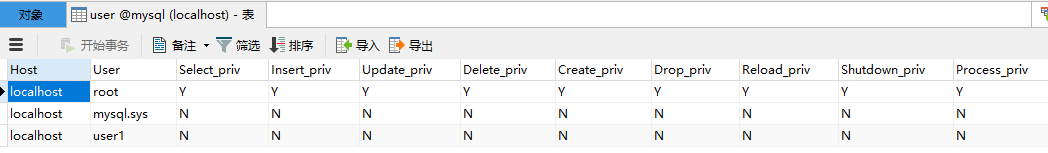
Create user ‘用户名’@‘允许登录地址/服务器’ identified by ‘密码’

说明：

1. 允许登录的地址/服务器就是，允许该设置的位置，来使用该设定的用户名密码，其他位置不行
2. 可见，mysql的安全身份验证，需要3个信息。



查看表



## 删除用户

drop user ‘用户名’@‘允许登录地址/服务器’

## 修改用户密码

修改直接的密码

Set password=password（‘密码’）

修改它人的密码（需要权限）

Set password for ‘用户名’@‘允许登录的地址’=password（‘密码’）

# 权限控制管理

## 权限是什么？

Mysql数据库，将其中所能做的所有事情，都分门别类分配到大约30多个权限中去了，其中每个权限，都是一个“单词”而已！比如：

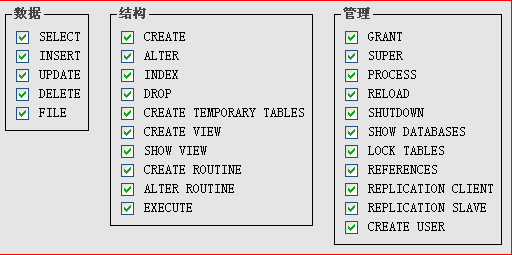
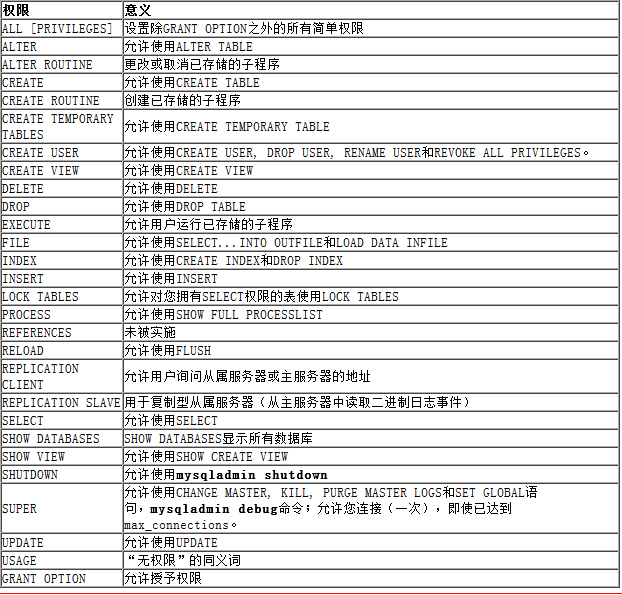
Select 代表可以查询数据

Update 代表可以修改数据

Delete 代表可以删除数据

…….

其中，有个权限叫all 表示所有权限

## 授予权限

grant 权限列表 on 某库 . 某个对象 to 用户名@登录位置 【identified by '密码'】

说明

1. 权限列表，就是，多个权限的名词，互相之间用逗号分开比如：select，insert ，update

也可以写all

1. 某库.某个对象，表示，给指定的某个数据库中的某个“下级单位”赋权

下级单位有：表名，视图名，存储过程名；存储函数名

其中，有2个特殊的语法：

\*.\*：代表所有数据库中的所有下级单位

某库.\*：代表指定的该库中的所有下级单位

1. 【identified by '密码'】是可省略部分，如果省略，就表示赋权的同时，修改密码
2. 但：如果用户不存在，此时会新增一个用户，并设置其密码

## 剥夺权限

形式

Revoke 权限列表 on 某库 . 某个对象 from ‘用户名’@‘允许登录位置’

其含义，跟grent中完全一样；

# 事务控制语言

什么叫做事务

想象一个场景：

小明给小花 汇款 5000元 买 IPHONE，操作界面不用管，不管什么操作界面，最终都要落实到这样两条语句的执行：

update 存款表 set money = money - 5000 where 账户=’小明’；

update 存款表 set money = money + 5000 where 账户=’小花’；

当，第一条语句执行成功，突然断电了（或任何其他情况），就会造成数据的“不一致”。

要解决这个问题，就是“事务”的功能：

事务就是用来保证多条“增删改”语句的执行的“一致性”：要么都执行完成，要么都没有执行；

## 事务的特点：

* 原子性：一个事务中的所有语句，应该做到：要么全做，要么一个都不做；
* 一致性：让数据保持逻辑上的“合理性”，比如：一个商品出库时，既要让商品库中的该商品数量减1，又要让对应用户的购物车中的该商品加1；
* 隔离性：如果多个事务同时并发执行，但每个事务就像各自独立执行一样。
* 持久性：一个事务执行成功，则对数据来说应该是一个明确的硬盘数据更改（而不仅仅是内存中的变化）。

## 事务模式：

事务模式是指：

在我们的cmd命令行模式中，是否开启了“一条语句就是一个事务”的这个开关：

默认情况下（安装后），这个模式是开启的，称为“自动提交模式”；

set autocommit = 1;

这样之后，每条增删改语句，都会立即生效；

我们可以把它关闭，那就是“人为提交模式”——即需要人为提交；

set autocommit = 0;

这样之后，所有增删改语句，都必须使用commit之后，才能生效；

# 事务执行的基本流程

## 只支持增删改语句

1. 开启一个事务 start transaction 也可以写成begin
2. 执行多条增删改语句；//也就是相当于希望这多条语句要作为一个“不可分割”的整体去执行的任务
3. 判断这些语句执行的结果情况：

If（没有出错）{

Commit //提交事务

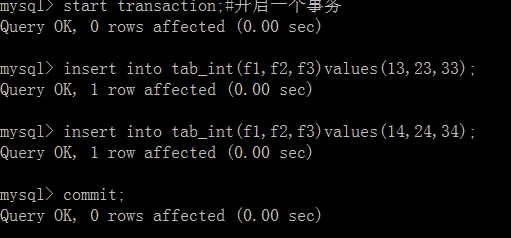
}else{

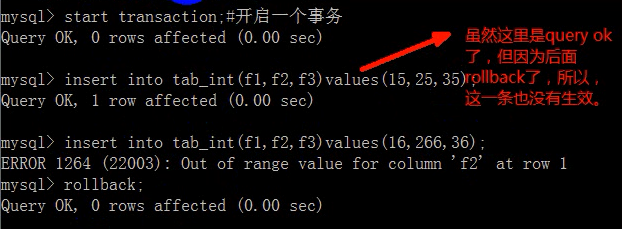
Rollback://回滚事务

}

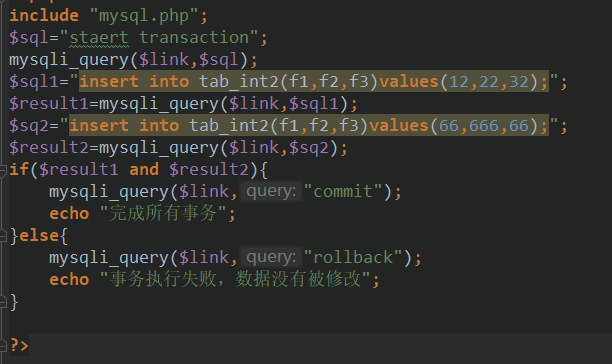
具体分两种情况：

在cmd中：凭肉眼来观察

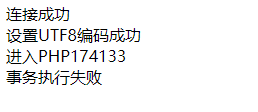




Php中：



结果为



MySQL编程

Mysql编程中的语句块包含符

其实就是相当于js或php中的大括号语法

【标识符】：Begin

//语句

End 【标识符】；//。。。。

标识符就是定义的任意名字而已，比如

If（条件判断）

Begin

//。。。。。

End

If（条件判断）

A：Begin

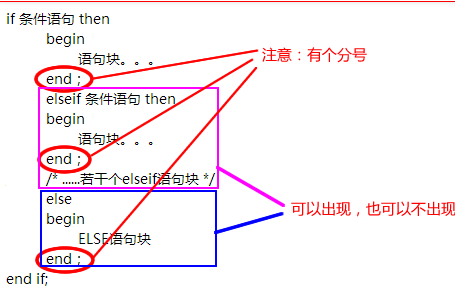
//。。。。。

End A

A就是标识符，他的作用是“标识”该语句块，以其可以在该语句块中“使用它”——其实就是退出；

# 流程控制语句

If语句



Case语句 其实相当于switch



语法1示意：

Case @v1

When 1 then //如果v1等于1，则表示满足条件，进入分支

Begin

//、、、、、

End

When 2 then //如果v1等于2，则表示满足条件，进入分支

Begin

//、、、、、

End

Else

Begin

//、、、、

End

End case

说明：

@v1表示一个变量，后面都是这样

语法2示意：

Case

When @v1>0 then //如果该条件满足，进入分支，否则继续判断

Begin

//、、、、

End；

When @v1<0 then

Begin

//、、、、、

End

Else

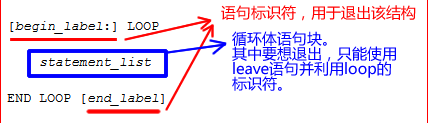
Begin

//、、、、

End；

End case；

## Loop循环语句.



标识符：loop

Begin

Leave 标识符

//这里循环的语句块。。。

//注意：这里必须有一个“退出循环”的逻辑机制，否则该循环就是死循环，其基本形式类似这样：

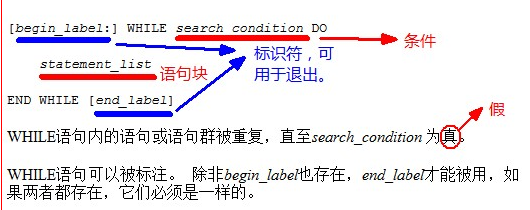
If（条件）then

Leave 标识符； //退出

End；

End loop 标识符

# While循环



举例：

Set @v1=1；//赋值语句

While @v1< 10 do

Begin

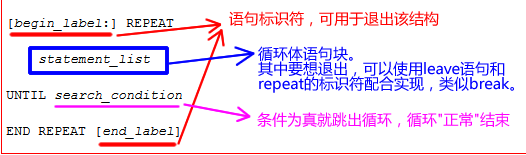
Insert into tab1 (id,name) valiues (null，@v1)

Set @v1=@v1+1

End

End while

# Repeat循环



类似 do while

举例：

Set @v1=1；//赋值语句

repeat @v1< 10 do

begin

Insert into tab1 (id,name) valiues (null，@v1)

Set @v1=@v1+1

End；

Until @v1>=10；

End repeat

# leave语句：

* 语法：

leave 标识符；

* 其作用是用来退出begin...end结构或其他具有标识符的结构。

# Mysql中的变量

.

Mysql中，有两种变量形式：

普通变量：不带@符号

定义形式：declare 变量名 类型名 【default 默认值】

//普通变量必须先这样定义

赋值形式：

Set 变量名 = 值

取值就直接使用变量名

使用场所：只能在编程环境中使用

什么是编程环境

只有三个：

1.定义函数的内部

2.定义存储过程的内部

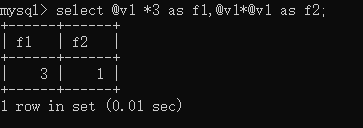
3.定义触发器的内部

会话变量：带@符号

定义形式：@变量名 = 值 （也是赋值形式）

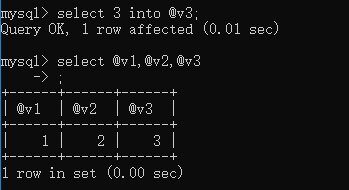
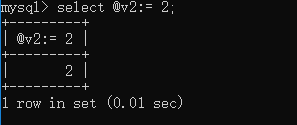
//无需定义，直接赋值 第一次就算是定义

使用场所：基本在哪里都可以使用



## 变量赋值有如下形式

* 语法1（针对普通变量）：set 变量名 = 表达式；#此语法中的变量必须先使用declare声明
* 语法2（针对会话变量）：set @变量名 = 表达式； #此方式可以无需declare语法声明，而是直接赋值，类似php定义变量并赋值。
* 语法3（针对会话变量）：select @变量名 := 表达式；#此语句会给该变量赋值，同时还会作为一个select语句输出“结果集”。
* 语法4（针对会话变量）：select 表达式 into @变量名；#此语句虽然看起来是select语句，但其实并不输出“结果集”，而只是给变量赋值。
* 注意：上述语法中变量形式的区别：“@变量名”形式可以在非程序语句中使用，而不带“@”的变量名只能在程序语句中（比如begin ...end范围)，且后者必须先声明（declare语法)才能赋值。



# Mysql中函数

# （存储）函数

函数，也说成“存储函数”，其实就是js或php中所说的函数

唯一的区别

这里的函数必须返回一个数据（值）；

定义形式



调用形式：

跟调用内部函数一样！比如

Select now（），8+3 as f2，func1（）

//now是系统函数 func1是自定义函数

如果在编程语句中：

Set @v1=now()

Set @v2=func()

#创建一个函数

#函数目标是取出3个数中的最大值

Create function getmaxvalue(p1 float,p2 float,p3 float)

Returns float #返回float类型

Begin

Declare result float;#说明变量 没有默认值

If(p1>=p2 and p1>=p3)then

Begin

Set result = p1;

End;

Elseif(p2>=p1 and p2>=p3) then

Begin

Set result = p2

End

Else

Begin

Set result = p3;

end

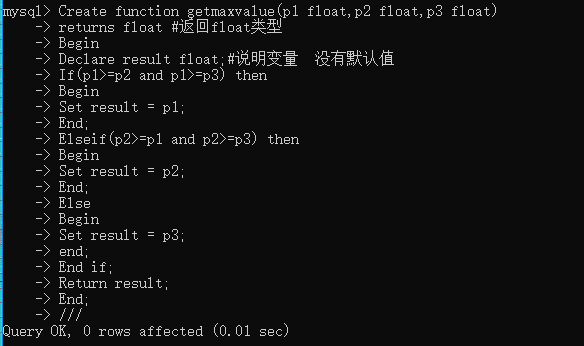
End if

Return result;

End;

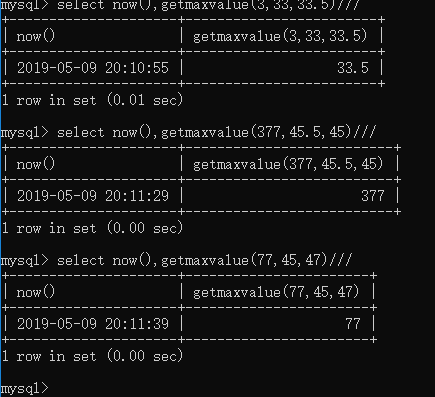
//注意，在cmd中执行该代码，需要更换“语句结束符”

Delimiter ///



**注意事项：**

1. **在函数内容中，可以有各种变量和流程控制的使用**
2. **在函数内部，也可以有各种增删改语句**
3. **在函数内部，不可以用select或其他“返回结果集”的查询类语句**



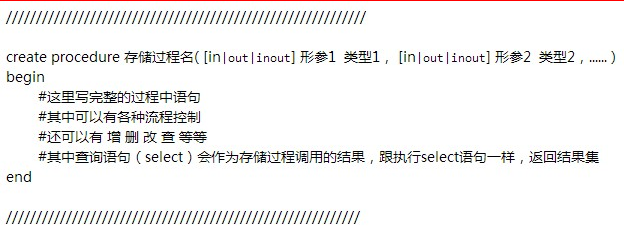
# 删除函数

Drop function 函数名

# 存储过程procedure

存储过程，其实还是函数，但其规定：不能有返回值

## 定义形式：



说明：

1. in 可省略 用于设定该变量是用来“接收实参数据”的，即“传入”；默认不写就是in
2. out 可省略 于设定该变量是用来“存储过程中数据”的，即“传出”，即函数中必须对他赋值；
3. inout 是in和out的结合，具有双向作用；
4. 对于out和inout设定，对于的实参就“必须”是一个变量，因为该变量是用于“接收传出数据”

#创建一个存储过程

#目标是：将3个数据写入到表tab\_int

#并返回该表第一个字段的前3大值的行

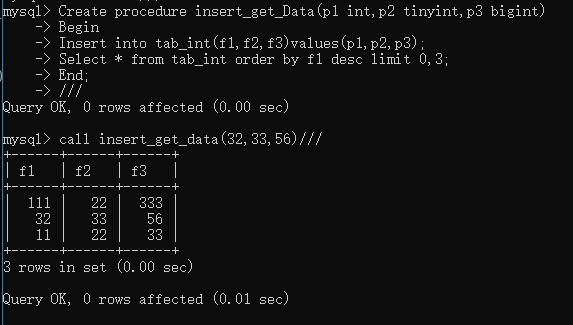
Create procedure insert\_get\_Data(p1 int,p2 tinyint,p3 bigint)

Begin

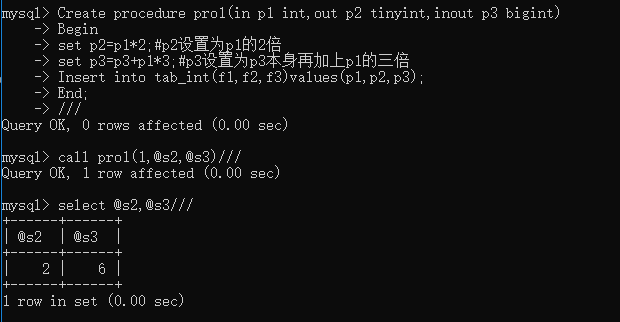
Insert into tab\_int(f1,f2,f3)values(p1,p2,p3);

Select \* from tab\_int order by f1 desc limit 0,3;

End;



举例2使用int和inout



有out的实参必须是用变量

## 调用存储过程

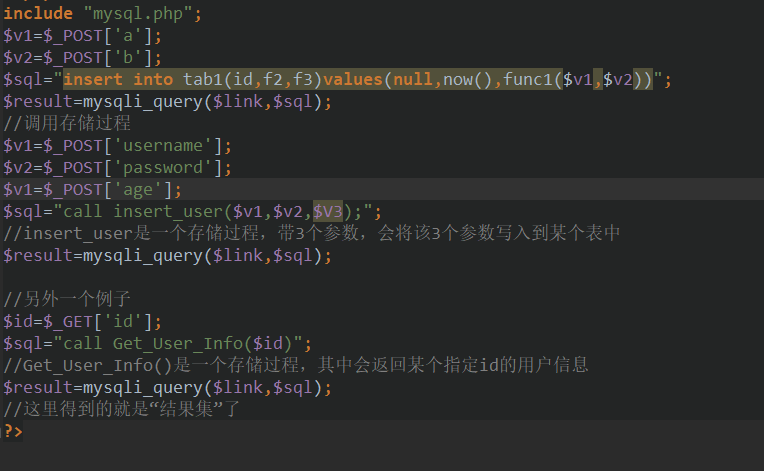
Call 存储过程名（实参1，实参2-----）

它应该是在“非编程环境中”调用，即增删改查环境下

## 删除存储过程

Drop procedure 存储过程名

# 在php中使用存储函数或存储过程的示意



# 触发器（trigger）

含义：

触发器：也是一段预先定义好的代码（跟存储过程和存储函数一样），并有个名字，但：

它不能调用，而是，在某个表发生某个事件（增删改）的时候，会自动“触发”而调用起来。

## 定义形式：

Create trigger 触发器名 触发时机 触发事件 on 表名 for rach row as

Begin

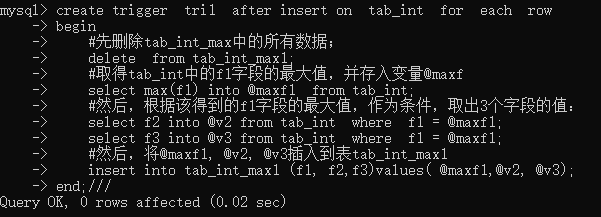
//这里，才是编程的位置，也就是触发器的内部语句

End；

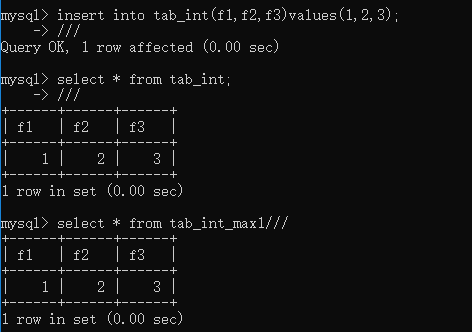
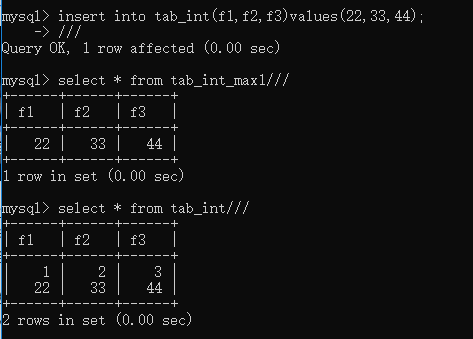
说明

1. 触发时机只有2个：before（在…之前），after（在…之后）
2. 触发事件，只有3个：insert update delete
3. 即触发器的含义是：在某个表上进入insert或update或delete之前（或之后），会去执行其中写好的代码（语句）；即每个表只有6个情形会调用该触发器；
4. 通常，触发器用于在对某个表进行增删改的操作的时候，需要同时去做另外一件事情的时候；
5. 在触发器的内部，有2个关键字代表某种特定的含义，可以用于获取数据：
   1. New 代表当前正要执行的insert或ipdate的时候的“新行”数据；通过他，可以获取这一新行数据的任意一个字段的值，形式为：
      * 1. Set @v1=new.id；//获得该新插入或update行的id字段的行（前提是有该id）
        2. Set @v2=new.age；//同上
   2. Old代表当前正要执行的delete的时候的“旧行”数据；通过他，可以获取这一旧行数据的任意一个字段的值，形式为：
      * 1. Set @v1=old.id；//获得该新插入或update行的id字段的行（前提是有该id）
        2. Set @v2= old.age；//同上

案例



测试

# 新的触发器

