存储过程

存储过程简称过程，procedure，是一种用来处理数据（增删改）的方式。简单点，我们也可以将其理解为没有返回值的函数。

创建过程

-- 基本语法

create procedure 过程名([参数列表])

begin

-- 过程体

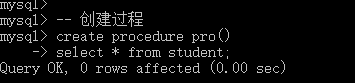
end

如果我们定义的过程的过程体内仅含有一条语句，则可以省略begin和end。执行如下语句，进行测试：

-- 创建过程

create procedure pro()

select \* from student;

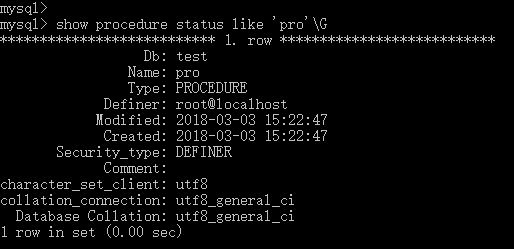
[](https://camo.githubusercontent.com/12cab0e50146d5f6fb1a93d97a858cbc220376b9/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830333033313532333137353131)

如上图所示，我们创建了一个名为pro()的过程，其目的就是为了查询student表中的数据。但实际上，过程多用于处理数据，查询并不多用。

查看过程

查看过程，基本语法为：

show procedure status + [like 'pattern'];

[](https://camo.githubusercontent.com/6a79ce8a227eb8f69589faf7e56670959f039bc5/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830333033313532363337353338)

查看过程的创建语句，基本语法为：

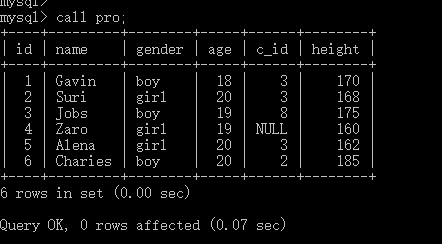
show create procedure + 过程名;

[](https://camo.githubusercontent.com/e876a0a776e5ae496fe411b5d35298b345f906fa/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830333033313532373434313936)

调用过程

由于函数有返回值，因此我们可以用select来调用函数。但是存储过程没有返回值，怎么办？实际上，对于存储过程，有一个专门的调用关键字call，调用过程的基本语法为：

call + 过程名;

[](https://camo.githubusercontent.com/54c5e831158d6c88d4b1265af3e595395912b5d1/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830333033313533303234383738)

修改过程 & 删除过程

过程只能先删除后新增，不能修改。删除过程的基本语法为：

drop procedure + 过程名;

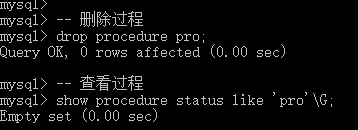
执行如下语句，进行测试：

-- 删除过程

drop procedure pro;

-- 查看过程

show procedure status like 'pro'\G;

[](https://camo.githubusercontent.com/09bb965780bb7041904d13b9a44be26a84b855a7/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830333033313533343039383733)

过程参数

函数的参数需要指定数据类型，过程比函数更加严格。过程有三种自己的参数类型，分别为：

in，数据只是从过程外部传入给过程内部使用，可以是数值也可以是变量；

out，此参数只能传递变量，且变量指向的数据需要先清空然后才能进入过程内部，该引用供过程内部使用，过程结束后可以将变量的值传递给过程外部使用；

inout，此参数只能传递变量，该变量的值可以给过程内部使用，过程结束后可以变量的值传递给过程外部使用。

因此，过程定义的具体形式应该为：

procedure 过程名(in 参数名字 参数类型, out 参数名字 参数类型, inout 参数名字 参数类型)

下面，我们定义一个简单的过程，并调用过程。代码如下：

delimiter $$

create procedure pro2(in var1 int, out var2 int, inout var3 int)

begin

-- 查看该过程传入的三个变量

select var1, var2, var3;

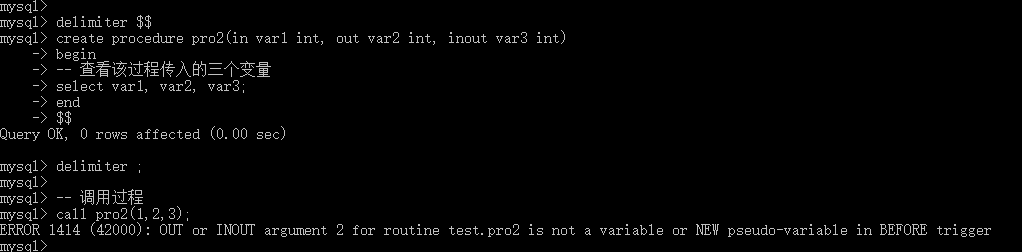
end

$$

delimiter ;

-- 调用过程

call pro2(1,2,3);

[](https://camo.githubusercontent.com/9f62bb9fe4bc2d099a0bf9bf6858bebb30fe4f9f/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830333033313635313530393537)

如上图所示，过程pro2创建成功。但是，在调用过程的时候出现错误，造成该错误的原因为：过程的out和inout两个参数只能接受变量，而我们传递了具体的数值，报错也就在情理之中啦！接下来，执行如下语句，进行测试：

-- 设置全局变量

set @var1 = 1;

set @var2 = 2;

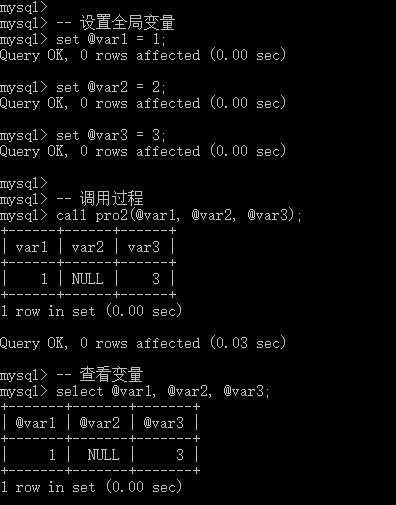
set @var3 = 3;

-- 调用过程

call pro2(@var1, @var2, @var3);

-- 查看变量

select @var1, @var2, @var3;

[](https://camo.githubusercontent.com/f9fbb99d5fc88fec080ef26738989dc3300dab36/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830333033313635383133313933)

如上图所示，在我们将变量传递给过程的时候，过程正常执行。此外，通过select语句我们可以看到传递给out类型参数的@var2的值在经过过程处理之后，被置为null值啦，这也是符合out类型参数的先清空后使用原则的。而且，由于out和inout只能接受变量作为参数，因此在过程内部对out和inout传入的变量的修改会影响到过程外部。在这里，值得我们注意是：存储过程对变量的操作是滞后的，即只有在过程结束的时候，才会将过程内部修改的值赋值给外部传入的对应的全局变量。执行如下语句，进行测试：

delimiter $$

create procedure pro3(in var1 int, out var2 int, inout var3 int)

begin

-- 查看该过程传入的三个变量

select var1, var2, var3;

-- 修改局部变量

set var1 = 10;

set var2 = 20;

set var3 = 30;

-- 查看局部变量

select var1, var2, var3;

-- 查看全局变量

select @var1, @var2, @var3;

-- 修改全局变量

set @var1 = 'a';

set @var2 = 'b';

set @var3 = 'c';

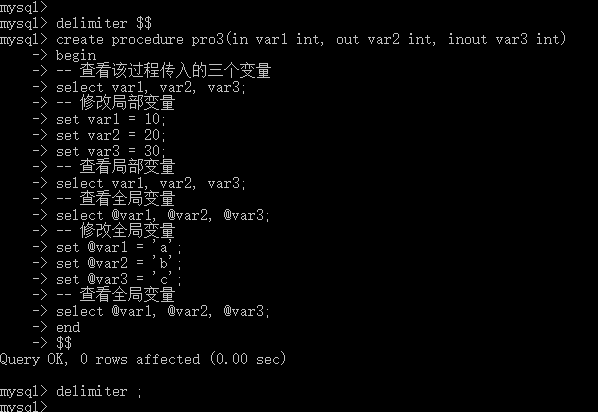
-- 查看全局变量

select @var1, @var2, @var3;

end

$$

delimiter ;

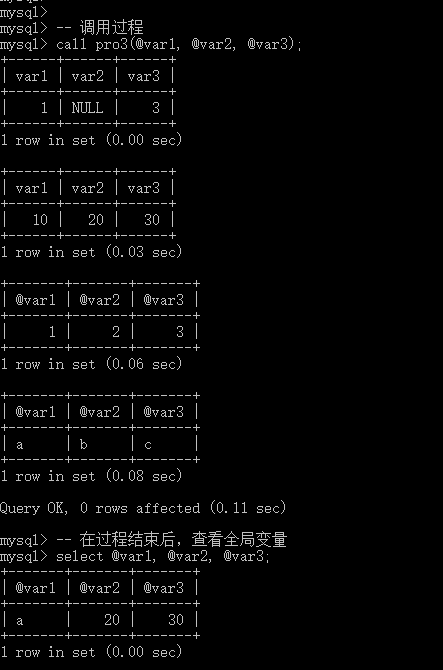
[](https://camo.githubusercontent.com/7906bc3366e2f710753090c2c428d8c473333802/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830333033313731323136343438)

-- 调用过程

call pro3(@var1, @var2, @var3);

-- 在过程结束后，查看全局变量

select @var1, @var2, @var3;

[](https://camo.githubusercontent.com/b6002de3d590d4362880894bebd8ec5bc3ea2837/687474703a2f2f696d672e626c6f672e6373646e2e6e65742f3230313830333033313731353036393132)

如上图所示，存储过程执行成功，且验证了我们结论，即：在存储过程没有结束的时候，对传入变量的修改并不会影响到对应的全局变量；只有在存储过程结束后，才会将对应的变量值赋值给out和inout类型的变量，而in类型的变量不受影响。