**第23章：使用存储过程**

1. **存储过程的定义**

存储过程，是预先定义好的、存储起来、为以后的使用而保存的一条或多条MYSQL语句的集合，可以通过call命令来调用执行。

存储过程某种意义上来讲是一种特殊的函数，可以有多个返回值，通常情况下存储过程独立地的来执行。

1. **为什么要使用存储过程**

1、**“一对多”**：

运用了“一对多”的原理，即一条SQL语句可以被多个地方使用！为了保证**数据的完整性**，便于所有开发人员和应用程序使用同一试验与测试，无需重复建立一系列相同的处理步骤；

2、**性能高**：进行复杂的操作时(比如多张表连表查询)可将那些复杂的操作用存储过程先封装起来然后进行查询，从而**简化复杂的操作，**同时使用存储过程比使用单独的SQL语句要快很多，其作为数据库一种特殊的内部对象，用法上相当于内部命令或内置函数，其**过程体是经过提前编译好的SQL语句**；

3、**一致性**：需要执行的步骤越多，出错的可能性就越大，存储过程可以有效地防止错误的发生；

4、**安全性**：简化对变动的管理，如果表名、列名或业务逻辑（或别的内容）有变化，只需要更改存储过程的代码，而使用它的人员甚至不需要知道这些变化。通过存储过程限制对基础数据的访问减少了数据讹误的机会（讹误：字句上的无意识的疏漏导致的差误）；可以指定只有某用户才可以进行访问！

5、**高速并减少网络流量**：只在创造时进行编译，以后每次执行都不需要重新编译，这与一般的SQL语句(一般的SQL语句每执行一次就编译一次)相比就速度加快了很多，网络应用程序中使用效果明显。

**缺点：**

1. **编写存储过程需要更高的SQL编程技能，对SQL脚本的语法与应用要求更高，需要系统级的开发人员，而且更丰富的经验；**
2. **安全访问权限的设定，只有部分人员被数据库管理员授予存储过程的创建权限，而大多数用户只有使用权限，这与数据库的安全访问有关。**
3. **对于团队开发，开始标准不定好的话，后期维护比较麻烦；**
4. **复杂的业务逻辑，用存储过程编程比较吃力，但事功倍；**
5. **使用存储过程**

**（一）基本语法**

**1.创建存储过程**  
create procedure sp\_name()  
begin  
.........  
end

**2.调用存储过程**  
基本语法：call sp\_name()  
注意：存储过程名称后面必须加括号，哪怕该存储过程没有参数传递

**3.删除存储过程**  
基本语法：  
drop procedure sp\_name

注意事项：  
不能在一个存储过程中删除另一个存储过程，只能调用另一个存储过程

**4.其他常用命令**

1）show procedure status  
显示[数据库](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "https://www.cnblogs.com/alliswelltome/p/_blank)中所有存储的存储过程基本信息，包括所属数据库，存储过程名称，创建时间等

2.show create procedure sp\_name  
显示某一个[MySQL](http://lib.csdn.net/base/mysql" \o "MySQL知识库" \t "https://www.cnblogs.com/alliswelltome/p/_blank)存储过程的详细信息

**（二）变量（内存中一个特定的位置，用来临时存储数据）**

自定义变量：DECLARE   a INT ; SET a=100;

可用以下语句代替：DECLARE a INT DEFAULT 100;

变量分为用户变量和系统变量，系统变量又分为会话和全局级变量；

**用户变量：用户变量名一般以@开头，滥用用户变量会导致程序难以理解及管理**

**1、 在mysql客户端使用用户变量**mysql> SELECT 'Hello World' into @x;  
mysql> SELECT @x;

mysql> SET @y='Goodbye Cruel World';  
mysql> select @y;

mysql> SET @z=1+2+3;  
mysql> select @z;

**2、 在存储过程中使用用户变量**

mysql> CREATE PROCEDURE GreetWorld( ) SELECT CONCAT(@greeting,' World');  
mysql> SET @greeting='Hello';  
mysql> CALL GreetWorld( );

**3、 在存储过程间传递全局范围的用户变量**mysql> CREATE PROCEDURE p1( )   SET @last\_procedure='p1';  
mysql> CREATE PROCEDURE p2( ) SELECT CONCAT('Last procedure was ',@last\_procedure);  
mysql> CALL p1( );  
mysql> CALL p2( );

**（三）流程控制  
1、顺序结构**  
**2、分支结构**if ...then...else(elseif)...end if  
case...when...end case

**3、循环结构**  
for循环  
while循环  
loop循环  
repeat until循环

注：  
区块定义，常用  
begin  
......  
end;  
也可以给区块起别名，如：  
lable:begin  
...........  
end lable;  
可以用leave lable;跳出区块，执行区块以后的代码

begin和end如同[C语言](http://lib.csdn.net/base/c" \o "C语言知识库" \t "https://www.cnblogs.com/alliswelltome/p/_blank)中的{ 和 }。

1. 输入和输出

**MYSQL存储过程的参数用在存储过程的定义，共有三种参数类型,IN,OUT,INOUT**Create procedure|function([[IN |OUT |INOUT ] 参数名 数据类形...])

**IN 输入参数**  
表示该参数的值必须在调用存储过程时指定，在存储过程中修改该参数的值不能被返回，为默认值

**OUT 输出参数**  
该值可在存储过程内部被改变，并可返回

**INOUT 输入输出参数**  
调用时指定，并且可被改变和返回

**1、IN参数例子：**CREATE PROCEDURE sp\_demo\_in\_parameter(IN p\_in INT)  
BEGIN  
SELECT p\_in; --查询输入参数  
SET p\_in=2; --修改  
select p\_in;--查看修改后的值  
END;

**执行结果:**mysql> set @p\_in=1  
mysql> call sp\_demo\_in\_parameter(@p\_in)  
略  
mysql> select @p\_in;  
略  
以上可以看出，p\_in虽然在存储过程中被修改，但并不影响@p\_id的值

**2、OUT参数例子**创建:  
mysql> CREATE PROCEDURE sp\_demo\_out\_parameter(OUT p\_out INT)  
BEGIN  
SELECT p\_out;/\*查看输出参数\*/  
SET p\_out=2;/\*修改参数值\*/  
SELECT p\_out;/\*看看有否变化\*/  
END;

**执行结果:**mysql> SET @p\_out=1  
mysql> CALL sp\_demo\_out\_parameter(@p\_out)  
略

mysql> SELECT @p\_out;  
略

**3、INOUT参数例子：**mysql> CREATE PROCEDURE sp\_demo\_inout\_parameter(INOUT p\_inout INT)  
BEGIN  
SELECT p\_inout;  
SET p\_inout=2;  
SELECT p\_inout;  
END;

**执行结果：**set @p\_inout=1  
call sp\_demo\_inout\_parameter(@p\_inout) //  
略  
select @p\_inout;  
略

**例1、带输入参数的存储过程**

drop procedure if exists sp1 //

create procedure sp1(in p int)

**comment 'insert into a int value'**

begin

/\* 将输入参数的值赋给变量 \*/

set @v1 = p;（这里只是让大家学习一种变量声明方式，直接在v1前面加@，表示声明变量，也可以用declare v1 int声明）

insert into test(id) values(v1);

end

//

/\* 调用这个存储过程 \*/

call sp1(1)//

/\* 去数据库查看调用之后的结果 \*/

select \* from test//

**例2、带输出参数的存储过程**

drop procedure if exists sp2 //

create procedure sp2(out p int)

begin

select max(id) into p from test;

end//

call sp2(@pv)//（调用该存储过程，注意：输出参数必须是一个带@符号的变量）

/\* 查询刚刚在存储过程中使用到的变量 \*/

select @pv//

**例3、带输入和输出参数的存储过程**

drop procedure if exists sp3 //

create procedure sp3(in p1 int , out p2 int)

begin

if p1 = 1 then

set @v = 10;（直接在v1前面加@，表示声明变量，也可以用declare v1 int声明）

else

set @v = 20;

end if;

/\* 语句体内可以执行多条sql，但必须以分号分隔 \*/

insert into test(id) values(@v);

select max(id) into p2 from test;

end//

/\*调用该存储过程，注意：输出参数必须是一个带@符号的变量\*/

call sp3(1,@ret)//

select @ret//

**例4、既做输入又做输出参数的存储过程**

drop procedure if exists sp4 //

create procedure sp4(inout p4 int)

begin

if p4 = 4 then

set @pg = 400;

else

set @pg = 500;

end if;

select @pg;

end//

call sp4(@pp)//

/\* 这里需要先设置一个已赋值的变量，然后再作为参数传入 \*/

set @pp = 4//

call sp4(@pp)//

**\*\*MYSQL命令行客户机的分隔符**

**DELIMITER \\**

**DROP PROCEDURE IF EXISTS `test`.`new\_proc`;\\**

**CREATE PROCEDURE `test`.`new\_proc` ()**

**BEGIN**

**/\*过程体\*/**

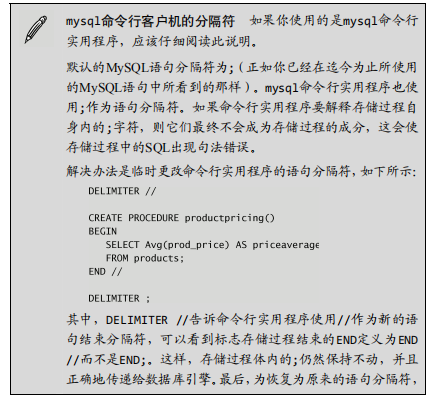
**SELECT Avg(prod\_price) AS priceaverage**

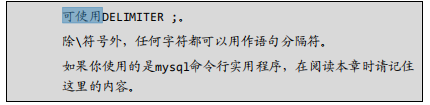
**FROM products;**

**END\\**

**DELIMITER ;**

**（MYSQL必知必会P166）**





**四、应用实例（适用于样例库）**

1、Creating Stored Procedures（创建）

CREATE PROCEDURE productpricing( )

BEGIN

SELECT Avg(prod\_price) AS priceaverage

FROM products;

END;

**（求出产品的平均价格）**

DELIMITER //

CREATE PROCEDURE productpricing()

BEGIN

SELECT Avg(prod\_price) AS priceaverage

FROM products;

END //

DELIMITER ;

2、Executing Stored Procedures（调用）

1）CALL productpricing(@pricelow,

@pricehigh,

@priceaverage);

2）CALL productpricing();

3、Dropping Stored Procedures（删除）

DROP PROCEDURE productpricing;

4、Working with Parameters（带参的存储过程）

1）带输出参数：

CREATE PROCEDURE productpricing(

OUT pl DECIMAL(8,2),

OUT ph DECIMAL(8,2),

OUT pa DECIMAL(8,2)

)

BEGIN

SELECT Min(prod\_price)

INTO pl

FROM products;

SELECT Max(prod\_price)

INTO ph

FROM products;

SELECT Avg(prod\_price)

INTO pa

FROM products;

END;

**CALL productpricing(@pricelow,@pricehigh,@priceaverage);**

**SELECT @pricehigh, @pricelow, @priceaverage;**

**（功能：分别返回产品的最高、最低、平均价格，并存入三个变量中，所有MYSQL变量在使用时都必须以@开头）**

2）带输入输出参数：

CREATE PROCEDURE ordertotal(

IN onumber INT,

OUT ototal DECIMAL(8,2)

)

BEGIN

SELECT Sum(item\_price\*quantity)

FROM orderitems

WHERE order\_num = onumber

INTO ototal;

END;

CALL ordertotal(20005, @total);

SELECT @total;

CALL ordertotal(20009, @total);

SELECT @total;

**（功能：求出某特定订单号的总金额）**

5、Building Intelligent Stored Procedures（创建智能存储过程）

-- Name: ordertotal

-- Parameters: onumber = order number

-- taxable = 0 if not taxable, 1 if taxable

-- ototal = order total variable

CREATE PROCEDURE ordertotal(

IN onumber INT,

IN taxable BOOLEAN,

OUT ototal DECIMAL(8,2)

) COMMENT ‘Obtain order total, optionally adding tax’

BEGIN

-- Declare variable for total

DECLARE total DECIMAL(8,2);

-- Declare tax percentage

DECLARE taxrate INT DEFAULT 6;

-- Get the order total

SELECT Sum(item\_price\*quantity)

FROM orderitems

WHERE order\_num = onumber

INTO total;

-- Is this taxable?

IF taxable THEN

-- Yes, so add taxrate to the total

SELECT total+(total/100\*taxrate) INTO total;

END IF;

-- And finally, save to out variable

SELECT total INTO ototal;

END;

CALL ordertotal(20005, 0, @total);

SELECT @total;

CALL ordertotal(20005, 1, @total);

SELECT @total;

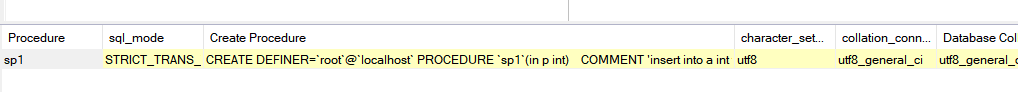
**（求是否含税的给定订单总金额）**

6、Inspecting Stored Procedures（检查存储过程）

1）SHOW CREATE PROCEDURE ordertotal;

**（功能：显示用于创建一个存储过程ordertotal的CREATE语句，即定义语句的本身）**

**例：**



1. SHOW PROCEDURE STATUS LIKE 'ordertotal' ;

**（功能：获得当前数据库中包括何时、何人创建等详细信息的存储过程列表，可以限制或指定一个过滤模式，如LIKE关键字的使用。）**

**例：**

