一、MYSQL储存过程简介：

储存过程是一个可编程的函数，它在数据库中创建并保存。它可以有SQL语句和一些特殊的控制结构组成。当希望在不同的应用程序或平台上执行相同的函数，或者封装特定功能时，存储过程是非常有用的。数据库中的存储过程可以看做是对编程中面向对象方法的模拟。它允许控制数据的访问方式。存储过程通常有以下优点：

1)存储过程能实现较快的执行速度。

如果某一操作包含大量的Transaction-SQL代码或分别被多次执行，那么存储过程要比批处理的执行速度快很多。因为存储过程是预编译的。在首次运行一个存储过程时查询，优化器对其进行分析优化，并且给出最终被存储在系统表中的执行计划。而批处理的Transaction-SQL语句在每次运行时都要进行编译和优化，速度相对要慢一些。

心得：编译优化，快！

2)存储过程允许标准组件是编程。

存储过程被创建后，可以在程序中被多次调用，而不必重新编写该存储过程的SQL语句。而且数据库专业人员可以随时对存储过程进行修改，对应用程序源代码毫无影响。

心得：封装与抽象，简单调用

3)存储过程可以用流控制语句编写，有很强的灵活性，可以完成复杂的判断和较复杂的运算。

心得：功能强大，逻辑强大

4)存储过程可被作为一种安全机制来充分利用。

系统管理员通过执行某一存储过程的权限进行限制，能够实现对相应的数据的访问权限的限制，避免了非授权用户对数据的访问，保证了数据的安全。

心得：限制与安全

5)存储过程能过减少网络流量。

针对同一个数据库对象的操作（如查询、修改），如果这一操作所涉及的Transaction-SQL语句被组织程存储过程，那么当在客户计算机上调用该存储过程时，网络中传送的只是该调用语句，从而大大增加了网络流量并降低了网络负载。

心得：减少网络流量（封装的好）

二. MySQL 创建一个最简单的存储过程

“pr\_add” 是个简单的 MySQL 存储过程，这个存储过程有两个 int 类型的输入参数 “a”、“b”，返回这两个参数的和。

drop procedure if exists pr\_add;//如果存储过程pr\_add存在，则删去；如果不存在，则什么事也不做。如果没有这句代码当存储过程pr\_add存在时，下面的创建会报错

//计算两个数之和

create PROCEDURE pr\_add

(

in a int,

in b int

)

begin

declare c int;//declare是声明变量的关键字

if a is null THEN

set a=1;

end if;

if b is null THEN

set b=1;

end if;

set c=a+b;

select c as sum;

end;

call pr\_add( null,null );//这是调用存储过程pr\_add

三、MySQL 存储过程特点

创建 MySQL 存储过程的简单语法为：

create procedure 存储过程名字()

(

[in|out|inout] 参数 datatype

)

begin

MySQL 语句;

end;

MySQL 存储过程参数如果不显式指定“in”、“out”、“inout”，则默认为“in”。习惯上，对于是“in” 的参数，我们都不会显式指定。

1 MySQL 存储过程名字后面的“()”是必须的，即使没有一个参数，也需要“()”

2 MySQL 存储过程参数，不能在参数名称前加“@”，如：“@a int”。下面的创建存储过程语法在 MySQL 中是错误的（在 SQL Server 中是正确的）。 MySQL 存储过程中的变量，不需要在变量名字前加“@”，虽然 MySQL 客户端用户变量要加个“@”。

create procedure pr\_add

(

@a int,// 错误

b int //正确

)

3 MySQL 存储过程的参数不能指定默认值。

4 MySQL 存储过程不需要在 procedure body 前面加 “as”。而 SQL Server 存储过程必须加 “as” 关键字。

create procedure pr\_add

(

a int,

b int

)

as - 错误，MySQL 不需要 “as”

begin

mysql statement ...;

end;

5 如果 MySQL 存储过程中包含多条 MySQL 语句，则需要 begin end 关键字。

create procedure pr\_add

(

a int,

b int

)

begin

mysql statement 1 ...;

mysql statement 2 ...;

end;

6 MySQL 存储过程中的每条语句的末尾，都要加上分号 “;”

...

declare c int;

if a is null then

set a = 0;

end if;

...

end;

7 不能在 MySQL 存储过程中使用 “return” 关键字。

set c = a + b;

select c as sum;

/\*

return c;- 不能在 MySQL 存储过程中使用。return 只能出现在函数中。

\*/

end;

8 调用 MySQL 存储过程时候，需要在过程名字后面加“()”，即使没有一个参数，也需要“()”，调用out及inout参数格式为@arguments\_name形式。

call pr\_no\_param();

1

9 因为 MySQL 存储过程参数没有默认值，所以在调用 MySQL 存储过程时候，不能省略参数。可以用 null 来替代。

call pr\_add(10, null);

1

1，实战前提

需要MySQL 5及以上 ，我用的是MYSQL的客户端Navicat Premium，贴出的代码都是我变异没有错误的。如果读者没有安装客户端或者在你的电脑上报错，这里需要用到是DELIMITER //和DELIMITER ;两句，DELIMITER是分割符的意思，因为MySQL默认以”;”为分隔符，如果我们没有声明分割符，那么编译器会把存储过程当成SQL语句进行处理，则存储过程的编译过程会报错，所以要事先用DELIMITER关键字申明当前段分隔符，这样MySQL才会将”;”当做存储过程中的代码，不会执行这些代码，用完了之后要把分隔符还原

2，变量

使用DECLARE来声明，DEFAULT赋默认值，SET赋值

Java代码 复制代码

DECLARE counter INT DEFAULT 0;

SET counter = counter+1;

3，条件判断

IF THEN、ELSEIF、ELSE、END IF

DROP PROCEDURE IF EXISTS discounted\_price;

CREATE PROCEDURE discounted\_price(normal\_price NUMERIC(8, 2), OUT discount\_price NUMERIC(8, 2))

BEGIN

IF (normal\_price > 500) THEN

SET discount\_price = normal\_price \* 0.8;

ELSEIF (normal\_price > 100 and normal\_price<=500) THEN

SET discount\_price = normal\_price \* 0.9;

ELSE

SET discount\_price = normal\_price;

END IF;

select discount\_price as price;

END;

call discounted\_price(600.0,@discount);//out参数调用时可以用@任意字符串

4，循环

LOOP、END LOOP

drop procedure if exists simple\_loop;

create procedure simple\_loop(out counter int)

BEGIN

declare temp int default 0;

set counter=0;

my\_loop:LOOP

set counter=counter+1;

set temp=temp+1;

if counter=10 THEN

leave my\_loop;

end if;

end loop my\_loop;

select temp as result;

end;

call simple\_loop(@a);

5、WHILE DO、END WHILE

DROP PROCEDURE IF EXISTS simple\_while;

CREATE PROCEDURE simple\_while(OUT counter INT)

BEGIN

declare temp int default 0;

SET counter =0;

WHILE counter != 10 DO

SET counter =counter+1;

set temp =temp+1;

END WHILE;

select counter as temp1;

END;

call simple\_while(@a);

6、REPEAT、UNTILL

drop PROCEDURE if exists simple\_repeat;

create procedure simple\_repeat(out counter int)

BEGIN

set counter=0;

REPEAT

set counter=counter+1;

until counter=10 end repeat;

select counter as temp;

end;

call simple\_repeat(@q);

7，存储方法

存储方法与存储过程的区别

1，存储方法的参数列表只允许IN类型的参数，而且没必要也不允许指定IN关键字

2，存储方法返回一个单一的值，值的类型在存储方法的头部定义

3，存储方法可以在SQL语句内部调用

4，存储方法不能返回结果集

语法：

　create function 函数([函数参数[,….]]) Returns 返回类型

　　Begin

　　　　If

　　　　　　Return (返回的数据)

　　　　Else

　　　　　　Return (返回的数据)

　　　　end if;

　　end;

一个简单的存储函数实例

drop function if exists purchase\_and\_redeem\_function;

CREATE function purchase\_and\_redeem\_function(date int)

returns varchar(80)

BEGIN

return (SELECT tbalance

FROM user\_purchase\_and\_redeem

WHERE report\_date=date); //这里面的SQL语句根据自己数据库表编写

END;

select purchase\_and\_redeem\_function(20140501);//这是调用存储函数

8，触发器

触发器在INSERT、UPDATE或DELETE等DML语句修改数据库表时触发

触发器的典型应用场景是重要的业务逻辑、提高性能、监控表的修改等

触发器可以在DML语句执行前或后触发

DROP TRIGGER sales\_trigger;

CREATE TRIGGER sales\_trigger

BEFORE INSERT ON sales

FOR EACH ROW

BEGIN

IF NEW.sale\_value > 500 THEN

SET NEW.free\_shipping = 'Y';

ELSE

SET NEW.free\_shipping = 'N';

END IF;

IF NEW.sale\_value > 1000 THEN

SET NEW.discount = NEW.sale\_value \* .15;

ELSE

SET NEW.discount = 0;

END IF;

END;