**[MySQL存储过程的创建及调用](https://www.cnblogs.com/geaozhang/p/6797357.html)**

*阅读目录：MySQL存储过程\_创建-调用-参数*

* *存储过程：SQL中的“脚本”*

*1.*[*创建存储过程*](https://www.cnblogs.com/geaozhang/p/6797357.html#chuangjian)

*2.*[*调用存储过程*](https://www.cnblogs.com/geaozhang/p/6797357.html#diaoyong)

*3.*[*存储过程体*](https://www.cnblogs.com/geaozhang/p/6797357.html#guochengti)

*4.*[*语句块标签*](https://www.cnblogs.com/geaozhang/p/6797357.html#biaoqian)

* [*存储过程的参数*](https://www.cnblogs.com/geaozhang/p/6797357.html#canshu)

*1.*[*in*](https://www.cnblogs.com/geaozhang/p/6797357.html#in)*：向过程里传参*

*2.*[*out*](https://www.cnblogs.com/geaozhang/p/6797357.html#out)*：过程向外传参值*

*3.*[*inout*](https://www.cnblogs.com/geaozhang/p/6797357.html#inout)*：in and out*

# SQL语句：先编译后执行

存储过程(Stored Procedure)：

　　一组可编程的函数，是为了完成特定功能的SQL语句集，经编译创建并保存在数据库中，用户可通过指定存储过程的名字并给定参数(需要时)来调用执行。

优点(为什么要用存储过程？)：

　　①将重复性很高的一些操作，封装到一个存储过程中，简化了对这些SQL的调用

②批量处理：SQL+循环，减少流量，也就是“跑批”

备注：（跑批：把sql语句做成批处理语句，保存为bat文件，然后运行该bat文件）

　　③统一接口，确保数据的安全

相对于oracle数据库来说，MySQL的存储过程相对功能较弱，使用较少。

一、存储过程的创建和调用

　　>存储过程就是具有名字的一段代码，用来完成一个特定的功能。

　　>创建的存储过程保存在数据库的数据字典中。

1、创建存储过程

CREATE

[DEFINER = { user | CURRENT\_USER }]

　PROCEDURE sp\_name ([proc\_parameter[,...]])

[characteristic ...] routine\_body

proc\_parameter:

[ IN | OUT | INOUT ] param\_name type

characteristic:

COMMENT 'string'

| LANGUAGE SQL

| [NOT] DETERMINISTIC

| { CONTAINS SQL | NO SQL | READS SQL DATA | MODIFIES SQL DATA }

| SQL SECURITY { DEFINER | INVOKER }

routine\_body:

　　Valid SQL routine statement

[begin\_label:] BEGIN

　　[statement\_list]

　　　　……

END [end\_label]

 #创建数据库，备份数据表用于示例操作

mysql> create database db1;

mysql> use db1;

mysql> create table PLAYERS as select \* from TENNIS.PLAYERS;

mysql> create table MATCHES as select \* from TENNIS.MATCHES;

（解释：上面语句的意思就是先获取到TENNIS.PLAYERS表中的所有记录，之后创建一张PLAYERS表，结构和TENNIS.PLAYERS表相同，记录为后面语句的查询结果。) 这是复制一张表最简单的方法

示例：创建一个存储过程，删除给定球员参加的所有比赛

mysql> delimiter $$　　#将语句的结束符号从分号;临时改为两个$$(可以是自定义)

mysql> CREATE PROCEDURE delete\_matches(IN p\_playerno INTEGER)

-> BEGIN

-> 　　DELETE FROM MATCHES

-> WHERE playerno = p\_playerno;

-> END$$

Query OK, 0 rows affected (0.01 sec)

mysql> delimiter ;　　#将语句的结束符号恢复为分号

解析：

　　默认情况下，存储过程和默认数据库相关联，如果想指定存储过程创建在某个特定的数据库下，那么在过程名前面加数据库名做前缀；

　　在定义过程时，使用DELIMITER $$ 命令将语句的结束符号从分号 ; 临时改为两个$$，使得过程体中使用的分号被直接传递到服务器，而不会被客户端（如mysql）解释。

2、调用存储过程：call sp\_name[(传参)];

mysql> select \* from MATCHES;

+---------+--------+----------+-----+------+

| MATCHNO | TEAMNO | PLAYERNO | WON | LOST |

+---------+--------+----------+-----+------+

| 1 | 1 | 6 | 3 | 1 |

| 7 | 1 | 57 | 3 | 0 |

| 8 | 1 | 8 | 0 | 3 |

| 9 | 2 | 27 | 3 | 2 |

| 11 | 2 | 112 | 2 | 3 |

+---------+--------+----------+-----+------+

5 rows in set (0.00 sec)

mysql> call delete\_matches(57);

Query OK, 1 row affected (0.03 sec)

mysql> select \* from MATCHES;

+---------+--------+----------+-----+------+

| MATCHNO | TEAMNO | PLAYERNO | WON | LOST |

+---------+--------+----------+-----+------+

| 1 | 1 | 6 | 3 | 1 |

| 8 | 1 | 8 | 0 | 3 |

| 9 | 2 | 27 | 3 | 2 |

| 11 | 2 | 112 | 2 | 3 |

+---------+--------+----------+-----+------+

4 rows in set (0.00 sec)

解析：

　　在存储过程中设置了需要传参的变量p\_playerno，调用存储过程的时候，通过传参将57赋值给p\_playerno，然后进行存储过程里的SQL操作。

3、存储过程体

　　>存储过程体包含了在过程调用时必须执行的语句，例如：dml、ddl语句，if-then-else和while-do语句、声明变量的declare语句等

　　>过程体格式：以begin开始，以end结束(可嵌套)

BEGIN

　　BEGIN

　　　　BEGIN

　　　　　　statements;

　　　　END

　　END

END

注意：每个嵌套块及其中的每条语句，必须以分号结束，表示过程体结束的begin-end块(又叫做复合语句compound statement)，则不需要分号。

4、为语句块贴标签

[begin\_label:] BEGIN

　　[statement\_list]

END [end\_label]

例如：

label1: BEGIN

　　label2: BEGIN

　　　　label3: BEGIN

　　　　　　statements;

　　　　END label3 ;

　　END label2;

END label1

标签有两个作用：

　　①增强代码的可读性

　　②在某些语句(例如:leave和iterate语句)，需要用到标签

二、存储过程的参数

　　存储过程可以有0个或多个参数，用于存储过程的定义。

3种参数类型：

　　IN输入参数：表示调用者向过程传入值（传入值可以是字面量或变量）

　　OUT输出参数：表示过程向调用者传出值(可以返回多个值)（传出值只能是变量）

　　INOUT输入输出参数：既表示调用者向过程传入值，又表示过程向调用者传出值（值只能是变量）

1、in输入参数

mysql> delimiter $$

mysql> create procedure in\_param(in p\_in int)

-> begin

-> 　　select p\_in;

-> 　　set p\_in=2;

-> select P\_in;

-> end$$

mysql> delimiter ;

mysql> set @p\_in=1;

mysql> call in\_param(@p\_in);

+------+

| p\_in |

+------+

| 1 |

+------+

+------+

| P\_in |

+------+

| 2 |

+------+

mysql> select @p\_in;

+-------+

| @p\_in |

+-------+

| 1 |

+-------+

#以上可以看出，p\_in在存储过程中被修改，但并不影响@p\_id的值，因为前者为局部变量、后者为全局变量。

2、out输出参数

mysql> delimiter //

mysql> create procedure out\_param(out p\_out int)

-> begin

-> select p\_out;

-> set p\_out=2;

-> select p\_out;

-> end

-> //

mysql> delimiter ;

mysql> set @p\_out=1;

mysql> call out\_param(@p\_out);

+-------+

| p\_out |

+-------+

| NULL |

+-------+

　　#因为out是向调用者输出参数，不接收输入的参数，所以存储过程里的p\_out为null

+-------+

| p\_out |

+-------+

| 2 |

+-------+

mysql> select @p\_out;

+--------+

| @p\_out |

+--------+

| 2 |

+--------+

　　#调用了out\_param存储过程，输出参数，改变了p\_out变量的值

3、inout输入参数

mysql> delimiter $$

mysql> create procedure inout\_param(inout p\_inout int)

-> begin

-> select p\_inout;

-> set p\_inout=2;

-> select p\_inout;

-> end

-> $$

mysql> delimiter ;

mysql> set @p\_inout=1;

mysql> call inout\_param(@p\_inout);

+---------+

| p\_inout |

+---------+

| 1 |

+---------+

+---------+

| p\_inout |

+---------+

| 2 |

+---------+

mysql> select @p\_inout;

+----------+

| @p\_inout |

+----------+

| 2 |

+----------+

#调用了inout\_param存储过程，接受了输入的参数，也输出参数，改变了变量

注意：

　　①如果过程没有参数，也必须在过程名后面写上小括号

　　　　例：CREATE PROCEDURE sp\_name ([proc\_parameter[,...]]) ……

　　②确保参数的名字不等于列的名字，否则在过程体中，参数名被当做列名来处理

墙裂建议：

*>输入值使用in参数；*

*>返回值使用out参数；*

*>inout参数就尽量的少用。*

@author：http://www.cnblogs.com/geaozhang/

分类: [MySQL](https://www.cnblogs.com/geaozhang/category/1326927.html)