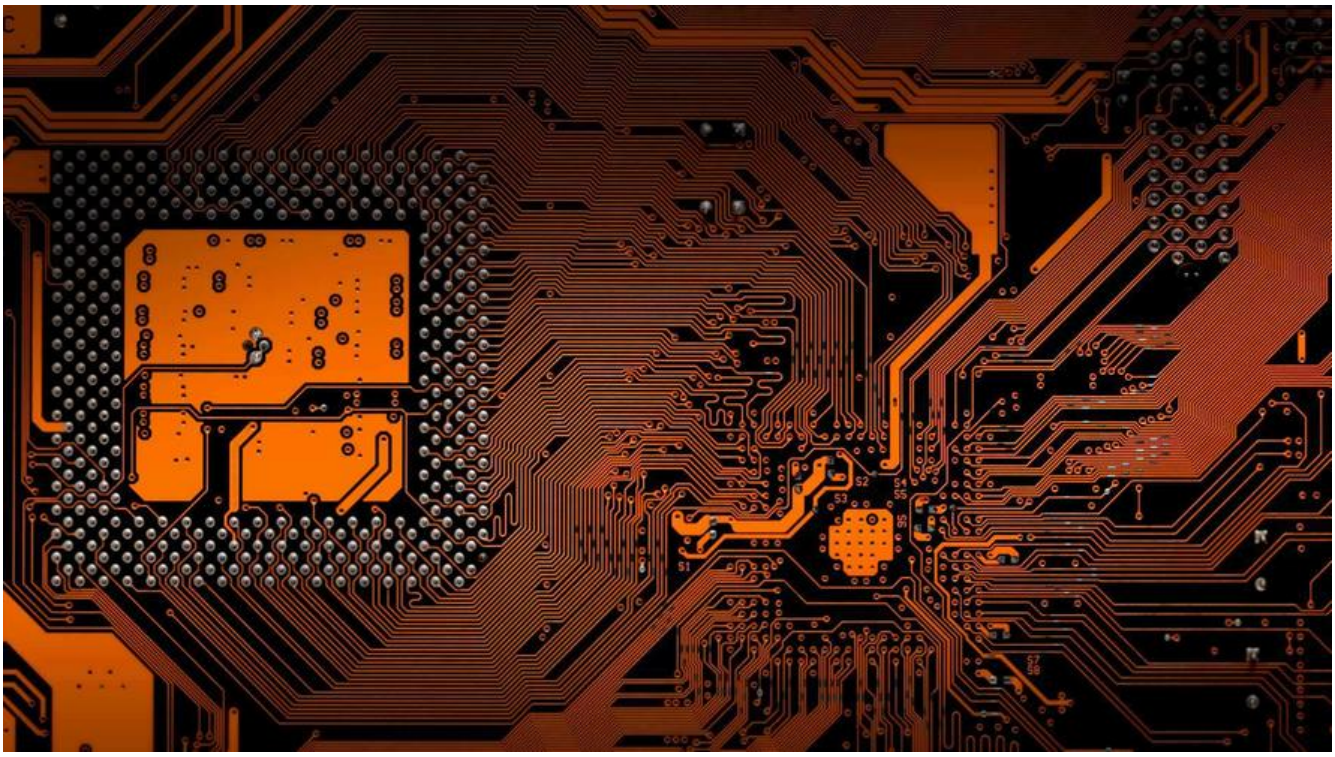
链滴

mysql 存储过程详细教程

作者：Ethan

原文链接：https://ld246.com/article/1523692259321

来源网站：链滴



** 记录MYSQL存储过程中的关键语法：**

DELIMITER // 声明语句结束符，用于区分;

CEATE PROCEDURE demo_in_
arameter(IN p_in int) 声明存储过程

BEGIN .... END 存储过程开始和结束符号

SET @p_in=1 变量赋值

DECLARE l_int int unsigned default 4000000; 变量定义

什么是mysql存储例程?

存储例程是存储在数据库服务器中的一组sql语句，通过在查询中调用一个指定的名称来执行这些sql句命令.

为什么要使用mysql存储过程?

我们都知道应用程序分为两种，一种是基于web，一种是基于桌面，他们都和数据库进行交互来完成据的存取工作。假设现在有一种应用程序包含了这两 种，现在要修改其中的一个查询sql语句，那么们可能要同时修改他们中对应的查询sql语句，当我们的应用程序很庞大很复杂的时候问题就出现这不易维 护！另外把sql查询语句放在我们的web程序或桌面中很容易遭到sql注入的破坏。而存储例程好可以帮我们解决这些问题。

存储过程(stored procedure)、存储例程(store routine)、存储函数区别

Mysql存储例程实际包含了存储过程和存储函数，它们被统称为存储例程。

其中存储过程主要完成在获取记录或插入记录或更新记录或删除记录，即完成select insert delete up ate等的工作。而存储函数只完成查询的工作，可接受输入参数并返回一个结果。

创建mysql存储过程、存储函数

create procedure 存储过程名(参数)

存储过程体

create function 存储函数名(参数)

下面是存储过程的例子：

```
mysql> DELIMITER //
mysql> CREATE PROCEDURE proc1(OUT s int)
  -> BEGIN
  -> SELECT COUNT(*) INTO s FROM user;
  -> END
  -> //
mysql> DELIMITER ;
```

注：

（1）这里需要注意的是DELIMITER//和DELIMITER;两句，DELIMITER是分割符的意思，因为MySQ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）存储过程根据需要可能会有输入、输出、输入输出参数，这里有一个输出参数s，类型是int型，果有多个参数用”,”分割开。

（3）过程体的开始与结束使用BEGIN与END进行标识。

这样，我们的一个MySQL存储过程就完成了，是不是很容易呢?看不懂也没关系，接下来，我们详细讲解。

参数

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,形式如：

CREATEPROCEDURE 存储过程名([[
N |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

IN参数例子

创建:

```
1.  mysql > DELIMITER //
2.  mysql > CREATE PROCEDURE demo_in_parameter(IN p_in int)
3.       -> BEGIN
4.       -> SELECT p_in;
```

```
5.      -> SET p_in=2;
6.      -> SELECT p_in;
7.      -> END;
8.      -> //
9.  mysql > DELIMITER ;
```

执行结果:

```
1.  mysql > SET @p_in=1;
2.  mysql > CALL demo_in_parameter(@p_in);
3.  +------+
4.  | p_in |
5.  +------+
6.  |   1  |
7.  +------+
8.
9.  +------+
10.| p_in |
11.+------+
12.|   2  |
13.+------+
14.
15.mysql> SELECT @p_in;
16.+-------+
17.| @p_in |
18.+-------+
19.|  1    |
20.+-------+
```

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_
d的值

OUT参数例子

创建:

```
1.  mysql > DELIMITER //
2.  mysql > CREATE PROCEDURE demo_out_parameter(OUT p_out int)
3.      -> BEGIN
4.      -> SELECT p_out;
5.      -> SET p_out=2;
6.      -> SELECT p_out;
7.      -> END;
8.      -> //
9.  mysql > DELIMITER ;
```

执行结果:

```
1.  mysql > SET @p_out=1;
2.  mysql > CALL sp_demo_out_parameter(@p_out);
3.  +-------+
```

```
| p_out |
+-------+
| NULL  |
+-------+

+-------+
| p_out |
+-------+
|   2   |
+-------+

mysql> SELECT @p_out;
+-------+
| p_out |
+-------+
|   2   |
+-------+
```

INOUT参数例子

创建:

```
mysql > DELIMITER //
mysql > CREATE PROCEDURE demo_inout_parameter(INOUT p_inout int)
    -> BEGIN
    -> SELECT p_inout;
    -> SET p_inout=2;
    -> SELECT p_inout;
    -> END;
    -> //
mysql > DELIMITER ;
```

执行结果:

```
mysql > SET @p_inout=1;
mysql > CALL demo_inout_parameter(@p_inout) ;
+----------+
| p_inout |
+----------+
|    1    |
+----------+

+----------+
| p_inout |
+----------+
|    2    |
+----------+

mysql > SELECT @p_inout;
+-----------+
| @p_inout |
+-----------+
```

```
19.|    2     |
20.+----------+
```

局部变量

变量定义

局部变量声明一定要放在存储过程体的开始

DECLAREvariable_name [,variable_
ame...] datatype [DEFAULT value];

其中，datatype为MySQL的数据类型，如:int, float, date,varchar(length)

例如:

```
1.  DECLARE l_int int unsigned default 4000000;
2.  DECLARE l_numeric number(8,2) DEFAULT 9.95;
3.  DECLARE l_date date DEFAULT '1999-12-31';
4.  DECLARE l_datetime datetime DEFAULT '1999-12-31 23:59:59';
5.  DECLARE l_varchar varchar(255) DEFAULT 'This will not be padded';
```

变量赋值

SET 变量名 = 表达式值 [,variable_name = expression ...]

用户变量

在MySQL客户端使用用户变量

```
1.  mysql > SELECT 'Hello World' into @x;
2.  mysql > SELECT @x;
3.  +-------------+
4.  |   @x        |
5.  +-------------+
6.  | Hello World |
7.  +-------------+
8.  mysql > SET @y='Goodbye Cruel World';
9.  mysql > SELECT @y;
10.+---------------------+
11.|     @y              |
12.+---------------------+
13.| Goodbye Cruel World |
14.+---------------------+
15.
16.mysql > SET @z=1+2+3;
17.mysql > SELECT @z;
18.+------+
19.| @z   |
20.+------+
21.|  6   |
```

22.+------+

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( );
+---------------------------+
| CONCAT(@greeting,' World') |
+---------------------------+
|  Hello World              |
+---------------------------+
```

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( );
+--------------------------------------------+
| CONCAT('Last procedure was ',@last_proc      |
+--------------------------------------------+
| Last procedure was p1                        |
+--------------------------------------------+
```

注意:

①用户变量名一般以@开头

②滥用用户变量会导致程序难以理解及管理

注释

MySQL存储过程可使用两种风格的注释

双模杠：–

该风格一般用于单行注释

c风格： 一般用于多行注释

例如：

```
mysql > DELIMITER //
mysql > CREATE PROCEDURE proc1 --name存储过程名
    -> (IN parameter1 INTEGER)
    -> BEGIN
    -> DECLARE variable1 CHAR(10);
    -> IF parameter1 = 17 THEN
    -> SET variable1 = 'birds';
    -> ELSE
    -> SET variable1 = 'beasts';
```

```
10.     -> END IF;
11.     -> INSERT INTO table1 VALUES (variable1);
12.     -> END
13.     -> //
14.mysql > DELIMITER ;
```

MySQL存储过程的调用

用call和你过程名以及一个括号，括号里面根据需要，加入参数，参数包括输入参数、输出参数、输输出参数。具体的调用方法可以参看上面的例子。

MySQL存储过程的查询

我们像知道一个数据库下面有那些表，我们一般采用showtables;进行查看。那么我们要查看某个数库下面的存储过程，是否也可以采用呢？答案是，我们可以查看某个数据库下面的存储过程，但是是一钟方式。

我们可以用

selectname from mysql.proc where db=’数据库名’；

或者

selectroutine_name from information_
chema.routines where routine_schema=’数据库名’；

或者

showprocedure status where db=’数据库名’；

进行查询。

如果我们想知道，某个存储过程的详细，那我们又该怎么做呢？是不是也可以像操作表一样用describe表名进行查看呢？

答案是：我们可以查看存储过程的详细，但是需要用另一种方法：

SHOWCREATE PROCEDURE 数据库.存储过程名;

就可以查看当前存储过程的详细。

MySQL存储过程的修改

ALTER PROCEDURE

更改用CREATE PROCEDURE 建立的预先指定的存储过程，其不会影响相关存储过程或存储功能。

MySQL存储过程的删除

删除一个存储过程比较简单，和删除表一样：

DROPPROCEDURE

从MySQL的表格中删除一个或多个存储过程。

MySQL存储过程的控制语句

变量作用域

内部的变量在其作用域范围内享有更高的优先权，当执行到end。变量时，内部变量消失，此时已经其作用域外，变量不再可见了，应为在存储

过程外再也不能找到这个申明的变量，但是你可以通过out参数或者将其值指派

给会话变量来保存其值。

```
1.  mysql > DELIMITER //
2.  mysql > CREATE PROCEDURE proc3()
3.       -> begin
4.       -> declare x1 varchar(5) default 'outer';
5.       -> begin
6.       -> declare x1 varchar(5) default 'inner';
7.       -> select x1;
8.       -> end;
9.       -> select x1;
10.      -> end;
11.      -> //
12.mysql > DELIMITER ;
```

条件语句

if-then -else语句

```
1.  mysql > DELIMITER //
2.  mysql > CREATE PROCEDURE proc2(IN parameter int)
3.       -> begin
4.       -> declare var int;
5.       -> set var=parameter+1;
6.       -> if var=0 then
7.       -> insert into t values(17);
8.       -> end if;
9.       -> if parameter=0 then
10.      -> update t set s1=s1+1;
11.      -> else
12.      -> update t set s1=s1+2;
13.      -> end if;
14.      -> end;
15.      -> //
16.mysql > DELIMITER ;
```

case语句：

```
1.  mysql > DELIMITER //
2.  mysql > CREATE PROCEDURE proc3 (in parameter int)
3.       -> begin
4.       -> declare var int;
5.       -> set var=parameter+1;
6.       -> case var
7.       -> when 0 then
8.       -> insert into t values(17);
```

```
9.      -> when 1 then
10.     -> insert into t values(18);
11.     -> else
12.     -> insert into t values(19);
13.     -> end case;
14.     -> end;
15.     -> //
16.mysql > DELIMITER ;
case
        when var=0 then
                insert into t values(30);
        when var>0 then
        when var<0 then
        else

end case
```

循环语句

while •••• end while：

```
1.  mysql > DELIMITER //
2.  mysql > CREATE PROCEDURE proc4()
3.      -> begin
4.      -> declare var int;
5.      -> set var=0;
6.      -> while var<6 do
7.      -> insert into t values(var);
8.      -> set var=var+1;
9.      -> end while;
10.     -> end;
11.     -> //
12.mysql > DELIMITER ;
 while条件 do
--循环体
endwhile
```

repeat•••• end repeat：

它在执行操作后检查结果，而while则是执行前进行检查。

```
1.  mysql > DELIMITER //
2.  mysql > CREATE PROCEDURE proc5 ()
3.      -> begin
4.      -> declare v int;
5.      -> set v=0;
6.      -> repeat
7.      -> insert into t values(v);
8.      -> set v=v+1;
9.      -> until v>=5
```

```
10.     -> end repeat;
11.     -> end;
12.     -> //
13.mysql > DELIMITER ;
 repeat
--循环体
until循环条件
endrepeat;
```

loop •••••endloop:

loop循环不需要初始条件，这点和while 循环相似，同时和repeat循环一样不需要结束条件, leave语的意义是离开循环。

```
1.  mysql > DELIMITER //
2.  mysql > CREATE PROCEDURE proc6 ()
3.       -> begin
4.       -> declare v int;
5.       -> set v=0;
6.       -> LOOP_LABLE:loop
7.       -> insert into t values(v);
8.       -> set v=v+1;
9.       -> if v >=5 then
10.     -> leave LOOP_LABLE;
11.     -> end if;
12.     -> end loop;
13.     -> end;
14.     -> //
15.mysql > DELIMITER ;
```

LABLES 标号：

标号可以用在begin repeat while 或者loop 语句前，语句标号只能在合法的语句前面使用。可以跳循环，使运行指令达到复合语句的最后一步。

ITERATE迭代

ITERATE:

```
1.    通过引用复合语句的标号,来从新开始复合语句
2.  mysql > DELIMITER //
3.  mysql > CREATE PROCEDURE proc10 ()
4.       -> begin
5.       -> declare v int;
6.       -> set v=0;
7.       -> LOOP_LABLE:loop
8.      -> if v=3 then
9.      -> set v=v+1;
10.       -> ITERATE LOOP_LABLE;
11.       -> end if;
```

```
12.     -> insert into t values(v);
13.     -> set v=v+1;
14.       -> if v>=5 then
15.       -> leave LOOP_LABLE;
16.     -> end if;
17.     -> end loop;
18.     -> end;
19.     -> //
20.mysql > DELIMITER ;
```

MySQL存储过程的基本函数

字符串类

```
CHARSET(str) //返回字串字符集
CONCAT (string2 [,... ]) //连接字串
INSTR (string ,substring ) //返回substring首次在string中出现的位置,不存在返回0
LCASE (string2 ) //转换成小写
LEFT (string2 ,length ) //从string2中的左边起取length个字符
LENGTH (string ) //string长度
LOAD_FILE (file_name ) //从文件读取内容
LOCATE (substring , string [,start_position ] ) 同INSTR,但可指定开始位置
LPAD (string2 ,length ,pad ) //重复用pad加在string开头,直到字串长度为length
LTRIM (string2 ) //去除前端空格
REPEAT (string2 ,count ) //重复count次
REPLACE (str ,search_str ,replace_str ) //在str中用replace_str替换search_str
RPAD (string2 ,length ,pad) //在str后用pad补充,直到长度为length
RTRIM (string2 ) //去除后端空格
STRCMP (string1 ,string2 ) //逐字符比较两字串大小,
SUBSTRING (str , position [,length ]) //从str的position开始,取length个字符,
```

注：mysql中处理字符串时，默认第一个字符下标为1，即参数position必须大于等于1

```
1.  mysql> select substring('abcd',0,2);
2.  +-----------------------+
3.  | substring('abcd',0,2) |
4.  +-----------------------+
5.  |                       |
6.  +-----------------------+
7.  1 row in set (0.00 sec)
8.
9.  mysql> select substring('abcd',1,2);
10.+-----------------------+
11.| substring('abcd',1,2) |
12.+-----------------------+
13.|     ab                |
14.+-----------------------+
15.1 row in set (0.02 sec)
```

```
TRIM([[BOTH|LEADING|TRAILING][padding] FROM]string2) //去除指定位置的指定字符
UCASE (string2 ) //转换成大写
RIGHT(string2,length) //取string2最后length个字符
SPACE(count) //生成count个空格
```

数学类

ABS (number2) //绝对值
BIN (decimal_number) //十进制转二进制
CEILING (number2) //向上取整
CONV(number2,from_base,to_base) //进制转换
FLOOR (number2) //向下取整
FORMAT (number,decimal_places) //保留小数位数
HEX (DecimalNumber) //转十六进制
注：HEX()中可传入字符串，则返回其ASC-11码，如HEX('DEF')返回4142143
也可以传入十进制整数，返回其十六进制编码，如HEX(25)返回19
LEAST (number , number2 [,..]) //求最小值
MOD (numerator ,denominator) //求余
POWER (number ,power) //求指数
RAND([seed]) //随机数
ROUND (number [,decimals]) //四舍五入,decimals为小数位数]
注：返回类型并非均为整数，如：
(1)默认变为整形值

```
mysql> select round(1.23);
+-------------+
| round(1.23) |
+-------------+
|           1 |
+-------------+
1 row in set (0.00 sec)

mysql> select round(1.56);
+-------------+
| round(1.56) |
+-------------+
|           2 |
+-------------+
1 row in set (0.00 sec)
```

(2)可以设定小数位数，返回浮点型数据

```
mysql> select round(1.567,2);
+----------------+
| round(1.567,2) |
+----------------+
|           1.57 |
+----------------+
1 row in set (0.00 sec)
```

SIGN (number2) //

日期时间类

ADDTIME (date2 ,time_interval)//将time_interval加到date2
CONVERT_TZ (datetime2 ,fromTZ ,toTZ) //转换时区
CURRENT_DATE ( ) //当前日期
CURRENT_TIME ( ) //当前时间
CURRENT_TIMESTAMP ( ) //当前时间戳
DATE (datetime) //返回datetime的日期部分
DATE_ADD (date2 , INTERVAL d_value d_type) //在date2中加上日期或时间
DATE_FORMAT (datetime ,FormatCodes) //使用formatcodes格式显示datetime

```
DATE_SUB (date2 , INTERVAL d_value d_type ) //在date2上减去一个时间
DATEDIFF (date1 ,date2 ) //两个日期差
DAY (date ) //返回日期的天
DAYNAME (date ) //英文星期
DAYOFWEEK (date ) //星期(1-7) ,1为星期天
DAYOFYEAR (date ) //一年中的第几天
EXTRACT (interval_name FROM date ) //从date中提取日期的指定部分
MAKEDATE (year ,day ) //给出年及年中的第几天,生成日期串
MAKETIME (hour ,minute ,second ) //生成时间串
MONTHNAME (date ) //英文月份名
NOW ( ) //当前时间
SEC_TO_TIME (seconds ) //秒数转成时间
STR_TO_DATE (string ,format ) //字串转成时间,以format格式显示
TIMEDIFF (datetime1 ,datetime2 ) //两个时间差
TIME_TO_SEC (time ) //时间转秒数]
WEEK (date_time [,start_of_week ]) //第几周
YEAR (datetime ) //年份
DAYOFMONTH(datetime) //月的第几天
HOUR(datetime) //小时
LAST_DAY(date) //date的月的最后日期
MICROSECOND(datetime) //微秒
MONTH(datetime) //月
MINUTE(datetime) //分返回符号,正负或0
SQRT(number2) //开平方
```

MySql分页存储过程

MySql测试版本：5.0.41-community-nt

```
DROP PROCEDURE IF EXISTS pr_pager;
CREATE PROCEDURE pr_pager(

    IN    p_table_name       VARCHAR(1024),
    IN    p_fields           VARCHAR(1024),
    IN    p_page_size        INT,
    IN    p_page_now         INT,
    IN    p_order_string     VARCHAR(128),
    IN    p_where_string     VARCHAR(1024),
    OUT   p_out_rows         INT

)
    NOT DETERMINISTIC
    SQL SECURITY DEFINER
    COMMENT '分页存储过程'

BEGIN


    DECLARE m_begin_row INT DEFAULT 0;
    DECLARE m_limit_string CHAR(64);


    SET m_begin_row = (p_page_now - 1) * p_page_size;
```

```
    SET m_limit_string = CONCAT(' LIMIT ', m_begin_row, ', ', p_page_size);

    SET @COUNT_STRING = CONCAT('SELECT COUNT(*) INTO @ROWS_TOTAL FROM ', p_tabl
_name, ' ', p_where_string);
    SET @MAIN_STRING = CONCAT('SELECT ', p_fields, ' FROM ', p_table_name, ' ', p_where_str
ng, ' ', p_order_string,m_limit_string);


    PREPARE count_stmt FROM @COUNT_STRING;
    EXECUTE count_stmt;
    DEALLOCATE PREPARE count_stmt;
    SET p_out_rows = @ROWS_TOTAL;

    PREPARE main_stmt FROM @MAIN_STRING;
    EXECUTE main_stmt;
    DEALLOCATE PREPARE main_stmt;

END;
```

调用

```
mysql> call pr_pager("t","var",3,3,"","",@result);
mysql> call pr_pager("t","var",3,2,"","",@result);
```