目录

冯靖

联系方式
sxyx2008@163.com
http://www.blogjava.net/sxyx2008

MySQL 存储过程

1.1 CREATE PROCEDURE （创建）

```
CREATE PROCEDURE 存储过程名 (参数列表)
   BEGIN
        SQL 语句代码块
   END
```

注意：

由括号包围的参数列必须总是存在。如果没有参数，也该使用一个空参数列()。每个参数默认都是一个 IN 参数。要指定为其它参数，可在参数名之前使用关键词 OUT 或 INOUT

在 mysql 客户端定义存储过程的时候使用 delimiter 命令来把语句定界符从;变为//。

当使用 delimiter 命令时，你应该避免使用反斜杠('\')字符，因为那是 MySQL 的转义字符。

如：

```
mysql> delimiter //
mysql> CREATE PROCEDURE simpleproc (OUT param1 INT)
    -> BEGIN
    ->   SELECT COUNT(*) INTO param1 FROM t;
    -> END
    -> //
Query OK, 0 rows affected (0.00 sec)
```

1.2 ALTER PROCEDURE （修改）

```
ALTER PROCEDURE  存储过程名 SQL 语句代码块
```

这个语句可以被用来改变一个存储程序的特征。

1.3 DROP PROCEDURE （删除）

```
DROP  PROCEDURE  IF  EXISTS 存储过程名
eg:DROP PROCEDURE IF EXISTS proc_employee (proc_employee 存储过程名)
```

这个语句被用来移除一个存储程序。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

1.4 SHOW CREATE PROCEDURE（类似于 SHOW CREATE TABLE，查看一个已存在的存储过程）

```
SHOW CREATE PROCEDURE  存储过程名
```

1.5 SHOW PROCEDURE STATUS （列出所有的存储过程）

```
SHOW PROCEDURE STATUS
```

1.6 CALL 语句（存储过程的调用）

```
CALL 存储过程名(参数列表)
```

CALL 语句调用一个先前用 CREATE PROCEDURE 创建的程序。

CALL 语句可以用声明为 OUT 或的 INOUT 参数的参数给它的调用者传回值。

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

1.7 BEGIN ... END（复合语句）

```
[begin_label:]
BEGIN
    [statement_list]
END
[end_label]
```

存储子程序可以使用 BEGIN ... END 复合语句来包含多个语句。

statement_list 代表一个或多个语句的列表。statement_list 之内每个语句都必须用分号（;）来结尾。

复合语句可以被标记。除非 begin_label 存在,否则 end_label 不能被给出,并且如果二者都存在,他们必须是同样的。

1.8 DECLARE 语句（用来声明局部变量）

DECLARE 语句被用来把不同项目局域到一个子程序：局部变量

DECLARE 仅被用在 BEGIN ... END 复合语句里，并且必须在复合语句的开头，在任何其它语句之前。

1.9 存储程序中的变量

1.1 DECLARE 局部变量

```
DECLARE var_name[,...] type [DEFAULT value]
这个语句被用来声明局部变量。
要给变量提供一个默认值，请包含一个 DEFAULT 子句。
值可以被指定为一个表达式，不需要为一个常数。
如果没有 DEFAULT 子句，初始值为 NULL。
局部变量的作用范围在它被声明的 BEGIN ... END 块内。
它可以被用在嵌套的块中，除了那些用相同名字声明变量的块。
```

1.2 变量 SET 语句

```
SET var_name = expr [, var_name = expr]
在存储程序中的 SET 语句是一般 SET 语句的扩展版本。
被参考变量可能是子程序内声明的变量，或者是全局服务器变量。
```

在存储程序中的 SET 语句作为预先存在的 SET 语法的一部分来实现。这允许 SET a=x, b=y, ...这样的扩展语法。
其中不同的变量类型（局域声明变量及全局和集体变量）可以被混合起来。
这也允许把局部变量和一些只对系统变量有意义的选项合并起来。

1.3 SELECT ... INTO 语句

```
SELECT col_name[,...] INTO var_name[,...] table_expr
```

这个 SELECT 语法把选定的列直接存储到变量。
因此，只有单一的行可以被取回。

```
SELECT id,data INTO x,y FROM test.t1 LIMIT 1;
```

注意，用户变量名在 MySQL 5.1 中是对大小写不敏感的。

重要: SQL 变量名不能和列名一样。如果 SELECT ... INTO 这样的 SQL 语句包含一个对列的参考，并包含一个与列相同名字的局部变量，MySQL 当前把参考解释为一个变量的名字。

1.10 MySQL 存储过程参数类型（in、out、inout）

此小节内容来自：Nonels'Blog

参见地址：http://www.blogjava.net/nonels/archive/2009/04/22/233324.html

MySQL 存储过程参数（in）

MySQL 存储过程"in"参数：跟 C 语言的函数参数的值传递类似， MySQL 存储过程内部可能会修改此参数，但对 in 类型参数的修改，对调用者（caller）来说是不可见的（not visible）。

MySQL 存储过程参数（out）

MySQL 存储过程"out"参数：从存储过程内部传值给调用者。在存储过程内部，该参数初始值为 null，无论调用者是否给存储过程参数设置值

MySQL 存储过程参数（inout）

MySQL 存储过程 inout 参数跟 out 类似，都可以从存储过程内部传值给调用者。不同的是：调用者还可以通过 inout 参数传递值给存储过程。

总结

如果仅仅想把数据传给 MySQL 存储过程，那就使用"in"类型参数；如果仅仅从 MySQL 存储过程返回值，那就使用"out"类型参数；如果需要把数据传给 MySQL 存储过程，还要经过一些计算后再传回给我们，此时，要使用"inout"类型参数。

1.11 例子：

1.1 创建存储过程

带(输出参数)返回值的存储过程：

```
--删除存储过程
DROP PROCEDURE IF EXISTS proc_employee_getCount
--创建存储过程
CREATE PROCEDURE proc_employee_getCount(out n int)
BEGIN
    SELECT COUNT(*) FROM employee ;
END
```

```
--MYSQL 调用存储过程
CALL proc_employee_getCount(@n);
```

带输入参数的存储过程：

```
--删除存储过程
DROP PROCEDURE IF EXISTS proc_employee_findById;
--创建存储过程
CREATE PROCEDURE proc_employee_findById(in n int)
BEGIN
    SELECT * FROM employee where id=n;
END
--定义变量
SET @n=1;
--调用存储过程
CALL proc_employee_findById(@n);
```

操作存储过程时应注意：

1. 删除存储过程时只需要指定存储过程名即可，不带括号；
2. 创建存储过程时，不管该存储过程有无参数，都需要带括号；
3. 在使用 SET 定义变量时应遵循 SET 的语法规则；
   SET @变量名=初始值；
4. 在定义存储过程参数列表时，应注意参数名与数据库中字段名区别开来，否则将出现无法预期的结果

1.12 Java 代码调用存储过程(JDBC)

相关 API：`java.sql.CallableStatement`

使用到 `java.sql.CallableStatement` 接口，该接口专门用来调用存储过程；

该对象的获得依赖于 `java.sql.Connection`;

通过 `Connection` 实例的 `prepareCall()`方法返回 `CallableStatement` 对象

`prepareCall()`内部为一固定写法`{call` 存储过程名`(`参数列表 `1,`参数列表 `2)}`可用`?`占位

`eg: connection.prepareCall("{call proc_employee(?)}");`

存储过程中参数处理：

输入参数：通过 `java.sql.CallableStatement` 实例的 `setXXX()`方法赋值，用法等同于 java.sql.PreparedStatement

输出参数:通过 `java.sql.CallableStatement` 实例的 `registerOutParameter(`参数位置, 参数类型`)`方法赋值,其中参数类型主要使用 `java.sql.Types` 中定义的类型

Java 代码调用带输入参数的存储过程 (根据输入 ID 查询雇员信息)

```
public void executeProcedure()
    {
        try {
            /**
             * callableStatement java.sql.CallableStatement
             * connection java.sql.Connection
             * jdbc调用存储过程原型
             * {call 存储过程名(参数列表1,参数列表2)} 可用?代替
             */
```

```
            callableStatement=connection.prepareCall("{call
proc_employee_findById(?)}");
            callableStatement.setInt(1, 1); //设置输入参数
            resultSet=callableStatement.executeQuery();//执行存储过程
            if(resultSet.next())
            {

    System.out.println(resultSet.getInt(1)+"\t"+resultSet.getString(2));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

Java 代码调用带输出参数的存储过程 (返回数据库中的记录数)

```
public void executeProcedure()
    {
        try {
            /**
             * callableStatement java.sql.CallableStatement
             * connection java.sql.Connection
             * jdbc调用存储过程原型
             * {call 存储过程名(参数列表1,参数列表2)} 可用?代替
             */
            callableStatement=connection.prepareCall("{call
proc_employee_getCount(?)}");
            //设置输出参数
            callableStatement.registerOutParameter(1, Types.INTEGER);
            //执行存储过程
            resultSet=callableStatement.executeQuery();
            if(resultSet.next())
            {
                System.out.println(resultSet.getInt(1));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
```

1.13 声明:

此文档中除 MySQL 存储过程参数类型（in、out、inout）小节来自网上，其余均为本人原创，欢迎大家转载，如有不足，请指教。