# MySQL存储的程序和视图

这一部分讨论存储的程序和视图，它们本质上是存储在数据库服务器上的由一系列SQL语句组成的数据库对象。这些数据库对象可以在以后执行。

存储的程序包括以下对象：

* 存储的例程，即存储过程和函数。使用[CALL](https://dev.mysql.com/doc/refman/5.7/en/call.html)语句调用存储过程 。存储过程没有返回值，但可以修改其参数以供之后的调用者检查。它还可以生成要返回给客户端程序的结果集。函数的使用非常像内置函数。可以在表达式中调用它，并在表达式求值期间返回一个值。（主动调用）
* 触发器。触发器是与表关联的命名数据库对象，当表发生特定事件（例如插入或更新）时，该对象将被激活并执行。（由事件触发）
* 事件。（定期调度执行，类比操作系统的crontab就好理解了）

视图是存储在数据库服务器的查询，当被引用时产生一个结果集。视图被作为虚拟表。

下边介绍这些数据库对象的使用，关于这些数据库对象的定义，在SQL语法那里进行说明。

## 1、定义存储的程序

每个存储的程序都包含一个由SQL语句组成的主体。该语句可能是由分号（;）分隔多个语句组成的复合语句，例如：

CREATE PROCEDURE dorepeat(p1 INT)

BEGIN

SET @x = 0;

REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;

END;

如果使用[**mysql**](https://dev.mysql.com/doc/refman/5.7/en/mysql.html)客户端程序来定义包含分号字符的存储得程序，则会出现问题。默认情况下，[**mysql**](https://dev.mysql.com/doc/refman/5.7/en/mysql.html)本身将分号识别为语句分隔符，因此你必须临时重新定义分隔符，以使[**mysql**](https://dev.mysql.com/doc/refman/5.7/en/mysql.html)将整个存储的程序定义传递到服务器。要重新定义[**mysql**](https://dev.mysql.com/doc/refman/5.7/en/mysql.html)客户端的分隔符，请使用 delimiter命令。以下示例显示如何为dorepeat()重新定义分隔符。将分隔符更改为//，以使整个定义作为单个语句传递到服务器，然后在调用过程之前恢复。这使得过程体内部使用的分隔符可以传递到服务器，而不是由[**mysql**](https://dev.mysql.com/doc/refman/5.7/en/mysql.html) 客户端本身解释。

mysql> delimiter //

mysql> CREATE PROCEDURE dorepeat(p1 INT)

-> BEGIN

-> SET @x = 0;

-> REPEAT SET @x = @x + 1; UNTIL @x > p1 END REPEAT;

-> END

-> //

Query OK, 0 rows affected (0.00 sec)

mysql> delimiter ;

mysql> CALL dorepeat(1000);

Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @x;

+------+

| @x |

+------+

| 1001 |

+------+

1 row in set (0.00 sec)

也可以将分隔符临时定义为其他字符，可以为一个也可以为多个字符。但是注意，不要定义为\，因为\为MySQL的转义字符。如下，因为以下函数内部无;，因此，无需使用delimiter来临时定义分隔符：

mysql> CREATE FUNCTION hello (s CHAR(20))

mysql> RETURNS CHAR(50) DETERMINISTIC

-> RETURN CONCAT('Hello, ',s,'!');

Query OK, 0 rows affected (0.00 sec)

mysql> SELECT hello('world');

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

| hello('world') |

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

| Hello, world! |

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

1 row in set (0.00 sec)

## 2、使用存储的程序（存储过程和函数）

MySQL支持的存储的例程包括存储过程和函数，存储的例程是可以存储在服务器中的一组SQL语句。这些语句被存储在MySQL的系统数据库mysql的表proc中。此表是在MySQL安装过程中创建的。如果要从早期版本升级到MySQL 5.7，请确保更新授权表以确保proc表存在。

存储的例程在某些情况下可能特别有用：

* 多个客户端应用程序以不同的语言编写或在不同的平台上工作，但需要执行相同的数据库操作。
* 当安全至关重要的时候 ，例如，银行所有常见的操作使用存储过程和函数。这提供了一致的、安全的环境，例程可以确保每个操作都被正确记录。在这样的设置中，应用程序和用户将无法直接访问数据库的表，只能执行特定的存储例程，这就防止了用户对数据库表的误操作。对于安全性要求很高的场景下是很有意义的。

存储的例程可以适当提高性能，因为这会减少服务器和客户端之间发送的信息。但是，这会增加数据库服务器的负载，因为更多的工作是在服务器端完成的。如果许多客户机（如Web服务器）仅由一个或几个数据库服务器提供服务，请考虑这一点。

MySQL遵循SQL：2003语法，这也是IBM DB2使用的。MySQL支持这里描述的所有语法，并在适当的情况下给出限制和扩展。

存储例程的二进制日志（binlog）记录按 [第23.7节“存储程序的二进制记录”中所述进行](https://dev.mysql.com/doc/refman/5.7/en/stored-programs-logging.html)。

**2.1 存储例程的语法**

存储例程包括存储过程和函数。使用[CREATE PROCEDURE](https://dev.mysql.com/doc/refman/5.7/en/create-procedure.html)和[CREATE FUNCTION](https://dev.mysql.com/doc/refman/5.7/en/create-function.html" \o "13.1.13 CREATE FUNCTION语法)语句创建（请参见 [第13.1.16节“创建过程和创建函数语法”](https://dev.mysql.com/doc/refman/5.7/en/create-procedure.html)）。使用[CALL](https://dev.mysql.com/doc/refman/5.7/en/call.html" \o "13.2.1 CALL语法)语句调用存储过程（参见 [第13.2.1节“调用语法”](https://dev.mysql.com/doc/refman/5.7/en/call.html)），并且存储过程只能使用输出变量回传值，而没有返回值。函数可以像语句一样调用，就像调用函数名一样，函数也可以返回一个标量值。存储例程定义的正文可以使用复合语句（参见 [第13.6节“复合语句语法”](https://dev.mysql.com/doc/refman/5.7/en/sql-syntax-compound-statements.html)）。

可以使用[DROP PROCEDURE](https://dev.mysql.com/doc/refman/5.7/en/drop-procedure.html" \o "13.1.27 DROP PROCEDURE和DROP FUNCTION语法)和 [DROP FUNCTION](https://dev.mysql.com/doc/refman/5.7/en/drop-function.html)删除存储的例程（请参见 [第13.1.27节“DROP PROCEDURE和DROP FUNCTION语法”](https://dev.mysql.com/doc/refman/5.7/en/drop-procedure.html)），并使用[ALTER PROCEDURE](https://dev.mysql.com/doc/refman/5.7/en/alter-procedure.html" \o "13.1.6 ALTER PROCEDURE语法)和 [ALTER FUNCTION](https://dev.mysql.com/doc/refman/5.7/en/alter-function.html)语句进行更改 （参见 [第13.1.6节“ALTER PROCEDURE语法”](https://dev.mysql.com/doc/refman/5.7/en/alter-procedure.html)）。

存储过程或函数与特定的数据库相关联（即：可以理解为存储过程和函数与表一样，都是数据库对象，属于某一个数据库）。这有几个含义：

* 当调用例程时，隐式执行USE  ***db\_name***（当例程终止时，它将被撤销，理解如下：例如例程proc1属于test数据库，但是此时我们的数据库为testttt，然后在客户端如下调用：CALL test.proc1，那么在存储过程执行时会先隐式执行USE test，proc执行完后再隐式执行USE testttt。）。 存储例程中不允许使用 [USE](https://dev.mysql.com/doc/refman/5.7/en/use.html)语句。
* 可以使用数据库名称限定存储的例程名称。这可以用于引用不在当前数据库中的例程。例如，要调用与数据库test相关联的存储过程 p或函数f，可以使用：CALL test.p()或者test.f()
* 当数据库被删除时，与它相关联的所有存储的例程也被删除。
* 创建存储的例程时，如果不指定数据库，那么例程属于当前数据库。具体如何指定，见SQL语法部分创建存储的例程部分的说明。

函数不能递归。

存储过程可以递归，但默认情况下为禁用。要启用递归，请将 [max\_sp\_recursion\_depth](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_max_sp_recursion_depth)服务器系统变量设置为大于零的值。存储过程递归增加了对线程堆栈空间的需求。如果增加 [max\_sp\_recursion\_depth](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_max_sp_recursion_depth)，可能需要在服务器启动时增加[thread\_stack](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html" \l "sysvar_thread_stack)的值来增加线程堆栈大小 。有关详细信息[，](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html" \o "5.1.5服务器系统变量)请参见[第5.1.5节“服务器系统变量”](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html" \o "5.1.5服务器系统变量)。

MySQL支持一个非常有用的扩展，可以在存储过程中使用常规的SELECT语句（即不使用游标或局部变量）。这样查询的结果集只是直接发送给客户端。多个[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html) 语句生成多个结果集，因此客户端必须使用支持多个结果集的MySQL客户端库。这意味着客户端必须使用至少4.1版本的MySQL客户端库，并且在客户端链接到服务器时应该指定CLIENT\_MULTI\_RESULTS选项。对于C程序，可以使用 [mysql\_real\_connect()](https://dev.mysql.com/doc/refman/5.7/en/mysql-real-connect.html)C API函数完成。请参见[第27.8.7.54节“mysql\_real\_connect（）”](https://dev.mysql.com/doc/refman/5.7/en/mysql-real-connect.html" \o "27.8.7.54 mysql_real_connect（）)和 [第27.8.16节“C API多语句执行支持”](https://dev.mysql.com/doc/refman/5.7/en/c-api-multiple-queries.html)。

**2.2 MySQL存储的例程与权限**

MySQL授权系统也将存储的例程考虑在内：

* [CREATE ROUTINE](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_create-routine)存储的例程的创建权限。
* [ALTER ROUTINE](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_alter-routine)需要该权限来更改或删除存储的例程。如果需要，该权限将自动授予例程的创建者，并且在该例程被删除时从创建者中收回。
* [EXECUTE](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_execute)存储的例程的执行权限。如果需要，此特权将自动授予例程的创建者（并且在该例程被删除时从该创建者中收回）。此外，SQL SECURITY 例程的默认特性是DEFINER，使访问与该例程相关联的数据库的用户能够执行该例程。
* 如果 [automatic\_sp\_privileges](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_automatic_sp_privileges) 系统变量为0， [EXECUTE](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_execute)和 [ALTER ROUTINE](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_alter-routine)权限不会自动授予创建者，并从日常的创造者下降。
* 例程的创建者是执行CREATE语句的帐户 。这可能不同于DEFINER在例程定义中命名的帐户 。

服务器mysql.proc响应于创建，更改或删除存储例程的语句来操作表。不支持服务器将注意到此表的手动操作。

2.3 存储例程的元数据

有关存储例程的元数据可以如下获得：

* 查询数据库INFORMATION\_SCHEMA的[ROUTINES](https://dev.mysql.com/doc/refman/5.7/en/routines-table.html)表 。见 [第24.20节“INFORMATION\_SCHEMA ROUTINES表”](https://dev.mysql.com/doc/refman/5.7/en/routines-table.html)。
* 使用[SHOW CREATE PROCEDURE](https://dev.mysql.com/doc/refman/5.7/en/show-create-procedure.html) 和[SHOW CREATE FUNCTION](https://dev.mysql.com/doc/refman/5.7/en/show-create-function.html) 语句查看例程的定义。请参见 [第13.7.5.9节“SHOW CREATE PROCEDURE语法”](https://dev.mysql.com/doc/refman/5.7/en/show-create-procedure.html)。
* 使用[SHOW PROCEDURE STATUS](https://dev.mysql.com/doc/refman/5.7/en/show-procedure-status.html) 和[SHOW FUNCTION STATUS](https://dev.mysql.com/doc/refman/5.7/en/show-function-status.html) 语句来查看例程的特征。请参见 [第13.7.5.28节“显示程序状态语法”](https://dev.mysql.com/doc/refman/5.7/en/show-procedure-status.html)。

2.4 存储过程，函数，触发器和LAST\_INSERT\_ID（）

作者：许富博

版权所有，文章以学习和交流为主，切勿用于商业用途。

限于本人水平有限，欢迎大家随时指正，联系方式：

[xufubobo@gmail.com](mailto:xufubobo@gmail.com)

[xufubobo@163.com](mailto:xufubobo@163.com)

[1332841493@qq.com](mailto:1332841493@qq.com)