# 概述

用户自定义函数（user-defined function，UDF）是一种对MySQL扩展的途径，其用法与内置函数相同。

# 必要条件

CREATE

/\*[DEFINER = { user | CURRENT\_USER }]\*/

FUNCTION `test`.`a`()

RETURNS TYPE

/\*LANGUAGE SQL

| [NOT] DETERMINISTIC

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

| SQL SECURITY { DEFINER | INVOKER }

| COMMENT 'string'\*/

BEGIN

RETURN ;

END$$

自定义函数的两个必要条件：

1. 参数
2. 返回值

函数可以返回任意类型的值，同样可以接收这些类型的参数。

## 参数

### [NOT] DETERMINISTIC

这个是用于binlog和主从复制等！DETERMINISTIC是确定的，意思就是写入binlog的时候，写入的是一个指定的常量；如unix\_timestamp()获取到的值是1，可能写入binlog的时候，unix\_timestamp()获取到的时间戳却成了3了，这个时候会出现数据不一致问题，所以引入了DETERMINISTIC！这是binlog安全的一种机制！

**一般情况下，NOT DETERMINISTIC不允许使用**，会报如下错误：

Error Code : 1418

This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration and binary logging is enabled (you \*might\* want to use the less safe log\_bin\_trust\_function\_creators variable)

可以从报错内容里面发现，设置log\_bin\_trust\_function\_creators函数就可以使用NOT DETERMINISTIC，但是二进制安全性极差！

### CONTAINS SQL

表示子程序不包含读或写数据的语句。

### NO SQL

表示子程序不包含SQL语句。

### READS SQL DATA

表示子程序包含读数据的语句，但不包含写数据的语句。

### MODIFIES SQL DATA

表示子程序包含写数据的语句。

### SQL SECURITY { DEFINER | INVOKER}

就是执行时的权限检查；如果一个函数定义为访问a表中的记录！这个时候就有权限检查。DEFINER就是以创建者的权限去检查，INVOKER是以执行者的权限去检查！

如果用户u1去调用函数f去访问a的记录(此处的DEFINER定义为u2)！当为DEFINER的时候，mysql会检查u1是否有execute函数f的权限，如果有，继续看定义者u2是否有访问a记录的权限；[u1是执行者，u2是定义者]

同理：当为INVOKER的时候，mysql会检查u1是否有execute函数f的权限，如果有，继续看u1是否有访问a记录的权限

### COMMENT 'string'

注释

## 返回值

RETURNS TYPE：返回类型，就是mysql里面所支持的所有列类型

# 参数

## log\_bin\_trust\_function\_creators

**参考：**

<https://blog.csdn.net/jlq_diligence/article/details/100023236>

<https://www.cnblogs.com/kerrycode/p/7641835.html>

当二进制日志启用后，这个变量就会启用。它控制是否可以信任存储函数创建者，不会创建写入二进制日志引起不安全事件的存储函数。如果设置为0（默认值），用户不得创建或修改存储函数，除非它们具有除CREATE ROUTINE或ALTER ROUTINE特权之外的SUPER权限。设置为0还强制使用DETERMINISTIC特性或READS SQL DATA或NO SQL特性声明函数的限制。 如果变量设置为1，MySQL不会对创建存储函数实施这些限制。 此变量也适用于触发器的创建。

当开启二进制日志后，如果变量log\_bin\_trust\_function\_creators为OFF，那么创建或修改存储函数就会报“ERROR 1418 (HY000): This function has none of DETERMINISTIC, NO SQL, or READS SQL DATA in its declaration and binary logging is enabled (you \*might\* want to use the less safe log\_bin\_trust\_function\_creators variable)”这样的错误。

那么为什么MySQL有这样的限制呢？因为二进制日志的一个重要功能是用于主从复制，而存储函数有可能导致主从的数据不一致。所以当开启二进制日志后，参数log\_bin\_trust\_function\_creators就会生效，限制存储函数的创建、修改、调用。那么此时如何解决这个问题呢？

1、如果数据库没有使用主从复制，那么就可以将参数log\_bin\_trust\_function\_creators设置为1。

mysql> set global log\_bin\_trust\_function\_creators=1;

这个动态设置的方式会在服务重启后失效，所以我们还必须在my.cnf中设置，加上log\_bin\_trust\_function\_creators=1，这样就会永久生效。

2、明确指明函数的类型，如果我们开启了二进制日志，那么我们就必须为我们的function指定一个参数。其中下面几种参数类型里面，只有 DETERMINISTIC，NO SQL和READS SQL DATA被支持。这样一来相当于明确的告知MySQL服务器这个函数不会修改数据。

1、DETERMINISTIC确定的

2、NO SQL没有SQl语句，当然也不会修改数据

3、READS SQL DATA只是读取数据，当然也不会修改数据

4、MODIFIES SQL DATA要修改数据

5、CONTAINS SQL包含了SQL语句

当有mysql本地或远程建立function或procedure时报上面的错误 或者如果开启了二进制日志，但是用户没有supper权限；那么他在创建trigger时会提示设置log\_bin\_trust\_function\_creators=1，经试验是log\_bin\_trust\_function\_creators值为off导致。

**设置：**

set global log\_bin\_trust\_function\_creators=1;

但重启后失效。

永久解决方案：

windows下my.ini[mysqld]加上：

log\_bin\_trust\_function\_creators=1

linux下/etc/my.cnf下my.ini[mysqld]加上：

log\_bin\_trust\_function\_creators=1

保存，重启服务器。

**注意事项：**

1. 主备机都需要设置该参数
2. 如果使用set global设置，则重启后会失效，MySQL8.0已经支持全局变量持久化，使用set persist。

# 操作

首先查看是否已经开启了创建函数的功能：

show variables like ‘%func%’;

如果变量的值为OFF，那么需要开启：

set global log\_bin\_trust\_function\_creators = 1;

## 创建

创建自定义函数：

CREATE FUNCTION function\_name

RETURNS

{STRING|INTEGER|REAL|DECIMAL}

routine\_body

关于函数体routine\_body：

1. 函数体由合法的SQL语句构成；
2. 函数体可以是简单的SELECT或INSERT语句；
3. 函数体如果为复合结构则使用BEGIN…END语句；
4. 复合结构可以包含声明，循环，控制结构。

1、创建不带参数的自定义函数：

2、创建带参数的自定义函数：

3、创建具有复合结构函数体的自定义函数：

## 查看

MySQL存储了存储过程和函数的状态信息，用户可以使用SHOW STATUS语句或SHOW CREATE语句来查看，也可直接从系统的information\_schema数据库中查询。

### SHOW STATUS

SHOW STATUS语句可以查看存储过程和函数的状态，其基本语法结构如下：



这个语句是一个MySQL的扩展，返回子程序的特征，如数据库、名字、类型、创建者及创建和修改日期。如果没有指定样式，那么根据使用的语句，所有存储程序或存储函数的信息都会被列出。其中，PROCEDURE和FUNCTION分别表示查看存储过程和函数；LIKE语句表示匹配存储过程或函数的名称。

### SHOW CREATE

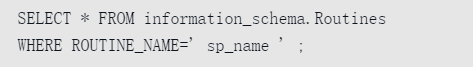
除了SHOW STATUS之外，MySQL还可以使用SHOW CREATE语句查看存储过程和函数的状态。



这个语句是一个MySQL的扩展。类似于SHOW CREATE TABLE，它返回一个可用来重新创建已命名子程序的确切字符串。PROCEDURE和FUNCTION分别表示查看存储过程和函数；sp\_name参数表示匹配存储过程或函数的名称。

### information\_schema

MySQL中存储过程和函数的信息存储在information\_schema数据库下的Routines表中。可以通过查询该表的记录来查询存储过程和函数的信息。其基本语法形式如下：



其中，ROUTINE\_NAME字段中存储的是存储过程和函数的名称；sp\_name参数表示存储过程或函数的名称。

在information\_schema数据库下的Routines表中，存储所有存储过程和函数的定义。使用SELECT语句查询Routines表中的存储过程和函数的定义时，一定要使用ROUTINE\_NAME字段指定存储过程或函数的名称。否则，将查询出所有的存储过程或函数的定义。如果有存储过程和存储函数名称相同，就需要同时指定ROUTINE\_TYPE字段表明查询的是哪种类型的存储程序。

## 调用

在MySQL中，存储函数的使用方法与MySQL内部函数的使用方法是一样的。换言之，用户自己定义的存储函数与MySQL内部函数是一个性质的。区别在于，存储函数是用户自己定义的，而内部函数是MySQL的开发者定义的。

调用自定义函数方法：select 自定义函数; //存储过程使用call调用

## 修改

使用ALTER语句可以修改存储过程或函数的特性，本节将介绍如何使用ALTER语句修改存储过程和函数。



其中，sp\_name参数表示存储过程或函数的名称；characteristic参数指定存储函数的特性，可能的取值有：

●　CONTAINS SQL，表示子程序包含SQL语句，但不包含读或写数据的语句。

●　NO SQL，表示子程序中不包含SQL语句。

●　READS SQL DATA，表示子程序中包含读数据的语句。

●　MODIFIES SQL DATA，表示子程序中包含写数据的语句。

●　SQL SECURITY { DEFINER | INVOKER }，指明谁有权限来执行。

●　DEFINER，表示只有定义者自己才能够执行。

●　INVOKER，表示调用者可以执行。

●　COMMENT 'string'，表示注释信息。

**提示：**

修改存储过程使用ALTER PROCEDURE语句，修改存储函数使用ALTER FUNCTION语句。但是，这两个语句的结构是一样的，语句中的所有参数也是一样的。而且，它们与创建存储过程或函数的语句中的参数也是基本一样的。

## 删除

删除自定义函数：

DROP FUNCTION [IF EXISTS] function\_name