1. 数据定义语句
2. Create database

CREATE {DATABASE | SCHEMA} [IF NOT EXISTS] *db\_name*

[*create\_specification*] ...

-------------------------------------------------------------------

*create\_specification*:

[DEFAULT] CHARACTER SET [=] *charset\_name*

| [DEFAULT] COLLATE [=] *collation\_name*

如果数据库存在并且您没有指定，则会发生错误 IF NOT EXISTS。

***create\_specification***选项指定数据库特征。数据库特征存储在db.opt数据库目录中的文件中。该CHARACTER SET子句指定默认的数据库字符集。该COLLATE子句指定默认的数据库排序规则。

1. Alter database

ALTER {DATABASE | SCHEMA} [*db\_name*]

*alter\_specification* ...

---------------------------------------------------------------------

ALTER {DATABASE | SCHEMA} *db\_name*

UPGRADE DATA DIRECTORY NAME //更新与数据库关联的目录的名称,以实现通过编码将数据库名称映射到数据库目录

----------------------------------------------------------------------

*alter\_specification*:

[DEFAULT] CHARACTER SET [=] *charset\_name //* CHARACTER SET条款的改变默认的数据库字符集。

| [DEFAULT] COLLATE [=] *collation\_name //* COLLATE条款的改变默认的数据库排序规则

更改数据库的整体特征，这些特征存储在db.opt数据库目录中的文件中，要使用[ALTER DATABASE](https://dev.mysql.com/doc/refman/5.7/en/alter-database.html)，您需要[ALTER](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_alter)数据库的 特权。 [ALTER SCHEMA](https://dev.mysql.com/doc/refman/5.7/en/alter-database.html)是一个同义词[ALTER DATABASE](https://dev.mysql.com/doc/refman/5.7/en/alter-database.html)。数据库名称可以从第一个语法中省略，在这种情况下，该语句适用于默认数据库。

1. Drop database

DROP {DATABASE | SCHEMA} [IF EXISTS] db\_name

[DROP DATABASE](https://dev.mysql.com/doc/refman/5.7/en/drop-database.html)删除数据库中的所有表并删除数据库。

在删除数据库时，专门为数据库授予的权限不会自动丢弃。他们必须手动删除。

删除数据库不会删除TEMPORARY在该数据库中创建的任何 表。TEMPORARY当创建它们的会话结束时，表格会自动删除。

[\*Create Table/Alter Table/Drop Table](11.28-MySql对Table的操作.docx)

1. Create Event

CREATE

[DEFINER = { *user* | CURRENT\_USER }]// CURRENT\_USER服务器用于认证当前客户端的MySQL帐户的用户名和主机名组合。此帐户决定您的访问权限

EVENT

[IF NOT EXISTS]

*event\_name //*事件名称不区分大小写s

ON SCHEDULE *schedule*

[ON COMPLETION [NOT] PRESERVE]

[ENABLE | DISABLE | DISABLE ON SLAVE]

[COMMENT '*string*']

DO *event\_body*;

----------------------------------------------------------------------

*schedule*:

AT *timestamp* [+ INTERVAL *interval*] ...

| EVERY *interval*

[STARTS *timestamp* [+ INTERVAL *interval*] ...]

[ENDS *timestamp* [+ INTERVAL *interval*] ...]

-----------------------------------------------------------------------

*interval*:

*quantity* {YEAR | QUARTER | MONTH | DAY | HOUR | MINUTE |

WEEK | SECOND | YEAR\_MONTH | DAY\_HOUR | DAY\_MINUTE |

DAY\_SECOND | HOUR\_MINUTE | HOUR\_SECOND | MINUTE\_SECOND}

该声明创建并安排新的事件。除非事件计划程序已启用，否则该事件将不会运行。

[CREATE EVENT](https://dev.mysql.com/doc/refman/5.7/en/create-event.html)要求[EVENT](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_event)创建事件的模式的 特权。它也可能需要 [SUPER](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_super)权限，具体取决于 DEFINER值

有效[CREATE EVENT](https://dev.mysql.com/doc/refman/5.7/en/create-event.html)声明的最低要求如下：

* 关键字[CREATE EVENT](https://dev.mysql.com/doc/refman/5.7/en/create-event.html)加上一个事件名称，它唯一地标识数据库模式中的事件。
* 一个ON SCHEDULE子句，决定事件执行的时间和频率。
* 一个[DO](https://dev.mysql.com/doc/refman/5.7/en/do.html)条款，其中包含SQL语句由事件执行。
* CREATE EVENT myevent
* ON SCHEDULE AT CURRENT\_TIMESTAMP + INTERVAL 1 HOUR
* DO
* UPDATE myschema.mytable SET mycol = mycol + 1;

//创建一个名为的事件 myevent。此事件在其创建一个小时后执行，即通过运行将myschema.mytable 表格mycol列的值递增1的SQL语句。

1. DEFINER子句指定在事件执行时检查访问特权时要使用的MySQL帐户。如果***user***给出值，它应该指定为MySQL账户 , 默认 值是执行语句的用户
2. 该ON SCHEDULE条款确定***event\_body*** 事件定义的时间，频率和时间重复。本条款采取两种形式之一：

一，AT ***timestamp***用于一次性事件，它指定事件只在日期和时间执行一次，日期和时间 ***timestamp***必须包含日期和时间，或者必须是解析为日期时间值的表达式。您可以使用[DATETIME](https://dev.mysql.com/doc/refman/5.7/en/datetime.html)或者 [TIMESTAMP](https://dev.mysql.com/doc/refman/5.7/en/datetime.html)类型的值来 达到这个目的。如果日期过去，则会发生警告

您可以使用[CURRENT\_TIMESTAMP](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_current-timestamp) 指定当前的日期和时间

AT CURRENT\_TIMESTAMP + INTERVAL 3 WEEK + INTERVAL 2 DAY 相当于“ 三个星期两天后 ”。

二，要定期重复操作，请使用 EVERY子句，可以配合starts ends使用，EVERY 12 HOUR STARTS CURRENT\_TIMESTAMP + INTERVAL 30 MINUTE ENDS CURRENT\_TIMESTAMP + INTERVAL 4 WEEKENDS //从30分钟后开始每12小时执行一次一直持续到4周后结束

1. 如果重复事件不在其调度间隔内终止，则结果可能是多个事件同时执行。如果这是不可取的，你应该建立一个机制来防止同时发生的事件。例如，您可以使用该[GET\_LOCK()](https://dev.mysql.com/doc/refman/5.7/en/miscellaneous-functions.html#function_get-lock)函数，或行或表锁定。
2. Do事件。在事件中使用[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)或 [SHOW](https://dev.mysql.com/doc/refman/5.7/en/show.html)仅仅返回结果集的 语句不起作用; 这些输出不会被发送到MySQL Monitor，也不会被存储到任何地方。但是，您可以使用[SELECT ... INTO](https://dev.mysql.com/doc/refman/5.7/en/select.html)和 [INSERT INTO ... SELECT](https://dev.mysql.com/doc/refman/5.7/en/insert-select.html)存储结果的语句 。
3. 通常情况下，一旦事件已经过期，立即下降。您可以通过指定覆盖此行为ON COMPLETION PRESERVE。
4. 您可以创建一个事件，但使用DISABLE关键字防止它被激活 。或者，您可以使用 ENABLE明确的默认状态，这是活跃的。
5. 您可以使用一个COMMENT条款提供一个事件的评论 。 ***comment***可能是您希望用于描述事件的最多64个字符的任何字符串。注释文本是字符串文字，必须用引号括起来。

delimiter |  
CREATE EVENT e\_daily

ON SCHEDULE

EVERY 1 DAY

COMMENT 'Saves total number of sessions then clears the table each day'

DO

BEGIN

INSERT INTO site\_activity.totals (time, total)

SELECT CURRENT\_TIMESTAMP, COUNT(\*)

FROM site\_activity.sessions;

DELETE FROM site\_activity.sessions;

END

delimiter ;

CREATE EVENT e\_call\_myproc

ON SCHEDULE

AT CURRENT\_TIMESTAMP + INTERVAL 1 DAY

DO CALL myproc(5, 27); //调用存储过程

5.Alter Event

ALTER

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

EVENT *event\_name*

[ON SCHEDULE *schedule*]

[ON COMPLETION [NOT] PRESERVE]

[RENAME TO *new\_event\_name*]

[ENABLE | DISABLE | DISABLE ON SLAVE]

[COMMENT '*string*']

[DO *event\_body*]

更改现有事件的一个或多个特征，而无需删除和重新创建它

在[ALTER EVENT](https://dev.mysql.com/doc/refman/5.7/en/alter-event.html)声明中指定选项仅用于要更改的特征; 省略的选项保持其现有的值。

各项属性与create event相同

事件重命名myevent为 yourevent：

ALTER EVENT myevent

RENAME TO yourevent;

您还可以将事件移至不同的数据库使用 ALTER EVENT ... RENAME TO ...和 ***db\_name.event\_name*** 表示法，如下所示：

ALTER EVENT olddb.myevent

RENAME TO newdb.myevent;

没有RENAME EVENT声明。

Eg：

CREATE EVENT myevent

ON SCHEDULE

EVERY 6 HOUR

COMMENT 'A sample comment.'

DO

UPDATE myschema.mytable SET mycol = mycol + 1;

6. Drop Event

DROP EVENT [IF EXISTS] event\_name

该声明将删除名为的事件 ***event\_name***。该事件立即停止激活，并从服务器上完全删除。

7. Create Function，Create Procedure

CREATE //创建存储过程

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

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

[*characteristic* ...] *routine\_body*

*------------------------------------------------------------------------*

CREATE //创建用户自定义函数

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

FUNCTION *sp\_name* ([*func\_parameter*[,...]])

RETURNS *type*

[*characteristic* ...] *routine\_body*

*proc\_parameter*:

[ IN | OUT | INOUT ] *param\_name* *type*

*func\_parameter*:

*param\_name* *type*

*type*:

*Any valid MySQL data type*

*characteristic*:

COMMENT '*string*'

| LANGUAGE SQL

| [NOT] DETERMINISTIC

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

| SQL SECURITY { DEFINER | INVOKER }

如果一个例程对于相同的输入参数总是产生相同的结果，则该例程被认为是“ 确定性的 ”，否则 被认为是“ 不确定性的 ”。如果在例程定义中既没有 DETERMINISTIC也没有NOT DETERMINISTIC给出，则默认是NOT DETERMINISTIC。要声明一个函数是确定性的，你必须DETERMINISTIC明确指定 。[NOW()](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-functions.html#function_now) 函数（或其同义词）的例程，或者 [RAND()](https://dev.mysql.com/doc/refman/5.7/en/mathematical-functions.html#function_rand)是非确定性的

* CONTAINS SQL表示该例程不包含读取或写入数据的语句。如果没有明确给出这些特征，则这是默认值。这样的语句的例子是SET @x = 1或者DO RELEASE\_LOCK('abc')，执行但是既不读也不写数据。
* NO SQL 表明该例程不包含SQL语句。
* READS SQL DATA指示例程包含读取数据的语句（例如 [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)），但不包含写入数据的语句。
* MODIFIES SQL DATA表示该例程包含可能写入数据的语句（例如 [INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html)或 [DELETE](https://dev.mysql.com/doc/refman/5.7/en/delete.html)）。

所述SQL SECURITY特性可以是 DEFINER或INVOKER以指定的安全上下文; 也就是说，例程是使用例程DEFINER子句中指定的帐户的权限 还是调用它的用户来执行。该帐户必须具有访问与该例程相关联的数据库的权限。默认值是 DEFINER。调用例程的用户必须具有[EXECUTE](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_execute)权限，DEFINER如果例程在定义者安全上下文中执行，必须使用该帐户。

*routine\_body*:

*Valid SQL routine statement* ***routine\_body***由一个有效的SQL例程语句。这可以是一个简单的语句，比如 [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)or [INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html)，或者使用BEGIN和写成的复合语句END。复合语句可以包含声明，循环和其他控制结构语句。

*----------------------------------------------------------------------------*

用于创建存储的函数和用户定义的函数。

该[CREATE FUNCTION](https://dev.mysql.com/doc/refman/5.7/en/create-function.html)语句也被用在MySQL中以支持UDF（用户定义的函数），UDF可以被看作是一个外部存储功能，存储函数与UDF共享其名称空间。

\*\*\*\*\*！！！！要调用存储过程，要使用call语句。

圆括号内的参数列表必须始终存在。如果没有参数，则()应该使用一个空的参数列表 。参数名称不区分大小写。每个参数IN默认是一个参数。要指定其他参数，请使用关键字 OUT或INOUT参数名称前。

指定一个参数为IN，， OUT或INOUT仅对一个有效PROCEDURE。对于一个 FUNCTION参数总是被视为 IN参数。

一个IN参数将一个值传递给一个过程。该过程可能会修改该值，但在过程返回时，调用者不能看到该修改。一个OUT参数将过程中的一个值传递给调用者。它的初始值NULL在过程中，当过程返回时，它的值对调用者是可见的。一个 INOUT参数是由呼叫者初始化，可以由程序进行修改，并且由所述方法制备的任何变化是可见的呼叫者时，过程返回。

对于每个OUT或INOUT 参数，在[CALL](https://dev.mysql.com/doc/refman/5.7/en/call.html)调用该过程的语句中传递用户定义的变量， 以便在过程返回时可以获取其值。如果要调用从另一个存储过程或函数内的过程中，你也可以通过常规的参数或局部常规变量作为IN或 INOUT参数。

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)

mysql> delimiter ;

mysql> CALL simpleproc(@a);

Query OK, 0 rows affected (0.00 sec)

mysql> SELECT @a;

RETURNS条款可以只为一个 FUNCTION，它是强制性的。它指示函数的返回类型，函数体必须包含一个语句

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');

9.Alter Procedure

ALTER PROCEDURE proc\_name [characteristic ...]

*characteristic*:

COMMENT '*string*'

| LANGUAGE SQL

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

| SQL SECURITY { DEFINER | INVOKER }

这个语句可以用来改变存储过程的特性。[ALTER PROCEDURE](https://dev.mysql.com/doc/refman/5.7/en/alter-procedure.html)声明中可能会指定多个更改 。但是，您不能使用此语句更改存储过程的参数或主体; 要进行这样的更改，您必须使用[DROP PROCEDURE](https://dev.mysql.com/doc/refman/5.7/en/drop-procedure.html)和删除并重新创建该过程[CREATE PROCEDURE](https://dev.mysql.com/doc/refman/5.7/en/create-procedure.html)。

10.Drop Procedure Drop Function

DROP {PROCEDURE | FUNCTION} [IF EXISTS] sp\_name

11.Create Index

[CREATE INDEX](https://dev.mysql.com/doc/refman/5.7/en/create-index.html)映射到一个 [ALTER TABLE](https://dev.mysql.com/doc/refman/5.7/en/alter-table.html)语句来创建索引。CREATE INDEX不能用来创建一个PRIMARY KEY; 使用 ALTER TABLE来代替，您可以在表上创建所有索引[CREATE TABLE](https://dev.mysql.com/doc/refman/5.7/en/create-table.html)。主表格决定了数据文件中行的物理布局。 [CREATE INDEX](https://dev.mysql.com/doc/refman/5.7/en/create-index.html)使您能够添加索引到现有的表。

+++在正在创建索引的表上运行SELECT可能会阻塞，因为服务器可能需要使用索引来查找记录; 并且，索引因为正在写入而被锁定。+++

CREATE [UNIQUE|FULLTEXT|SPATIAL] INDEX *index\_name*

[*index\_type*]

ON *tbl\_name* (*index\_col\_name*,...)

[*index\_option*]

[*algorithm\_option* | *lock\_option*] ...

UNIQUE索引创建的约束，使得该指数的所有值必须是不同的。如果尝试使用与现有行匹配的键值添加新行，则会发生错误。对于所有引擎，UNIQUE索引允许NULL可以包含多个列的值NULL。如果您为UNIQUE索引中的列指定前缀值，则列值在前缀中必须是唯一的。

FULLTEXT 索引仅支持 [InnoDB](https://dev.mysql.com/doc/refman/5.7/en/innodb-storage-engine.html)和 [MyISAM](https://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html)表格，并且可以只包括 [CHAR](https://dev.mysql.com/doc/refman/5.7/en/char.html)， [VARCHAR](https://dev.mysql.com/doc/refman/5.7/en/char.html)和 [TEXT](https://dev.mysql.com/doc/refman/5.7/en/blob.html)列。索引总是在整个列上发生; 列前缀索引不受支持，如果指定了任何前缀长度将被忽略。

SPATIAL 空间索引

*index\_col\_name*:

*col\_name* [(*length*)] [ASC | DESC]

ASC或DESC。这些关键字被允许用于指定升序或降序索引值存储的将来扩展。目前，他们被解析，但被忽略; 索引值始终以升序存储。

对于字符串列，可以创建仅使用列值的前导部分 的索引，使用 语法来指定索引前缀长度： ***col\_name***(***length***)

* 前缀可以指定 [CHAR](https://dev.mysql.com/doc/refman/5.7/en/char.html)， [VARCHAR](https://dev.mysql.com/doc/refman/5.7/en/char.html)， [BINARY](https://dev.mysql.com/doc/refman/5.7/en/binary-varbinary.html)，和 [VARBINARY](https://dev.mysql.com/doc/refman/5.7/en/binary-varbinary.html)列索引。
* 前缀*必须*用于指定 [BLOB](https://dev.mysql.com/doc/refman/5.7/en/blob.html)和 [TEXT](https://dev.mysql.com/doc/refman/5.7/en/blob.html)列索引。
* 前缀限制以字节为单位，而在前缀长度[CREATE TABLE](https://dev.mysql.com/doc/refman/5.7/en/create-table.html)， [ALTER TABLE](https://dev.mysql.com/doc/refman/5.7/en/alter-table.html)和 [CREATE INDEX](https://dev.mysql.com/doc/refman/5.7/en/create-index.html)语句被解释为非二进制串类型的字符数（[CHAR](https://dev.mysql.com/doc/refman/5.7/en/char.html)， [VARCHAR](https://dev.mysql.com/doc/refman/5.7/en/char.html)， [TEXT](https://dev.mysql.com/doc/refman/5.7/en/blob.html)二进制串类型（）和字节数[BINARY](https://dev.mysql.com/doc/refman/5.7/en/binary-varbinary.html)， [VARBINARY](https://dev.mysql.com/doc/refman/5.7/en/binary-varbinary.html)， [BLOB](https://dev.mysql.com/doc/refman/5.7/en/blob.html)）。在为使用多字节字符集的非二进制字符串列指定前缀长度时考虑到这一点。
* 对于空间列，无法给出前缀值，如本节后面所述。

此处显示的语句使用该name列的前10个字符（假定 name具有非二进制字符串类型）创建一个索引：

CREATE INDEX part\_of\_name ON customer (name(10));

**Eg：**

CREATE   
TABLE`my\_table` （`field1` varchar（400）CHARACTER SET utf8mb4，  
KEY`idx\_my\_table\_field1`（`field1`（191）） //在varchar上创建一个带前缀的索引

*index\_option*:

KEY\_BLOCK\_SIZE [=] *value*

| *index\_type*

| WITH PARSER *parser\_name*

| COMMENT '*string*'

1. KEY\_BLOCK\_SIZE可选地指定用于索引关键字块的字节大小。该值被视为提示; 如有必要，可以使用不同的尺寸。

2.WITH PARSER ***parser\_name***

该选项只能用于FULLTEXT 索引。如果全文索引和搜索操作需要特殊处理，它将一个解析器插件与索引关联起来

3. COMMENT '***string***' 注释

索引定义可以包含最多1024个字符的可选注释。

*index\_type*:

USING {BTREE | HASH}

***index\_type （***USING ***type\_name）***某些存储引擎允许您在创建索引时指定索引类型。

例如：

CREATE TABLE lookup (id INT) ENGINE = MEMORY;

CREATE INDEX id\_index ON lookup (id) USING BTREE; //使用BTREE类型索引

该***index\_type***条款不能用于FULLTEXT INDEX或SPATIAL INDEX规格。全文索引实现是依赖于存储引擎的。空间索引被实现为R-tree索引。

*algorithm\_option*:

ALGORITHM [=] {DEFAULT|INPLACE|COPY}

*lock\_option*:

LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}

12.Drop Index

DROP INDEX *index\_name* ON *tbl\_name*

[*algorithm\_option* | *lock\_option*] ...

*algorithm\_option*:

ALGORITHM [=] {DEFAULT|INPLACE|COPY}

*lock\_option*:

LOCK [=] {DEFAULT|NONE|SHARED|EXCLUSIVE}

该语句被映射到一个[ALTER TABLE](https://dev.mysql.com/doc/refman/5.7/en/alter-table.html)语句来删除索引。

要删除主键，索引名称始终 PRIMARY是必须指定为带引号的标识符，因为PRIMARY它是保留字：

DROP INDEX `PRIMARY` ON t;

13.Create Logfile Group

CREATE LOGFILE GROUP *logfile\_group*

ADD UNDOFILE '*undo\_file*'

[INITIAL\_SIZE [=] *initial\_size*]

[UNDO\_BUFFER\_SIZE [=] *undo\_buffer\_size*]

[REDO\_BUFFER\_SIZE [=] *redo\_buffer\_size*]

[NODEGROUP [=] *nodegroup\_id*]

[WAIT]

[COMMENT [=] '*string*']

ENGINE [=] *engine\_name*

此语句创建一个名为“ ' ***logfile\_group***的单个 UNDO文件的新日志文件组 ***undo\_file***。一个 [CREATE LOGFILE GROUP](https://dev.mysql.com/doc/refman/5.7/en/create-logfile-group.html)陈述只有一个ADD UNDOFILE条款。

mysql> CREATE LOGFILE GROUP lg1

-> ADD UNDOFILE 'undo.dat' INITIAL\_SIZE = 10M;

Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> SHOW WARNINGS;

1 row in set (0.00 sec)

mysql> CREATE LOGFILE GROUP lg1

-> ADD UNDOFILE 'undo.dat' INITIAL\_SIZE = 10M

-> ENGINE = NDB;

Query OK, 0 rows affected (2.97 sec)

14.Alter Logfile Group

ALTER LOGFILE GROUP lg\_3

ADD UNDOFILE 'undo\_10.dat'

INITIAL\_SIZE=32M

ENGINE=NDBCLUSTER;

15.Drop Logfile Group

DROP LOGFILE GROUP *logfile\_group*

ENGINE [=] *engine\_name*

该语句将删除名为的日志文件组 ***logfile\_group***。日志文件组必须已经存在，否则会导致错误。在删除日志文件组之前，您必须删除使用该日志文件组进行UNDO 日志记录的所有表空间。

16.Create Server

CREATE SERVER *server\_name*

FOREIGN DATA WRAPPER *wrapper\_name*

OPTIONS (*option* [, *option*] ...)

*option*:

{ HOST *character-literal*

| DATABASE *character-literal*

| USER *character-literal*

| PASSWORD *character-literal*

| SOCKET *character-literal*

| OWNER *character-literal*

| PORT *numeric-literal* }

此语句创建用于FEDERATED存储引擎的服务器的定义 。该CREATE SERVER语句servers在mysql 数据库的表中创建一个新行 。这个声明需要 [SUPER](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_super)特权。

本***server\_name*** 应该是唯一的参考服务器。服务器定义在服务器范围内是全局的，不可能将服务器定义限定到特定的数据库。 ***server\_name***最大长度为64个字符（长度超过64个字符的名称被无提示地截断），并且不区分大小写。您可以将该名称指定为带引号的字符串。

该***wrapper\_name*** 只能是mysql，并且可以用单引号引起来。***wrapper\_name***目前还不支持其他值 。

CREATE SERVER s

FOREIGN DATA WRAPPER mysql

OPTIONS (USER 'Remote', HOST '198.51.100.106', DATABASE 'test');

创建到FEDERATED表的连接时，可以使用存储在表中的数据：

CREATE TABLE t (s1 INT) ENGINE=FEDERATED CONNECTION='s';

17.Create Server

ALTER SERVER *server\_name*

OPTIONS (*option* [, *option*] ...)

ALTER SERVER s OPTIONS (USER 'sally');

18.Create Server

DROP SERVER [ IF EXISTS ] server\_name

删除名为的服务器的服务器定义 ***server\_name***。mysql.servers表中相应的行被删除。

19.Create Trigger

这个声明创建一个新的触发器。触发器是一个与表关联的已命名数据库对象，当表发生特定事件时会激活该对象。触发器与名为的表相关联，该表 ***tbl\_name***必须引用永久表。您不能将触发器与TEMPORARY表或视图相关联 。

CREATE

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

TRIGGER *trigger\_name*

*trigger\_time* *trigger\_event*

ON *tbl\_name* FOR EACH ROW

[*trigger\_order*]

*trigger\_body*

*trigger\_time*: { BEFORE | AFTER }

Before与After区别：

before：(insert、update)可以对new进行修改，after不能对new进行修改，两者都不能修改old数据。

对于INSERT语句, 只有NEW是合法的；

对于DELETE语句，只有OLD才合法；

对于UPDATE语句，NEW、OLD可以同时使用。

*trigger\_event*: { INSERT | UPDATE | DELETE }

***trigger\_event***表示激活触发器的操作类型。这些 ***trigger\_event***值是允许的：

* [INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html)：无论何时将新行插入表中，触发器都会激活; 例如，通过[INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html)， [LOAD DATA](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)和 [REPLACE](https://dev.mysql.com/doc/refman/5.7/en/replace.html)语句。
* [UPDATE](https://dev.mysql.com/doc/refman/5.7/en/update.html)：只要行被修改，触发器就会激活; 例如通过 [UPDATE](https://dev.mysql.com/doc/refman/5.7/en/update.html)声明。
* [DELETE](https://dev.mysql.com/doc/refman/5.7/en/delete.html)：只要从表中删除一行，触发器就会激活; 例如通过 [DELETE](https://dev.mysql.com/doc/refman/5.7/en/delete.html)和 [REPLACE](https://dev.mysql.com/doc/refman/5.7/en/replace.html)声明。 [DROP TABLE](https://dev.mysql.com/doc/refman/5.7/en/drop-table.html)并且 [TRUNCATE TABLE](https://dev.mysql.com/doc/refman/5.7/en/truncate-table.html)表中的语句*不*激活这个触发器，因为它们不使用[DELETE](https://dev.mysql.com/doc/refman/5.7/en/delete.html)。删除分区也不会激活 [DELETE](https://dev.mysql.com/doc/refman/5.7/en/delete.html)触发器。
* ***trigger\_event***不代表激活触发器的字面类型的SQL语句，因为它代表了一种类型的表操作。例如， [INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html)触发器不仅会激活[INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html)语句，还会 激活[LOAD DATA](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)语句，因为这两个语句都会将行插入到表中。级联外键操作不激活触发器。

*trigger\_order*: { FOLLOWS | PRECEDES } *other\_trigger\_name*

，您可以有两个BEFORE UPDATE 触发器的表。默认情况下，具有相同触发器事件和操作时间的触发器按照创建的顺序激活。为了影响触发顺序，请指定一个 ***trigger\_order***子句，用于指示 FOLLOWS或者PRECEDES和现有触发器的名称也具有相同的触发事件和操作时间。随着FOLLOWS，新的触发器在现有触发器之后激活。与 PRECEDES，新的触发器在现有触发器之前激活。

***trigger\_body***是触发器激活时要执行的语句。要执行多个语句，请使用 [BEGIN ... END](https://dev.mysql.com/doc/refman/5.7/en/begin-end.html) 复合语句结构。这也使您可以使用存储例程中允许的相同语句

mysql> CREATE TABLE account (acct\_num INT, amount DECIMAL(10,2));

Query OK, 0 rows affected (0.03 sec)

mysql> CREATE TRIGGER ins\_sum BEFORE INSERT ON account

FOR EACH ROW SET @sum = @sum + NEW.amount;

mysql> SET @sum = 0;

mysql> INSERT INTO account VALUES(137,14.98),(141,1937.50),(97,-100.00);

mysql> SELECT @sum AS 'Total amount inserted';

20. Drop Trigger

DROP TRIGGER [IF EXISTS] [schema\_name.]trigger\_name

如果删除表格，也会删除表格的触发器。

21. Create View/Alter View

创建一个新的视图，或者在OR REPLACE给出该子句的情况下替换现有的视图。

一个视图属于一个数据库。默认情况下，默认数据库中会创建一个新的视图。要在给定数据库中显式创建视图，请使用***db\_name.view\_name***语法来限定视图名称与数据库名称：CREATE/Alter VIEW test.v AS SELECT \* FROM t;

CREATE

[OR REPLACE]

[ALGORITHM = {UNDEFINED | MERGE | TEMPTABLE}]

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

[SQL SECURITY { DEFINER | INVOKER }]

VIEW *view\_name* [(*column\_list*)]

AS *select\_statement*

[WITH [CASCADED | LOCAL] CHECK OPTION]

***select\_statement***是一个 [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)声明，提供了视图的定义。（从视图中选择，实际上使用[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html) 语句。）***select\_statement***可以从基表或其他视图中进行选择。

ALGORITHM子句影响MySQL如何处理视图。在DEFINER与 SQL SECURITY子句指定的安全上下文在查看调用时检查访问权限时使用。

***column\_list***子句指定 为逗号分隔标识符列表。名称数量 ***column\_list***必须与[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)语句检索的列数相同 。

一个视图可以从多种[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)语句中创建 。它可以引用基表或其他视图。它可以使用连接 [UNION](https://dev.mysql.com/doc/refman/5.7/en/union.html)，和子查询。该 [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)甚至不需要引用任何表：

CREATE VIEW v\_today (today) AS SELECT CURRENT\_DATE;

视图定义受以下限制：

* 该[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)语句不能引用系统变量或用户定义的变量。
* 在存储的程序中， [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)语句不能引用程序参数或局部变量。
* 该[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)语句不能引用准备好的语句参数。
* 定义中提到的任何表格或视图都必须存在。如果在创建视图之后删除定义引用的表或视图，则使用视图会导致错误。要检查这种问题的视图定义，请使用该[CHECK TABLE](https://dev.mysql.com/doc/refman/5.7/en/check-table.html)语句。
* 定义不能引用一个TEMPORARY 表，并且你不能创建一个TEMPORARY 视图。
* 您无法将触发器与视图相关联。
* [SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)语句中的 列名称的别名 将根据64个字符的最大列长度（不是最大别名长度为256个字符）进行检查。

ORDER BY在视图定义中是允许的，但如果使用具有自己的语句从视图中进行选择，它将被忽略ORDER BY

21. Drop View

DROP VIEW [IF EXISTS]

*view\_name* [, *view\_name*] ...

[RESTRICT | CASCADE]

2.综合例子

1.创建一个判断一个表格有没有指定的索引，如果没有则添加一个索引的存储过程，通过call直接调用：

DELIMITER $$  
DROP PROCEDURE IF EXISTS `create\_index\_if\_not\_exists`$$  
  
CREATE DEFINER=`user`@`%` PROCEDURE `create\_index\_if\_not\_exists`(table\_name\_vc varchar(50), index\_name\_vc varchar(50), field\_list\_vc varchar(200))  
SQL SECURITY INVOKER  
BEGIN  
  
set @Index\_cnt = (  
select count(1) cnt  
FROM INFORMATION\_SCHEMA.STATISTICS  
WHERE table\_name = table\_name\_vc  
and index\_name = index\_name\_vc  
);  
  
IF ifnull(@Index\_cnt,0) = 0 THEN set @index\_sql = concat('Alter table ',table\_name\_vc,' ADD INDEX ',index\_name\_vc,'(',field\_list\_vc,');');  
  
PREPARE stmt FROM @index\_sql;  
EXECUTE stmt;  
  
DEALLOCATE PREPARE stmt;  
  
END IF;  
  
END$$  
DELIMITER ;