# MySQL-SQL语法

这里介绍MySQL支持的语句和语法。

## 1、数据定义语言

**（1）ALTER DATABASE语法**

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

*alter\_specification*:

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

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

ALTER DATABASE用于更改数据库的整体特性。要使用ALTER DATABASE，你需要ALTER DATABASE的 权限。 ALTER SCHEMA是ALTER DATABASE的同义词。数据库名称也可以省略，在这种情况下，该语句修改的为默认数据库。

CHARACTER SET子句更改数据库的默认字符集。COLLATE子句更改数据库的默认排序规则。可以分别使用SHOW CHARACTER SET和SHOW COLLATION 查看可用的字符集和排序规则。

**（2）ALTER EVENT语法**

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 '*comment*']

[DO *event\_body*]

该[ALTER EVENT](https://dev.mysql.com/doc/refman/5.7/en/alter-event.html)语句更改现有事件的一个或多个特征，而不需要删除并重新创建它。的语法的每个的 DEFINER，ON SCHEDULE， ON COMPLETION，COMMENT， ENABLE/ DISABLE，和 [DO](https://dev.mysql.com/doc/refman/5.7/en/do.html)条款是完全一样的当用于如[CREATE EVENT](https://dev.mysql.com/doc/refman/5.7/en/create-event.html)。（见[第13.1.12节“创建事件语法”](https://dev.mysql.com/doc/refman/5.7/en/create-event.html)。）

任何用户都可以更改在该用户具有该[EVENT](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_event)权限的数据库上定义的事件。当用户执行成功[ALTER EVENT](https://dev.mysql.com/doc/refman/5.7/en/alter-event.html)语句时，该用户成为受影响事件的定义者。

在以下每个示例中，假定命名的事件 myevent定义如下：

CREATE EVENT myevent

ON SCHEDULE

EVERY 6 HOUR

COMMENT 'A sample comment.'

DO

UPDATE myschema.mytable SET mycol = mycol + 1;

以下声明myevent从每六小时一次更改一次计划， 每十二小时一次，从运行语句开始的四个小时开始：

ALTER EVENT myevent

ON SCHEDULE

EVERY 12 HOUR

STARTS CURRENT\_TIMESTAMP + INTERVAL 4 HOUR;

可以在单个语句中更改事件的多个特征。此示例将执行的SQL语句更改为myevent从中删除所有记录 的SQL语句mytable; 它还会更改事件的计划，以便在此[ALTER EVENT](https://dev.mysql.com/doc/refman/5.7/en/alter-event.html)语句运行后一天执行一次 。

ALTER EVENT myevent

ON SCHEDULE

AT CURRENT\_TIMESTAMP + INTERVAL 1 DAY

DO

TRUNCATE TABLE myschema.mytable;

在[ALTER EVENT](https://dev.mysql.com/doc/refman/5.7/en/alter-event.html)声明中指定仅用于要更改的特征的选项; 省略选项保留其现有值。这包括任何默认值，[CREATE EVENT](https://dev.mysql.com/doc/refman/5.7/en/create-event.html)例如ENABLE。

要禁用myevent，请使用以下 [ALTER EVENT](https://dev.mysql.com/doc/refman/5.7/en/alter-event.html)语句：

ALTER EVENT myevent DISABLE;

该ON SCHEDULE子句可以使用涉及内置的MySQL函数和用户变量的表达式来获取它包含的任何一个***timestamp***或多个***interval***值。您不能在这些表达式中使用存储的例程或用户定义的函数，并且不能使用任何表引用; 但是，您可以使用SELECT FROM DUAL。对于这两个[ALTER EVENT](https://dev.mysql.com/doc/refman/5.7/en/alter-event.html)和 [CREATE EVENT](https://dev.mysql.com/doc/refman/5.7/en/create-event.html)语句都是如此 。在这种情况下，对存储的例程，用户定义的函数和表的引用是特别不允许的，并且会出现错误（参见错误＃22830）。

虽然 在其子句[ALTER EVENT](https://dev.mysql.com/doc/refman/5.7/en/alter-event.html)中包含另一个[ALTER EVENT](https://dev.mysql.com/doc/refman/5.7/en/alter-event.html)语句的[DO](https://dev.mysql.com/doc/refman/5.7/en/do.html)语句似乎成功，但是当服务器尝试执行生成的计划事件时，执行失败并显示错误。

要重命名事件，请使用[ALTER EVENT](https://dev.mysql.com/doc/refman/5.7/en/alter-event.html)语句的RENAME TO子句。此语句将事件重命名myevent为 yourevent：

ALTER EVENT myevent RENAME TO yourevent;

您还可以使用ALTER EVENT ... RENAME TO ...和 ***db\_name.event\_name*** 符号将事件移动到不同的数据库 ，如下所示：

ALTER EVENT olddb.myevent RENAME TO newdb.myevent;

要执行以前的语句，执行它的用户必须具有对数据库和数据库的[EVENT](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_event)权限 。 olddbnewdb

**注意**

没有RENAME EVENT声明

该值DISABLE ON SLAVE用于复制从站，而不是ENABLE或 DISABLE指示在主站上创建并复制到从站的事件，但不在从站上执行。通常，DISABLE ON SLAVE根据需要自动设置; 但是，在某些情况下，您可能需要或需要手动更改它。有关详细信息[，](https://dev.mysql.com/doc/refman/5.7/en/replication-features-invoked.html)请参见[第16.4.1.12节“复制调用的功能”](https://dev.mysql.com/doc/refman/5.7/en/replication-features-invoked.html)。

**（3）ALTER FUNCTION语法**

ALTER FUNCTION *func\_name* [*characteristic* ...]

*characteristic*:

COMMENT '*string*'

| LANGUAGE SQL

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

| SQL SECURITY { DEFINER | INVOKER }

此语句可用于更改存储函数的特性。在一个[ALTER FUNCTION](https://dev.mysql.com/doc/refman/5.7/en/alter-function.html)语句中可以指定多个更改 。但是，您不能使用此语句更改存储函数的参数或正文; 要进行这样的更改，您必须使用[DROP FUNCTION](https://dev.mysql.com/doc/refman/5.7/en/drop-function.html)和删除并重新创建该功能[CREATE FUNCTION](https://dev.mysql.com/doc/refman/5.7/en/create-function.html)。您必须具有[ALTER ROUTINE](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_alter-routine) 该功能的权限。（该权限被自动授予函数创建者。）如果启用了二进制日志记录，则[ALTER FUNCTION](https://dev.mysql.com/doc/refman/5.7/en/alter-function.html) 语句也可能需要 [SUPER](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_super)权限，如 [第23.7节“存储程序的二进制日志记录”中所述](https://dev.mysql.com/doc/refman/5.7/en/stored-programs-logging.html)。

**（4）ALTER INSTANCE语法**

ALTER INSTANCE ROTATE INNODB MASTER KEY

ALTER INSTANCE在MySQL 5.7.11中引入，定义了适用于MySQL服务器实例的操作。

该ALTER INSTANCE ROTATE INNODB MASTER KEY 语句用于旋转用于InnoDB表空间加密的主加密密钥 。必须加载一个keyring插件来使用这个语句。有关密钥环插件的信息，请参见[第6.5.4节“MySQL密钥环”](https://dev.mysql.com/doc/refman/5.7/en/keyring.html)。关键旋转需要[SUPER](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_super)特权。

ALTER INSTANCE ROTATE INNODB MASTER KEY 支持并发DML。但是，它不能与[CREATE TABLE ... ENCRYPTION](https://dev.mysql.com/doc/refman/5.7/en/create-table.html)或并行[ALTER TABLE ... ENCRYPTION](https://dev.mysql.com/doc/refman/5.7/en/alter-table.html)运行，并且采取锁来防止并发执行这些语句可能引起的冲突。如果其中一个冲突的语句正在运行，则必须先完成另一个语句，然后再继续执行。

ALTER INSTANCE 操作将写入二进制日志，以便它们可以在复制的服务器上执行。

有关其他ALTER INSTANCE ROTATE INNODB MASTER KEY使用信息，请参见 [第14.7.10节“InnoDB表空间加密”](https://dev.mysql.com/doc/refman/5.7/en/innodb-tablespace-encryption.html)。有关密钥环插件的信息，请参见[第6.5.4节“MySQL密钥环”](https://dev.mysql.com/doc/refman/5.7/en/keyring.html)。

（5）

## 2、数据操纵语言

LOAD DATA INFILE语法

LOAD DATA [LOW\_PRIORITY | CONCURRENT] [LOCAL] INFILE '*file\_name*'

[REPLACE | IGNORE]

INTO TABLE *tbl\_name*

[PARTITION (*partition\_name* [, *partition\_name*] ...)]

[CHARACTER SET *charset\_name*]

[{FIELDS | COLUMNS}

[TERMINATED BY '*string*']

[[OPTIONALLY] ENCLOSED BY '*char*']

[ESCAPED BY '*char*']

]

[LINES

[STARTING BY '*string*']

[TERMINATED BY '*string*']

]

[IGNORE *number* {LINES | ROWS}]

[(*col\_name\_or\_user\_var*

[, *col\_name\_or\_user\_var*] ...)]

[SET *col\_name*={*expr* | DEFAULT},

[, *col\_name*={*expr* | DEFAULT}] ...]

LOAD DATA INFILE用于将文本文件的数据导入到一张已经建好的表中。与SELECT ......INTO OUTFILE相对。这两个语句中的FIELDS和 LINES子句的语法是相同的。两个子句都是可选的，但是如果同时指定了两个子句，则FIELDS必须放在LINES之前。也可以使用[**mysqlimport**](https://dev.mysql.com/doc/refman/5.7/en/mysqlimport.html)实用程序load数据文件 ，它通过向服务器发送[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)语句来进行操作 。 [--local](https://dev.mysql.com/doc/refman/5.7/en/mysqlimport.html#option_mysqlimport_local)选项使[**mysqlimport**](https://dev.mysql.com/doc/refman/5.7/en/mysqlimport.html)从客户端主机读取数据文件。如果客户端和服务器支持压缩协议，可以指定[--compress](https://dev.mysql.com/doc/refman/5.7/en/mysqlimport.html#option_mysqlimport_compress)选项以在慢速网络上获得更好的性能。即：mysqlimport可以用于远程导入。文件名必须为字符串。[character\_set\_filesystem](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_character_set_filesystem)系统变量控制文件名的解释。该语句也支持分区，分区可以指定多个，当使用此选项时，如果文件中的任何行无法插入到列表中指定的任何分区或子分区中，则该语句将失败，显示错误，并找到与给定分区集不匹配的行。有关更多信息和示例，请参见 [第22.5节“分区选择”](https://dev.mysql.com/doc/refman/5.7/en/partitioning-selection.html)。

**以下未看：**

对于使用采用表锁的存储引擎的分区表，例如[MyISAM](https://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html)，LOAD DATA不能修剪任何分区锁。这不适用于使用采用行级锁定的存储引擎的表，例如[InnoDB](https://dev.mysql.com/doc/refman/5.7/en/innodb-storage-engine.html)。有关详细信息，请参见 [第22.6.4节“分区和锁定”](https://dev.mysql.com/doc/refman/5.7/en/partitioning-limitations-locking.html)。

服务器使用[character\_set\_database](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_character_set_database)系统变量指定的字符集 来解释文件中的信息。 [SET NAMES](https://dev.mysql.com/doc/refman/5.7/en/set-names.html)而设置[character\_set\_client](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_character_set_client)不影响输入的解释。如果输入文件的内容使用与默认值不同的字符集，则通常最好使用该CHARACTER SET子句指定文件的字符集 。一个字符集 binary指定“ 不转换”。”

[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)将文件中的所有字段解释为具有相同的字符集，而不管字段值加载到哪些列的数据类型。为了正确解释文件内容，您必须确保使用正确的字符集进行写入。例如，如果您使用[**mysqldump -T**](https://dev.mysql.com/doc/refman/5.7/en/mysqldump.html)或通过[SELECT ... INTO OUTFILE](https://dev.mysql.com/doc/refman/5.7/en/select-into.html)在[**mysql中**](https://dev.mysql.com/doc/refman/5.7/en/mysql.html)发出 语句编写数据文件 ，请确保使用--default-character-set选项，以便将输出写入要加载文件时要使用的字符集[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)。

**注意**

这是不可能的加载使用数据文件 ucs2，utf16， utf16le，或者utf32 字符集。

如果使用LOW\_PRIORITY，[LOAD DATA](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)语句的执行将 延迟，直到其他客户端从表中读取为止。这会影响只使用表级锁只存储引擎（例如 MyISAM，MEMORY和 MERGE）。

如果您指定CONCURRENT一个MyISAM满足并发插入条件的 表（即中间没有空闲块），则其他线程可以在[LOAD DATA](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)执行时从表中检索数据 。[LOAD DATA](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)即使没有其他线程同时使用该表，此选项也会影响一点的性能。

使用基于行的复制，CONCURRENT无论MySQL版本如何，都将被复制。使用基于语句的复制CONCURRENT不会在MySQL 5.5.1之前复制（参见错误＃34628）。有关更多信息，请参见 [第16.4.1.18节“复制和加载数据INFILE”](https://dev.mysql.com/doc/refman/5.7/en/replication-features-load-data.html)。

该LOCAL关键字影响的文件和错误处理预期的位置，如下文所述。 LOCAL只有当您的服务器和客户端都配置为允许它时才可以工作。例如，如果 在系统变量禁用的情况下启动了 [**mysqld**](https://dev.mysql.com/doc/refman/5.7/en/mysqld.html)[local\_infile](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_local_infile)，LOCAL则不起作用。请参见 [第6.1.6节“LOAD DATA LOCAL的安全问题”](https://dev.mysql.com/doc/refman/5.7/en/load-data-local.html)。

LOCAL关键字影响预期找到文件的位置：

* 如果LOCAL指定了该文件，客户端程序将在客户端主机上读取该文件，并将其发送到服务器。该文件可以作为完整路径名称来指定其确切的位置。如果作为相对路径名称指定，则相对于启动客户端程序的目录来解释该名称。

使用LOCAL时 [LOAD DATA](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)，在服务器的临时目录中创建文件的副本。这是 *不是*由价值决定的目录[tmpdir](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_tmpdir)或[slave\_load\_tmpdir](https://dev.mysql.com/doc/refman/5.7/en/replication-options-slave.html#sysvar_slave_load_tmpdir)，而是操作系统的临时目录，而不是MySQL服务器的配置。（通常，系统临时目录/tmp位于Linux系统和C:\WINDOWS\TEMPWindows上。）缺少足够的空间用于此目录中的副本可能会导致该[LOAD DATA LOCAL](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)语句失败。

* 如果LOCAL未指定，该文件必须位于服务器主机上，并由服务器直接读取。服务器使用以下规则来定位文件：
  + 如果文件名是绝对路径名，服务器将按照给定的方式使用它。
  + 如果文件名是具有一个或多个主要组件的相对路径名，则服务器将相对于服务器的数据目录搜索该文件。
  + 如果没有提供没有引导组件的文件名，服务器将在默认数据库的数据库目录中查找该文件。

在非LOCAL情况下，这些规则意味着./myfile.txt从服务器的数据目录myfile.txt读取名为的文件，而从默认数据库的数据库目录读取名为的文件 。例如，如果 db1是默认数据库，则以下 [LOAD DATA](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)语句data.txt从数据库目录中 读取该文件 db1，即使该语句显式地将该文件加载到db2数据库中的表中：

LOAD DATA INFILE 'data.txt' INTO TABLE db2.my\_table;

非LOCAL负载操作读取位于服务器上的文本文件。出于安全考虑，此类操作要求您有[FILE](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_file) 权限。请参见[第6.2.1节“MySQL提供的特权”](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html)。此外，非LOCAL负载操作需要 [secure\_file\_priv](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_secure_file_priv)系统变量设置。如果变量值是非空目录名称，则要加载的文件必须位于该目录中。如果变量值为空（这是不安全的），则该文件只能由服务器读取。

使用LOCAL比让服务器直接访问文件要慢一些，因为文件的内容必须通过客户端的连接发送到服务器。另一方面，您不需要[FILE](https://dev.mysql.com/doc/refman/5.7/en/privileges-provided.html#priv_file)加载本地文件的 权限。

LOCAL 也会影响错误处理：

* 随着[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)数据解释和重复键错误终止操作。
* 随着[LOAD DATA LOCAL INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)数据解释和重复键错误成为警告，并且操作继续，因为服务器无法在操作中间停止传输文件。对于重复键错误，这与IGNORE指定的相同。 IGNORE将在本节后面进一步解释。

该REPLACE和IGNORE 关键字控制对唯一键值重复现有行的输入行的处理：

* 如果指定REPLACE，则输入行替换现有行。换句话说，具有与主键或唯一索引相同的值作为现有行的行。请参见 [第13.2.8节“REPLACE语法”](https://dev.mysql.com/doc/refman/5.7/en/replace.html)。
* 如果指定IGNORE，则将重复在唯一键值上重复现有行的行。有关详细信息，请参阅[IGNORE关键字和严格SQL模式的比较](https://dev.mysql.com/doc/refman/5.7/en/sql-mode.html#ignore-strict-comparison)。
* 如果不指定任一选项，则行为取决于是否LOCAL指定关键字。没有LOCAL，当找到重复的键值时，会发生错误，并忽略文本文件的其余部分。使用LOCAL，默认行为与IGNORE指定时相同; 这是因为服务器无法在操作中间停止传输文件。

要在加载操作期间忽略外键约束，请SET foreign\_key\_checks = 0在执行前发出语句[LOAD DATA](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)。

如果[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)在空MyISAM表上使用，则所有非唯一索引将在单独的批处理中创建（至于 [REPAIR TABLE](https://dev.mysql.com/doc/refman/5.7/en/repair-table.html)）。通常，[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)当您有很多索引时，这会 更快。在某些极端情况下，您可以ALTER TABLE ... DISABLE KEYS 在将文件加载到表中之前将其关闭，并在加载文件ALTER TABLE ... ENABLE KEYS后重新创建索引，从而更快地创建索引。请参见[第8.2.4.1节“优化INSERT语句”](https://dev.mysql.com/doc/refman/5.7/en/insert-optimization.html)。

**以下都看了：**

对于[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)和 [SELECT ... INTO OUTFILE](https://dev.mysql.com/doc/refman/5.7/en/select-into.html)语句，FIELDS和LINES子句的语法是一样的。两个子句都是可选的，但是如果同时指定了两个子句，则FIELDS 必须在LINES之前。

如果指定了FIELDS子句，则子句FIELDS的每个子句（TERMINATED BY， [OPTIONALLY] ENCLOSED BY和ESCAPED BY）也是可选的，但必须至少指定一个子句。

如果指定了no FIELDS或 LINES子句，那么将和以下默认值一样：

FIELDS TERMINATED BY '\t' ENCLOSED BY '' ESCAPED BY '\\'

LINES TERMINATED BY '\n' STARTING BY ''

（反斜杠是MySQL SQL语句中字符串中的转义字符，因此要指定反斜杠，必须使用两个反斜杠。转义序列'\t'和'\n'分别指定制表符和换行符。）

换句话说，默认情况下 [LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)在读取输入时按照如下条件解析：

* 行之间以换行符分割。
* 不要跳过任何行前缀。
* 行内字段之间的分隔符为\t。
* 字段值不会包含在任何引号内。
* 反斜杠默认为转义字符。例如， \t，\n，和 \\分别表示制表符，换行，反斜线。有关转义序列的完整列表，请参阅以后对FIELDS ESCAPED BY的讨论。

相反，默认情况下 [SELECT ... INTO OUTFILE](https://dev.mysql.com/doc/refman/5.7/en/select-into.html)在写入输出时如下进行：

* 行内字段之间的分隔符为\t。
* 字段值不会包含在任何引号内。
* 使用\作为转义字符来表示字段内出现的字表符、换行符、以及\本身等转义字符序列。
* 在行尾加上换行符。

**注意**

如果在Windows系统上生成了文本文件，则可能必须使用LINES TERMINATED BY '\r\n' 来正确读取文件，因为Windows程序通常使用两个字符作为行终止符。一些程序，如 **写字板**，可能会在写入文件时用\r作为行终止符。要读这些文件，请使用 LINES TERMINATED BY '\r'。

如果所有行都有一个想要忽略的公用前缀，则可以使用LINES STARTING BY '***prefix\_string***'子句跳过前缀*以及其前的任何内容*。如果某一行不包含前缀，则跳过整行。假设发出以下语句：

LOAD DATA INFILE '/tmp/test.txt' INTO TABLE test

FIELDS TERMINATED BY ',' LINES STARTING BY 'xxx';

如果数据文件如下所示：

xxx"abc",1

something xxx"def",2

"ghi",3

将生成两行数据：("abc",1)和 ("def",2)。文件中的第三行将被跳过，因为它不包含前缀。

选项IGNORE ***number*** LINES可用于忽略文件开头的行。例如，可以使用IGNORE 1 LINES跳过包含列名称的初始标题行：

LOAD DATA INFILE '/tmp/test.txt' INTO TABLE test IGNORE 1 LINES;

当你先使用[SELECT ... INTO OUTFILE](https://dev.mysql.com/doc/refman/5.7/en/select-into.html)将数据库中的数据写入一个文件，然后使用 [LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)读取文件到数据库，这两个语句的field-和line-handling选项必须匹配。否则， [LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)将无法正确解释文件的内容。假设你使用 [SELECT ... INTO OUTFILE](https://dev.mysql.com/doc/refman/5.7/en/select-into.html)字段之间以逗号分隔写入文件：

SELECT \* INTO OUTFILE 'data.txt'

FIELDS TERMINATED BY ','

FROM table2;

要读回逗号分隔的文件，正确的语句是：

LOAD DATA INFILE 'data.txt' INTO TABLE table2

FIELDS TERMINATED BY ',';

如果你尝试使用以下语句读入文件，则它将无法正常工作，因为它会指示 [LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)在字段之间查找字表符：

LOAD DATA INFILE 'data.txt' INTO TABLE table2

FIELDS TERMINATED BY '\t';

可能的结果是，每个输入行将被解释为单个字段。

[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html" \o "13.2.6 LOAD DATA INFILE语法)可用于读取从外部来源获取的文件。例如，许多程序导出的数据格式为：字段之间逗号分隔，字段的值由引号包起来，，并带有一个列名的初始行。如果这样的一个文件中的行由回车符/换行对终止，则以下语句说明了加载该文件时需要使用的选项：

LOAD DATA INFILE 'data.txt' INTO TABLE *tbl\_name*

FIELDS TERMINATED BY ',' ENCLOSED BY '"'

LINES TERMINATED BY '\r\n'

IGNORE 1 LINES;

如果输入值不一定包含在引号内，请在ENCLOSED BY关键字之前使用OPTIONALLY。

任何字段或行处理选项都可以指定一个空字符串（''）。如果不是空的，FIELDS [OPTIONALLY] ENCLOSED BY和FIELDS ESCAPED BY必须是一个单个字符（不一定是一个字节）。FIELDS TERMINATED BY，LINES STARTING BY和LINES TERMINATED BY可以是多个字符。例如，要导出通过回车/换行符对终止的行，或者读取包含这些行的文件，请指定一个LINES TERMINATED BY '\r\n'子句。

要读取包含由包含的行分隔的笑话的文件%%，可以这样做

CREATE TABLE jokes

(a INT NOT NULL AUTO\_INCREMENT PRIMARY KEY,

joke TEXT NOT NULL);

LOAD DATA INFILE '/tmp/jokes.txt' INTO TABLE jokes

FIELDS TERMINATED BY ''

LINES TERMINATED BY '\n%%\n' (joke);

FIELDS [OPTIONALLY] ENCLOSED BY控制字段的引用。对于output（[SELECT ... INTO OUTFILE](https://dev.mysql.com/doc/refman/5.7/en/select-into.html)），如果省略该字OPTIONALLY，则所有字段都由ENCLOSED BY字符括起 。此处显示了此类输出的示例（使用逗号作为字段分隔符）：

"1","a string","100.20"

"2","a string containing a , comma","102.20"

"3","a string containing a \" quote","102.20"

"4","a string containing a \", quote and comma","102.20"

如果指定OPTIONALLY的 ENCLOSED BY字符仅用于从具有字符串数据类型（如柱围值 [CHAR](https://dev.mysql.com/doc/refman/5.7/en/char.html)， [BINARY](https://dev.mysql.com/doc/refman/5.7/en/binary-varbinary.html)， [TEXT](https://dev.mysql.com/doc/refman/5.7/en/blob.html)，或 [ENUM](https://dev.mysql.com/doc/refman/5.7/en/enum.html)）：

1,"a string",100.20

2,"a string containing a , comma",102.20

3,"a string containing a \" quote",102.20

4,"a string containing a \", quote and comma",102.20

ENCLOSED BY字段值中字符的 出现通过将字符前缀到ESCAPED BY字符进行转义 。另外，如果指定一个空ESCAPED BY值，可能会无意中生成无法正确读取的输出 [LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)。例如，如果转义字符为空，则刚刚显示的上述输出将显示如下。请注意，第四行中的第二个字段包含引号后面的逗号，（错误地）似乎终止字段：

1,"a string",100.20

2,"a string containing a , comma",102.20

3,"a string containing a " quote",102.20

4,"a string containing a ", quote and comma",102.20

对于输入，ENCLOSED BY字符（如果存在）从字段值的末尾剥离。（无论是否OPTIONALLY指定，都是正确的; OPTIONALLY对输入解释没有影响）。ENCLOSED BY 字符前面的ESCAPED BY字符的出现被解释为当前字段值的一部分。

如果字段以ENCLOSED BY 字符开头，则只有在字段或行TERMINATED BY序列后面，该字符的实例才被识别为终止字段值 。为了避免歧义，ENCLOSED BY字段值中字符的出现可以加倍，并被解释为字符的单个实例。例如，如果ENCLOSED BY '"'指定，则引用引号如下所示处理：

"The ""BIG"" boss" -> The "BIG" boss

The "BIG" boss -> The "BIG" boss

The ""BIG"" boss -> The ""BIG"" boss

FIELDS ESCAPED BY 控制如何读取或写入特殊字符：

* 对于输入，如果FIELDS ESCAPED BY 字符不为空，则该字符的出现将被剥离，并且字符串的字面值作为字段值的一部分。一些两个字符的序列是异常，其中第一个字符是转义字符。这些序列显示在下表中（ \用于转义字符）。NULL本节稍后将介绍处理规则 。

| **字符** | **逃脱序列** |
| --- | --- |
| \0 | ASCII NUL（X'00'）字符 |
| \b | 退格字符 |
| \n | 换行符（换行）字符 |
| \r | 回车符 |
| \t | 标签字符 |
| \Z | ASCII 26（控制+ Z） |
| \N | 空值 |

* 有关\-escape语法的更多信息，请参见[第9.1.1节“字符串文字”](https://dev.mysql.com/doc/refman/5.7/en/string-literals.html)。
* 如果FIELDS ESCAPED BY字符为空，则不会发生转义序列解释。
* 对于输出，如果FIELDS ESCAPED BY 字符不为空，则用于在输出上前缀以下字符：
  + 该FIELDS ESCAPED BY字符
  + 该FIELDS [OPTIONALLY] ENCLOSED BY 字符
  + 第一个字符FIELDS TERMINATED BY和LINES TERMINATED BY 值
  + ASCII 0（转义字符后面实际写入的是ASCII码 0，而不是零值字节）

如果FIELDS ESCAPED BY字符是空的，没有字符被转义，并且NULL 是输出NULL，没有 \N。指定空转义字符可能不是一个好主意，特别是如果数据中的字段值包含刚刚给出的列表中的任何字符。

在某些情况下，现场和线路处理选项可以互动：

* 如果LINES TERMINATED BY是一个空字符串，FIELDS TERMINATED BY并且非空，则行也被终止FIELDS TERMINATED BY。
* 如果FIELDS TERMINATED BY和 FIELDS ENCLOSED BY值都为空（''），则使用固定行（非限制）格式。使用固定行格式，字段之间不使用分隔符（但是您仍然可以使用行终止符）。相反，使用足够宽的字段宽度读取和写入列值，以容纳字段中的所有值。为 [TINYINT](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html)， [SMALLINT](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html)， [MEDIUMINT](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html)， [INT](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html)，和 [BIGINT](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html)，场宽度是4，6，8，11，和20，分别，不管声明显示宽度是什么。

LINES TERMINATED BY仍然用于分隔线。如果一行不包含所有字段，其余列将被设置为其默认值。如果您没有行终止符，则应将其设置为 ''。在这种情况下，文本文件必须包含每行的所有字段。

固定行格式也影响NULL值的处理 ，如下所述。

**注意**

如果使用多字节字符集，固定大小格式不起作用。

办理NULL值根据不同 FIELDS和LINES使用的选项：

* 对于默认值FIELDS和 LINES值，NULL写入作为\N输出的字段值，字段值\N被读取 NULL为输入（假设 ESCAPED BY字符为\）。
* 如果FIELDS ENCLOSED BY不为空，则将包含文字的字段NULL作为其值读取为NULL值。这与字符中NULL包含 的字词不同，FIELDS ENCLOSED BY字符串被读取'NULL'。
* 如果FIELDS ESCAPED BY是空的， NULL写为单词 NULL。
* 用固定行格式（其使用时FIELDS TERMINATED BY和FIELDS ENCLOSED BY都为空），NULL被写为空字符串。这将导致NULL表中的值和空字符串在写入文件时无法区分，因为它们都被写为空字符串。如果您需要能够在读取文件时将其分开，则不应使用固定行格式。

尝试加载NULL到NOT NULL列会导致在列的数据类型和警告或严格SQL模式下的错误的隐式默认值分配。[第11.7节“数据类型默认值”](https://dev.mysql.com/doc/refman/5.7/en/data-type-defaults.html)中讨论隐式默认值 。

有些情况不支持 [LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)：

* 固定大小的行（FIELDS TERMINATED BY和 FIELDS ENCLOSED BY两个空）和 [BLOB](https://dev.mysql.com/doc/refman/5.7/en/blob.html)或 [TEXT](https://dev.mysql.com/doc/refman/5.7/en/blob.html)列。
* 如果您指定一个与另一个相同或前缀的分隔符，则[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)无法正确解释输入。例如，以下FIELDS条款会引起问题:

FIELDS TERMINATED BY '"' ENCLOSED BY '"'

* 如果FIELDS ESCAPED BY为空，则包含值的出现FIELDS ENCLOSED BY或LINES TERMINATED BY 后跟的字段值将FIELDS TERMINATED BY导致[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)停止早期读取字段或行。这是因为 [LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)无法正确确定字段或行值在哪里结束。

以下示例加载表的所有列 persondata：

LOAD DATA INFILE 'persondata.txt' INTO TABLE persondata;

默认情况下，如果在[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)语句末尾没有提供列列表 ，则输入行将包含每个表列的字段。如果要仅加载表的一些列，请指定列列表：

LOAD DATA INFILE 'persondata.txt' INTO TABLE persondata (col1,col2,...);

如果输入文件中的字段顺序与表中列的顺序不同，则还必须指定列列表。否则，MySQL不能指定如何将输入字段与表列匹配。

列列表可以包含列名或用户变量。使用用户变量，该SET子句使您能够在将结果分配给列之前对其值执行转换。

SET子句 中的用户变量可以通过多种方式使用。以下示例直接使用第一个输入列作为值t1.column1，并将第二个输入列分配给经过除法运算的用户变量，然后再使用以下值t1.column2：

LOAD DATA INFILE 'file.txt'

INTO TABLE t1

(column1, @var1)

SET column2 = @var1/100;

该SET子句可用于提供不从输入文件导出的值。以下语句设置 column3为当前日期和时间：

LOAD DATA INFILE 'file.txt'

INTO TABLE t1

(column1, column2)

SET column3 = CURRENT\_TIMESTAMP;

您还可以通过将输入值分配给用户变量并且不将变量分配给表列来舍弃输入值：

LOAD DATA INFILE 'file.txt'

INTO TABLE t1

(column1, @dummy, column2, @dummy, column3);

列/变量列表和SET子句的使用受以下限制：

* SET子句中的 赋值在赋值运算符的左侧只能有列名。
* 您可以在SET作业右侧使用子查询 。返回要分配给列的值的子查询可能仅是标量子查询。此外，您不能使用子查询从正在加载的表中进行选择。
* IGNORE列/变量列表或SET子句不处理 由子句 忽略的行。
* 用固定行格式加载数据时，不能使用用户变量，因为用户变量没有显示宽度。

处理输入行时，[LOAD DATA](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)将其分割为字段，并根据列/变量列表和SET子句（如果存在）使用值。然后将结果行插入到表中。如果表中有BEFORE INSERT或 AFTER INSERT触发了这些表，它们将分别在插入行之前或之后被激活。

如果输入行的字段太多，则会忽略额外的字段，并增加警告数。

如果输入行的字段太少，那么输入字段缺少的表列被设置为其默认值。[第11.7节“数据类型默认值”中](https://dev.mysql.com/doc/refman/5.7/en/data-type-defaults.html)介绍了默认值分配 。

空字段值被解释为与缺少的字段不同：

* 对于字符串类型，列设置为空字符串。
* 对于数字类型，列设置为0。
* 对于日期和时间类型，列将该类型设置为适当的 “ 零 ”值。请参见 [第11.3节“日期和时间类型”](https://dev.mysql.com/doc/refman/5.7/en/date-and-time-types.html)。

如果在一个[INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html)或多个 [UPDATE](https://dev.mysql.com/doc/refman/5.7/en/update.html)语句中明确地将字符串，数字或日期或时间类型明确指定为空字符串，则这些值将相同。

如果将SQL模式设置为限制值，处理空或不正确的字段值与刚描述的不同。例如，如果[sql\_mode](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_sql_mode)设置为 [TRADITIONAL](https://dev.mysql.com/doc/refman/5.7/en/sql-mode.html#sqlmode_traditional)空，则转换一个空值或一个值（例如'x'数字列）会导致错误，而不会转换为0.（使用 LOCAL或者IGNORE，即使使用限制[sql\_mode](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_sql_mode)值也可以使用警告而不是错误， 使用与非限制性SQL模式相同的最近值行为插入行，这是因为服务器无法在操作中间停止传输文件。）

[TIMESTAMP](https://dev.mysql.com/doc/refman/5.7/en/datetime.html)列仅当NULL 列的值（即，\N）和列未被声明为允许NULL值时设置为当前日期和时间，或者[TIMESTAMP](https://dev.mysql.com/doc/refman/5.7/en/datetime.html)列的默认值是当前时间戳，并且从字段中省略列表指定字段列表时。

[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)将所有输入视为字符串，因此您不能以语句的方式使用数字值[ENUM](https://dev.mysql.com/doc/refman/5.7/en/enum.html)或 [SET](https://dev.mysql.com/doc/refman/5.7/en/set.html)列 [INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html)。所有 [ENUM](https://dev.mysql.com/doc/refman/5.7/en/enum.html)和 [SET](https://dev.mysql.com/doc/refman/5.7/en/set.html)值必须指定为字符串。

[BIT](https://dev.mysql.com/doc/refman/5.7/en/bit-type.html)值不能使用二进制符号直接加载（例如， b'011010'）。要解决这个问题，请使用该 SET条款去除前导 b'和尾随，'并执行base-2到base-10转换，以便MySQL将值[BIT](https://dev.mysql.com/doc/refman/5.7/en/bit-type.html)正确加载到列中：

shell> cat /tmp/bit\_test.txt

b'10'

b'1111111'

shell> mysql test

mysql> LOAD DATA INFILE '/tmp/bit\_test.txt'

INTO TABLE bit\_test (@var1)

SET b = CAST(CONV(MID(@var1, 3, LENGTH(@var1)-3), 2, 10) AS UNSIGNED);

Query OK, 2 rows affected (0.00 sec)

Records: 2 Deleted: 0 Skipped: 0 Warnings: 0

mysql> SELECT BIN(b+0) FROM bit\_test;

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

| BIN(b+0) |

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

| 10 |

| 1111111 |

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

2 rows in set (0.00 sec)

为[BIT](https://dev.mysql.com/doc/refman/5.7/en/bit-type.html)在值 0b的二进制表示法（例如， 0b011010），用这个SET 子句代替以剥离领先0b：

SET b = CAST(CONV(MID(@var1, 3, LENGTH(@var1)-2), 2, 10) AS UNSIGNED)

在Unix上，如果您需要[LOAD DATA](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)从管道中读取，可以使用以下技术（该示例将/目录的列表加载到表中db1.t1）：

mkfifo /mysql/data/db1/ls.dat

chmod 666 /mysql/data/db1/ls.dat

find / -ls > /mysql/data/db1/ls.dat &

mysql -e "LOAD DATA INFILE 'ls.dat' INTO TABLE t1" db1

在这里，您必须运行生成要加载的数据的命令和单独的终端上的[**mysql**](https://dev.mysql.com/doc/refman/5.7/en/mysql.html)命令，或者在后台运行数据生成过程（如上例所示）。如果不这样做，管道将阻塞，直到[**mysql**](https://dev.mysql.com/doc/refman/5.7/en/mysql.html) 进程读取数据。

当[LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)语句完成，它返回以下格式的信息字符串：

Records: 1 Deleted: 0 Skipped: 0 Warnings: 0

警告发生在与使用该[INSERT](https://dev.mysql.com/doc/refman/5.7/en/insert.html)语句插入值相同的情况下（请参见[第13.2.5节“插入语法”](https://dev.mysql.com/doc/refman/5.7/en/insert.html)），但 [LOAD DATA INFILE](https://dev.mysql.com/doc/refman/5.7/en/load-data.html)在输入行中的字段太少或太多时也会生成警告。

您可以使用[SHOW WARNINGS](https://dev.mysql.com/doc/refman/5.7/en/show-warnings.html)第一个[max\_error\_count](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_max_error_count) 警告列表作为有关出现问题的信息。请参见 [第13.7.5.40节“显示警告语法”](https://dev.mysql.com/doc/refman/5.7/en/show-warnings.html)。

如果您正在使用C API，可以通过调用该[mysql\_info()](https://dev.mysql.com/doc/refman/5.7/en/mysql-info.html)函数来获取关于该语句的信息 。请参见 [第27.8.7.36节“mysql\_info（）”](https://dev.mysql.com/doc/refman/5.7/en/mysql-info.html)。

3、