1.Create Table

默认情况下，表使用默认的数据库创建 [InnoDB](https://dev.mysql.com/doc/refman/5.7/en/innodb-storage-engine.html" \o "InnoDB存储引擎)存储引擎。如果表存在，如果没有默认数据库，或者数据库不存在，则会发生错误。

1. CREATE [TEMPORARY] TABLE [IF NOT EXISTS] *tbl\_name*

(*create\_definition*,...)

[*table\_options*]

[*partition\_options*]

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

TEMPORARY创建表格时 可以使用关键字。一个TEMPORARY表只在当前会话中可见，而当会话关闭时自动删除。

## create\_definition:

*col\_name* *column\_definition*

| [CONSTRAINT [*symbol*]] PRIMARY KEY [*index\_type*] (*index\_col\_name*,...)

[*index\_option*] ...

| {INDEX|KEY} [*index\_name*] [*index\_type*] (*index\_col\_name*,...)

[*index\_option*] ...

| [CONSTRAINT [*symbol*]] UNIQUE [INDEX|KEY]

[*index\_name*] [*index\_type*] (*index\_col\_name*,...)

[*index\_option*] ...

| {FULLTEXT|SPATIAL} [INDEX|KEY] [*index\_name*] (*index\_col\_name*,...)

[*index\_option*] ...

| [CONSTRAINT [*symbol*]] FOREIGN KEY

[*index\_name*] (*index\_col\_name*,...) *reference\_definition*

| CHECK (*expr*)

1. CONSTRAINT ***symbol***如果给出了该子句，则该值（如果使用）在数据库中必须是唯一的。重复会 导致错误。
2. PRIMARY KEY一个唯一索引，其中所有的键列必须被定义为 NOT NULL。如果它们没有被明确声明NOT NULL，那么MySQL会隐式地声明它（默默地）。在创建的表中，PRIMARY KEY首先放置a ，然后是所有UNIQUE 索引，然后是非唯一索引。这有助于MySQL优化器优先使用哪个索引，并且更快速地检测重复UNIQUE密钥。A PRIMARY KEY可以是多列索引。但是，不能使用PRIMARY KEY列规范中的键属性创建多列索引。这样做只会将单个列标记为主要列。你必须使用一个单独的 条款。 PRIMARY KEY(***index\_col\_name***, ...)
3. UNIQUE一个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。
4. FULLTEXT一个FULLTEXT指标是用于全文搜索索引的一种特殊类型。
5. MySQL支持外键，它允许你在表之间交叉引用相关数据，外键约束有助于保持这种扩展数据的一致性。,外键可以保持在主表进行修改列时与之关联的列也会发生修改

### column\_definition:

*data\_type* [NOT NULL | NULL] [DEFAULT *default\_value*]

[AUTO\_INCREMENT] [UNIQUE [KEY]] [[PRIMARY] KEY]

[COMMENT '*string*']

[COLUMN\_FORMAT {FIXED|DYNAMIC|DEFAULT}]

[STORAGE {DISK|MEMORY|DEFAULT}]

[*reference\_definition*]

| *data\_type* [GENERATED ALWAYS] AS (*expression*)

[VIRTUAL | STORED] [NOT NULL | NULL]

[UNIQUE [KEY]] [[PRIMARY] KEY]

[COMMENT '*string*']

1.某些属性不适用于所有数据类型。 AUTO\_INCREMENT仅适用于整数和浮点类型。DEFAULT并不适用于[BLOB](https://dev.mysql.com/doc/refman/5.7/en/blob.html)， [TEXT](https://dev.mysql.com/doc/refman/5.7/en/blob.html)， GEOMETRY，和 [JSON](https://dev.mysql.com/doc/refman/5.7/en/json.html)类型。

2.字符数据类型（[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)）可以包括 CHARACTER SET（CHARSET ）和 COLLATE属性来指定该列的字符集和归类。

3. 如果既没有NULL也没有NOT NULL被指定，该列被视为 NULL已被指定。在MySQL 5.7中，只有InnoDB， MyISAM以及MEMORY 在可以有列存储引擎支持索引 NULL值。在其他情况下，您必须声明索引列NOT NULL

4.default指定列的默认值。有一个例外，默认值必须是一个常数; 它不能是一个功能或表达。不可以通过NOW（）等函数来指定。[BLOB](https://dev.mysql.com/doc/refman/5.7/en/blob.html)， [TEXT](https://dev.mysql.com/doc/refman/5.7/en/blob.html)和 [JSON](https://dev.mysql.com/doc/refman/5.7/en/json.html)列不能被分配一个默认值。

5．AUTO\_INCREMENT整数或浮点列可以具有附加属性AUTO\_INCREMENT。自动增长

要AUTO\_INCREMENT在插入行后检索值，请使用 [LAST\_INSERT\_ID()](https://dev.mysql.com/doc/refman/5.7/en/information-functions.html#function_last-insert-id)SQL函数或[mysql\_insert\_id()](https://dev.mysql.com/doc/refman/5.7/en/mysql-insert-id.html)C API函数。AUTO\_INCREMENT每个表 只能有一列，它必须被索引，并且不能有 DEFAULT值。

6.可以使用该COMMENT选项指定列的注释

7.COLUMN\_FORMAT目前对使用存储引擎NDB以外的表的列不起作用。指定列固定存储宽度

8. STORAGE对于[NDB](https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster.html)表，可以通过使用STORAGE子句来指定列是存储在磁盘上还是存储在内存中。STORAGE DISK导致列存储在磁盘上，并 STORAGE MEMORY导致使用内存中的存储，使用该语句的create语句必须包含包含一个 TABLESPACE条款。对于[NDB](https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster.html)表，STORAGE DEFAULT相当于STORAGE MEMORY。

9.GENERATED ALWAYS用于指定生成的列表达式。有关[生成的列的](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_generated_column)信息，请参见 [第13.1.18.8节“创建表和生成的列”](https://dev.mysql.com/doc/refman/5.7/en/create-table-generated-columns.html)。

#### data\_type:

BIT[(*length*)]

| TINYINT[(*length*)] [UNSIGNED] [ZEROFILL]

| SMALLINT[(*length*)] [UNSIGNED] [ZEROFILL]

| MEDIUMINT[(*length*)] [UNSIGNED] [ZEROFILL]

| INT[(*length*)] [UNSIGNED] [ZEROFILL]

| INTEGER[(*length*)] [UNSIGNED] [ZEROFILL]

| BIGINT[(*length*)] [UNSIGNED] [ZEROFILL]

| REAL[(*length*,*decimals*)] [UNSIGNED] [ZEROFILL]

| DOUBLE[(*length*,*decimals*)] [UNSIGNED] [ZEROFILL]

| FLOAT[(*length*,*decimals*)] [UNSIGNED] [ZEROFILL]

| DECIMAL[(*length*[,*decimals*])] [UNSIGNED] [ZEROFILL]

| NUMERIC[(*length*[,*decimals*])] [UNSIGNED] [ZEROFILL]

| DATE

| TIME[(*fsp*)]

| TIMESTAMP[(*fsp*)]

| DATETIME[(*fsp*)]

| YEAR

| CHAR[(*length*)]

[CHARACTER SET *charset\_name*] [COLLATE *collation\_name*]

| VARCHAR(*length*)

[CHARACTER SET *charset\_name*] [COLLATE *collation\_name*]

| BINARY[(*length*)]

| VARBINARY(*length*)

| TINYBLOB

| BLOB[(*length*)]

| MEDIUMBLOB

| LONGBLOB

| TINYTEXT

[CHARACTER SET *charset\_name*] [COLLATE *collation\_name*]

| TEXT[(*length*)]

[CHARACTER SET *charset\_name*] [COLLATE *collation\_name*]

| MEDIUMTEXT

[CHARACTER SET *charset\_name*] [COLLATE *collation\_name*]

| LONGTEXT

[CHARACTER SET *charset\_name*] [COLLATE *collation\_name*]

| ENUM(*value1*,*value2*,*value3*,...)

[CHARACTER SET *charset\_name*] [COLLATE *collation\_name*]

| SET(*value1*,*value2*,*value3*,...)

[CHARACTER SET *charset\_name*] [COLLATE *collation\_name*]

| JSON

| spatial type *空间数据类型*

### index\_col\_name:

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

### index\_type:

USING {BTREE | HASH}

### index\_option:

KEY\_BLOCK\_SIZE [=] *value*

| *index\_type*

| WITH PARSER *parser\_name*

| COMMENT '*string*'

### reference\_definition: //外键操作设置

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

[MATCH FULL | MATCH PARTIAL | MATCH SIMPLE]

[ON DELETE *reference\_option*]

[ON UPDATE *reference\_option*]

***index\_name***代表一个外键ID。***index\_name***如果在可以支持外键的子表上已经有了一个显式定义的索引，则该值将被忽略。否则，MySQL会隐式创建一个按照以下规则命名的外键索引：

* 如果已定义，CONSTRAINT ***symbol***则使用该值。否则，使用该FOREIGN KEY ***index\_name***值。
* 如果既没有CONSTRAINT ***symbol***或FOREIGN KEY ***index\_name***定义，使用引用外键列的名称所产生的外键索引名。

外键定义受以下条件限制：

1. 外键关系涉及一个保存中心数据值的 父表，以及一个具有相同值的 子表指向其父项。该 FOREIGN KEY子句在子表中指定。父表和子表必须使用相同的存储引擎。他们不能是 TEMPORARY桌子。
2. 外键和引用键中的相应列必须具有相似的数据类型。整数类型的大小和符号必须相同。字符串类型的长度不需要相同。对于非二进制（字符）字符串列，字符集和归类必须相同。
3. 何时启用foreign\_key\_checks（默认设置）时，不允许在包含外键约束中使用的字符串列的表上使用字符集转换
4. MySQL需要对外键和引用键进行索引，以便外键检查可以很快并且不需要表扫描。在引用表中，必须有一个索引，其中外键列以相同顺序排列为 第一列。如果索引表不存在，则会在引用表上自动创建这样的索引。如果您创建另一个可用于强制实施外键约束的索引，则此索引可能稍后会自动放弃。 index\_name，如果给出，如前所述使用。
5. InnoDB允许一个外键引用任何列或一组列。但是，在被引用的表中，必须有一个索引，其中被引用的列按照相同的顺序被列为第一列。NDB 需要在引用为外键的任何列上显式唯一键（或主键）
6. 外键列上的索引前缀不受支持。这样做的一个后果是， BLOB和 TEXT列不能被包括在一个外键，因为对这些列的索引必须总是包含一个前缀长度。
7. 如果给出了该子句，则该值（如果使用）在数据库中必须是唯一的。重复 将导致错误
8. InnoDB当前不支持具有用户定义分区的表的外键。这包括父表和子表。

#### reference\_option:

RESTRICT | CASCADE | SET NULL | NO ACTION | SET DEFAULT

* 1. CASCADE：从父表中删除或更新行，并自动删除或更新子表中的匹配行。这两个ON DELETE CASCADE和ON UPDATE CASCADE 支持。在两个表之间，不要定义ON UPDATE CASCADE在父表或子表中的同一列上执行的几个 子句。
  2. SET NULL：删除或更新父表中的行，并将子表中的外键列设置为NULL。这两个 ON DELETE SET NULL和ON UPDATE SET NULL子句都支持。如果您指定了一个SET NULL操作，请 确保您没有将子表中的列声明为NOT NULL
  3. RESTRICT：拒绝父表的删除或更新操作。指定 RESTRICT（或NO ACTION）与删除ON DELETEor ON UPDATE子句相同。
  4. NO ACTION：来自标准SQL的关键字。在MySQL中，相当于RESTRICT。如果被引用的表中存在相关的外键值，MySQL服务器会拒绝父表的删除或更新操作。有些数据库系统有延期检查，NO ACTION是延期检查。在MySQL中，外键约束被立即检查，所以与之NO ACTION相同 RESTRICT。
  5. SET DEFAULT：这个动作由MySQL解析器认可，但两者 InnoDB并 NDB拒绝包含表定义ON DELETE SET DEFAULT或 ON UPDATE SET DEFAULT条款。

**对于没有指定的ON DELETE或ON UPDATE默认的动作总是RESTRICT。**

CREATE TABLE parent (

id INT NOT NULL,

PRIMARY KEY (id)

) ENGINE=INNODB;

CREATE TABLE child (

id INT,

parent\_id INT,

INDEX par\_ind (parent\_id),

FOREIGN KEY (parent\_id)

REFERENCES parent(id)

ON DELETE CASCADE

) ENGINE=INNODB;

##### 添加外键

ALTER TABLE *tbl\_name*

ADD [CONSTRAINT [*symbol*]] FOREIGN KEY

[*index\_name*] (*index\_col\_name*, ...)

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

[ON DELETE *reference\_option*]

[ON UPDATE *reference\_option*]

##### 丢弃外键

ALTER TABLE tbl\_name DROP FOREIGN KEY fk\_symbol;

## table\_options:

表选项用于优化表的行为。在大多数情况下，你不必指定任何一个。除非另有说明，这些选项适用于所有存储引擎。

*table\_option* [[,] *table\_option*] ...

### table\_option:

AUTO\_INCREMENT [=] *value*

| AVG\_ROW\_LENGTH [=] *value*

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

| CHECKSUM [=] {0 | 1}

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

| COMMENT [=] '*string*'

| COMPRESSION [=] {'ZLIB'|'LZ4'|'NONE'}

| CONNECTION [=] '*connect\_string*'

| {DATA|INDEX} DIRECTORY [=] '*absolute path to directory*'

| DELAY\_KEY\_WRITE [=] {0 | 1}

| ENCRYPTION [=] {'Y' | 'N'}

| ENGINE [=] *engine\_name*

| INSERT\_METHOD [=] { NO | FIRST | LAST }

| KEY\_BLOCK\_SIZE [=] *value*

| MAX\_ROWS [=] *value*

| MIN\_ROWS [=] *value*

| PACK\_KEYS [=] {0 | 1 | DEFAULT}

| PASSWORD [=] '*string*'

| ROW\_FORMAT [=] {DEFAULT|DYNAMIC|FIXED|COMPRESSED|REDUNDANT|COMPACT}

| STATS\_AUTO\_RECALC [=] {DEFAULT|0|1}

| STATS\_PERSISTENT [=] {DEFAULT|0|1}

| STATS\_SAMPLE\_PAGES [=] *value*

| TABLESPACE *tablespace\_name* [STORAGE {DISK|MEMORY|DEFAULT}] //STORAGE设置存储范围

| UNION [=] (*tbl\_name*[,*tbl\_name*]...)

1. AUTO\_INCREMENT 设置表格表格 的初始增长值或者为不支持AUTO\_INCREMENT表格选项的引擎设置第一个自动增量值，请在创建表格后插入一个“ dummy ”行，其值小于所需值，然后删除虚拟行。也可以在alter table时对AUTO\_INCREMENT的值进行修改。Alter table XXX auto\_increment = ?; ?不可以小于当前列最大值
2. AVG\_ROW\_LENGTH 表格的平均行长度的近似值。您只需要为具有可变大小行的大型表设置此项。当你创建一个MyISAM表时，MySQL使用MAX\_ROWS和 AVG\_ROW\_LENGTH选项的乘积来决定结果表的大小。如果您不指定任一选项，则MyISAM数据和索引文件的最大大小默认为256TB。（如果你的操作系统不支持大的文件，那么表的大小受到文件大小限制的限制。）如果你想缩小指针的大小以使索引更小，更快，而且你不需要大文件，可以通过设置[myisam\_data\_pointer\_size](https://dev.mysql.com/doc/refman/5.7/en/server-system-variables.html#sysvar_myisam_data_pointer_size) 系统变量来减少默认的指针大小 。如果你希望你所有的表能够增长到默认的限制以上，并且愿意让你的表稍微慢一点，那么你可以通过设置这个变量来增加默认的指针大小。将该值设置为7可以使表格尺寸达到65,536TB。
3. [DEFAULT] CHARACTER SET指定表的默认字符集。 CHARSET是一个同义词CHARACTER SET。如果字符集名称是 DEFAULT，则使用数据库字符集。
4. CHECKSUM如果你希望MySQL为所有行维护一个实时校验和（也就是MySQL在表更改时自动更新的校验和），则将其设置为1。这使表更新慢一点，但也更容易找到损坏的表。该[CHECKSUM TABLE](https://dev.mysql.com/doc/refman/5.7/en/checksum-table.html)声明报告校验和。
5. [DEFAULT] COLLATE指定表的默认排序规则。
6. COMMENT 为表格添加一个注解
7. COMPRESSION用于InnoDB表级别压缩的压缩算法 。
8. CONNECTION FEDERATED 表 的连接字符串
9. DATA DIRECTORY， INDEX DIRECTORY 指定放置 表的数据文件和索引文件的位置。创建MyISAM表时当创建带有该选项的表时，MySQL不会创建对应于数据库名称的子目录。文件在指定的目录中创建。 DATA DIRECTORY='***directory***'INDEX DIRECTORY='***directory***'
10. DELAY\_KEY\_WRITE如果要延迟表的关键更新，则将其设置为1，直到表关闭。
11. ENCRYPTION将该ENCRYPTION选项 设置'Y'为对InnoDB在每个[文件表格](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_file_per_table) 空间中创建的表 启用页面级数据加密。选项值不区分大小写。该 ENCRYPTION选项是与InnoDB表空间加密功能一起引入的
12. ENGINE 使用下表中显示的名称之一指定表的存储引擎。引擎名称可以不加引号或引用。

| **存储引擎** | **描述** |
| --- | --- |
| InnoDB | 具有行锁定和外键的事务安全表。新表的默认存储引擎。如果你有MySQL的经验但是新手， 请参阅 [第14章*InnoDB存储引擎*](https://dev.mysql.com/doc/refman/5.7/en/innodb-storage-engine.html)，尤其是[第14.1节“InnoDB入门”](https://dev.mysql.com/doc/refman/5.7/en/innodb-introduction.html)InnoDB。 |
| MyISAM | 主要用于只读或读取主要工作负载的二进制便携式存储引擎。请参见 [第15.2节“MyISAM存储引擎”](https://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html)。 |
| MEMORY | 此存储引擎的数据仅存储在内存中。请参见 [第15.3节“内存存储引擎”](https://dev.mysql.com/doc/refman/5.7/en/memory-storage-engine.html)。 |
| CSV | 以逗号分隔值格式存储行的表格。请参见 [第15.4节“CSV存储引擎”](https://dev.mysql.com/doc/refman/5.7/en/csv-storage-engine.html)。 |
| ARCHIVE | 归档存储引擎。请参见 [第15.5节“ARCHIVE存储引擎”](https://dev.mysql.com/doc/refman/5.7/en/archive-storage-engine.html)。 |
| EXAMPLE | 示例引擎。请参见[第15.9节“示例存储引擎”](https://dev.mysql.com/doc/refman/5.7/en/example-storage-engine.html" \o "15.9示例存储引擎)。 |
| FEDERATED | 访问远程表的存储引擎。请参见 [第15.8节“联邦存储引擎”](https://dev.mysql.com/doc/refman/5.7/en/federated-storage-engine.html)。 |
| HEAP | 这是一个同义词MEMORY。 |
| MERGE | 集合MyISAM表作为一个表。也被称为MRG\_MyISAM。请参见 [第15.7节“MERGE存储引擎”](https://dev.mysql.com/doc/refman/5.7/en/merge-storage-engine.html)。 |
| [NDB](https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster.html) | 集群，容错，基于内存的表，支持事务和外键。也被称为 [NDBCLUSTER](https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster.html)。请参阅 [第21章，*MySQL NDB集群7.5和NDB集群7.6*](https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster.html)。 |

1. INSERT\_METHOD如果要将数据插入到MERGE 表中，则必须使用INSERT\_METHOD 应插入行的表进行指定。 INSERT\_METHOD是MERGE仅用于表格的选项 。使用值 FIRST或LAST插入到第一个或最后一个表，或值 NO以防止插入
2. KEY\_BLOCK\_SIZE对于[MyISAM](https://dev.mysql.com/doc/refman/5.7/en/myisam-storage-engine.html)表， 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。
3. MAX\_ROWS您计划在表中存储的最大行数。这不是硬性的限制，而是对存储引擎的暗示，表必须能够存储至少这么多行。
4. MIN\_ROWS您计划在表格中存储的最小行数。该 [MEMORY](https://dev.mysql.com/doc/refman/5.7/en/memory-storage-engine.html)存储引擎使用此选项，因为一个关于内存使用提示。
5. PACK\_KEYS只适用于MyISAM表格。如果您希望有较小的索引，请将此选项设置为1。这通常会使更新更慢，读取速度更快。将该选项设置为0将禁用所有按键的打包。将其设置为 DEFAULT告诉存储引擎只封装长[CHAR](https://dev.mysql.com/doc/refman/5.7/en/char.html" \o "11.4.1 CHAR和VARCHAR类型)， [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)，或 [VARBINARY](https://dev.mysql.com/doc/refman/5.7/en/binary-varbinary.html)列。默认只打包字符串
6. PASSWORD 此选项未使用。
7. ROW\_FORMAT 定义存储行的物理格式。
8. STATS\_AUTO\_RECALC指定是否自动重新计算 [持续的统计数据](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_persistent_statistics)为InnoDB表。
9. STATS\_PERSISTENT 指定是否启用 [持续统计](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_persistent_statistics)为一个InnoDB表。
10. STATS\_SAMPLE\_PAGES估计索引列的基数和其他统计信息时要抽样的索引页的数量。
11. TABLESPACE该TABLESPACE选项用于在InnoDB常规表空间中创建表。TABLESPACE选项可用于将InnoDB表分区或子分区分配给 [常规表空间](https://dev.mysql.com/doc/refman/5.7/en/glossary.html#glos_general_tablespace)。

## [partition\_optio ns:](https://dev.mysql.com/doc/refman/5.7/en/create-table.html) 创建分区表

PARTITION BY

{ [LINEAR] HASH(*expr*)

| [LINEAR] KEY [ALGORITHM={1|2}] (*column\_list*)

| RANGE{(*expr*) | COLUMNS(*column\_list*)}

| LIST{(*expr*) | COLUMNS(*column\_list*)} }

[PARTITIONS *num*]

[SUBPARTITION BY

{ [LINEAR] HASH(*expr*)

| [LINEAR] KEY [ALGORITHM={1|2}] (*column\_list*) }

[SUBPARTITIONS *num*]

]

[(*partition\_definition* [, *partition\_definition*] ...)]

### partition\_definition:

PARTITION *partition\_name*

[VALUES

{LESS THAN {(*expr* | *value\_list*) | MAXVALUE}

|

IN (*value\_list*)}]

[[STORAGE] ENGINE [=] *engine\_name*]

[COMMENT [=] '*string*' ]

[DATA DIRECTORY [=] '*data\_dir*']

[INDEX DIRECTORY [=] '*index\_dir*']

[MAX\_ROWS [=] *max\_number\_of\_rows*]

[MIN\_ROWS [=] *min\_number\_of\_rows*]

[TABLESPACE [=] tablespace\_name]

[(*subpartition\_definition* [, *subpartition\_definition*] ...)]

### subpartition\_definition:

SUBPARTITION *logical\_name*

[[STORAGE] ENGINE [=] *engine\_name*]

[COMMENT [=] '*string*' ]

[DATA DIRECTORY [=] '*data\_dir*']

[INDEX DIRECTORY [=] '*index\_dir*']

[MAX\_ROWS [=] *max\_number\_of\_rows*]

[MIN\_ROWS [=] *min\_number\_of\_rows*]

[TABLESPACE [=] tablespace\_name]

### query\_expression:

SELECT ... (*Some valid select or union statement*)

2. CREATE TABLE ... SELECT语法

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] *tbl\_name*

[(*create\_definition*,...)]

[*table\_options*]

[*partition\_options*]

[IGNORE | REPLACE]

[AS] *query\_expression*

要从另一个表中创建一个表，请在[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html" \o "13.2.9 SELECT语法)语句的末尾添加一条 [CREATE TABLE](https://dev.mysql.com/doc/refman/5.7/en/create-table.html)语句：CREATE TABLE new\_tbl AS SELECT \* FROM orig\_tbl;

在由此产生的表中 [CREATE TABLE ... SELECT](https://dev.mysql.com/doc/refman/5.7/en/create-table.html)，仅在[CREATE TABLE](https://dev.mysql.com/doc/refman/5.7/en/create-table.html)部分中命名的列 首先出现。在这两部分命名的列或仅在[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)部分命名的列 。[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html" \o "13.2.9 SELECT语法)列的数据类型也可以通过指定[CREATE TABLE](https://dev.mysql.com/doc/refman/5.7/en/create-table.html)零件中的列来覆盖 。

如果在将数据复制到表格时发生任何错误，则会自动删除而不是创建。

您可以先于[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)通过 IGNORE或REPLACE指示如何处理重复的唯一键值的行。与此同时IGNORE，丢弃在唯一键值上复制现有行的行。用 REPLACE新行替换具有相同唯一键值的行。如果既没有IGNORE也没有 REPLACE被指定，重复的唯一键值导致错误。

因为基本[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)语句中的行的排序 不能总是被确定，CREATE TABLE ... IGNORE SELECT 并且CREATE TABLE ... REPLACE SELECT 语句被标记为不安全的基于语句的复制。这些语句在使用基于语句的模式时会在错误日志中产生警告，并在使用MIXED模式时使用基于行的格式写入二进制日志 。

[CREATE TABLE ... SELECT](https://dev.mysql.com/doc/refman/5.7/en/create-table.html)不会自动为您创建任何索引。这样做是为了尽可能灵活地做出陈述。如果要在创建的表中创建索引，则应在[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)语句前指定这些索引 ：

mysql> CREATE TABLE bar (UNIQUE (n)) SELECT n FROM foo;

3. CREATE TABLE ... LIKE语法

CREATE [TEMPORARY] TABLE [IF NOT EXISTS] *tbl\_name*

{ LIKE *old\_tbl\_name* | (LIKE *old\_tbl\_name*) }

LIKE 仅适用于基表，不适用于视图。

用于CREATE TABLE ... LIKE根据另一个表的定义（包括原始表中定义的所有列属性和索引）创建一个空表：

因为CREATE TABLE ... LIKE目标表保留了原始表中生成的列信息。

CREATE TABLE ... LIKE不保留为原始表指定的任何 DATA DIRECTORY或INDEX DIRECTORY表选项或任何外键定义。

如果原始表是一张TEMPORARY表， CREATE TABLE ... LIKE则不保留 TEMPORARY。要创建 TEMPORARY目标表，请使用 CREATE TEMPORARY TABLE ... LIKE。

对于CREATE TABLE ... SELECT目标表，不保留有关选定源表中的列是否为生成列的信息。[SELECT](https://dev.mysql.com/doc/refman/5.7/en/select.html)语句的 部分不能将值分配给目标表中的生成列。

一些数据类型的转换可能会发生。例如，该 AUTO\_INCREMENT属性不会保留，并且[VARCHAR](https://dev.mysql.com/doc/refman/5.7/en/char.html)列可以成为 [CHAR](https://dev.mysql.com/doc/refman/5.7/en/char.html)列。再培训属性是NULL（或NOT NULL），对于有他们这些列， CHARACTER SET，COLLATION， COMMENT，和DEFAULT 条款。

在创建表时 [CREATE TABLE ... SELECT](https://dev.mysql.com/doc/refman/5.7/en/create-table-select.html)，请确保在查询中使用别名函数调用或表达式。如果您不这样做，则 CREATE说明可能会失败或导致不需要的列名称。

CREATE TABLE artists\_and\_works

SELECT artist.name, COUNT(work.artist\_id) AS number\_of\_works

FROM artist LEFT JOIN work ON artist.id = work.artist\_id

GROUP BY artist.id;

\您还可以在创建的表中明确指定列的数据类型：

CREATE TABLE foo (a TINYINT NOT NULL) SELECT b+1 AS a FROM bar;

4.alter table 例子

1.

CREATE TABLE t1 (a INTEGER, b CHAR(10));

将表格重命名t1为 t2：

ALTER TABLE t1 RENAME t2;

若要更改列一个从 [INTEGER](https://dev.mysql.com/doc/refman/5.7/en/integer-types.html)到TINYINT NOT NULL（名字一样），并更改列 b从CHAR(10)到 CHAR(20)以及来自重命名 b到c：

ALTER TABLE t2 MODIFY a TINYINT NOT NULL, CHANGE b c CHAR(20); //modify更改单个列的字符集

要添加[TIMESTAMP](https://dev.mysql.com/doc/refman/5.7/en/datetime.html)名为的新列d：

ALTER TABLE t2 ADD d TIMESTAMP;

在列上添加索引d和 在列上添加UNIQUE索引a：

ALTER TABLE t2 ADD INDEX (d), ADD UNIQUE (a);

删除列c：

ALTER TABLE t2 DROP COLUMN c;

要添加一个AUTO\_INCREMENT名为c的新整数列：

ALTER TABLE t2 ADD c INT UNSIGNED NOT NULL AUTO\_INCREMENT,

ADD PRIMARY KEY (c);

2. 对于[NDB](https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster.html)表格，也可以更改用于表格或列的存储类型。例如，考虑[NDB](https://dev.mysql.com/doc/refman/5.7/en/mysql-cluster.html)创建一个如下所示的表格：

mysql> CREATE TABLE t1 (c1 INT) TABLESPACE ts\_1 ENGINE NDB;

要将此表转换为基于磁盘的存储，可以使用以下[ALTER TABLE](https://dev.mysql.com/doc/refman/5.7/en/alter-table.html)语句：

mysql> ALTER TABLE t1 TABLESPACE ts\_1 STORAGE DISK;

5. TRUNCATE TABLE

TRUNCATE [TABLE] tbl\_name

完全清空一张表 ，类似于delete，删除所有行