```
assignment_list:
    assignment [, assignment] ...
```

REPLACE works exactly like INSERT, except that if an old row in the table has the same value as a new row for a PRIMARY KEY or a UNIQUE index, the old row is deleted before the new row is inserted. See Section 13.2.6, "INSERT Statement".

REPLACE is a MySQL extension to the SQL standard. It either inserts, or *deletes* and inserts. For another MySQL extension to standard SQL—that either inserts or *updates*—see Section 13.2.6.2, "INSERT ... ON DUPLICATE KEY UPDATE Statement".

DELAYED inserts and replaces were deprecated in MySQL 5.6. In MySQL 8.0, DELAYED is not supported. The server recognizes but ignores the DELAYED keyword, handles the replace as a nondelayed replace, and generates an ER_WARN_LEGACY_SYNTAX_CONVERTED warning: REPLACE DELAYED is no longer supported. The statement was converted to REPLACE. The DELAYED keyword is scheduled for removal in a future release. release.

Note

REPLACE makes sense only if a table has a PRIMARY KEY or UNIQUE index. Otherwise, it becomes equivalent to INSERT, because there is no index to be used to determine whether a new row duplicates another.

Values for all columns are taken from the values specified in the REPLACE statement. Any missing columns are set to their default values, just as happens for INSERT. You cannot refer to values from the current row and use them in the new row. If you use an assignment such as SET *col_name* = *col_name* + 1, the reference to the column name on the right hand side is treated as DEFAULT(*col_name*), so the assignment is equivalent to SET *col_name* = DEFAULT(*col_name*) + 1.

In MySQL 8.0.19 and later, you can specify the column values that REPLACE attempts to insert using VALUES ROW().

To use REPLACE, you must have both the INSERT and DELETE privileges for the table.

If a generated column is replaced explicitly, the only permitted value is DEFAULT. For information about generated columns, see Section 13.1.20.8, "CREATE TABLE and Generated Columns".

REPLACE supports explicit partition selection using the PARTITION clause with a list of comma-separated names of partitions, subpartitions, or both. As with INSERT, if it is not possible to insert the new row into any of these partitions or subpartitions, the REPLACE statement fails with the error Found a row not matching the given partition set. For more information and examples, see Section 24.5, "Partition Selection".

The REPLACE statement returns a count to indicate the number of rows affected. This is the sum of the rows deleted and inserted. If the count is 1 for a single-row REPLACE, a row was inserted and no rows were deleted. If the count is greater than 1, one or more old rows were deleted before the new row was inserted. It is possible for a single row to replace more than one old row if the table contains multiple unique indexes and the new row duplicates values for different old rows in different unique indexes.

The affected-rows count makes it easy to determine whether REPLACE only added a row or whether it also replaced any rows: Check whether the count is 1 (added) or greater (replaced).

If you are using the C API, the affected-rows count can be obtained using the mysql_affected_rows() function.

You cannot replace into a table and select from the same table in a subquery.