

13.1.10. UPDATE Syntax

Single-table syntax:

```
UPDATE [LOW_PRIORITY] [IGNORE] tbl_name
  SET col_name1=expr1 [, col_name2=expr2 ...]
  [WHERE where_definition]
  [ORDER BY ...]
  [LIMIT row_count]
```

Multiple-table syntax:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_references
  SET col_name1=expr1 [, col_name2=expr2 ...]
  [WHERE where_definition]
```

The UPDATE statement updates columns in existing table rows with new values. The SET clause indicates which columns to modify and the values they should be given. The WHERE clause, if given, specifies which rows should be updated. Otherwise, all rows are updated. If the ORDER BY clause is specified, the rows are updated in the order that is specified. The LIMIT clause places a limit on the number of rows that can be updated.

The UPDATE statement supports the following modifiers:

- If you specify the LOW_PRIORITY keyword, execution of the UPDATE is delayed until no other clients are reading from the table.
- If you specify the IGNORE keyword, the update statement does not abort even if errors occur during the update. Rows for which duplicate-key conflicts occur are not updated. Rows for which columns are updated to values that would cause data conversion errors are updated to the closest valid values instead.

If you access a column from *tbl_name* in an expression, UPDATE uses the current value of the column. For example, the following statement sets the age column to one more than its current value:

```
mysql> UPDATE persondata SET age=age+1;
```

UPDATE assignments are evaluated from left to right. For example, the following statement doubles the age column, then increments it:

```
mysql> UPDATE persondata SET age=age*2, age=age+1;
```

If you set a column to the value it currently has, MySQL notices this and doesn't update it.

If you update a column that has been declared NOT NULL by setting to NULL, the column is set to the default value appropriate for the column type and the warning count is incremented. The default value is 0 for numeric types, the empty string (' ') for string types, and the ``zero" value for date and time types.

UPDATE returns the number of rows that were actually changed. In MySQL 3.22 or later, the mysql_info() C API function returns the number of rows that were matched and updated and the number of warnings that occurred during the UPDATE.

Starting from MySQL 3.23, you can use `LIMIT row_count` to restrict the scope of the `UPDATE`. A `LIMIT` clause works as follows:

- Before MySQL 4.0.13, `LIMIT` is a rows-affected restriction. The statement stops as soon as it has changed `row_count` rows that satisfy the `WHERE` clause.
- From 4.0.13 on, `LIMIT` is a rows-matched restriction. The statement stops as soon as it has found `row_count` rows that satisfy the `WHERE` clause, whether or not they actually were changed.

If an `UPDATE` statement includes an `ORDER BY` clause, the rows are updated in the order specified by the clause. `ORDER BY` can be used from MySQL 4.0.0.

Starting with MySQL 4.0.4, you can also perform `UPDATE` operations that cover multiple tables. The *table_references* part lists the tables involved in the join. Its syntax is described in [Section 13.1.7.1, "JOIN Syntax"](#). Here is an example:

```
UPDATE items,month SET items.price=month.price
WHERE items.id=month.id;
```

The example shows an inner join using the comma operator, but multiple-table `UPDATE` statements can use any type of join allowed in `SELECT` statements, such as `LEFT JOIN`.

Note: You cannot use `ORDER BY` or `LIMIT` with multiple-table `UPDATE`.

Before MySQL 4.0.18, you need the `UPDATE` privilege for all tables used in a multiple-table `UPDATE`, even if they were not updated. As of MySQL 4.0.18, you need only the `SELECT` privilege for any columns that are read but not modified.

If you use a multiple-table `UPDATE` statement involving `InnoDB` tables for which there are foreign key constraints, the MySQL optimizer might process tables in an order that differs from that of their parent/child relationship. In this case, the statement fails and rolls back. Instead, update a single table and rely on the `ON UPDATE` capabilities that `InnoDB` provides to cause the other tables to be modified accordingly.

Currently, you cannot update a table and select from the same table in a subquery.