## 13.7.1.8 REVOKE Statement

```
REVOKE [IF EXISTS]
    priv_type [(column_list)]
      [, priv_type [(column_list)]] ...
    ON [object_type] priv_level
    FROM user_or_role [, user_or_role] ...
    [IGNORE UNKNOWN USER]

REVOKE [IF EXISTS] ALL [PRIVILEGES], GRANT OPTION
    FROM user_or_role [, user_or_role] ...
    [IGNORE UNKNOWN USER]

REVOKE [IF EXISTS] PROXY ON user_or_role
    FROM user_or_role [, user_or_role] ...
    [IGNORE UNKNOWN USER]

REVOKE [IF EXISTS] role [, role ] ...
    FROM user_or_role [, user_or_role ] ...
    [IGNORE UNKNOWN USER]

user_or_role: {
    user (see Section 6.2.4, “Specifying Account Names”)
  | role (see Section 6.2.5, “Specifying Role Names”
}
```

The `REVOKE` statement enables system administrators to revoke privileges and roles, which can be revoked from user accounts and roles.

For details on the levels at which privileges exist, the permissible ***`priv_type`***, ***`priv_level`***, and ***`object_type`*** values, and the syntax for specifying users and passwords, see Section 13.7.1.6, “GRANT Statement”.

For information about roles, see Section 6.2.10, “Using Roles”.

When the `read_only` system variable is enabled, `REVOKE` requires the `CONNECTION_ADMIN` or privilege (or the deprecated `SUPER` privilege), in addition to any other required privileges described in the following discussion.

Beginning with MySQL 8.0.30, all the forms shown for `REVOKE` support an `IF EXISTS` option as well as an `IGNORE UNKNOWN USER` option. With neither of these modifications, `REVOKE` either succeeds for all named users and roles, or rolls back and has no effect if any error occurs; the statement is written to the binary log only if it succeeds for all named users and roles. The precise effects of `IF EXISTS` and `IGNORE UNKNOWN USER` are discussed later in this section.

Each account name uses the format described in Section 6.2.4, “Specifying Account Names”. Each role name uses the format described in Section 6.2.5, “Specifying Role Names”. For example:

```
REVOKE INSERT ON *.* FROM 'jeffrey'@'localhost';
REVOKE 'role1', 'role2' FROM 'user1'@'localhost', 'user2'@'localhost';
REVOKE SELECT ON world.* FROM 'role3';
```

The host name part of the account or role name, if omitted, defaults to `'%'`.

To use the first `REVOKE` syntax, you must have the `GRANT OPTION` privilege, and you must have the privileges that you are revoking.

To revoke all privileges, use the second syntax, which drops all global, database, table, column, and routine privileges for the named users or roles:

```
REVOKE ALL PRIVILEGES, GRANT OPTION
  FROM user_or_role [, user_or_role] ...
```

`REVOKE ALL PRIVILEGES, GRANT OPTION` does not revoke any roles.

To use this `REVOKE` syntax, you must have the global `CREATE USER` privilege, or the `UPDATE` privilege for the `mysql` system schema.

The syntax for which the `REVOKE` keyword is followed by one or more role names takes a `FROM` clause indicating one or more users or roles from which to revoke the roles.

The `IF EXISTS` and `IGNORE UNKNOWN USER` options (MySQL 8.0.30 and later) have the effects listed here:

- `IF EXISTS` means that, if the target user or role exists but no such privilege or role is found assigned to the target for any reason, a warning is raised, instead of an error; if no privilege or role named by the statement is assigned to the target, the statement has no (other) effect. Otherwise, `REVOKE` executes normally; if the user does not exist, the statement raises an error.

  *Example*: Given table `t1` in database `test`, we execute the following statements, with the results shown.

  ```
  mysql> CREATE USER jerry@localhost;
  Query OK, 0 rows affected (0.01 sec)
  ```

```
mysql> REVOKE SELECT ON test.t1 FROM jerry@localhost;
ERROR 1147 (42000): There is no such grant defined for user 'jerry' on host
'localhost' on table 't1'
mysql> REVOKE IF EXISTS SELECT ON test.t1 FROM jerry@localhost;
Query OK, 0 rows affected, 1 warning (0.00 sec)

mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Warning
   Code: 1147
Message: There is no such grant defined for user 'jerry' on host 'localhost' o
table 't1'
1 row in set (0.00 sec)
```

`IF EXISTS` causes an error to be demoted to a warning even if the privilege or role named does not exist, or the statement attempts to assign it at the wrong level.

- If the `REVOKE` statement includes `IGNORE UNKNOWN USER`, the statement raises a warning for any target user or role named in the statement but not found; if no target named by the statement exists, `REVOKE` succeeds but has no actual effect. Otherwise, the statement executes as usual, and attempting to revoke a privilege not assigned to the target for whatever reason raises an error, as expected.

  *Example* (continuing from the previous example):

```
mysql> DROP USER IF EXISTS jerry@localhost;
Query OK, 0 rows affected (0.01 sec)

mysql> REVOKE SELECT ON test.t1 FROM jerry@localhost;
ERROR 1147 (42000): There is no such grant defined for user 'jerry' on host
'localhost' on table 't1'
mysql> REVOKE SELECT ON test.t1 FROM jerry@localhost IGNORE UNKNOWN USER;
Query OK, 0 rows affected, 1 warning (0.01 sec)

mysql> SHOW WARNINGS\G
*************************** 1. row ***************************
  Level: Warning
   Code: 3162
Message: Authorization ID jerry does not exist.
1 row in set (0.00 sec)
```

- The combination of `IF EXISTS` and `IGNORE UNKNOWN USER` means that `REVOKE` never raises an error for an unknown target user or role or for an unassigned or unavailable privilege, and the statement as whole in such cases succeeds; roles or privileges are removed from existing target users or roles whenever possible, and any revocation which is not possible raises a warning and executes as a NOOP.

*Example* (again continuing from example in the previous item):

```
# No such user, no such role
mysql> DROP ROLE IF EXISTS Bogus;
Query OK, 0 rows affected, 1 warning (0.02 sec)

mysql> SHOW WARNINGS;
+-------+------+----------------------------------------------+
| Level | Code | Message                                      |
+-------+------+----------------------------------------------+
| Note  | 3162 | Authorization ID 'Bogus'@'%' does not exist. |
+-------+------+----------------------------------------------+
1 row in set (0.00 sec)

# This statement attempts to revoke a nonexistent role from a nonexistent use
mysql> REVOKE Bogus ON test FROM jerry@localhost;
ERROR 3619 (HY000): Illegal privilege level specified for test

# The same, with IF EXISTS
mysql> REVOKE IF EXISTS Bogus ON test FROM jerry@localhost;
ERROR 1147 (42000): There is no such grant defined for user 'jerry' on host
'localhost' on table 'test'

# The same, with IGNORE UNKNOWN USER
mysql> REVOKE Bogus ON test FROM jerry@localhost IGNORE UNKNOWN USER;
ERROR 3619 (HY000): Illegal privilege level specified for test

# The same, with both options
mysql> REVOKE IF EXISTS Bogus ON test FROM jerry@localhost IGNORE UNKNOWN USE
Query OK, 0 rows affected, 2 warnings (0.01 sec)

mysql> SHOW WARNINGS;
+---------+------+--------------------------------------------+
| Level   | Code | Message                                    |
+---------+------+--------------------------------------------+
| Warning | 3619 | Illegal privilege level specified for test |
| Warning | 3162 | Authorization ID jerry does not exist.     |
+---------+------+--------------------------------------------+
2 rows in set (0.00 sec)
```

Roles named in the mandatory_roles system variable value cannot be revoked. When IF EXISTS and IGNORE UNKNOWN USER are used together in a statement that tries to remove a mandatory privilege, the error normally raised by attempting to do this is demoted to a warning; the statement executes successfully, but does not make any changes.

A revoked role immediately affects any user account from which it was revoked, such that within any current session for the account, its privileges are adjusted for the next statement executed.

Revoking a role revokes the role itself, not the privileges that it represents. Suppose that an account is granted a role that includes a given privilege, and is also granted the privilege explicitly or another role that includes the privilege. In this case, the account still possesses that privilege if the first role is revoked. For example, if an account is granted two roles that each include `SELECT`, the account still can select after either role is revoked.

`REVOKE ALL ON *.*` (at the global level) revokes all granted static global privileges and all granted dynamic privileges.

A revoked privilege that is granted but not known to the server is revoked with a warning. This situation can occur for dynamic privileges. For example, a dynamic privilege can be granted while the component that registers it is installed, but if that component is subsequently uninstalled, the privilege becomes unregistered, although accounts that possess the privilege still possess it and it can be revoked from them.

`REVOKE` removes privileges, but does not remove rows from the `mysql.user` system table. To remove a user account entirely, use `DROP USER`. See Section 13.7.1.5, “DROP USER Statement”.

If the grant tables hold privilege rows that contain mixed-case database or table names and the `lower_case_table_names` system variable is set to a nonzero value, `REVOKE` cannot be used to revoke these privileges. It is necessary in such cases to manipulate the grant tables directly. (`GRANT` does not create such rows when `lower_case_table_names` is set, but such rows might have been created prior to setting the variable. The `lower_case_table_names` setting can only be configured when initializing the server.)

When successfully executed from the **mysql** program, `REVOKE` responds with `Query OK, 0 rows affected`. To determine what privileges remain after the operation, use `SHOW GRANTS`. See Section 13.7.7.21, “SHOW GRANTS Statement”.