MySQLTUTORIAL

The Ultimate Guide To MySQL Roles By Examples

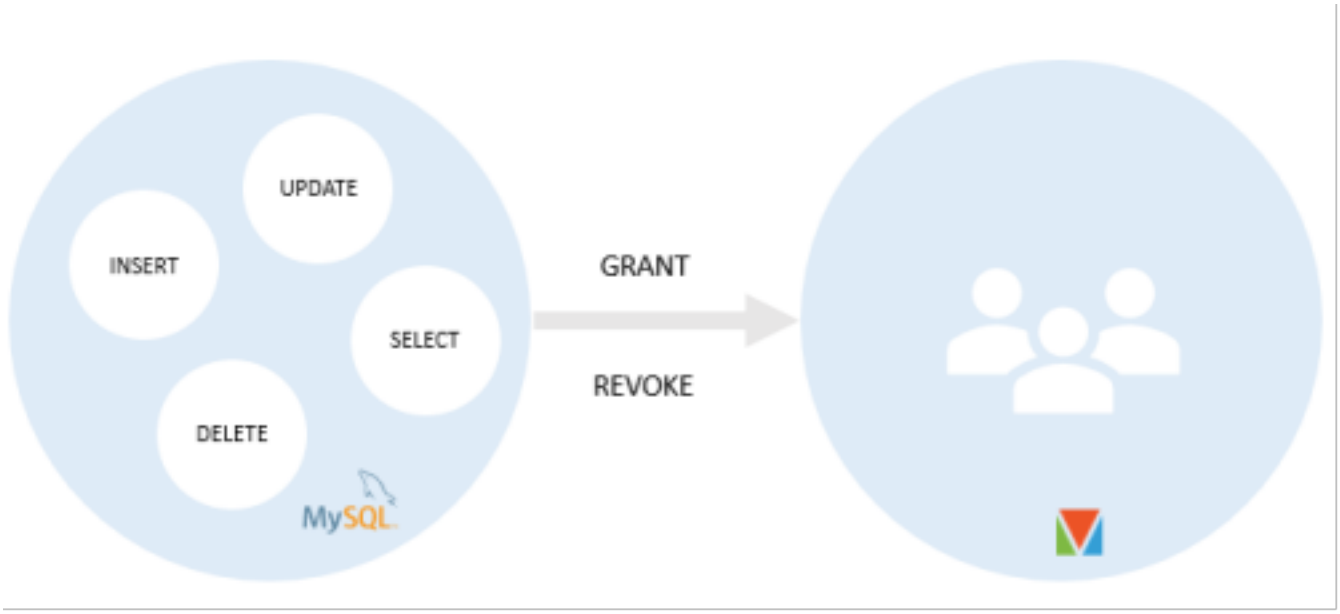
Summary: in this tutorial, you will learn how to use MySQL roles to simplify the privilege managements.

Introduction to MySQL roles

Typically, you have multiple users with the same set of privileges. Previously, the only way to grant (https://www.mysqltutorial.org/mysql-grant.aspx) and revoke (https://www.mysqltutorial.org/mysql-revoke.aspx) privileges to multiple users is to change the privileges of each user individually, which is time-consuming.

To make it easier, MySQL provided a new object called role. A role is a named collection of privileges.

Like user accounts, you can grant privileges to roles and revoke privileges from them.



If you want to grant the same set of privileges to multiple users, you follow these steps:

- First, create a new role.

- Second, grant privileges to the role.
- Third, grant the role to the users.

In case you want to change the privileges of the users, you need to change the privileges of the granted role only. The changes will take effect to all users to which the role granted.

MySQL role example

First, create a new database (https://www.mysqltutorial.org/mysql-create-table/) named CRM, which stands for customer relationship management.

```
CREATE DATABASE crm;
```

Next, use the `crm` database:

```
USE crm;
```

Then, create `customer` table inside the `CRM` database.

```
CREATE TABLE customers(
    id INT PRIMARY KEY AUTO_INCREMENT,
    first_name VARCHAR(255) NOT NULL,
    last_name VARCHAR(255) NOT NULL,
    phone VARCHAR(15) NOT NULL,
    email VARCHAR(255)
);
```

After that, insert data (https://www.mysqltutorial.org/mysql-insert-statement.aspx) into the `customers` table.

```
INSERT INTO customers(first_name,last_name,phone,email)
VALUES('John','Doe','(408)-987-7654','john.doe@mysqltutorial.org'),
      ('Lily','Bush','(408)-987-7985','lily.bush@mysqltutorial.org');
```

Finally, verify the insert by using the following `SELECT` (https://www.mysqltutorial.org/mysql-select-statement-query-data.aspx) statement:

```
SELECT * FROM customers;
```

	id	first_name	last_name	phone	email
▶	1	John	Doe	(408)-987-7654	john.doe@mysqltutorial.org
	2	Lily	Bush	(408)-987-7985	lily.bush@mysqltutorial.org

Creating roles

Suppose you develop an application that uses the `CRM` database. To interact with the `CRM` database, you need to create accounts for developers who need full access to the database. In addition, you need to create accounts for users who need only read access and others who need both read/write access.

To avoid granting privileges to each user account individually, you create a set of roles and grant the appropriate roles to each user account.

To create new roles, you use `CREATE ROLE` statement:

```
CREATE ROLE
    crm_dev,
    crm_read,
    crm_write;
```

The role name is similar to the user account that consists of two parts: the name and host:

```
role_name@host_name
```

If you omit the host part, it defaults to '%' that means any host.

Granting privileges to roles

To grant privileges to a role, you use `GRANT` statement. The following statement grants all

privileges to `crm_dev` role:

```
GRANT ALL
ON crm.*
TO crm_dev;
```

The following statement grants `SELECT` privilege to `crm_read` role:

```
GRANT SELECT
ON crm.*
TO crm_read;
```

The following statement grants `INSERT`, `UPDATE`, and `DELETE` privileges to `crm_write` role:

```
GRANT INSERT, UPDATE, DELETE
ON crm.*
TO crm_write;
```

Assigning roles to user accounts

Suppose you need one user account as the developer, one user account that can have read-only access and two user accounts that can have read/write access.

To create new users, you use `CREATE USER` (https://www.mysqltutorial.org/mysql-create-user.aspx) statements as follows:

```
-- developer user
CREATE USER crm_dev1@localhost IDENTIFIED BY 'Secure$1782';
-- read access user
CREATE USER crm_read1@localhost IDENTIFIED BY 'Secure$5432';
-- read/write users
CREATE USER crm_write1@localhost IDENTIFIED BY 'Secure$9075';
CREATE USER crm_write2@localhost IDENTIFIED BY 'Secure$3452';
```

To assign roles to users, you use `GRANT` statement.

The following statement grants the `crm_rev` role to the user account `crm_dev1@localhost`:

```
GRANT crm_dev
TO crm_dev1@localhost;
```

The following statement grants the `crm_read` role to the user account `crm_read1@localhost`:

```
GRANT crm_read
TO crm_read1@localhost;
```

The following statement grants the `crm_read` and `crm_write` roles to the user accounts `crm_write1@localhost` and `crm_write2@localhost`:

```
GRANT crm_read,
    crm_write
TO crm_write1@localhost,
    crm_write2@localhost;
```

To verify the role assignments, you use the `SHOW GRANTS` (https://www.mysqltutorial.org/mysql-adminsitration/mysql-show-grants/) statement as the following example:

```
SHOW GRANTS FOR crm_dev1@localhost;
```

The statement returned the following result set:

Grants for crm_dev1@localhost
GRANT USAGE ON *.* TO `crm_dev1`@`localhost`
GRANT `crm_dev`@`%` TO `crm_dev1`@`localhost`

As you can see, it just returned granted roles. To show the privileges that roles represent, you use the `USING` clause with the name of the granted roles as follows:

```
SHOW GRANTS
```

```
FOR crm_write1@localhost
USING crm_write;
```

The statement returns the following output:

	Grants for crm_write1@localhost
▶	GRANT USAGE ON *.* TO `crm_write1`@`localhost`
	GRANT INSERT, UPDATE, DELETE ON `crm`.* TO `crm_write1`@`localhost`
	GRANT `crm_read`@`%`,`crm_write`@`%` TO `crm_write1`@`localhost`

Setting default roles

Now if you connect to the MySQL using the `crm_read1` user account and try to access the `CRM` database:

```
>mysql -u crm_read1 -p
Enter password: ***********
mysql>USE crm;
```

The statement issued the following error message:

```
ERROR 1044 (42000): Access denied for user 'crm_read1'@'localhost' to database 'crm'
```

This is because when you granted roles to a user account, it did not automatically make the roles to become active when the user account connects to the database server.

If you invoke the `CURRENT_ROLE()` function, it will return `NONE`, meaning no active roles.

```
SELECT current_role();
```

Here is the output:

```
+----------------+
| current_role() |
+----------------+
```

```
| NONE           |
+----------------+
1 row in set (0.00 sec)
```

To specify which roles should be active each time a user account connects to the database server, you use the `SET DEFAULT ROLE` statement.

The following statement sets the default for the `crm_read1@localhost` account all its assigned roles.

```
SET DEFAULT ROLE ALL TO crm_read1@localhost;
```

Now, if you connect to the MySQL database server using the `crm_read1` user account and invoke the `CURRENT_ROLE()` function:

```
>mysql -u crm_read1 -p
Enter password: ***********
mysql> select current_role();
```

You will see the default roles for `crm_read1` user account.

```
+----------------+
| current_role() |
+----------------+
| `crm_read`@`%` |
+----------------+
1 row in set (0.00 sec)
```

You can test the privileges of `crm_read` account by switching the current database to `CRM`, executing a `SELECT` statement and a `DELETE` statement as follows:

```
mysql> use crm;
Database changed
mysql> SELECT COUNT(*) FROM customers;
```

```
+----------+
| COUNT(*) |
+----------+
|        2 |
+----------+
1 row in set (0.00 sec)

mysql> DELETE FROM customers;
ERROR 1142 (42000): DELETE command denied to user 'crm_read1'@'localhost' for table 'cu
```

It worked as expected. When we issued the `DELETE` statement, MySQL issued an error because `crm_read1` user account has only read access.

Setting active roles

A user account can modify the current user's effective privileges within the current session by specifying which granted role are active.

The following statement set the active role to `NONE`, meaning no active role.

```
SET ROLE NONE;
```

To set active roles to all granted role, you use:

```
SET ROLE ALL;
```

To set active roles to default roles that set by the `SET DEFAULT ROLE` statement, you use:

```
SET ROLE DEFAULT;
```

To set active named roles, you use:

```
SET ROLE
    granted_role_1
```

```
    [,granted_role_2, ...]
```

Revoking privileges from roles

To revoke privileges from a specific role, you use the `REVOKE` (https://www.mysqltutorial.org/mysql-revoke.aspx) statement. The `REVOKE` statement takes effect not only the role but also any account granted the role.

For example, to temporarily make all read/write users read-only, you change the `crm_write` role as follows:

```
REVOKE INSERT, UPDATE, DELETE
ON crm.*
FROM crm_write;
```

To restore the privileges, you need to re-grant them as follows:

```
GRANT INSERT, UPDATE, DELETE
ON crm.*
FOR crm_write;
```

Removing roles

To delete one or more roles, you use the `DROP ROLE` statement as follows:

```
DROP ROLE role_name[, role_name, ...];
```

Like the `REVOKE` statement, the `DROP ROLE` statement revokes roles from every user account to which they were granted.

For example, to remove the `crm_read` , `crm_write` roles, you use the following statement:

```
DROP ROLE crm_read, crm_write;
```

Copying privileges from a user account to another

MySQL treats user accounts like roles, therefore, you can grant a user account to another user account like granting a role to that user account. This allows you to copy privileges from a user to another user.

Suppose you need another developer account for the `CRM` database:

First, create the new user account:

```
CREATE USER crm_dev2@localhost
IDENTIFIED BY 'Secure$6275';
```

Second, copy privileges from the `crm_dev1` user account to `crm_dev2` user account as follows:

```
GRANT crm_dev1@localhost
TO crm_dev2@localhost;
```

In this tutorial, you have learned how to use MySQL roles to make it easier to manage privileges of user accounts.