MySQLTUTORIAL

The Ultimate Guide to MySQL Roles

Summary: in this tutorial, you will learn how to use MySQL roles to streamline privilege management.

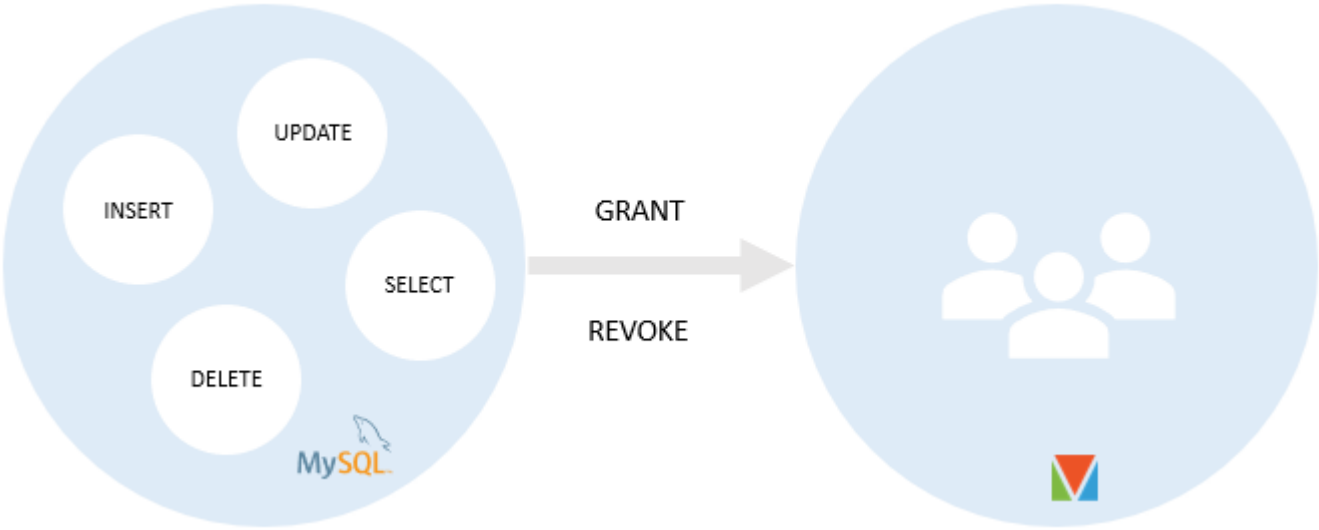
Introduction to MySQL roles

Typically, a MySQL database server may have multiple users with the same set of privileges.

Previously, the only way to grant and revoke the same privileges to multiple users was to modify the privileges of each user individually, which was time-consuming.

To streamline this process, MySQL introduced a new concept known as role. A role is essentially a named collection of privileges.

Just like user accounts, you can grant privileges to roles and revoke privileges from them. This feature helps you to simplify privilege management significantly.



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

- First, create a new role.
- Second, grant privileges to the role.
- Third, grant the role to the users.

When you want to change the privileges of the users, you only need to change the privileges of the granted role. These changes will apply to all users to whom the role has been granted.

MySQL roles example

First, create a new database named `crm`, which stands for Customer Relationship Management.

```
CREATE DATABASE crm;
```

Next, switch the current database to the `crm` database:

```
USE crm;
```

Then, create `customers` 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 two rows 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` 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 are developing an application that utilizes the `CRM` database. To interact with the `CRM` database, you need to create accounts for developers who need full access to the database. Additionally, you need to create accounts for users who only need read access and others who require both read and write access.

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

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

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

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

```
role_name@host_name
```

If you omit the host part, it defaults to '%' which 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 with read-only access, and two user accounts that can have read/write access.

To create new users, you use `CREATE USER` 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 the `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` statement as the following example:

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

The statement returned the following result set:

As you can see, it just returned granted roles. To display the privileges represented by these roles, you use the `USING` clause with the names of the granted roles as follows:

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

The statement returns the following output:

Setting default roles

If you connect to 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 'cr
```

This is because when you granted roles to a user account, it didn't automatically activate the roles when the user account connects to the database server.

If you invoke the `CURRENT_ROLE()` function, it will return `NONE`, indicating that no active roles are set:

```
SELECT current_role();
```

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 can use the `SET DEFAULT ROLE` statement.

The following statement sets the default role for the `crm_read1@localhost` account to include 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 then invoke the `CURRENT_ROLE()` function, you will see the default roles for `crm_read1` user account.

```
>mysql -u crm_read1 -p
Enter password: ***********
```

```
mysql> select current_role();
```

Output:

```
+----------------+
| 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` and 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 tabl
```

It worked as expected. When we issued the `DELETE` statement, MySQL issued an error because the `crm_read1` user account had 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 roles are active.

The following statement set the active role to `NONE`, indicating that no role is active:

```
SET ROLE NONE;
```

To set active roles to all granted roles, you use the following statement:

```
SET ROLE ALL;
```

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

```
SET ROLE DEFAULT;
```

To set named roles as active, you can use the following statement:

```
SET ROLE
    granted_role_1
    [,granted_role_2, ...]
```

Revoking privileges from roles

To revoke privileges from a specific role, you use the `REVOKE` statement. The `REVOKE` statement affects not only the role but also any account that has been granted the role.

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

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

To restore the privileges, you'll 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:

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

Similar to 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` and `crm_write` roles, you can use the following statement:

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

Copying privileges from one user account to another

MySQL treats user accounts like roles; as a result, 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 one user to another.

Suppose you want 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;
```

Summary

- Use roles to simplify and streamline privilege management within your database.
- Use the `CREATE ROLE` statement to create named collections of privileges.
- Use the `GRANT` statement to grant privileges to a role.
- Use the `SHOW GRANTS` statement to display the granted privileges of a role.

- Use the `REVOKE` statement to revoke privileges from a role.
- Use the `DROP ROLE` statement to delete a role.