MySQL Access Control

Access Control

Currently, users can only connect to MySQL on slade either from slade or from cook. Further, all users must be explicitely added to the list of users within MySQL. If you need to use MySQL on slade then please see a member of the MySQL admin team.

Once given access to MySQL, you can either connect using another program or application, such as R or Python, or you can connect from the command line. The examples here will illustrate a connection from the command line on slade.

Before logging in to mysql for the first time with your temporary password you should change this password. Although you can do this from the interactive mysql command line, the following method prevents your password from being shown in your terminal screen or recorded in any log. Also note that this should not be the same password as you use for your nomal IT/email account:

```
$ mysqladmin password -u test_usr123 -p
Enter password:
New password:
Confirm new password:
$
```

To start an interactive mysql session, give the user account you want to use and give the '-p' flag to get a prompt for your password. You should never put your password on the command line as this will be recorded in your history:

```
$ mysql –u test_usr123 –p
Enter password:
mysql>
```

Set which database or schema to use so that you can refer directly to the tables within the schema:

```
mysql> USE test_db;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql>
```

Roles

MySQL 8 introduces the concept of roles. Although roles have existed in other database systems for a while, MySQL had always relied on explicit user privileges. Briefly, a role captures a set of different privileges which can be assigned to users as a complete set rather than needing to be assigned to each user individually.

Setting Up and Assigning Roles

Since roles are listed alongside user accounts when showing users, the convention adopted here is to use the prefix 'role_' to differentiate the two types. Roles are created in a similar way to standard users, but without a password. (These are not login accounts so this is not a problem.)

```
$ mysql –u root –p
Enter password:
mysql> CREATE ROLE 'role_test_read';
Query OK, 0 rows affected (0.00 sec)
mysql> CREATE ROLE 'role_test_admin';
Query OK, 0 rows affected (0.00 sec)
mysql>
```

For these examples, a test database or schema 'test_db' has been created so that two roles can be assigned suitable privileges:

```
mysql> GRANT SELECT ON `test_db`.* TO 'role_test_read';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT ALL ON `test_db`.* TO 'role_test_admin';
Query OK, 0 rows affected (0.00 sec)
mysql>
```

User accounts can be assigned multiple roles:

```
mysql> GRANT 'role_test_read' TO 'test_usr123'@'%', 'test_usr789'@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> GRANT 'role_test_admin' TO 'test_usr789'@'%';
Query OK, 0 rows affected (0.00 sec)
mysql>
```

Note that if the privileges for a role are later updated then the user will see the updated privileges, not the privileges which were in force when the role was assigned to the user. By default, no roles will take effect when the user logs in. However, the role which should take effect by default can be set in the server configuration files, or 'on the fly':

```
mysql> SET DEFAULT ROLE role_test_read TO 'test_usr123'@'%', 'test_usr789'@'%';
Query OK, 0 rows affected (0.00 sec)
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
mysql> quit;
```

Although only two roles have been setup here, and both on a single database or schema, any number of roles could be defined with any combination of privileges for any number of databases. Ideally, the roles which are set up will form natural groups according to the access requirements for the databases.

Read Only Roles

Back to our example and user test_usr123. List the available tables and check that select works:

```
mysql> SHOW TABLES;
+------------------+
| Tables_in_test_db |
+------------------+
| 1_drug_names      |
+------------------+
1 row in set (0.00 sec)
mysql> SELECT * FROM 1_drug_names LIMIT 0,2;
+--------------+---------------+--------------+-------------+
| drug_name    | drug_type     | drug_class1  | drug_class2 |
+--------------+---------------+--------------+-------------+
           |   | insulin       | insulin      |
           |de | glibenclamide | sulfonylurea |
+--------------+---------------+--------------+-------------+
2 rows in set (0.00 sec)
mysql> SELECT drug_type, COUNT(drug_type) FROM 1_drug_names GROUP
BY drug_type LIMIT 0,3;
+------------------------------+------------------+
| drug_type                    | COUNT(drug_type) |
+------------------------------+------------------+
| insulin                      |               57 |
| glibenclamide                |               10 |
| acarbose                     |                2 |
3 rows in set (0.00 sec)
mysql>
```

However, creating a view fails:

```
mysql> CREATE VIEW num_drugs_by_type AS SELECT drug_type,
COUNT(drug_type) AS num_drugs FROM 1_drug_names GROUP BY drug_type;
ERROR 1142 (42000): CREATE VIEW command denied to user
'test_usr123'@'localhost' for table 'num_drugs_by_type'
mysql>
```

A look at the roles assigned to the account confirms why:

```
mysql> SHOW GRANTS FOR CURRENT_USER();
+-----------------------------------------------+
| Grants for test_usr123@%                      |
+-----------------------------------------------+
| GRANT USAGE ON *.* TO `test_usr123`@`%`       |
| GRANT SELECT ON `test_db`.* TO `test_usr123`@`%` |
| GRANT `role_test_read`@`%` TO `test_usr123`@`%` |
+-----------------------------------------------+
3 rows in set (0.00 sec)
mysql>
```

The user only has SELECT privileges for test_db. Note that if the user account name had been used instead of 'CURRENT_USER()' then not all of this information would be shown:

```
mysql> SHOW GRANTS FOR test_usr123;
+-------------------------------------------------+
| Grants for test_usr123@%                        |
+-------------------------------------------------+
| GRANT USAGE ON *.* TO `test_usr123`@`%`         |
| GRANT `role_test_read`@`%` TO `test_usr123`@`%` |
+-------------------------------------------------+
2 rows in set (0.00 sec)
mysql>
```

Admin Roles

Although a user account can be assigned an admin role, it will probably not be turned on by default. User 'test_usr789' has two roles assigned, a read role and an admin role:

```
mysql> SHOW GRANTS FOR CURRENT_USER();
+-----------------------------------------------------------------------+
| Grants for test_usr789@%                                              |
+-----------------------------------------------------------------------+
| GRANT USAGE ON *.* TO `test_usr789`@`%`                               |
| GRANT SELECT ON `test_db`.* TO `test_usr789`@`%`                      |
| GRANT `role_test_admin`@`%`,`role_test_read`@`%` TO `test_usr789`@`%` |
+-----------------------------------------------------------------------+
3 rows in set (0.00 sec)
mysql>
```

However, the second line indicates that the user only has SELECT privileges active. The active (current) role is the read role:

```
mysql> SELECT CURRENT_ROLE();
+----------------------+
| CURRENT_ROLE()       |
+----------------------+
| `role_test_read`@`%` |
+----------------------+
1 row in set (0.00 sec)
mysql>
```

An attempt at creating a view will fail:

```
mysql> CREATE VIEW num_drugs_by_type AS SELECT drug_type,
COUNT(drug_type) AS num_drugs FROM 1_drug_names GROUP BY drug_type;
ERROR 1142 (42000): CREATE VIEW command denied to user
'test_usr789'@'localhost' for table 'num_drugs_by_type'
mysql>
```

To enable the assigned admin privileges the user needs to switch to the admin role. Then the create view call succeeds:

```
mysql> SET ROLE 'role_test_admin';
Query OK, 0 rows affected (0.00 sec)
mysql> SELECT CURRENT_ROLE();
```

```
+-----------------------+
| CURRENT_ROLE()        |
+-----------------------+
| `role_test_admin`@`%` |
+-----------------------+
1 row in set (0.00 sec)
mysql> CREATE VIEW num_drugs_by_type AS SELECT drug_type,
COUNT(drug_type) AS num_drugs FROM 1_drug_names GROUP BY drug_type;
Query OK, 0 rows affected (0.01 sec)
mysql>
```

The user can either switch back to the 'read' role explicitely, or switch to the role which was set as default:

```
mysql> SET ROLE DEFAULT;
Query OK, 0 rows affected (0.00 sec)
mysql> SELECT CURRENT_ROLE();
+-----------------------+
| current_role()        |
+-----------------------+
| `role_test_read`@`%` |
+-----------------------+
1 row in set (0.00 sec)
mysql>
```

Best Practice

If you have been assigned a number of roles, including an admin role, you should be careful which role you are using. Always use the role which gives you just enough privileges for what you are doing. Do not automatically switch to an admin role every time you login. Rather, only switch when you need those extra privileges, and then switch back when less permissive privileges will do.