ROLES in MySQL

ROLES in MySQL

- A MySQL role is a named collection of privileges.
- Like user accounts, roles can have privileges granted to and revoked from them.
- A user account can be granted roles, which grants to the account the privileges associated with each role.
- This enables assignment of sets of privileges to accounts and provides a convenient alternative to granting individual privileges, both for conceptualizing desired privilege assignments and implementing them.
- CREATE ROLE and DROP ROLE create and remove roles.
- GRANT and REVOKE assign privileges to revoke privileges from user accounts and roles.
- SHOW GRANTS displays privilege and role assignments for user accounts and roles.
- The CURRENT_ROLE() function displays the active roles within the current session.

CREATE ROLE And GRANT Privileges

- An application uses a database named sakila .
- Associated with the application, there can be accounts for developers who create and maintain the application, and for users who interact with it.
- Developers need full access to the database. Some users need only read access, others need read/write access.
- To avoid granting privileges individually to possibly many user accounts, create roles as names for the required privilege sets. This makes it easy to grant the required privileges to user accounts, by granting the appropriate roles.
- To create the roles, use the CREATE ROLE statement:
- CREATE ROLE 'app_developer', 'app_read', 'app_write';
- GRANT ALL ON sakila.* TO 'app_developer';
- GRANT SELECT ON sakila.* TO 'app_read';
- GRANT INSERT, UPDATE, DELETE ON sakila.* TO 'app_write';

CREATE USER for ROLES in MySQL

- Now suppose that initially you require one developer account, two user accounts that need read-only access, and one user account that needs read/write access. Use CREATE USER to create the accounts:
- CREATE USER 'dev1'@'localhost' IDENTIFIED BY 'dev1p@ss';
- CREATE USER 'read_user1'@'localhost' IDENTIFIED BY 'read_user1p@ss';
- CREATE USER 'read_user2'@'localhost' IDENTIFIED BY 'read_user2p@ss';
- CREATE USER 'rw_user1'@'localhost' IDENTIFIED BY 'rw_user1p@ss';

Assign ROLE to USER

- To assign each user account its required privileges, you could use GRANT statements of the same form as just shown, but that requires enumerating individual privileges for each user. Instead, use an alternative GRANT syntax that permits granting roles rather than privileges:
- GRANT 'app_developer' TO 'dev1'@'localhost';
- GRANT 'app_read' TO 'read_user1'@'localhost', 'read_user2'@'localhost';
- GRANT 'app_read', 'app_write' TO 'rw_user1'@'localhost';

Checking ROLE Privileges

- To verify the privileges assigned to an account, use SHOW GRANTS.
- SHOW GRANTS FOR 'dev1'@'localhost';
- However, that shows each granted role without “expanding” it to the privileges the role represents. To show role privileges as well, add a USING clause naming the granted roles for which to display privileges:
- SHOW GRANTS FOR 'dev1'@'localhost' USING 'app_developer';

REVOKE ROLE OR ROLE Privileges

- Just as roles can be granted to an account, they can be revoked from an account:
- REVOKE role FROM user;
- REVOKE can also be applied to a role to modify the privileges granted to it.
- This affects not only the role itself, but any account granted that role. Suppose that you want to temporarily make all application users read only.
- To do this, use REVOKE to revoke the modification privileges from the app_write role:
- REVOKE INSERT, UPDATE, DELETE ON app_db.* FROM 'app_write';
- As it happens, that leaves the role with no privileges at all
- Because revoking privileges from a role affects the privileges for any user who is assigned the modified role, rw_user1 now has no table modification privileges (INSERT, UPDATE, and DELETE are no longer present):

DROP ROLE

- To drop roles, use DROP ROLE:
- DROP ROLE 'app_read', 'app_write';
- Dropping a role revokes it from every account to which it was granted.