Quick guide to roles and permissions

The following example illustrates how to manage roles and permissions.

You are a DBA and the **sqream** superuser. You wish to create the following sets of groups to which the **security officer** or the **department admins** can assign new users (note that the department admins and the security officer are not superusers):

- **security officer** – role for users who can change roles and permissions
- **database architect** – role for users who can create/modify table structure DDL
- **updater** - role for users who can modify tables data (DML)
- **reader** - role for users who can read data, execute functions, use views, etc.
- **udf author** - role for users who can create User Defined Functions

The example assumes the following:

- database is **MYDB**
- schema is **dwh_schema**

As the superuser, connect to any database and run the following:

1. Create the role **r_security_officer and give it the ability to login and use database MYDB.**

```
CREATE ROLE r_security_officer;
GRANT LOGIN to r_security_officer;
GRANT PASSWORD 'pass' to r_security_officer;
GRANT CONNECT ON DATABASE mydb to r_security_officer;
```

2. Create the role **r_database_architect** and give it the needed permissions in schema dwh_schema:

Permissions: **USAGE**, **CREATE** and **DDL**

```
CREATE ROLE r_database_architect;
GRANT connect ON DATABASE mydb TO r_database_architect;
GRANT usage,create,ddl ON SCHEMA dwh_schema TO r_database_architect;
```

3. Create the role **r_updater** and give it the needed permissions in schema dwh_ schema on tables created by the **r_database_architect** role group:

Permissions:**SELECT/INSERT/DELETE on ALL tables**

Run **ALTER DEFAULT PERMISSION** so that the permission will be granted for new tables in that schema as well.

```
CREATE ROLE r_updater;
GRANT connect ON DATABASE mydb TO r_updater;
GRANT usage ON SCHEMA dwh_schema TO r_updater;
GRANT SELECT,INSERT,DELETE ON ALL TABLES IN SCHEMA dwh_schema TO r_
updater;
ALTER DEFAULT PERMISSIONS FOR r_database_architect IN dwh_schema FOR
TABLES GRANT SELECT,INSERT,DELETE TO r_updater;
```

4. **Create the role r_udf_author** and give it the needed permissions.

Permissions:

- **SELECT** on **all the tables** in schema **dwh_schema**
- **CREATE FUNCTIONS** (UDF)

Run **ALTER DEFAULT PERMISSION** so that the permission will be granted for new tables in that schema as well.

```
CREATE ROLE r_udf_author;
GRANT connect ON DATABASE mydb TO r_udf_author;
GRANT usage ON SCHEMA dwh_schema TO r_udf_author;
GRANT CREATE FUNCTION ON DATABASE mydb TO r_udf_author;
GRANT SELECT ON ALL TABLES IN SCHEMA dwh_schema TO r_udf_author;
ALTER DEFAULT PERMISSIONS FOR r_database_architect IN dwh_schema FOR
TABLES GRANT SELECT TO r_udf_author;
```

5. Create the role **r_reader** and give it the needed permissions in schema dwh_schema on tables created by the **r_database_architect** role group:

Permissions:

- **SELECT** on **all the tables** in schema **dwh_schema**
- **EXECUTE ALL FUNCTIONS** (UDFs)

Run **ALTER DEFAULT PERMISSION** so that the permission will be granted for new tables in that schema as well.

```
CREATE ROLE r_reader;
GRANT connect ON DATABASE mydb TO r_reader;
GRANT usage ON SCHEMA dwh_schema TO r_reader;
GRANT SELECT ON ALL TABLES IN SCHEMA dwh_schema TO r_reader;
ALTER DEFAULT PERMISSIONS FOR r_database_architect IN dwh_schema FOR
TABLES GRANT SELECT TO r_reader;
GRANT EXECUTE ON ALL FUNCTIONS TO r_reader;
```

NOTE: GRANT EXECUTE FUCTION affects only existing functions.

6. Give the role **r_security_officer** the ability to grant all the new roles to others:

```
GRANT r_database_architect TO r_security_officer WITH ADMIN OPTION;
GRANT r_updater TO r_security_officer WITH ADMIN OPTION;
GRANT r_reader TO r_security_officer WITH ADMIN OPTION;
GRANT r_udf_author TO r_security_officer WITH ADMIN OPTION;
```

At this point, the security officer (who is not a superuser) can grant any of the roles they were defined as admin of to any new users created by the superuser (role with login/password).
As a superuser:

1. Create the roles user1, user2, user3 etc.

```
CREATE ROLE user1;
GRANT LOGIN to user1;
GRANT PASSWORD 'pass1' to user1;
CREATE ROLE user2;
GRANT LOGIN to user2;
GRANT PASSWORD 'pass2' to user2;
CREATE ROLE user3;
GRANT LOGIN to user3;
GRANT PASSWORD 'pass3' to user3;
CREATE ROLE user4;
GRANT LOGIN to user4;
GRANT PASSWORD 'pass4' to user4;
```

As the security officer:

```
GRANT r_database_architect TO user1;
GRANT r_reader TO user2;
GRANT r_udf_author TO user3;
GRANT r_updater TO user4;
```

Note that the 'with admin option' can be used in hierarchy. For example, if each department wishes to have its own dept_admin role, the superuser can create this role and grant it the required permissions with admin option so they can then assign the roles to users in their department.

Hierarchy example:

1. As superuser:

```
CREATE ROLE dept1_admin;
GRANT LOGIN TO dept1_admin;
GRANT PASSWORD 'password' TO dept1_admin;
GRANT CONNECT ON DATABASE mydb TO dept1_admin;
```

2. As the security officer or superuser:

```
GRANT r_reader TO dept1_admin WITH ADMIN OPTION;
```

3. As the dept1_admin:

```
GRANT r_reader TO user2;
```