How to Create a Snowflake Role Hierarchy

Snowflake is one of the most popular Data Warehouses, enabling businesses to easily store and process large amounts of data. It also offers a bunch of analytical solutions that are designed to be faster and much easier to use as compared to conventional tools. Snowflake is an incredibly common choice among larger businesses since it uses its own proprietary SQL Query Engine, allowing it to analyze data faster and also adding a range of innovative capabilities that you won't find with standard data storage providers. This article will discuss how to create a Snowflake Role Hierarchy.

Snowflake Data Cloud also offers a pretty fantastic model for assigning and defining hierarchical roles. This is used as the standard throughout the Snowflake Data Cloud and makes it easy for businesses to assign roles to different users. Roles play an integral role in managing data, allowing administrators to manage access throughout the company's Snowflake account. Creating a Snowflake Role Hierarchy is fairly straightforward. Lets discuss that in detail in the later sections.

Table of Contents

- Conclusion

Snowflake is a Cloud Data Warehousing solution provided as a SaaS offering. It is built on Amazon Web Service, Microsoft Azure, or Google Cloud infrastructure that provides an unbounded platform for storing and retrieving data. Snowflake Data Warehouse uses a different proprietary SQL Database Engine with a unique architecture designed for the cloud.

The architecture of Snowflake separates its "**Compute**" and "**Storage**" units, thereby scaling differently. This allows the customers to use and pay for both services independently. It means organizations that have high storage demands but less need for CPU cycles, or vice versa, do not have to pay for an integrated bundle that requires payment for both, making it very attractive to companies. Like other popular Data Warehouses, it also uses Columnar Storage for parallel query execution.

With Snowflake, there is no hardware or software to select, install, configure, or manage, therefore, making it ideal for organizations that do not want to have dedicated resources for setup, maintenance, and support for in-house servers. Snowflake security and sharing functionalities make it easy for organizations to quickly share and secure data in real-time using any available ETL solution. Snowflake's architecture allows flexibility with **Big Data**. Snowflake is known for its scalability and relative ease of use when compared to other Data Warehouses in the market.

Now that you're familiar with Snowflake, let's dive straight into Snowflake Role Hierarchy.

The Importance of Roles in Snowflake

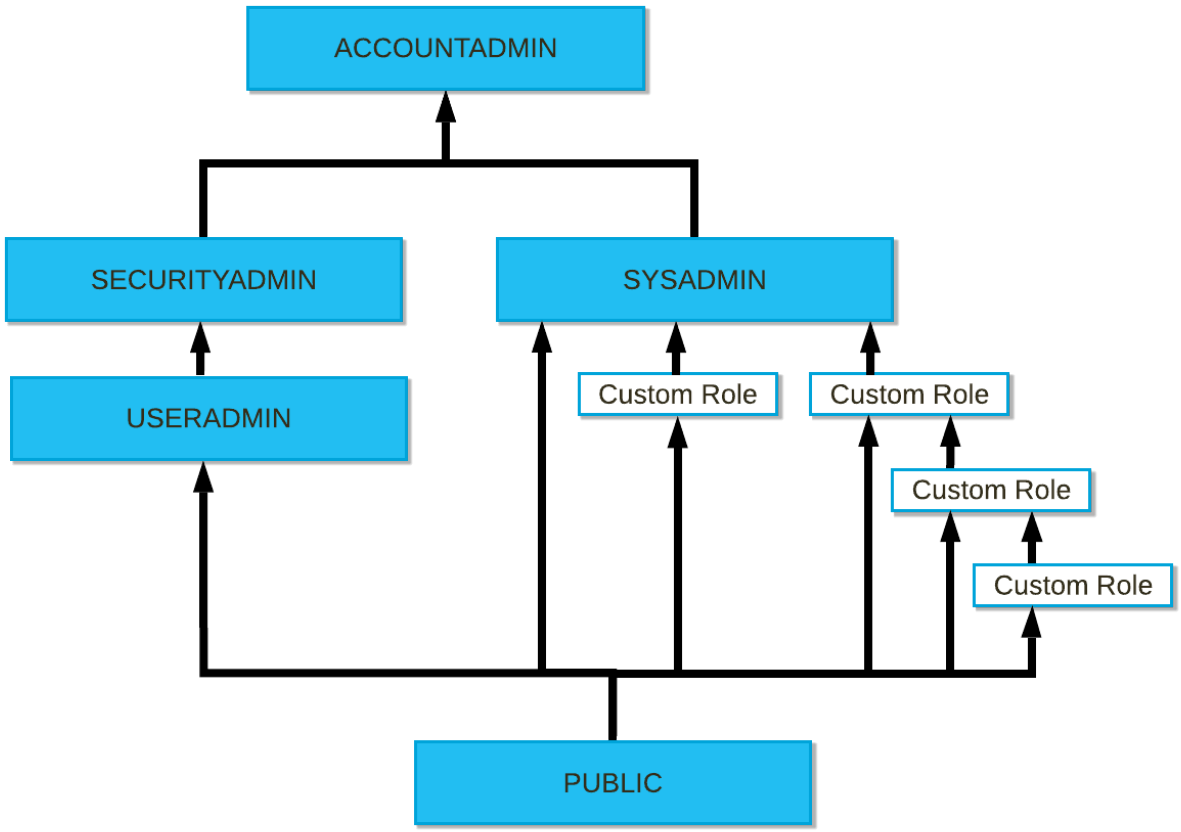

Image Source: www.docs.snowflake.com

Before creating a Snowflake Role Hierarchy, it's important to understand why roles are so important. Defining roles allows you to easily manage access to the data stored in your Snowflake account. Instead of allowing every user to access and manipulate the data, only specified individuals with access can do so.

When you create a Snowflake Role Hierarchy, you can re-use roles and grant the same to different users. This means you don't have to make as many SQL

statements otherwise. It's also much easier to simply revert and modify SQL statements to change access rights that you've given to different users.

When you grant statements to each user, you'll have to create one statement per user, instead of creating one statement per role. This is problematic since it greatly enhances security risks and exposes your data to external attacks. You can also grant roles to other roles, which allows each one to automatically inherit the other's privileges.

This reduces the need for making excess grants, and thus reduces the margin for error. More importantly, it allows users to select the least powerful role for a task, so they don't accidentally modify or corrupt data when they primarily only want to read it.

Different Types of Roles in Snowflake

The Snowflake Data Cloud gives you different types of roles, including both system and custom roles. There are five system roles that you can assign, each of which is described below.

- PUBLIC
- SYSADMIN
- USERADMIN
- SECURITYADMIN
- ACCOUNTADMIN

PUBLIC

This is the default role assigned to every user who makes an account. Anything assigned to this role is generally accessible by everyone.

SYSADMIN

This role focuses primarily on managing system objects. Snowflake recommends that each custom role is assigned to SYSADMIN as it allows users to grant similar privileges to all custom roles too.

USERADMIN

This role primarily focuses on user management. The permissions for this role allow assigned individuals to create and modify other users and assign different roles to them.

SECURITYADMIN

The SECURITYADMIN role allows assignees to manage object grants throughout the platform. As a result, the role comes with a MANAGE GRANTS privilege by default. More importantly, the USERADMIN role is also assigned to the SECURITYADMIN automatically.

ACCOUNTADMIN

Finally, there's the ACCOUNTADMIN role, which is the highest role available in Snowflake. Since it covers both SYSADMIN and SECURITYADMIN and all underlying privileges, it is important for managers to limit its access as much as possible.

Creating a Snowflake Role Hierarchy

It's now time to create a Snowflake Role Hierarchy. If you already have a Database and a schema set up, you can simply partition the privileges to different users who will be working on the project. That's all you have to do.

You also have to decide the access level you want to offer to each user, as different users will require varying levels of access.

For instance, an Engineer or a Data Analyst may require access to tables so they can easily insert and manage data, whereas a Junior Intern may only require access to read data. An Administrator, on the other hand, should have access to both and should get additional privileges for project management.

In the following commands, you are simply creating roles for a project entitled "**Example Project**." Once you have created the Database and the schemas, the next step is to create access levels for each. You can use the following commands:

```
USE ROLE SECURITYADMIN;
```

```
CREATE role example_administrator
```

```
CREATE role example_analyst
```

```
CREATE role example_engineer
```

This will allow you to create roles for different access levels. Now, the next step is to create a hierarchy by linking the roles so that each role that's superlative to another will automatically have the same privileges as the ones that rank below. The Analyst Role, for example, will have all the privileges of the Engineer, whereas the Administrator should have access to both.

```
USE ROLE SECURITYADMIN;
```

```
GRANT role example_engineer TO role example_analyst

GRANT role example_analyst TO role example_administrator
```

Applying Privileges for Different Roles in Snowflake

Now that you have linked the roles, it's time to start applying the privileges to the roles.

- The Engineer
- The Analyst
- The Administrator

The Engineer

The Engineer generally requires reading privileges in most cases, so they'll require the following grants:

- USAGE for the database
- SELECT for all the tables or other informative data
- USAGE for running queries on individual datasets

The commands are as follows:

```
USE ROLE SECURITYADMIN;

GRANT USAGE ON DATABASE example TO role example_engineer;

GRANT USAGE ON SCHEMA example.dataset TO role example_engineer;

GRANT SELECT on ALL FUTURE TABLES IN SCHEMA example.dataset TO
ROLE example_engineer;
```

```
GRANT USAGE ON STORAGE example_analysts TO role
example_engineer;
```

The Analyst

The role of the Analyst involves adding new data from time to time and analyzing it. As a result, they'll need the following permissions:

- USAGE on the datasets for ETL
- INSERT for adding new views or tables
- CREATE VIEW and CREATE TABLE privileges

```
GRANT INSERT ON ALL FUTURE TABLES IN SCHEMA example.dataset TO
role example_analyst;

GRANT USAGE on DATASET example_etl TO role example_analyst;

GRANT CREATE TABLE, CREATE VIEW ON SCHEMA example_dataset TO
role example_analyst;
```

The Administrator

This is the role that supersedes all the others. The Administrator automatically inherits privileges from the Analyst and the Engineer, allowing them to create multiple stages or sections to ingest data from different sources. As a result, they will require privileges from both.

```
USE ROLE SECURITYADMIN;

GRANT CREATE STAGE ON SCHEMA example_dataset TO ROLE
example_administrator;
```

Granting Roles to New Users

Now that you have set up the role hierarchy, the next step is to create a few users and then grant them the specific roles. These users can get to work and start utilizing the data in the datasets. To do this, you have to run the following commands:

```
USE ROLE USERADMIN;

CREATE USER user_1;

CREATE USER user_2;

CREATE USER user_3;


USE ROLE SECURITYADMIN;


GRANT ROLE example_administrator TO USER user_1;

GRANT ROLE example_analyst TO USER user_2;

GRANT ROLE example_engineer TO USER user_3;
```

That’s it, creating a Snowflake Role Hierarchy is as simple as that.