**Chapter 4. Account Access and Security**

It starts with a description of account identifiers and account parameters. It introduces network policies that allow or deny access to Snowflake from specific IP addresses. Next, it provides an overview of user authentication methods. Finally, it dives into more details about granting privileges to users through roles.

## Account Identifiers

* In URLs for accessing the Snowflake web interface
* In SnowSQL and other connectors or drivers for connecting to Snowflake
* In third-party applications and services that interact with Snowflake
* In API calls with external systems
* In Snowflake features such as secure data sharing or database replication

The URL of a Snowflake account has the following format:

**account\_identifier.snowflakecomputing.com**

Snowflake supports two formats of account identifiers:

* Legacy format
* Account name within an organization

## Legacy Format

You can identify a Snowflake account by a combination of an account locator and optionally by the cloud region and the cloud name. An account locator is a random string of characters and numbers that is assigned by Snowflake when the account is created. It cannot be changed afterward.

**account\_locator.cloud\_region.cloud**

**account\_locator.cloud\_region**

**account\_locator**

Using the account name within an organization is now the preferred method.

## Account Name Within an Organization

An organization is a Snowflake object associated with a business entity. The organization name is chosen when the first Snowflake account is provisioned for the organization. The organization name must be unique with respect to all other organizations in Snowflake.

The account identifier for an account in an organization is made up of the organization name and the account name. The delimiter between the organization name and the account name may be a hyphen, an underscore, or a dot, depending on where and how the identifier is used:

* When the account identifier is used in URLs and other general-purpose scenarios, a hyphen serves as the delimiter, for example: organization\_name-account\_name
* When the account identifier is used in features where hyphens are not supported in URLs, an underscore serves as the delimiter, for example: organization\_name\_account\_name
* When the account identifier is used in SQL commands, a dot serves as the delimiter, for example: organization\_name.account\_name

## Parameter Management

* Account parameters that affect the entire account
* Session parameters that affect users and sessions
* Object parameters that affect specific objects, such as warehouses, databases, schemas, and tables

All parameters are initially set with default values, which can be subsequently overridden at each of the levels. A Snowflake administrator with sufficient privileges can override parameter values at the account level. Users can override the parameter values at their own session level. Object parameter values can be overridden by users who have sufficient privileges on the object.

To view a list of the parameters and their values in your account, as well as their default values, execute the SHOW PARAMETERS command with the ACCOUNT keyword:

**show parameters in account;**

One parameter that you might want to review is the TIMEZONE parameter, which is initially set as America/Los Angeles. You can override it to your local time zone if it is different. You can do this at the account level using the ACCOUNTADMIN role or at the session level using your current role.

**alter session set TIMEZONE = 'Europe/London';**

## Network Security and Policies

A security administrator can create a network policy to allow or deny access to specified IP addresses. This can be accomplished by creating IP allowed lists or IP blocked lists or both.

To create network policies, a user must have any of the following privileges:

* A SECURITYADMIN role
* A role higher than SECURITYADMIN
* A role with the global CREATE NETWORK POLICY privilege

Network policies can be assigned to a Snowflake account or to a particular user. Although multiple network policies can be created, only a single network policy can be assigned to an account or a user at a time.

In addition, when a network policy is assigned to an account or a user while they are already connected to Snowflake, they cannot execute further statements if their IP address doesn’t comply with the network policy.

Note When a network policy includes values for the allowed IP address list and the blocked IP address list, Snowflake checks the blocked IP address list first.

As a Snowflake user without administrative privileges, you will not be able to create, assign, or manage network policies. But you can view whether a network policy is set on your account using the SHOW PARAMETERS command:

**show parameters like 'network\_policy' in account;**

You can also view whether a network policy is set at your user level using the following command, where you provide your username that you use to log in to Snowflake:

**show parameters like 'network\_policy' in user <yourusername>;**

When even more security is needed, Snowflake Business Critical edition (or higher) supports the AWS PrivateLink service. This is an AWS service for creating private VPC (virtual private cloud) endpoints that allow direct, secure connectivity between AWS VPCs and the Snowflake VPC without traversing the public Internet. To take advantage of this functionality, your Snowflake account must be in the same AWS region as the AWS VPCs.

## User Authentication

* Multi-factor authentication (MFA)
* Federated authentication
* Single sign-on (SSO)

Snowflake administrators who have the ACCOUNTADMIN, SYSADMIN, or SECURITYADMIN role set up federated authentication.

## Multi-Factor Authentication

MFA is provided by Duo Security and is included as a Snowflake service. You only have to install the Duo Mobile application on one of the supported platforms, including iOS, Android, Windows, and others.

You will not be able to unenroll from MFA after you have enrolled. To unenroll, you must ask your Snowflake administrator to unenroll you. If you are using the 30-day free Snowflake trial account, you can unroll the FIRSTUSER from MFA by logging into your Snowflake account using your initial credentials (the same credentials that were used to create FIRSTUSER) and executing this statement:

**alter user FirstUser set disable\_mfa = true;**

Alternatively, if you are experiencing any technical issues with MFA, you can ask your Snowflake administrator to allow you to temporarily bypass MFA and log in directly for a limited time. For example, to bypass MFA for ten minutes, you can execute the following statement:

**alter user FirstUser set mins\_to\_bypass\_mfa = 10;**

## Federated Authentication

In a federated environment, user authentication is separated from user access using an entity that provides independent authentication of user credentials. Snowflake supports authentication through an external identity provider. Once users are authenticated by the identity provider, they can securely initiate Snowflake sessions without having to log in to Snowflake each time.

Snowflake supports most SAML 2.0-compliant vendors as identity providers, including:

* Google G Suite
* Microsoft AzureAD (Active Directory)
* OneLogin
* Ping Identity PingOne

The following vendors provide native Snowflake support for federated authentication:

* Okta
* Microsoft ADFS (Active Directory Federation Services)

For example, when federated authentication is configured in a Snowflake account, the login screen in the Snowflake web interface looks similar to what is shown in Figure [4-6](https://learning.oreilly.com/library/view/snowpro-core-certification/9781484290781/html/536134_1_En_4_Chapter.xhtml#Fig6). There is an additional option, called Sign in Using AzureAD, because federated authentication was set up using Microsoft AzureAD as the identity provider.

* If the user has already been authenticated by the identity provider, they are immediately granted access to Snowflake.
* If the user has not yet been authenticated by the identity provider, they are taken to the identity provider interface where they authenticate, after which they are granted access to Snowflake.

Depending on the identity provider, users have two possibilities to log out of Snowflake:

* **Standard logout.** A user must log out of both Snowflake and the identity provider. When a user logs out of just a single Snowflake session, any of their other Snowflake sessions remain connected. They can still initiate new Snowflake sessions if they remain logged in the identity provider.
* **Global logout.** When a user logs out of the identity provider, all of their Snowflake sessions log out as well. With a web-based identity provider such as Okta, closing the browser does not necessarily end the identity provider session. As long as the identity provider session is active, the user can still access Snowflake.

With respect to session timeout, there are two scenarios depending on which session timed out:

* **Snowflake timeout.** When a user is logged into Snowflake via an identity provider and their Snowflake session expires due to inactivity, they will be asked to authenticate via the identity provider again to continue to work in the expired session.
* **Identity provider timeout.** When a user’s session in the identity provider reaches the defined timeout period, their current Snowflake sessions will not be affected. However, if the user wants to initiate additional Snowflake sessions, they will have to log into the identity provider again.

## Session Policies

The default timeout period of a Snowflake session is four hours. This timeout period can be overridden by creating a session policy that defines the idle session timeout period in minutes. Similarly, as with network policies, session policies can be assigned to an account or to a user. A session policy assigned to a user takes precedence over a session policy assigned to an account.

To create a session policy, use the CREATE SESSION POLICY command. This command takes two optional parameters that can be set depending on whether the session policy will apply to the web user interface or to any of the Snowflake clients:

* SESSION\_UI\_IDLE\_TIMEOUT\_MINS. Applies to the web user interface.
* SESSION\_IDLE\_TIMEOUT\_MINS. Applies to Snowflake clients.

Both parameters can be set to a value between 5 and 240 minutes. Whenever a session policy has not been set, Snowflake will use the default value of 240 minutes (4 hours). Once the session expires, the user will have to authenticate again.

## Access Control (can be defined as either discretionary or role-based)

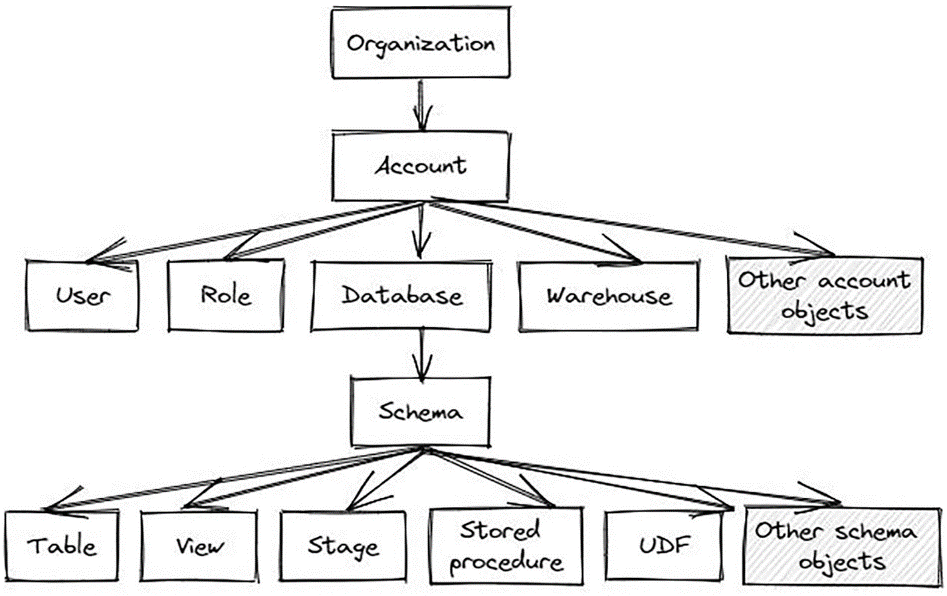
Administrators can give detailed access control that allows or restricts access to objects for specific users or roles. They can also define what operations can be performed on objects to which users have access and they can define which users are allowed to create, modify, and grant access control policies.

* **Securable object.** Any Snowflake object, such as a database, schema, table, view, stored procedure, and so on, to which access can be granted.
* **User.** Either a person or a client program that can log into Snowflake and to whom access to securable objects can be granted.
* **Role.** A set of privileges can be granted to a role and, in turn, a role can be assigned to a user or to another role.
* **Privilege.** One or more allowed types of access to a securable objects, such as the privilege to read or the privilege to modify and so on.

## Securable Objects

Snowflake securable objects are organized as a hierarchy of containers. Hierarchy of Snowflake objects and containers:

A securable object is owned by the role that was used to create the object. This role has the OWNERSHIP privilege on the object.

* **Regular schema.** The role that owns the objects has all privileges related to the object, including the ability to grant or revoke privileges on the object to other users or roles as well as to transfer ownership.
* **Managed access schema.** All objects in a managed access schema are owned by the schema itself. There are no individual object owners. Privileges on objects in a managed access schema can be granted either by the schema owner or a role that has the MANAGE GRANTS privilege.

A user administrator is a user who has one of the following privileges:

* A USERADMIN role
* A role higher than USERADMIN
* A role with the global CREATE USER privilege

**Note:** When you assign a default role to a user, this does not immediately grant them the role. You must also explicitly grant the role to the user:

**CREATE USER “AARON”;**

**ALTER USER “AARON” SET DEFAULT\_ROLE=”ML\_ROLE”;**

**GRANT ROLE ML\_ROLE TO USER "AARON";**

**DESC USER “AARON”;**

**create user SecondUser**

**first\_name = 'Second'**

**last\_name = 'User'**

**display\_name = 'Second User'**

**password = 'Password456!'**

**email = 'seconduser@email.com';**

## Roles

Privileges on securable objects can be granted to and revoked from roles. Roles are then assigned to users to control what actions they may perform on the securable objects.

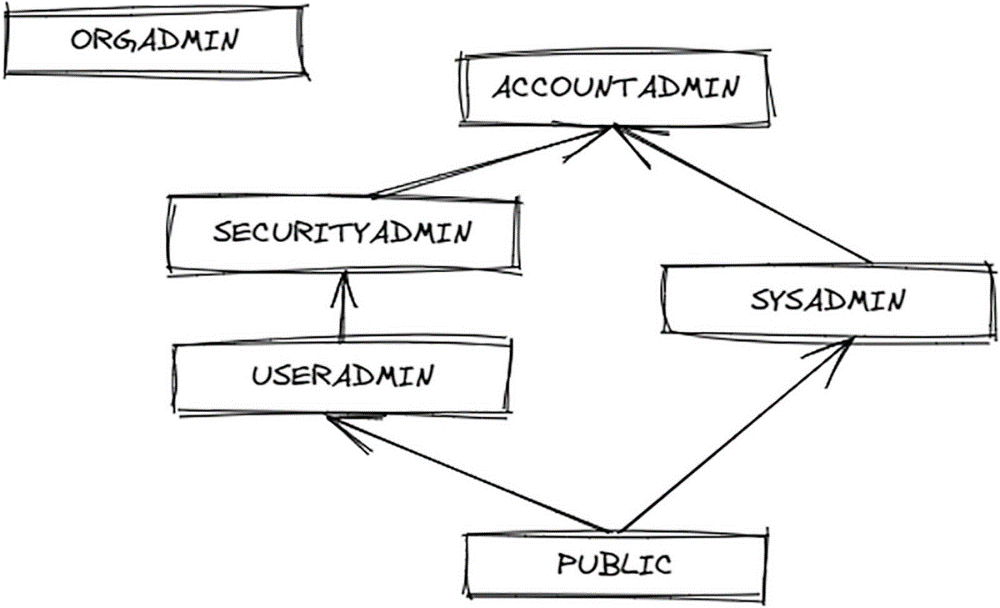
A hierarchy of roles can be created by granting roles to other roles. A role that is at a higher level of the hierarchy inherits all privileges granted to roles at a lower level of the hierarchy. Snowflake provides a set of system-defined roles as well as functionality for defining a hierarchy of custom roles.

## System-Defined Roles

These built-in roles cannot be dropped and the privileges that are granted to these roles by Snowflake cannot be revoked.

With the exception of ORGADMIN, the system-defined roles are structured according to a hierarchy, in which each role inherits the privileges of all the roles below the current role. ORGADMIN is a separate system role that is required to perform tasks at the organization level.

The ORGADMIN or the Organization Administrator role is used to perform tasks at the organization level. This role can create and manage accounts within the organization, view usage information across the organization, and other related tasks.

The ACCOUNTADMIN or the Account Administrator role is the most powerful role in a Snowflake account. It is used to perform account administration tasks, set account parameters, view account usage information, and other related tasks.

The SECURITYADMIN or the Security Administrator role is used to create, monitor, and manage users and roles. This role has the MANAGE GRANTS privilege, which allows it to grant or revoke any privilege.

The USERADMIN or the User and Role Administrator role is used for managing users and roles. This role has the CREATE USER and CREATE ROLE privileges, which allow it to create users and roles in the account.

The SYSADMIN or the System Administrator role is used to create virtual warehouses, databases, schemas, tables, and other objects in an account.

The PUBLIC role is granted to every user and every role in an account. Any securable objects owned by the PUBLIC role are accessible to every other user and role in the account.

## Custom Roles

The USERADMIN role or any other role with the CREATE ROLE privilege can create custom roles. Once a role is defined, it must be granted to a user or to another role in order to take effect.

Snowflake recommends creating a hierarchy of custom roles and granting the topmost custom role to the SYSADMIN system-defined role. This will allow system administrators to access and manage all objects in the account.

## Granting and Revoking Privileges

In addition to granting and revoking privileges on existing objects in a schema, Snowflake supports defining privileges on future objects that will be created in a schema.

Privileges are managed using the GRANT and REVOKE commands. Let’s walk through an example. You must be logged into Snowflake with a user who has sufficient privileges to create users and roles, as well as the ability to grant roles.

**use role USERADMIN;**

**create role ANALYST;**

Then switch to the SYSADMIN role, which you will use to create a database named REPORTING, a schema named MARKETDATA, and a table named CUSTOMER:

**use role SYSADMIN;**

**create database REPORTING;**

**create schema MARKETDATA;**

**create table CUSTOMER (**

**c\_custkey NUMBER(38,0) NOT NULL,**

**c\_name VARCHAR(40),**

**c\_address VARCHAR(255),**

**c\_country VARCHAR(3),**

**c\_mktsegment VARCHAR(255));**

Now you will grant read-only access on the CUSTOMER table to the ANALYST role. In addition to granting access on the table itself, access must also be granted to securable objects in the hierarchy above the current object. This means that privileges that allow the role to use the schema and the database where the table is contained must also be granted to the role.

You are still using the SYSADMIN role because this role owns the securable objects and is allowed to grant privileges:

**grant select on table CUSTOMER to role ANALYST;**

**grant usage on schema MARKETDATA to role ANALYST;**

**grant usage on database REPORTING to role ANALYST;**

Finally, switching back to the USERADMIN role, you can grant the newly created custom role ANALYST to the SECONDUSER user that you created previously:

**use role USERADMIN;**

**grant role ANALYST to user SECONDUSER;**

**ALTER USER SECONDUSER SET DEFAULT\_ROLE=ANALYST;**

Let’s create a second role named DATAENGINEER that will have full access to table CUSTOMER and grant this role to SECONDUSER as well.

**use role USERADMIN;**

**create role DATAENGINEER;**

**use role SYSADMIN;**

**grant all on table CUSTOMER to role DATAENGINEER;**

**grant usage on schema MARKETDATA to role DATAENGINEER;**

**grant usage on database REPORTING to role DATAENGINEER;**

**use role USERADMIN;**

**grant role DATAENGINEER to user SECONDUSER;**

## Role Hierarchy and Privilege Inheritance

The current role in the session is determined by Snowflake according to the following order of precedence:

* If the connection information included the name of a role and that role has been granted to the user, this role is used in the session as the current role.
* If the connection information did not include the name of a role and the user has a default role assigned, the default role is used.
* If the connection information did not include the name of a role and the user has no default role assigned, the PUBLIC role is used.

The current role in a session is also referred to as the primary role. In addition to the primary role, a user can also activate one or more secondary roles in the session by executing the USE SECONDARY ROLES command. A default set of secondary roles can also be defined for a user, similarly as their default primary role using the DEFAULT\_SECONDARY\_ROLES keyword with the CREATE USER or ALTER USER commands.

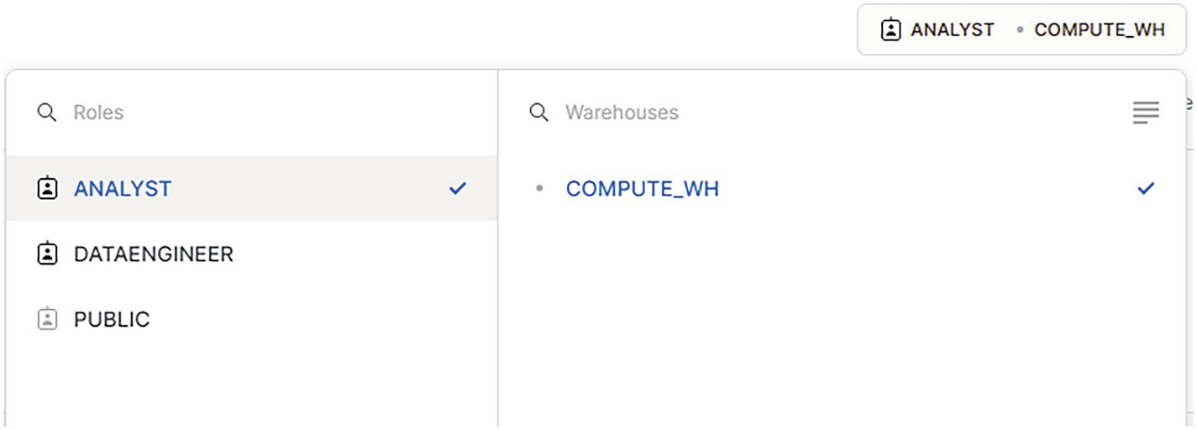
When a user has more than one role activated in a session, they can execute SQL statements using the combined privileges of the primary and all secondary roles. These are applied differently whether the user is creating objects or performing other actions on objects:

* The primary role is used to determine whether the user has privileges to execute CREATE statements for Snowflake objects. This role will also be set as the owner of any object that the user creates.
* To execute SQL statements other than CREATE, Snowflake will allow execution if the user has sufficient privileges granted to the primary or to any of their secondary roles.

Secondary roles are useful in large organizations that have a large number of defined custom roles, such as separate sets of roles for each database. When a developer needs privileges to execute SQL statements that access more than one database in the same query, for example to copy data from one database to another, they can activate a primary and a secondary role, one that grants them access to each of the databases in their current session.

## Setting Primary and Secondary Roles

A user can set their primary role in a session using the Roles and Warehouses drop-down list at the top right in Snowsight.



A user can set their primary role by executing the USE ROLE command. For example, when logged into Snowflake as SECONDUSER, you use the following command to set the primary role to ANALYST:

**use role ANALYST;**

To set one or more secondary roles, the USE SECONDARY ROLES command can be executed. For example, to set the secondary role to DATAENGINEER, use this command:

**use secondary roles DATAENGINEER;**

A user can view their current secondary roles using the CURRENT\_SECONDARY\_ROLES function:

**select current\_secondary\_roles();**

## Summary

This chapter addressed various Snowflake security topics. It described account identifiers and account parameters. It introduced network policies that allow or deny access to Snowflake from specific IP addresses. It also provided an overview of user authentication methods. Finally, it discussed granting privileges to users through roles.