Snowflake Roles & Access Control

Vikram Singh Walia - Enterprise SE

June 2023

Snowflake's Roles & Access Controls

Overview

Snowflake access control system is meant to make sure that only authorized users and applications can access data and perform actions in the Snowflake environment.

Snowflake uses a combination of **Role-Based Access Control (RBAC)** and **Discretionary Access Control (DAC)** to provide a flexible and granular access control.

Elements of Security:

Securable object:

- It is an entity that can be secured and to which access can be granted.
- Access to a securable object is, by default, denied unless allowed by a grant.
- Examples of securable objects are databases, schemas, tables, views, and functions.

Roles:

- It is an entity to which privileges can be granted.
- Roles are used to manage and control access to securable objects in Snowflake.
- Roles are assigned to users, and a user can have multiple roles.
- Roles can also be assigned to other roles, creating a role hierarchy that enables more granular level control.

Privilege*:*

- It is a defined level of access to a securable object.
- Privileges are used to control the granularity of access granted.
- Multiple distinct privileges can be used to control access to a securable object, such as the privileges of selecting, updating or deleting from a table.

User*:*

- It is an entity to which you can define privileges.
- Users are granted privileges through roles assigned to them.
- Users can be assigned to one or more roles, granting them access to securable objects in Snowflake.

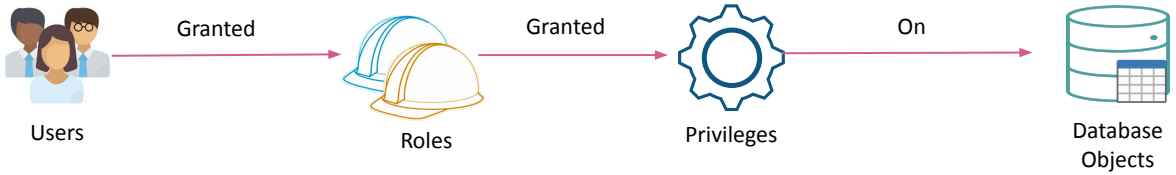
Snowflake's Roles & Access Controls

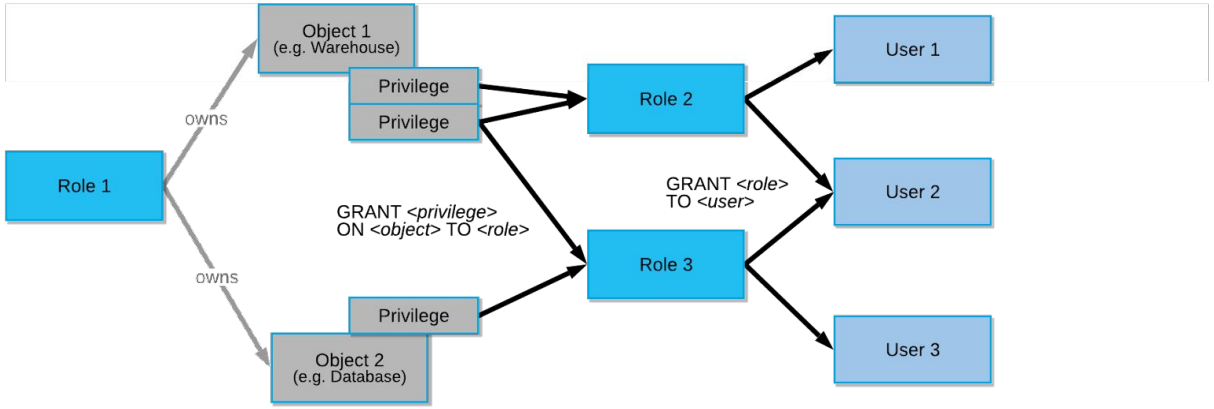
Understanding Access Control and its Relationships

Key points to understand the Access control relationships in Snowflake:

- Access to securable objects is allowed via privileges assigned to roles

  (e.g., databases, schemas, tables, or views)
- Roles can be assigned to other roles or individual users
- Each securable object in Snowflake has an owner who can grant access to other roles.
- Snowflake model differs from a user-based access control model, where rights and privileges are assigned to each user or group of users.
- Roles are granted to users; privileges are granted to those roles on objects, providing access to the user.

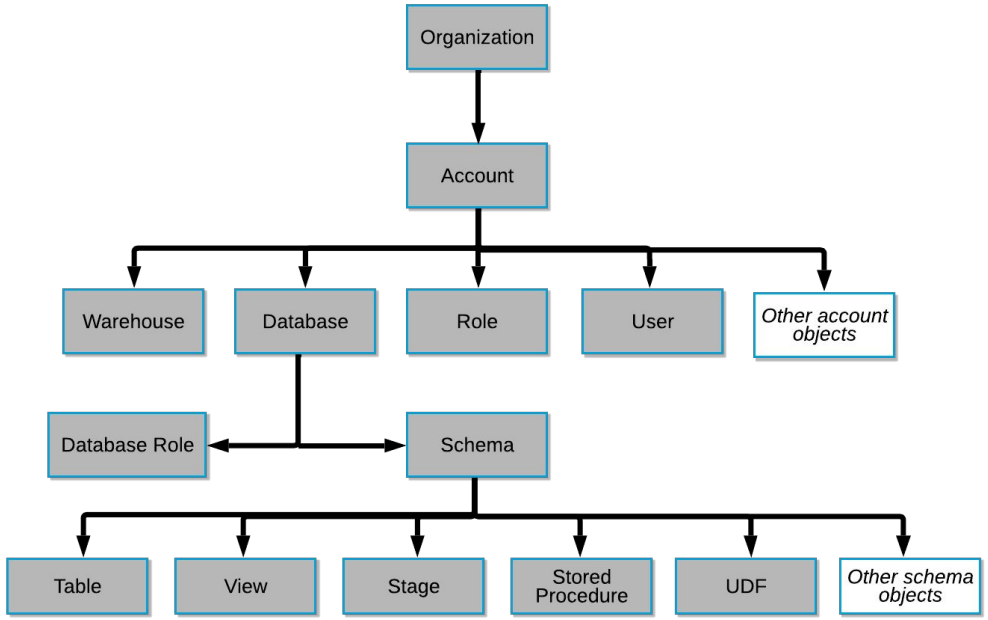


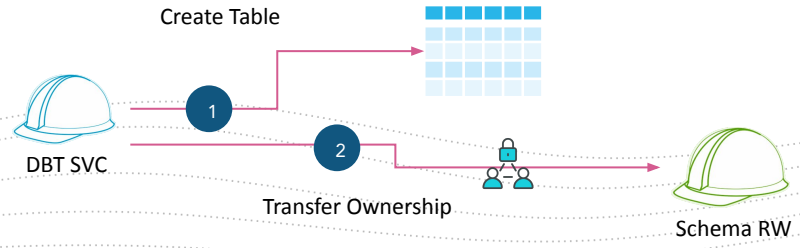


Snowflake's Roles & Access Controls

What are Securable Objects

- Every securable object is nested within a logical container in a hierarchy.
- The Organization is at the topmost container, while individual secure objects such as Tables, Views, Stages, UDF, Procedures, and other objects are stored within a Schema object, which is contained in a Database, and all the Databases are contained within the Account object.
- Each securable object is associated with a single role, usually the role that created it. Users who are in control of this particular role can control over the securable object.
- The owner role has all privileges on the object by default, including granting or revoking privileges on the object to other roles.
- Ownership can be transferred from one role to another.



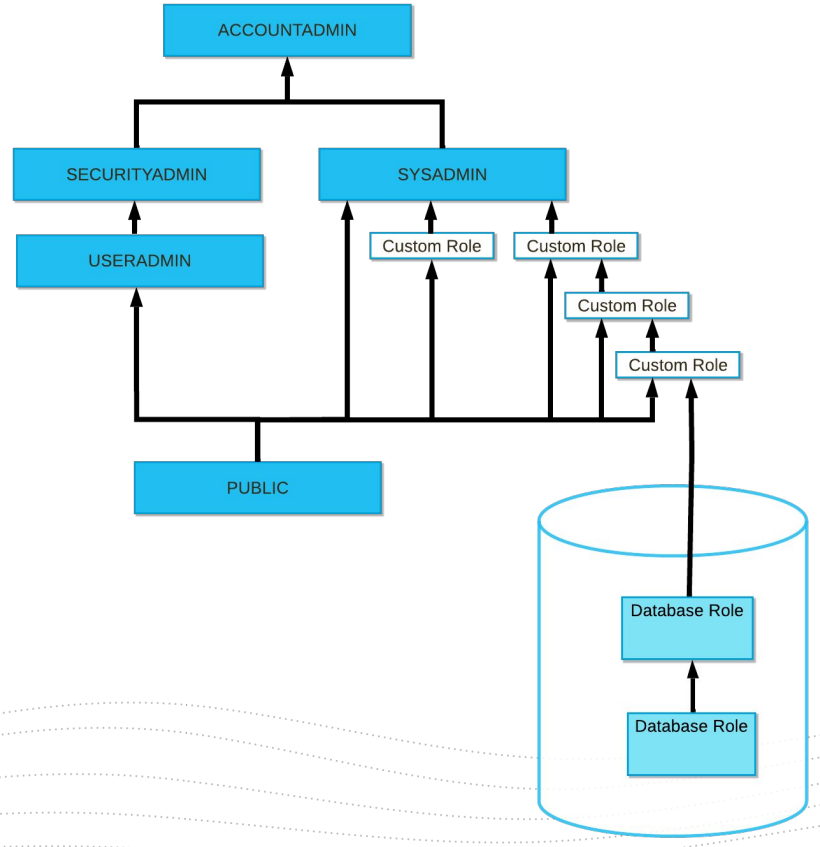


Snowflake's Roles & Access Controls

What are Roles and What Roles are Available?

- Roles are the entities to which privileges on securable objects can be granted and revoked.
- Their main purpose is to authorize users to carry out necessary actions within the organization.
- A user can be assigned multiple roles, which permits them to switch between roles and execute multiple actions using distinct sets of privileges.
- Each role is assigned a set of privileges, allowing users assigned to the role to access the resources they need.
- Roles can also be nested, allowing for more granular control over access to securable objects.

Snowflake's System Default Roles:



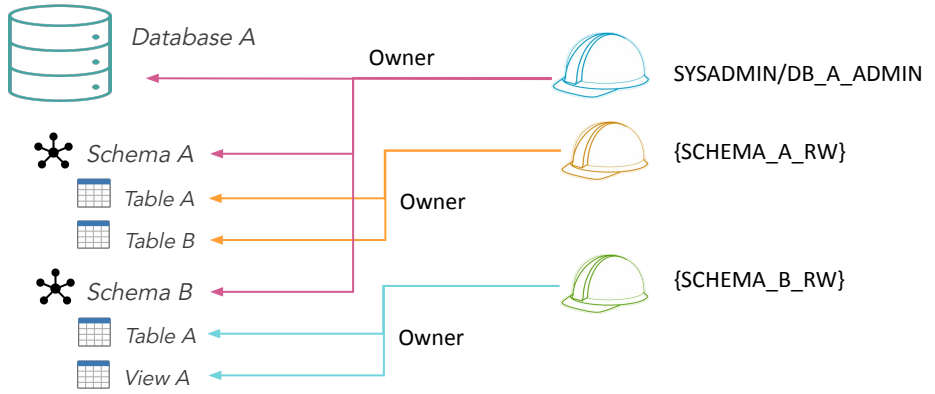
Snowflake's Roles & Access Controls

What is Privileges in Snowflake

Privileges define specific actions that users or roles are allowed to perform on securable objects in Snowflake.

Privileges are managed using the GRANT and REVOKE commands.

In non-managed schemas, these GRANT and REVOKE commands can only be used by the role that owns an object or any Snowflake roles with the MANAGE GRANTS privilege for that particular object whereas, in managed schemas, only the schema owner or a role with the MANAGE GRANTS privilege can grant privileges on objects in the schema, including FUTURE GRANTS, which centralizes privilege management.



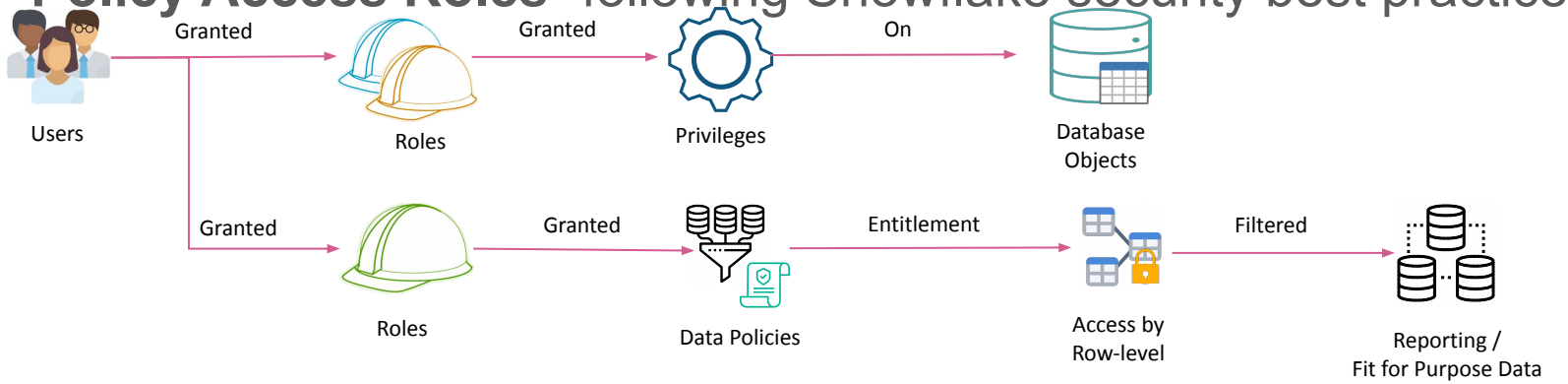
Role Privilege Matrix

Role	AD Group	privilege	granted_on	Role {env}_{db}							Database	Warehouse		Schema				Stage	
				_RO	_SEC_RO	_SUPPORT_RO	_SUPPORT_SEC_RO	_RW	_SEC_RW	_ADMIN	ADMIN_SELF_SERVICE	TASK_ADMIN	{env}_{db}_DB	_INGEST_WH	_TRANSFORM_WH	_ANALYTICS_WH	INGEST	TRANSFORM	PUBLISH	FUTURE	Grant Option?	INGEST. {env}_{db}_S3_RO	INGEST. {env}_{db}_S3_RW
{env}_{db}_RO	SNF_{env}_{db}_RO	USAGE	WAREHOUSE													X							
{env}_{db}_RO	SNF_{env}_{db}_RO	USAGE	DATABASE										X										
{env}_{db}_RO	SNF_{env}_{db}_RO	USAGE	SCHEMA																X	X			
{env}_{db}_RO	SNF_{env}_{db}_RO	SELECT	TABLES																X	X			
{env}_{db}_RO	SNF_{env}_{db}_RO	SELECT	VIEWS																X	X			
{env}_{db}_RO	SNF_{env}_{db}_RO	SELECT	MATERIALIZED VIEWS																X	X			
{env}_{db}_SEC_RO		USAGE	ROLE	X																			
{env}_{db}_SUPPORT_RO		USAGE	ROLE	X																			
{env}_{db}_SUPPORT_RO		USAGE	WAREHOUSE													X							
{env}_{db}_SUPPORT_RO		USAGE	DATABASE										X										
{env}_{db}_SUPPORT_RO		USAGE	SCHEMA														X	X	X	X			
{env}_{db}_SUPPORT_RO		SELECT	TABLES										X										
{env}_{db}_SUPPORT_RO		SELECT	VIEWS										X										
{env}_{db}_SUPPORT_RO		SELECT	MATERIALIZED VIEWS										X										
{env}_{db}_SUPPORT_SEC_RO		USAGE	ROLE		X	X																	
{env}_{db}_RW		USAGE	ROLE	X		X						X											
{env}_{db}_RW		USAGE	STAGE																			X	X
{env}_{db}_RW		USAGE	WAREHOUSE											X	X								
{env}_{db}_RW		ADD SEARCH OPTIMIZATION	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE EXTERNAL TABLE	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE FILE FORMAT	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE FUNCTION	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE MATERIALIZED VIEW	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE PIPE	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE PROCEDURE	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE SEQUENCE	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE STAGE	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE STREAM	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE TABLE	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE TASK	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE TEMPORARY TABLE	SCHEMA														X	X	X	X			
{env}_{db}_RW		CREATE VIEW	SCHEMA														X	X	X	X			
{env}_{db}_RW		MODIFY	SCHEMA														X	X	X	X			

Snowflake Security Architecture Overview

Information Architecture – Role Based Access Design

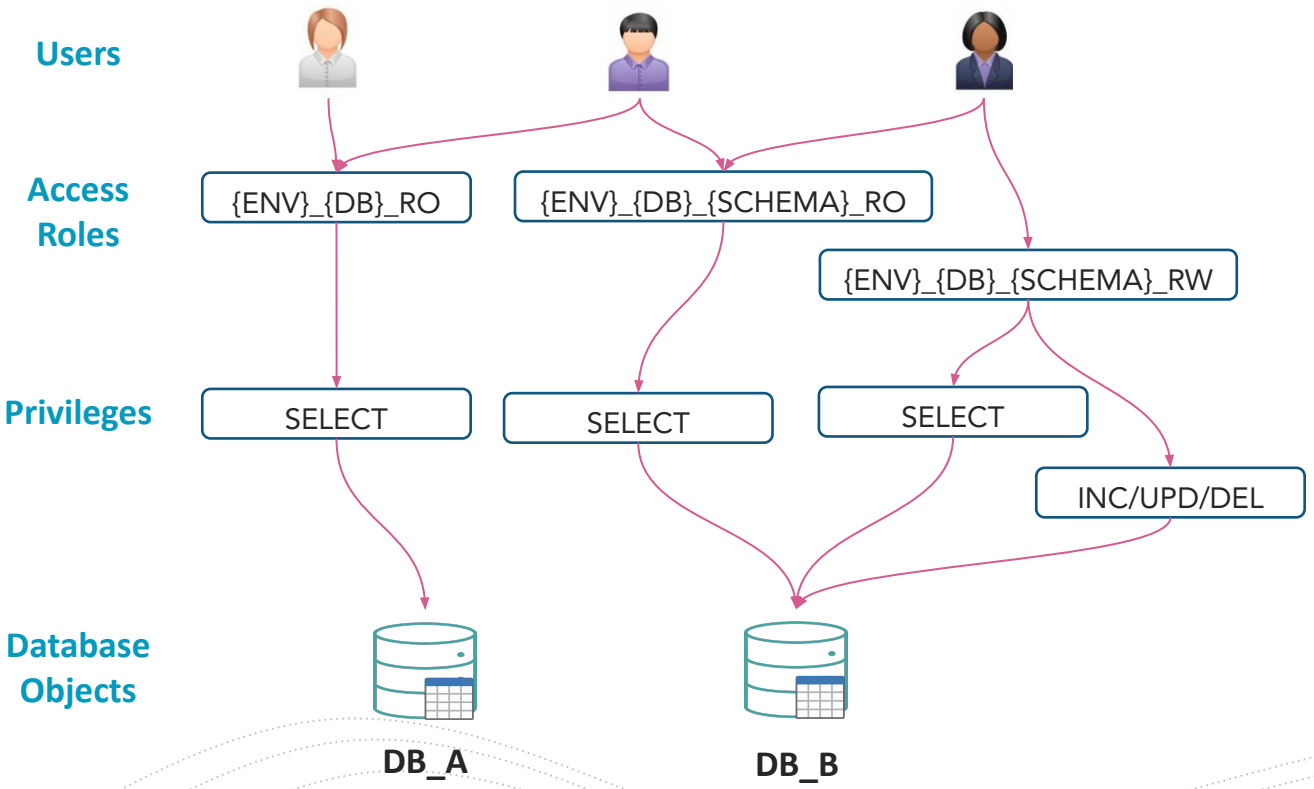
- Within Snowflake we are enhancing our governance and data access controls through **Role-based Access Controls (RBAC)** & **Discretionary Access Control (DAC)**.
- We are creating new role types that include “**Functional Roles**”, “**Object Access Roles**” and “**Policy Access Roles**” following Snowflake security best practices.



- Roles that clearly defining who does what job and what data access each role has access to.
- Common framework that normalizes and streamlines the role defining process.

Snowflake Security Architecture Overview

Simplified Role Based Access Model



- This role-based access model was the initial deployment model used on Snowflake.
- While it seems to have a well-defined approach for access privileges granted to users and objects, we see limitations to scale for larger deployments.
- As access levels to more granular objects and data become required, this approach will be come cumbersome to manage.
- Especially when the level of granularity for which access roles is defined at lower levels such to specific tables/views.
- Also, this approach limits the use or governed role with a policy security model deployment.

Snowflake Security Architecture Overview

Functional + Object + Policy Role Based Access Model

- By introducing the Functional role model this will help us to overcome the granularity and scalability limitations we face with our correct basic-role models.
- We will map the Functional Role to business functions within the organization and for unique use cases.
- An example of this is the Financial Analysts role we are designing for Release one of project Great Lakes.
- Naturally, such a role is typically assigned to users based on their job function.
- In addition to the function of Financial Analysts, there are further data access securities that help to define the record level access per the Financial Analyst's region or Finance display role access defined within the core source application.

Functional Roles


Finance Analyst


Raw Finance Analyst

Object Access Roles


Finance Data Access

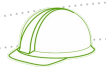

RAW Finance Data Access

Policy Roles


Finance Global Display Role


Finance Americas Display Role


Finance EMEA Display Role


Finance Asia Display Role


Finance US IHS Display Role

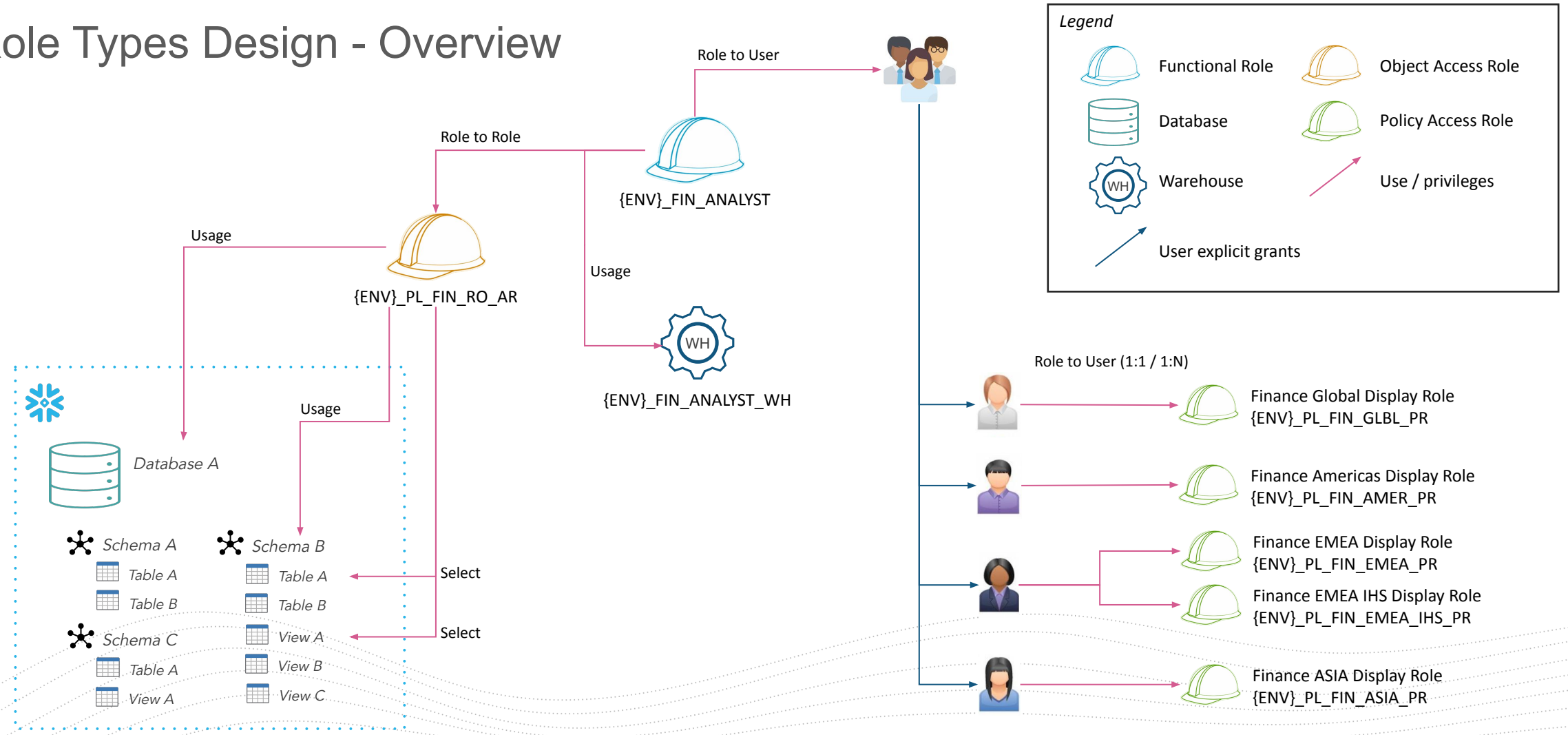

Finance LATAM ISH Display Role


Finance EMEA IHS Display Role


Finance MDT Foundation Display Role

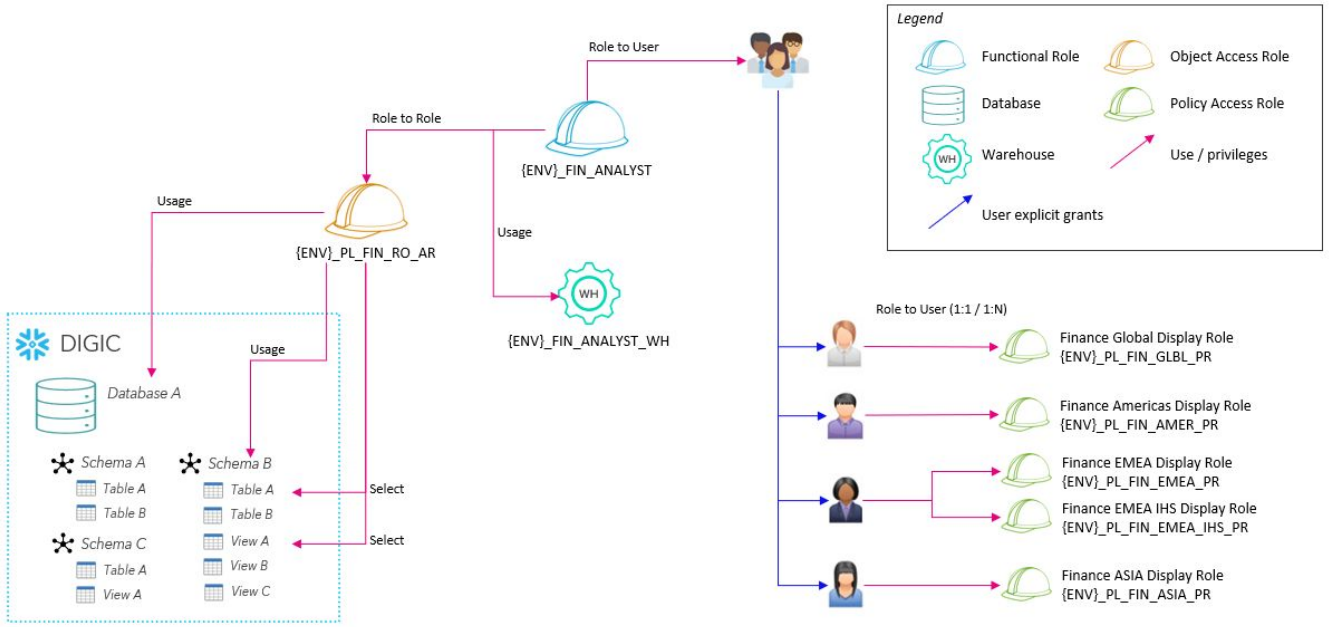
Snowflake Security Architecture Overview

Role Types Design - Overview



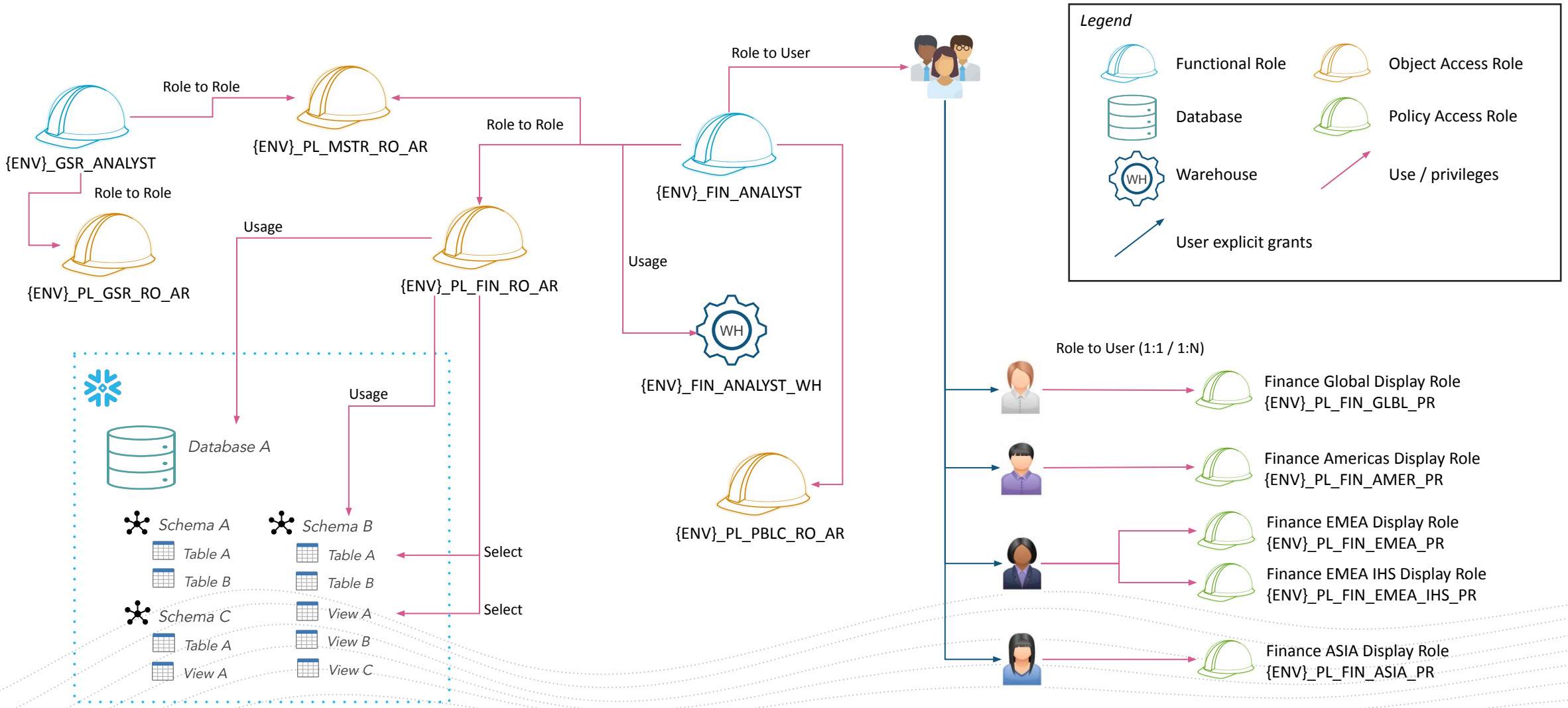
Snowflake Security Architecture Overview

Role Types Design - Overview



- Benefits
  - Functional role design (+)
  - Isolation of data (+)
  - Compute only at the functional role-level (+)
  - Indirect data policy roles support granting access to record-level data / dynamic masking of sensitive columns (+)
  - Access Roles are grantable to 1-many functional roles (+)
  - Clearer framework to administer and maintenance of data access (+)
  - Object access roles can inherit from other Object access roles (+)
- Drawbacks
  - Increased role management (-)

Snowflake Security Architecture Overview



Snowflake Security Architecture Overview

Role Types Design - Naming

	Desc
ENV	DEV, TEST, STAGE, UAT, PROD – these are the different database environments
Layer/Type	RAW, PL
Layer/ Type Desc	RAW-layer (SDH), Publish-layer (CDH/FDH)
Domain	CDF – Primary Domain of Data
Sub-domain/Function	CDF – Sub-domain Domain of Data – Function / Purpose of regional / use-case demarcation
Modifier	Optional – additive modifier to further breakdown the naming
Privileges	Optional – additive modifier to further breakdown the naming
Object Type	Role: FR, AR, PR, Other: DB, WH
Object Name	Optional - Explicit naming of object
Role Name	{ENV}_{Layer/Type}_{Domain}_{Sub-domain}_{Object-type}
AD Group	*SNF*_{ENV}_{Layer/Type}_{Domain}_{Sub-domain}_{Object-type}
Role Type/Description	Role: Functional Role (FR), Object Access Role (AR), Policy Access Role (PR), Other: Database (DB), Warehouse (WH)

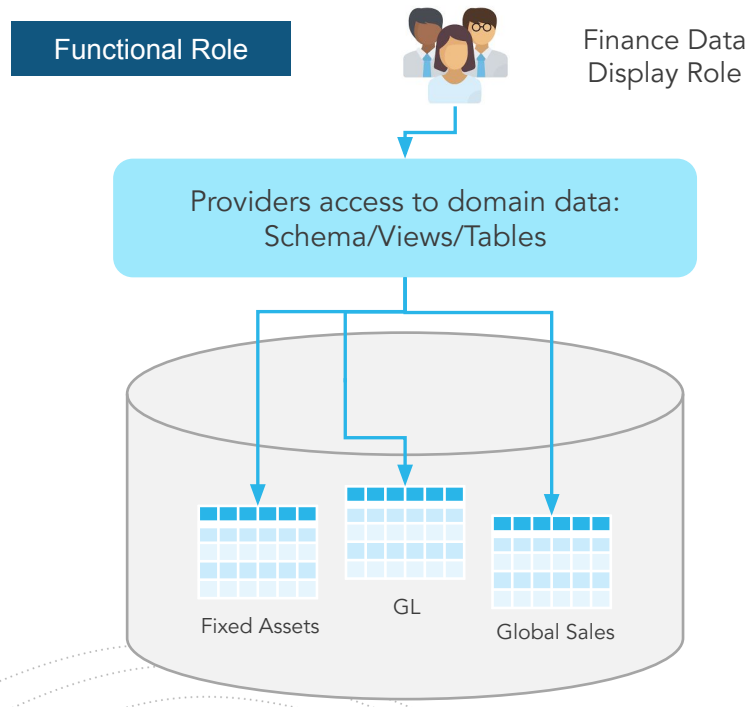
Role Naming:

- Functional Roles: {ENV}_{Layer/Type}_{Domain}_{Sub-domain}_{Functional-Role-Access-Type}
    - Raw-layer Access: DEV_RAW_FIN_ANALYST
    - Publish Layer Access: DEV_FIN_ANALYST
        - Rather then keeping PL in the name, we are dropping the {Layer/Type} for Publish Layer on the Functional Role.
        - By design the RAW functional role is straightforward and by eliminating the PL from the Publish Layer it makes it easier for the user-base.

Role-level and Row-level Security

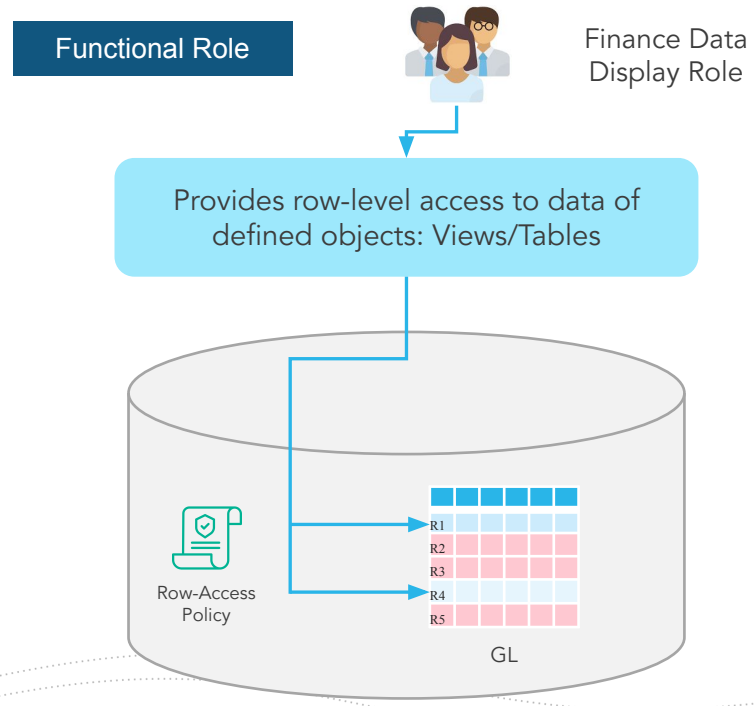
Clarification on Each Intended Use

Role-level



Role-level Security manages access to Data Object (e.g., Tables/Views)

Row-level



Row-level Security manages access to individual Records or Data Sets

Role-level and Row-level Security

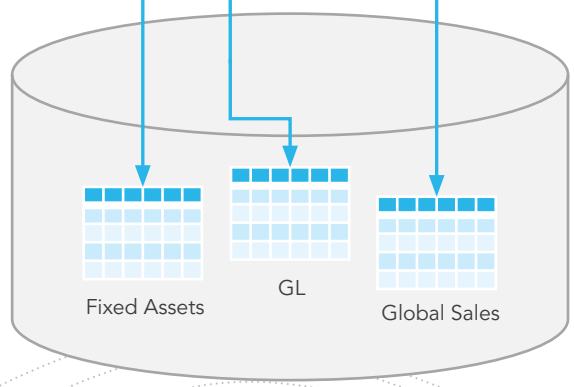
Clarification on Each Intended Use

Role-level

Functional Role

Finance Regional Display Role

Providers access to domain data Schema/Views/Tables

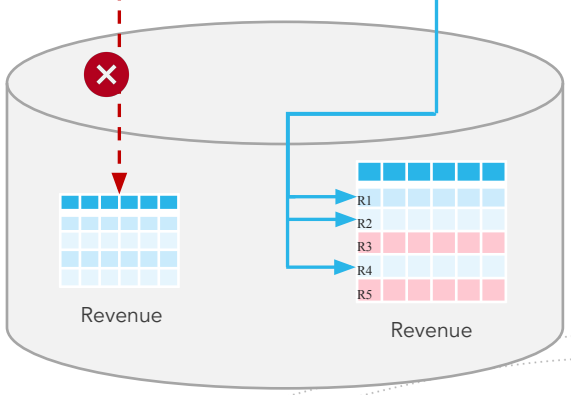


Role-level Security manages access to Data Object (e.g., Tables/Views)

Role-by-Classification

Classification Roles

No data grants; used as a means of classification / authorization



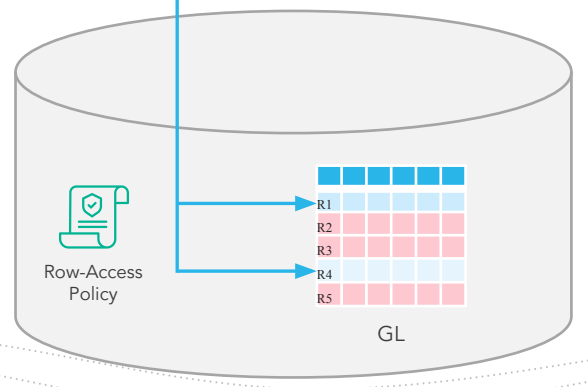
Policies that correspond based on the categories of Roles

Row-level

Functional Role

Finance Regional Display Role

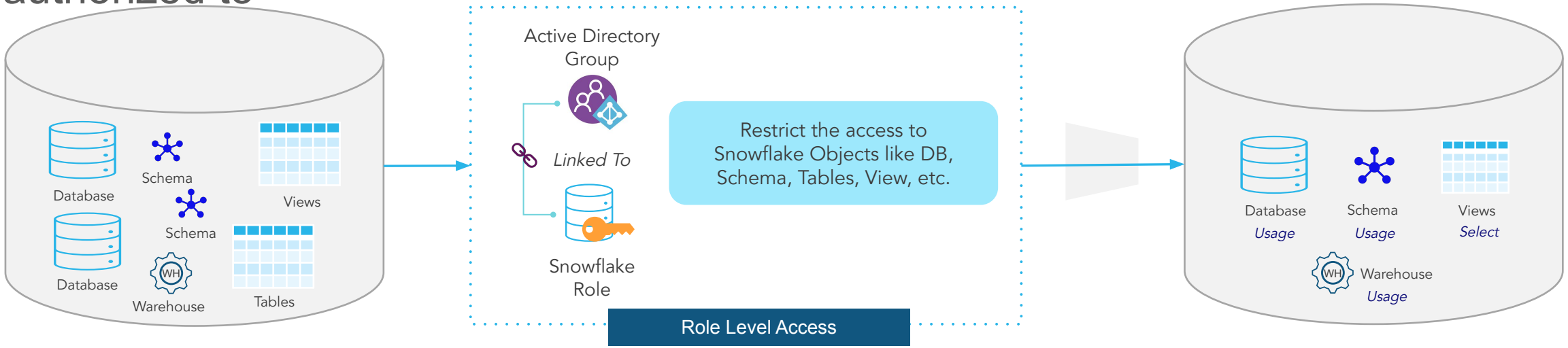
Provides row-level access to data from defined objects Views/Tables



Row-level Security manages access to individual Records or Data Sets

Role-level Security

Goal: To provide access to the controlled data to the users for which they are authorized to

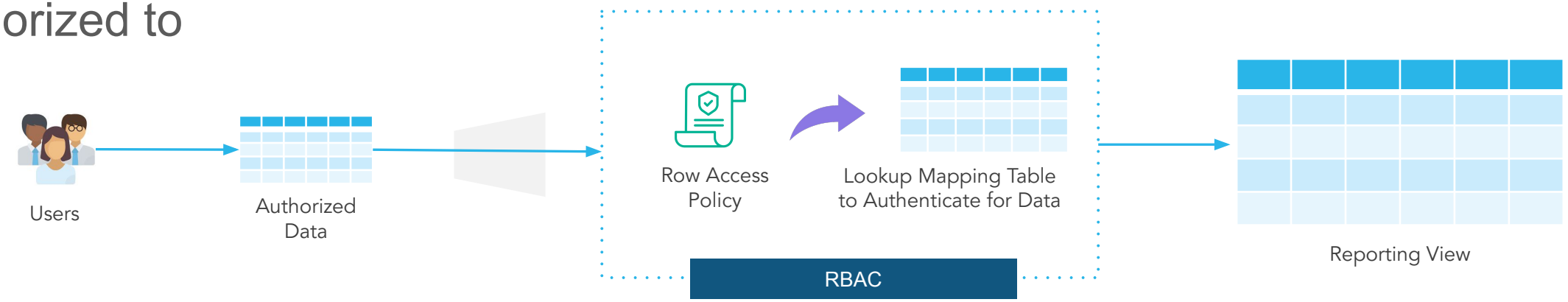


Role-level Security Review

1. Role-level Security is the Objects Authorization.
2. Controls the DDL / DML operations on the Snowflake Objects (DB, Schemas, View, Tables, etc.).
3. In terms on Workspace, the Role-level Security will open Workspace access at a Persona-level
4. This in used to govern the access for individual Functions or Personas.

Row-level Security

Goal: To provide access to the controlled data to the users for which they are authorized to

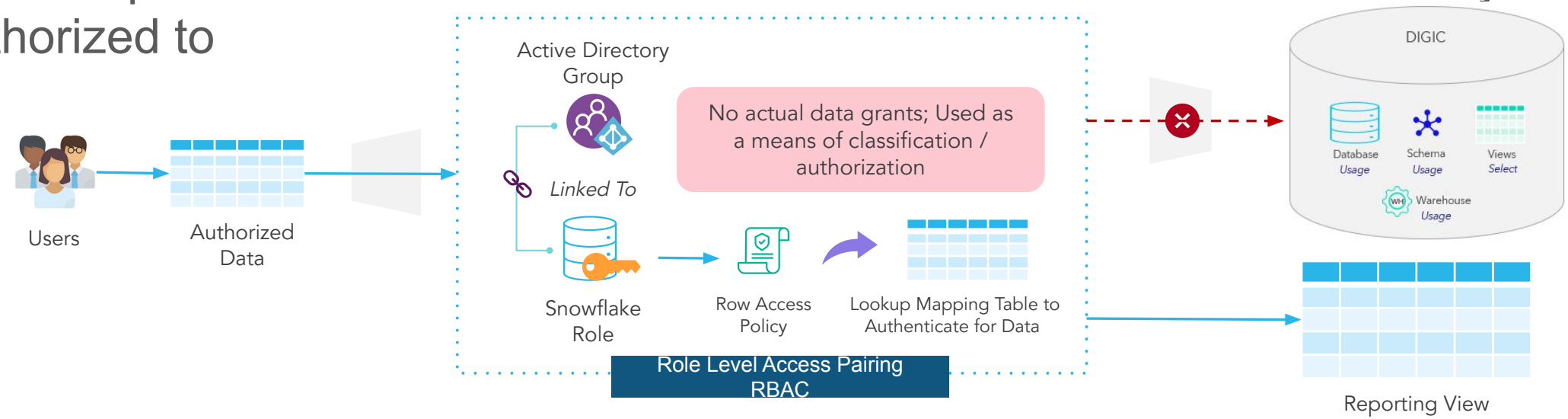


Row-level Security Review

1. Row-level security is the data-level access control.
2. It controls the access of the individual records of a View or a Table for Individual Users.
3. This will be used to govern the access of data based on domain specified entitlement mappings.
   (e.g., Company code, Region, Customer, Vendor, Controlling areas, Chart of accounts, Plant, Purchase organization, Sales organization)

Role-by-Classification Security

Goal: To provide access to the controlled data to the users for which they are authorized to



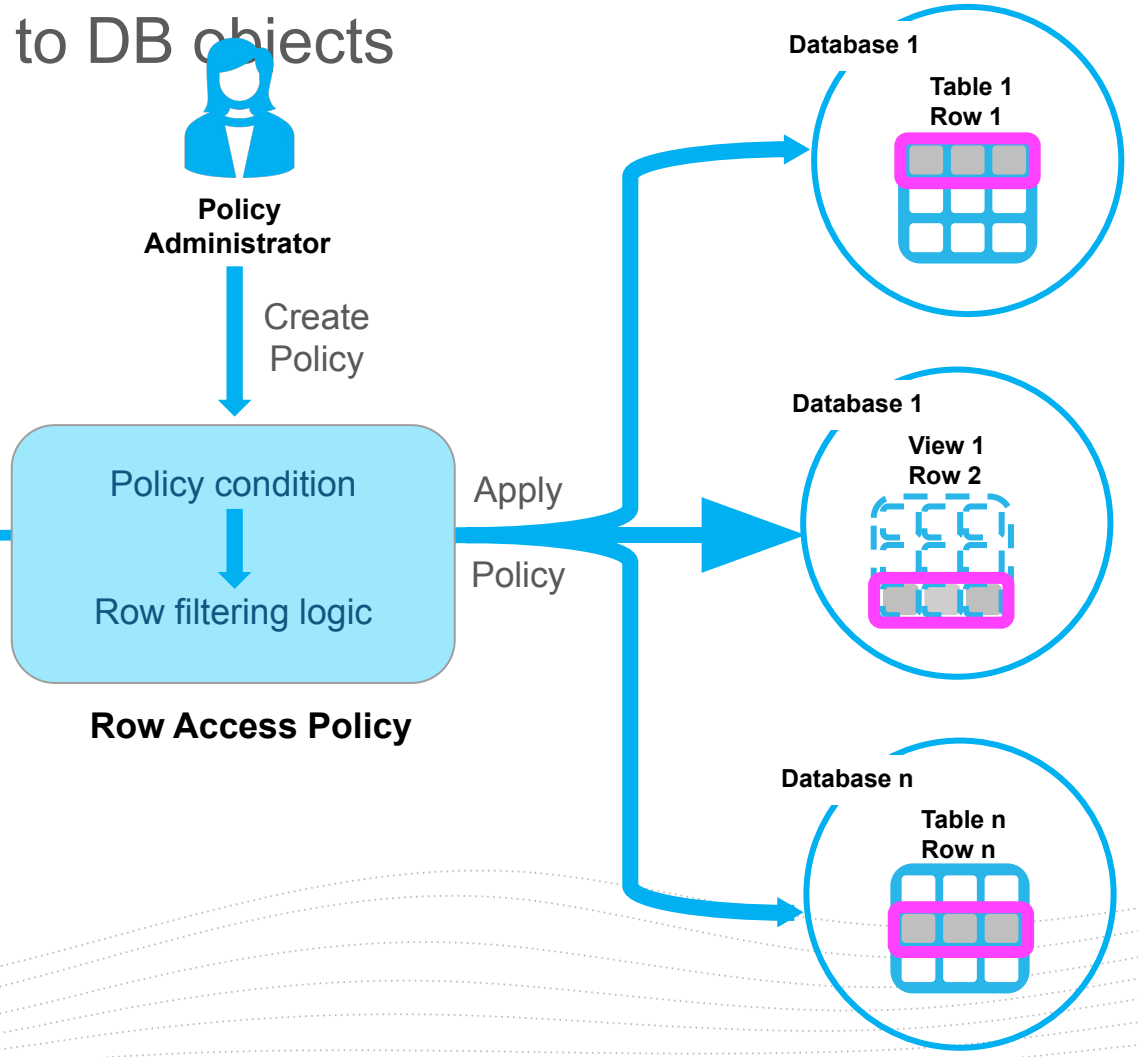
Role-by Classification Security Review

1. The use of AD groups that link to Snowflake roles.
2. The Snowflake roles do not actually provide any data object grants.
3. The roles are solely used by Data Policies as a means of classification / authorization to data.
4. Policies to use CURRENT_AVAILABLE_ROLES() function to determine user have classified access to enable data access (toggle: on/off data access).

What is a Row Access Policy?

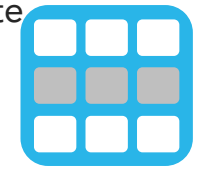
A high-level overview on how to apply to DB objects

- A row access policy provides row-level security to control which rows are returned in a query result set.
- It determines whether a given row in a table or view can be seen by a user.
- The row-access policy can work with the following types of statements:
  - `SELECT`
  - Rows specified in `UPDATE`, `DELETE`, and `MERGE`.
- A row access policy can be relatively simple to allow one particular role to view rows, or it be more complex to include a mapping table in the policy definition to determine access to rows in the query result



Row Access Policies Workflow

1. Identify / Create table



2. Setup mapping table

```
CREATE ROLE tax_user_east;
CREATE ROLE tax_user_central;
CREATE ROLE tax_user_west;
```

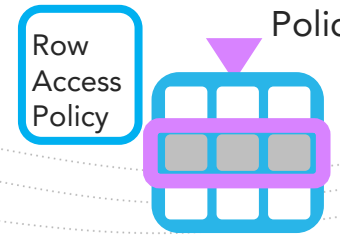
Taxuser_role	Taxpayer_state
TAX_USER_EAST	FL
TAX_USER_EAST	SC
TAX_USER_CENTRAL	IL
TAX_USER_CENTRAL	TX
TAX_USER_WEST	CA
TAX_USER_WEST	CO

3. Create Row Access Policy

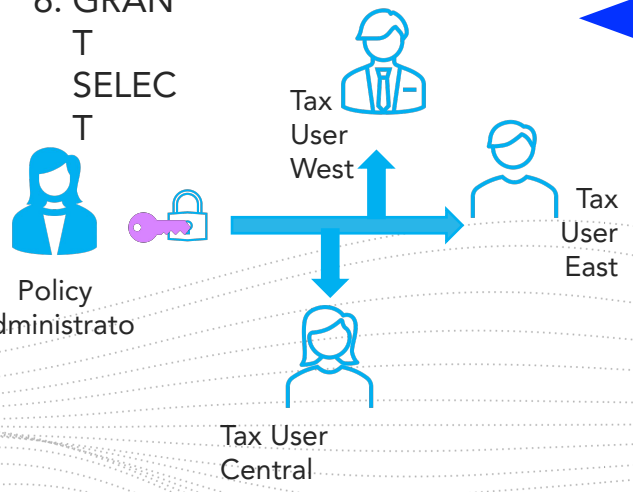
```
CREATE or REPLACE row access policy taxdata
as (taxpayertype varchar(30), taxpayerstate
returns Boolean ->
       exists (
            SELECT 1 from tax_mapping
              where taxuser_role = current_
              and taxpayer_type=taxpayertyp
         )
       or
       exists (
            SELECT 1 from taxuser_mapping
              where taxuser_role = current_role()
              and taxpayer_state=taxpayerstate);
```

4. GRANT

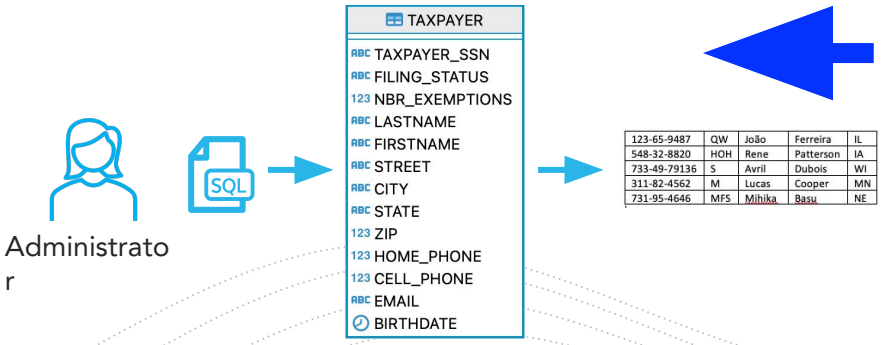
5. Apply Row Access Policy



6. GRANT SELECT



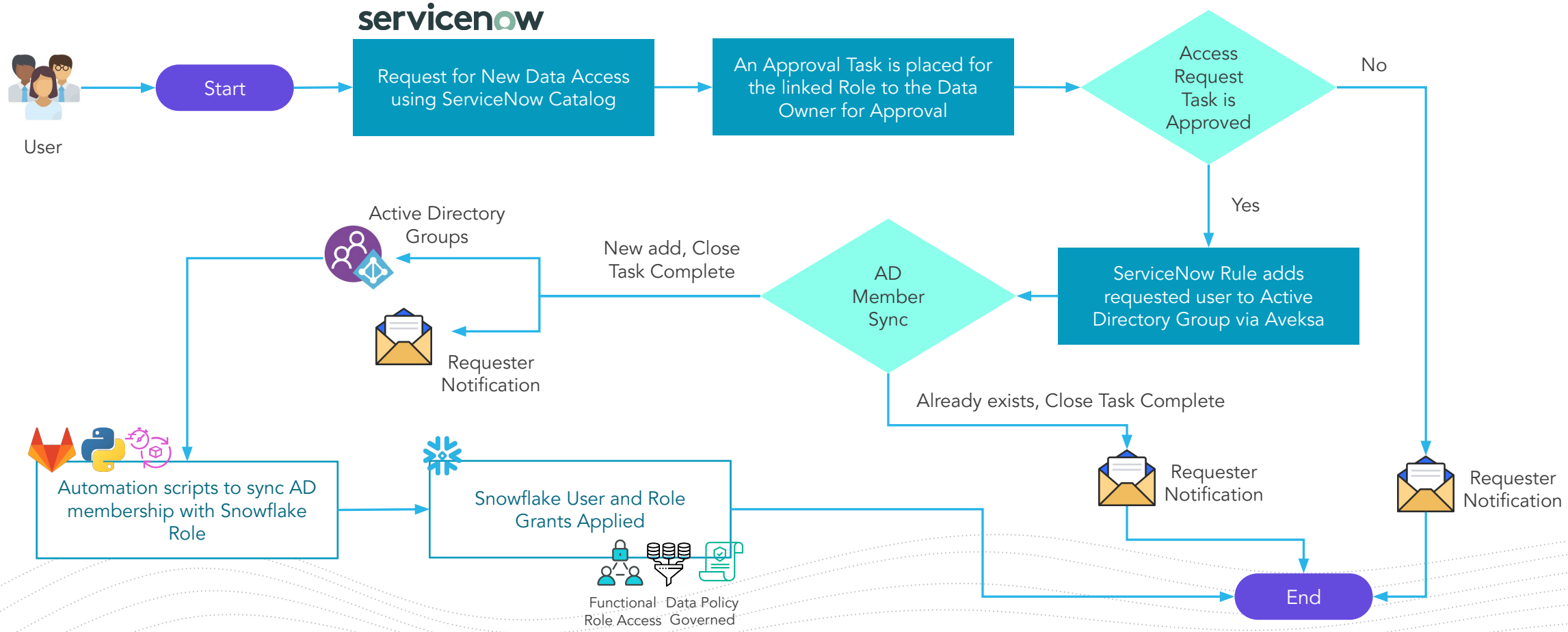
7. Test



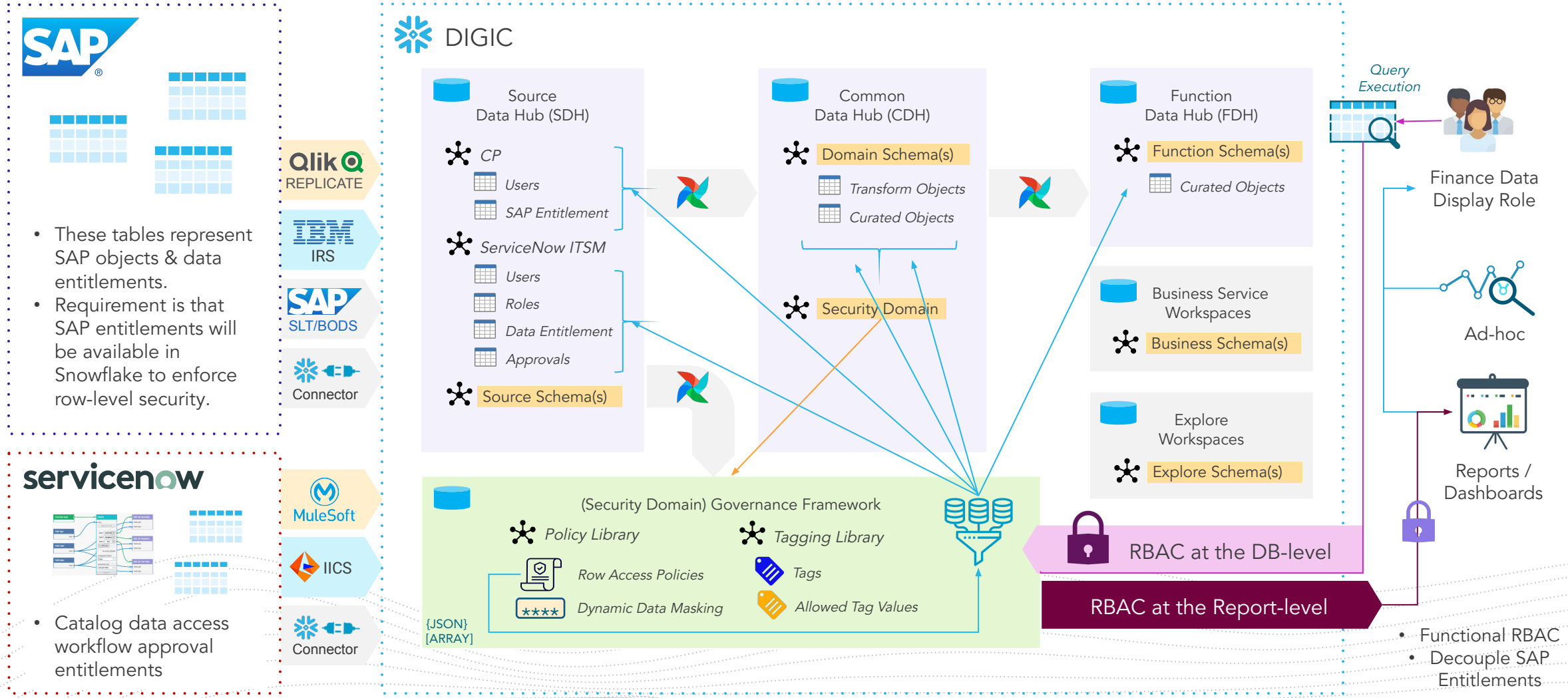
123-65-9487	QW	João	Ferreira	IL
548-32-8820	HOH	Rene	Patterson	IA
733-49-79136	S	Avril	Dubois	WI
311-82-4562	M	Lucas	Cooper	MN
731-95-4646	MFS	Mihika	Basu	NE

Data Access Request Flow

Review the high-level access Request, Approval and Assignment Automation



Layering in RBAC-Centric Entitlement



- These tables represent SAP objects & data entitlements.
- Requirement is that SAP entitlements will be available in Snowflake to enforce row-level security.

- Catalog data access workflow approval entitlements

- Functional RBAC
- Decouple SAP Entitlements

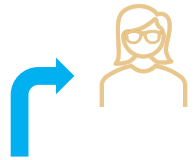
What is Dynamic Data Masking

Data Owner

Patient Id	Fname	Lname	DOB	Ins Co Id	Marital Status	Employer	Email
14-622148	Paul	Atkins	01191956	Q59753984	M	TechNo	Atkinsp@mailcorp.com
22-305487	Marek	Dvorak	06142000	U28703318	S	Smiths	MarDv@mails.com
18-621448	Kerstin	Nilsson	11301978	A95248736	D	Alta Health	KNilsson@Workin.com
12-558430	Ciaran	Byrne	08151945	L44896523	W	Retired	Ciaran@scortha.com

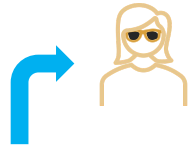
Base Table or View

Results seen



Authorized role

Patient Id	Fname	Lname	DOB	Ins Co Id	Marital Status	Employer	Email
*********	Paul	Atkins	****1956	Q*****984	M	TechNo	**********
*********	Marek	Dvorak	****2000	U*****318	S	Smiths	**********
*********	Kerstin	Nilsson	****1978	A*****736	D	Alta Health	**********
*********	Ciaran	Byrne	****1945	L*****523	W	Retired	**********



Public

Patient Id	Fname	Lname	DOB	Ins Co Id	Marital Status	Employer	Email
++++++++	+++++	+++++	+++++++	++++++++	+++++	++++++++	++++++++++++++++
++++++++	+++++	+++++	+++++++	++++++++	+++++	++++++++	++++++++++++++++
++++++++	+++++	+++++	+++++++	++++++++	+++++	++++++++	++++++++++++++++
++++++++	+++++	+++++	+++++++	++++++++	+++++	++++++++	++++++++++++++++

- A way to protect sensitive data by changing data visible in the result set
- A policy that specifies what data is shown in the result set and is enforced at query runtime
- Masking is a column-level security feature that selectively conceals plain-text data
- Like row access policies – data masking is a schema-level object

Why use Dynamic Data Masking

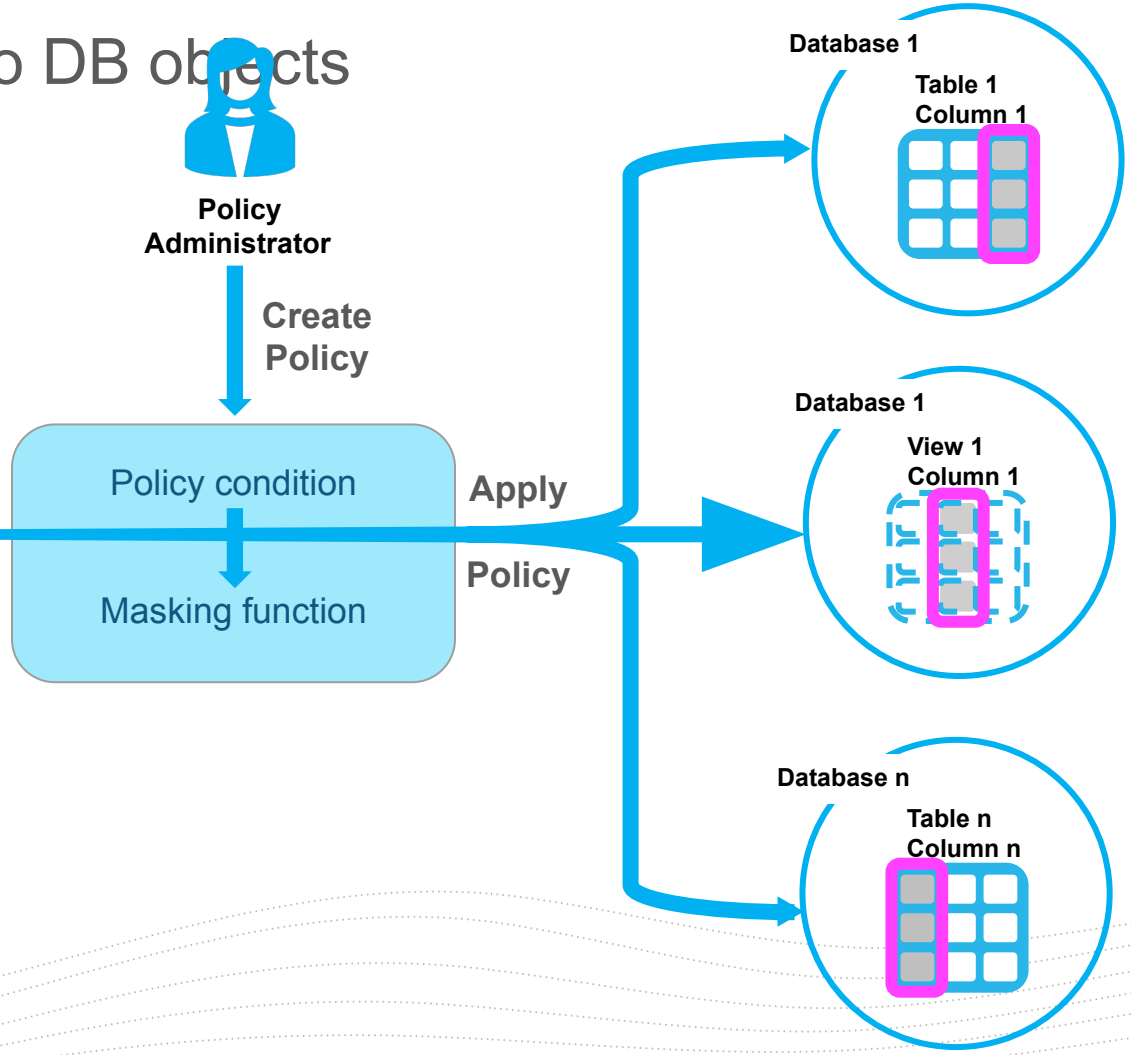
- Safeguard sensitive data from unauthorized access
- Masked data cannot be reverse engineered or decoded
- Supports segregation of duties
- PII and PHI data protection is required by many compliance directives

- Data masking, in conjunction with RBAC, ensures granular access control.
- Transform data only when queried
- Benefits to data masking
  - Ease of use
  - Data administration and separation of duties
  - Data authorization and governance
  - Data sharing
  - Change management

What is a Masking Policy?

A high-level overview on how to apply to DB objects

- Condition(s) and masking function applied when condition(s) met.
- Support all data types, shared data, and streams.
- Carry over policy associations when cloned.



Data Masking Policy Workflow

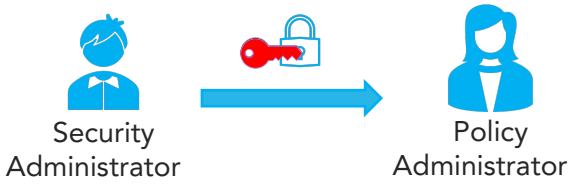
Assumes Management is Centralized

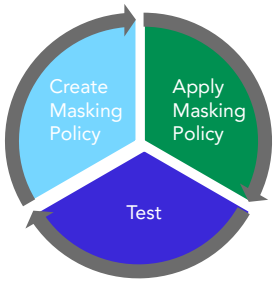
2. Create Masking Policy

```
CREATE or REPLACE masking policy ssn_mask as
     (val string) returns string ->
     CASE
        when current_role() in ('SYSADMIN') ther
        when current_role() in('TAXDATA_STEWARD
        repeat('*', length(val)-4 || right(val,
        else '*** REDACTED ***'
     END;
```

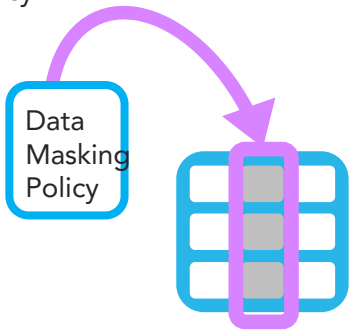
Data Governance Policy

1. **GRANT**

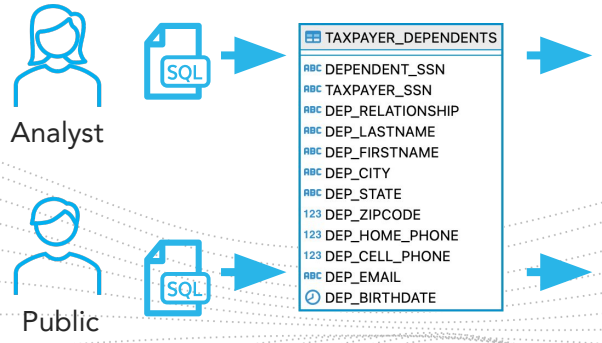




3. Apply Data Masking Policy



4. Test

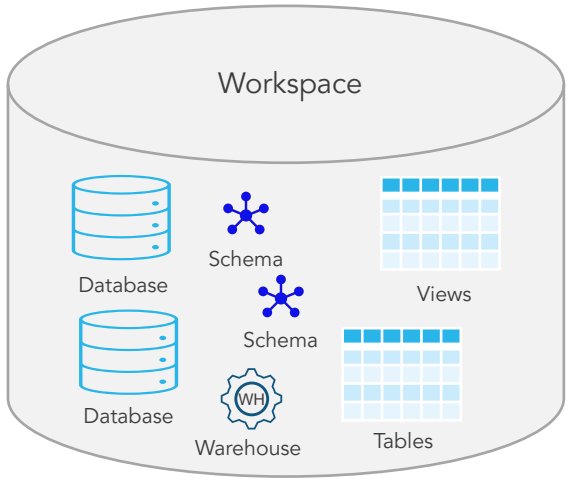


DEPENDENT_SSN	TAXPAYER_SSN	DEP_RELATIONSHIP
*****6798	*****5237	Spouse
*****9001	*****5237	Daughter

DEPENDENT_SSN	TAXPAYER_SSN	DEP_RELATIONSHIP
*** REDACTED ***	*** REDACTED ***	Spouse
*** REDACTED ***	*** REDACTED ***	Daughter

Workspace Roles Hierarchy

Overview



{ENV}_{WORKSPACE}_DB
- *DEV_{WORKSPACE}_DB*
- *TEST_{WORKSPACE}_DB*
- *STAGE_{WORKSPACE}_DB*
- *PROD_{WORKSPACE}_DB*

{ENV}_{WORKSPACE}_{WORKLOAD}_{?SIZE?}_DB
- *{ENV}_{WORKSPACE}_INGEST_WH*
- *{ENV}_{WORKSPACE}_TRANSFORM_WH*
- *{ENV}_{WORKSPACE}_ANALYTICS_WH*

Optional {?SIZE?}:
- *{ENV}_{WORKSPACE}_ANALYTICS_MD_WH*
- *{ENV}_{WORKSPACE}_ANALYTICS_LG_WH*
- *{ENV}_{WORKSPACE}_ANALYTICS_HG_WH*
- *{ENV}_{WORKSPACE}_ANALYTICS_XG_WH*

{ENV}_{WORKSPACE}_{ROLE}
- *{ENV}_{WORKSPACE}_ADMIN*
- *{ENV}_{WORKSPACE}_RW*
- *{ENV}_{WORKSPACE}_RO*

Schema-level:
- *{ENV}_{WORKSPACE}_{SCHEMA}_RW*
- *{ENV}_{WORKSPACE}_{SCHEMA}_RO*

Custom
- *CSS_{WORKSPACE}_{SCHEMA}_RO*
- *CSS_{WORKSPACE}_{SCHEMA}_{USECASE}_RO*
- *{ENV}_{WORKSPACE}_{SCHEMA}_{USECASE}_RO*