## Role-Based Access Control (RBAC) in Snowflake and Azure

This document outlines the design, implementation, and validation of a Role-Based Access Control (RBAC) framework for a Snowflake data platform integrated with Microsoft Azure. The primary goal is to establish a secure, scalable, and maintainable access control model that ensures users and applications have precisely the permissions necessary to perform their duties, adhering to the principle of least privilege.

The solution leverages Azure Active Directory (AAD) as the central identity provider and Snowflake's native RBAC capabilities to create a seamless and robust security posture for data assets.

**Objectives and Goals**

The implementation aims to achieve the following key objectives:

- **Centralized Identity Management:** Utilize Azure AD for single sign-on (SSO) and centralized user/group provisioning.
- **Principle of Least Privilege:** Ensure users and roles are granted only the minimum permissions required for their tasks.
- **Separation of Duties:** Clearly demarcate responsibilities between security, data engineering, analytics, and business intelligence teams.
- **Scalability and Maintainability:** Design a role hierarchy that is easy to extend and manage as the organization and data ecosystem grow.
- **Auditability:** Maintain a clear and easily understandable permission structure for compliance and auditing purposes.

**System Design & Architecture**

**1. Core RBAC Concepts in Snowflake**

Snowflake's access control model is built around a hierarchy of **Securables**, **Roles**, and **Users**.

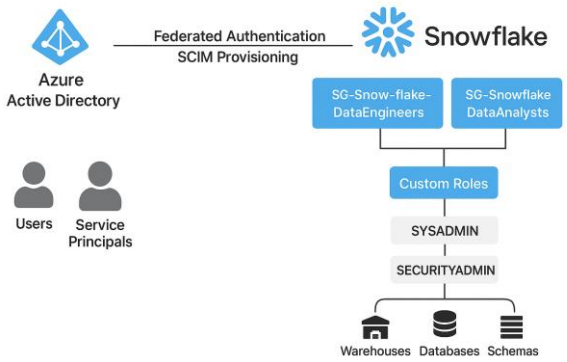
- **Securables:** Objects in Snowflake to which access can be granted (e.g., Databases, Schemas, Tables, Views, Warehouses).
- **Roles:** A container for a set of privileges. Roles are granted to users or other roles.
- **Users:** Individual identities (human or service accounts) that are granted roles to interact with Snowflake.
- **Privileges:** Defined actions that can be performed on a securable (e.g., USAGE, SELECT, INSERT, CREATE).
- **Role Hierarchy:** Roles can be granted to other roles, creating an inheritance chain. A user's effective privileges are the union of all privileges from all roles granted to them, either directly or through inheritance.

**2. Integration with Azure Active Directory (AAD)**

Azure AD serves as the identity provider (IdP) for federated authentication. This allows users to leverage their existing corporate credentials to access Snowflake.

- **AAD Security Groups:** AAD security groups are mapped to Snowflake roles. This is a best practice as it allows for access management to be handled within Azure AD.
- **SCIM Provisioning (Optional but Recommended):** System for Cross-domain Identity Management (SCIM) can be used to automatically provision users and groups from Azure AD to Snowflake, ensuring consistency and reducing manual overhead.

**3. System Architecture Diagram**




**Implementation Steps**

**1. Phase 1: Azure Active Directory Configuration**

1. **Create Security Groups:** In Azure AD, create security groups for each functional role defined in your Snowflake RBAC model.
    - Example Groups: SG-Snowflake-DataEngineers, SG-Snowflake-DataAnalysts, SG-Snowflake-ReadOnly.
2. **Populate Groups:** Add the appropriate users and/or service principals to these groups.
3. **Configure Snowflake with AAD:** In Snowflake, configure the security integration to use Azure AD for federated authentication. (This typically involves providing Azure AD tenant ID and application details).

**2. Phase 2: Snowflake Security Hierarchy Setup**

Using a SECURITYADMIN role, execute the following SQL statements to create the core role hierarchy.

```sql
-- Create custom roles
USE ROLE SECURITYADMIN;

CREATE ROLE IF NOT EXISTS ROLE_DATA_ENGINEER;
CREATE ROLE IF NOT EXISTS ROLE_DATA_ANALYST;
CREATE ROLE IF NOT EXISTS ROLE_READ_ONLY;

-- Grant roles to the system roles for manageability
GRANT ROLE ROLE_DATA_ENGINEER TO ROLE SYSADMIN;
GRANT ROLE ROLE_DATA_ANALYST TO ROLE SYSADMIN;
GRANT ROLE ROLE_READ_ONLY TO ROLE SYSADMIN;
```

### 3. Phase 3: Privilege and Access Granting

This phase involves granting specific privileges on securables to the custom roles.

1. **Grant Warehouse Privileges:**

```sql
USE ROLE SECURITYADMIN;
GRANT USAGE ON WAREHOUSE COMPUTE_WH TO ROLE ROLE_DATA_ENGINEER;
GRANT USAGE ON WAREHOUSE COMPUTE_WH TO ROLE ROLE_DATA_ANALYST;
GRANT USAGE ON WAREHOUSE COMPUTE_WH TO ROLE ROLE_READ_ONLY;
```

2. **Grant Database and Schema Privileges:**

```sql
USE ROLE SECURITYADMIN;
GRANT USAGE ON DATABASE PROD_DB TO ROLE ROLE_DATA_ENGINEER;
GRANT USAGE ON DATABASE PROD_DB TO ROLE ROLE_DATA_ANALYST;
GRANT USAGE ON DATABASE PROD_DB TO ROLE ROLE_READ_ONLY;

GRANT USAGE ON ALL SCHEMAS IN DATABASE PROD_DB TO ROLE ROLE_DATA_ENGINEER;
GRANT USAGE ON ALL SCHEMAS IN DATABASE PROD_DB TO ROLE ROLE_DATA_ANALYST;
GRANT USAGE ON ALL SCHEMAS IN DATABASE PROD_DB TO ROLE ROLE_READ_ONLY;
```

3. **Map AAD Groups to Snowflake Roles:** This is the critical link. The AAD group becomes a user in Snowflake and is granted a role.

```
-- Create a user in Snowflake that represents the AAD group (Note: The name must match)
CREATE USER "SG-Snowflake-DataEngineers" DISPLAY_NAME = "AAD Group: Data Engineers";
CREATE USER "SG-Snowflake-DataAnalysts" DISPLAY_NAME = "AAD Group: Data Analysts";

-- Grant the Snowflake roles to the AAD group users
GRANT ROLE ROLE_DATA_ENGINEER TO USER "SG-Snowflake-DataEngineers";
GRANT ROLE ROLE_DATA_ANALYST TO USER "SG-Snowflake-DataAnalysts";
```

### 4. Phase 4: Validation and Testing

1. **User Login:** Verify that users can log in via SSO using their Azure AD credentials.
2. **Role Verification:** After logging in, a user can run SELECT CURRENT_ROLE(); to see their current role. They can see all available roles with SHOW ROLES;.
3. **Privilege Testing:** Test the effective privileges for each role.
    - **As a Data Analyst:** Verify SELECT operations succeed on tables in PROD_DB, while INSERT/UPDATE operations are correctly denied.
    - **As a Read-Only User:** Verify SELECT works only in the REPORTING schema and is denied elsewhere.