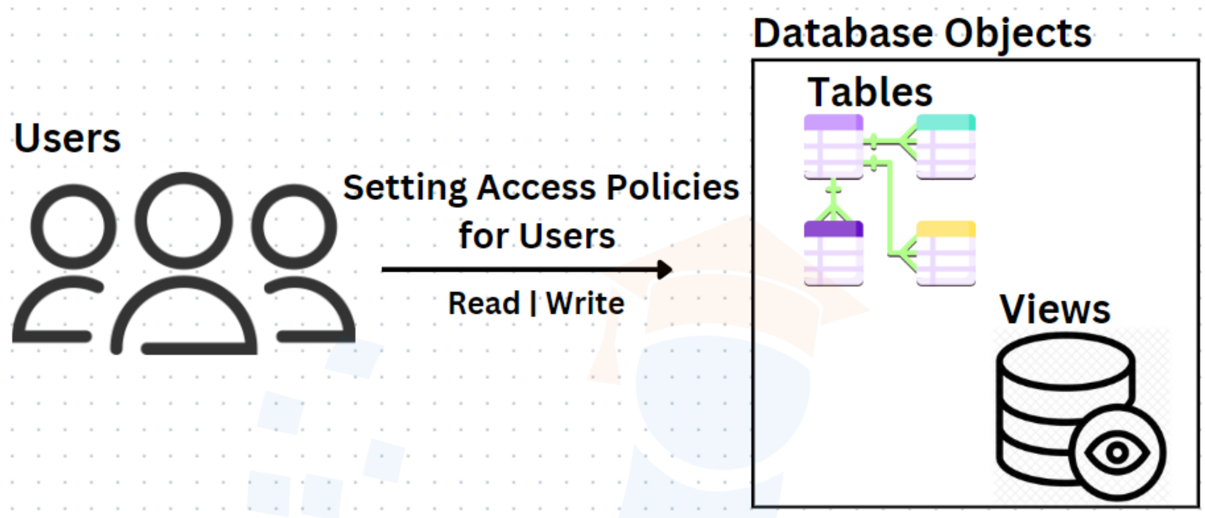
Unity Catalog

Unified Governance Solution for all the Data Assets in the Lakehouse.

Data Governance is a practice of making data secure by setting data access policies for Users.

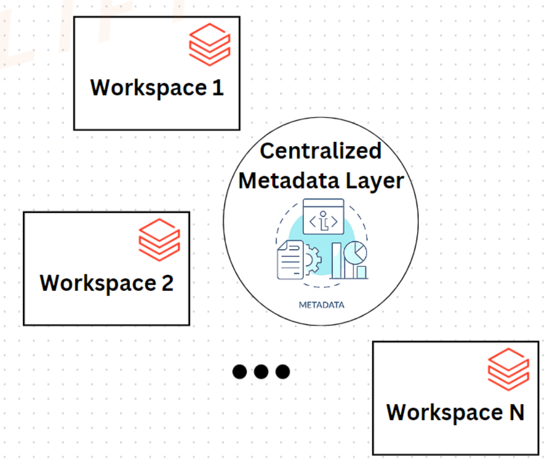


Note:

- Unity Catalog is a data governance solution to make data secure by granting only required permissions to a specific group of users.
- Centralized Data Governance across clusters and not limited to individual workspaces.

Key Features of Unity Catalog

1. Unity Catalog allows for the implementation of a centralized metadata layer. This ensures that the databases, tables and views can be shared across multiple workspaces.



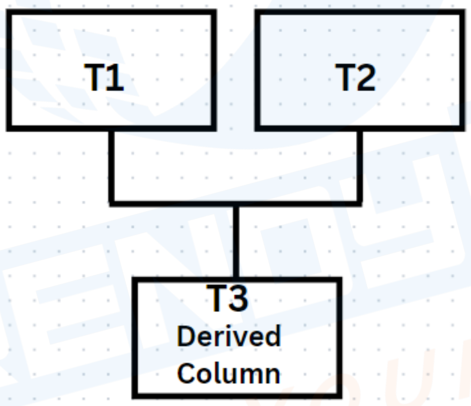
2. Unity Catalog Security Model is based on Standard ANSI SQL and allows admins to grant permissions to the users to access the database objects in the data lake using a familiar SQL like syntax.
3. Unity Catalog supports built-in auditing by capturing user level audit logs of actions performed against the metastore.

   This enables building a strong security system as the admins can access fine-grained details of who accessed the datasets and what actions were performed.

4. Unity Catalog also offers additional features like **Delta Sharing, Data Lineage and Data Discovery.**

   - Unity Catalog natively supports **Delta Sharing**, it is the world's first open source protocol for secure data sharing. Data (in any format, especially parquet format) existing in the data lake can be securely shared to any computing platform.

   - All the transformations/actions performed on the data are captured in the form of a graph termed as **Data Lineage**. This lets us back-trace to know how the data (ex-derived column) was generated.



   - **Data Discovery** is an open search capability to search based on keywords to get the results.

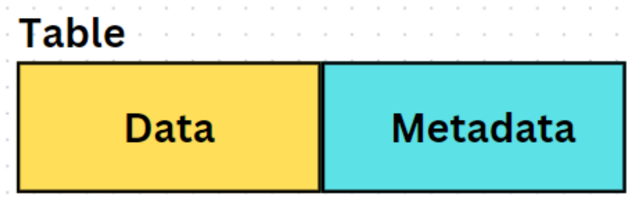
Limitations of implementing Data Governance without Unity Catalog

- Without a Unity Catalog, the metastore, user management and access control are defined at the workspace level and not centralized. The data objects such as databases and tables which are created in one workspace will not be visible in other workspace.

Configuring Metastore for Unity Catalog

Step 1 : Configure Storage for Metastore.

Create a Storage Account (ADLS Gen2) which will hold the metadata.



Step 2 : Create a Metastore through Databricks UI by connecting it to the Storage Account location created in the previous step. Multiple workspaces can be linked to this Metastore.

Steps to Create and Configure Metastore

1. **Create a Storage Account** with hierarchical namespace enabled, ADLS Gen2 will contain the metadata related to Unity Catalog Metastore and the objects.

   Note: In case of Managed Tables, the storage account will also contain the actual data along with the metadata.

2. **Create a Storage Container** within the Storage Account that will hold the Unity Catalog Metastore's metadata and the managed data.

3. **Allow Databricks to connect and access the Storage Account.** Create a Databricks Access Connector that will hold a managed identity through which Databricks can access the Storage Container.

   - Create a Resource "Access Connector for Azure Databricks" in the Azure portal. (Access Connector is a better alternative to Service Principal as the hassle of creating key vaults and managing the keys are taken care internally by the Access Connector)

   - Allow the Access Connector to access the Storage Account by setting the Access Control(IAM) policy of the Storage Account

Navigate to Storage Account -> Select the Access Control (IAM) -> Add -> Role Assignment -> Storage Blob Data Contributor Role -> Assign Access to Managed Identity -> Select the Access Connector created in the previous step under the Managed Identity field.

Steps to Create a Metastore through Databricks UI

1. **Create an Azure Databricks Workspace** - Choose a **Premium (Role-based Access Controls)** under the pricing tier field while creating the workspace.

2. **Create another Databricks Workspace** - Choose a **Standard (Apache Spark secure with Azure AD)** under the pricing tier field while creating the workspace.

3. Once the Workspace is deployed, launch the workspace by logging in to the workspace as an account admin.

4. Navigate to the **Data** icon on the left panel and click on **Create Metastore** option.

- Name the metastore.

-Make sure to select the region where the storage account and most of the workspaces are located.

-Provide the **Storage Account Path** (ADLS Gen2 Container path) created in the earlier steps.

-Provide the **Access Connector ID** and click on Create to create the metastore.

-Linking the metastore to the centralized workspace. Select the workspace to which the metastore needs to be assigned -> Assign -> Enable.

Note: Unity Catalog is only available on the Premium tier. Therefore, we can choose only the premium databricks workspace and not the standard.

Unity Catalog Object Model

Metastore is the top level container for the metadata. Each metastore exposes 3 level namespaces.

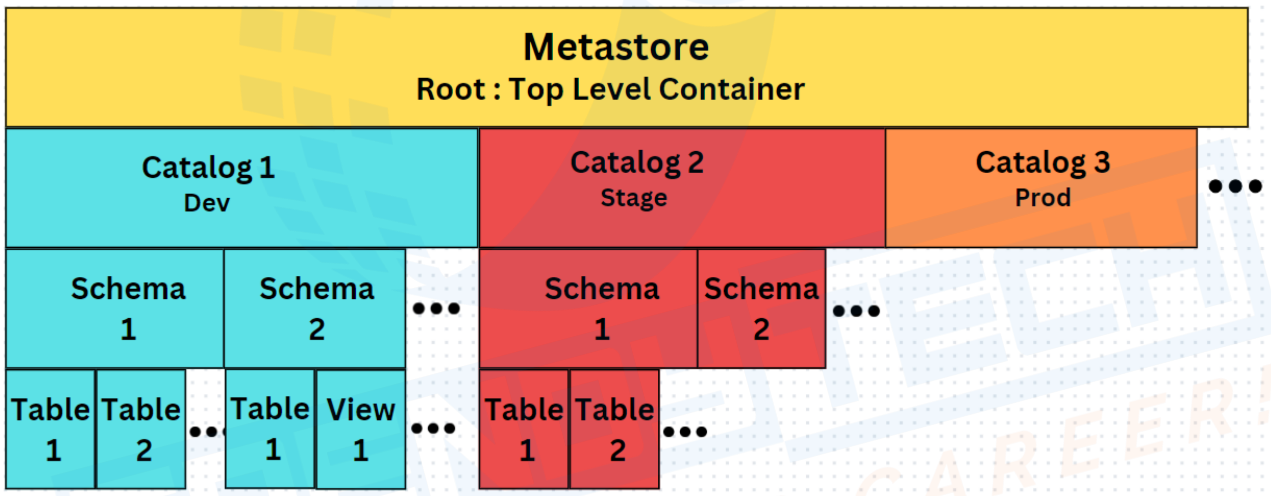
Catalog.Schema.Table/View

Note : This is used to store the metadata and data for Managed Tables.

Catalog is the first layer in the object hierarchy and can be used on logical grouping like Environments (Dev, Stage, Prod..) / Teams (Sales, Technical, HR..)

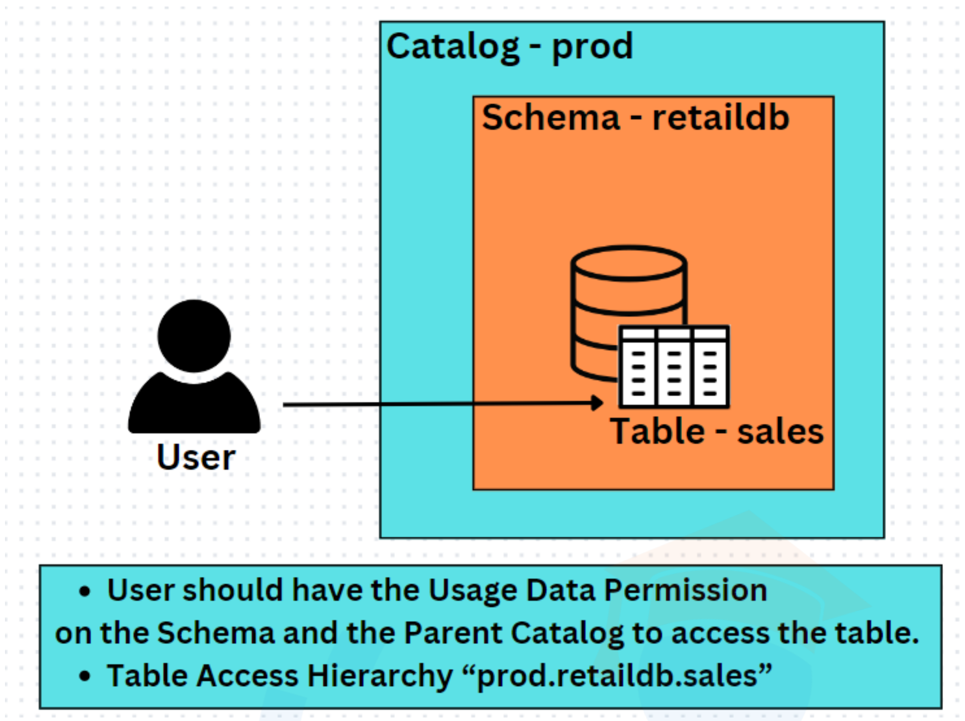
- Users can see all the Catalogs on which they have been assigned the usage date permissions.

Schema is the second layer in the object hierarchy. Schema is like a database that organises tables and views



In order to access or list the tables or views inside the schema,

- Users must have the usage data permissions on the schema and its parent catalog
- Select permission on the table.



- User should have the Usage Data Permission on the Schema and the Parent Catalog to access the table.
- Table Access Hierarchy "prod.retaildb.sales"

Table is the lowest level in the object hierarchy. Tables can be **Managed** or **External**

- In the External **Table,** data is stored in an External Location (Like a cloud storage, Ex : ADLS Gen2).

  When an external table is dropped, only the metadata is deleted and not the actual data.

  External tables are created when direct access to the data used by other tools is required.

- In the **Managed Table,** data is stored in the Root Storage.

  When a managed table is deleted, the metadata will be deleted and the underlying data will be deleted from the cloud within 30 days.

Note :

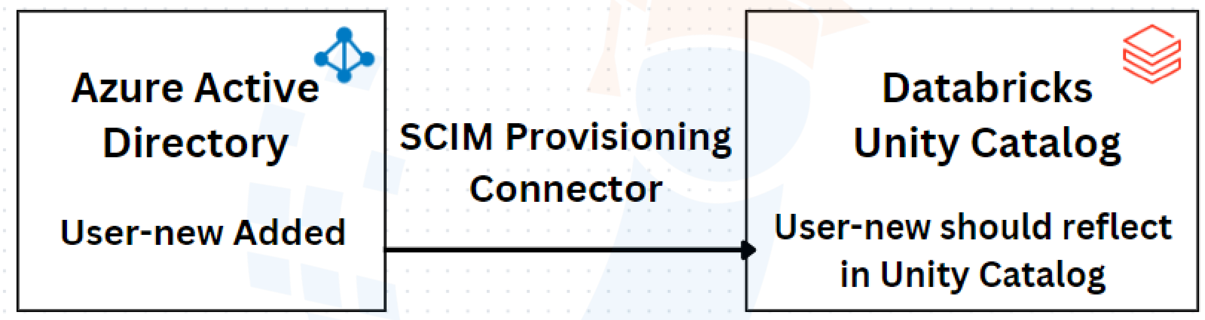
- Read-only **Views** can be created from Tables.
- In order to create a table, users must have **Create** and **Usage Permission** on the schema.
- In order to query a table, users must have select permission on the table and data usage permissions on its parent catalog & schema.

- Managed table is the default way to create tables in Unity Catalog and Delta formats are used to store the underlying data.

View is a read-only object created from one or more tables. It resides in the third level in the object hierarchy, same level as the tables.

Granting Access Rights / Assigning Roles to Users

Instead of manually adding each of the users individually, the industry standard of adding and assigning roles to the users is by syncing the user details from Azure Active Directory to Databricks Unity Catalog.


Azure Active Directory
User-new Added
SCIM Provisioning Connector
Databricks Unity Catalog
User-new should reflect in Unity Catalog

- The sync between the Azure Active Directory and the Databricks Unity Catalog is possible with the **SCIM Provisioning Connector**.
- **SCIM** is an open standard that allows for automatic user provisioning among multiple systems.

Creating a SCIM Connector

Step 1:

In the **Databricks Account** console -> Select **Settings** in the left main panel -> Click on the **User Provisioning** -> **Generate SCIM Token** -> Save the **SCIM Token** and the **SCIM URL.**

Step 2:

Navigate to the Azure Active Directory in the Azure Account -> Select **Enterprise Application** -> Create a new application (search for **Azure Databricks SCIM Provisioning Connector)** -> next click on **Provisioning** option -> Automatic Provisioning -> Enter the **SCIM URL** as the **Tenant URL** and the **SCIM Token** as the **Secret Token** -> **Test Connection** -> under the scope field, choose - **sync only assigned users and groups** option -> Turn on the **Provisioning Status** -> Assign the users manually by selecting the users to be synced.

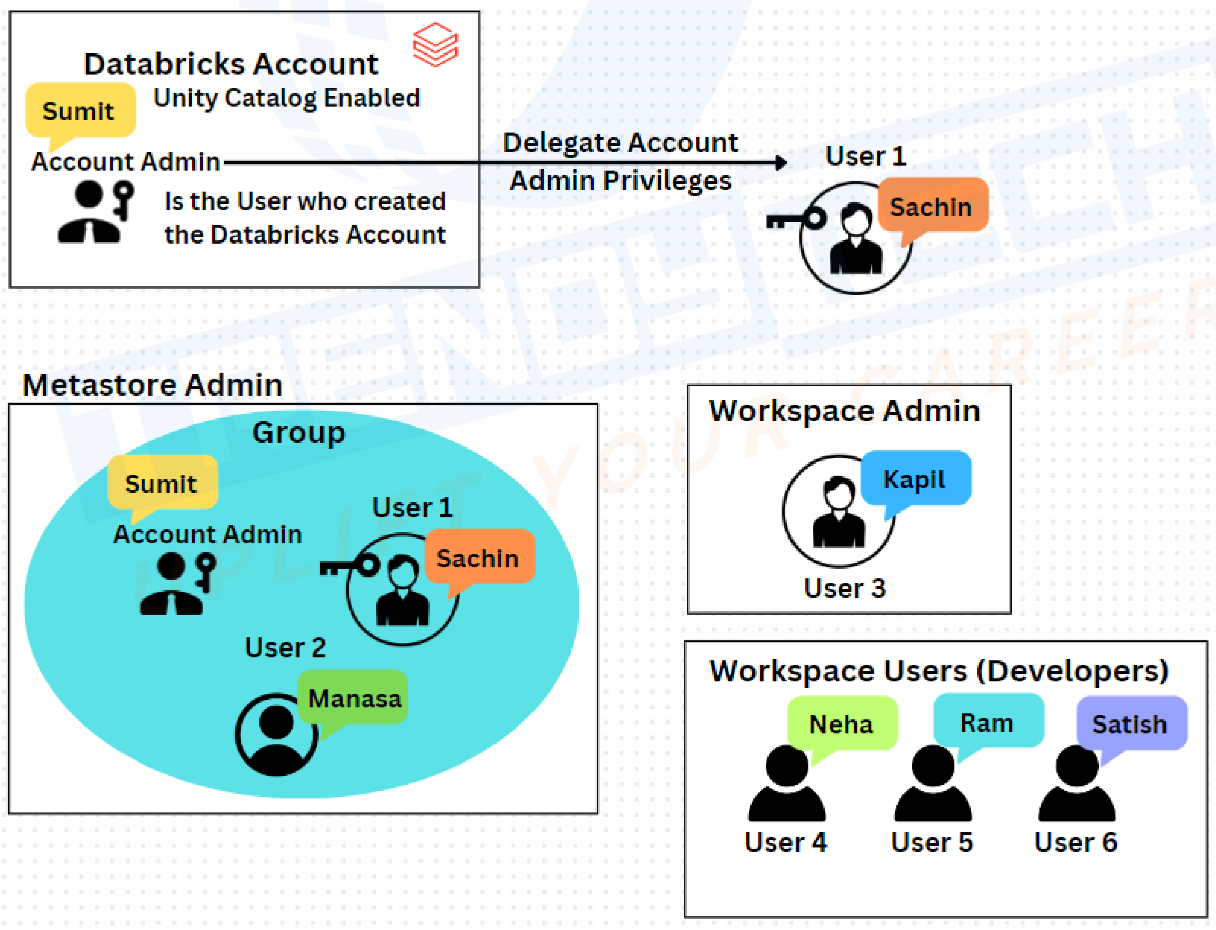
Unity Catalog Admin Roles

1. **Account Admin -** The user creating the Unity Catalog enabled Databricks is the Account Admin. This user will have access to the Account Console and can manage the accounts and the Databricks Workspaces.

   Note: Account Admin role can only be assigned to an individual and not to a Group.

2. **Metastore Admin**
3. **Workspace Admin**

Example Scenario :



Note : Workspace users are the developers who would be working on the Notebooks to develop the code. A group can be created for the Workspace users.

Account Admin can perform the following :

1. Create the Databricks Workspace.
2. Create the Metastore.
3. Link Databricks Workspace to Unity Catalog.
4. Delegate the Account Admin privileges to a New User.
5. Add Users.
6. Create Groups.
7. Create Metastore Admins.
8. Create Workspace Admins.

Associating the Databricks Workspace to the respective Users and Groups

In order to provide the necessary access permissions to the access the workspace for the respective groups, navigate to the respective Databricks Workspace and click on the **Permissions** tab -> **Add Permissions** -> In the pop-up window you can select the groups / user / service principal to be made as Admins or Normal Users/Developers.

Metastore Admin can perform the following :

1. Can create a Catalog.
2. Can create Schema and Objects(Tables & Views). Ideally Metastore Admin creates the Catalog. The Schema(Database) and the Objects inside the database are created by the developers.
3. The user who creates the catalog by default becomes the owner.
4. Can grant the permissions to access the objects in metastore for the developer groups.

Workspace Admin can perform the following :

1. Can create a New Cluster / Compute Resources.
2. Can access the Workspace Admin Console to Add Users and grant necessary permissions.
3. Can Add Groups in the local workspace.
4. Can access the Global init scripts that should be executed on all the nodes of the clusters launched from the Workspace.

5. Workspace admin can grant permissions like - sharing Notebooks / Clusters among workspace users through the Access Control options under the Workspace Settings tab.
6. Workspace admins can grant 3 different types of execution permissions to developers to execute the code (Developers don't have the option to create a cluster by default)

3 modes to execute the code on the cluster by developers

a. **Unrestricted Cluster Creation Policy**

   With this permission, the developer will be able to create a cluster of their choice. (Not Recommended)

b. **Shared Cluster**

   A shared cluster will be created by Workspace Admin and the developers can attach the Notebooks to the Shared Cluster to execute the code. By granting the permission **Can Attach To**, the workspace admin can allow the developers to attach the Notebooks to the cluster for execution.

c. **Restricted Cluster Creation Policy**

   In this case, the workspace admin grants the cluster creation permission to the developers but with restrictions on the compute resources. (**Restricted policies - Personal Compute, Shared Compute, Power User Compute, Job Compute**)

Workspace User can perform the following :

1. Can create a new Notebook that can be executed on the existing cluster. By default workspace users don't have the permissions to create a new cluster / compute resource.
2. Can create a cluster if the workspace admin grants the necessary permissions.
3. If the Workspace users are granted access to a shared cluster, they can only attach the Notebooks to the cluster for execution. However, developers cannot start / stop the cluster.