Unity Catalog in Azure DataBricks

Unity Catalog is a centralized data catalog that provides access control, auditing, lineage, quality monitoring, and data discovery capabilities across Azure Databricks workspaces.

It is a centralized data governance solution in Databricks that manages data access, organization, lineage, and compliance across multiple workspaces and data sources.

Why to use Unity Catalog?

Unity Catalog provides **centralized, fine-grained governance** for all data and AI assets in Databricks.

It solves limitations of legacy hive metastore which is workspace-specific and provides limited security.

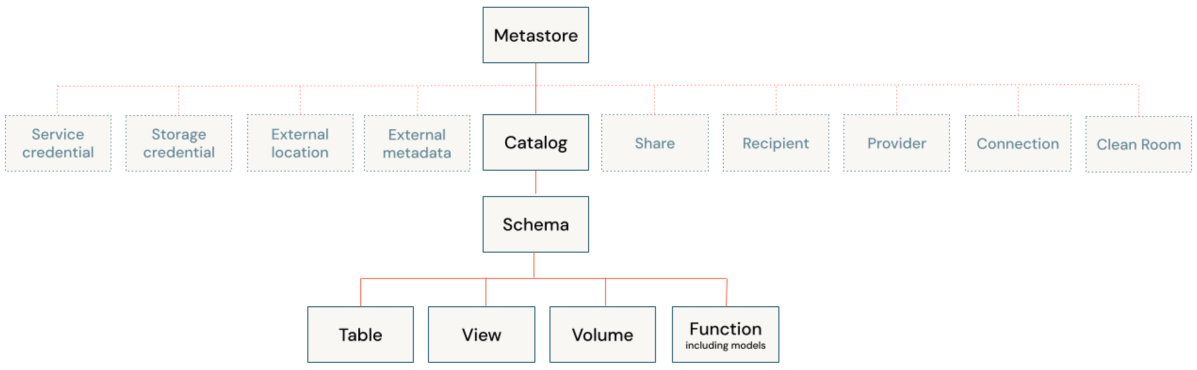
Key features of Unity Catalog include:

- **Define once, secure everywhere**: Unity Catalog offers a single place to administer data access policies that apply across all workspaces in a region.
- **Standards-compliant security model**: Unity Catalog's security model is based on standard ANSI SQL and allows administrators to grant permissions in their existing data lake using familiar syntax.
- **Built-in auditing and lineage**: Unity Catalog automatically captures user-level audit logs that record access to your data. Unity Catalog also captures lineage data that tracks how data assets are created and used across all languages.
- **Data discovery**: Unity Catalog lets you tag and document data assets, and provides a search interface to help data consumers find data.
- **System tables**: Unity Catalog lets you easily access and query your account's operational data, including audit logs, billable usage, and lineage.

MetaStore :

The metastore is the top-level container for metadata in Unity Catalog. It registers metadata about data and AI assets and the permissions that govern access to them. Unity Catalog metastore runs in a multi-tenant environment and represents a logical boundary for the segregation of data by region for a given Azure Databricks account.

Data Organization / Architecture in Unity Catalog :



1. **Catalogs (Level 1)** –

   Catalogs are used to organize data assets and are typically used as Top level in data isolation scheme.

   Each catalog typically has its own *managed storage location* to store managed tables and volumes, providing physical data isolation at the catalog level.

   For example, catalog owners have all privileges on the catalog and the objects in the catalog, and they can grant access to any object in the catalog. Users with SELECT on a catalog can read any table in the catalog. Users with CREATE TABLE on a catalog can create a table in any schema in the catalog.

   It Represents a collection of schemas (databases).

2. **Schemas (Level 2) –**

   Schemas (also known as databases) contain tables, views, volumes, AI models, and functions. Schemas organize data and AI assets into logical categories that are more granular than catalogs.

   In Unity Catalog, a schema is a child of a catalog and can contain tables, views, volumes, models, and functions. Schemas provide more granular categories of data organization than catalogs.

   Schemas Represent a logical grouping of tables and views.

3. **Tables (Level 3) –**
   Tables are collections of data organized by rows and columns.
   A table resides in a schema and contains rows of data. The default table type created in Azure Databricks is a Unity Catalog managed table

   Tables can be either *managed*, with Unity Catalog managing the full lifecycle of the table, or *external*, with Unity Catalog managing access to the data from within Azure Databricks

   Views are saved queries against one or more tables.

   Volumes represent logical volumes of data in cloud object storage. Typically they are used for non-tabular data.

   Functions are units of saved logic that return a scalar value or set of rows.

   Referencing data with 3-level namespace :
   <catalog_name>.<schema_name>.<table_or_view_name>
   For e.g : sales_catalog.customer_data.customer_transactions

Unity Catalog uses a **3-level namespace**: catalog.schema.table

- **Catalog** → Top-level container for schemas (e.g., main, sales_data)
- **Schema** → Group of tables/views (like a database)
- **Table/View** → Actual datasets
  Example: main.sales.transactions

Unity Catalog = One place to organize, secure, and track all data in Databricks.