Unity Catalog :

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

Unity Catalog provides capabilities for interoperability with different lakehouse formats (like Apache Iceberg and Delta Lake) and provides secured access to data stored in Unity catalog in controlled manner.

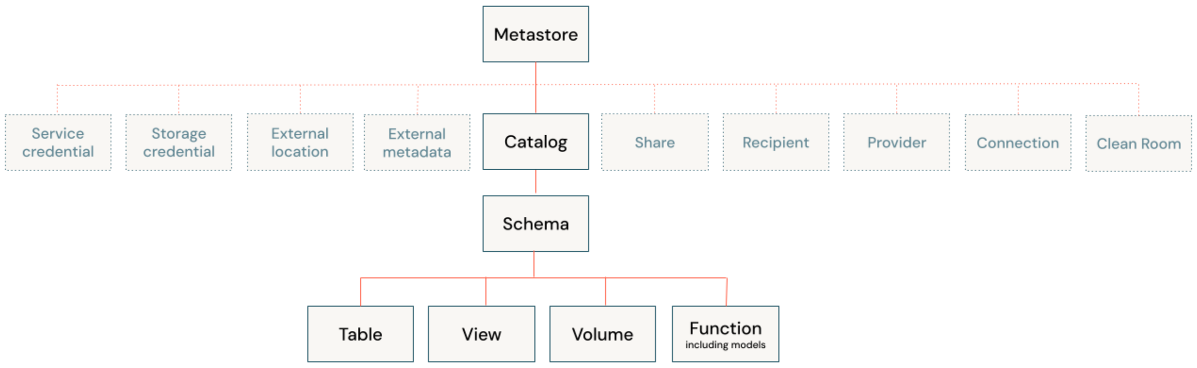
Why use Unity Catalog in Databricks?

The Databricks Unity Catalog is a centralized data governance layer that allows for granular security control and managing data and metadata assets in a unified system within Databricks. Additionally, the unity catalog provides tools for access control, audits, logs and lineage.

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.

- ## The Unity Catalog object model



In a Unity Catalog metastore, the three-level database object hierarchy consists of catalogs that contain schemas, which in turn contain data and AI objects, like tables and models. This hierarchy is represented as a three-level namespace (catalog.schema.table-etc) when you reference tables, views, volumes, models, and functions.

Level one:

- **Catalogs** are used to organize your data assets and are typically used as the top level in your data isolation scheme. Catalogs often mirror organizational units or software development lifecycle scopes.
- **Non-data securable objects**, such as storage credentials and external locations, are used to manage your data governance model in Unity Catalog. These also live directly under the metastore.

Level two:

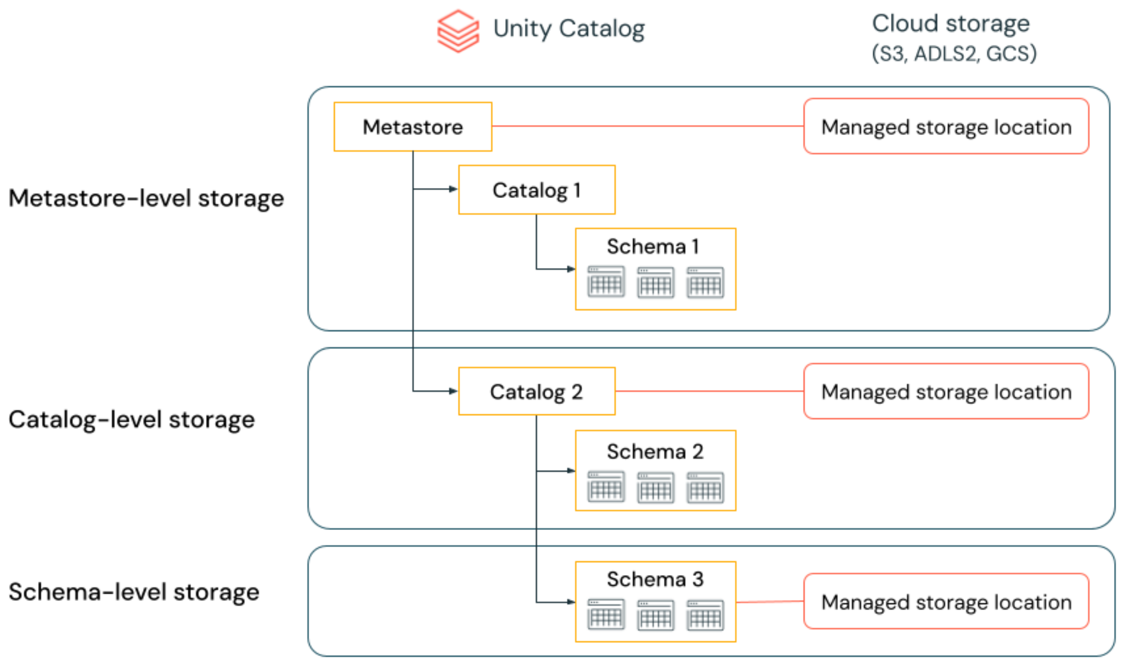
- **Schemas** (also known as databases) contain tables, views, volumes, AI models, and functions. Schemas organize data and AI assets into logical categories that are more granular than catalogs. Typically a schema represents a single use case, project, or team sandbox.

Level three:

- **Tables** are collections of data organized by rows and columns. 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, but not managing access to the data in cloud storage from other clients.

- **Views** are saved queries against one or more tables.
- **Volumes** represent logical volumes of data in cloud object storage. You can use volumes to store, organize, and access files in any format, including structured, semi-structured, and unstructured data. Typically they are used for non-tabular data. Volumes can be either *managed*, with Unity Catalog managing the full lifecycle and layout of the data in storage, or *external*, with Unity Catalog managing access to the data from within Azure Databricks, but not managing access to the data in cloud storage from other clients.
- **Functions** are units of saved logic that return a scalar value or set of rows.
- **Models** are AI models packaged with MLflow and registered in Unity Catalog as functions.

Managed storage location hierarchy


Unity Catalog
Cloud storage
(S3, ADLS2, GCS)
Metastore-level storage
Metastore
Managed storage location
Catalog 1
Schema 1
Catalog-level storage
Catalog 2
Managed storage location
Schema 2
Schema-level storage
Schema 3
Managed storage location