**Unity Catalog – Mini Project Documentation**

**For Coding Challenge**

**What is Unity Catalog?**

Unity Catalog is a centralized governance solution for all data and AI assets in the Databricks Lakehouse Platform. It provides a standardized way to organize, secure, and audit data across multiple workspaces and cloud environments.

Unity Catalog introduces a three-level namespace (Catalog → Schema → Table) and supports fine-grained access control, lineage, and auditing.

**Why Do We Need Unity Catalog?**

Without a unified governance system, enterprises face challenges:

* Inconsistent data access policies across teams.
* Difficulty in auditing and compliance.
* No visibility into data lineage.
* Security risks from unmanaged storage.

Unity Catalog addresses these issues by:

* Offering centralized governance across all workspaces.
* Ensuring granular, role-based access control.
* Enabling lineage tracking and auditing.
* Supporting compliance frameworks

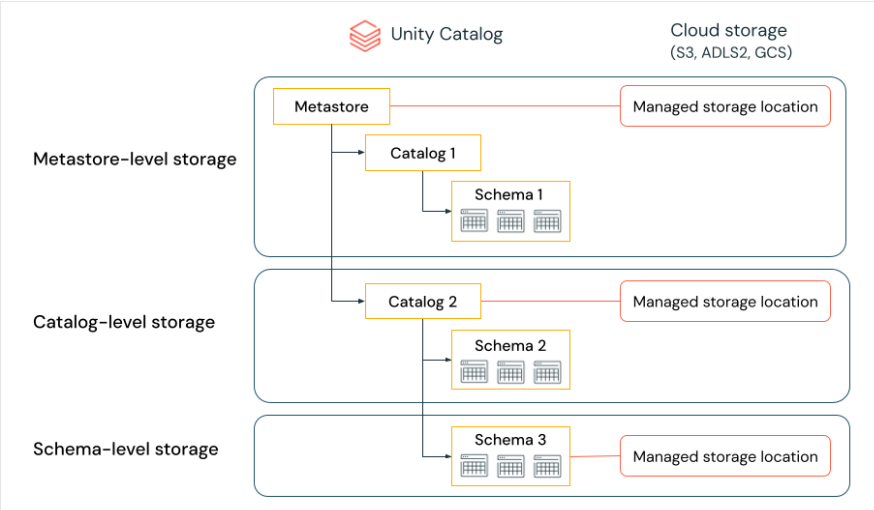
**Features of Unity Catalog**

* Centralized Metadata – one metastore for multiple workspaces.
* Hierarchical Namespace – catalogs, schemas, and tables.
* Access Control – permissions down to row/column level.
* Data Lineage – tracks how data is created, transformed, and consumed.
* Audit Logging – records all operations for compliance.
* Multi-Cloud Integration – works with ADLS, S3, and GCS.
* Shared Governance – applies consistent policies across teams.

**Storage Levels in Unity Catalog**

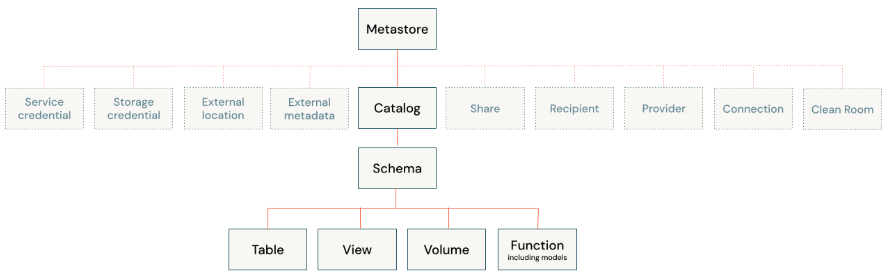
Unity Catalog organizes data at different levels of storage:

* Metastore-level storage: Centralized metadata repository for all catalogs.
* Catalog-level storage: Logical grouping of schemas under a catalog.
* Schema-level storage: Holds related tables, views, and functions.

****

**Unity Catalog Object Model**

The object model defines how data assets are structured:

****

**How to Organize Data in Unity Catalog**

Data follows a three-level namespace:

* Catalog → Top-level container for organizing schemas.
* Schema → Grouping of related tables and views (like a database).
* Table/View → Data assets that store actual records.

**Example SQL:**

#Create a catalog

CREATE CATALOG Sales\_Catalog;

#Create a schema inside the catalog

CREATE SCHEMA Sales\_Catalog.Retail\_Data;

#Create a table inside the schema

CREATE TABLE Sales\_Catalog.Retail\_Data.Transactions (

transaction\_id INT,

customer\_id INT,

amount DOUBLE,

date DATE

);

**Benefits of Unity Catalog**

* Centralized data governance.
* Fine-grained access controls.
* Simplifies compliance and auditing.
* Ensures consistent organization across teams.
* Improves security and reduces risk.
* Provides data lineage for better understanding of workflows.

**Example Use Case**

**Scenario:**

* **Sales Team** → needs full access to transaction data.
* **Analyst Team** → needs only read permissions.
* **Admin Team** → manages all data and policies.

**SQL Permissions Example:**

GRANT SELECT ON TABLE Sales\_Catalog.Retail\_Data.Transactions TO analyst\_role;

GRANT INSERT ON TABLE Sales\_Catalog.Retail\_Data.Transactions TO sales\_role;

GRANT ALL PRIVILEGES ON SCHEMA Sales\_Catalog.Retail\_Data TO admin\_role;

**Conclusion**

Unity Catalog provides a single, secure, and centralized governance layer for the Lakehouse platform. With features like hierarchical namespaces, fine-grained permissions, lineage, and audit logging, it helps organizations ensure compliance, data security, and efficient collaboration.

By organizing data into catalogs, schemas, and tables, Unity Catalog makes data easy to discover, govern, and use responsibly across multiple teams and cloud environments.