**Unity Catalog in Databricks**

**1. Introduction**

Unity Catalog is a **unified governance solution** for all data and AI assets in Databricks.  
It provides:

* **Centralized access control**
* **Data lineage**
* **Audit logging**
* **Data discovery**

It works **across multiple Databricks workspaces** within the same account.

**2. Why Unity Catalog?**

Before Unity Catalog:

* Access control was **workspace-level**.
* Hard to manage permissions across multiple workspaces.
* No built-in **lineage** or **central governance**.

With Unity Catalog:

* Access control is **centralized at the metastore level**.
* One set of **data security rules** applies across all workspaces.
* Tracks **who accessed what data** and **when**.
* Allows **cross-workspace data sharing**.

**3. Key Components**

Unity Catalog introduces a **3-level namespace**:

**<catalog>.<schema>.<table>**

| **Level** | **Description** |
| --- | --- |
| **Catalog** | Top-level container for schemas. Represents a group of data assets. |
| **Schema** | Similar to a database; contains tables, views, and functions. |
| **Table/View** | The actual data object you query. |

Example:

**SELECT \* FROM finance\_db.customers.transactions;**

Here:

* finance\_db → **Catalog**
* customers → **Schema**
* transactions → **Table**

**4. Unity Catalog Metastore**

A **metastore** stores metadata about data assets (location, permissions, schema).

**Steps to Create Unity Catalog Metastore:**

1. In the **Databricks account console**, go to **Data** → **Metastores**.
2. Click **Create Metastore**.
3. Provide:
   * **Name** (e.g., MainMetastore)
   * **Region** (same as workspace region)
   * **S3 bucket path** (for managed storage)
   * **IAM role** (for AWS access)
4. Click **Create**.

**5. Enabling Unity Catalog for a Workspace**

1. In the **Account Console**, open the **Workspaces** page.
2. Select your workspace.
3. Click **Assign Metastore**.
4. Choose your newly created **Unity Catalog Metastore**.
5. Click **Assign**.

Now, the workspace can use Unity Catalog objects.

**6. Creating Unity Catalog Objects**

**6.1 Create a Catalog**

**CREATE CATALOG finance\_db**

**COMMENT 'Catalog for finance-related data';**

**6.2 Create a Schema**

**CREATE SCHEMA finance\_db.customers**

**COMMENT 'Contains customer-related tables';**

**6.3 Create a Table**

**CREATE TABLE finance\_db.customers.transactions (**

**transaction\_id STRING,**

**customer\_id STRING,**

**amount DOUBLE,**

**date DATE**

**)**

**USING DELTA**

**LOCATION 's3://my-bucket/finance/transactions/';**

**7. Permissions in Unity Catalog**

* Uses **ANSI SQL GRANT/REVOKE**.
* Permissions can be applied at **catalog**, **schema**, or **table** level.

Example:

**-- Grant SELECT on a table**

**GRANT SELECT ON TABLE finance\_db.customers.transactions TO `data\_analyst`;**

**-- Grant USAGE on a schema**

**GRANT USAGE ON SCHEMA finance\_db.customers TO `analyst\_group`;**

**8. Data Lineage**

Unity Catalog tracks:

* **Which queries** read/write data.
* **Who ran them**.
* **When they were executed**.

Example:  
If a table is updated via:

**INSERT INTO finance\_db.customers.transactions**

**SELECT \* FROM staging.transactions\_raw;**

The lineage graph will show:

* staging.transactions\_raw as the source.
* finance\_db.customers.transactions as the target.

**9. Benefits Summary**

* ✅ **Centralized Governance** – Manage permissions for all workspaces from one place.
* ✅ **Cross-workspace Data Access** – Share data without duplication.
* ✅ **Lineage Tracking** – Full visibility into data usage.
* ✅ **Secure & Auditable** – Integrated with audit logs.

**10. Example Workflow**

1. **Admin** creates MainMetastore.
2. **Assigns** metastore to Finance Workspace.
3. **Creates** catalog finance\_db.
4. **Creates** schema customers.
5. **Creates** Delta table transactions.
6. **Grants** SELECT to analysts group.
7. **Analysts** query:

**SELECT \* FROM finance\_db.customers.transactions**

**WHERE amount > 1000;**