Unity Catalog

Bavatharani S

1. Introduction

Unity Catalog is Databricks' centralized data governance solution designed to manage data and AI assets across multiple workspaces and cloud environments.

It provides a unified interface for managing permissions, access control, and data lineage. With Unity Catalog, organizations can ensure security, compliance, and transparency for all their data assets in the modern data lakehouse.

2. Objectives

- Centralize governance across all Databricks workspaces.
- Provide fine-grained access control for users and groups.
- Enable end-to-end data lineage tracking.
- Improve security and compliance with detailed audit logs.
- Support secure data sharing across teams and organizations.

3. Architecture Overview

Unity Catalog follows a three-level namespace model that organizes data assets:

1. Metastore – The top-level container that holds catalogs and schemas, assigned to a workspace.
2. Catalog – A logical grouping of schemas, usually by department or project.
3. Schema – Organizes tables, views, and functions within a catalog.
4. Tables/Views – The actual structured data objects.

This architecture ensures logical organization, consistency, and governance across different environments.

4. Key Features

- Centralized Governance – Manage permissions in a single place for all workspaces.

- Data Lineage – Track how data flows and transforms across pipelines.
- Secure Sharing – Share datasets safely across accounts and organizations.
- Attribute-Based Access Control (ABAC) – Grant permissions based on user attributes.
- Audit Logging – Record all access requests, permission changes, and usage activities.
- Multi-Cloud Support – Works across AWS, Azure, and GCP with consistent governance.

5. Unity Catalog Components

Component	Description
Metastore	Container for catalogs and schemas, assigned to Databricks workspaces.
Catalog	Logical grouping for schemas, often aligned with departments or projects.
Schema	Organizes tables, views, and functions within a catalog.
Table/View	Stores actual data (tables) and query definitions (views).
External Location	Points to external storage (S3, ADLS, or GCS).
Storage Credential	Provides secure authentication to access external storage.

6. Workflow of Unity Catalog

1. Create and assign a Metastore to a workspace.
2. Configure Storage Credentials to securely access cloud storage.
3. Define External Locations for linking external data.
4. Create Catalogs, then add Schemas inside them.
5. Create Tables and Views within schemas.
6. Assign permissions to users and groups.
7. Track data lineage and review audit logs for compliance.

7. Example SQL Commands in Unity Catalog

```
-- Create a new catalog
CREATE CATALOG sales_catalog;
-- Create a schema inside catalog
CREATE SCHEMA sales_schema;
-- Create a table
CREATE TABLE sales_schema.orders (
  order_id INT,
  customer STRING,
  amount DOUBLE
);
-- Grant select permissions to an analyst
GRANT SELECT ON TABLE sales_schema.orders TO `data_analyst`;
```

8. Best Practices

- Organize catalogs by department, project, or business unit.
- Use groups and roles instead of assigning permissions to individual users.
- Apply service principals for automated workloads.
- Enable audit logging to meet compliance requirements.
- Regularly review permissions to ensure least privilege access.
- Maintain clear naming conventions for catalogs, schemas, and tables.

9. Conclusion

Unity Catalog simplifies and strengthens data governance in Databricks by centralizing permissions, enforcing security, and tracking data lineage.

It enables organizations to manage data consistently across multi-cloud and multi-workspace environments, ensuring both compliance and collaboration.

With its fine-grained access controls, secure sharing capabilities, and strong governance features, Unity Catalog is an essential tool for enterprises adopting the lakehouse architecture.