**Akilesh K**

k.akilesh123@gmail.com

**Data engineering - Batch 1**

**Date: 19-02-24**

# DAY 21 – Azure Databricks-Unity catalog

## Unity Catalog

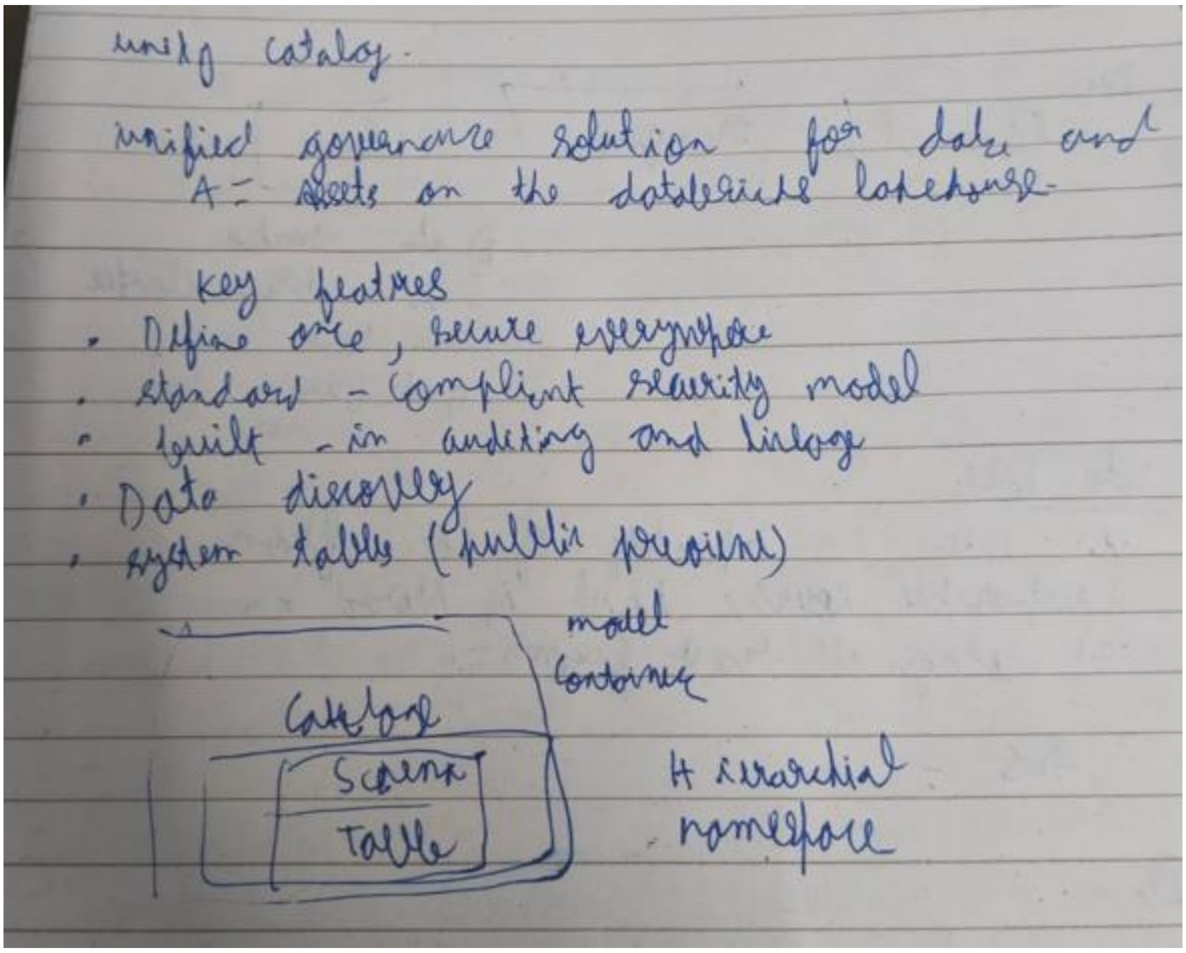
Unity Catalog is a comprehensive solution for managing data access policies and administering data assets within an organization. Key features of Unity Catalog include:

- Define once, secure everywhere
- Standards-compliant security model
- Built-in auditing and lineage
- Data discovery
- System tables

Unity Catalog streamlines data access management, ensures compliance with security standards, facilitates data discovery, and provides insights into data usage and lineage, making it a valuable tool for organizations seeking to effectively manage and utilize their data assets.

Unity Catalog introduces the following concepts to manage relationships between data in Azure Databricks and cloud object storage:

- Storage credentials.
- External locations
- Managed storage locations
- Volumes
- Tables.



unity catalog.

unified governance solution for data and AI assets on the databricks lakehouse.

key features

- Define once, secure everywhere
- standard - compliant security model
- built - in auditing and lineage
- Data discovery
- system tables (public preview)

model
container
Catalog
Schema
Table
Hierarchial
namespace

In Unity Catalog, the hierarchy of primary data objects flows from metastore to table or volume:

- Metastore
- Catalog
- Schema
- Tables, views, and volumes
- Models

**Metastores:**

- Top-level container of objects in Unity Catalog.
- Registers metadata about data and AI assets, along with permissions.
- Each Azure Databricks workspace must have a metastore attached to use Unity Catalog.
- Can optionally be configured with managed storage in Azure Data Lake Storage Gen2.

**Catalogs:**

- First layer of Unity Catalog's namespace, organizing data assets.
- Users can see catalogs they've been assigned USE CATALOG permission on.
- Default permissions may be granted on automatically provisioned catalogs.
- Used to create and manage schemas.

**Schemas:**

- Second layer of Unity Catalog's namespace, organizing tables and views.
- Users can see schemas they've been assigned USE SCHEMA permission on.
- Default schema named default accessible to all users.
- Used to create and manage tables.

**Tables:**

- Third layer of Unity Catalog's namespace, containing rows of data.
- Managed tables are managed by Unity Catalog, while external tables are not.
- Users require appropriate permissions to create and query tables.
- Tables can be stored in various file formats and accessed using different permissions.

**Views:**

- Read-only objects created from tables and views in a metastore.
- Can be created from multiple schemas and catalogs.
- Dynamic views enable row- and column-level permissions.

**Volumes:**

- Siblings to tables, views, and other objects under a schema.
- Contain directories and files for data stored in any format.
- Managed volumes are managed by Unity Catalog, while external volumes are not.
- Used to provide non-tabular access to data, with permissions for reading and modifying files.

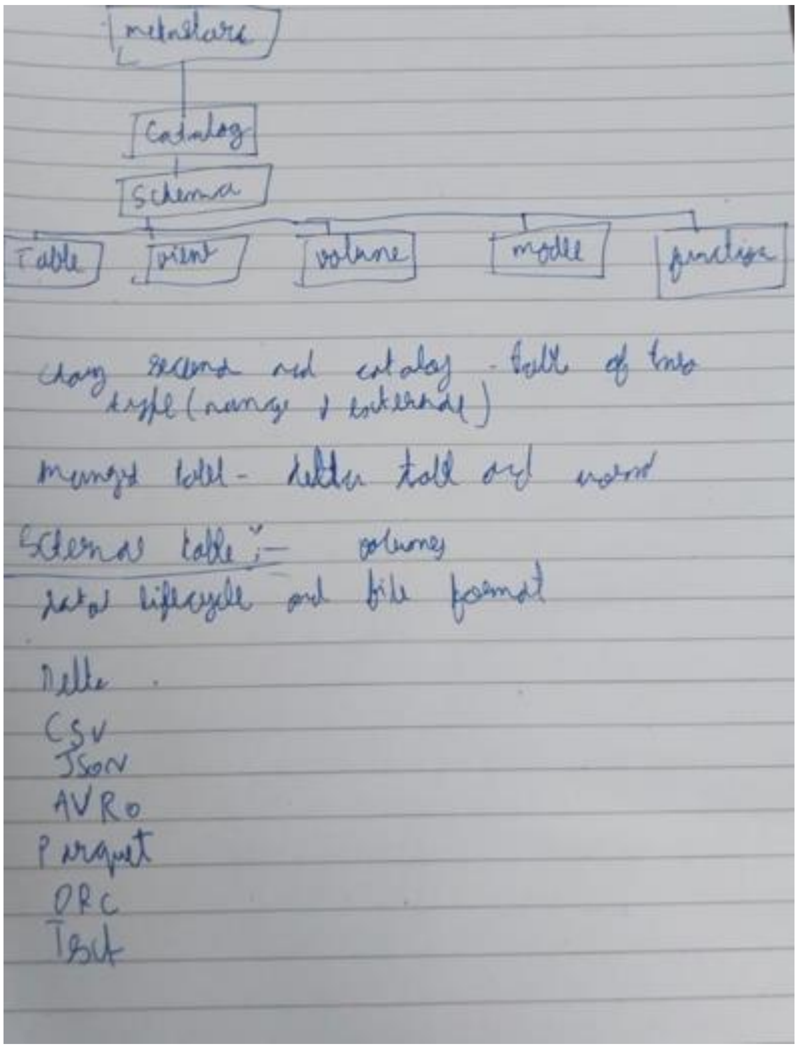
**Models:**

- Refers to machine learning models registered in the MLflow Model Registry.
- Reside in the third layer of Unity Catalog's namespace.

- Users require appropriate privileges to create and access models.

**Managed Storage:**

- Allows storing managed tables and volumes at various levels in the Unity Catalog object hierarchy.
- Storage at lower levels overrides storage defined at higher levels.
- Managed storage locations cannot overlap with external tables or volumes.
- Optional to assign managed storage at metastore, catalog, or schema levels.


metastore
Catalog
Schema
Table
view
volume
model
function

chang scema and catalog - table of two type (managed & external)

managed table - delete table and record

Schemas table :- volumes

data lifecycle and file format

Delta

CSV

JSON

AVRO

Parquet

ORC

Text

**Create a catalog**

To create a catalog, use the CREATE CATALOG command with spark.sql. You must be a metastore admin or user with the CREATE CATALOG privilege on the metastore to create a catalog. If your workspace was enabled for Unity Catalog by default, then workspace admins have the CREATE CATALOG privilege by default

If your workspace was enabled for Unity Catalog by default, then there may be no managed storage location for the metastore, and you must create a location for the new catalog.

- Create a catalog.

  spark.sql("CREATE CATALOG IF NOT EXISTS quickstart_catalog")

- Set the current catalog.

  spark.sql("USE CATALOG quickstart_catalog")

- Show all catalogs in the metastore.

  display(spark.sql("SHOW CATALOGS"))

- Grant create and use catalog permissions for the catalog to all users on the account.

  spark.sql("""GRANT CREATE, USE CATALOG ON CATALOG quickstart_catalog

  TO `account users`""")

- Show grants on the quickstart catalog.

  display(spark.sql("SHOW GRANT ON CATALOG quickstart_catalog"))

**Create and manage schemas (databases)**

Schemas, also called databases, are the second level of the Unity Catalog three-level namespace. Schemas logically organize tables and views.

- Create a schema in the catalog that was set earlier.

  spark.sql("""CREATE SCHEMA IF NOT EXISTS quickstart_schema

  COMMENT 'A new Unity Catalog schema called quickstart_schema'""")

- Show schemas in the catalog that was set earlier.

  display(spark.sql("SHOW SCHEMAS"))

- Describe the schema.

  display(spark.sql("DESCRIBE SCHEMA EXTENDED quickstart_schema"))

- Grant create table, and use schema permissions for the schema to all users on the account.

  spark.sql(""" GRANT CREATE TABLE, USE SCHEMA ON SCHEMA quickstart_schema

  TO `account users`""")

**Create a managed table**

Managed tables are the default way to create table with Unity Catalog. The table is created in the managed storage location configured for the metastore, catalog, or schema.

- Set the current schema.

  spark.sql("USE quickstart_schema")

- Show the current database (also called a schema).

  spark.catalog.currentDatabase()

- Create a managed Delta table in the catalog that was set earlier.

  spark.sql("CREATE OR REPLACE TABLE quickstart_table (id STRING)")

- Grant select and modify permissions for the table to all users on the account.

  spark.sql("""GRANT SELECT, MODIFY ON TABLE quickstart_table TO `account users`""")

- List the available tables in the catalog that was set earlier.

  display(spark.sql("SHOW TABLES"))

- Insert 10 rows into the table.

  spark.range(10).selectExpr("id").write.insertInto("quickstart_table")

- Show the table.

  display(spark.table("quickstart_table"))

**Setting up a workflow in Azure Databricks**

- Create Spark notebooks or jobs within the Databricks workspace to develop your data processing pipelines, machine learning models, or any other tasks.
- Use the Databricks Jobs feature to schedule the execution of your notebooks or jobs.
- Specify the schedule (e.g., daily, hourly, etc.) and recurrence pattern for each job.
- Optimize your Databricks clusters and jobs for performance and cost efficiency.
- Document your workflow, including data sources, transformations, and outputs.