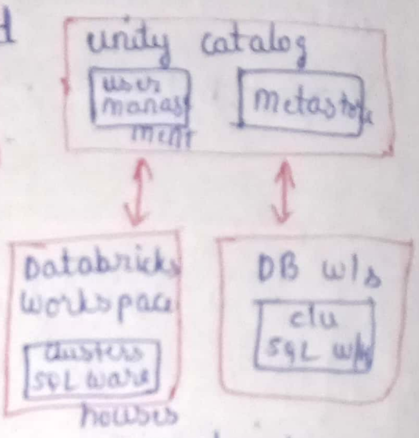
19/2/24

Unity Catalog :-

* a unified goverance solution for Data on the Databricks lakehouse (data lake + data warehouse)

* Unity catalog provides centralized access control, auditing, → monitor in detail lineage, ↱ see, the data flow for any workload & data discovery capabilities across Azure : databricks workspaces.



* mainly deals with user management, metastore

key features :-

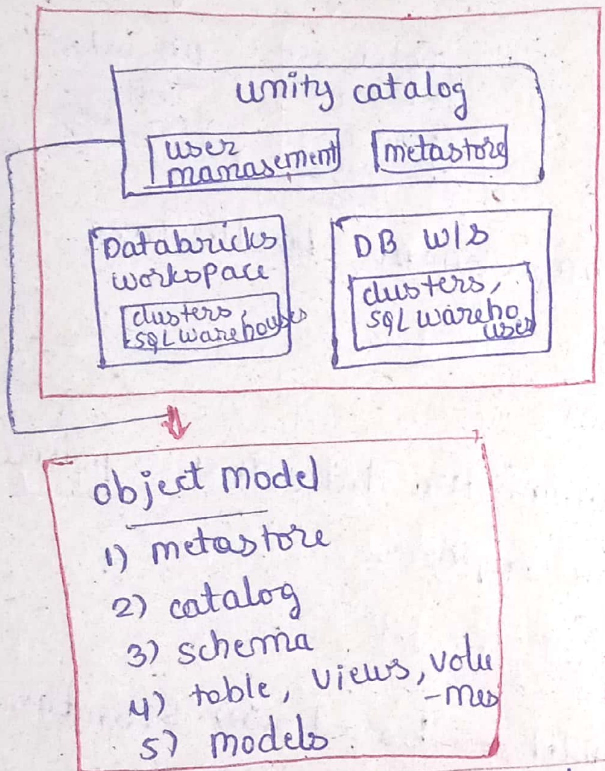
1) Define once, secure anywhere :-
offers single place to administer data access policies that apply across all workspaces.

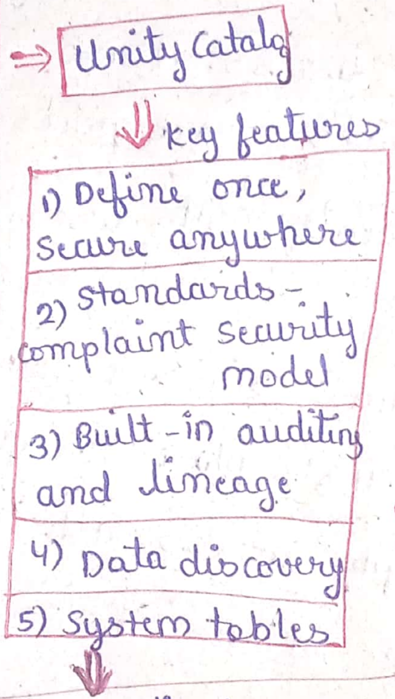
2) Standards - complaint security model
* Unity Catalog security model ← based on standard ANSI SQL (somewhat diff from mySQL) & allows administrators to grant permissions in existing datalake like databases, tables, views.

3) Built-in auditing and lineage.
* UC automatically captures user-level audit logs that record access to data
* UC ~~captuall~~ captures lineage data that tracks how data assets are created & used across all languages

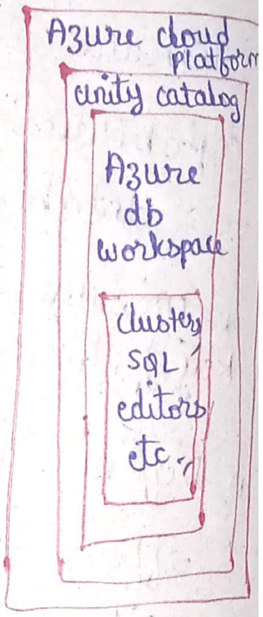
4) Data discovery :- provides search ~~intef~~ interface to help data consumers find data.

5) System tables (Public preview)
allows to easily access & query data including audi
logs, billable usage & lineage.





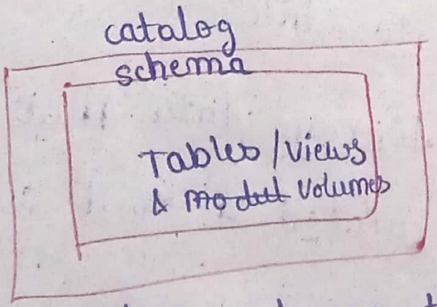
Unity Catalog
key features
1) Define once, secure anywhere
2) standards - complaint security model
3) Built-in auditing and lineage
4) Data discovery
5) System tables



unity catalog provides centralized access control, auditing, lineage & data discovery capabilities across Azure dbricks workspace

~~How~~ Unity catalog govern access to data in cloud object storage

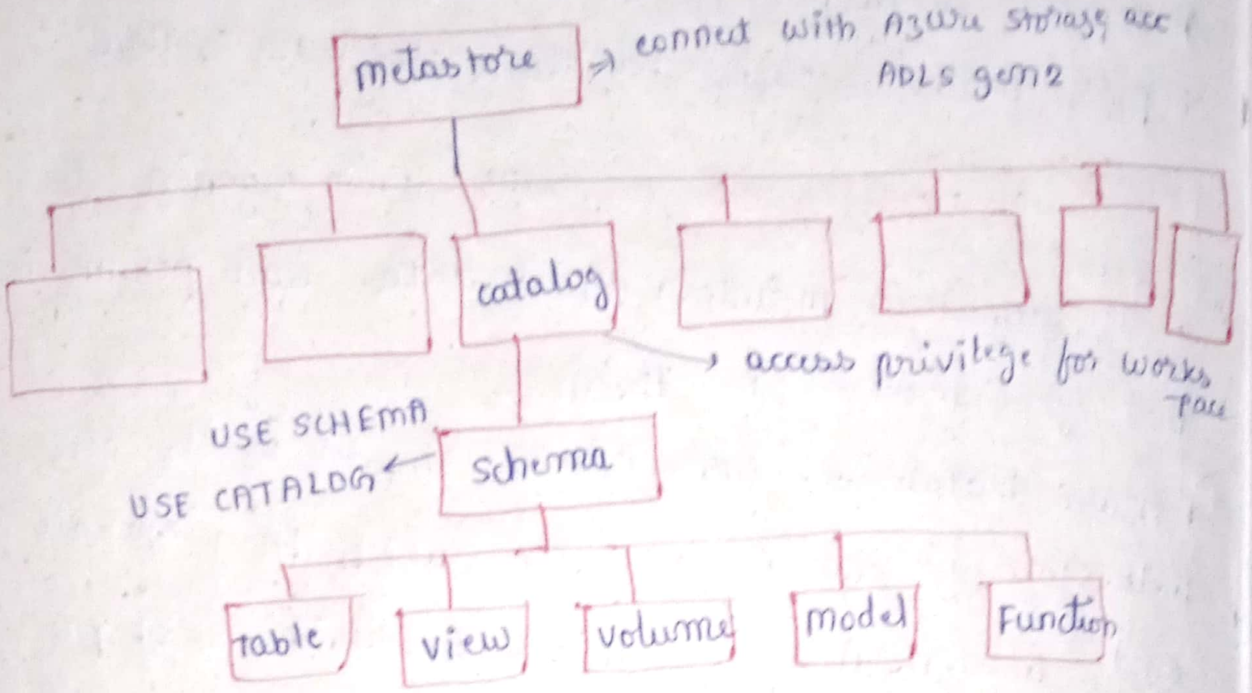
cloud obj storage => storage / acc in Azure services



catalog . schema . table => Heirarchical namespace
metastore

Unity catalog object model

non-tabular data
⇓
stored in volumes


metastore
⇒ connect with Azure storage acc / ADLS gen2
catalog
→ access privilege for workspace
USE SCHEMA
USE CATALOG
schema
table
view
volume
model
Function

Hierarchy of primary data objects flows from metastore to table or volume.

1) ~~metastore~~ :-
 * ~~top-level~~

* change schema & catalog - table & two types ↳ managed ↳ external
* Externally managed data has file layout (or) file formats
  - (i) DELTA
  - (ii) CSV
  - (iii) JSON
  - (iv) AVRO
  - (v) PARQUET
  - (vi) ORC
  - (vii) TEXT

Metastore :-

① Metastore :-

* top level container for meta data (or) objects.

* Each metastore exposes a three - level namespace

⇓

catalog . schema . table ⇒ will organize data

* It registers metadata about data and permission that govern access to them.

* Azure Databricks acc admin should create one meta store for each region in which they operate and assign them to Azure Databricks workspaces in same region.

* For workspace to use unity catalog, it must have unity catalog metastore attached.

* this metastore is distinct from Hive metastore in Azure Databricks workspaces that have not been enabled for unity catalog.

② Catalogs

* first layer of Unity catalog's three - level namespace

* used to organize data assets.

* USE CATALOG ⇒ to see all catalogs

③ Schemas

* also called database

* second layer of Unity catalog's three - level namespace.

* schema organizes tables & views

* USE SCHEMA Permission } to see all the schemas
  USE CATALOG " on which they have been
  assigned on parent catalog

* SELECT => to access / list a table (or) view.
* If workspace was enabled for Unity catalog manually
  it includes default schema named default in the main
  catalog => accessible to all in workspace.

④ Tables

* Third layer of Unity catalog's three-level namespace
* Contains rows of data.
* ~~To Cre~~ CREATE SCHEMA } permissions required
  USE SCHEMA on parent catalog.
  USE CATALOG

* SELECT
  USE SCHEMA } to query data.
  USE CATALOG

$$\begin{tikzcd}
 & \text{Tables} \arrow[dl, dash] \arrow[dr, dash] & \\
\text{Managed} & & \text{External}
\end{tikzcd}$$

Managed tables
1) default way to create tables in Unity Catalog.
2) Unity Catalog manages life cycle and life layout
   of these tables

$$\Downarrow$$

create, update,
delete etc.,

3) Stored in root storage location
4) Use delta table format
5) If managed table is dropped, its underlying data is deleted within 30 days.

External tables

1) tables whose data life cycle & file layout ⇒ not managed by unity catalog.
2) Use external tables to register large amounts of existing data in Unity catalog
3) If E.T is dropped, underlying data is not deleted
4) We can manage privileges on external tables & use them in queries in same way as managed tables

Ex:- DELTA
JSON
CSV
AVRO
PARQUET
ORC
TEXT

} different file formats.

Unity Catalog

1) Create a catalog

* To create a catalog, use CREATE CATALOG command
* We must be a metastore admin or user with the CREATE CATALOG privilege on metastore to create catalog.

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

(or)

create a catalog and specify the managed locatio
spark.sql(" CREATE CATALOG IF NOT EXISTS
quickstart_catalog MANAGED LOCATION
'<location-Path>' ")

# set the current catalog
spart.sql(" USE CATALOG quickstart_catalog")

# show all catalogs in metastore
display(spark.sql("SHOW CATALOGS"))

Grant create and use catalog permissions for catalog to all users on account.
spark.sql(""" GRANT CREATE, USE CATALOG
ON CATALOG quickstart_catalog
TO 'account users' """)

show grants on quickstart catalog
display(spark.sql(" SHOW GRANT ON CATALOG
quickstart_catalog"))

Create and manage schemas (databases)

Schemas are also called databases
↳ second level of UC three-level namespace
↳ logically organize tables and views

Create schema in the catalog that was set earlier

```
spark.sql("""
    CREATE SCHEMA IF NOT EXISTS quickstart_sch
  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, use schema permissions for schema to all users on account

```
spark.sql("""
  GRANT CREATE TABLE, USE SCHEMA
  ON SCHEMA quickstart_schema
  TO `account users` """)
```

Create a managed table

set current schema
```
spark.sql("USE quickstart_schema")
```

show the current database
```
spark.catalog.currentDatabase()
```

Create a managed delta table in catalog that was set earlier
```
spark.sql("CREATE OR REPLACE TABLE quickstart_table
                               (id STRING)")
```

Grant select and modify permissions for table to all users on account
```
spark.sql("""GRANT SELECT, MODIFY ON TABLE quickstart_table
          TO `account_users` """)
```

# list all available tables in catalog
display(spark.sql(" SHOW TABLES "))

Insert 10 rows into table

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

# Show table
display(spark.table("quickstart_table"))

Show all available tables in schema.
display(spark.sql(" SHOW TABLES IN quickstart
_schema"))