**Unity Catalog in Databricks — Detailed Overview**

**1. Introduction**

Databricks Unity Catalog is a **unified governance solution** for all data and AI assets within the Databricks Lakehouse Platform.  
Before Unity Catalog, each Databricks workspace had its own **Hive Metastore**, which meant:

* Permissions were managed separately for each workspace.
* Sharing data across workspaces required duplication.
* Governance and auditing were inconsistent.

Unity Catalog solves these issues by:

* Providing **centralized metadata management** for multiple workspaces.
* Introducing a **three-level namespace** for data organization.
* Offering fine-grained access controls for **data, files, ML models, and AI assets**.
* Tracking **data lineage** for compliance and debugging.

**2. Core Concepts**

**2.1 Metastore**

The **Metastore** is the highest-level container in Unity Catalog.  
It holds:

* **Catalogs** (collections of schemas)
* **Permissions** (who can access which catalogs/schemas/tables)
* **Storage root location** (where managed tables are stored)

A metastore is **unique to your Databricks account** and can be shared across multiple workspaces.

**2.2 Catalog**

A **Catalog** is a top-level namespace inside the metastore.  
Think of it like a “database server” in traditional RDBMS:

* Groups together schemas that have related datasets.
* Helps enforce **logical separation** between business domains.

Example:

finance

marketing

engineering

**2.3 Schema**

A **Schema** is the second-level namespace inside a catalog.  
It’s similar to a “database” inside SQL Server or PostgreSQL:

* Groups together related tables and views.
* Defines default permissions for contained objects.

Example:

finance.payroll

finance.tax

marketing.advertising

**2.4 Table/View**

* **Table** → Stores structured data (managed or external).
* **View** → Logical definition of a dataset (query-based).

Example:

finance.payroll.salaries

marketing.advertising.campaigns

**3. Three-Level Namespace**

Unity Catalog enforces the naming convention:

<catalog>.<schema>.<table>

Example:

sales\_data.analytics.orders

* **sales\_data** → Catalog
* **analytics** → Schema
* **orders** → Table

This makes it easier to:

* Avoid name collisions.
* Clearly locate datasets in large organizations.
* Manage permissions at each level.

**4. Key Features**

**4.1 Centralized Access Control**

* Control permissions at the **catalog**, **schema**, or **table** level.
* Uses SQL GRANT and REVOKE commands.
* Permissions are enforced across all workspaces using the same metastore.

Example:

GRANT SELECT ON TABLE sales\_data.analytics.orders TO `data\_team`;

**4.2 Fine-Grained Security**

* **Row-level security** → Filter rows based on user identity.
* **Column-level security** → Mask or hide sensitive fields.

**4.3 Data Lineage**

* Automatically tracks **upstream** (where data came from) and **downstream** (what depends on it) relationships.
* Useful for:
  + Impact analysis before schema changes.
  + Compliance audits.
  + Debugging data pipelines.

**4.4 Multi-Format Support**

Unity Catalog can register:

* **Managed tables** → Data stored in the metastore’s default storage root.
* **External tables** → Data stored outside Databricks (e.g., S3, ADLS).
* **Volumes** → Direct access to files and directories in cloud storage.

**5. Setting Up Unity Catalog**

**5.1 Prerequisites**

* Databricks account admin privileges.
* A cloud storage location (S3, ADLS Gen2, or GCS).
* Required cloud IAM roles/policies set up.

**5.2 Creating a Metastore**

1. Go to **Account Console** → **Data** → **Metastores**.
2. Click **Create Metastore**.
3. Provide:
   * **Name** → e.g., MainMetastore
   * **Cloud storage root** → e.g., s3://databricks-metastore-root/
4. Click **Create**.

**5.3 Assigning Metastore to Workspace**

1. In the Metastore details page, click **Assign to Workspace**.
2. Select your workspace → Click **Assign**.
3. Optionally, set it as **Default Metastore**.

**6. Working with Unity Catalog Objects**

**Creating a Catalog**

CREATE CATALOG sales\_data;

**Creating a Schema**

CREATE SCHEMA sales\_data.analytics;

**Creating a Table**

CREATE TABLE sales\_data.analytics.orders (

order\_id STRING,

amount DECIMAL(10,2),

order\_date DATE

);

**Querying Data**

SELECT \* FROM sales\_data.analytics.orders;

**7. Benefits of Unity Catalog**

* **Single source of truth** for all metadata.
* Simplifies **cross-workspace data sharing**.
* Built-in **security, compliance, and auditing**.
* Clear **namespace hierarchy** for large organizations.
* Works seamlessly with **Databricks SQL, Data Science, and Machine Learning workloads**.

**8. Summary Table**

| **Term** | **Meaning** | **Example** |
| --- | --- | --- |
| Metastore | Top-level container for catalogs | MainMetastore |
| Catalog | Groups related schemas | sales\_data |
| Schema | Groups related tables/views | analytics |
| Table/View | Stores/query data | orders |

If you want, I can now turn this into a **formatted PDF with diagrams** so you have a clean 4-page study document for Unity Catalog. That would make it easier for revision.  
Do you want me to prepare that?