Unity Catalog

Comprehensive Guide to Unified Data Governance

Databricks 14-Days AI Challenge

Day 8

Agenda

- **Introduction** to Unity Catalog
- **Three-Level Namespace** Hierarchy
- **Catalog → Schema → Table**
- **Managed vs External** Tables
- **Access Control** with GRANT/REVOKE
- **Data Lineage**
- **Practical Implementation**
- **Best Practices**

What is Unity Catalog?

Definition

Unity Catalog is Databricks' **unified governance solution** for all data and AI assets.

Think of it as:

- The **central librarian** for your entire data ecosystem
- It knows **where everything is stored**
- It controls **who can access what**
- It tracks **how data flows** through your organization

Why Unity Catalog Matters

Challenge	Without Unity Catalog	With Unity Catalog
Data Discovery	Scattered metadata	Single searchable catalog
Access Control	Inconsistent, workspace-specific	Centralized, consistent
Audit Trail	Limited visibility	Complete lineage & logs
Data Sharing	Complex, insecure	Secure, governed
Compliance	Manual tracking	Automated reporting

Core Capabilities

1. Centralized Governance

Single place to manage **all data assets**

2. Fine-grained Access Control

Control who can do what, **at any level**

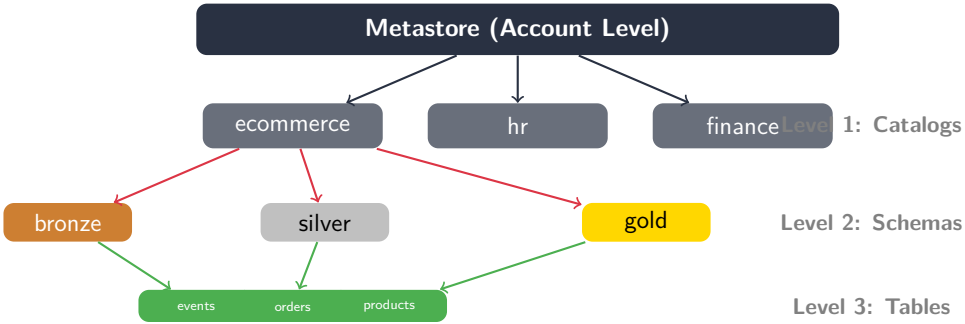
3. Data Lineage

Track where data **comes from** and **goes**

4. Data Sharing

Securely share data across organizations

Three-Level Namespace Hierarchy



The Complete Address Format

Fully Qualified Name Pattern

```
catalog.schema.table
```

Example: `ecommerce.gold.products`

Level	Analogy	Example	Purpose
Metastore	Country	(Implicit)	Top-level container
Catalog	City	`ecommerce`	Groups by domain
Schema	Street	`gold`	Groups by data layer
Table	House	`products`	Actual data

Level 1: Catalog

Key Characteristics:

- First component of three-part naming
- Contains **multiple schemas**
- **Isolation boundary** for permissions
- Represents a **business domain**

Creating a Catalog

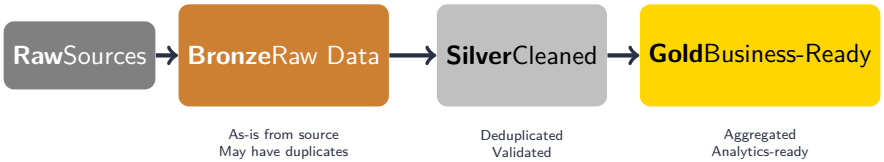
```
CREATE CATALOG ecommerce
COMMENT 'E-commerce data';

USE CATALOG ecommerce;
```

Catalog Strategies:

By Domain	By Environment	By Region
ecommerce, marketing	dev, staging, prod	us_west, eu_central

Level 2: Schema - Medallion Architecture



Creating Schemas

```
CREATE SCHEMA bronze COMMENT 'Raw ingested data';
CREATE SCHEMA silver COMMENT 'Cleaned data';
CREATE SCHEMA gold COMMENT 'Business-ready aggregations';
```

Level 3: Tables and Views

Object Type	Description	Stores Data?
Managed Table	UC manages metadata & data files	Yes
External Table	UC manages only metadata	External
View	Saved SQL query	No
Materialized View	Precomputed view stored as table	Yes
Function	User-defined functions (UDFs)	No

Creating a Managed Table

```
CREATE TABLE bronze.events (
  event_id STRING, user_id STRING, event_timestamp TIMESTAMP
) USING DELTA;
```

Managed vs External Tables

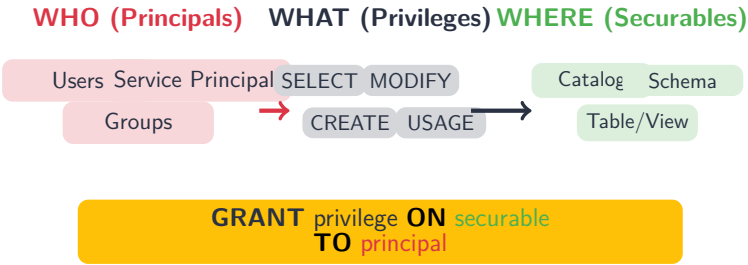
When to Use Each Type

Choose Managed Tables

- Starting **fresh** with new data
- Want **automatic** storage handling
- Working on **isolated** projects
- Want **automatic cleanup** on delete
- **Simplicity** is priority

- Have **existing** data in cloud storage
- **Multiple systems** access same files
- Want to **preserve** data after drop
- **Migrating** from existing data lake
- Need **control** over storage location

Access Control - The Security Model



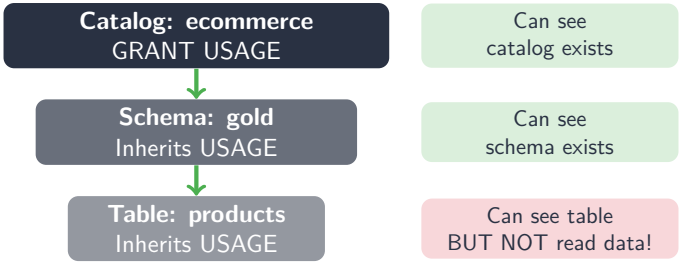
Key Privileges

Privilege	Description	Applies To
`SELECT`	Read data from tables/views	Tables, Views
`MODIFY`	Add, update, delete data	Tables
`CREATE`	Create objects within container	Catalogs, Schemas
`USAGE`	Access the container (required)	Catalogs, Schemas
`ALL PRIVILEGES`	All available privileges	All securables
`EXECUTE`	Run functions	Functions

Important

`USAGE` on parent containers is **required** before granting table-level privileges!

Privilege Inheritance



To read data: `USAGE` on Catalog + `USAGE` on Schema + `SELECT` on Table

GRANT/REVOKE Examples

GRANT Examples

```
-- Grant SELECT on specific table
GRANT SELECT ON TABLE gold.products TO `analysts@company.com`;

-- Grant all privileges on schema
GRANT ALL PRIVILEGES ON SCHEMA silver TO `engineers@company.com`;

-- Grant USAGE to access catalog
GRANT USAGE ON CATALOG ecommerce TO `data_team`;
```

REVOKE Example

```
REVOKE SELECT ON TABLE gold.products FROM `intern@company.com`;
```

Permission Patterns

Data Analysts (Read-Only Gold)

```
GRANT USAGE ON CATALOG ecommerce
  TO `analysts@company.com`;
GRANT USAGE ON SCHEMA gold
  TO `analysts@company.com`;
GRANT SELECT ON ALL TABLES IN
  SCHEMA gold TO `analysts@company.com`;
```

Data Engineers (Full Access)

```
GRANT ALL PRIVILEGES ON CATALOG
  ecommerce TO `engineers@company.com`;
```

Views for Controlled Access

Create views to expose only **specific columns/rows**

Data Lineage

Definition

Data lineage tracks the **complete journey of data**—where it comes from, how it transforms, and where it ends up.

Unity Catalog automatically captures: Table lineage, Column lineage, Query lineage, User lineage

Lineage Types & Benefits

Types of Lineage:

- **Table Lineage** - Source to target tables
- **Column Lineage** - Source to target columns
- **Query Lineage** - Which queries access tables
- **User Lineage** - Who interacts with data

Benefits:

- **Impact Analysis** - See downstream dependencies
- **Root Cause** - Trace data errors
- **Compliance** - Show data provenance
- **Documentation** - Auto-generated flows

Viewing Lineage

Navigate to **Catalog** → Find your table → Click **Lineage** tab

Naming Conventions

Level	Convention	Example
Catalog	lowercase, business domain	`ecommerce`, `hr`
Schema	lowercase, data layer	`bronze`, `silver`, `gold`
Table	lowercase_snake_case	`customer_orders`
View	prefix with purpose	`rpt_sales_summary`

Permission Best Practices

- Use **Groups**, not individual users
- Apply **Principle of Least Privilege**
- Use **Views** for row/column filtering
- Conduct **Regular Access Reviews**

Anti-Patterns to Avoid

Anti-Pattern	Problem	Better Approach
ALL PRIVILEGES broadly	Over-permissioning	Grant specific privileges
Granting to individuals	Hard to manage	Use groups
Skipping USAGE grants	Tables inaccessible	Always grant USAGE first
One giant catalog	No isolation	Separate by domain
Direct table access	No data protection	Use views

Permission Quick Reference

To Do This...	Grant This...
See a catalog exists	`USAGE ON CATALOG`
See schemas in a catalog	`USAGE ON CATALOG`
See tables in a schema	`USAGE ON CATALOG + SCHEMA`
Read data from a table	Above + `SELECT ON TABLE`
Insert/Update data	Above + `MODIFY ON TABLE`
Create new tables	Above + `CREATE ON SCHEMA`
Full control	`ALL PRIVILEGES`

Summary

Unity Catalog Provides

1. **Three-Level Hierarchy**: Catalog → Schema → Table
2. **Flexible Table Types**: Managed for simplicity, External for existing data
3. **Fine-Grained Access Control**: GRANT/REVOKE with inheritance
4. **Automatic Data Lineage**: Track data flow without manual effort
5. **Secure Data Sharing**: Views provide controlled access

*By implementing these patterns, you create a **well-governed**, **secure**, and **maintainable** data platform.*

Thank You!

Unity Catalog - Day 8

Databricks 14-Days AI Challenge