Data Governance & Unity Catalog

1. Unity Catalog Fundamentals

What is Unity Catalog?

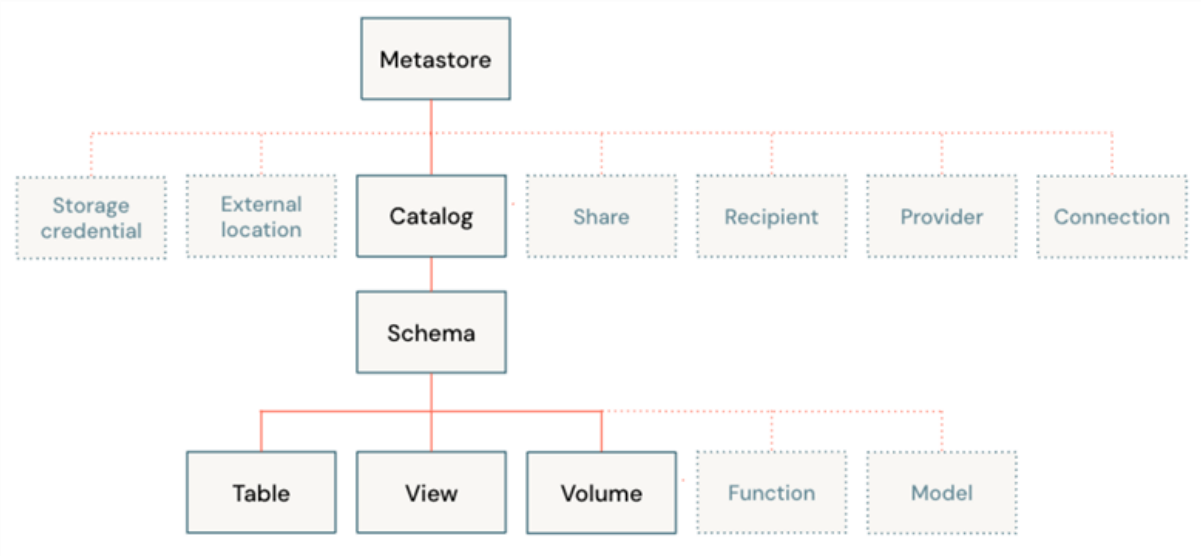
Databricks' **centralized governance solution** providing:

- Unified data discovery across workspaces
- Fine-grained access control (FGAC)
- End-to-end data lineage tracking
- Auditing and compliance capabilities

Key Components

Component	Description
Metastore	Top-level container for metadata
3-Level Namespace	catalog.schema.table hierarchy
Access Controls	SQL-standard GRANT/REVOKE
Audit Logs	All access and operations tracking

Unity Catalog Architecture



Metastore → Workspaces → Data Objects

2. Setup & Configuration

Creating a Metastore

bash

```
# Using Databricks CLI
databricks metastores create \
  --name "enterprise_metastore" \
  --storage-root "s3://company-uc-metastore" \
  --region us-west-2
```

Workspace Attachment

bash

```
databricks metastores assign \
  --metastore-id <metastore-id> \
  --workspace-id <workspace-id>
```

Admin Configuration

sql

```
-- Set initial admins
GRANT CREATE CATALOG ON METASTORE TO `admin-group@company.com`;

-- Enable on workspace
SET spark.databricks.sql.initial.catalog.name = 'main';
```

3. 3-Level Namespace

Hierarchy Structure

text

```
main (catalog)
├── finance (schema)
│   ├── transactions (table)
│   └── budgets (view)
└── hr (schema)
    ├── employees (table)
    └── salaries (secured table)
```

Creating Objects

sql

```
-- Catalog
CREATE CATALOG IF NOT EXISTS production;

-- Schema
CREATE SCHEMA production.finance
COMMENT "Financial data domain";

-- Table
CREATE TABLE production.finance.transactions (
  id STRING,
  amount DECIMAL(18,2),
  date DATE
) USING DELTA;

-- View
CREATE VIEW production.finance.current_quarter AS
SELECT * FROM transactions
```

```
WHERE date BETWEEN '2023-10-01' AND '2023-12-31';
```

4. External Data Configuration

Mounting Cloud Storage

sql

```
CREATE EXTERNAL LOCATION company_data
URL 's3://company-data-lake/'
WITH (STORAGE CREDENTIAL `aws_iam_role`);

GRANT READ FILES ON EXTERNAL LOCATION company_data TO `analysts@company.com`;
```

External Tables

sql

```
CREATE TABLE production.analytics.web_logs
USING PARQUET
LOCATION 's3://company-data-lake/logs/web/'
```

Data Federation

sql

```
CREATE FOREIGN CATALOG snowflake_catalog
CONNECTION snowflake_conn
OPTIONS (
  database='SNOWFLAKE_DB',
  schema='PUBLIC'
);
```

5. Access Control Model

Permission Hierarchy

text

```
METASTORE
└── CATALOG
    └── SCHEMA
        └── TABLE/VIEW
```

Grant Examples

```
-- Catalog-level
GRANT USE CATALOG ON CATALOG production TO `finance-team@company.com`;

-- Schema-level
GRANT SELECT ON SCHEMA production.finance TO `analysts@company.com`;

-- Table-level
GRANT SELECT, MODIFY ON TABLE production.hr.employees TO `hr-managers@company.com`;

-- Row-level
CREATE ROW FILTER hr.employee_filter
AS (dept_id IN (SELECT dept_id FROM hr.user_departments WHERE user =
current_user()));

ALTER TABLE hr.salaries SET ROW FILTER hr.employee_filter ON (employee_id);
```

Column Masking

sql

```
CREATE MASK hr.salary_mask
AS (CASE
      WHEN is_member('exec-team') THEN salary
      ELSE NULL
    END);

ALTER TABLE hr.employees
ALTER COLUMN salary SET MASK hr.salary_mask;
```

---

6. Data Governance Features

Data Lineage

sql

```
-- Query lineage
SELECT * FROM system.access.lineage
WHERE object_name = 'production.finance.transactions';

-- Visualized in UI: See upstream/downstream dependencies
```

Audit Logging

sql

```
-- Access patterns
SELECT * FROM system.access.audit
WHERE table_name = 'salaries'
ORDER BY event_time DESC
LIMIT 100;

-- Data changes
SELECT * FROM delta.history('production.finance.transactions');
```

Data Quality Monitoring

sql

```
CREATE EXPECTATION production.finance.valid_transactions
ON TABLE transactions
EXPECT (amount > 0) AS "positive_amount";

SELECT * FROM system.expectations.violations;
```

---

7. Mini Project Implementation

Step 1: Setup Environment

bash

```
# Create project catalog
databricks unity-catalog catalogs create --name "project_team"

# Assign permissions
databricks unity-catalog grants create \
  --principal "project-team@company.com" \
  --privileges "USE_CATALOG, CREATE_SCHEMA" \
  --catalog "project_team"
```

Step 2: Data Discovery

sql

```
-- Search metadata
SELECT * FROM system.information_schema.tables
WHERE table_name LIKE '%customer%';

-- Annotate assets
ALTER TABLE sales.customers SET TBLPROPERTIES (
  'description' = 'Master customer records from CRM system',
  'owner' = 'data-engineering@company.com'
);
```

Step 3: Access Control

sql

```
-- Create secure view
CREATE VIEW project_team.analytics.customer_segment AS
SELECT
  id,
  segment,
  region
FROM production.sales.customers;

-- Grant access
GRANT SELECT ON VIEW project_team.analytics.customer_segment
TO `marketing-team@company.com`;
```

Step 4: Lineage Tracking

python

```
# Notebook cell to document lineage
dbutils.lineage.track(
  inputs=["production.sales.raw_orders"],
  outputs=["project_team.analytics.order_summary"],
  transformation_type="aggregation"
)
```

Step 5: Auditing

sql

```
-- Create audit dashboard
CREATE VIEW project_team.monitoring.access_logs AS
SELECT
  user_identity,
  table_name,
  action_name,
  event_time
FROM system.access.audit
WHERE catalog = 'project_team'
ORDER BY event_time DESC;
```

---

8. Best Practices

Naming Conventions

```
- Catalogs: `{environment}_{domain}` (prod_sales, dev_hr)
- Schemas: `{functional_area}` (finance, marketing, operations)
- Tables: `{entity}_{granularity}` (customers_daily, transactions_fact)
```

Lifecycle Management

sql

```
-- Promote from dev to prod
CREATE TABLE prod_analytics.customers
DEEP CLONE dev_analytics.customers;

-- Archive old data
CREATE CATALOG archive COMMENT "Historical data";
```

Security Policies

1. **Least Privilege**: Start with minimal access
2. **Role-Based Access**: Group permissions
3. **Regular Reviews**: Quarterly permission audits

9. Troubleshooting

Issue	Solution
Permission denied	Verify GRANT statements at proper level
Missing metadata	Check metastore assignment to workspace
Cross-workspace access	Ensure shared metastore configuration
External table errors	Validate storage credential permissions

---

10. Complete Cheat Sheet

UC CLI Commands

```
# Metastores
databricks metastores list

# Catalogs
databricks unity-catalog catalogs create --name "new_catalog"

# Permissions
databricks unity-catalog grants update --principal "user@co.com" --privileges
"SELECT"
```

SQL Commands

```
-- Data sharing
CREATE SHARE marketing_data;
ADD TABLE sales.customers TO SHARE marketing_data;

-- Cleanup
DROP CATALOG old_catalog CASCADE;
```

---

11. Learning Resources

- Unity Catalog Documentation
- Databricks Academy: Data Governance
- Unity Catalog API Reference