



Manage Data Access for Analytics



Module Objectives

By the end of this course, you will be able to:

1. Describe Unity Catalog key concepts and how it integrates with the Databricks platform
2. Access Unity Catalog through clusters and SQL warehouses
3. Create and govern data assets in Unity Catalog
4. Adopt Databricks recommendations into your organization's Unity Catalog-based solutions

Module Agenda

Manage Data Access for Analytics with Unity Catalog

Introduction to Unity Catalog

- DE 6.1 – Introduction to Unity Catalog
- DE 6.2 – Overview of Data Governance
- DE 6.3 – Unity Catalog Key Concepts
- DE 6.4 – Unity Catalog Architecture
- DE 6.5 – Unity Catalog Identities
- DE 6.6 – Managing Principals in Unity Catalog
- DE 6.7 – Managing Catalog Metastores

Compute Resources in Unity Catalog

- DE 6.8 – Compute Resources
- DE 6.9 – Creating Compute Resources

Data Access Control in Unity Catalog

- DE 6.10 – Data Access Control in Databricks
- DE 6.11 – Security Model
- DE 6.12 – External Storage
- DE 6.13 – Creating and Governing Data
- DE 6.14 – Create and Share Tables
- DE 6.15 – Create External Tables

Unity Catalog Best Practices

- DE 6.16 – Best Practices
- DE 6.17 – Data Segregation
- DE 6.18 – Identity Management
- DE 6.19 – External Storage
- DE 6.20 – Upgrade a Table to Unity Catalog
- DE 6.21 – Create Views and Limiting Table Access



Introduction to Unity Catalog

Overview of Data Governance

80% of organizations seeking to scale digital business **will fail** because they do not take a modern approach to **data and analytics governance**

Source: [Gartner](#)

Data Governance

Four key functional areas

Data Access Control

Control who has access to which data

Data Access Audit

Capture and record all access to data

Data Lineage

Capture upstream sources and downstream consumers

Data Discovery

Ability to search for and discover authorized assets



Governance for data, analytics and AI is complex



Databricks Unity Catalog

Unified governance for data, analytics and AI



Unity Catalog

Overview



Unified governance across clouds

Fine-grained governance for data lakes across clouds – based on open standard ANSI SQL.

1



Unified data and AI assets

Centrally share, audit, secure and manage all data types with one simple interface.

2



Unified existing catalogs

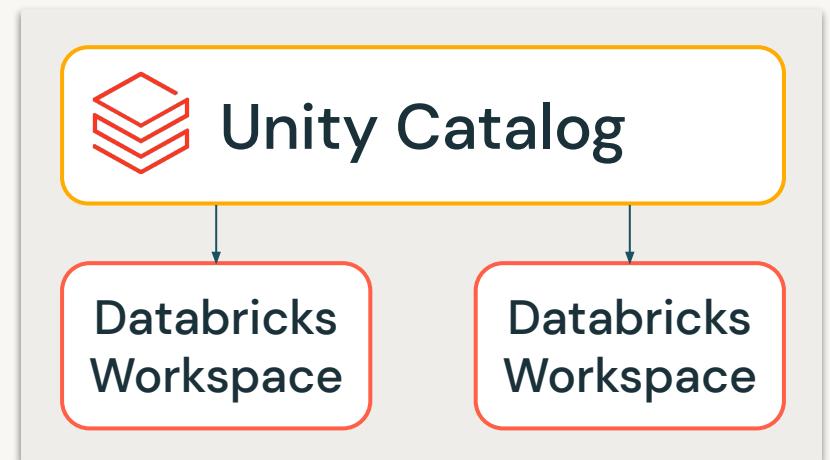
Works in concert with existing data, storage, and catalogs – no hard migration required.

3

Unity Catalog

Key Capabilities

- Centralized metadata and user management
- Centralized data access controls
- Data access auditing
- Data lineage
- Data search and discovery
- Secure data sharing with Delta Sharing



```
GRANT ... ON ... TO ...
REVOKE ... ON ... FROM ...
```

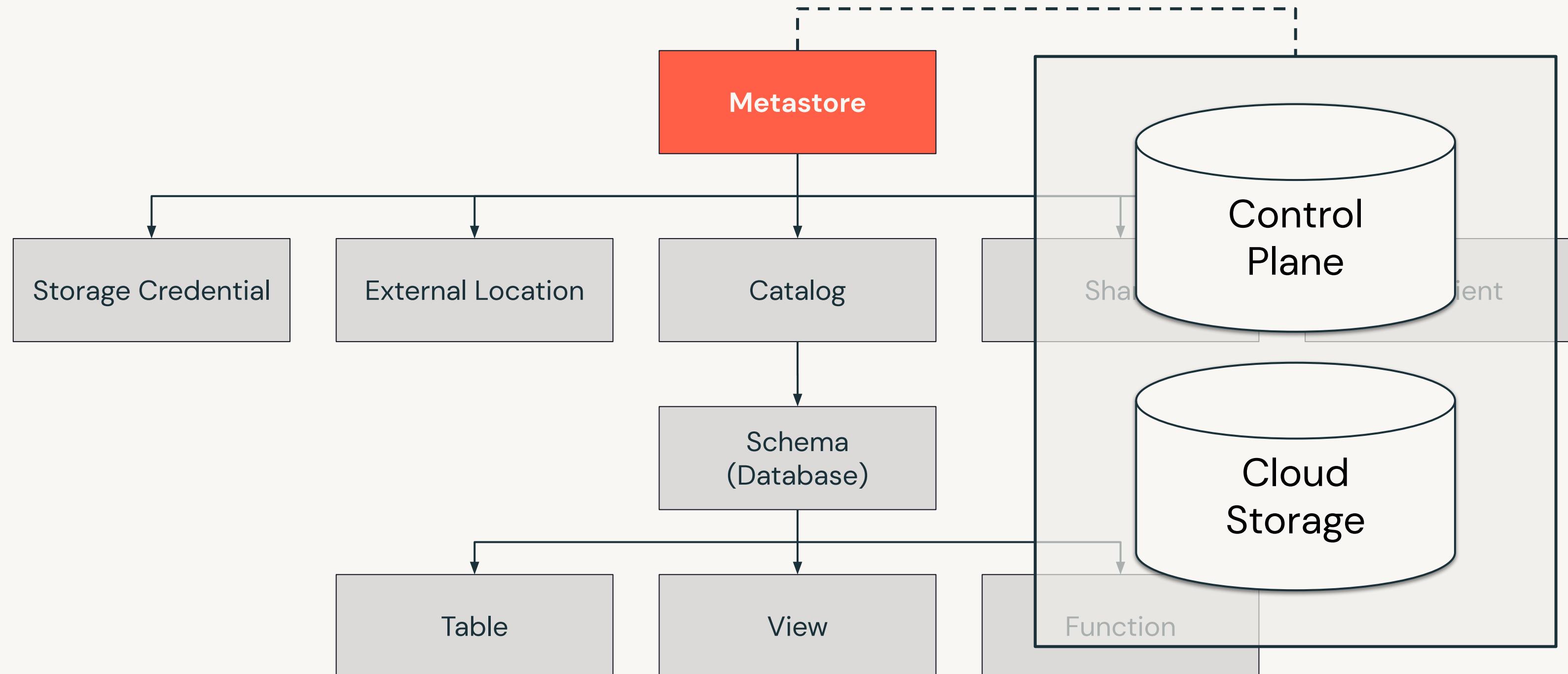
Catalogs, Databases (schemas),
Tables, Views, Storage
credentials, External locations

Unity Catalog

Key Concepts

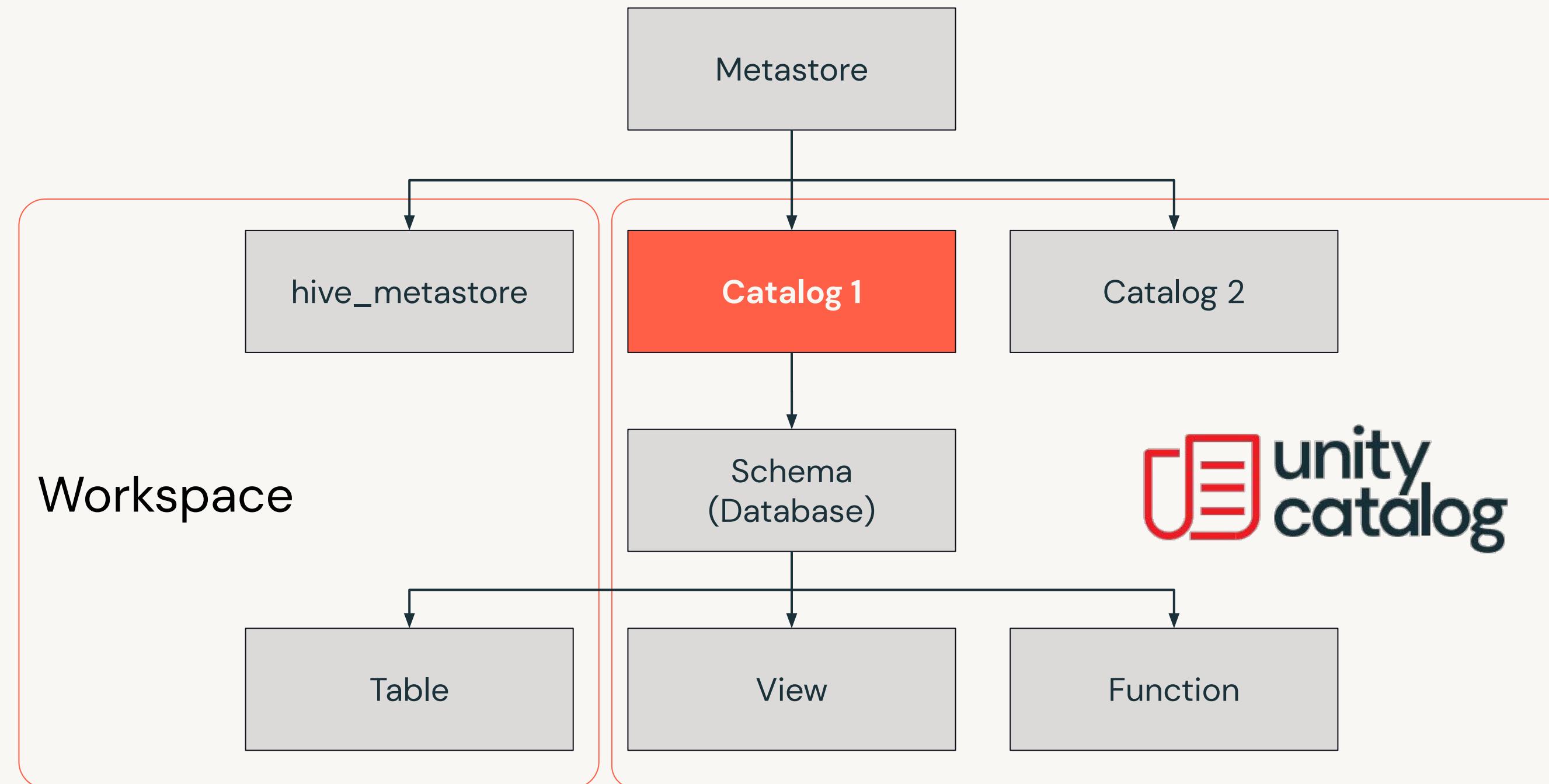
Metastore

Unity Catalog metastore elements



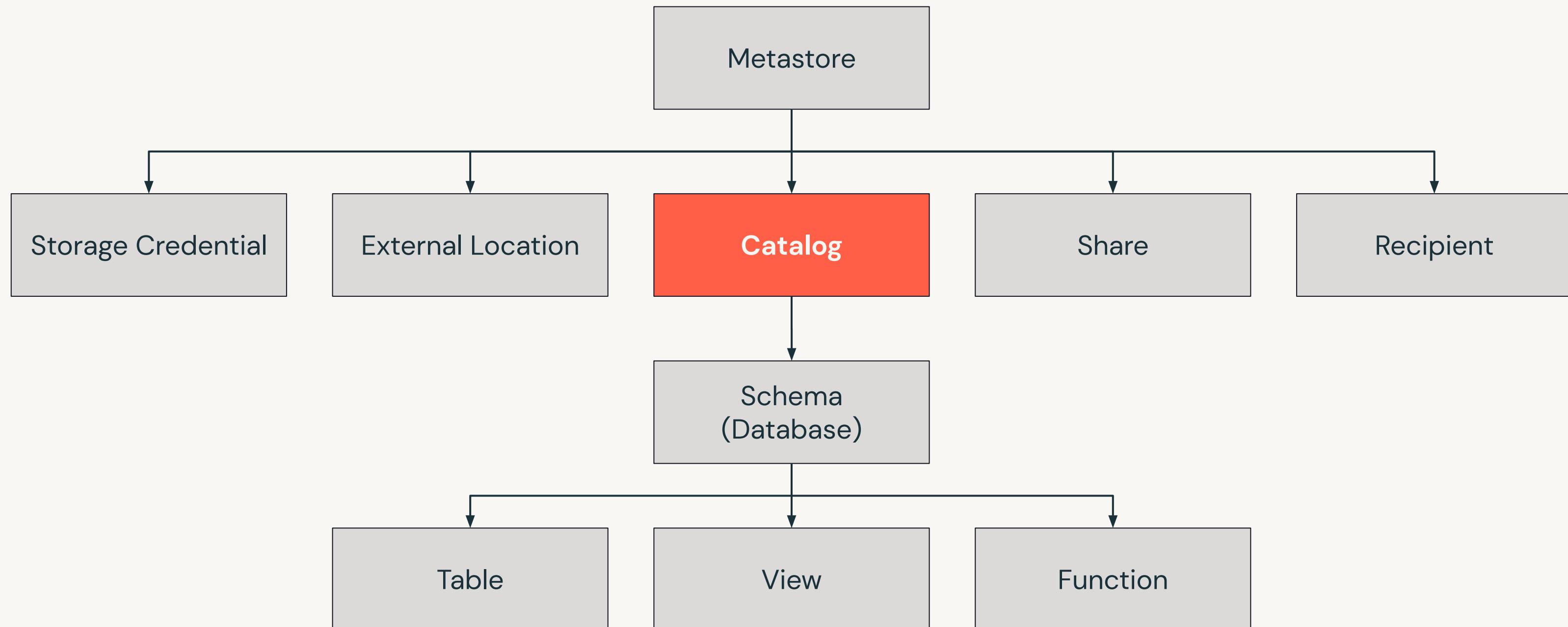
Metastore

Accessing legacy Hive metastore



Catalog

Top-level container for data objects



Catalog

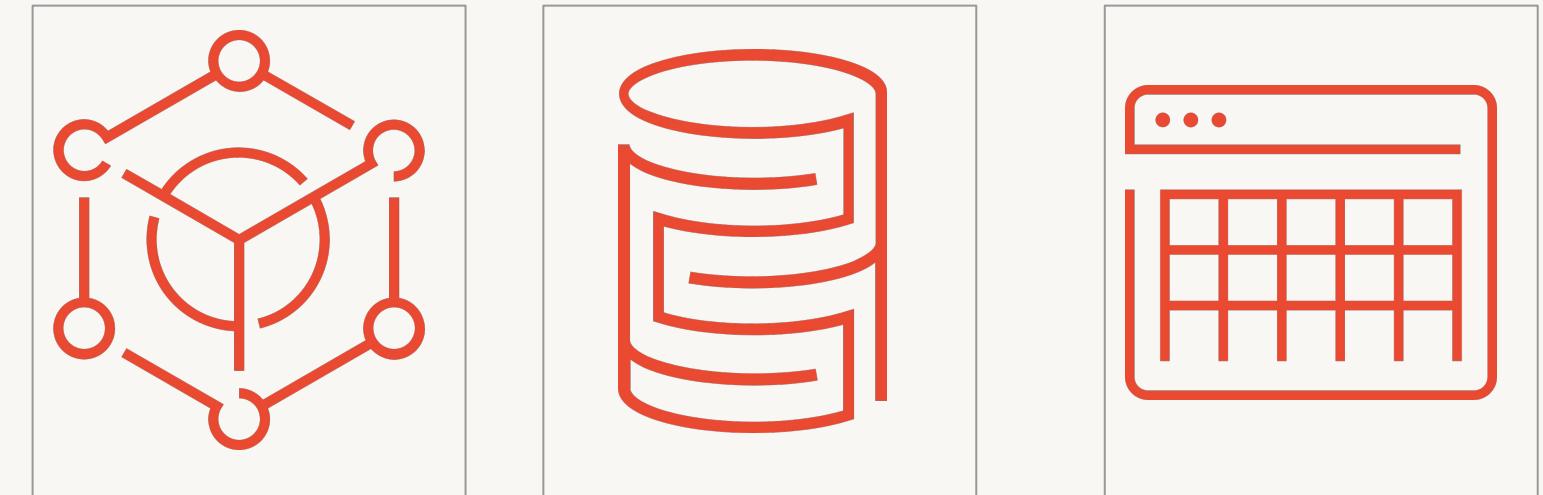
Three-level namespace

Traditional SQL two-level
namespace



`SELECT * FROM schema.table`

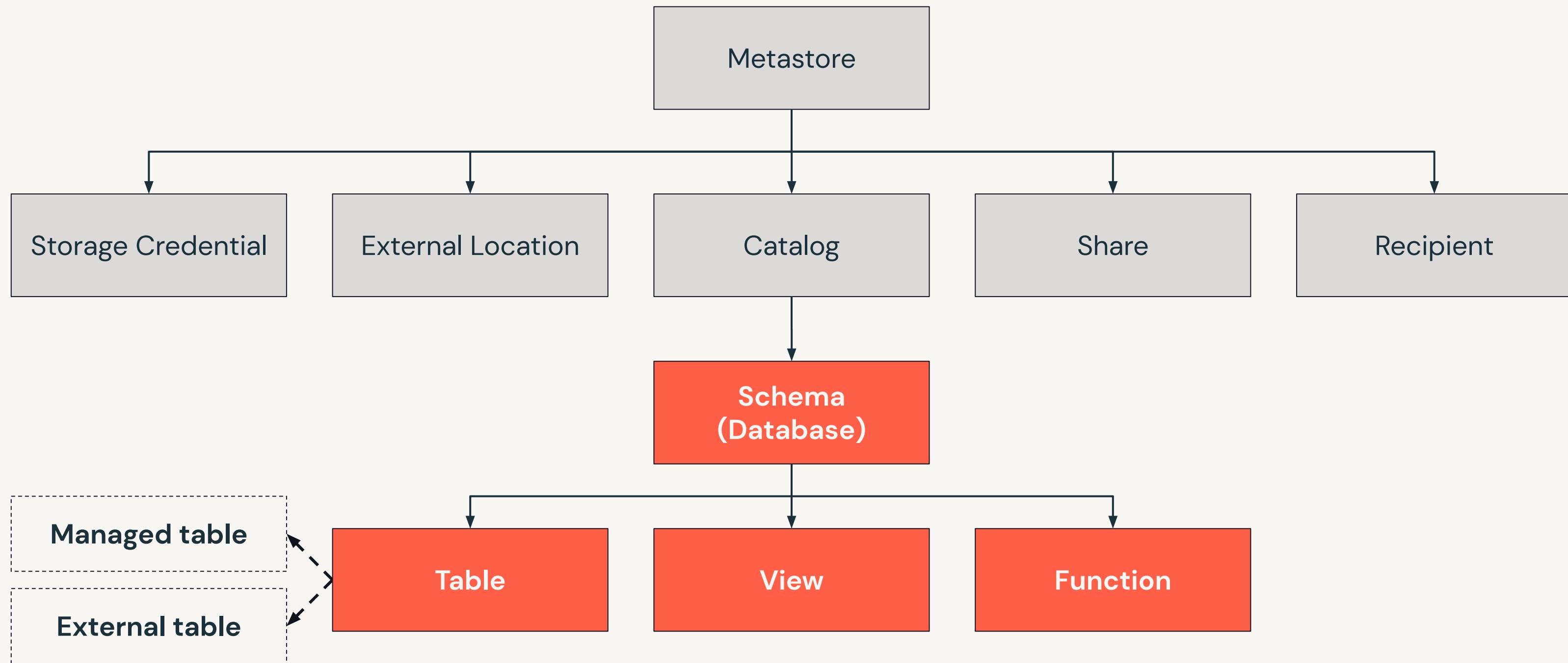
Unity Catalog three-level
namespace



`SELECT * FROM catalog.schema.table`

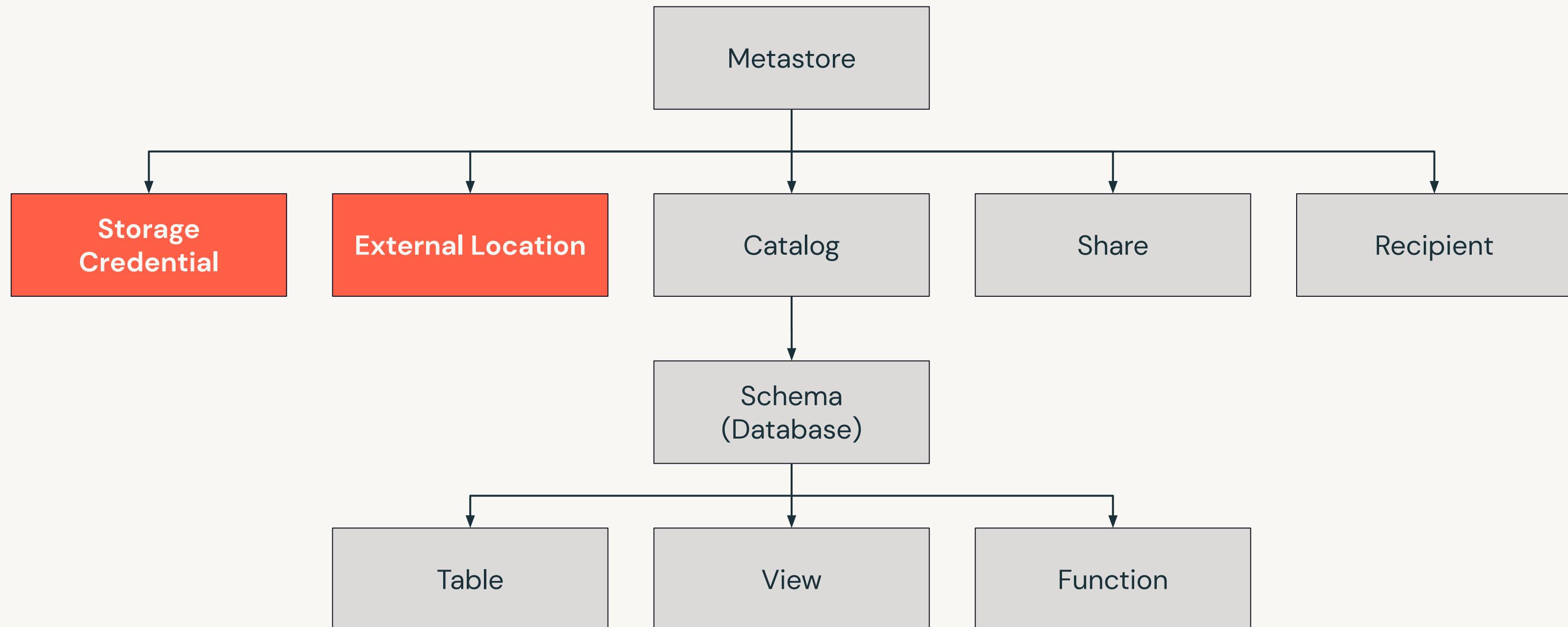
Data Objects

Schema (database), tables, views, functions



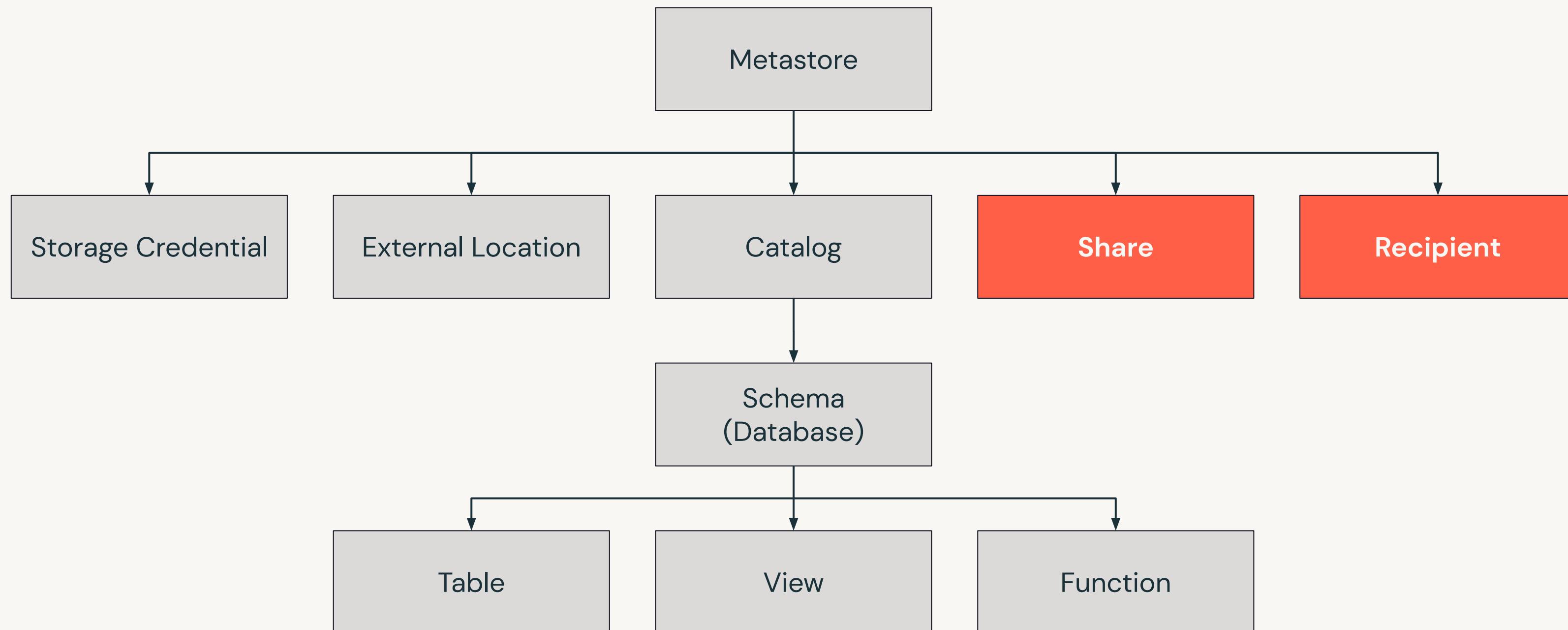
External Storage

Storage credentials and external locations



Delta Sharing

Shares and recipients

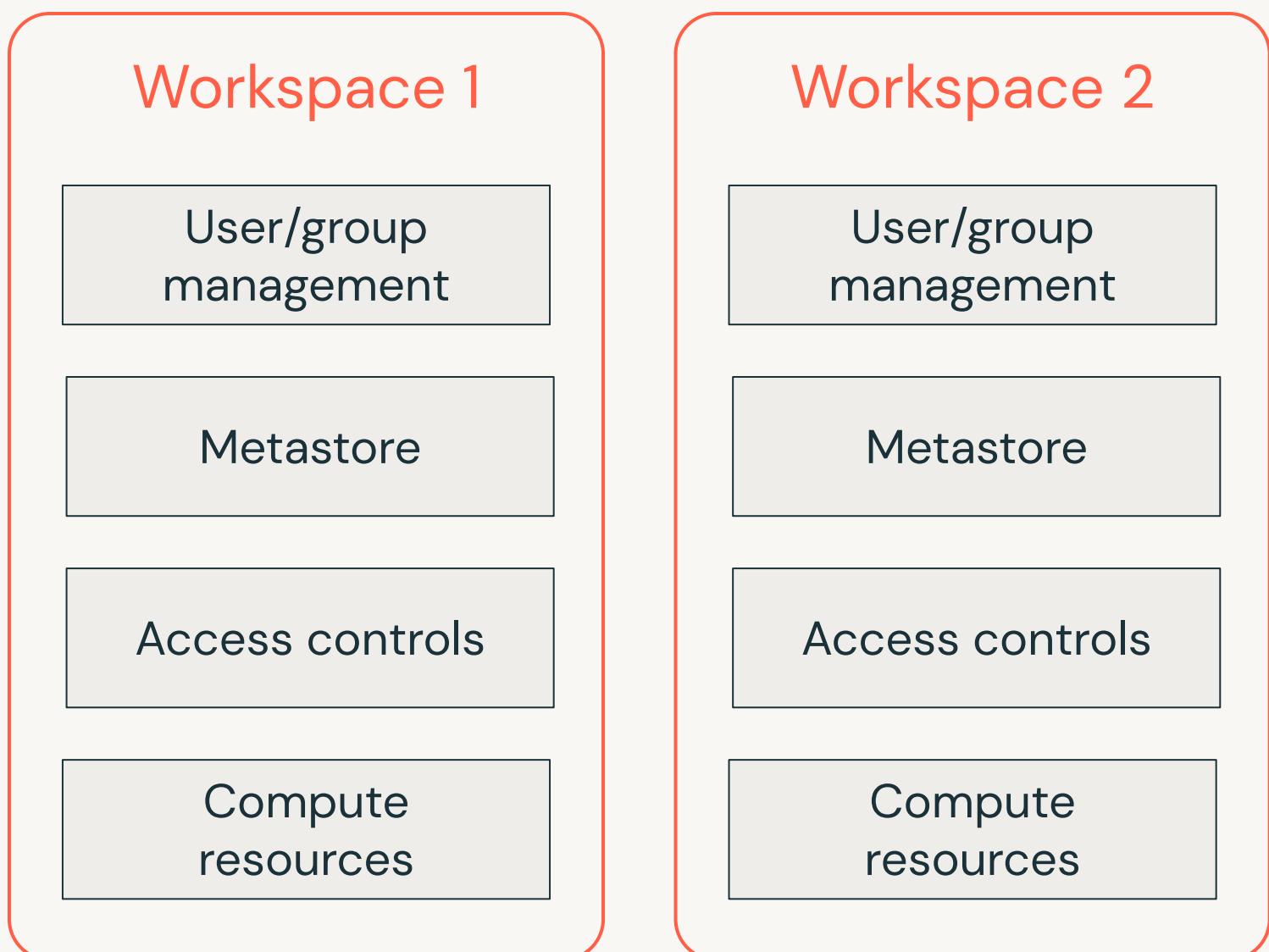


Unity Catalog

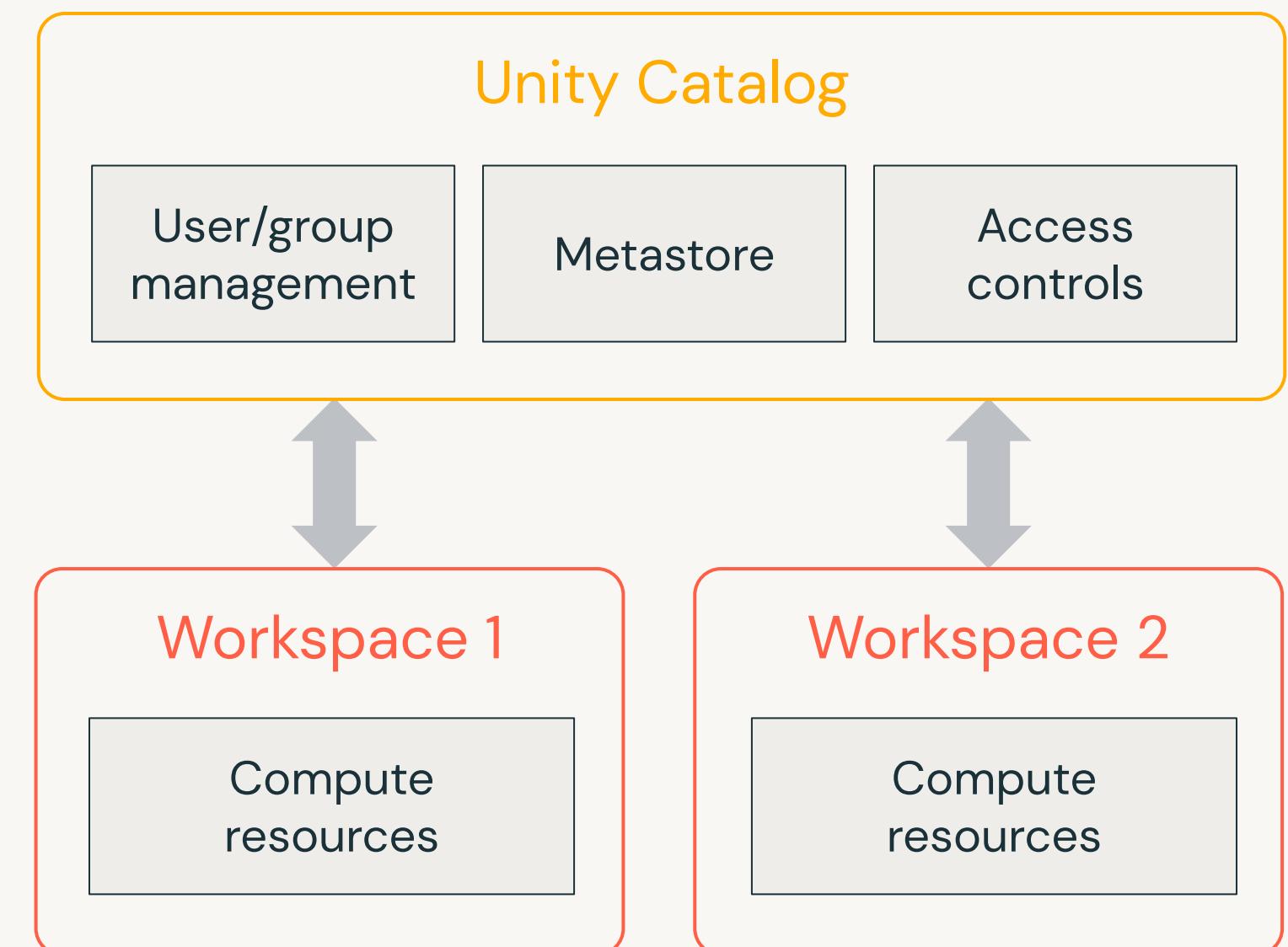
Architecture

Architecture

Before Unity Catalog

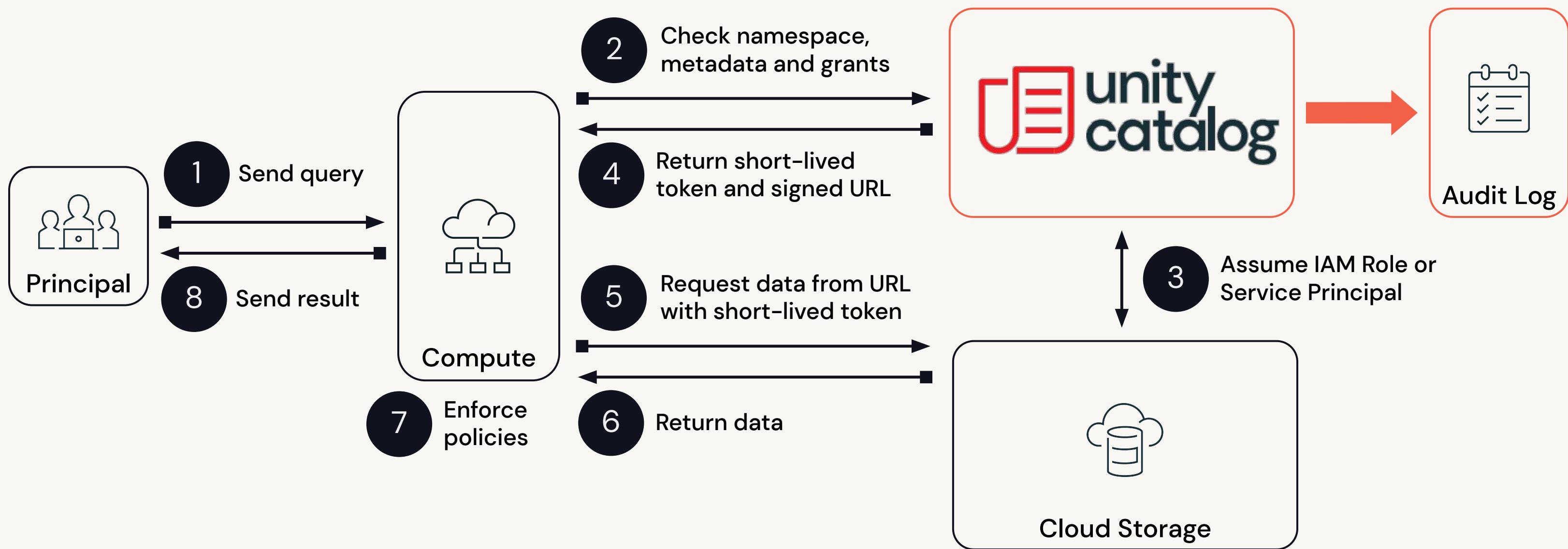


With Unity Catalog



Query Lifecycle

Unity Catalog Security Model



Compute Resources and Unity Catalog

Compute Resources for Unity Catalog

Cluster Access Mode

Modes supporting UC

Single user

Multiple language support, not shareable

Shared

Shareable, Python and SQL, legacy table ACLs

Modes not supporting UC

No isolation shared

Multiple language support



Cluster Access Mode

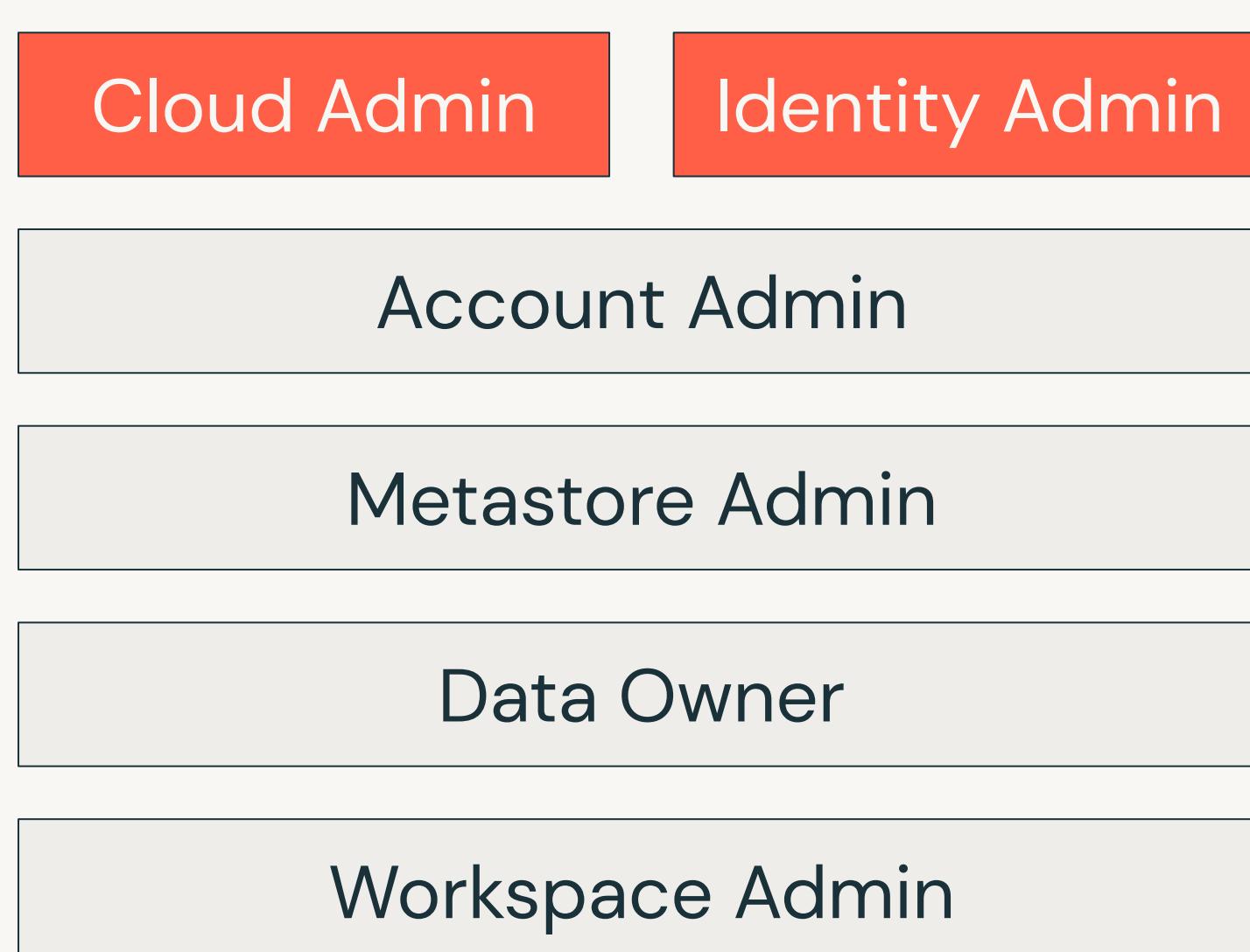
Feature matrix

Access mode	Supported languages	Legacy table ACL	Credential passthrough	Shareable	RDD API	DBFS Fuse mounts	Init scripts and libraries	Dynamic views	Machine learning
No Isolation Shared	All			●	●	●	●		●
Single user	All				●	●	●	●	●
Shared	SQL Python	●		●				●	

Roles and Identities in Unity Catalog

Unity Catalog

Roles



Cloud Admin

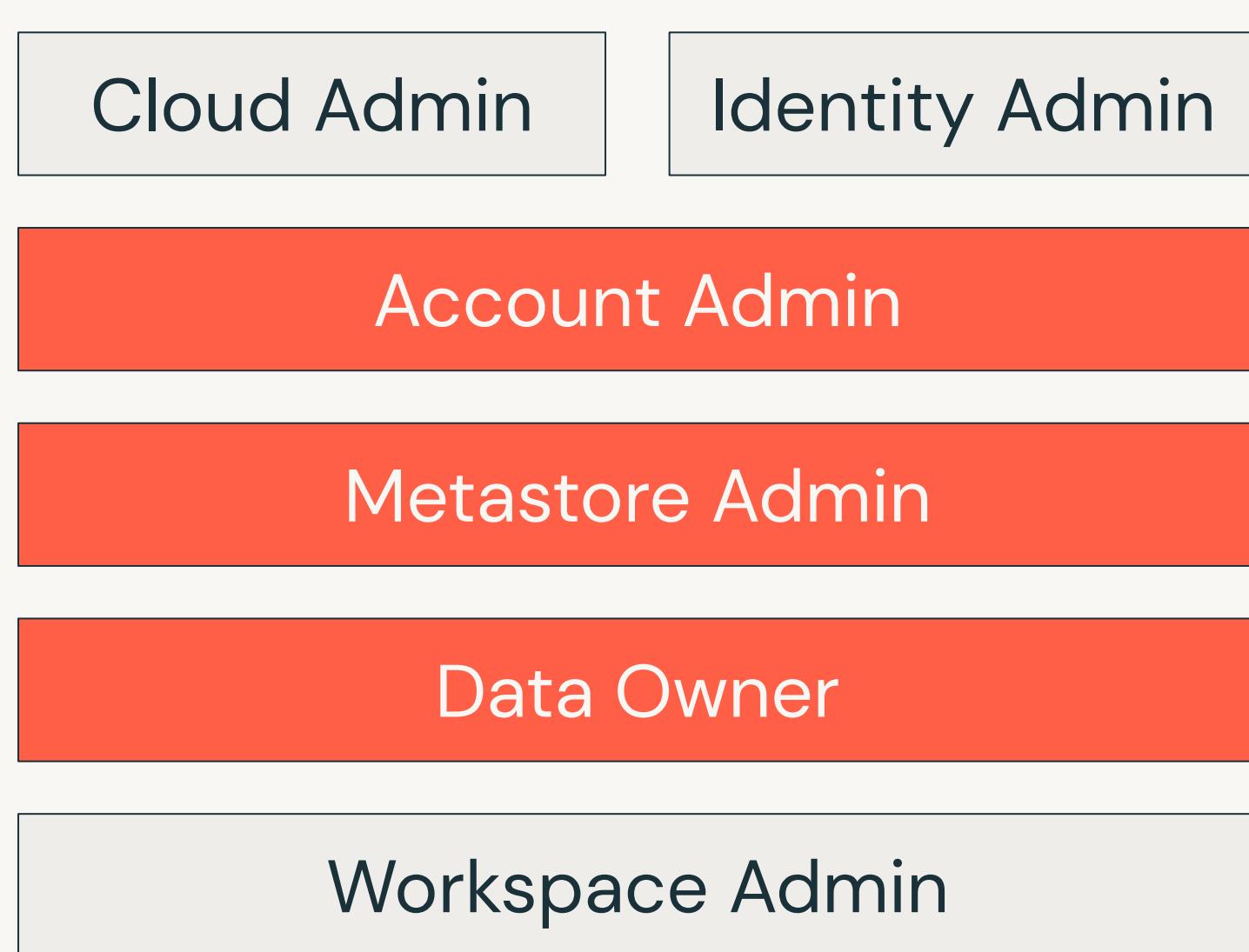
- Manage underlying cloud resources
 - Storage accounts/buckets
 - IAM role/service principals/managed identities

Identity Admin

- Manage users and groups in the identity provider (IdP)
- Provision into account (with account admin)

Unity Catalog

Roles



Account Admin

- Create or delete metastores, assign metastores to workspaces
- Manage users and groups, integrate with IdP
- Full access to all data objects

Metastore Admin

- Create or drop, grant privileges on, and change ownership of catalogs and other data objects

Data Owner – owns data objects they created

- Create nested objects, grant privileges on, and change ownership of owned objects

Unity Catalog

Roles

Cloud Admin

Identity Admin

Account Admin

Metastore Admin

Data Owner

Workspace Admin

Workspace Admin

- Manages permissions on workspace assets
- Restricts access to cluster creation
- Adds or removes users
- Elevates users permissions
- Grant privileges to others
- Change job ownership



Unity Catalog

Identities

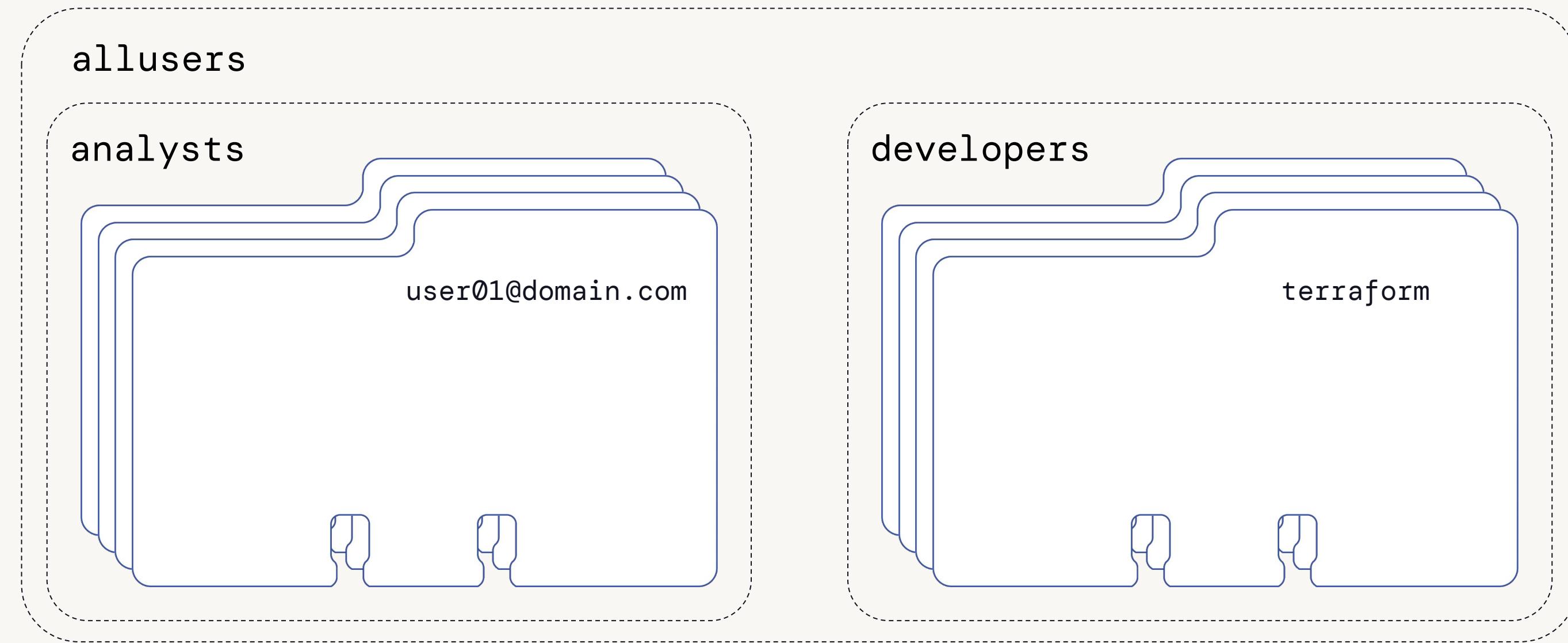
- User
- Account Administrator
- Service Principal
- Service Principal with administrative privileges



Unity Catalog

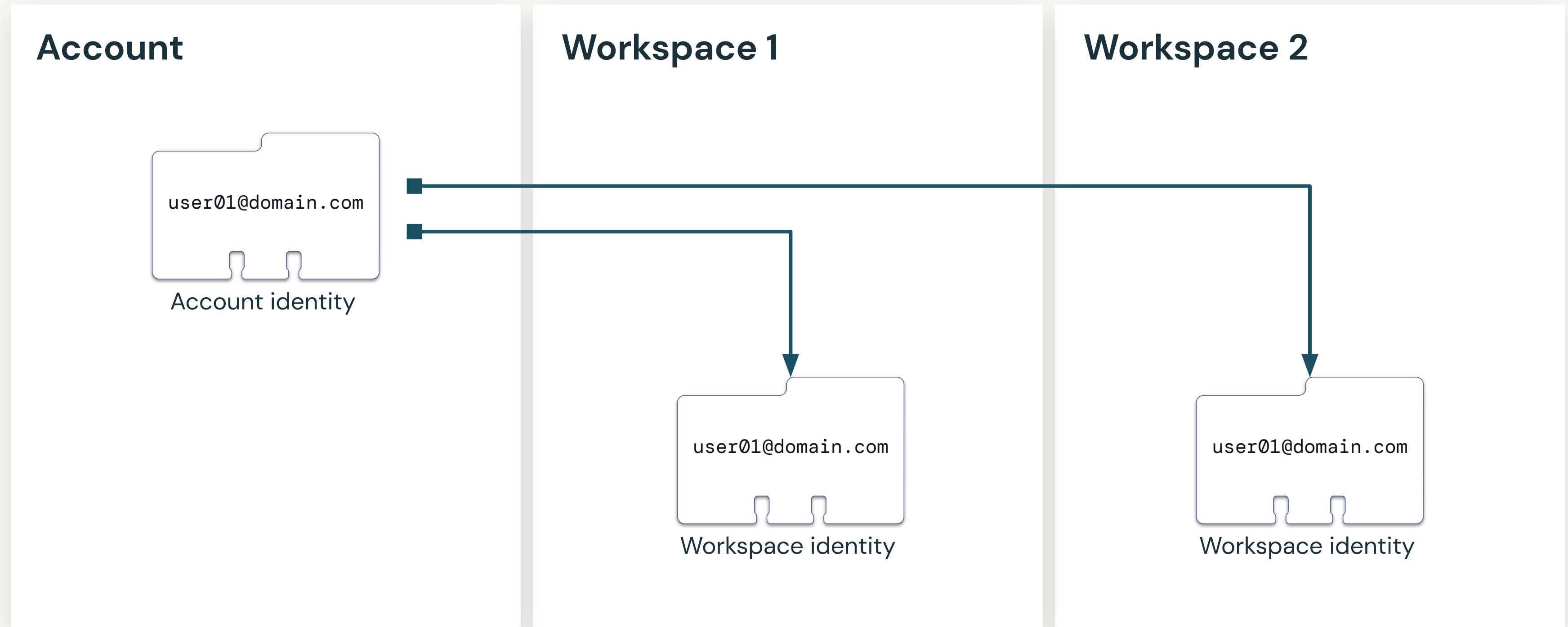
Identities

- Groups



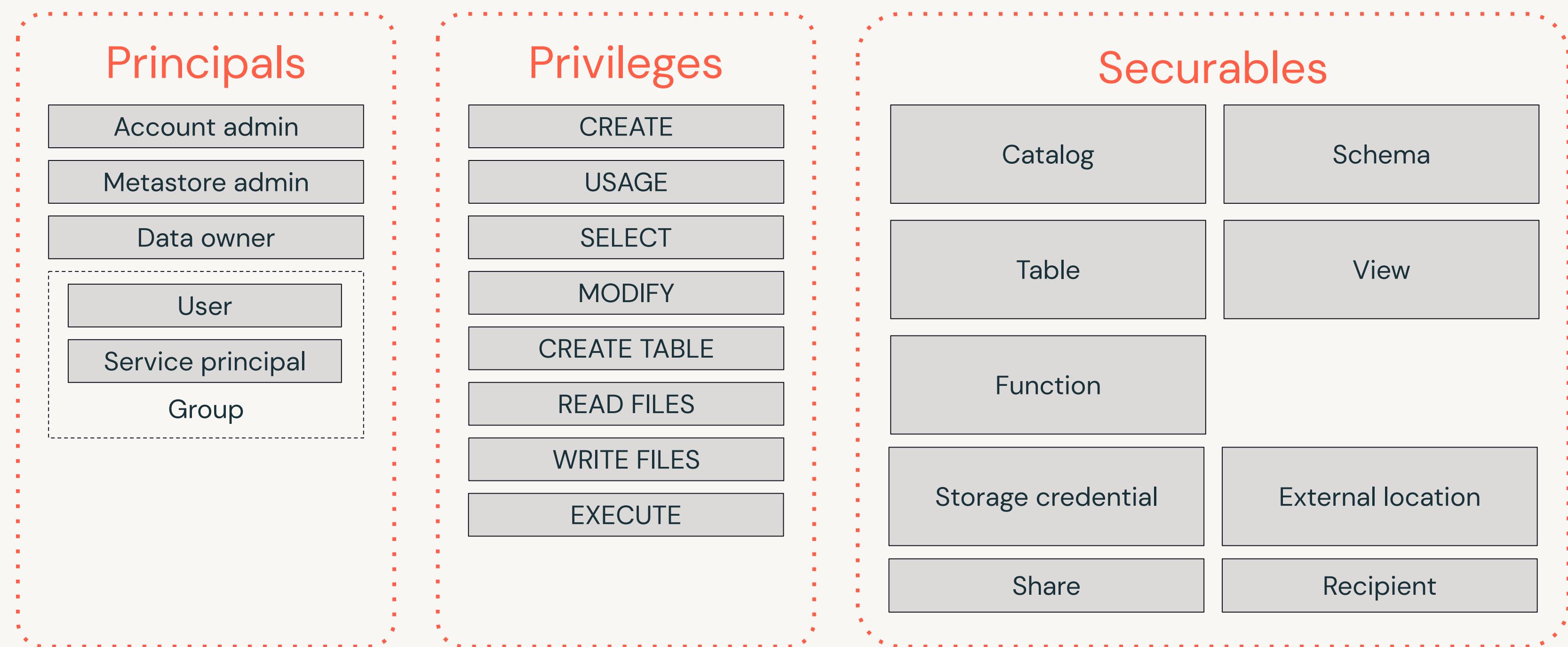
Unity Catalog

Identity Federation

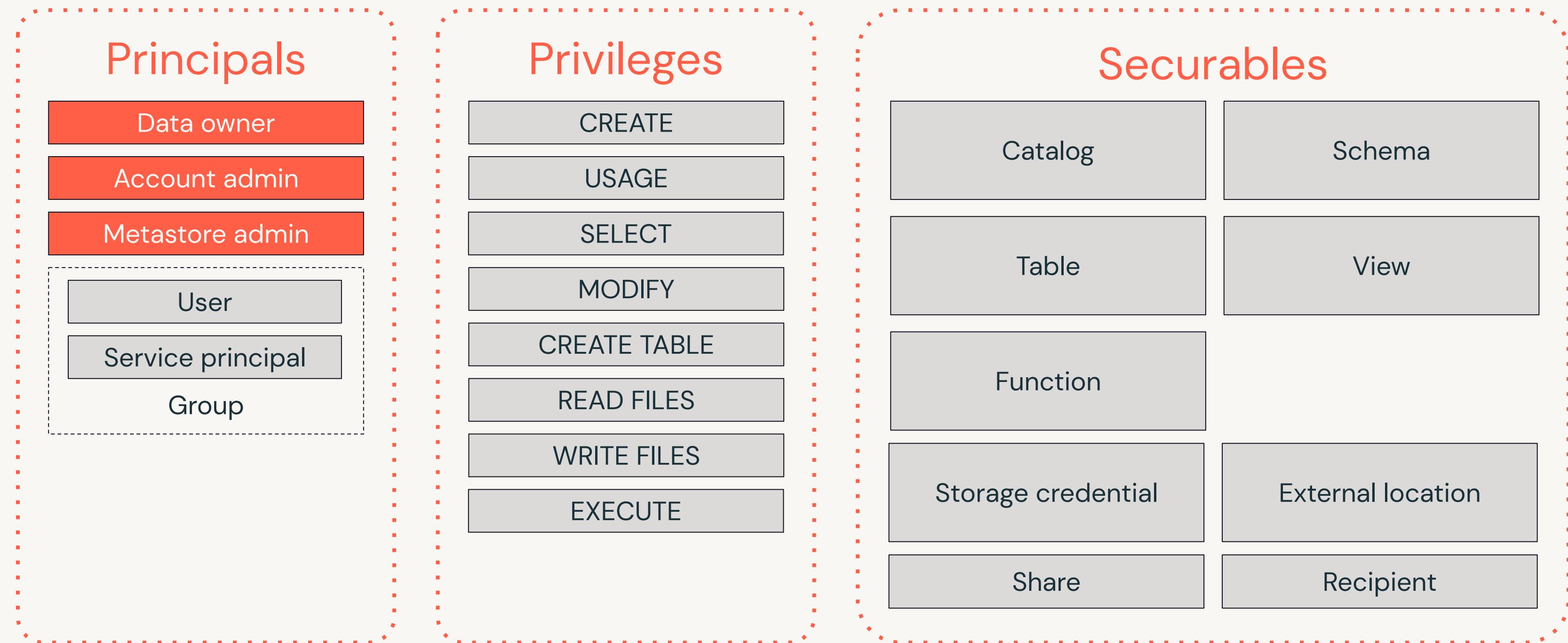


Data Access Control in Unity Catalog

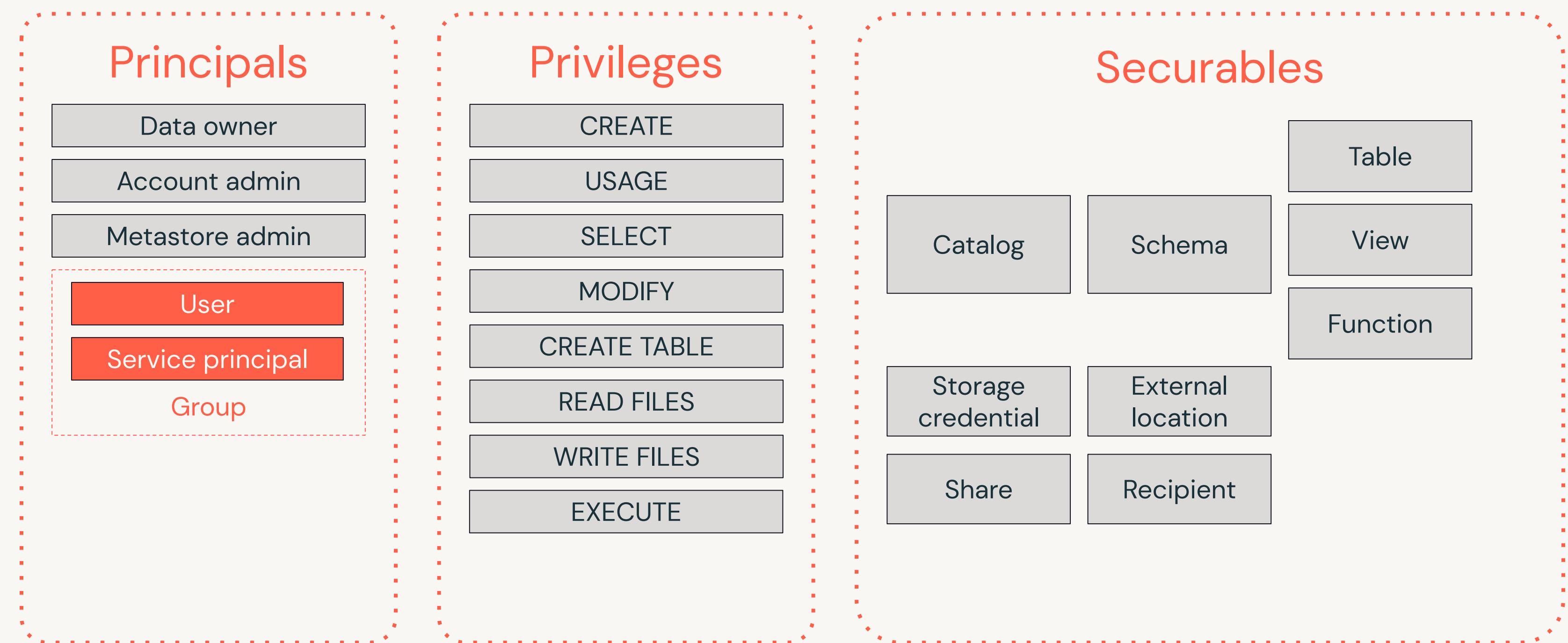
Security model



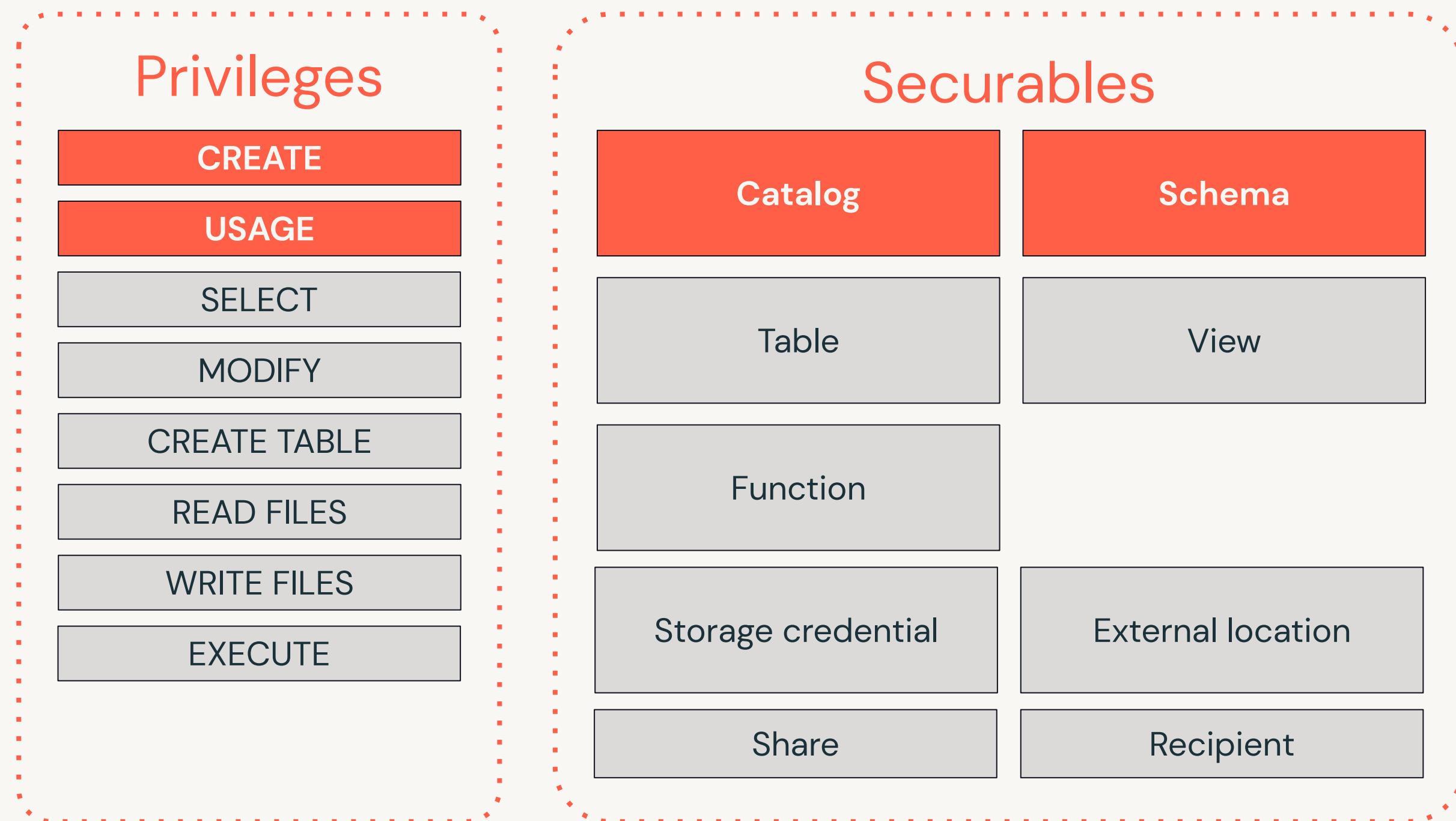
Security model



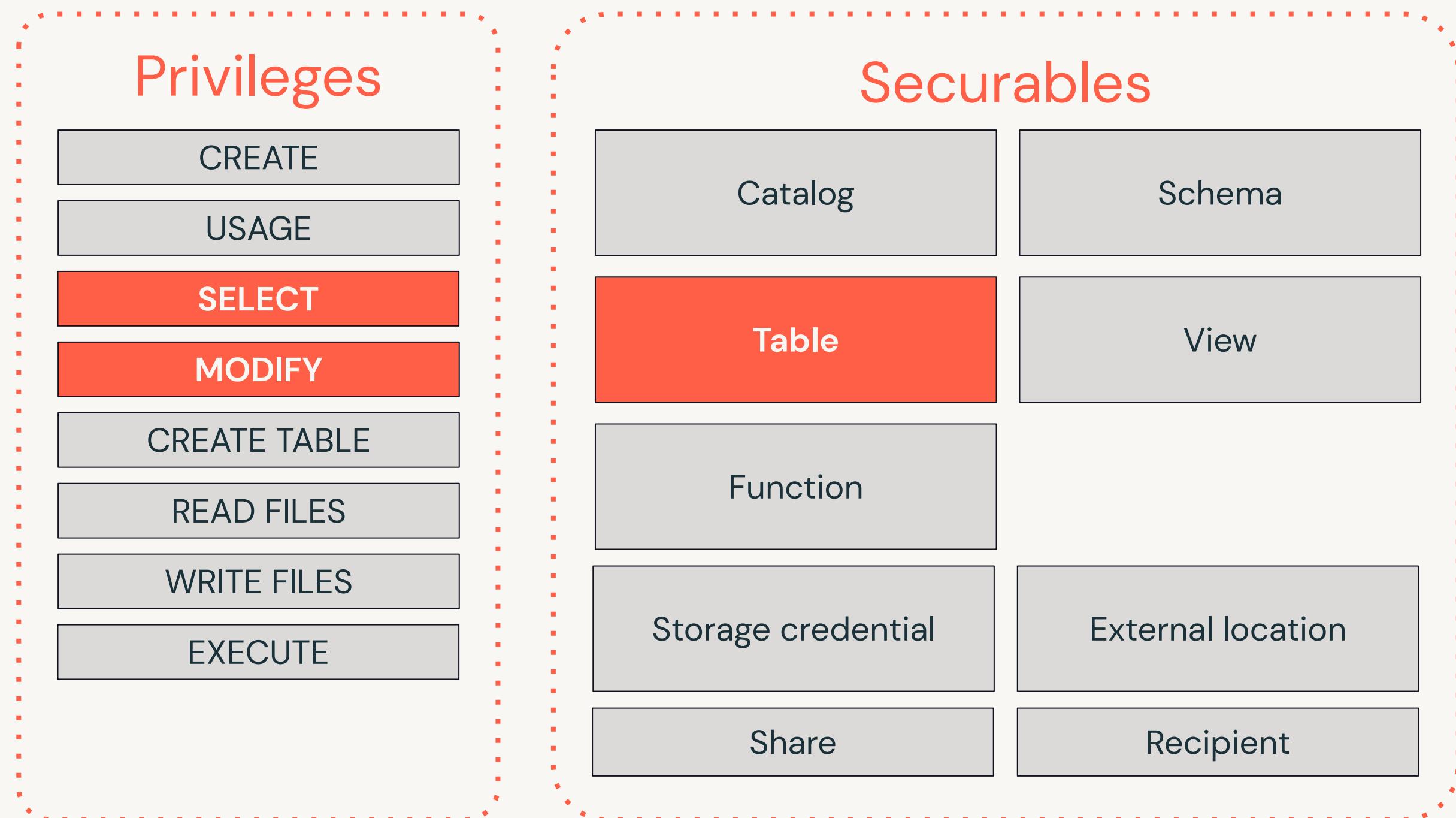
Security model



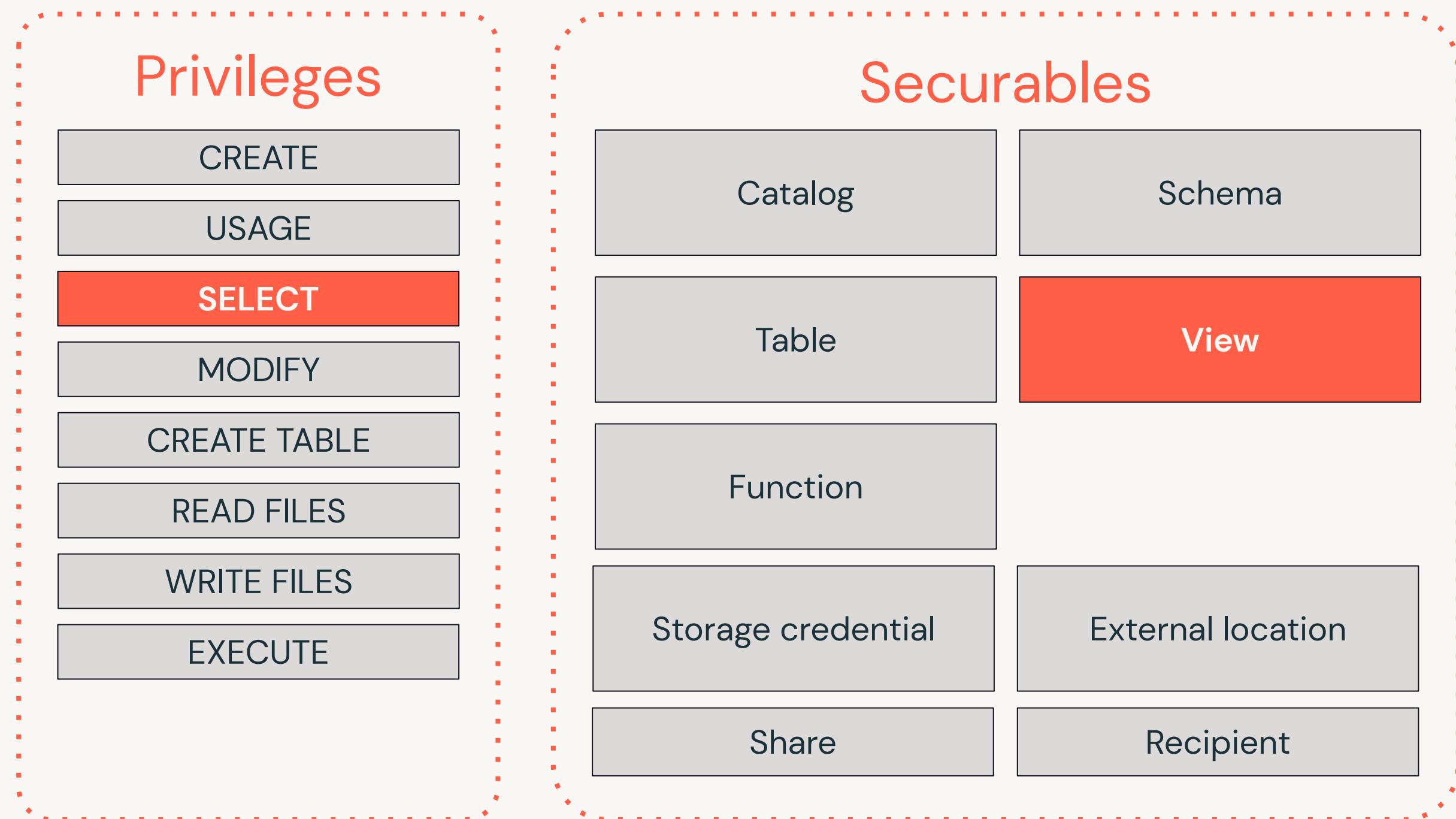
Security model



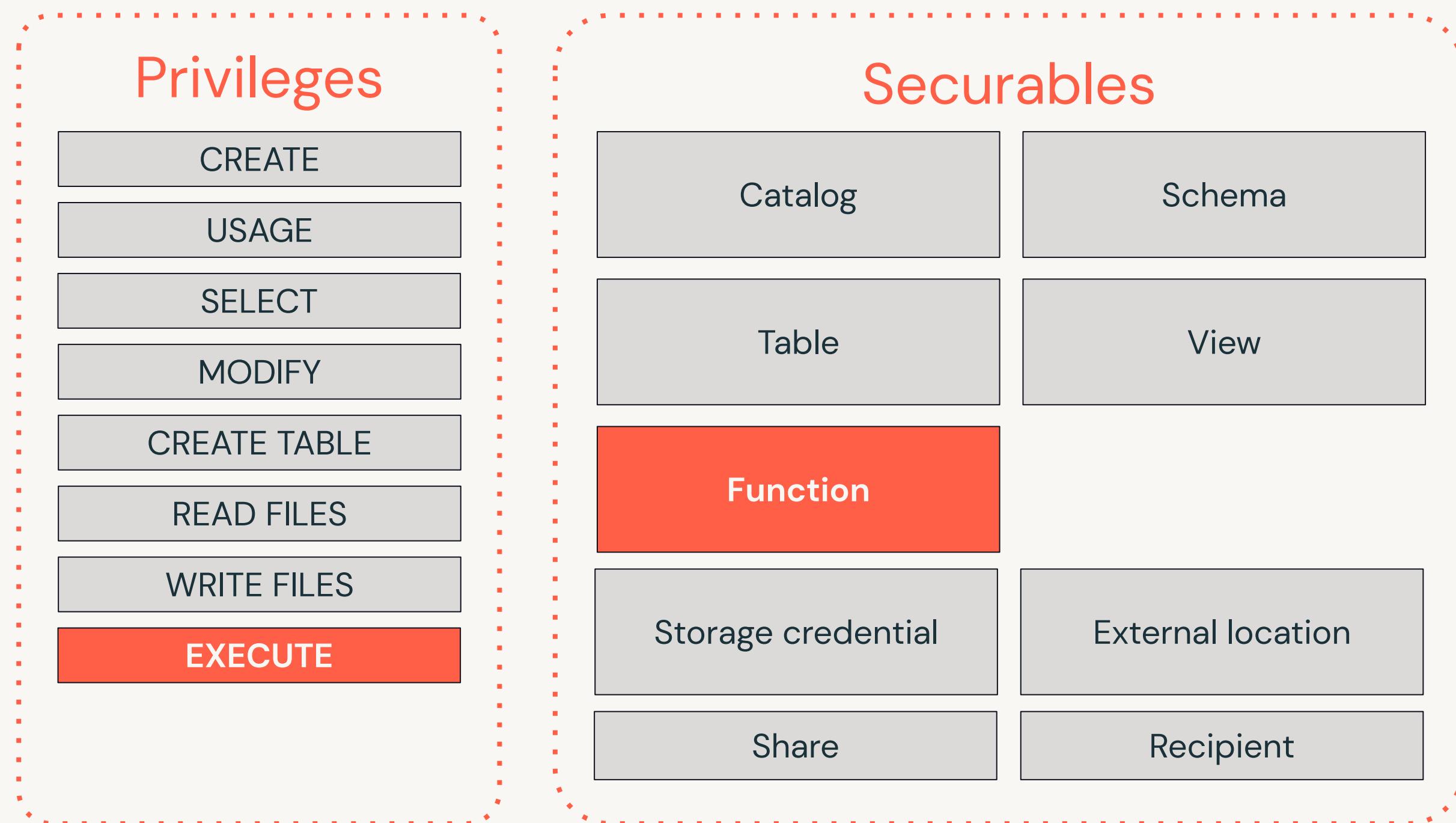
Security model



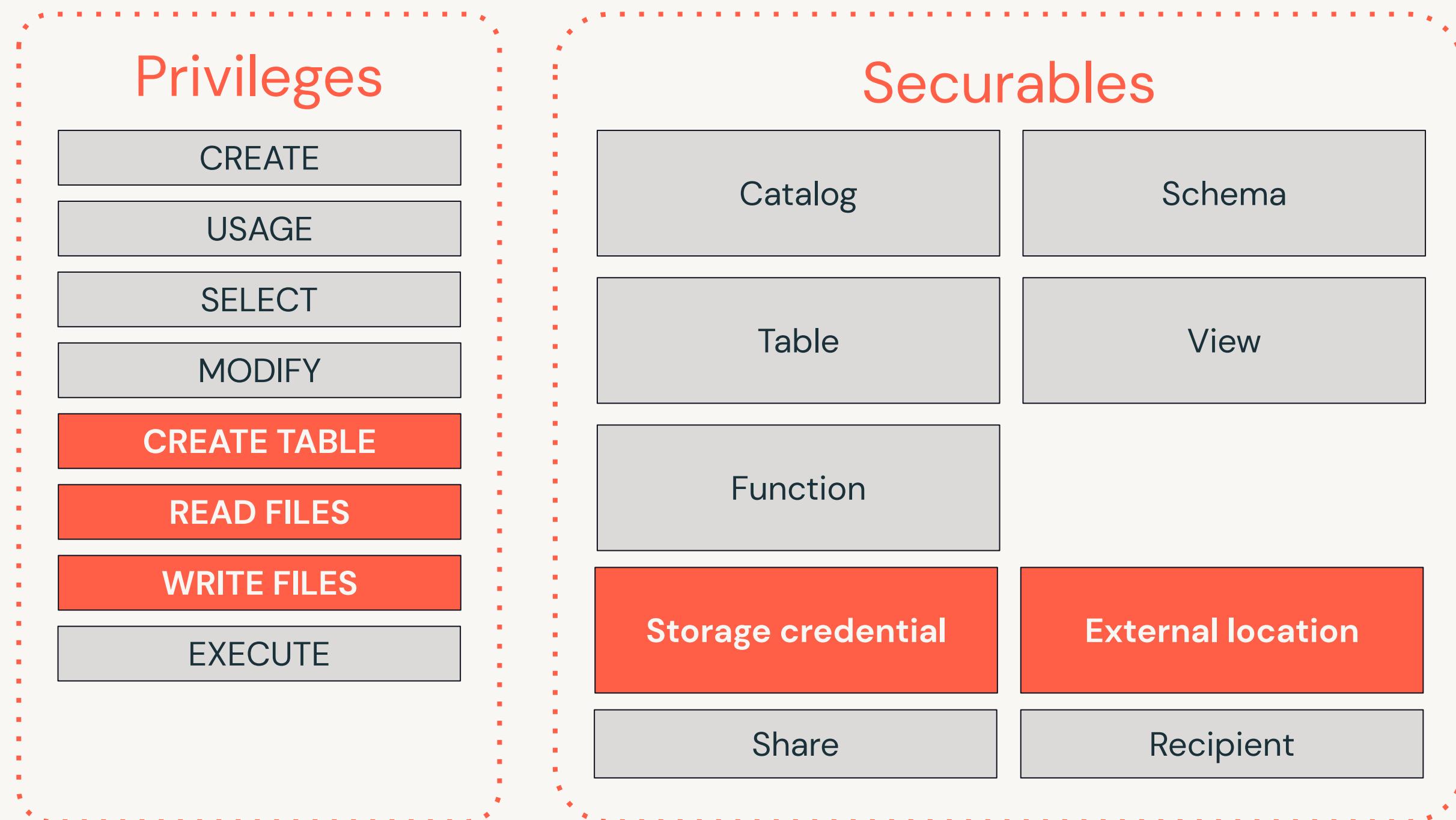
Security model



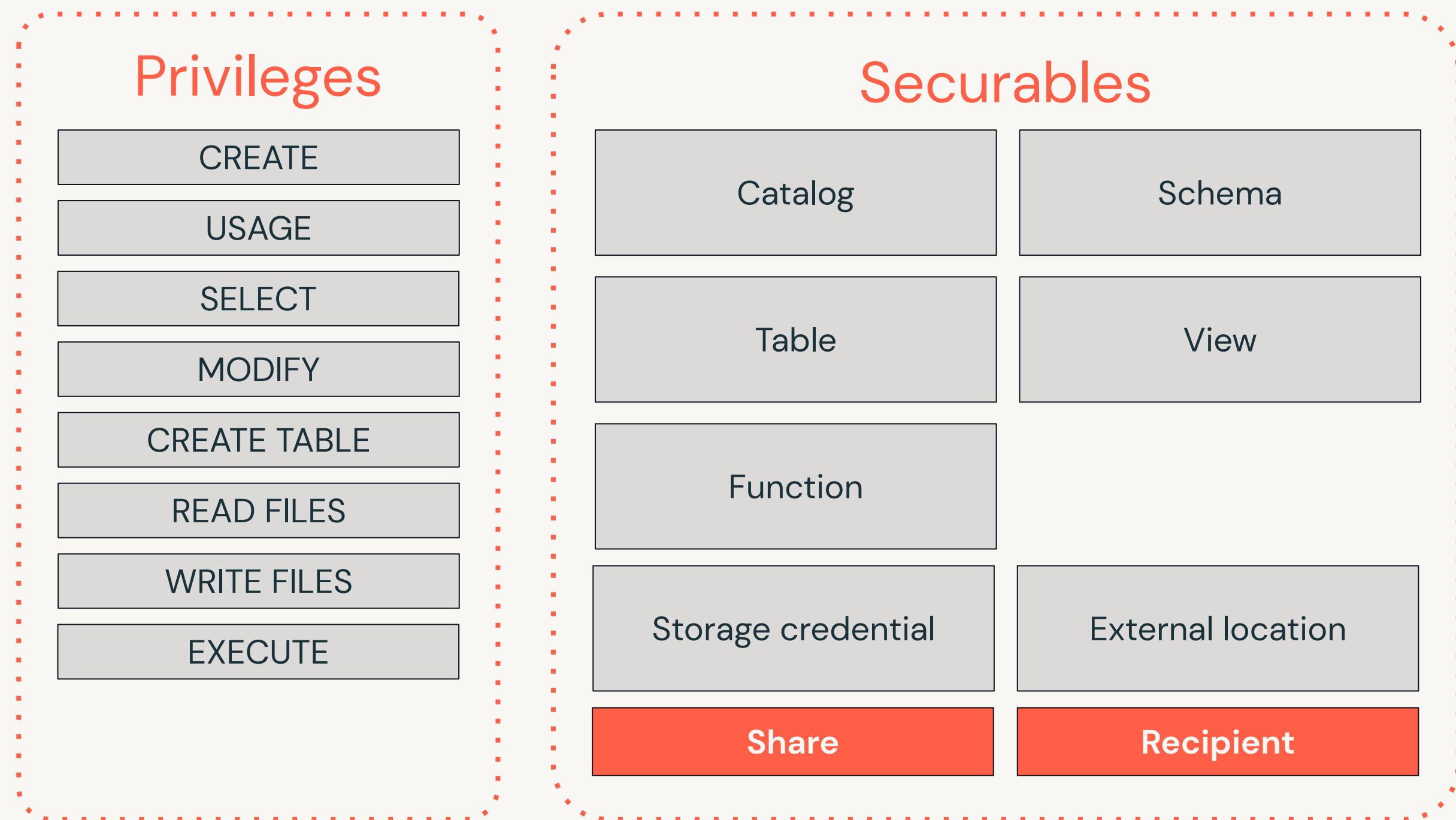
Security model



Security model

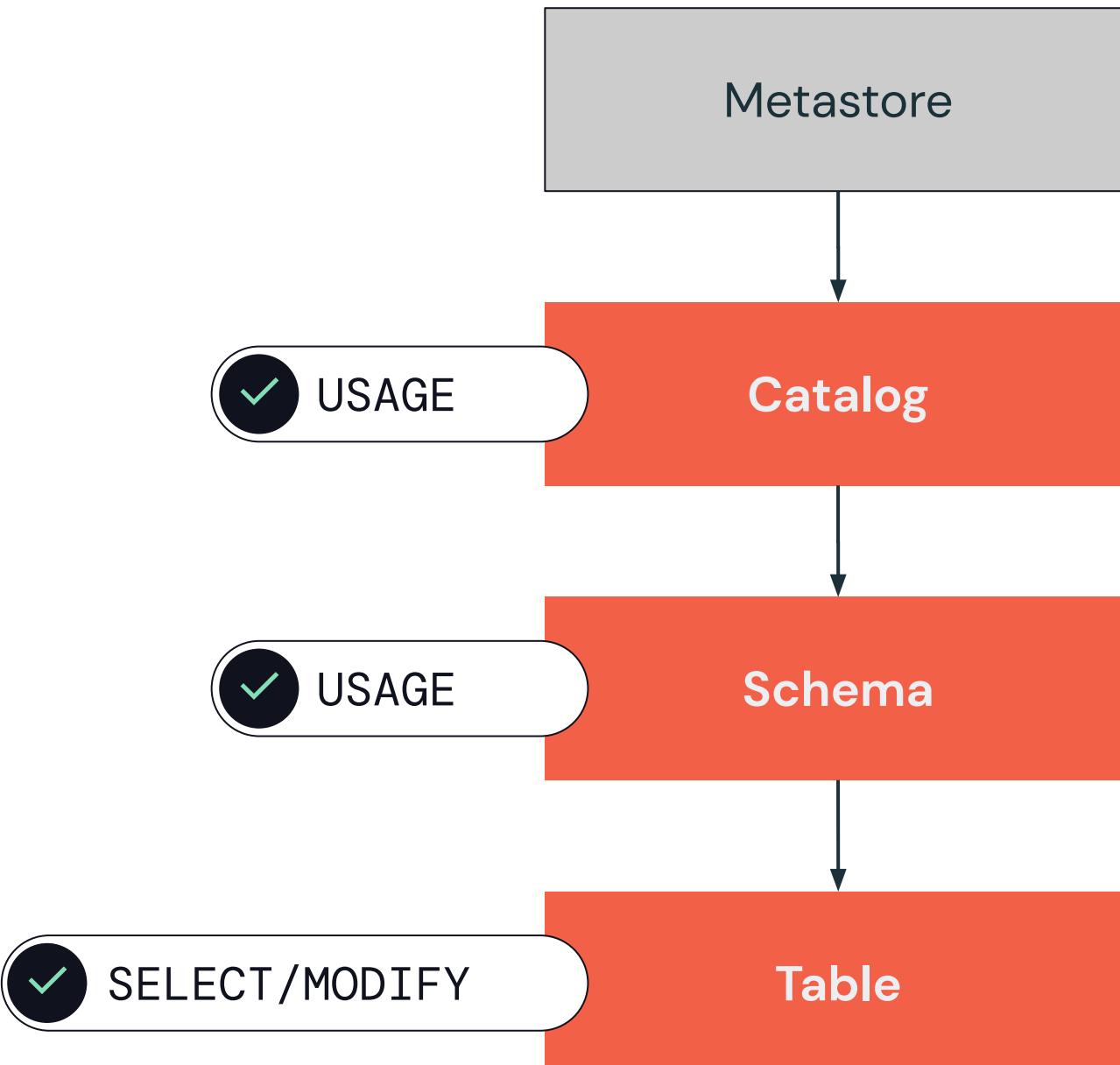


Security model



Privilege Recap

Tables



Querying tables (**SELECT**)

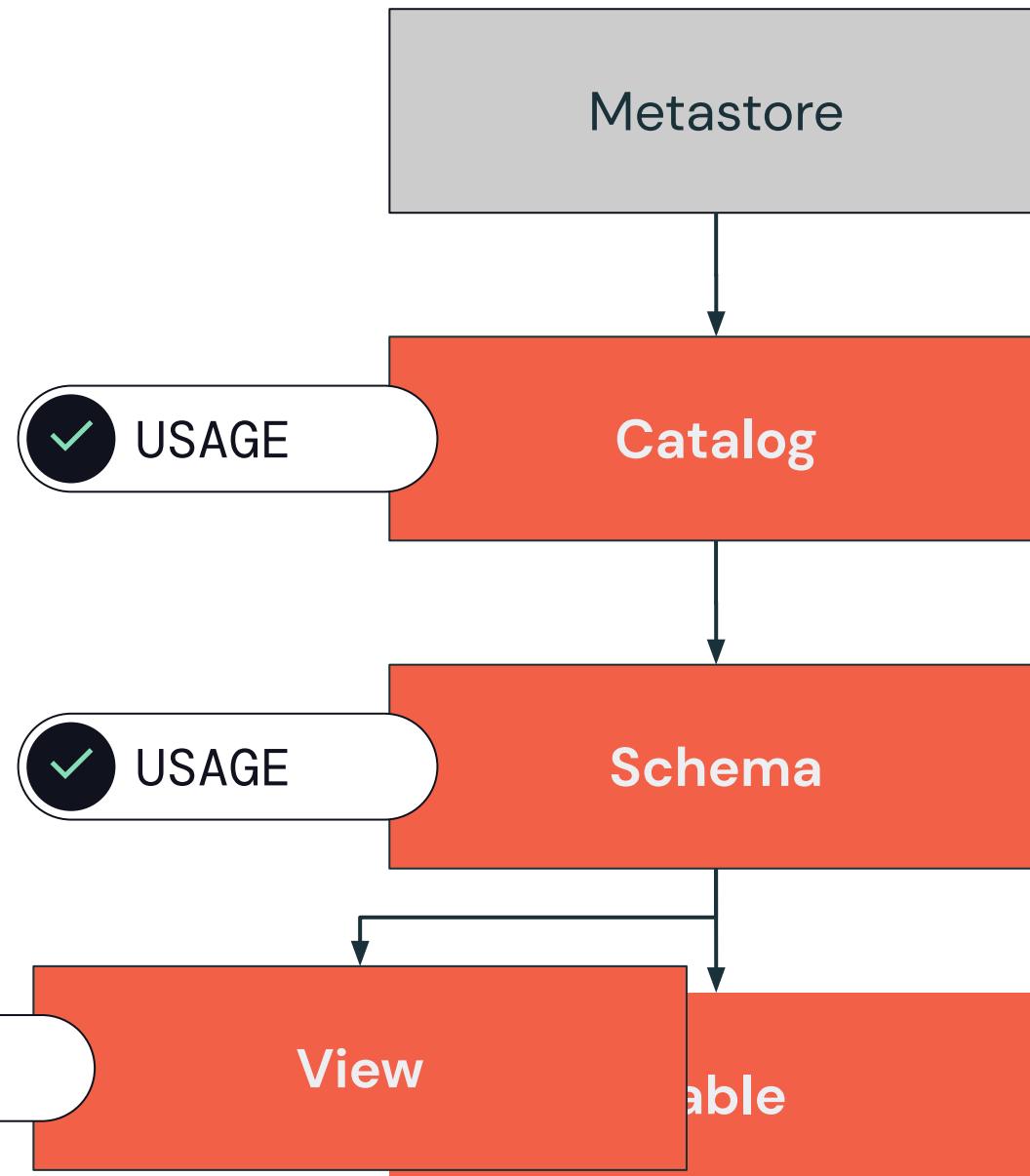
Modifying tables (**MODIFY**)

- Data (INSERT, DELETE)
- Metadata (ALTER)

Traversing containers (**USAGE**)

Privilege Recap

Views



Abstract complex queries

- Aggregations
- Transformations
- Joins
- Filters

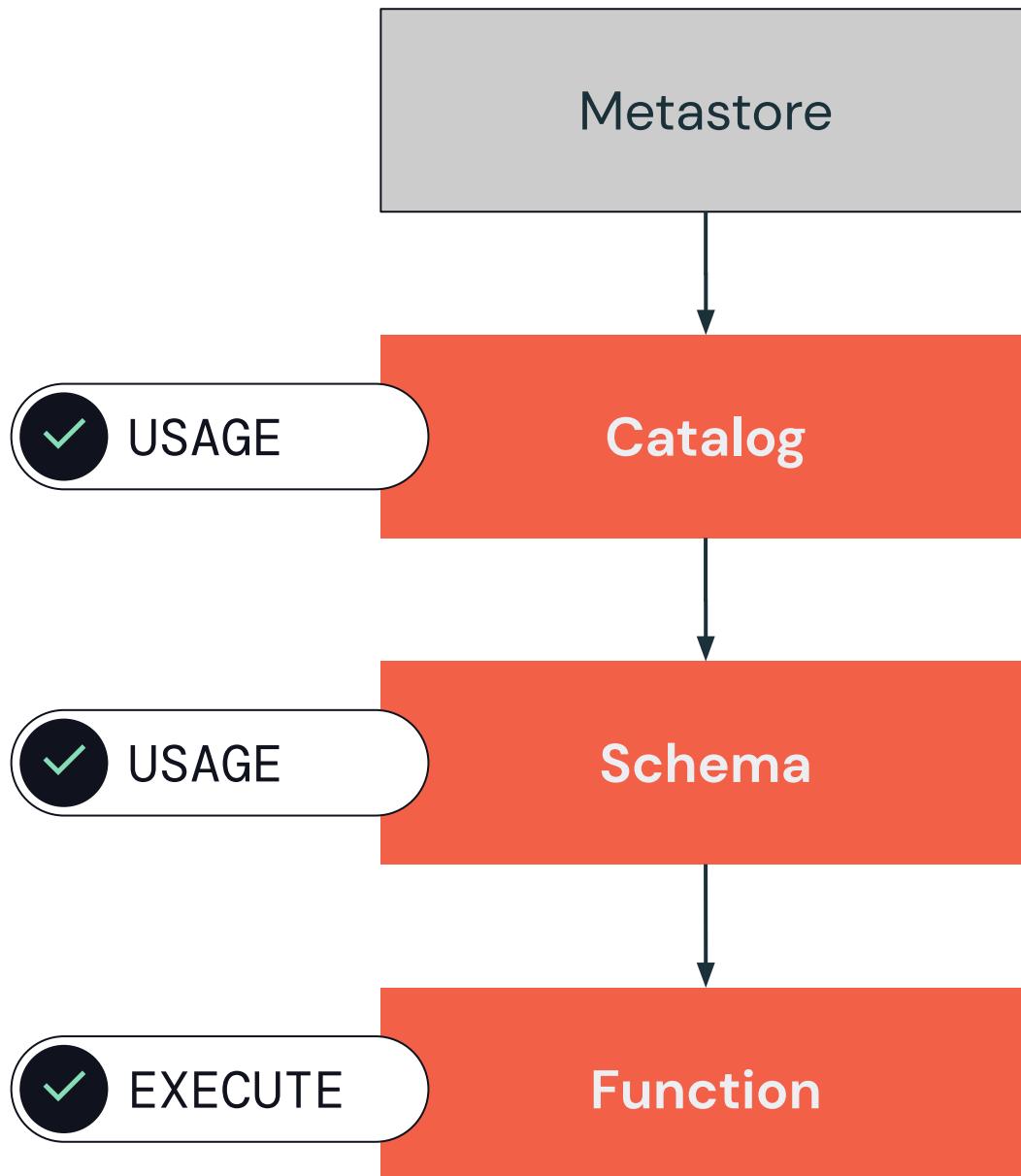
Enhanced table access control

Querying views (**SELECT**)

Traversing containers (**USAGE**)

Privilege Recap

Functions



Provide custom code via user-defined functions

Using functions (**EXECUTE**)

Traversing containers (**USAGE**)

Dynamic Views

Limit access to columns

Omit column values from output

■	■	■
■	■	■
■	■	■
■	■	■
■	■	■

Limit access to rows

Omit rows from output

■	■	■	■
■	■	■	■
■	■	■	■
■	■	■	■
■	■	■	■

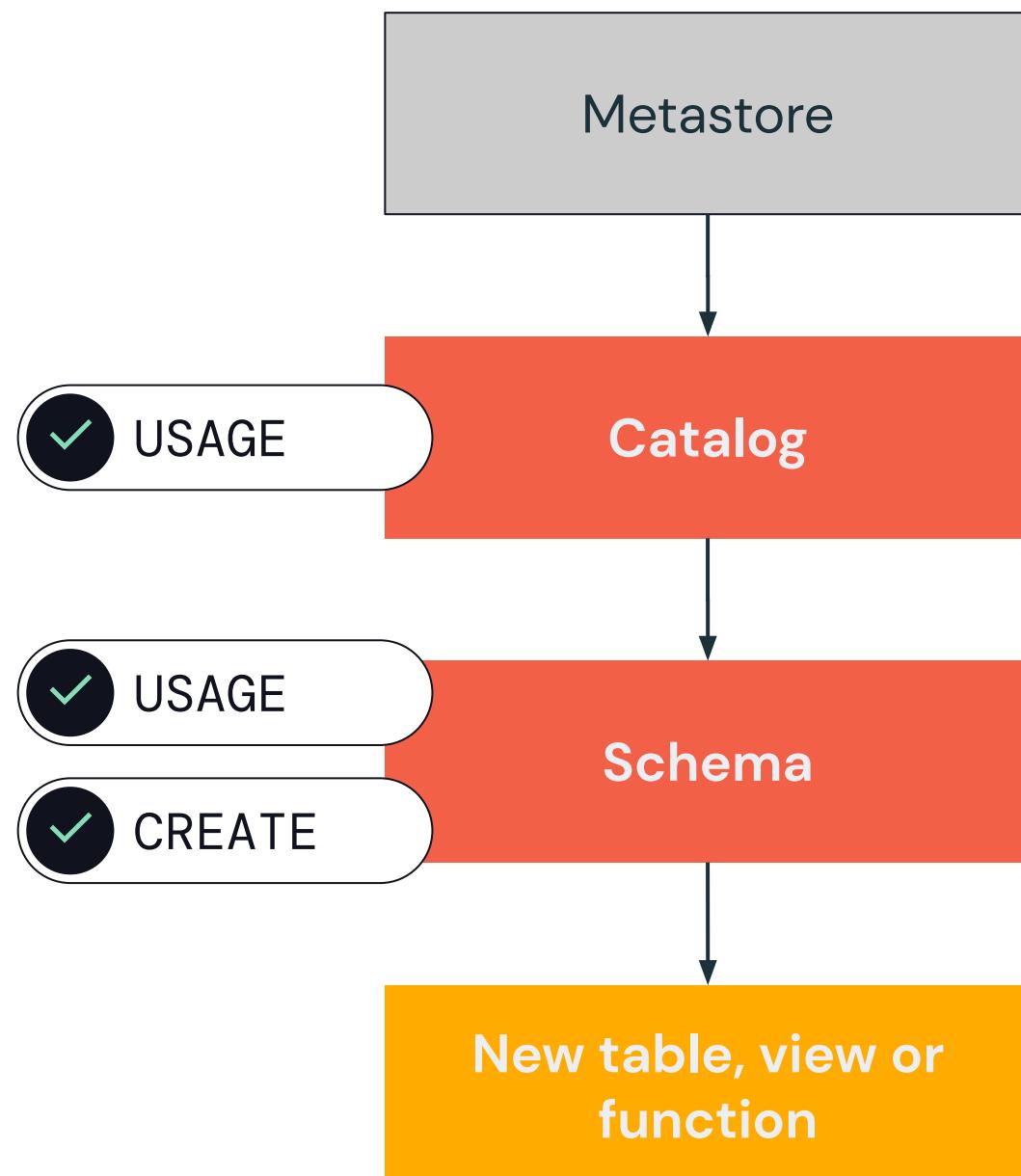
Data Masking

Obscure data

•••••@databricks.com

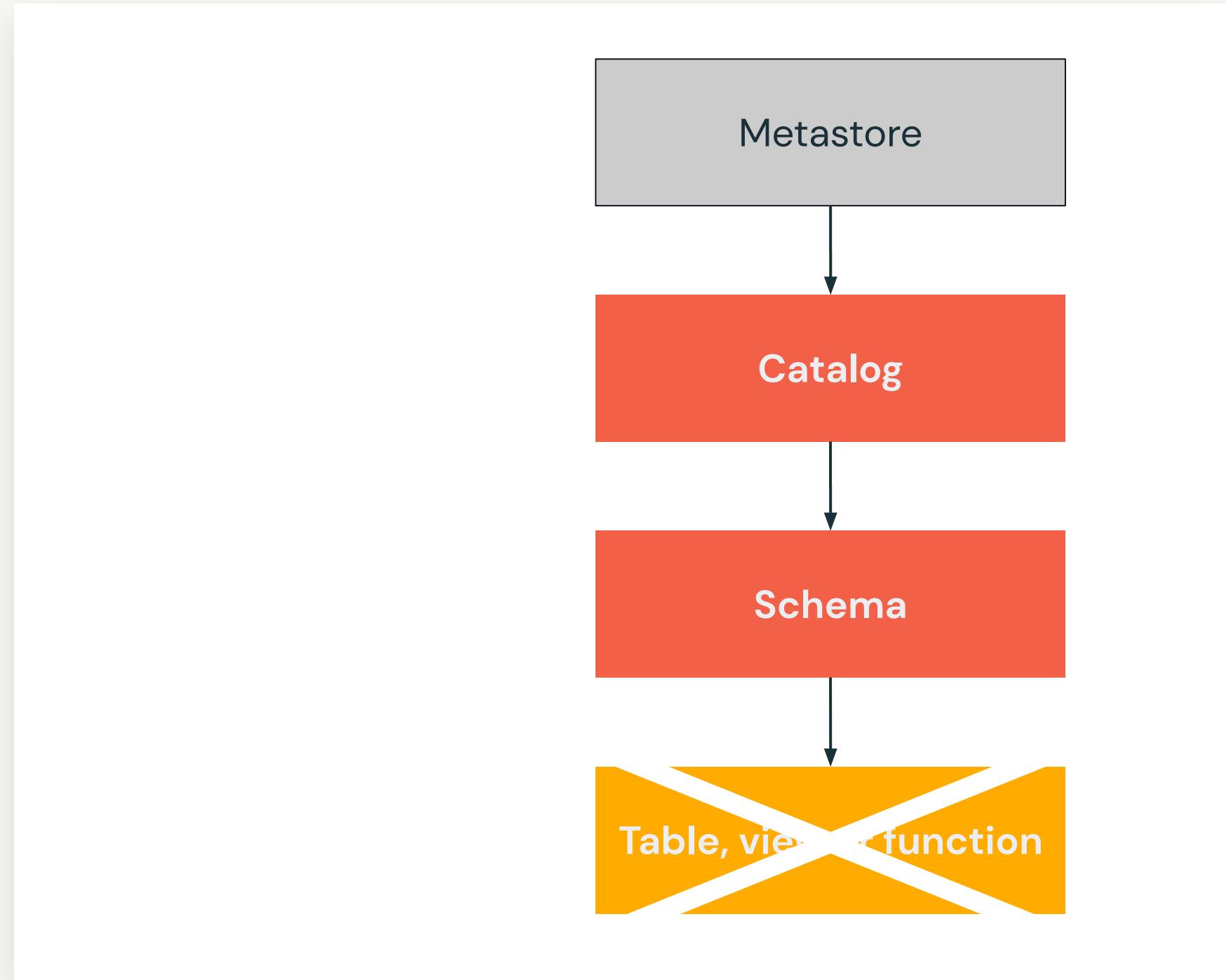
Can be conditional on a specific user/service principal or group membership through Databricks-provided functions

Creating New Objects



Creating new objects (**CREATE**)
Traversing containers (**USAGE**)

Deleting Objects



DROP objects

Unity Catalog External Storage

Storage Credentials and External Locations

Storage Credential

Enables Unity Catalog to connect to an external cloud storage

Examples include:

- IAM role for AWS S3
- Service principal for Azure Storage

External Location

Cloud storage path + storage credential

- Self-contained object for accessing specific locations in cloud storage
- Fine-grained control over external storage



Storage Credentials and External Locations

Access Control

CREATE TABLE

Create an External Table directly using this Storage Credential

READ FILES

Read files directly using this Storage Credential

WRITE FILES

Write files directly using this Storage Credential

Storage Credential

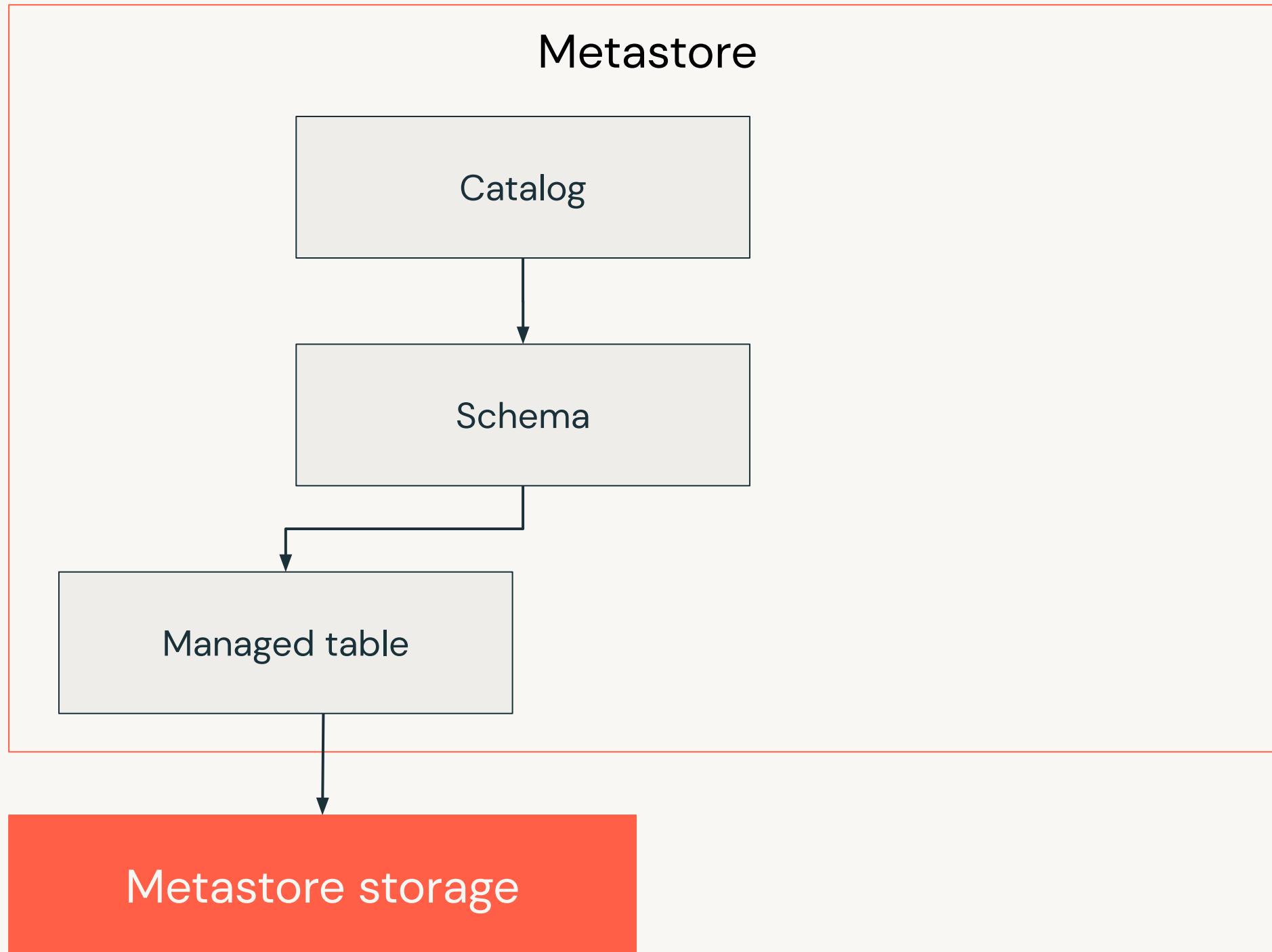
Create an External Table from files governed by this External Location

External Location

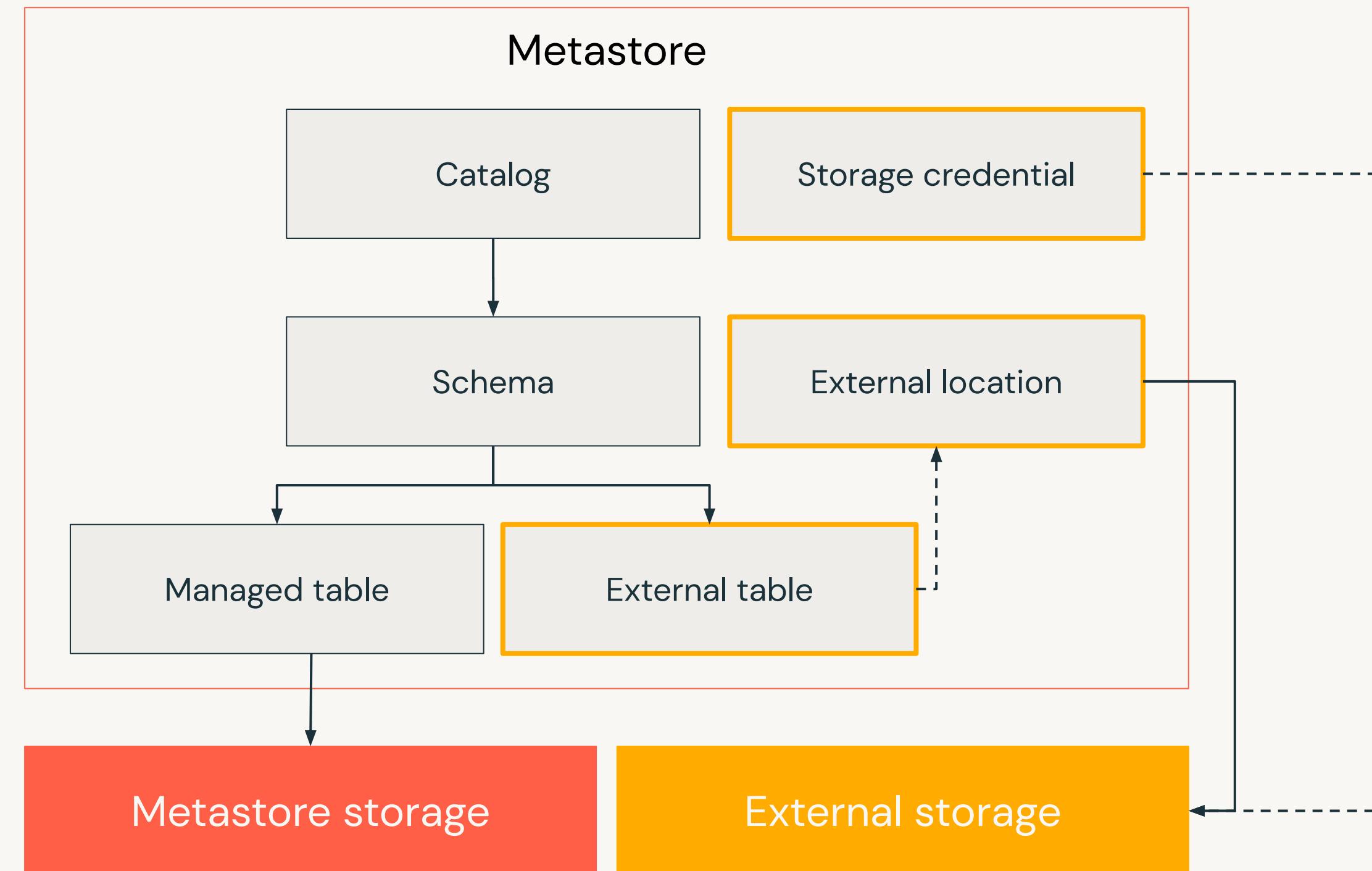
Read files governed by this External Location

Write files governed by this External Location

Managed Tables



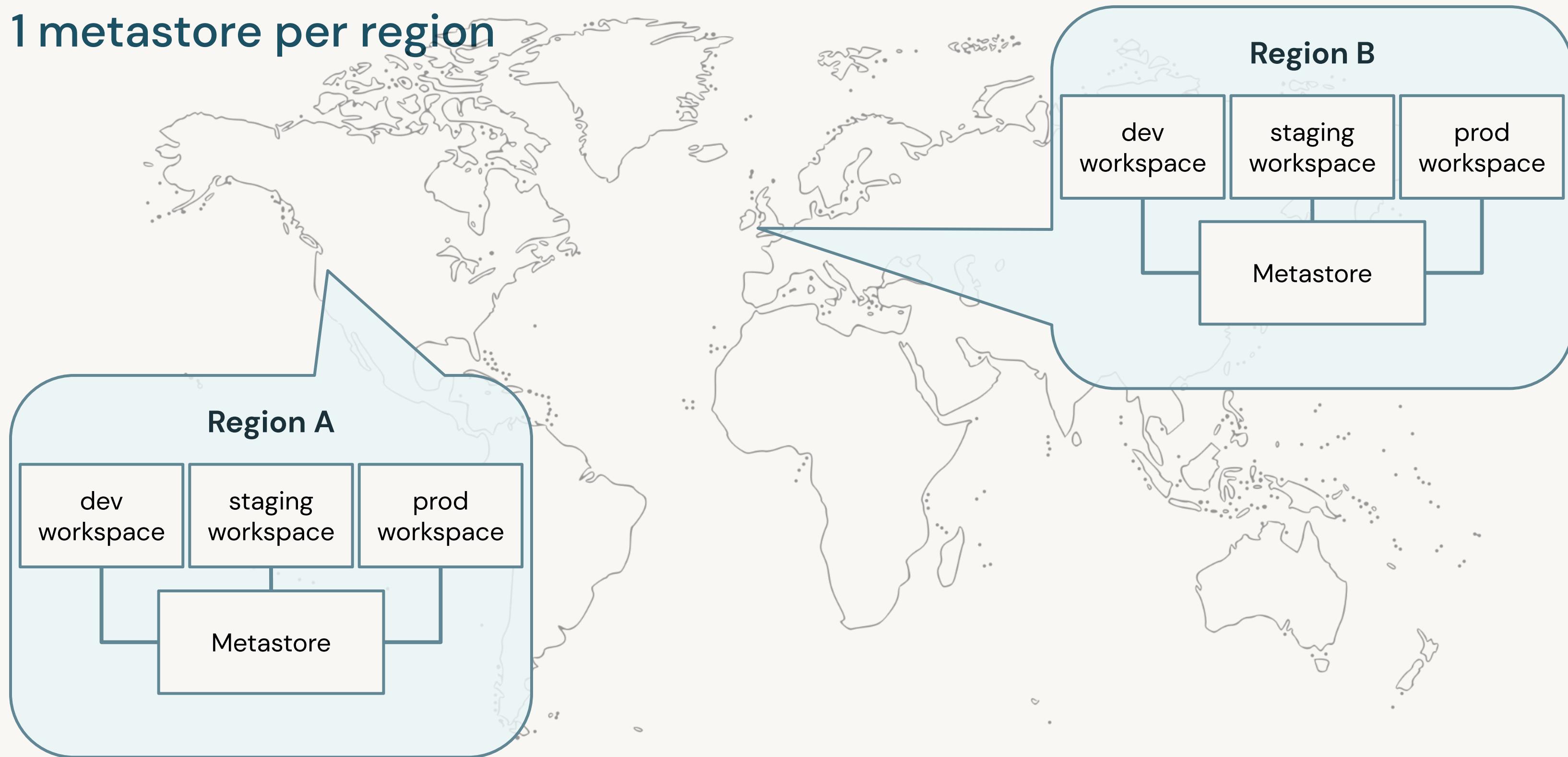
External Tables



Unity Catalog Patterns and Best Practices

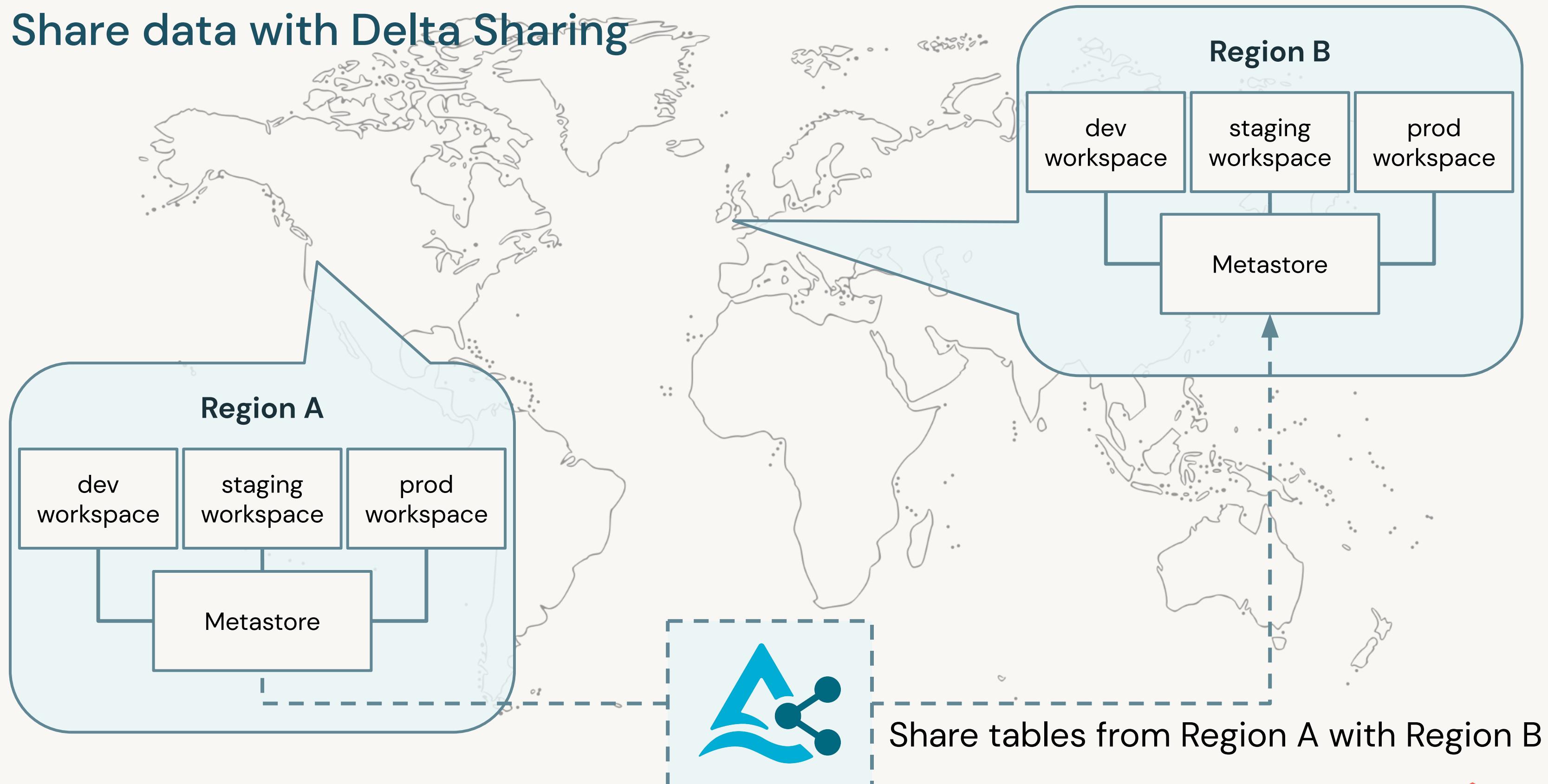
UC Patterns & Best Practices

1 metastore per region



UC Patterns & Best Practices

Share data with Delta Sharing



UC Patterns & Best Practices

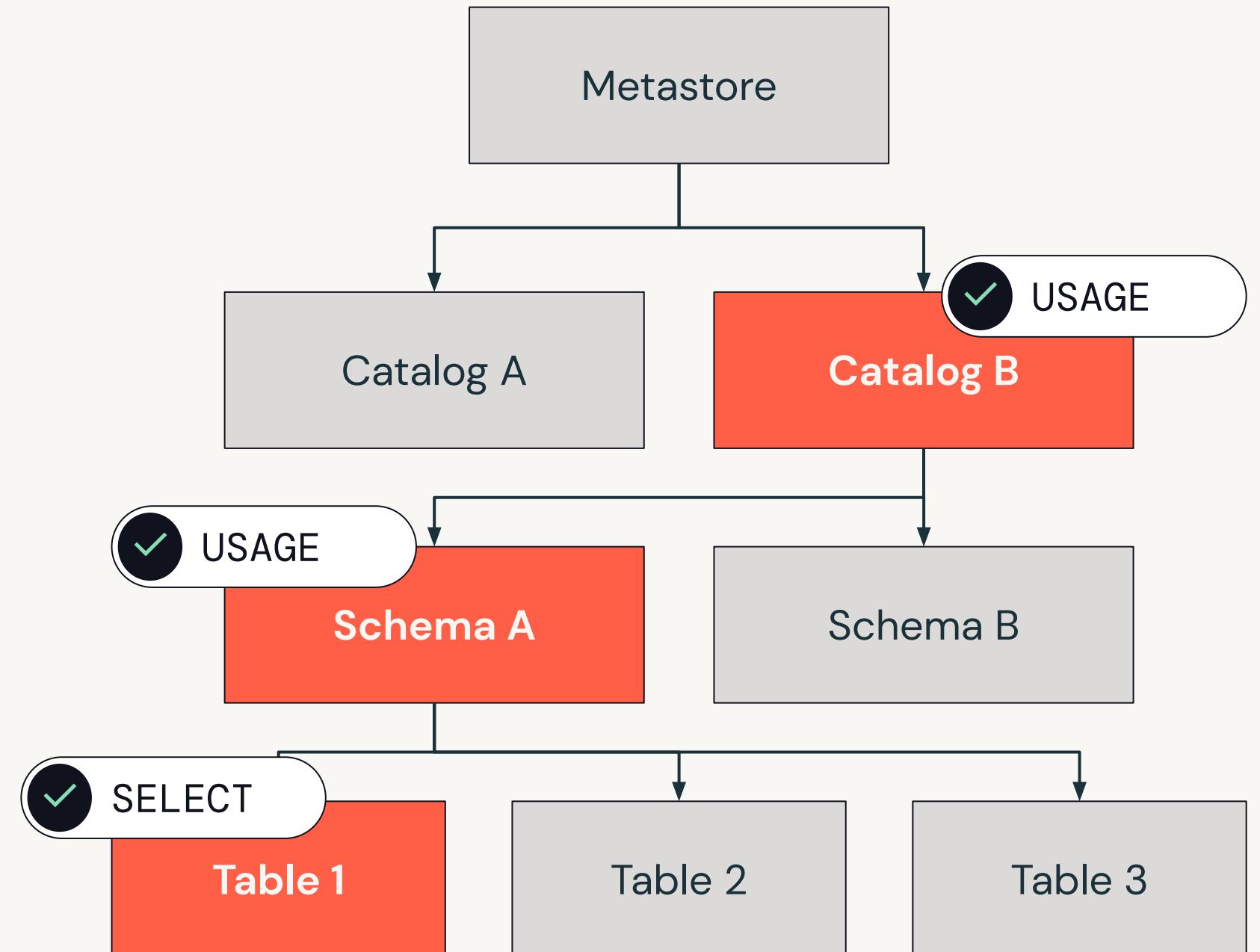
Data Segregation

Use **catalogs** (not metastores) to segregate data

Apply permissions appropriately

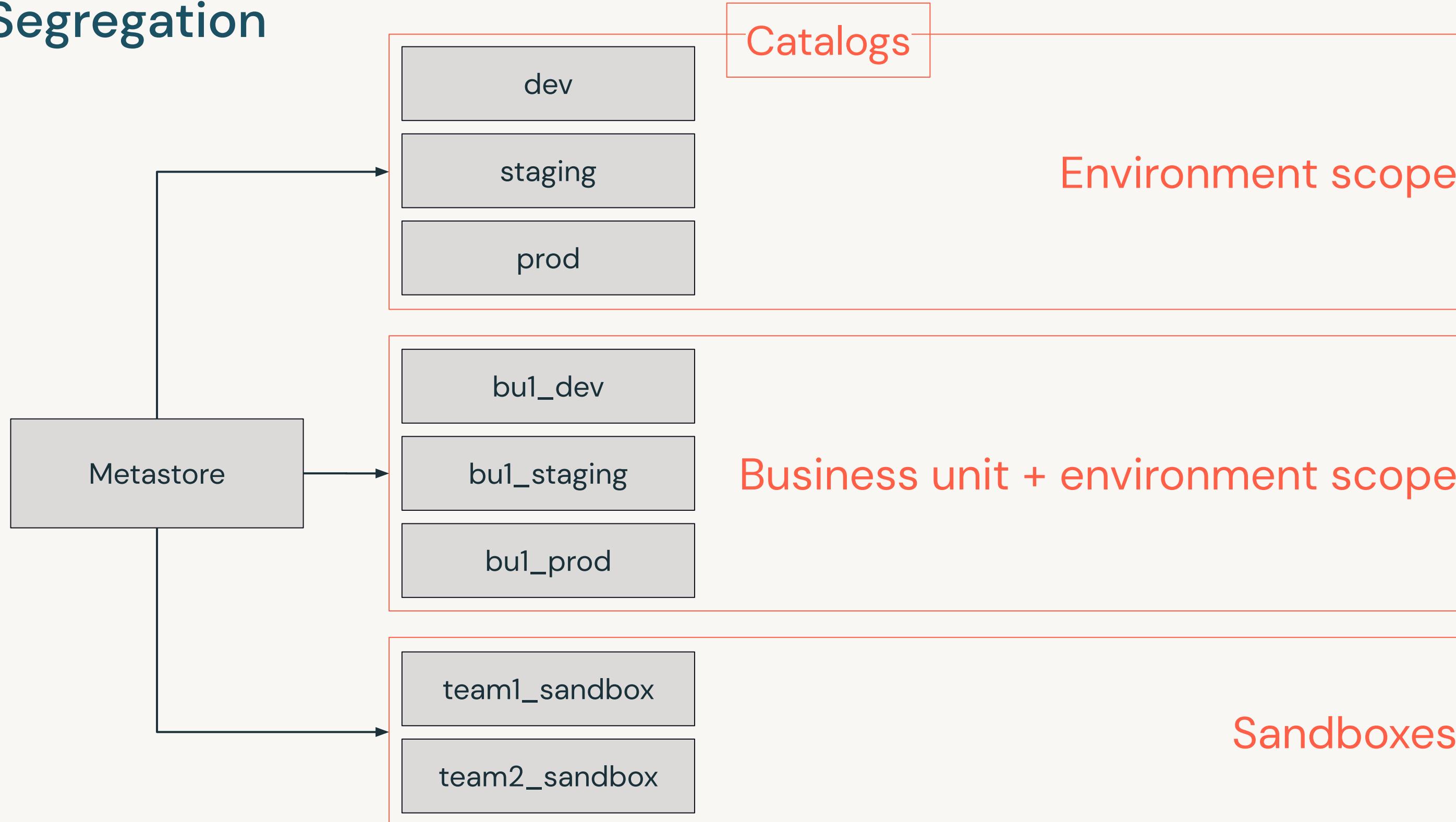
For example, grant to group B:

- **USAGE** on catalog B
- **USAGE** on all applicable schemas in catalog B
- **SELECT/MODIFY** on applicable tables



UC Patterns & Best Practices

Data Segregation



UC Patterns & Best Practices

Identity Management

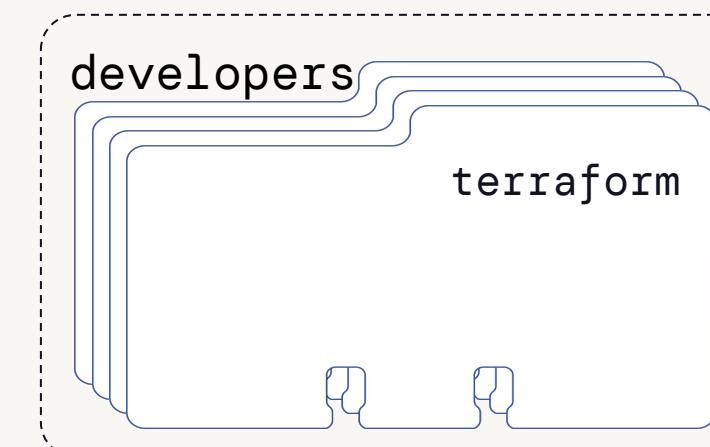
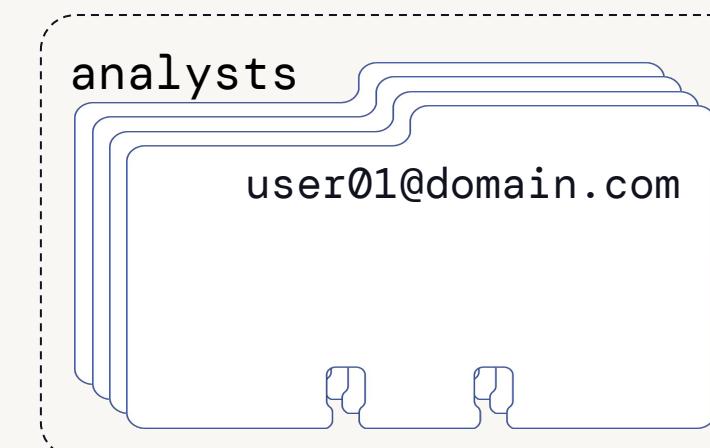
Account-level Identities

Manage all identities at the account-level

Enable UC for workspaces to enable identity federation

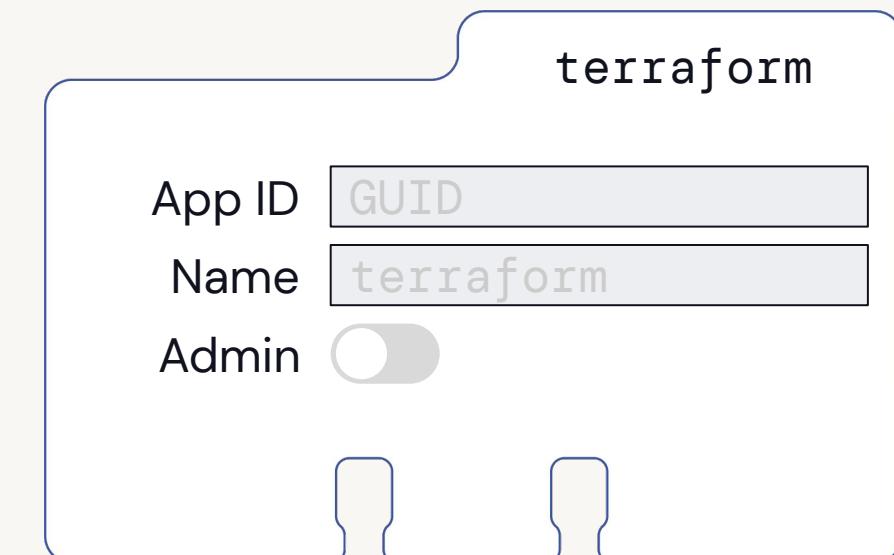
Groups

Use groups rather than users to assign access and ownership to securable objects



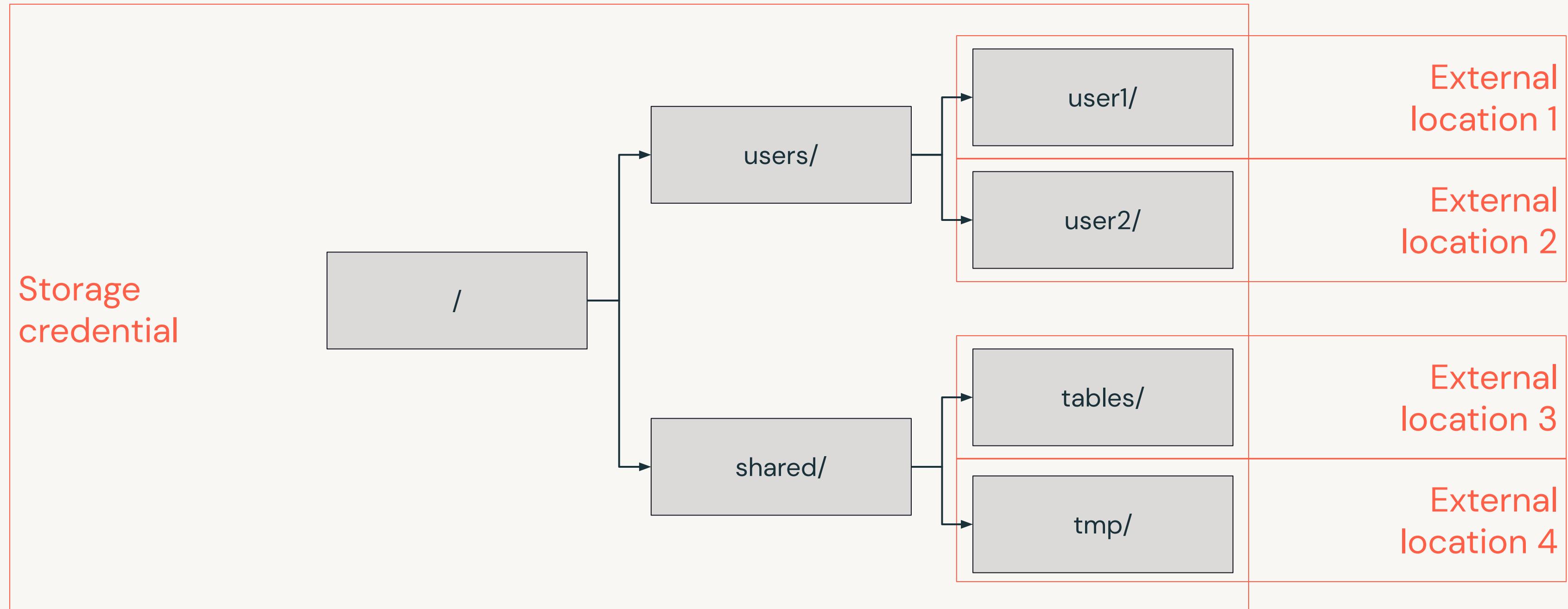
Service Principals

Use service principals to run production jobs



UC Patterns & Best Practices

Storage Credentials and External Locations



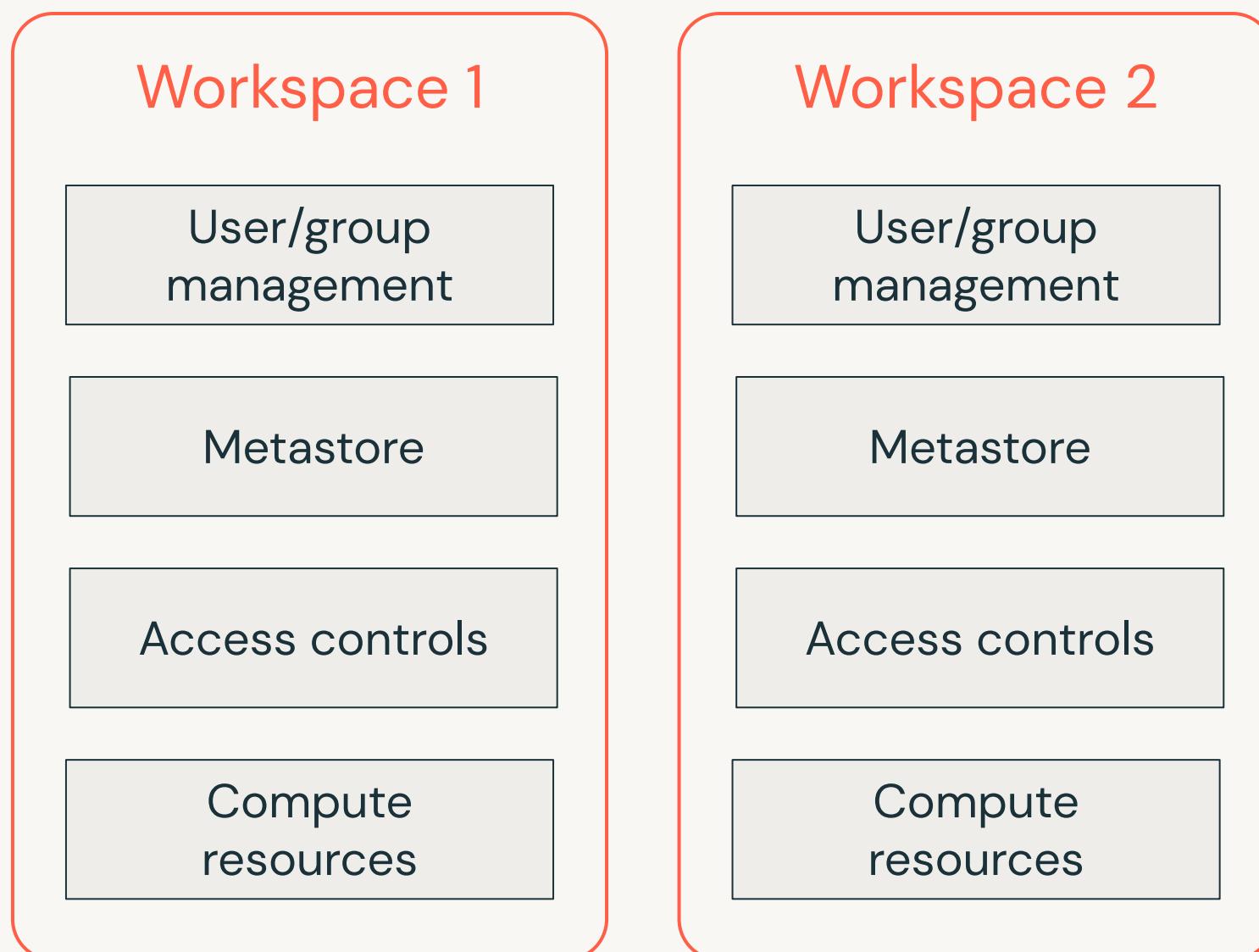


Unity Catalog Key Capabilities

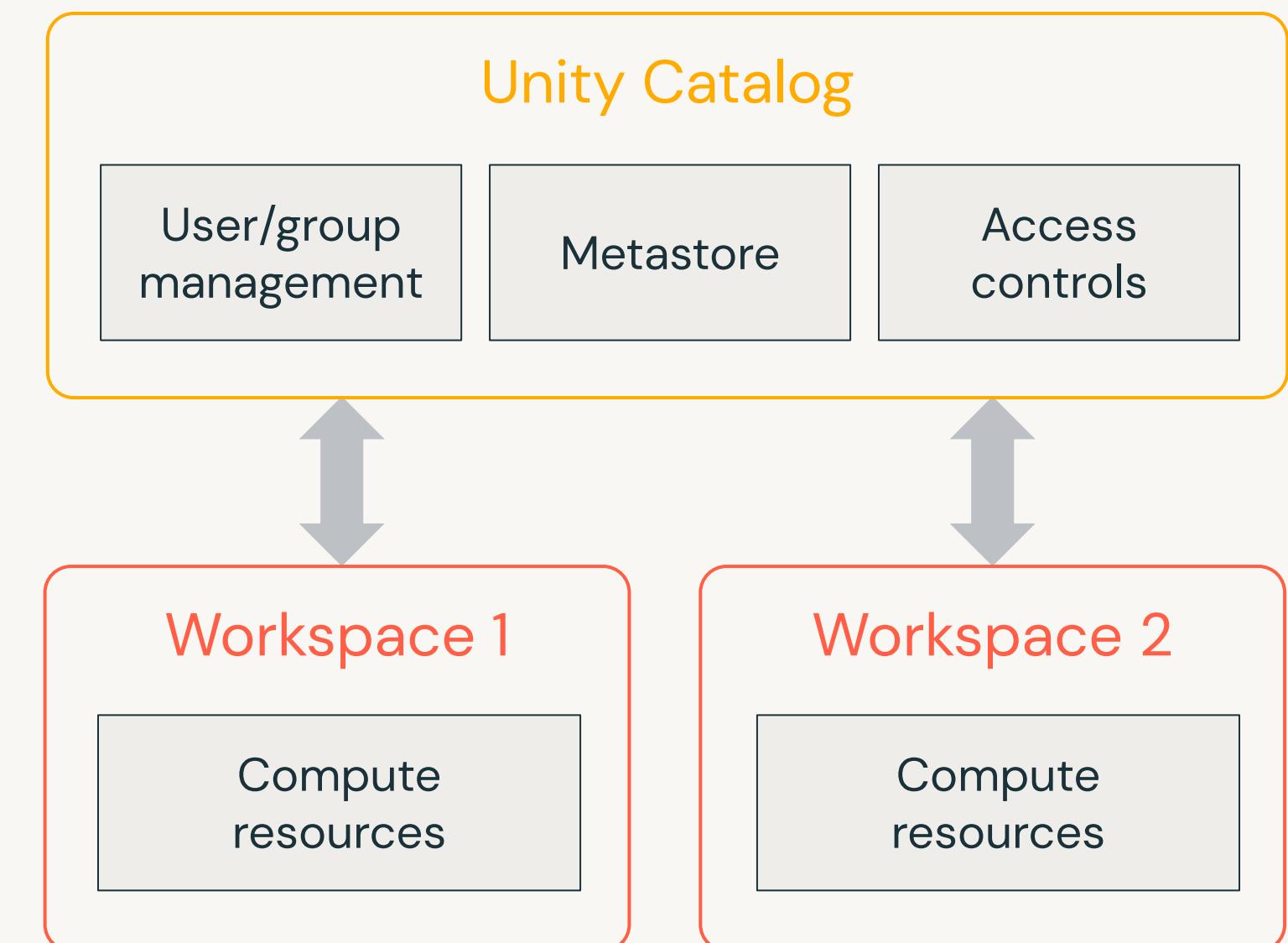
Centralized metadata and user management

Unity Catalog Architecture

Before Unity Catalog



With Unity Catalog



Centralized Access Controls

Centrally grant and manage access permissions across workloads

Using ANSI SQL DCL

```
GRANT <privilege> ON <securable_type>  
<securable_name> TO `<principal>`
```

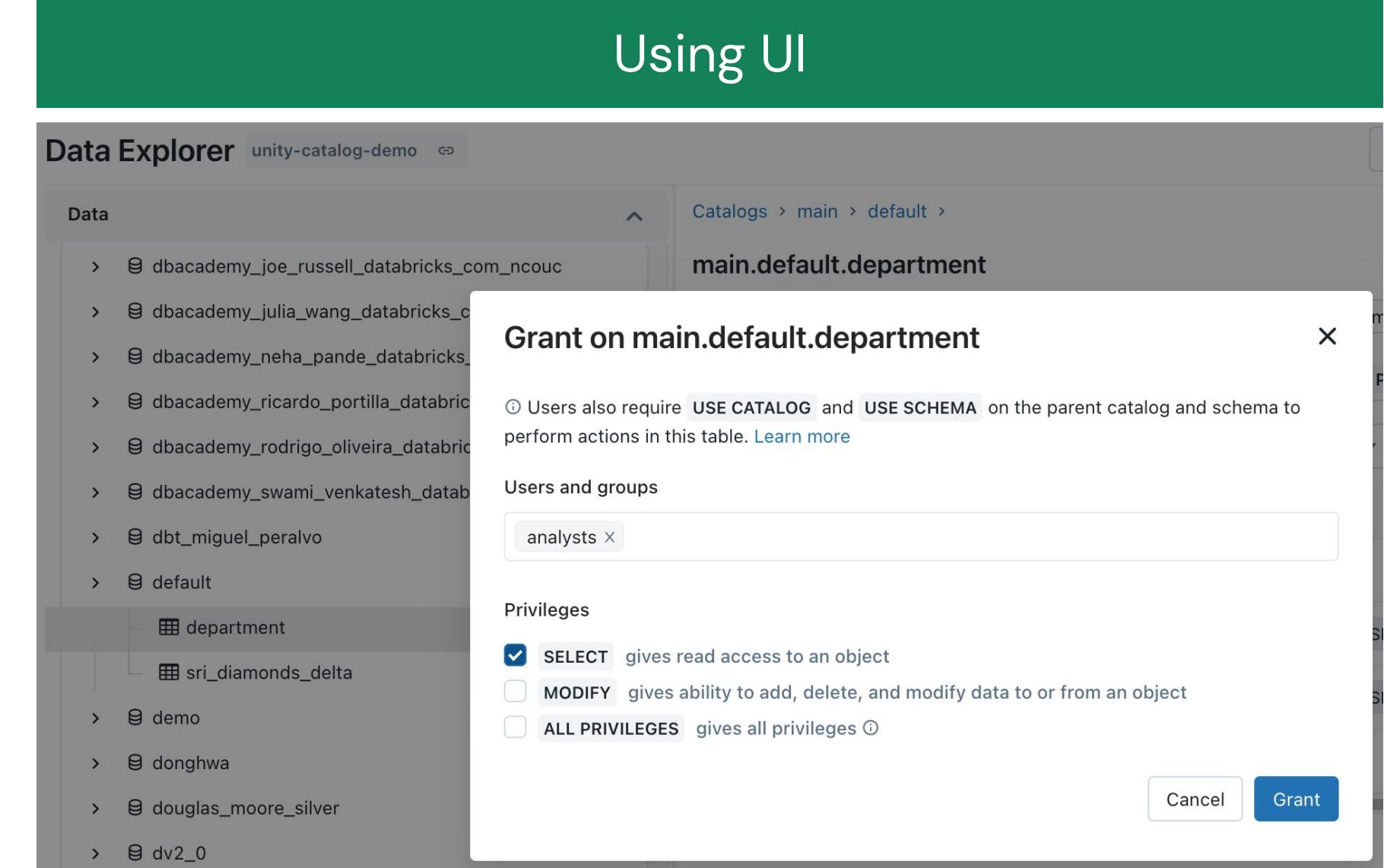
```
GRANT SELECT ON iot.events TO engineers
```

Choose
permission level

'Table'= collection of
files in S3/ADLS

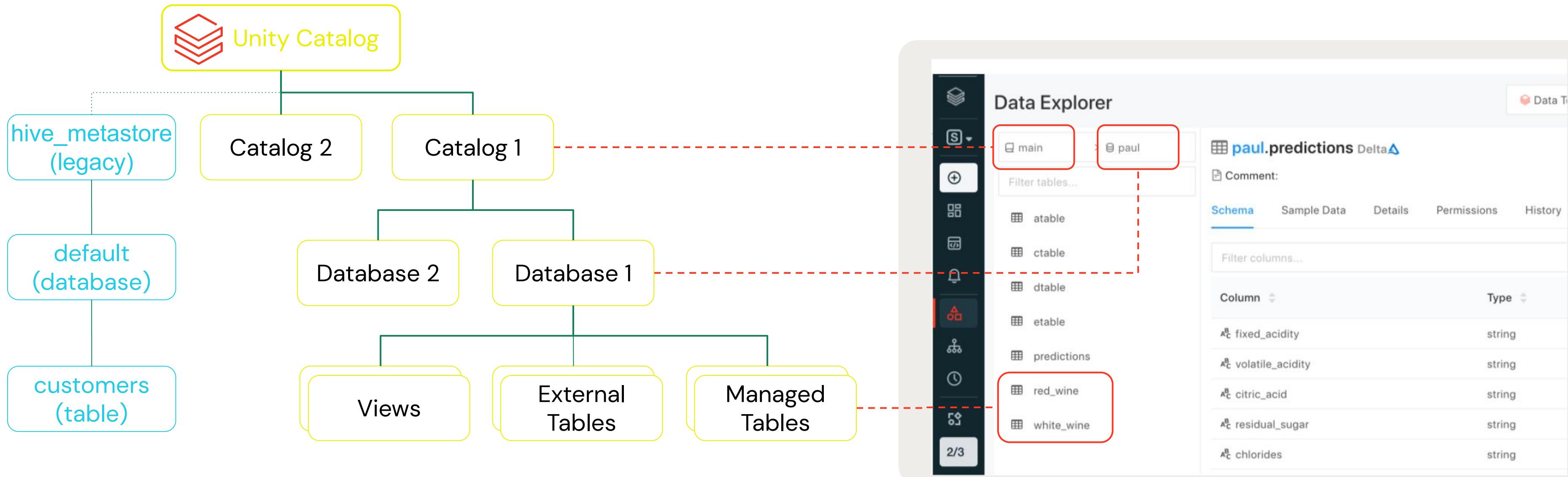
Sync groups from
your identity
provider

Using UI



Three level namespace

Seamless access to your existing metastores

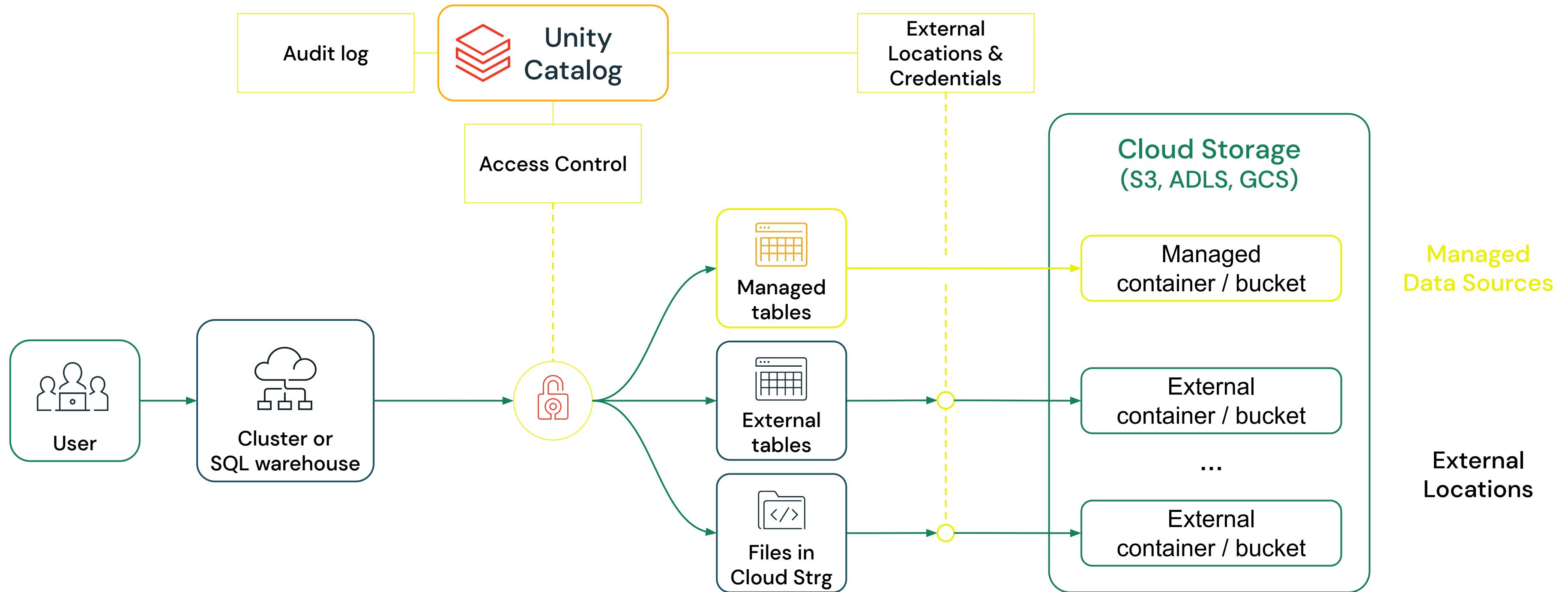


```
SELECT * FROM main.student.example; -- <catalog>.<database>.<table>
```

```
SELECT * FROM hive_metastore.default.customers;
```

Managed Data Sources & External Locations

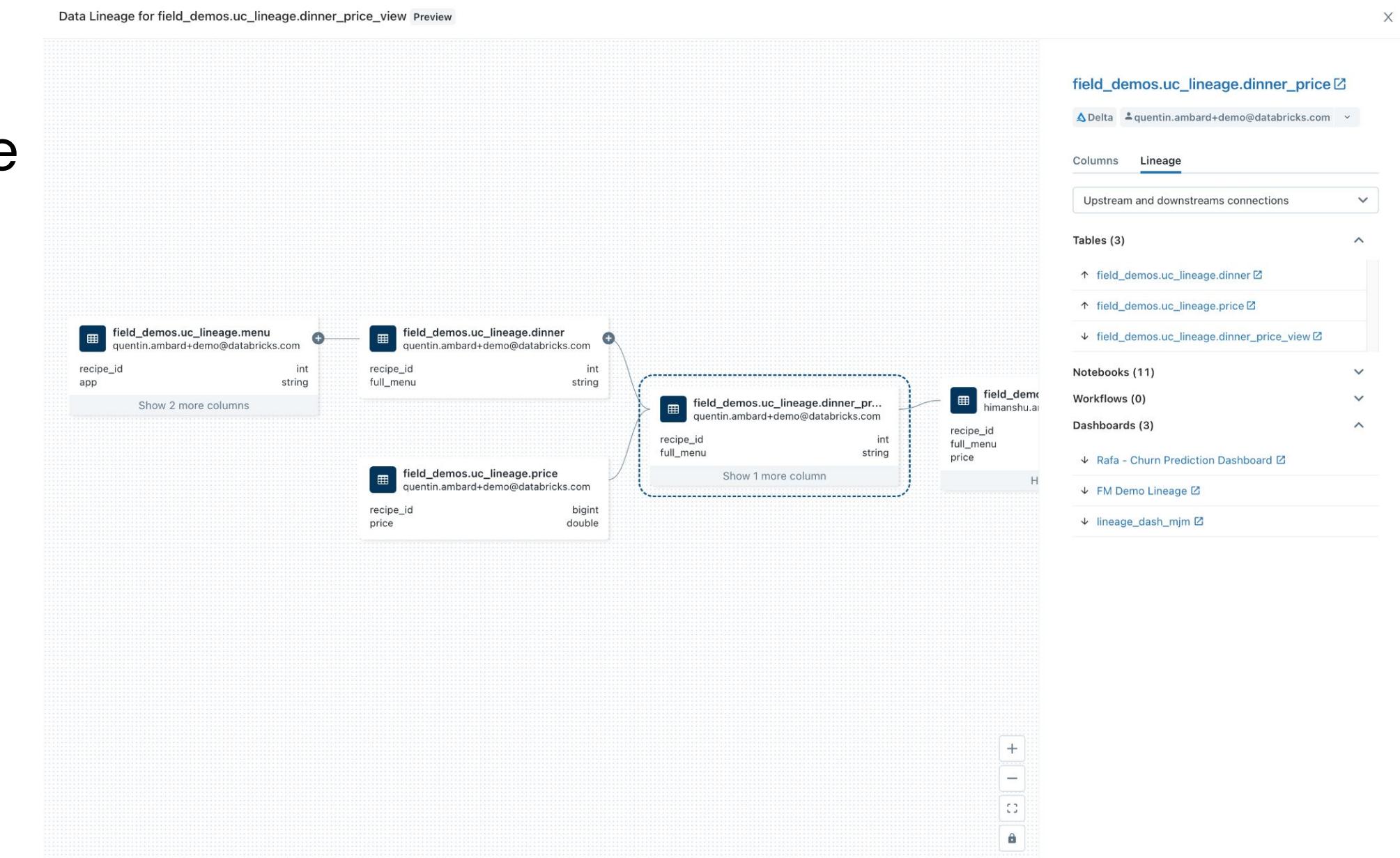
Simplify data access management across clouds



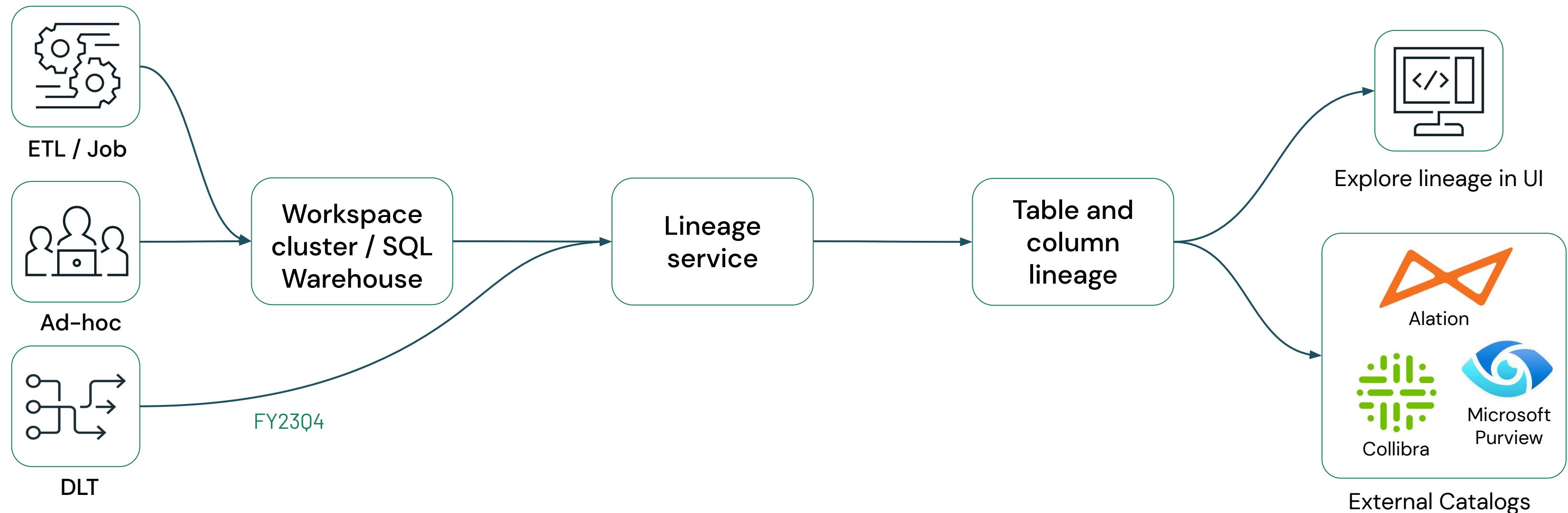
Automated lineage for all workloads

End-to-end visibility into how data flows and consumed in your organization

- Auto-capture runtime data lineage on a Databricks cluster or SQL warehouse
- Track lineage down to the table and column level
- Leverage common permission model from Unity Catalog
- Lineage across tables, dashboards, workflows, notebooks



Lineage flow - How it works

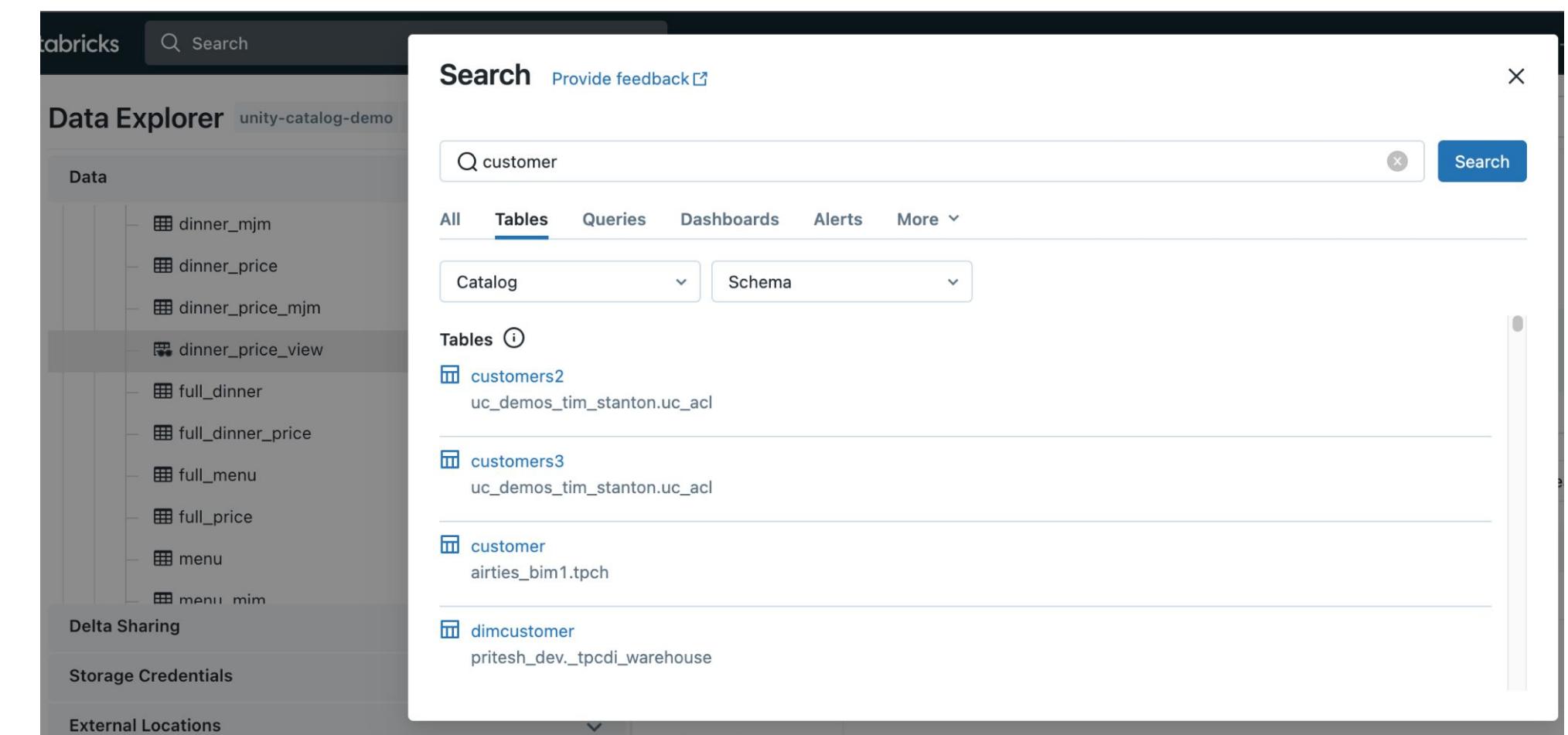


- Code (any language) is submitted to a cluster or SQL warehouse or DLT* executes data flow
- Lineage service analyzes logs emitted from the cluster, and pulls metadata from DLT
- Assembles column and table level lineage
- Presented to the end user graphically in Databricks
- Lineage can be exported via API and imported into other tool

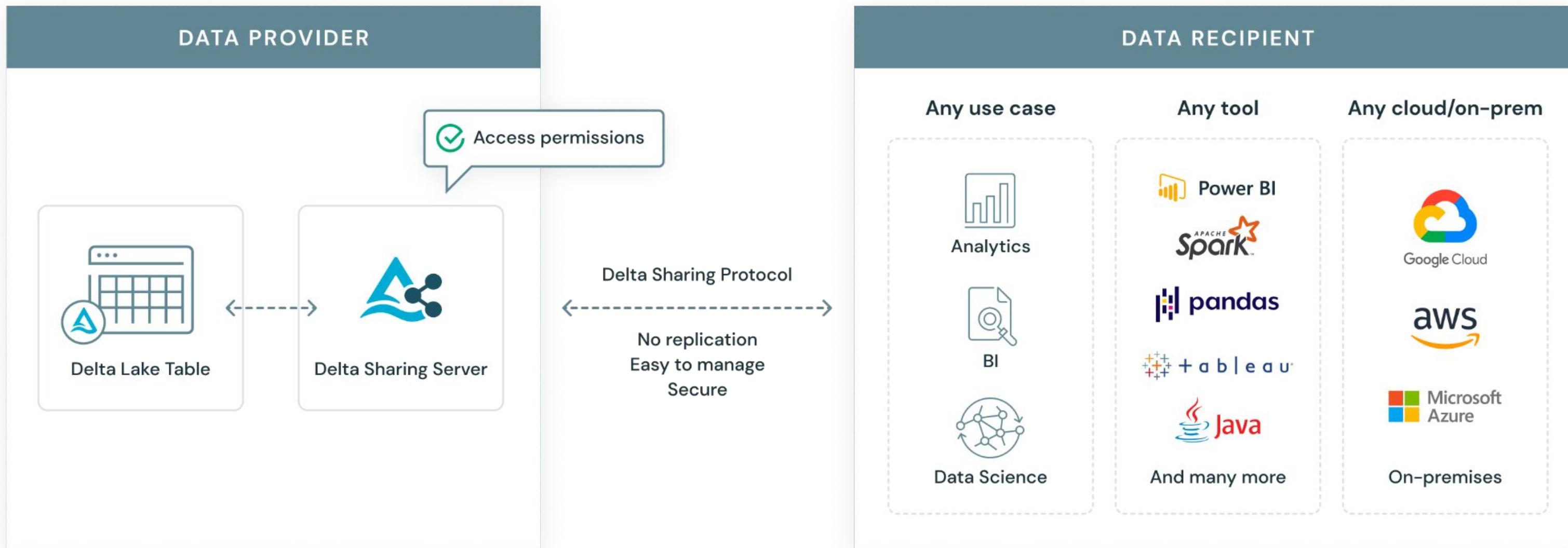
Built-in search and discovery

Accelerate time to value with low latency data discovery

- UI to search for data assets stored in Unity Catalog
- Unified UI across DSML + DBSQL
- Leverage common permission model from Unity Catalog



An open standard for secure sharing of data assets



Unity Catalog -Architecture

