**Databricks with Unity Catalog**:

**1. Catalog-Schema-Table ACL Hierarchy**

**Overview**

Databricks Unity Catalog introduces a fine-grained **Access Control List (ACL)** model to manage permissions across three levels of hierarchy:

* **Catalog**: A top-level container for schemas (databases).
* **Schema**: Holds tables, views, and functions.
* **Table/View**: Data access and manipulation layer.

**Access Types**

| **Level** | **Permissions** | **Description** |
| --- | --- | --- |
| Catalog | USE CATALOG, CREATE SCHEMA, MODIFY | Controls who can use or create schemas. |
| Schema | USE SCHEMA, CREATE, MODIFY, USAGE | Controls who can create or access tables/views. |
| Table | SELECT, INSERT, UPDATE, DELETE | Grants fine-grained data access to users or groups. |

**Best Practices**

* Grant broader access at the catalog level and restrict more tightly at table level.
* Use groups rather than individuals for easier permission management.
* Audit access via Unity Catalog audit logs.

**2. Dynamic Row Filters**

**Purpose**

Dynamic Row-Level Security (RLS) restricts data visibility based on user attributes or roles at query runtime.

**Use Cases**

* Multi-tenant architecture (e.g., each user only sees their company data).
* Geography-based filtering (e.g., user only sees rows from their country).

**Example**

sql

CREATE OR REPLACE ROW FILTER row\_filter AS (region = current\_user\_region());

**Implementation Notes**

* Supported in Unity Catalog.
* Works well with audit logs and external identity providers.
* Filters apply transparently to users.

**3. Column-Mask UDF**

**What is it?**

A custom UDF (User Defined Function) that modifies or hides sensitive column values based on logic defined per user/group.

**Common Techniques**

* **Masking**: Replace values with XXXX-XX-1234
* **Nullification**: Return NULL if user lacks permission
* **Hashing**: Return a hash for unauthorized users

**Example**

sql

CREATE FUNCTION mask\_email(email STRING) RETURNS STRING

RETURN IF(current\_user() IN ('admin'), email, 'hidden@example.com');

**Applications**

* Protecting PII or financial data.
* Showing partial values (e.g., last 4 digits of SSN).

**4. Tagging PII & GDPR Flags**

**Purpose**

Tagging enables data discovery and governance by associating metadata with sensitive fields.

**Types of Tags**

* PII (Personal Identifiable Information)
* GDPR compliance
* Internal classifications (e.g., confidential, restricted)

**Syntax Example**

sql

ALTER TABLE customers ALTER COLUMN email SET TAGS ('pii': 'true', 'gdpr': 'yes');

**Benefits**

* Facilitates automatic policy enforcement.
* Enables lineage and classification tools to scan and report on sensitive data.
* Supports compliance and data minimization strategies.

**5. Lineage Explorer**

**Definition**

A built-in visual interface in Unity Catalog to understand the **data flow** from source to consumer.

**Capabilities**

* Shows how data transforms across tables, jobs, and notebooks.
* Offers **column-level lineage**.
* Tracks all changes and dependencies over time.

**Use Cases**

* Impact analysis before schema changes.
* Debugging and compliance audits.
* Understanding complex data pipelines.

**Best Practices**

* Regularly monitor lineage for high-value assets.
* Combine with tags to trace sensitive data paths.

**6. Audit Logs → Log Analytics Integration**

**Purpose**

Enable detailed tracking of **who did what, when, and where** across data assets.

**Captured Events**

* Read/write access
* Schema changes
* Permission grants/revokes
* Job executions

**Log Destinations**

* **Azure Log Analytics**
* **AWS CloudWatch**
* **Splunk**, **SIEMs**, etc.

**Alerting Example**

* Slack/email alerts on unauthorized access attempts.
* Set thresholds for bulk data reads.

**Log Retention Strategy**

* Retain logs for 1–7 years based on compliance needs.
* Store in secure, immutable storage (e.g., Azure Storage with WORM).

**Conclusion & Next Steps**

| **Topic** | **Description** |
| --- | --- |
| ACL Hierarchy | Base-level permissions framework |
| Row Filters | Secure dynamic record-level access |
| Column-Mask UDFs | Mask sensitive data conditionally |
| Tagging | Metadata-based governance control |
| Lineage Explorer | Visual data flow and dependency map |
| Audit Logs | Continuous security, compliance tracking |