Overview of the Three-Level Namespace in Unity Catalog

Databricks **Unity Catalog** organizes data objects into a three-level namespace that enables fine-grained access control and management of data objects. The three levels are:

1. **Catalog**: The top-level container, which contains schemas (databases) and tables. Each catalog represents an organizational boundary and is often used to segregate data at the business level.
2. **Schema**: The second level, which contains tables and views. Schemas (also called databases) group logically related tables and views.
3. **Table/View**: The lowest level of the hierarchy, representing the actual data stored in a table or a view that refers to data.

Example:

```
CatalogName.SchemaName.TableName
```

Unity Catalog Structure Example

Catalog: `financial_data`

- **Schema**: `transactions`
  - **Table**: `credit_card_payments`
  - **Table**: `bank_transfers`
- **Schema**: `accounts`
  - **Table**: `customer_accounts`
  - **View**: `active_customers_view`

Complete Example:

```
financial_data.transactions.credit_card_payments
```

Creating Unity Catalog Objects in Databricks

Step 1: Create a Unity Catalog Metastore

Before creating any objects, you need to create a metastore that holds catalogs and schemas.

```
# Step 1: Create a new Metastore using SQL
CREATE METASTORE my_metastore;
```

1. **Create Metastore via Databricks CLI or Admin Console**:
   - In Databricks Admin Console, go to the **Metastore** tab.
   - Create a new metastore and assign it to your workspace.

Step 2: Create a Catalog

Once the metastore is set up, you can create catalogs.

```
-- SQL to create a Catalog
CREATE CATALOG financial_data;
```

Step 3: Create a Schema (within a Catalog)

After creating a catalog, you can create schemas to organize your tables and views.

```
-- SQL to create a Schema
CREATE SCHEMA financial_data.transactions;
CREATE SCHEMA financial_data.accounts;
```

Step 4: Create Tables in Unity Catalog

Create tables within the schema.

```
-- SQL to create Tables in a Schema
CREATE TABLE financial_data.transactions.credit_card_payments (
  PaymentID INT,
  Amount DECIMAL(10,2),
  PaymentDate DATE
);

CREATE TABLE financial_data.transactions.bank_transfers (
  TransferID INT,
  Amount DECIMAL(10,2),
  TransferDate DATE
);
```

Step 5: Create Views in Unity Catalog

You can also create views to display subsets of data in tables.

```
-- SQL to create a View
CREATE VIEW financial_data.accounts.active_customers_view AS
SELECT CustomerID, AccountID, Status
FROM financial_data.accounts.customer_accounts
WHERE Status = 'Active';
```

Key Concepts

1. **Catalog**: Represents the highest level in the namespace and contains multiple schemas. It segregates data for different organizational units.
2. **Schema**: Represents the second level of the namespace, containing multiple tables and views. Schemas group tables and views that belong to the same logical domain.
3. **Table**: The third level in the namespace, representing the actual dataset where data resides. Tables can be partitioned, and data is stored in Delta Lake format, Parquet, or other supported formats.
4. **View**: A virtual table generated from a SQL query. It does not physically store data but provides a way to query and organize subsets of data from tables.
