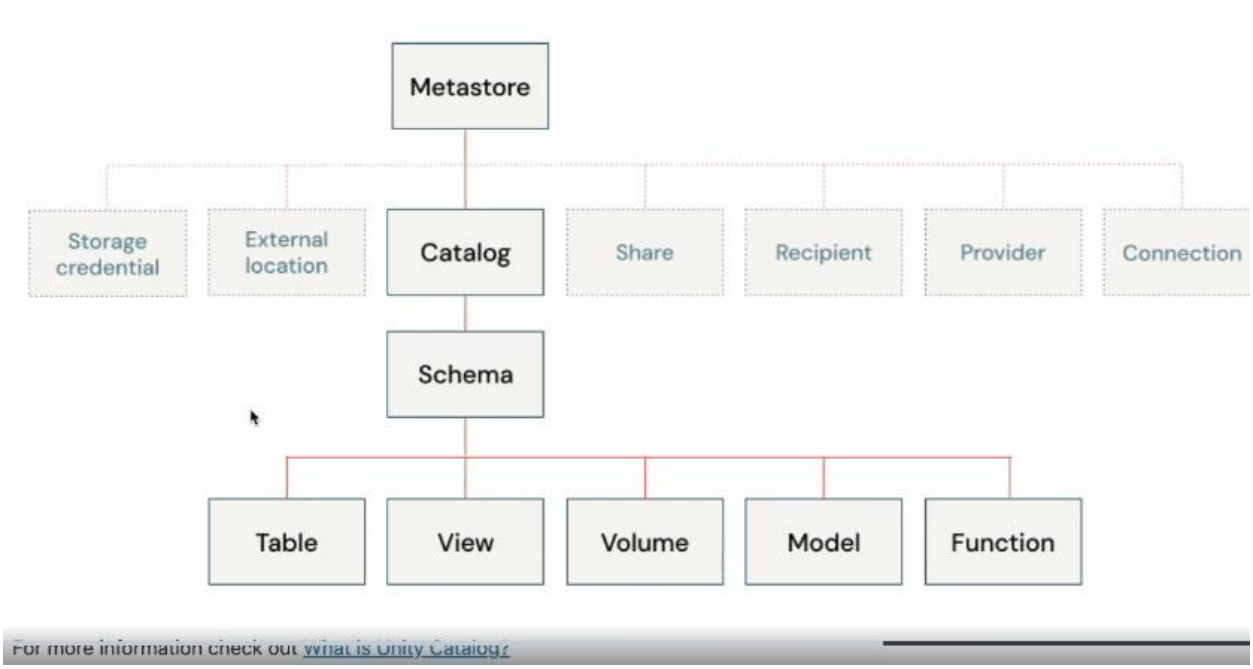
## Creating and working with delta lake table




**USE CTALOGE ${module_cataloge};**

**USE SCHEMA IDENTIFIER{ :my_schema};**

- View your set cataloge schema

SELECT

Current_cataloge() as current_cataloge;

Current_schema() as current_schema;

Use DESCRIBE SCHEMA EXTENDED statement to display the metadata and properties of the schema.

**DECRIBE SCHEMA EXTENDED IDENTIFIER( :my_schema);**

Use show tables statement to display the available tables in the schema.

**SHOW TABLES;**

Use show volumes statement to list all of the volumes available in schema. Volumes are unity catalog objects representing a logical volume of storage in a cloud objects storage location. Volume provide capabilities for accessing, storing, governing and organizing files. While tables provide governance over tabular datasets, volume add governance over non-tabular datasets. You can use volumes to store and access files in any format, including structured, semi and unstructured data

**SHOW VOLUMES;**

**LIST '/volume/${module_cataloge}/${my_schema}/myfiles';**

Create a delta table from csv file. all the tables on databricks are delta lake by default.

**SELECT * from csv.' / volume/${module_cataloge}/${my_schema}/myfiles /employee.csv');**

It shows the data and the columns are in the row as well.

**SELECT * from read_files(' / volume/${module_cataloge}/${my_schema}/myfiles /employee.csv',**

**Format => 'csv',**

**Header => true,**

**Infer schema => true);**

There is a rescue column to provide by default to rescue data that doesn't match any schema.

%SQL

- Drop the table if it is already exists for demonstarion purpose

  **DROP TABLE IF  EXISTES current employees:**

- Create a delta table using csv file select from file regarding the created table(CTAS)

  **CREATE TABLE current_employees USING DELTA**

  **AS**

  **SELECTED ID,FIRSTNAME,COUNTRY,ROLE**

  **FROM read_files( ' / volume/${module_cataloge}/${my_schema}/myfiles /employee.csv',)**

  **Format => 'csv',**

  **Header => true,**

  **Inferschema => true);**

  Infer schema helps to data to infer the schema and type of column

  **DESCIBE DETAILS current_employee;**

  Additional information about delta table.

  The version columns display the table is on version 0.

  The timestamp indicates when table is created.

  The operation shows what operation was performed.

  **DESCIBE EXTENDED current_employee;**

  Display the results of metadata of the table.

## The insert, update, delete transaction on DelatLake:

**SELECT * from current_employee;**

- Insert 2 employees into the table

**INSERT INTO current_employee VALUES (ID,'NAME','COUNTRY','ROLE');**

- Update a record in the table

**UPDATE current_empolyee**

**SET ROLE = 'JUNIOR ENGINEER'**

**WHERE ID = 1111;**

- Delete a record in the table

**DELETE FROM current_employee**

**WHERE ID = 3333;**

Each operation that modifies a Delta Lake table creates a new table version.View the history of the table. The table has 4 versions 0 through 4:

Version 0: the orginal table that was created.

Version 1: contains the write opertation that inserted 2 new employees.

Version 2:contains update

Version 3: contains delete operation

Version 4: contains the optimize operation on the table. (it has done by default)

**Use Time Travel to read previous versions of the delta tables**

you can use the history information to audit operations, rollback a table or query a table at aspecific point in the time using time travel.

By default if you selelct * from current_empolyees orderby ID it show the last version.

Use time travel to use the view the table to the DELETE operation. Notice that the table shows 6 rows before any records deleted.

Time travel takes advantage of the power of Delta Lake transaction log to access data that is no longer in the table.

## Drop Delta table

**DROP TABLE IF EXISTES current_employee;**

- Recourse: data bricks academy