Delta lake with Uniform

Every uniform Apache Hudi, Delta Lake or Apache Iceberg has json for meta data and parquet for data

Parquet files remain the same but meta data automatically generated to make Delta accessible as Iceberg or Hudi readers.

Data Ingestion

How to bring raw data to our Delta Lake. There are different Techniques:

Set the default cataloge and schema, and be sure if tables are existed.

USE CATALOGE ${module_cataloge};

USE SCHEMA IDENTIFIER(:my_schema);

SHOW TABLES;

LIST 'path ';

Remember that the source may be a file and it is possible to read a file and then create a table from it and store it in our storage, it can help also for versioning (time table) since you save the table in create command!

- **CTAS**: create table by selecting data from an existing table or data source. (DATA AND SCHEMA)

  Drop the table if already existes
  DROP TABLE IF EXISTS current_employee_ctas;

  Create the table using CTAS
  CREATE TABLE mydeltatable AS SELECT
  AS
  SELECTED ID, Firstname, Country, Role
  FROM read_file('path' , format =>'csv' , header => true, inferSchema => true);

  Display available tables in your schema
  SHOW TABLES;
  REMEMBER THAT THERE IS AN ALTERNATE METHOD FOR CTAS AND IT IS CREATING THE VIEW

  Drop the table if exists
  DROP TABLE IF EXISTS current_employees_ctas;

  Create temporary view
  CRAETE OR REPLACE TEMP VIEW vw_currrent_employees
  USING CSV
  OPTIONS ('path', header= 'true', delimiter = ',');

Use temporary view in CTAS statement or create the table
CRAETE OR REPLACE TABLE current_employees_ctas AS
SELECT *
FROM vw_cureent _employees;

- **UPLOAD UI**: provides a point and click interface to upload files and create tables and it allows to upload csv, tsv, parquet, avro … it enables you to upload the file directly to a Unity Catalog volumes.
  a. Select the catalog icon in the navigation bar.
  b. Type the module's catalg name getstarted in the search bar.
  c. Select the refresh icon to refresh the getstarted catalog.
  d. Expand the getstartted catalog. Within the catalog you should see a variety of schemas
  e. Locate and expand the relevant **schema**. You can find the correct schema in the **setup notes**.
  f. Use the **Upload UI** to select and upload your file. Ensure you choose the correct **file format** and **destination** within the schema.
  g. Once uploaded, verify that the file appears in the Unity Catalog and is available for use in queries or analysis
- **COPY INTO**: load **files** from a file location into a delta table or support various files format and storage locations and handle the schema changes. (you can put location of the files not just one file)
  The important is **Idempotent** that skipped from the already loaded file in the sources for increasing efficiency.
  Create a table from employee.csv file using COPY INTO statement. The copy into statement loads data from a file location onto Delta table. This is a retriable and idempotent operation- file in the source location that have already been loaded are skipped.

  Drop the table if exists
  DROP TABLE IF EXISTS current_employee_copyinto;

  Create an empty table with the **column data type,** so in **COPY INTO** you must know the **columns type.**

  CREATE TABLE current_employee_copyinto (ID INT, FirstName STRING, Country STRING, Role STRING);

  COPY INTO my_catalog.my_schema.my_table
  FROM 'path'
  FILEFORMAT= CSV
  FORMAT_OPTIONS ('header' = 'true' , 'inferSchema' = 'true');

  COPY INTO mydeltatable
  FROM 'the path'
  FILE_FORMAT = ' format'
  FILE_OPTION = ('format_option)
  And then select shows the table

Key Features of COPY INTO

☑ **Incremental Loads** – Avoids duplicate data by skipping previously ingested files.
☑ **Schema Evolution** – Can automatically merge new columns.
☑ **Efficient** – Faster than INSERT INTO for large-scale ingestion.

Schema Evolution in Databricks

Schema evolution is a feature in **Delta Lake** that allows you to **automatically update the schema** of a table when new columns or changes are detected in the incoming data. This is especially useful when using COPY INTO, MERGE, or INSERT INTO to load data from external sources.

Why is Schema Evolution Important?

In real-world scenarios, data sources often evolve, meaning:

- New columns may be added.
- Existing column data types may change.
- The order of columns may shift.

Without schema evolution, these changes could cause **errors or require manual intervention** to update the schema. **Delta Lake automates this process**, making it easier to manage evolving datasets.

How to Enable Schema Evolution?

COPY INTO my_catalog.my_schema.my_table

FROM 's3://my-bucket/data/'

FILEFORMAT = CSV

COPY_OPTIONS ('mergeSchema' = 'true');

mergeSchema = 'true' allows new columns in the source data to be **automatically added** to the table.

DESCRIBE HISTORY current_employee_copyinto;

As a instance the delta table versioning or history, imagine there are 3 versions:

Version 0 is the initial empty table created by the CREATE TABLE statement.

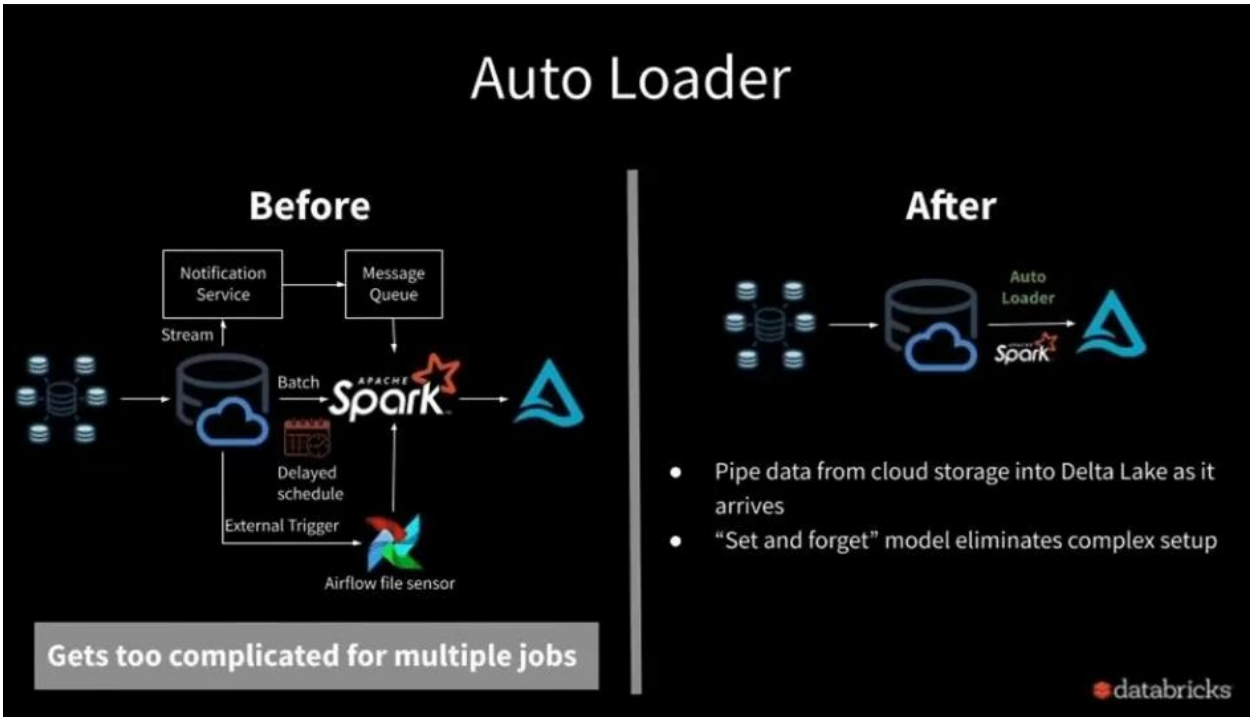
Version 1 is the first COPY INTO statement that loaded into the delta table.

Version 2 is the second COPY INTO statement that only loaded the new rows into the delta table.

When Should You Use Schema Evolution?

- **When data sources change frequently** (e.g., logs, IoT data).
- **When loading semi-structured data** (JSON, CSV with evolving fields).
- **When running ETL pipelines** that must handle schema drift automatically.

- **AUTO LOADER**: incrementally or streaming processes new data files as they arrive in cloud storage. **Automatically infers schema and accommodates schema changes,** and includes a **rescue data** column for data that does not adhere to the schema.



Clean up
Drop the tables
DROP TABLE IF EXISTS current_employee;
% Python
# get the widget values into python varaibles
Cataloge_name = dbuilts.widgets.get('module_catalog')
My_schema = dbuilts.widgets.get('my_schema')

remove employee.csv
Dbuilts.fs.rm(f"/volumes/{catalog_name}/{my_schema}/myfiles/employee.csv")