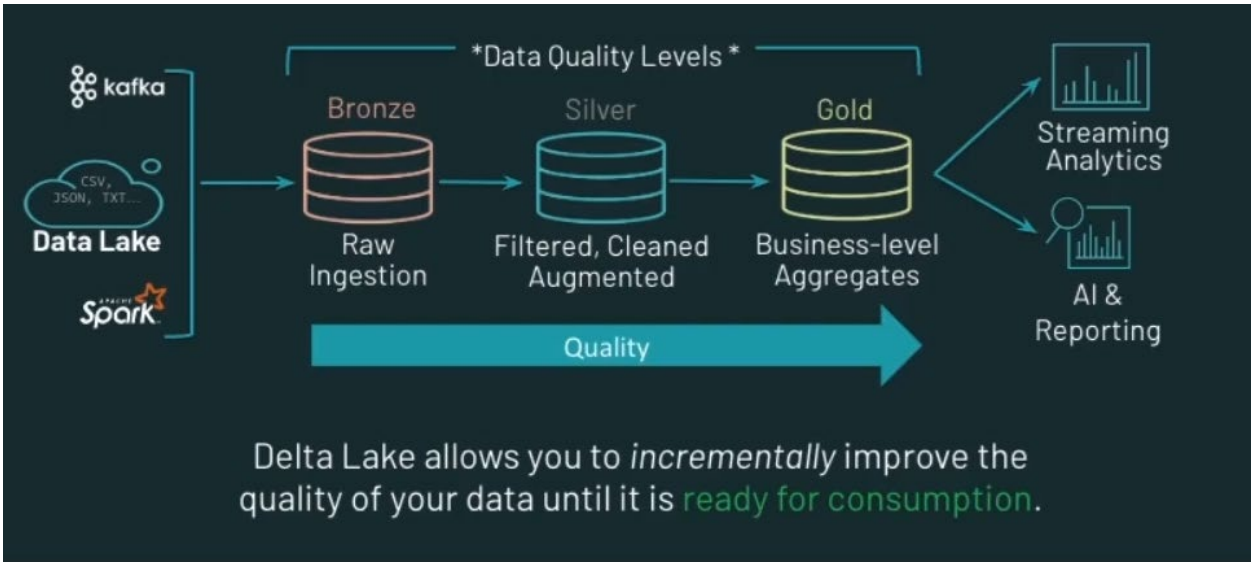
Medallion architecture for data transformation in Delta Lake



As you ingest the data from the sources into the delta lake through batch and streaming method, you can begin processing and transforming data in data bricks. For almost cases we use from Medallion architecture for data transformation in Delta Lake. A data design pattern used to organize data within a lake house. This is built on the bronze, silver and gold medallion data quality levels. Databricks didn't invent this pattern but it is very fit for delta lake. The goal is incrementally and progressively improve the structure and quality of data as it moves each layer, from the original raw data in data lake in its native format, to the bronze layer, to the silver layer, and finally the gold layer consumer level data.

In the bronze layer we simply ingest the raw data from external sources, we get data as quickly as possible even though it may be dirty. And we keep the data, making it useful for both current and future projects, adding any additional metadata columns that capture the load date, time, process ID and more. By ingesting the raw data, we avoid exposing ourselves to bugs in the systems or processing logic.

We have the data as it originally existed, and we can always go back to it. This is feasible because the storage is cheap. Depending on the regulatory requirements, you can remove any personally identifiable information, or PII, as you ingest data into your Delta Lake.

In the silver layer of the lake house, data is filtered, cleaned, joined and enriched version of the bronze data, the tables in the silver layer to find the structure of the data and enforce the schema or evolve the schema as necessary. The silver layer becomes a single source of truth for the enterprise across many projects. We fix errors, add in business data and rules. Examples of this layer tables would be clean tables like all unique customers, transactions and more.

Gold level tables contain clean data, ready for consumers consumption. Depending on their use case, they may include business level aggregate of the silver data. Gold data is formatted for specific projects or reports, delivered downstream to user and applications. These data can be stored in delta format, for use by spark jobs or SQL queries. Many engines have delta lake readers and with uniform, delta tables can also be read natively by other applications.

Thanks to the solid foundation of Delta's ACID support, data bricks can offer more advanced features in the runtime itself. Databricks is able to provide support for delete, update and merges in the cloud data

lake. These features are groundbreaking for data lakes and delta lake is easy to use. It supports sql, most well known API for data, python, scala ,with ACID compliance and sql support, it becomes much easier to mange data pipeline in the change., when the business logic changes you can simply update or delete the silver or gold tables and restart the stream or batch processes in order to put the changes into effect. Typically, a data engineering pipeline is a quite complex it involves stages, starting with reading data from various sources, including streaming or batch files and data lakes. Data from different sources ingest to bronze table in raw form. And bronze under multiple processes transform to silver which are cleaned, joined and enriched data quality. And finally, the cleaned and processes silver tables are aggregated into gold tables and these gold tables are used for various purposes including BI and reporting, machine learning AI and streaming takes to support business needs.

USE CATALOG ${module_catalog};

USE SCHEMA IDENTIFIER(:my_schema);

SHOW TABLES;

LIST '/volume/catalog/schema/file/'

The simple example of medallion architecture

Objective: create pipeline that can be scheduled to run automatically the pipeline will:

a. Ingest all csv files from the my_files volume and create a bronze table.
b. prepare the bronze table by adding new columns and create silver table.
c. Create a gold aggregated table for consumers

1- Bronze

--drop the table if exists

DROP TABLE IF EXISTS current_employee_bronze;

--create an empty table and columns

CREATE TABLE IF NOT EXISTS current_employee_bronze(

ID INT, FirstName STRING, Country STRING, Role STRING, InputFile STRING);

--create the bronze raw ingestion table and include the csv file name for rows(the csv file is small and we use copy into technique to ingest with the infer schema)

COPY INTO current_employee_bronz

--input metadata column(created a column wich shows where each row come from. This extra meta data helps to debug anything)

FROM

(SELECT * _metadata.file_name AS InputFile FROM 'path')

FILEFORMAT = csv

FORMAT_OPTIONS (‘header’ = ‘true’ , inferSchema’ = ‘true’);

--view the bronze table

SELECT *

FROM current_employee_bronze;

2- Silver

In the silver layer we want to transform the bronze table and insert the resulting rows into the silver table. Create and display a temporary view which select the columns ID, First Name, Country and convert Role to uppercase and add a new columns Timestamp and date so we have 6 columns.

--create a temp view to use to merge into the final silver table

CRAETE OR REPLCASE TEMP VIEW temp_view_silver AS

SELECT

ID,

FirstName,

Country,

Upper(Role) as Role,

Current_timestamp() as TimeStamp,

Data(timestamp) as Date

FROM current_employee_bronz;

--display the results of view

SELECT * FROM temp_view_silver;

It is just a view and Now we want to have delta table. So lets to create the delta table.

We have a current_employee_silver and the view of it.

--drop the table

DROM TABLE IF EXISTS current_employee_silver;

--create an **empty** table and specify the column datatype

CREATE TABLE IF NOT EXISTS a current_employee_silver (

ID INT,

FirstNmae STRING,

Country STRING,

Role STRING,

Timestamp TIMESTAMP,

Date DATE);

--insert records from the view when not matched with the target silver table

MERGE INTO a current_employee_silver AS TARGET

USING tem_view_employee_silver AS SOURCE

ON target.ID = source.ID

WHEN NOT MATCHED THEN INSERT *;

3- Gold

Objective: aggregate the silver table to create the final gold table.

Create a temporary view named temp_view_total_roles that aggregate the total numv=ber of the employees by **role** then display the results of the view.

CREATE OR REPLACE TEMP VIEW temp_view_total_roles AS

SELECT

Role,

Count(*) as TotalEmployee

FROM current_employee_silver

GROUP BY Role;

SELECT *

From temp_view_total_roles;

Now we create final gold table named total_roles_gold with the specified columns:

--dropping the tables

DROP TABLE IF EXISTS total_roles_gold;

CREATE TABLE IF NOT EXISTS total_roles_gold (

Role STRING,

TotalEmployee INT);

Then inserting all rows from the aggregated temporary view into the total_roles_gold table, overwriting the existing data in the table. This will allow to see the history as the gold table is updated if there is any new employee added.

INSERT OVERWRITE TABLE total_roles_gold

SELECT *

FROM temp_view_total_roles;

query that tables we use

SELECT *

FROM total_roles_gold;

Data governance and security

Objective: now we have our valuable asset (gold data). view the lineage of **gold** table and learn how to set its **permission**.

Complete the following to open your schema in the catalog explore

a. Select the catalog icon in the left navigation bar
b. Select the schema and choose open in catalog explorer
c. We have different tables in our schema(databases)
d. Select the specific one

We have different details of overview, sample data, details, permission, history, lineage, insights and quality.

Fir permission we can grant permissions in the environment, select the grant and add individual or group to have it.

In the history regarding the time travel there are versions of the history of the tables for example the original table created and the version for overwrite and update.

In the lineage there is a graph of the comes from the previous tables and it is possible to expand. (From gold to silver and bronze and the original files)

In the insight we can see who use and access the tables.

And in the quality, you can create monitor track changes in data

Cleanup

--drop the temporary view

DROP VIEW IF EXISTS temp_view_total_roles;

DROP VIEW IF EXISTS temp_view_empoloyees_silver;

--drop The tables

DROP TABLE IF EXISTS current_employees_ bronze;

DROP TABLE IF EXISTS current_employees_ silver;

DROP TABLE IF EXISTS total_roles_ gold;