05 Medallion Architecture

> https://medium.com/@junshan0/medallion-architecture-what-why-and-how-ce07421ef06f

ETL and ELT

The three characters in these two terms (ETL or ELT) have the same meanings, but the orders of operations are different.

- In ETL, data is transformed into a predefined format, then gets loaded into target storage, while in
- ELT, data is loaded into the destination, and gets transformed only when requested.
- In ETL's case, you will need to know what the target state looks like before you can design and implement the transformations, while in ELT, you can load the data regardless of its format, and transformation happens when requested. Thus, the ELT process is often preferred in current data engineering architectures, especially with the rise of data lake systems, which store large volumes of data in raw formats.

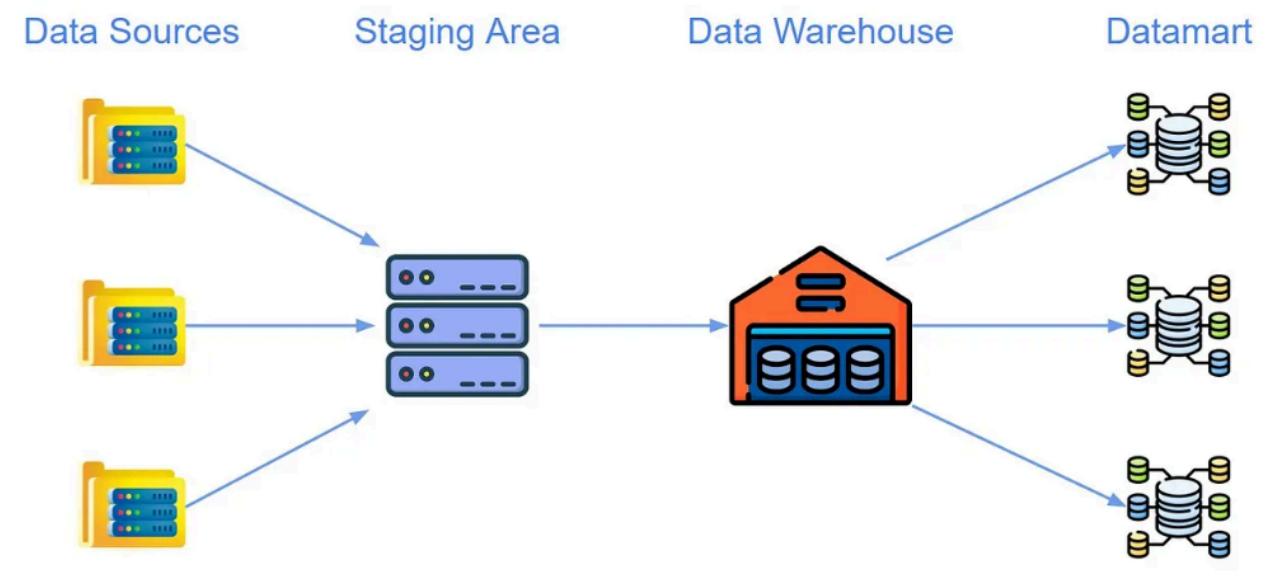
The Challenge of Organizing Data in a Data Lake

However, the introduction of data lakes and ELT processes also introduced a problem: How should you organize the data in data lakes, and design ELT processes accordingly? Traditional data warehouse usually consists of the following three layers:

1. Staging Area: This is an intermediate storage area where data is temporarily held during the ETL process. Staging areas allow for data validation, transformation, and error handling before it is loaded into the data warehouse.
2. Data Warehouse: The central repository for storing structured and organized data. It typically consists of a relational database optimized for analytical queries and reporting. Data in data warehouses is organized by business subject areas.

Sometimes, there is an additional layer before data warehouses, where you have all the data organized by sources, likely the transaction systems that generate the data. This layer is usually used for storage of data that must be retained for a short time. It can also be used for operational reporting, and is often called Operational Data Store (ODS).

3. Data Mart: A subset of the data warehouse that is focused on a specific business function or department. Data marts are often designed for easier access and analysis by specific user groups. The data marts are usually in star schema.



Medallion Architecture Lecture Notes

Introduction

Definition: Medallion architecture is a data design pattern for organizing data in a lakehouse environment. It logically structures data into progressive layers (Bronze, Silver, Gold) to incrementally improve data quality, structure, and usability as it flows through the system. This ensures atomicity, consistency, isolation, and durability (ACID) properties while optimizing for analytics, machine learning, and reporting.

- Also known as "multi-hop architecture" or "medallion lakehouse architecture."
- Goal: Prevent data lakes from becoming "data swamps" by defining clear boundaries for data ingestion, transformation, and consumption.

History and Adoption:

- Pioneered by Databricks as a best practice for lakehouse architectures.
- Adopted by Microsoft as the standard for Fabric's OneLake (central data storage in Microsoft Fabric, released Nov 2023).
- Widely used in modern data platforms for scalability, flexibility, and governance.

Key Principles:

- Incremental enhancement: Data starts raw and is refined progressively.
- Flexibility: Supports various data models (e.g., relational, dimensional, Data Vault, NoSQL) and ecosystems (data lakes, warehouses, lakehouses).
- ELT Focus: Load raw data first, transform as needed, contrasting with traditional ETL.

Background Concepts

Data Lake: Centralized repository for storing raw, structured, semi-structured, or unstructured data at scale (e.g., JSON, images, logs). No upfront structuring required; supports diverse tools for analysis and ML.

Data Lakehouse: Hybrid of data lake (scalability/flexibility) and data warehouse (consistency/integrity). Stores raw data in lakes and refined data in warehouses for analytics.

ETL vs. ELT:

- **ETL (Extract, Transform, Load)**: Transform data before loading; requires predefined schemas.
- **ELT (Extract, Load, Transform)**: Load raw data first, transform on-demand; ideal for data lakes due to raw storage and scalability.

Traditional Data Warehouse Layers (for comparison):

- Staging Area: Temporary for ETL validation.
- Operational Data Store (ODS): Source-organized for short-term retention/operational reporting.
- Data Warehouse: Subject-area organized, cleansed data.
- Data Mart: Function-specific subsets (e.g., star schema).

Challenges in Organizing Data Lakes

Raw data storage leads to flexibility but creates issues:

- Lack of structure/integrity for reporting.
- Intermediate transformations create "data swamps" (obsolete, half-processed files cluttering storage).
- No clear boundaries: When to save intermediates? How to organize for reuse without redundancy? ELT exacerbates this: Transformations happen on-the-fly, but caching results is inefficient without guidelines.
  Medallion addresses this by defining layered states: Raw → Cleansed/Enriched → Curated/Aggregated.

Medallion Architecture Overview

- **Core Structure**: Three layers (Bronze, Silver, Gold) representing data maturity.
  - Data flows sequentially: External sources → Bronze → Silver → Gold.
  - Each layer adds value (validation, transformation) while preserving lineage and auditability.
- **Multi-Hop Nature**: Data "hops" through layers, allowing reprocessing from any point.
- **Fit in Ecosystems**:
  - Typically in lakehouses: Bronze/Silver in data lakes (e.g., Delta Lake format), Gold in warehouses or marts.
  - Supports multi-cloud: Bronze in various clouds, Silver/Gold centralized.
  - Not a specific data model; can use 3NF, star schema, etc.

Bronze Layer (Raw Zone)

- **Purpose**: Initial landing for all source data in original format; acts as single source of truth.
- **Characteristics** (from Databricks, Fabric, and article):
  - Raw, unvalidated data (structured, semi-structured, unstructured).

    - Append-only, immutable; includes metadata (e.g., load timestamp, process ID, file name for provenance).
    - Supports Change Data Capture (CDC) for historical archiving, auditing, and reprocessing.
    - Minimal processing: No cleanup/validation; store fields as strings/VARIANT/binary to handle schema changes.
    - Sources: Cloud storage (S3, ADLS, GCS), message buses (Kafka), databases, federated systems.
- **Intended Users**: Data engineers, compliance/audit teams.
- **Implementation Tips**:
    - Use Parquet/Delta Lake for storage; shortcuts in Fabric to avoid copying.
    - Ingest via streaming/batch; retain all history.
    - Example: Raw sales logs from CRM systems stored as-is.

Silver Layer (Enriched/Validated Zone)

- **Purpose**: "Just-enough" transformations to create a unified enterprise view; prepares data for further analytics.
- **Characteristics**:
    - Cleansed, validated, enriched data from Bronze.
    - Operations: Data cleansing (remove duplicates, fix typos, standardize formats), verification (quality checks, business rules), conforming (standardize schemas), matching/integration (generate universal keys, joins).
    - Non-aggregated; at least one full record representation.
    - Handles schema evolution, nulls, deduplication, late-arriving data.
    - Starts modeling: Flatten nested data (e.g., JSON to tables); akin to ODS/DWH but with minimal extras.
    - Avoid direct ingestion; read from Bronze via streaming/batch.
- **Intended Users**: Data engineers, analysts (for detailed queries), scientists (for ML models).
- **Implementation Tips**:
    - Use Delta tables for ACID support; enforce schemas.
    - "Just-Enough" Rule: Only essential transforms; reserve complex joins/aggregations for Gold unless reusable.
    - Example: Cleansed customer data from multiple sources, with standardized addresses and a common ID.
- **Comparison to Traditional**: Similar to ODS (source-integrated) or DWH (subject-organized), but schema-agnostic.

Gold Layer (Curated Zone)

- **Purpose**: Business-ready, optimized data for consumption; tailored for reporting, ML, and apps.
- **Characteristics**:
    - Highly refined, aggregated, denormalized views.
    - Applies business logic: Aggregations (sums, averages), filters (time/region), dimensional modeling (star schema).
    - Minimal joins; read-optimized for performance.
    - May have multiple Gold layers (e.g., per department: HR, Finance).

    - Not for large historical data (use Silver for that).
- **Intended Users**: Business analysts, BI developers, executives, ML engineers.
- **Implementation Tips**:
    - Use data warehouses or Delta tables; create materialized views (e.g., weekly sales aggregates).
    - In Fabric: Gold as warehouse for SQL access.
    - Example: Star schema data marts for sales reporting (fact tables with dimensions like product, time).

Implementation Guidelines

- **General**:
    - Storage: Delta Lake (Fabric/Databricks) for reliability, time travel, and optimizations (V-Order, Z-ordering).
    - Partitioning: By date/key for query speed; aim for ~1GB files.
    - Operations: Use MERGE for upserts; VACUUM for historical cleanup (min 7-day retention).
    - Workspaces: Separate lakehouses per layer in Fabric for governance.
- **Databricks-Specific**:
    - Build layers via Unity Catalog; use streaming for append-only.
    - Model semi-structured data in Silver (e.g., VARIANT for JSON).
- **Microsoft Fabric-Specific**:
    - Use OneLake; shortcuts for zero-copy access.
    - Engines: Spark for transforms, SQL/Power BI for queries.
    - Deployment: Bronze/Silver as lakehouses, Gold as warehouse.
- **Best Practices**:
    - Governance: Track lineage, metadata; ensure compliance.
    - Cost: Batch/trigger ingestion; optimize partitions.
    - Scalability: Multi-cloud support; incremental loads.

Benefits

- **Data Quality**: Progressive refinement reduces errors; ACID guarantees.
- **Efficiency**: Reusable intermediates; no data swamps; optimized queries.
- **Flexibility**: Handles diverse data/models; supports ELT in evolving businesses.
- **Governance**: Easy auditing, lineage, compliance; reduces silos/duplication.
- **Analytics/ML Readiness**: Foundation for BI, ML, and advanced use cases.
- **Cost Savings**: Less movement/duplication; scalable storage.

Design Patterns and Variations

- **In Lakehouses**: Bronze/Silver in lakes (raw/flexible), Gold in marts (analytical).
- **Multi-Cloud**: Bronze per cloud/source, centralized Silver/Gold.
- **Extensions**: Add Platinum layer for specialized ML datasets (not standard).
- **Common Pitfalls**: Over-transforming Silver; ignoring schema evolution.

Final Thoughts

- Medallion architecture is ideal for modern, cloud-native data platforms, enabling incremental improvements and robust governance.
- As of 2025, it's a de facto standard in Databricks and Fabric, evolving with AI/ML integrations.
- For hands-on: Explore Databricks notebooks or Fabric labs for building layers.

References

- Medium Article by Jun Shan
- Databricks Official Docs
- Microsoft Fabric Docs