Different ETL (Extract, Transform, Load) Architecture Designs

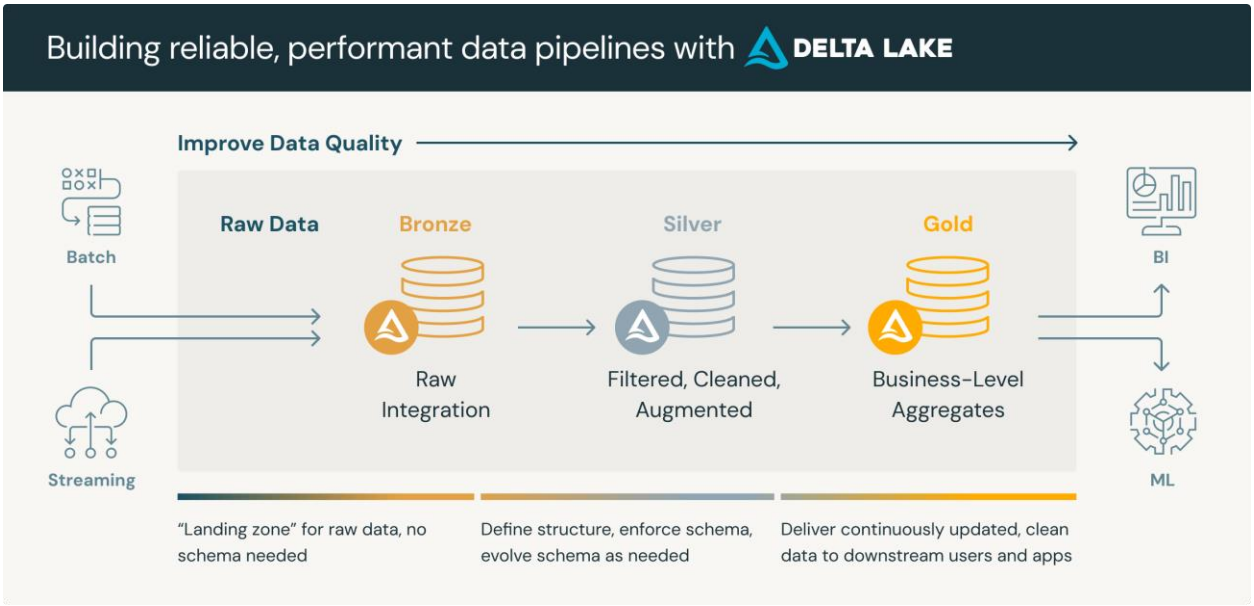
Introduction

Selecting the appropriate architecture for your data requirements is a critical initial step in establishing your project. Various ETL architectures are available to address diverse business needs and data complexities. This article explores common ETL architecture patterns and their best-fit scenarios, including the Medallion Architecture, Lambda Architecture, Kappa Architecture, Data Vault Architecture, Kimball's Dimensional Data Warehouse, Inmon's Corporate Information Factory, Lakehouse Architecture, and various ETL Pattern Variations. Each architecture is designed to tackle specific data management challenges, offering distinct advantages and use cases.

Medallion Architecture

The Medallion Architecture represents a layered approach to data processing that is frequently employed in data lakes. This architecture is designed to address the complexities of managing large volumes of diverse data by organizing it into progressively refined layers. The primary objective is to improve data quality and traceability while providing a clear framework for data governance.

Architecture and Layers



Ref: https://www.databricks.com/glossary/medallion-architecture

Bronze Layer

The Bronze Layer serves as the initial stage of data ingestion. In this layer, raw data is collected from multiple sources, including point-of-sale systems, online transactions, IoT devices, and other data-generating systems. This raw data is stored in its original format within a data lake, which allows for the storage of large volumes of diverse data types without immediate transformation.

Silver Layer

The Silver Layer focuses on cleaning and enriching the raw data ingested in the Bronze Layer. During this phase, data undergoes a series of transformations designed to improve its quality and usability. Key processes include the removal of duplicates, standardization of data formats, correction of errors, and integration of additional contextual information, such as customer demographics or product metadata.

Gold Layer

The Gold Layer represents the final stage in the Medallion Architecture, where data is aggregated and refined for specific business applications. This layer focuses on creating business-specific datasets that are optimized for reporting, dashboarding, and advanced analytics. The data in the Gold Layer is highly curated and often tailored to meet the needs of various business units, such as finance, marketing, and operations.

Possible Scenario

Scenario: A retail company wants to manage and analyze large volumes of transaction data from multiple stores and online sales channels.

Bronze Layer: Raw transaction data from point-of-sale systems and online transactions are ingested into a data lake.

Silver Layer: The raw data is cleaned, removing duplicates and standardizing formats. Enriched with additional data such as customer information and product details.

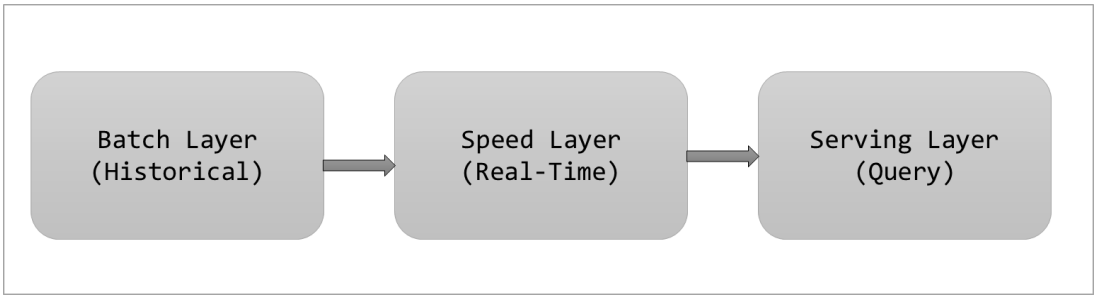
Gold Layer: Aggregated data is prepared for business-specific analysis, such as sales reports, inventory management, and customer behavior analysis.

Lambda Architecture

The Lambda Architecture is a data processing paradigm designed to handle massive quantities of data by leveraging both batch and real-time processing methods. It addresses the need for real-time analytics while simultaneously providing the capability to perform in-depth historical data analysis. This architecture is particularly effective in scenarios where both low-latency updates and comprehensive data processing are required.

Lambda Architecture divides data processing into three main components: the Batch Layer, the Speed Layer, and the Serving Layer. Each component plays a specific role in ensuring that data is processed efficiently and made available for querying in a timely manner.

Architecture and Layers



Ref: https://www.databricks.com/glossary/lambda-architecture

Batch Layer

The Batch Layer is responsible for processing and storing large volumes of historical data. This component performs comprehensive data processing tasks that can tolerate higher latencies, such as computing long-term aggregates, trends, and patterns. The processed data is stored in a batch view, which provides a consolidated view of the historical data.

Speed Layer

The Speed Layer handles real-time data streams, ensuring that recent data is processed with minimal latency. This component complements the Batch Layer by providing immediate insights and updates based on the latest data. The Speed Layer is typically built using stream processing frameworks such as Apache Storm, Flink, or Kafka Streams.

The primary goal of the Speed Layer is to process real-time data quickly and produce incremental updates that can be combined with the batch view in the Serving Layer. This ensures that users have access to both the most recent data and the historical data processed by the Batch Layer.

Serving Layer

The Serving Layer merges the batch view and the real-time view to provide a comprehensive and up-to-date view of the data. This layer supports queries that require access to both historical and real-time data, enabling users to perform complex analytical tasks.

The Serving Layer is designed to optimize query performance by integrating the outputs from the Batch Layer and the Speed Layer. This component ensures that users can retrieve accurate and timely insights from the data, regardless of its origin.

Possible Scenario

Scenario: A financial services company needs to process and analyze stock market data in real-time while also performing batch processing for historical trend analysis. Aim is to provide the traders with real-time market insights while also enabling analysts to study long-term market trends.

Batch Layer: Historical stock data is processed in large batches to compute long-term trends and patterns.

Speed Layer: Real-time stock price data is processed for immediate alerts and decisions.

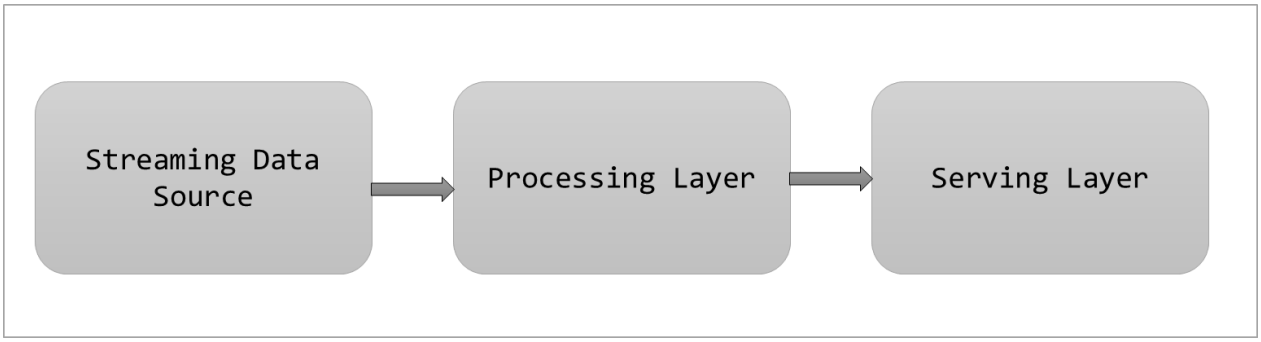
Serving Layer: Combines real-time data with batch data to provide a comprehensive view for analytics and reporting.

Kappa Architecture

The Kappa Architecture is a streamlined version of the Lambda Architecture, designed to handle real-time data processing exclusively. It eliminates the batch processing component, focusing solely on processing data streams in real-time. This architecture is particularly suited for use cases where historical data processing is not required, and the primary focus is on low-latency data ingestion and processing.

Kappa Architecture simplifies the data pipeline by using a single streaming layer to ingest, process, and store data. This approach reduces the complexity of managing separate batch and speed layers, making it easier to maintain and scale.

Architecture and Layers



Streaming Layer

The Streaming Layer is the core component of the Kappa Architecture, responsible for ingesting and processing data in real-time. This layer leverages stream processing frameworks like Apache Kafka, Apache Flink, or Apache Pulsar to handle continuous data streams.

The primary function of the Streaming Layer is to process incoming data events as they occur, applying transformations and computations on-the-fly. This ensures that data is processed with minimal latency and made available for immediate use.

Serving Layer

The Serving Layer in the Kappa Architecture provides a real-time view of the processed data. This layer stores the output of the Streaming Layer and supports real-time queries and analytics. The Serving Layer is designed to handle high query loads and deliver low-latency responses.

The integration between the Streaming Layer and the Serving Layer ensures that users always have access to the most current data, enabling real-time decision-making and analytics.

Possible Scenario

Scenario: A social media platform needs to monitor and analyze user activity data in real-time to detect trending topics and respond to user interactions immediately.

Streaming Layer: Ingests real-time user activity data (likes, shares, comments) and processes it to identify trends and generate metrics.

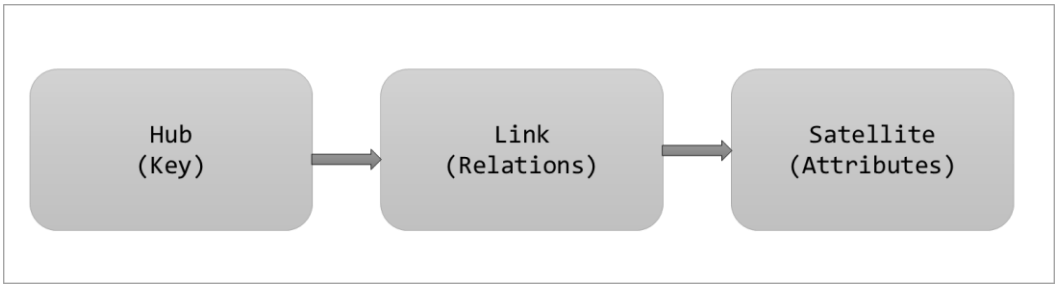
Serving Layer: Provides real-time analytics dashboards and notifications for trending topics and user engagement.

Data Vault Architecture

Data Vault Architecture is a data modeling technique designed to store historical data in a way that supports auditing and traceability. This architecture is built to handle large-scale data warehousing requirements, providing a flexible and scalable framework for managing historical data.

Data Vault Architecture consists of three primary components: Hubs, Links, and Satellites. These components work together to capture the historical state of data, maintain relationships between data entities, and store descriptive attributes.

Architecture and Layers



Hub

The Hub component contains unique business keys that represent core business entities. Hubs are designed to provide a consistent and immutable representation of business entities, ensuring data integrity and uniqueness. Each Hub table stores the primary keys for the business entities and their associated metadata, such as load timestamps and source identifiers.

Link

The Link component represents relationships between Hubs. Links capture the associations between business entities, maintaining the historical context of these relationships. Each Link table stores the foreign keys from the related Hubs, along with metadata that tracks the creation and modification of these relationships.

Satellite

The Satellite component stores descriptive attributes and historical data for the business entities represented in the Hubs. Satellites capture the changes to the attributes over time, allowing for detailed auditing and historical analysis. Each Satellite table is linked to a Hub or Link table and includes metadata that tracks the timing and source of the data changes.

Possible Scenario

Scenario: A healthcare provider needs to store and manage patient records, treatment histories, and medical billing information with strict auditing and historical tracking requirements.

Hub: Stores unique patient IDs and primary keys for treatments and billing.

Link: Represents relationships between patients, treatments, and billing records.

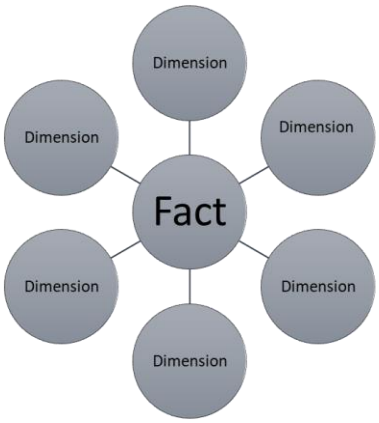
Satellite: Stores descriptive attributes and historical data such as treatment details, billing amounts, and changes over time.

Kimball’s Dimensional Data Warehouse

Kimball’s Dimensional Data Warehouse is based on Ralph Kimball's dimensional modeling approach, which focuses on creating user-friendly data structures for querying and reporting. This architecture emphasizes simplicity and ease of use, making it accessible to business users and analysts who need to perform ad-hoc queries and generate reports.

The Dimensional Data Warehouse architecture organizes data into Fact and Dimension tables, which are designed to support efficient querying and analysis. Fact tables store quantitative data, while Dimension tables store descriptive attributes related to the facts.

Architecture and Layers



Fact Tables

Fact tables are the core components of the Dimensional Data Warehouse, storing quantitative data for analysis. These tables contain measures such as sales revenue, transaction counts, and other numerical metrics. Fact tables are designed to support aggregation and summarization, enabling users to analyze data across various dimensions.

Dimension Tables

Dimension tables store descriptive attributes that provide context to the measures in the Fact tables. These attributes include information such as product names, customer demographics, dates, and locations. Dimension tables are structured to support hierarchical and categorical data, making it easy to filter, group, and drill down into the data during analysis.

Possible Scenario

Scenario: A marketing agency needs to analyze campaign performance, customer interactions, and sales data to optimize marketing strategies.

Fact Tables: Store quantitative data such as campaign impressions, clicks, and sales revenue.

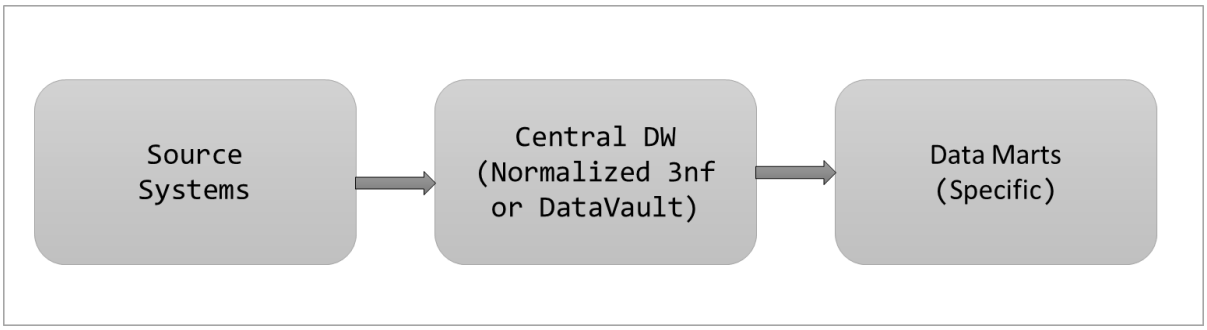
Dimension Tables: Store descriptive attributes like campaign names, dates, customer demographics, and product categories.

Inmon's Corporate Information Factory

Inmon's Corporate Information Factory (CIF) is based on Bill Inmon's approach to data warehousing, which emphasizes a centralized data warehouse with normalized data structures. This architecture is designed to provide a robust, scalable, and flexible data warehouse that integrates data from various sources and supports enterprise-wide reporting and analysis.

The Corporate Information Factory consists of a Centralized Data Warehouse and Data Marts. The Centralized Data Warehouse stores normalized data, ensuring data consistency and integration. Data Marts are denormalized subsets of data tailored to specific business areas or departments.

Architecture and Layers



Centralized Data Warehouse

The Centralized Data Warehouse is the core component of the Corporate Information Factory, storing normalized data from various source systems. This component ensures data consistency and integration across the organization, providing a single source of truth for enterprise data.

Data Marts

Data Marts are denormalized subsets of data extracted from the Centralized Data Warehouse. These components are designed to support specific business areas or departments, such as sales, finance, or human resources. Data Marts provide simplified and optimized data structures for efficient querying and analysis.

Possible Scenario

Scenario: A multinational corporation needs a robust data warehouse to integrate data from various departments (finance, HR, sales) for enterprise-wide reporting and analysis.

Centralized Data Warehouse: Stores normalized data from all departments, ensuring consistency and integration.

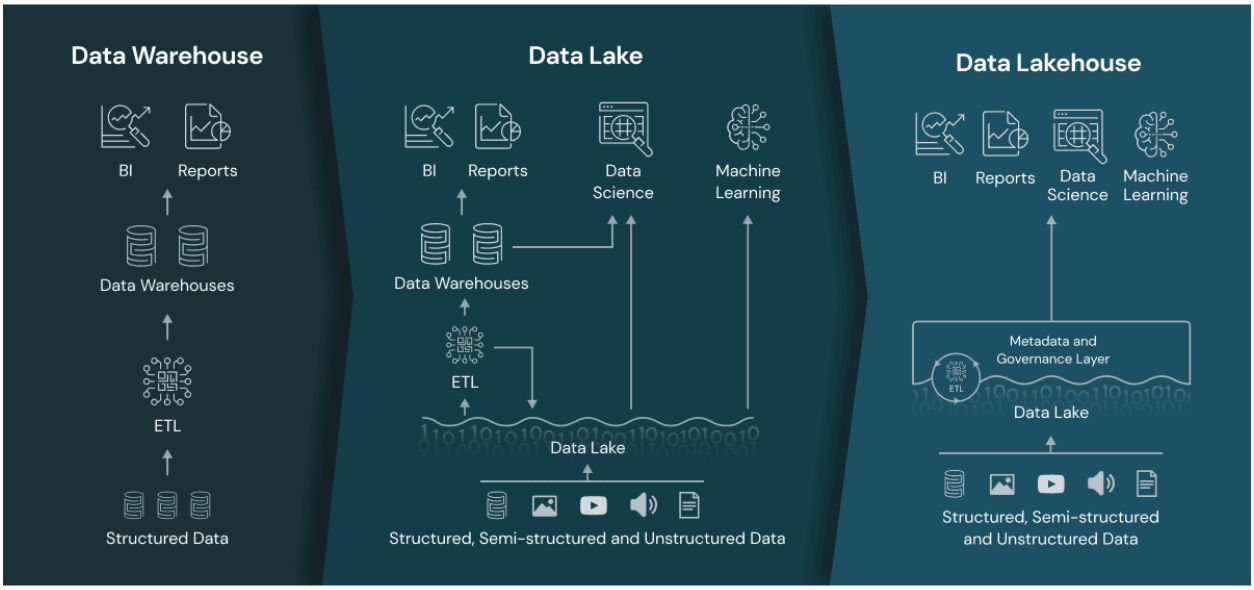
Data Marts: Specific data subsets for individual departments, such as sales data mart, finance data mart, and HR data mart.

Lakehouse Architecture

The Lakehouse Architecture is a modern data architecture that combines the flexibility of data lakes with the data management capabilities of data warehouses. This hybrid approach aims to provide a

unified platform for managing structured, semi-structured, and unstructured data, supporting both analytics and transactional workloads.

Architecture and Layers



Ref: https://www.databricks.com/blog/2020/01/30/what-is-a-data-lakehouse.html

Data Lake

The Data Lake component stores raw and semi-structured data, including log files, JSON data, and CSV files. This component leverages distributed storage systems like Amazon S3, Azure Data Lake, or Hadoop Distributed File System (HDFS) to store large volumes of diverse data types. The Data Lake provides a scalable and cost-effective storage solution for ingesting and managing raw data.

Data Warehouse

The Data Warehouse component manages structured data and supports ACID transactions for business-critical applications. This component leverages relational database management systems (RDBMS) or data warehouse solutions like Amazon Redshift, Google BigQuery, or Snowflake to enforce schema, ensure data consistency, and provide robust query capabilities. The Data Warehouse integrates with the Data Lake, allowing for seamless data movement and processing between the two components.

Possible Scenario

Scenario: A tech company wants to manage both structured data (e.g., sales records) and unstructured data (e.g., log files) efficiently, leveraging the benefits of both data lakes and data warehouses.

Data Lake: Stores raw and semi-structured data such as log files, JSON data, and CSV files.

Data Warehouse: Manages structured data and supports ACID transactions for business-critical applications.

ETL Pattern Variations

Batch ETL

Batch ETL is a traditional data processing approach where data is extracted, transformed, and loaded in scheduled batches. This method processes data at regular intervals, typically during off-peak hours, to reduce the impact on operational systems. Batch ETL is suitable for scenarios where real-time data processing is not critical, and data updates can be performed periodically.

Possible Scenario

Scenario: A university processes student enrollment data and course registrations nightly to update the student information system.

Implementation: Nightly jobs extract data from registration systems, transform it by cleaning and validating, and load it into the central student information system.

Real-time ETL

Real-time ETL processes data and loads it in real-time as it is ingested. This approach ensures that data is processed and made available immediately, supporting applications that require instant data updates. Real-time ETL leverages streaming data platforms and real-time processing frameworks to handle continuous data ingestion and transformation.

Possible Scenario

Scenario: An e-commerce website updates product inventory in real-time to reflect purchases and stock levels accurately.

Implementation: Streams data from purchase transactions and inventory updates, transforming it in real-time to update the product inventory database.

Micro-batch ETL

Micro-batch ETL processes small batches of data at frequent intervals, blending batch and real-time approaches. This method provides a balance between the efficiency of batch processing and the timeliness of real-time processing. Micro-batch ETL is designed to handle near-real-time data updates without the complexity and resource demands of continuous streaming.

Possible Scenario

Scenario: A news aggregator processes incoming news articles every few minutes to update its website with the latest content. Balancing near-real-time updates with efficient resource use for handling large volumes of incoming news articles.

Implementation: Ingests new articles in micro-batches, transforms them (e.g., tagging, categorizing), and loads them into the content management system every few minutes.

Summary

This **article** presents a comprehensive overview of ETL architecture designs, each tailored to specific data management needs and organizational goals. The Medallion Architecture uses layered data processing for data lakes, while the Lambda Architecture combines batch and real-time processing for financial market data. The Kappa Architecture focuses on real-time processing for social media activity monitoring, and the Data Vault Architecture supports historical data storage with auditing for healthcare records. Kimball's Dimensional Data Warehouse provides user-friendly analytics for marketing agencies, and Inmon's Corporate Information Factory offers a centralized data warehouse for multinational corporations. The Lakehouse Architecture unifies data management for structured and unstructured data in tech companies. ETL pattern variations, including batch, real-time, and micro-batch ETL, cater to different data processing needs, from nightly updates to real-time inventory management.

Each architecture provides unique methods for handling ETL processes, ensuring data quality, reliability, and performance, thus supporting various business needs and data complexities effectively.