Data and Data Sources

Data Catalogue

A data catalog is essentially an organized inventory of an organization's data assets. It uses metadata (data about data) to make it easier for users to find, understand, and use data.

Features:

- **Metadata Management:** Storing information about data sources, schemas, data lineage, and other relevant details.
- **Data Discovery:** Enabling users to search and find relevant data assets.
- **Data Lineage:** Tracking the origin and flow of data through various systems.
- **Data Governance:** Enforcing policies and controls related to data usage and access.

what-is-a-data-catalog/

Data Catalogue and Data Pipelines

A data catalog provides the "what" and "where" of data, while data pipelines handle the "how" of moving and transforming it.

- A data catalog enhances data pipelines by providing visibility into the data's origin, transformations, and quality.
- Data pipelines populate the data catalog with metadata as data moves through the stages.
- Data catalogs help data engineers and analysts understand the impact of changes to data pipelines.

Data Pipelines

A **data pipeline** is a set of processes that automate the movement, transformation, and processing of data from source to destination.

It ensures data is collected, cleaned, enriched, and stored efficiently for analysis or machine learning.

- guide-to-data-pipelines/

Key Components of a Data Pipeline

A data pipeline is a series of processes that move and transform data from one or more sources to a destination. Common stages include:

Extraction/Collection: Retrieving data from various sources, such as databases, APIs, files, or streaming platforms.

Ingestion: Bringing the extracted data into a staging area. Batch data ingestion and streaming data ingestion.

Storage: Storing data in a data warehouse, data lake, or a database. In case of ETL this step comes after data preparation.

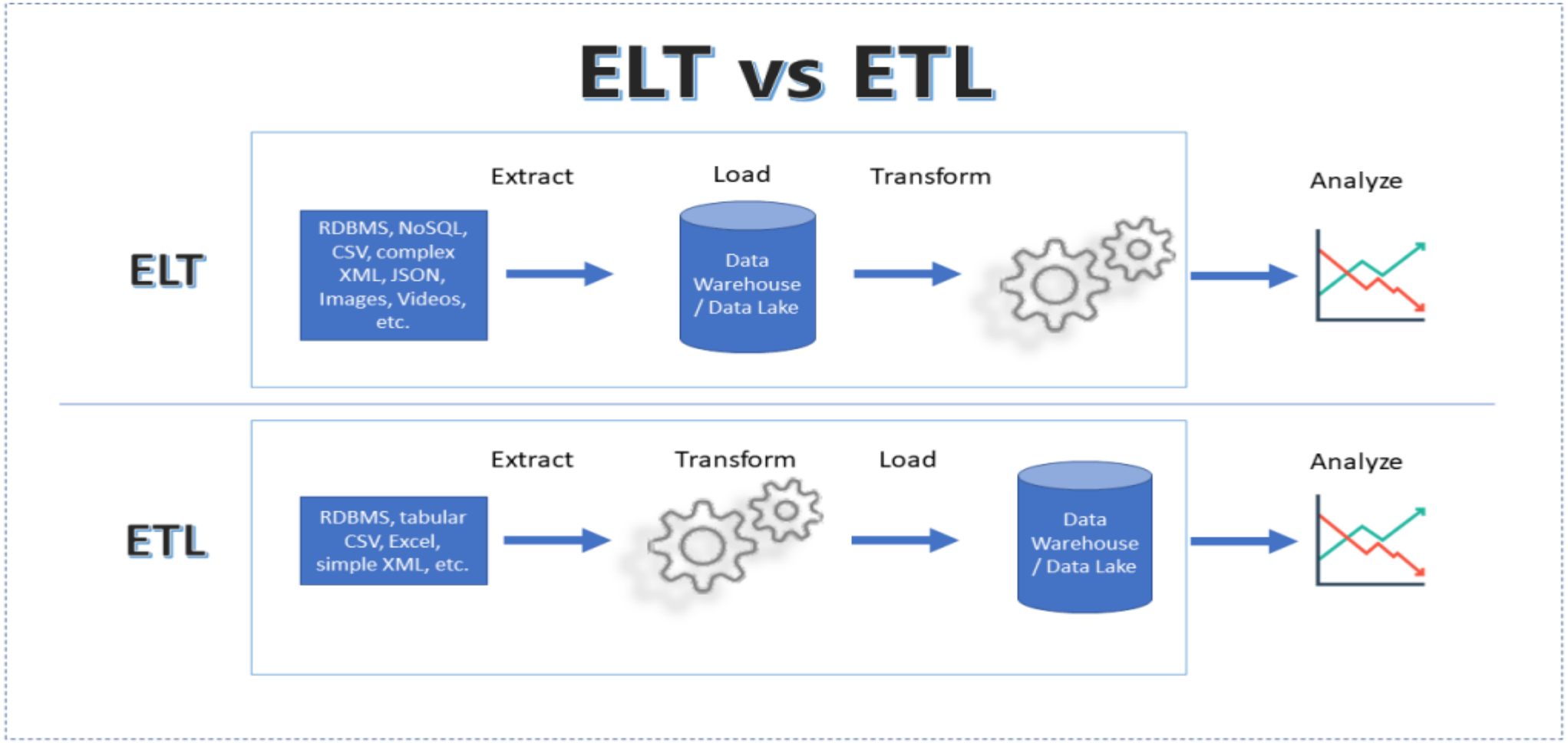
Key Components of a Data Pipeline

Data Preparation/Transformation:

- **Cleaning:** Identifying and correcting errors, inconsistencies, and missing values in the data.
- **Wrangling:** Transforming and structuring the data to make it suitable for analysis or other purposes. This may involve: Filtering, Aggregating, Joining, Formatting.
- **Exploration and data analysis:** This is where data analysts begin to look at the data, to find patterns, and to understand the data that has been gathered. Querying, reporting, or using data for machine learning.

Versioning & Monitoring : Maintaining a history of data changes, allowing for tracking and rollback if necessary. This is very important for data governance, and for reproducibility of analysis. Managing dependencies, scheduling tasks, and ensuring system reliability.

Stages of a Data Pipeline



Common Data Pipeline Patterns

Batch Processing Pipeline

- Processes data in chunks at scheduled intervals.
- Suitable for large-scale ETL workloads.
- **Example:** Nightly aggregation of customer transactions for financial reporting.

Technology Stack:

- Apache Spark, Apache Hadoop, Airflow, AWS Glue

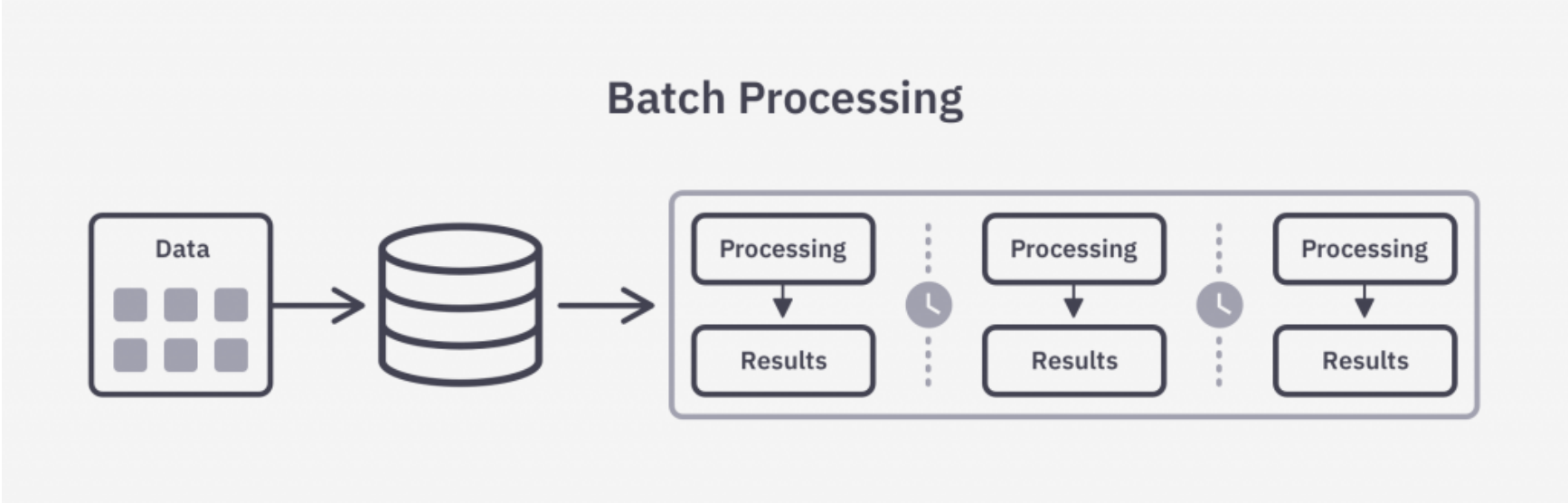
Streaming Data Pipeline

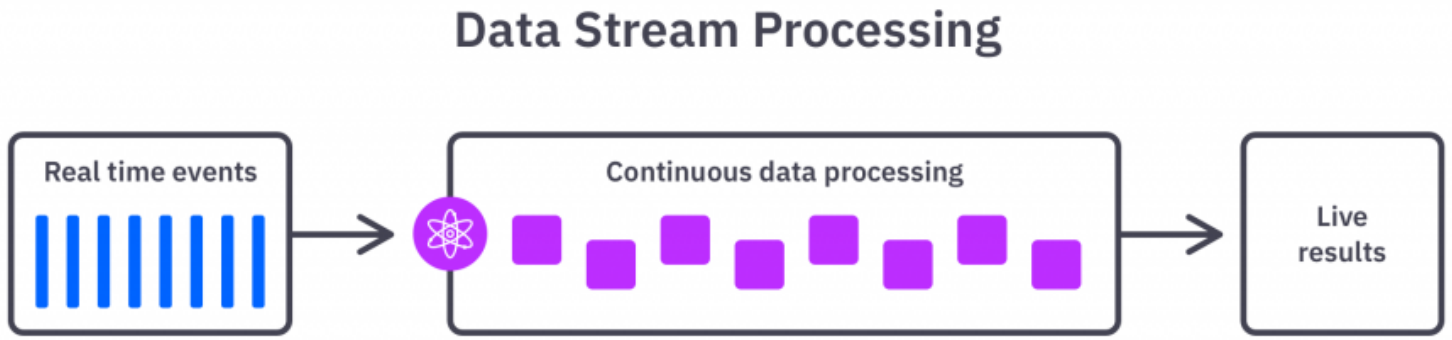
- Processes data in real-time or near real-time.
- Suitable for applications like fraud detection, live analytics, and IoT.
- **Example:** Monitoring website clicks or detecting fraudulent credit card transactions.

Technology Stack:

- Apache Kafka, Apache Flink, Spark Streaming, AWS Kinesis

Common Data Pipeline Patterns





Feature	Batch Processing	Stream Processing
Data Processing	Processes a large volume of data at once.	Processes data as it arrives, record by record.
Latency	High latency, as processing happens after data collection.	Low latency, providing near real-time insights.
Throughput	Can handle vast amounts of data at once.	Optimized for real-time but might handle less data volume at a given time.
Use Case	Ideal for historical analysis or large-scale data transformations.	Best for real-time analytics, monitoring, and alerts.
Complexity	Relatively simpler to implement with predefined datasets.	More complex, requires handling continuous streams.
Data Scope	Operates on a finite set of data.	Operates on potentially infinite streams of data.
Error Handling	Errors can be identified and corrected before execution.	Requires real-time handling of errors and failures.
Resource Usage	Resource-intensive during processing, idle otherwise.	Continuous use of resources.
Cost	Cost-effective for large volumes of data.	More expensive due to continuous processing.

Common Data Pipeline Patterns

Lambda Architecture (Hybrid Batch + Stream)

• Combines batch and real-time processing.

• **Example:** A weather app that uses real-time sensor data for short-term forecasts and batch data for long-term trends.

Layers:

• **Batch Layer:** Stores historical data.

• **Speed Layer:** Processes real-time data.

• **Serving Layer:** Merges both for a unified view.

Technology Stack:

- Apache Kafka, Apache Spark, HDFS, NoSQL Databases

Common Data Pipeline Patterns

Data Lake + Data Warehouse Hybrid

• Stores **raw data** in a **data lake** (e.g., AWS S3, Azure Data Lake).

• Transforms and moves structured data into a **data warehouse** (e.g., Snowflake, Redshift).

Example: An e-commerce company storing all transactions in a data lake but using a warehouse for analytics.

Technology Stack:

- AWS S3, Azure Data Lake, Snowflake, BigQuery

Best Practices for Data Pipelines

✓ **Use a Scalable Architecture** – Design for growing data volume.

✓ **Ensure Data Quality** – Use validation and anomaly detection.

✓ **Automate Orchestration** – Schedule and monitor pipelines with Apache Airflow.

✓ **Optimize Performance** – Use caching, indexing, and parallel processing.

✓ **Implement Security & Governance** – Encrypt data, use access controls, and comply with GDPR.

Data Transformation

Need of data transformation:

- To ensure data is consistent and compatible across different systems.
- To improve data quality by cleaning and standardizing it.
- To make data more suitable for specific analytical or modeling needs.

Data transformation includes:

1. **Data cleaning:**
    - Removing or correcting errors, inconsistencies, and duplicates.
    - Handling missing values (e.g., imputation).
2. **Standardization:**
    - Formatting data consistently (e.g., date formats, units of measure).
    - Normalizing or scaling numerical data.

Data Transformation

3. **Structuring:**
   - Changing the data's organization (e.g., pivoting, aggregating).
   - Converting data types (e.g., string to integer).
4. **Enrichment:** Adding new data or deriving new values from existing data. Joining data from multiple sources.
5. **Filtering:** Removing unwanted data.
6. **Aggregation:** Summarizing data.

Feature Management

Feature Selection

Selecting the most relevant features from a dataset while eliminating redundant or irrelevant ones.

- **Methods:**
  - **Filter Methods** (e.g., Correlation, Mutual Information)
  - **Wrapper Methods** (e.g., Recursive Feature Elimination)
  - **Embedded Methods** (e.g., Lasso Regression)

Feature Engineering

Creating new features from raw data to enhance model learning.

- **Common Techniques:**
  - **Polynomial Features** (e.g., x2,x3x^2, x^3x2,x3)
  - **Domain-Specific Transformations** (e.g., Date-Time Feature Extraction)
  - **Aggregations and Grouping** (e.g., Mean Purchase Amount per User)

Feature Management

Feature Transformation

Modifying features to meet the assumptions of machine learning algorithms.

•**Techniques:**

- **Scaling** (Standardization, Min-Max Normalization)
- **Encoding** (One-Hot Encoding, Label Encoding)
- **Log Transformations** (for skewed data)

Feature Store & Feature Versioning

Managing and reusing features efficiently in ML pipelines.

•**Feature Store Tools:** Tecton, Feast, AWS SageMaker Feature Store

•**Feature Versioning:** Tracking feature changes across different ML models.