

Foundations and Strategies in Data Engineering



CORE PRINCIPLES AND METHODS
FOR EFFECTIVE DATA MANAGEMENT



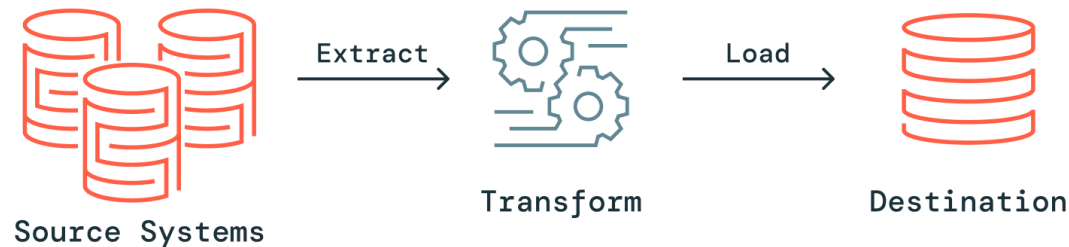
Agenda

- Essentials of Data Engineering
- Data Modeling and System Architectures
- Modern Data Processing Approaches
- Data Ingestion Techniques and Sources
- Real-Time Data Ingestion and Data Lake Architecture
- Incremental Loading and Change Data Capture
- Streaming Pipeline Reliability and Data Quality
- Schema Management in Modern Data Systems

Essentials of Data Engineering

Defining Data Engineering and Its Core Activities

ETL Process



Definition of Data Engineering

Data Engineering involves designing and maintaining systems for managing large volumes of data efficiently.

Building Data Pipelines

Creating robust ETL and ELT pipelines to extract, transform, and load data effectively.

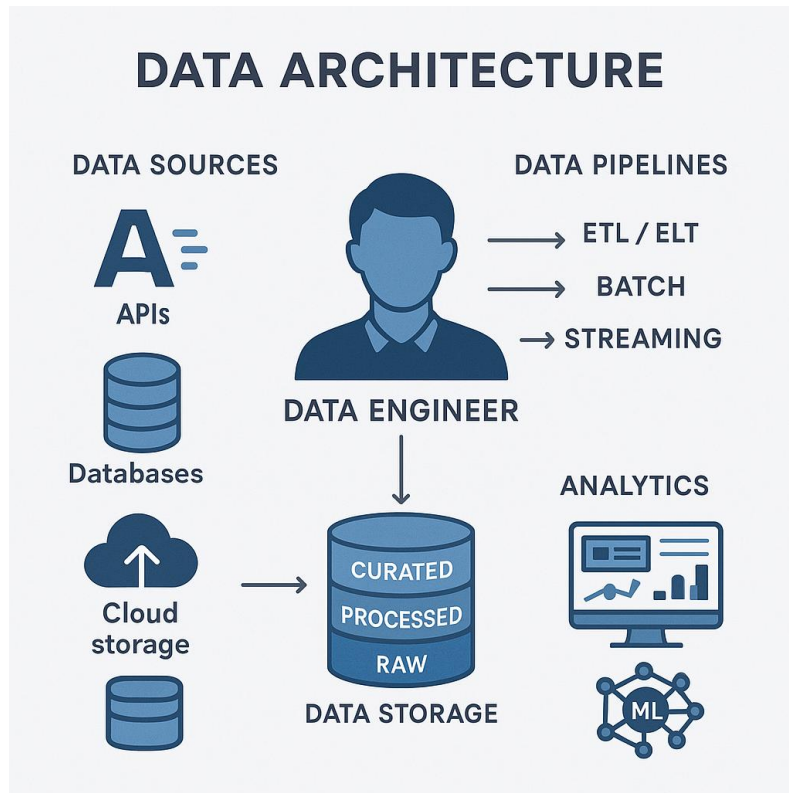
Data Quality and Reliability

Ensuring data cleaning, transformation, and enrichment to maintain high data quality and reliability.

Performance and Cost Optimization

Optimizing data systems for performance efficiency and cost-effectiveness.

Key Responsibilities of Data Engineers



Data Pipeline Development

Design and build scalable data pipelines using tools like Databricks, Spark, and cloud services.

Data Integration

Ingest data from diverse sources including APIs, databases, and cloud storage systems.


Collaboration and Automation

Work closely with data scientists and automate ingestion, transformation, and validation tasks.

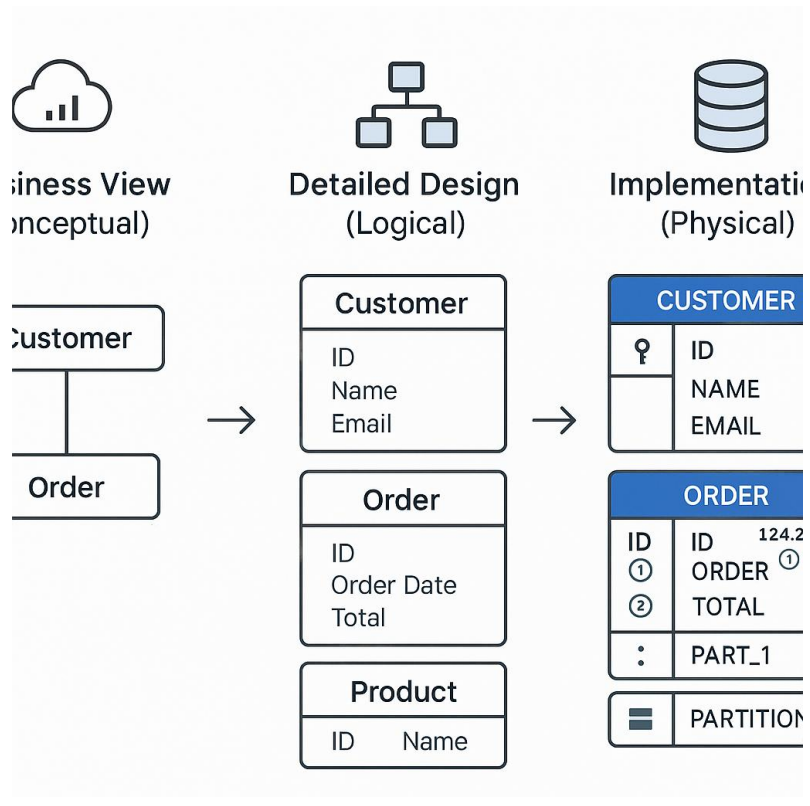
Governance and Compliance

Ensure data security, compliance, and maintain data lineage throughout the data lifecycle.

Data Modeling and System Architectures



Types of Data Models: Conceptual, Logical, and Physical



Conceptual Data Model

Provides a high-level view of data entities and their relationships, focusing on data requirements.

Logical Data Model

Details attributes, keys, and relationships without depending on any database platform or technology.

Physical Data Model

Focuses on implementation with tables, indexes, storage formats optimized for specific platforms.



OLTP vs. OLAP: Transactional and Analytical Systems

OLTP Characteristics

OLTP systems are optimized for high volume transactions with data integrity and use highly normalized schemas.


OLAP Characteristics

OLAP systems focus on complex queries and data analysis using denormalized schemas like star or snowflake.

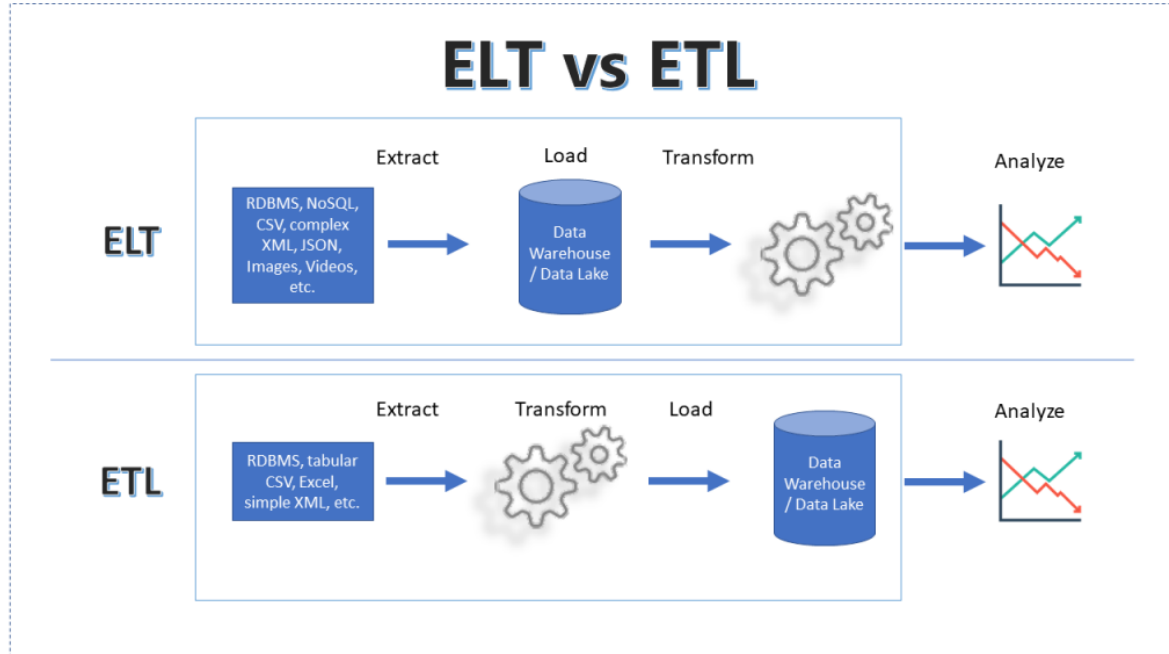
Key Usage Differences

OLTP supports day-to-day business processes, while OLAP supports business intelligence and reporting.

Modern Data Processing Approaches

The background of the slide is a dark, textured collage of various data visualization elements. It includes several line graphs with fluctuating data points, bar charts with vertical bars of varying heights, and a pie chart in the lower right corner. There are also some faint, semi-transparent text elements and grid lines scattered across the background, giving it a technical and data-driven appearance. A solid blue horizontal line is positioned below the main title.

ETL vs. ELT: Evolving Data Pipeline Paradigms



ETL Workflow

ETL extracts data, transforms it before loading into the destination, ideal for legacy systems needing pre-load cleansing.

ELT Workflow

ELT loads raw data first, then transforms it within the destination, suited for modern cloud data architectures.

Modern Cloud Benefits

Modern ELT leverages compute and storage separation for efficiency, supporting semi-structured data and scalable compute.

Databricks Optimization

Delta Lake and Spark SQL enhance ELT efficiency with ACID transactions and scalable compute resources.

Batch and Streaming Processing: Use Cases and Tools

Batch Processing Overview

Batch processing collects and processes data at set intervals, ideal for large volumes and non-real-time needs.

Use Cases for Batch

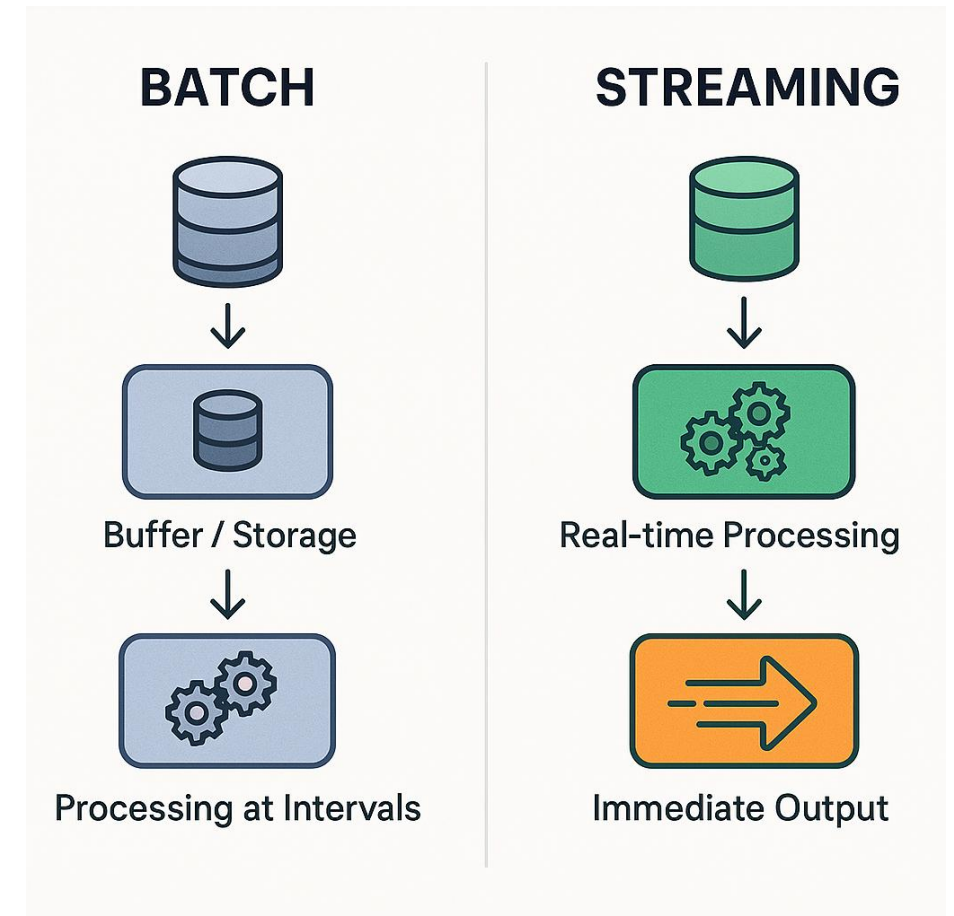
Typical use cases include ETL jobs and nightly report generation for large datasets.

Streaming Processing Overview

Streaming processes data immediately upon arrival, supporting low latency and real-time scenarios like IoT and fraud detection.

Streaming Tools

Popular tools for streaming include Structured Streaming in Databricks and Apache Kafka for scalable data pipelines.



Data Ingestion Techniques and Sources

Overview of Data Ingestion Methods



Bulk Load

Bulk load ingests large datasets at once, ideal for initial loads or migrations.

Incremental Load

Incremental load processes only new or changed data since the last ingestion.

Change Data Capture

CDC tracks and replicates only changes in data for efficient synchronization.

Streaming Ingestion

Streaming ingestion continuously captures data in real-time for analytics.

Extracting Data from APIs, Databases, and Object Storage



APIs for Data Extraction

APIs use REST and GraphQL endpoints with mechanisms for authentication, pagination, and rate limits.

Database Connectivity

Use JDBC and ODBC connectors to connect relational databases like SQL Server, MySQL, Oracle, and PostgreSQL.

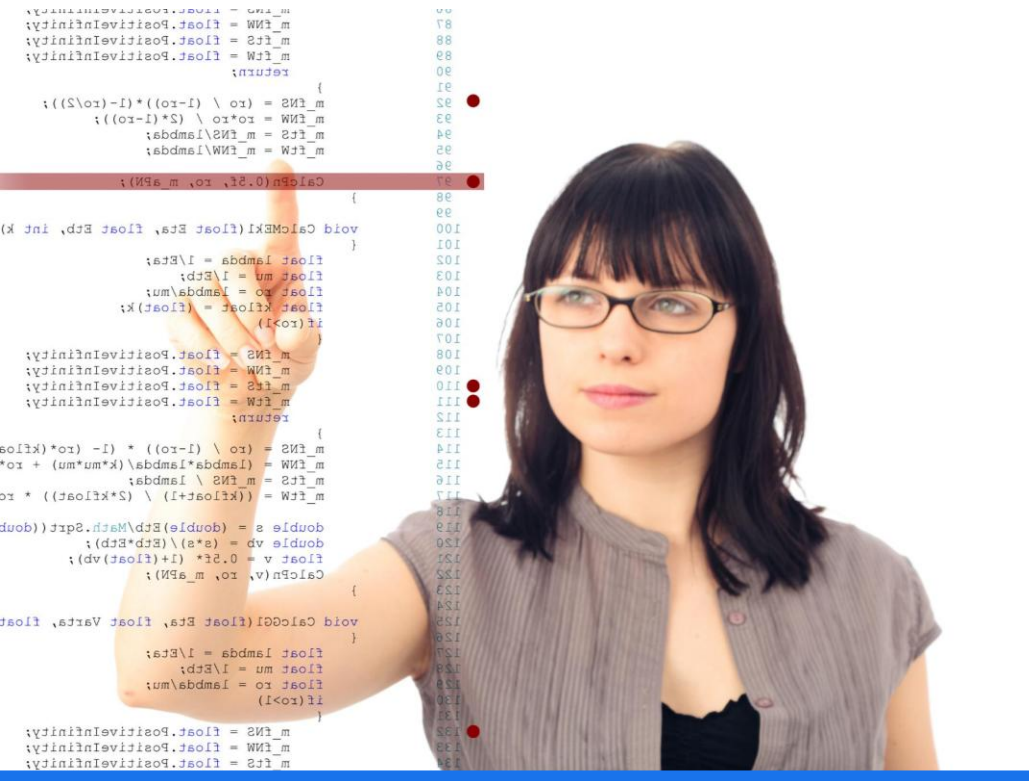
Cloud Object Storage

Cloud storage platforms such as S3, HDFS, ADLS, and GCS support file-based data ingestion formats like CSV and Parquet.

Databricks Integration

Databricks offers utilities like `dbutils.fs` and `spark.read` with connectors for major cloud storage providers.

Working with Diverse Data Formats



Common Data Formats

CSV is simple and human-readable, while JSON supports nested, semi-structured data popular in APIs.

Hierarchical and Legacy Formats

XML is hierarchical and verbose, often used in legacy and enterprise systems for structured data.

Big Data Optimized Formats

Parquet and ORC are columnar formats optimized for big data processing with complex type support.

High-Performance Binary Formats

Protobuf and Thrift are compact binary formats designed for efficient, cross-language data exchange.

Real-Time Data Ingestion and Data Lake Architecture

Streaming Platforms: Kafka, Kinesis, and Pub/Sub



Apache Kafka Features

Open-source and distributed streaming platform offering high throughput and fault tolerance for real-time data processing.

AWS Kinesis Capabilities

Managed AWS streaming service supporting shards, automatic scaling, and data replay for flexible streaming applications.

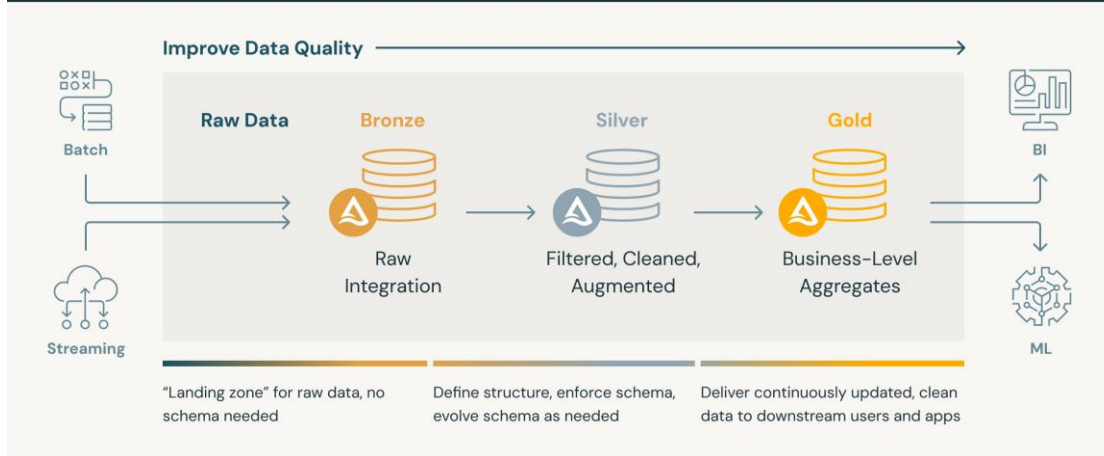
Google Pub/Sub Overview

Managed messaging service on Google Cloud supporting pub/sub pattern and scalable data streaming solutions.

Databricks Streaming Integration

Native connectors allow seamless ingestion of streaming data directly into Delta Lake for unified analytics.

Building reliable, performant data pipelines with DELTA LAKE



Raw Layer (Bronze)

The raw layer stores unprocessed data in its original form, maintaining full fidelity for auditing and replay.

Processed Layer (Silver)

Processed data is cleaned, filtered, and transformed to be suitable for analytics and data integration.

Curated Layer (Gold)

Curated data is business-ready, high-quality, aggregated for dashboards, machine learning, and reporting needs.

Data Lake Layers: Raw, Processed, and Curated

Incremental Loading and Change Data Capture

Incremental Load and CDC: Concepts and Benefits



Incremental Load Defined

Incremental load processes only data changes since the last update using timestamps or versioning.



Change Data Capture (CDC)

CDC captures inserts, updates, and deletes to process only changed records efficiently.



Benefits of Incremental Load and CDC

These methods reduce compute power, bandwidth usage, and overall processing time significantly.

Log-Based and Trigger-Based CDC Mechanisms



Log-Based CDC Mechanism

Monitors database transaction logs to detect changes with low latency and minimal source impact.

Advantages of Log-Based CDC

Captures all changes including deletes, ensuring comprehensive data replication with minimal delay.

Trigger-Based CDC Mechanism

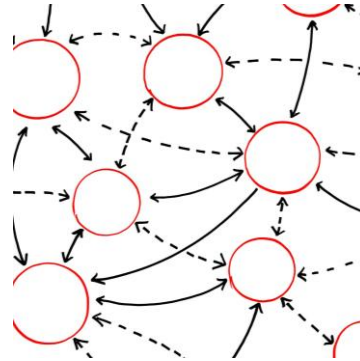
Uses database triggers to capture changes in real time, allowing inclusion of custom logic.

Limitations of Trigger-Based CDC

May increase database load and is harder to scale compared to log-based CDC.

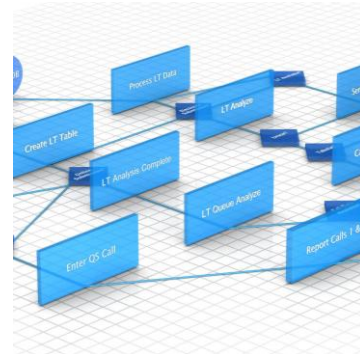
Streaming Pipeline Reliability and Data Quality

Watermarking and Checkpointing for Streaming Pipelines



Watermarking Overview

Watermarking defines completeness point and manages late-arriving streaming data effectively.



Ensuring Correct Aggregations

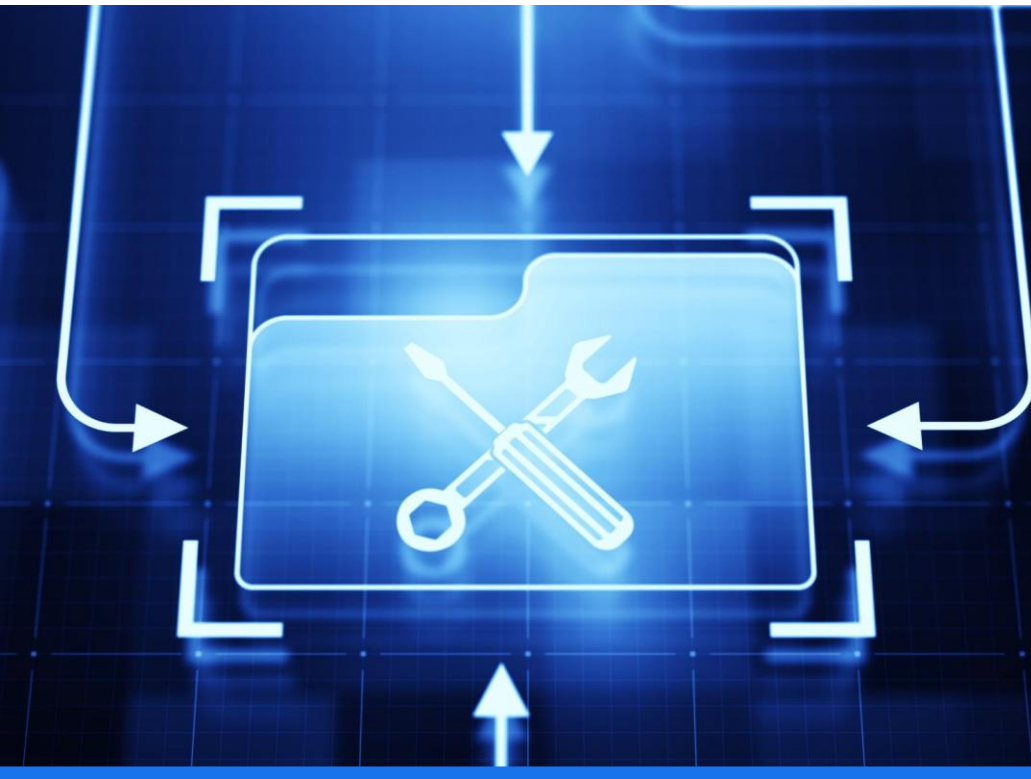
Watermarking ensures accurate aggregations by considering timely and late data in streams.



Checkpointing Functionality

Checkpointing stores streaming job states to enable fault tolerance and exactly-once processing.

Data Validation and Quality Assurance



Validation Steps Overview

Key validation steps include schema checks, null values, duplicate detection, and ensuring referential integrity.

Data Type and Schema Checks

Ensuring data types match expected schema prevents errors and maintains data consistency.

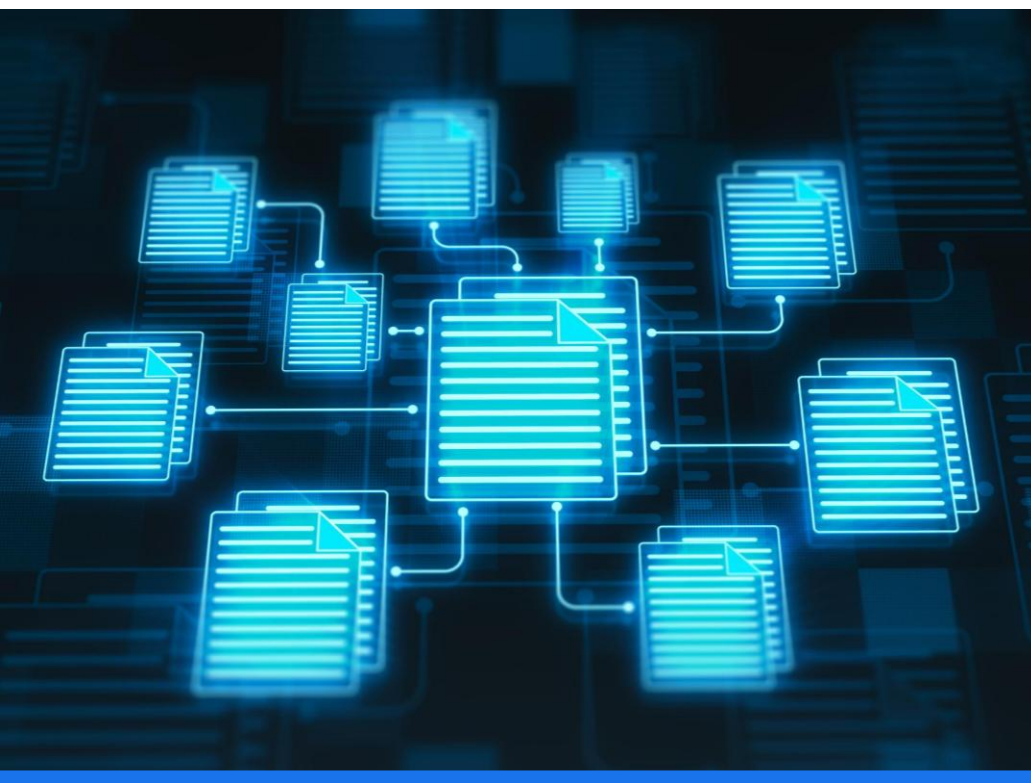
Null and Duplicate Detection

Detecting null values and duplicates improves data quality and reliability.

Validation Tools

Popular tools include Great Expectations, Databricks Expectations with PySpark, and custom validation logic.

Handling Corrupt Records in Large Datasets



Permissive Mode Usage

Using permissive mode helps isolate corrupt records for further examination and handling.

Logging and Manual Review

Logging bad records enables manual review and correction to improve data quality.

Automated Alerting

Automating alerts notifies teams promptly about data quality issues for quick response.

Schema Evolution

Flexible parsing and schema evolution support handling changes and inconsistencies in datasets.

Schema Management in Modern Data Systems

Schema Evolution and Enforcement in Avro & Parquet

Schema Evolution

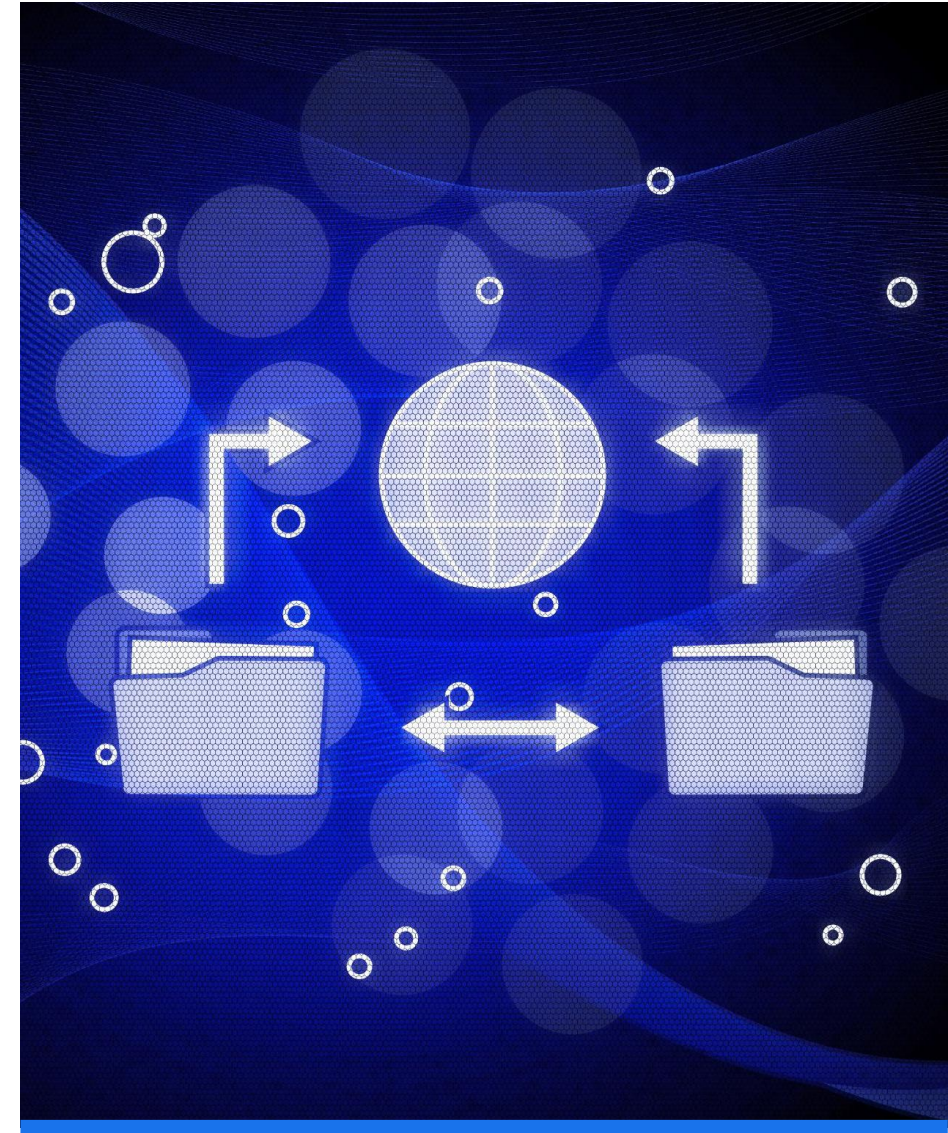
Allows adding, removing, or changing fields without breaking compatibility in data schemas.

Schema Enforcement

Prevents invalid data from being written, ensuring data quality and consistency.

Best Practices


Version schemas and implement automated pipeline checks to maintain data integrity.



Conclusion

Foundation of Data Engineering

Data engineering forms the foundation for creating scalable and reliable data systems essential for business growth.



Core Concepts Understanding

Grasping core concepts and strategies enables effective design of data infrastructures and pipelines.

Designing Effective Pipelines

Well-designed data infrastructures support diverse business needs through efficient data pipelines.