DataOps

Data Engineering

Fundamentals

Data Engineering Fundamentals

"

- Data Sources
- Data Formats
- Data Models
- Data Storage Engines & Processing
- Modes of Dataflow

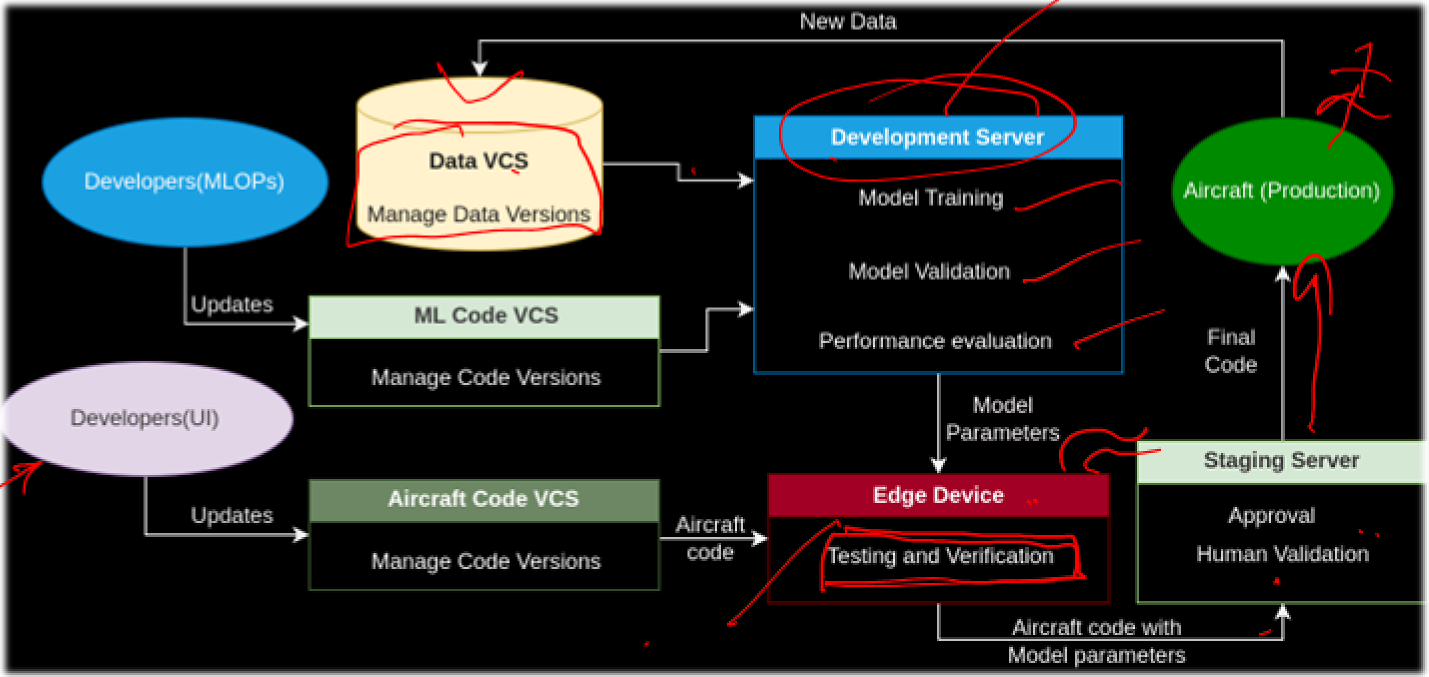
Data Engineering

AI-Agent – Atypical architecture

Data Engineering

AI-Agent – Atypical architecture

Sup - #100



Data Sources

Data Engineering

Data

- something raw, meaningless
- structured, unstructured
- numbers, text, symbols, pixe

Data Engineering

Database

- If data models describe the data in the real world, database specify how the data should be stored on machine
- Storing data is only interesting if we intend to retrieve it later
- Need to know not only how it's formatted but also structured
- Use cases
  - Feature engineering, prediction services, etc.
- Types of Data (storage):
  - Historical data in data storage engines
  - Streaming data in real-time transports

Data Engineering

Data Sources

- ML system needs to work with many different sources with different characteristics, purposes and different processing methods
- User Input:
    - Text, image, video, files, etc
        - High risk for errors
    - User requires fast response, thus needs to be processed quickly
- System-Generated Data
    - Data generated by different components of the system:
        - results, log, system outputs, user activities, etc.
    - Debugging ML system is hard, so log everything

Data Engineering

Data Sources

- Internal Database
  - Data generated by various services and enterprise application within the entity
    - Inventory, customer relationship, users, sales, etc
    - Could be used in different components of ML systems
- First-party Data
  - Data of users or customers
- Second-party Data
  - Collected by another company on their own customers and made it available to us
- Third-party Data

Data Formats

Data Formats

- Storing data isn't always straightforward since data comes from multiple sources with different access patterns
- Challenges:
  - How to store different type of data, for e.g., multimodal data?
  - Where to economically store and how to access with low latency?
  - How to store complex data so that it can be loaded and run correctly on different hardware?

Data Formats

Format	Binary/Text	Human-readable	Example use cases
JSON	Text	Yes	Everywhere
CSV	*Text*	*Yes*	*Everywhere*
Parquet	*Binary*	*No*	*Hadoop, Amazon Redshift*
Avro	Binary primary	No	Hadoop
Protobuf	Binary primary	No	Google, TensorFlow (TFRecord)
Pickle	Binary	No	Python, PyTorch serialization

Data Formats

JSON

- JavaScript Object Notation
- Most programming languages can generate and parse
- Painful to change the schema
- Human readable
- More space needed since text file

```json
{
  "name": "John Doe",
  "age": 30,
  "email": "johndoe@example.com",
  "address": {
    "street": "123 Main Street",
    "city": "New York",
    "state": "NY",
    "zip": "10001"
  },
  "hobbies": ["reading", "traveling", "photography"],
  "isStudent": false
}
```

Data Formats

Row-major Vs. Column-major

- CSV – Comma-separated values
  - Text format and row-major
  - Good for accessing samples & w
  - Low performance
- Parquet – Hadoop, Amazon Reds
  - Binary format and column-major
  - Highly optimized for big data processing
  - Offers efficient compression and encoding schemes
  - Good for accessing features and read
  - Up to 2X faster to upload and needs 6X less storage in Amazon S3, compared to text

```
employee.parquet:

----------------------------------------
|  Name      |  Age  |  Department    |
----------------------------------------
| John Doe |   30    |  Engineering   |
| Jane Smith|   28    |   HR            |
| Alex Brown|   32    |   Sales         |
----------------------------------------
```

R1 R2 C1 C2 C3

[R1 R2 ... Rn] Row major

Column major = [C1, C2, ... CN]

Data Formats

NumPy Vs. Pandas

- NumPy
  - an n-dimensional array object called ndarray, which is a homogeneous, fixed-size array
  - Major order can be specified, when created, it's row-major by default
- Pandas
  - Two primary data structures:
    - Series and DataFrame.
  - Series: A one-dimensional labeled array capable of holding any data type. It is similar to a column in a spreadsheet or a database table.
  - DataFrame: A two-dimensional labeled data structure with

Data Formats

Text (2ms)

- Human-readable: Text data is represented as plain text (e.g., ASCII, UTF-8).
- Interoperability: Text files can be opened and processed by various software applications and programming languages
- Portability: Text files can be easily shared and transferred between different systems.
- Larger file size: Textual representation may require more space to store the same information compared to binary formats.
- Slower processing: Parsing and processing text data can be slower compared to binary data due to the need for data conversion and interpretation. Common text formats include

Data Formats

Binary

- Compact representation: Binary data is represented in a more compact form, often using numeric values, binary codes, or serialized objects.
- Efficient storage: Binary formats typically require less storage space compared to text formats, especially for large datasets.
- Faster processing: Binary data can be processed faster as it can be directly interpreted by machines without the need for ~~data conversion or~~ parsing.
- Limited human-readability: Binary data is not easily readable or interpretable by humans without specialized tools or knowledge.

“Data Models

Data Models

What is a Data Model?

- Data model describes how data is represented
  - An employees can be described using their qualifications, experience, origin, background, etc.
  - Alternatively, employees can also be described using their job profile, team, performance, impact, etc.
- Representation of data not only affects the way the systems are built, but also the problems it can solve
- Major classification
  - Relational Model and NoSQL model

Data Models

Relational Model

- The relational data model is the foundation of modern database systems
- It organizes data into tables consisting of rows and columns, representing entities and their attributes
- Key Concepts:
  - Tables: Data is stored in tables with predefined columns and rows
  - Rows: Each row represents a unique record or instance of an entity
  - Columns: Columns define the attributes or properties of the entities
  - Primary Key: A unique identifier for each row in a table

Data Models

Relational Model

Column-major:

- Data is stored and retrieved column by column
- Good for accessing features

Row-major:

- Data is stored and retrieved row by row
- Good for accessing samples

	Column 1	Column 2	Column 3
Example 1	...	...	...
Example 2	...	...	...
Example 3	...	...	...

Data Models

Relational Model

- Querying Data:
  - SQL (Structured Query Language) is used to retrieve and manipulate data
    - SELECT statement: Retrieves data from one or more tables
    - JOIN operation: Combines rows from two or more tables based on related columns
    - WHERE clause: Filters data based on specified conditions
- Advantages of Relational Data Model
  - Data consistency: Relationships ensure consistent data across tables
  - Flexibility: Easy to add, modify, or remove data without affecting other parts

Data Models

Relational Model

- Limitations of Relational Data Model:
  - Complex relationships: Many-to-many relationships may require additional tables
    - Eg: students can enroll in multiple classes, and each class can have multiple students
  - Performance impact: Join operations on large tables can impact query performance
  - Vertical scalability: Scaling relational databases vertically (adding more resources to a single server) has limitations
- Real-world Examples:
  - Examples of popular relational database management systems (RDBMS): Oracle, MySQL, SQL Server, PostgreSQL

user management [Role-based access]

Data Models

NoSQL

- Not Only SQL is a class of database systems that provide flexible data models beyond the traditional relational model
- Designed to handle large-scale distributed data and non-uniform data structures
- Key Concepts:
  - Schema-less: Do not enforce a fixed schema, allowing for dynamic and flexible data structures
  - Document-oriented: Some NoSQL databases store data in documents (e.g., JSON, XML), representing entities as self-contained documents
    - All information about a record is stored in a document, and easy to retrieve

Data Models

NoSQL

- Graph databases
  - NoSQL databases designed for highly connected data, storing entities as nodes and relationships as edges.
- Scalability
  - Built for horizontal scalability, allowing them to handle large volumes of data and high traffic loads.
  - Distributed architecture: Data is distributed across multiple nodes, enabling parallel processing and fault tolerance.
  - Sharding: Data is partitioned and distributed across multiple servers for efficient storage and retrieval.

Data Models

NoSQL

- Data Model Flexibility:
  - Schema-flexible, allowing for agile development and accommodating evolving data structures
- No need for migrations
  - Changes to the data model can be made on-the-fly without requiring complex schema migrations
- Dynamic attributes
  - Documents or entities can have varying attributes, providing flexibility for different data requirements

Data Models

NoSQL

- Use Cases:
  - Big data analytics: Excel at handling large volumes of diverse data
  - Suitable for real-time applications, such as social networks, IoT, and gaming.
  - Document-oriented NoSQL databases are well-suited for content management systems and CMS-like applications.
- NoSQL Databases:
  - MongoDB: A popular document-oriented NoSQL database.
  - Apache Cassandra: A distributed columnar NoSQL database.
  - Neo4j: A graph database for highly connected data.
  - Redis: A key-value store with in-memory caching capabilities

Data Models

Structured Vs. Unstructured Data

Structured

- Follows a predefined format and is organized into fixed fields and records.
- Typically stored in relational databases with tables, rows, and columns.
- A clear and rigid schema that defines the data types and relationships.
- Enables straightforward data querying, aggregation, and analysis.
- Excel spreadsheets, SQL databases, CSV files.

Unstructured Data

- Lacks a predefined format or structure and doesn't fit into traditional databases.
- Can be in the form of text documents, images, videos, social media posts, etc.
- Doesn't adhere to a rigid schema, making it challenging to define and categorize.
- Analyzing unstructured data requires advanced techniques like natural language processing, image recognition, etc.
- Text documents, social media feeds, multimedia content

Data Models

Structured Vs. Unstructured Data

- Business requirements change over time, committing to predefined schema can become too restricting
- Unstructured allows more flexible storage options.
- Data warehouse:
  - A repository for storing structured data.
- Data Lakes:
  - A repository for storing unstructured data
  - Used to store raw data before processing

“Data Storage Engines & Processing (Database)

Data Storage Engines & Processing

Storage engine, also known as Database

- An implementation of how data is stored and retrieved on machines
- Optimized for two types of workloads
  - Transactional processing
  - Analytical processing

Data Storage Engines & Processing

Transactional Processing

- Focuses on managing day-to-day business operations and transactions.
- Involves creating, updating, and deleting individual records in a real-time or near-real-time manner.
- Handle concurrent access from multiple users/applications and ensure consistency.
- ACID properties: Transactional processing emphasizes data integrity and consistency through
    - Atomicity: Ensures all steps are successfully complete, the transaction fails if one step fails
    - Consistency: All transactions must follow predefined rules
    - Isolation: Each transaction is private, no race condition on data access

Data Storage Engines & Processing

Transactional Processing

- Relational model:
  - Transactional systems often use relational databases with well-defined schemas and structured data.
  - OLTP databases: Online Transaction Processing (OLTP) databases are optimized for handling frequent read and write operations.
- Use Cases:
  - Order processing: Managing customer orders, inventory, and payment transactions in e-commerce systems.
  - Banking transactions: Handling account transfers, withdrawals, and deposits in banking systems.
  - Point of sale (POS) systems: Processing sales transactions in

Data Storage Engines & Processing

Transactional Processing

- Focuses on gaining insights and understanding trends from large volumes of data.
- Involves complex queries, aggregations, and data manipulations for decision-making and business intelligence.
- Read-intensive operations: Analytical processing primarily involves querying and analyzing data without frequent updates or modifications.
- Complex queries: Analytical systems execute complex queries involving aggregations, joins, and calculations on large datasets.
- Historical and summarized data: Analytical systems often work

Data Storage Engines & Processing

Transactional Processing

- OLAP databases: Online Analytical Processing (OLAP) databases or data warehouses are commonly used for analytical processing.
- Use Cases:
  - Business intelligence: Analyzing sales data, customer behavior, and market trends to make informed business decisions.
  - Data mining: Identifying patterns, correlations, and anomalies in large datasets for predictive analytics and pattern recognition.
  - Reporting and dashboards: Generating summary reports and interactive dashboards to visualize and communicate data insights.

Data Storage Engines & Processing

Analytical Processing

- OLTP and OLAP have become outdated
  - Separation of transactional and analytical databases was due to limitation of technology
    - Was hard to have databases that could handle both queries
    - We have transactional databases that can handle analytical queries and vice versa
      - CocroachDB
      - Apache Iceberg, DuckDB
  - Storage and processing are tightly coupled
    - May result in data being stored in multiple databases to handle different queries
    - Decouple storage and processing: Google BigQuery, Snowflake

Data Storage Engines & Processing

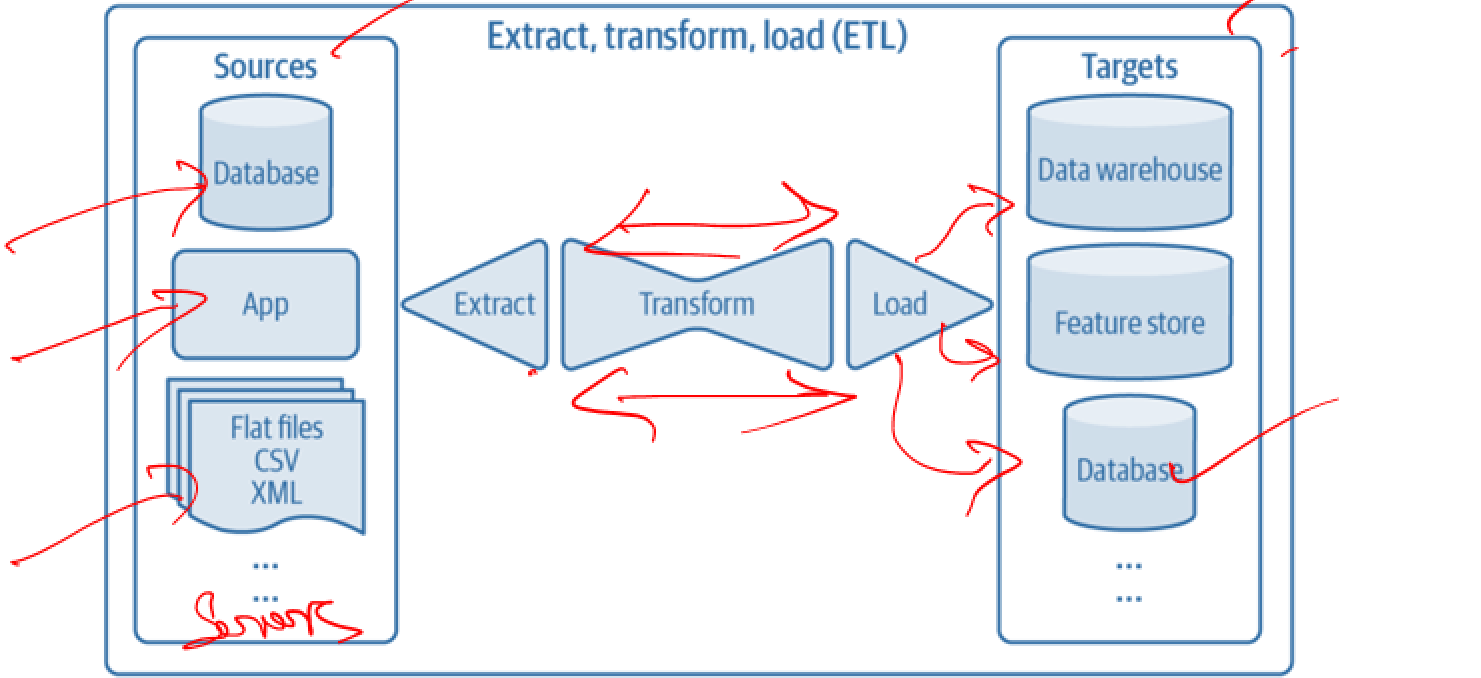
Extract, Transform, and Load

- A process used in data integration and data warehousing
  - to extract data from various sources
  - transform it into a desired format
  - and load it into a target system for analysis, reporting, or other purposes
- Essential for integrating data from disparate sources, consolidating it into a unified format, and making it available for analysis, reporting, and decision-making.
- Workflows needs to be automated, allow for regular data updates and maintaining data consistency over time, to build ML Systems.

Data Storage Engines & Processing

Extract, Transform, and Load

Data lake



Data Storage Engines & Processing

Extract

- Data is gathered or retrieved from multiple sources, which can include databases, files, APIs, web services, or external systems.
- Extracted data is typically in its raw or source form, reflecting the structure and format of the source system.
- Data might be corrupt and mal-formatted
- Validation is must, and reject the data that doesn't meet the requirement
- Any wrongdoing affects the entire ETL process

Data Storage Engines & Processing

Transform

- Extracted data is processed and transformed to meet the requirements of the target system or data model.

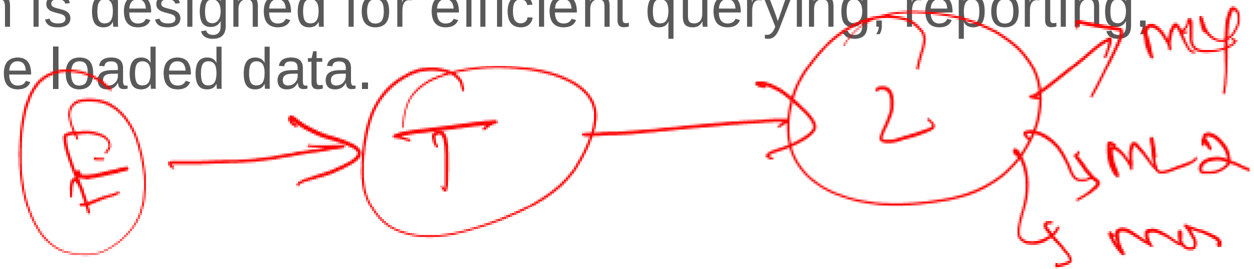
ML

- Involves cleaning, filtering, aggregating, merging, or modifying the extracted data to ensure consistency, compatibility, and quality.
- May include data cleansing (removing duplicates, handling missing values), data aggregation (summarizing data, calculating metrics), data enrichment (merging with external datasets), and data validation (verifying data integrity, applying business rules).
- Might need to join from multiple sources

Data Storage Engines & Processing

Load

- Transformed data is loaded into the target system, which can be a data warehouse, data mart, analytical database, any other storage system or ML system.
- Loading involves inserting or updating the transformed data in the target system, ensuring data integrity and maintaining relationships.
- The target system is designed for efficient querying, reporting, and analysis of the loaded data.



Modes of Dataflow

Modes of Dataflow

Main Modes

- Data Passing through Database
  - Data is stored and transferred through relational or non-relational databases.
  - Databases provide persistent storage and structured data management.
  - Data is accessed using SQL or NoSQL queries.
  - Examples: Oracle, MySQL, MongoDB.
- Data Passing through Services
  - Data is exchanged through web services or APIs.
  - Services provide a standardized way of communication between systems.

Modes of Dataflow

Main Modes

- Data Passing through Real-time Data Transport
  - Data is streamed or transferred in real-time for immediate processing or analysis.
  - Real-time data transport enables low-latency data delivery.
  - Data is sent continuously or in small batches.
  - Examples: Messaging systems (Kafka, RabbitMQ), real-time streaming platforms (Apache Kafka, Apache Flink).

Modes of Dataflow

Use Cases

- Database Mode
  - Traditional business applications, transactional systems, data warehousing.
- Service Mode
  - Integrating systems, microservices architecture, API-based interactions.
- Real-time Mode
  - IoT applications, real-time analytics, event-driven systems.
- Hybrid Approaches:
  - Often, multiple modes of dataflow are combined to meet specific requirements.

Modes of Dataflow

To be considered

- Data Volume:
  - Consider the volume of data being transferred and ensure scalability.
- Data Consistency:
  - Maintain data consistency across different modes of dataflow.
- Data Security:
  - Implement security measures to protect data during transfer.
- Latency Requirements:
  - Choose the appropriate mode based on the required data processing speed.

Data Processing

Batch Processing Vs. Stream Processing

- Once data arrives in data storage engines like databases, data lakes, or data warehouse, it become historical data
- Batch Processing:
  - Historical data is often processed in batch jobs.
  - Periodically, for e.g., once a day to compute average surge charge change.
- This is opposite to streaming data since the data is still streaming in.
  - When done right, streaming processing can give low latency (no need to write to database)

Data Processing

Batch Processing Vs. Stream Processing

- Processed in fixed-size batches
- Collected over a period and processed together as a batch
- Typically involves processing large volumes of data at once
- Typically higher latency
- Overnight data processing, generating reports

- Processed in real-time as continuous streams
- Processed as it arrives, piece by piece
- Supports real-time analytics, immediate insights, and near-instantaneous processing
- Low higher latency
- Real-time analytics, fraud detection, IoT data

Data Engineering Fundamentals

Summary

- Discussed the importance of data in developing ML Systems
- Learned to choose the right format to store and access data
- Discussed different formats, data models, data engines and pr

Data Sources

Data Formats

Data Models

Data Storage Engines & Processing

Modes of Dataflow

“The Evolving Landscape of Data Storage and Processing

Good to know . . .

Data Engineering

What is data engineering?

- The practice of taking raw data from a data source and processing it so it’s stored and organized for a downstream use case such as data analytics, business intelligence (BI) or machine learning (ML) model training.
- In other words, it’s the process of preparing data so value/insight can be extracted from it.

https://www.databricks.com/resources/ebook/big-book-of-data-engineering

Understanding Key Concepts in Modern Data Engineering

Data Warehouse - The Traditional Approach

- A centralized repository for structured data, specifically designed for analytical queries and reporting.
- Key Characteristics:
  - Structured data (relational tables)
  - Pre-defined schema (data is modeled before ingestion)
  - Optimized for read-heavy workloads (BI, reporting)
  - Historical data analysis
- Use Cases:
  - Business intelligence, reporting, dashboards, analytics.

Understanding Key Concepts in Modern Data Engineering

Data Lake - The Rise of Flexibility

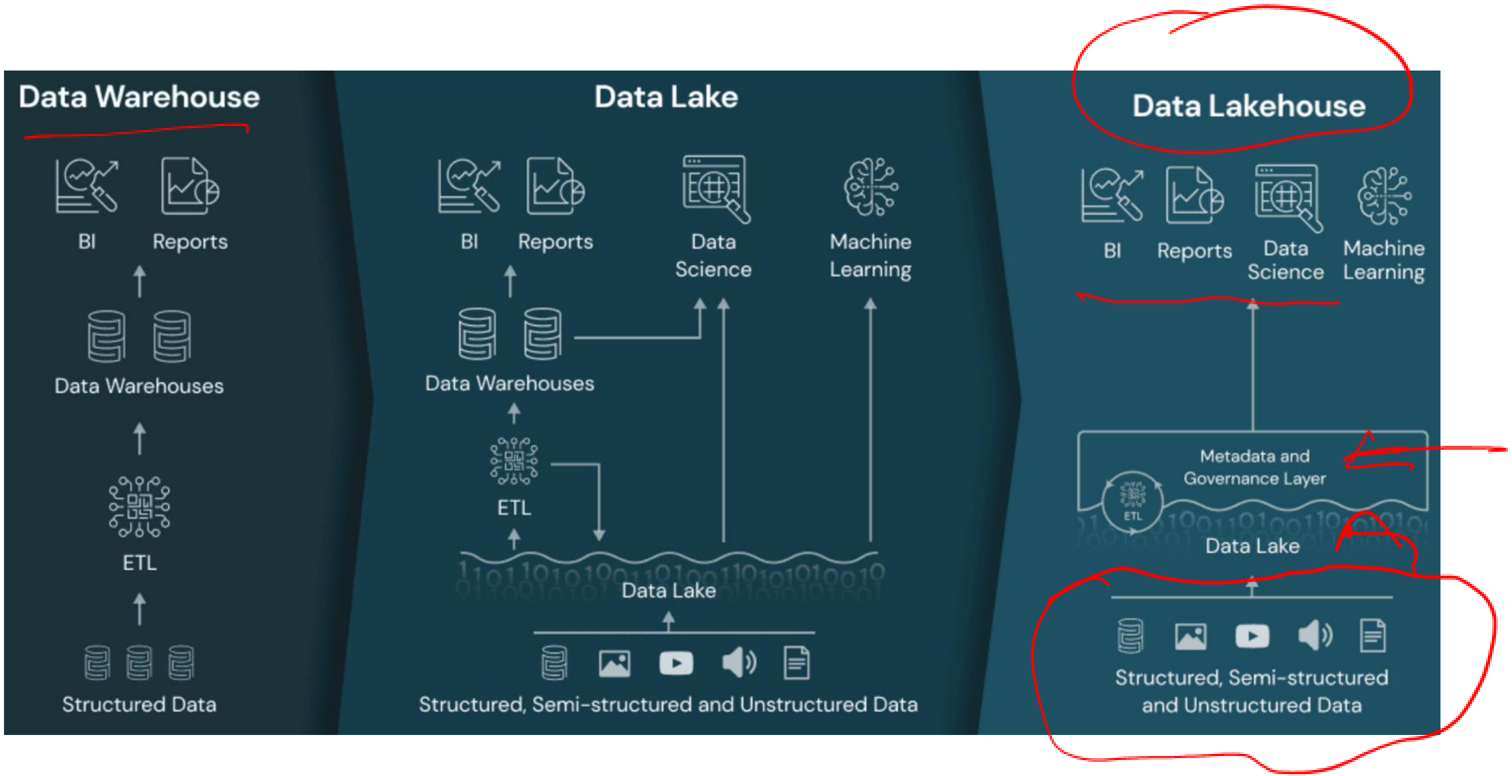
- A centralized repository for storing all types of data, both structured and unstructured, in its native format.
- Key Characteristics:
  - Any data type (structured, semi-structured, unstructured)
  - Schema-on-read (data is modeled when queried)
  - Scalable and cost-effective storage
  - Raw data exploration and discovery
- Use Cases:
  - Data science, machine learning, big data analytics, data exploration.
- Challenges:
  - Data governance, data quality, data discovery can be complex.

Understanding Key Concepts in Modern Data Engineering

Data Lakehouse - Bridging the Gap

- Combines the scalability and flexibility of a data lake with the data management capabilities of a data warehouse.
- Key Characteristics:
  - Supports both structured and unstructured data
  - Enables ACID (Atomicity, Consistency, Isolation, Durability) transactions and data governance
  - ACID properties are crucial for ensuring data integrity and reliability, especially in applications that handle critical data, such as financial transactions, healthcare records, and e-commerce operations
  - Optimized for both read and write workloads
  - Open and interoperable
- Benefits:
  - Faster time to insights, reduced data duplication, improved data quality.

Understanding Key Concepts in Modern Data Engineering



Understanding Key Concepts in Modern Data Engineering

Delta Lake - The Foundation of the Lakehouse

- An open-source storage layer that brings reliability and performance to data lakes.
- Key Characteristics:
  - ACID transactions for data reliability
  - Schema enforcement and evolution
  - Time travel (data versioning)
  - Unified batch and streaming data processing
  - Optimized for Apache Spark
- Benefits:
  - Improves data quality, simplifies data management, enables real-time analytics

https://www.databricks.com/wp-content/uploads/2020/08/p975-armbrust.pdf
https://www.cidrdb.org/cidr2021/papers/cidr2021_paper17.pdf

Understanding Key Concepts in Modern Data Engineering

Data Mesh - Decentralized Data Ownership

- A decentralized approach where different business domains manage their own data as self-service data products.
- Key Features:
  - Domain-oriented design, federated governance, data-as-a-product
- Benefits:
  - Scalability, ownership, agility.
- Example Tools:
  - DataHub, OpenMetadata, Starburst

Understanding Key Concepts in Modern Data Engineering

Data Fabric - Unified Data Access

- An integrated data architecture that enables real-time and secure data access across hybrid and multi-cloud environments.
- Key Features:
  - Unified data access, automation, security.
- Benefits:
  - Real-time insights, improved data governance, simplified data management.
- Example Tools:
  - IBM Data Fabric, Talend, Informatica

Data Engineering

Components and Tools

- **Data Ingestion**
  - Data ingestion is the process of bringing data from one or more data sources into a data platform.
  - Tools - Apache Kafka, Apache NiFi, Amazon Kinesis.
- **Data Transformation**
  - Takes raw ingested data and uses a series of steps (referred to as "transformations") to filter, standardize, clean and finally aggregate it so it's stored in a usable way.
  - A popular pattern is **the medallion architecture**, which defines three stages in the process — Bronze, Silver and Gold.
  - Tools - Apache Spark, Dask.
- **Data Storage:** Solutions - HDFS, S3, Google Cloud Storage, Delta Lake.
- **Data Quality:** Tools - Great Expectations, Apache Griffin.

Data Engineering

Components and Tools

- **Architecture**: Decoupling, modularity, fault tolerance.
- **Orchestration**: Tools - Apache Airflow, Prefect, Apache NiFi.
- **Monitoring**: Tools - Prometheus, Grafana, ELK stack.
- **Case Study**: Real-world scalable DataOps pipeline example.

Data Engineering

Components and Tools

- **Distributed Training:**
  Tools - Horovod, Dask-ML, Spark MLlib.
- **Model Deployment:**
  Kubernetes, KFServing, MLflow. ClearML
- **Monitoring/Retraining:**
  Tools - Evidently AI, Prometheus, Grafana, CI/CD with Jenkins. hub Actions

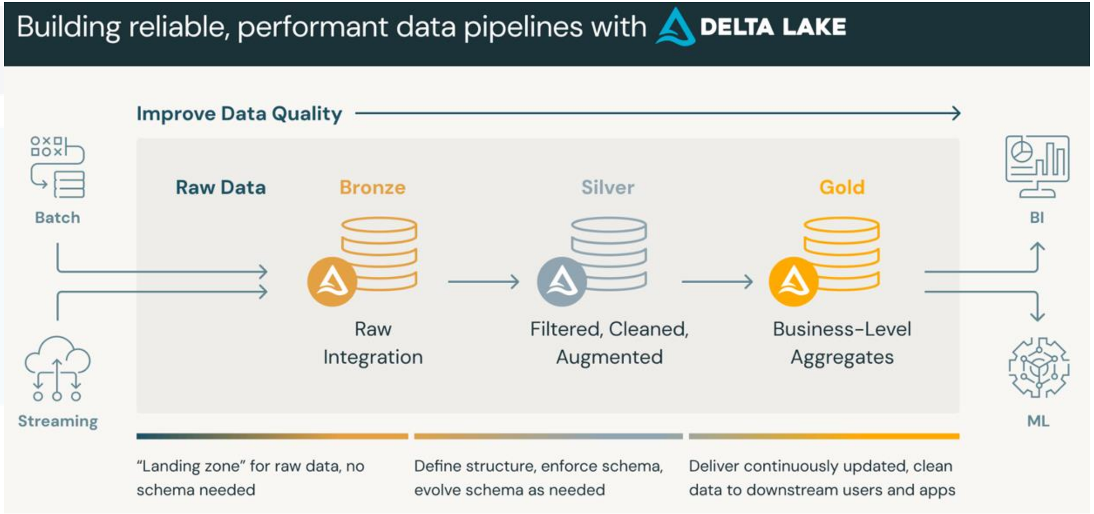
Data Engineering

Orchestration

Data Engineering

What is a medallion architecture?

- A data design pattern used to logically organize data in a lakehouse, with the goal of incrementally and progressively improving the structure and quality of data
- Fows through different layers, from Bronze ⇒ Silver ⇒ Gold layer tables
- Also referred to as "multi-hop" architectures.



Source: databricks

Data Engineering – An Example

Event Streaming

- **Real-Time Processing**: Event streaming includes the manipulation, processing, and real-time reaction to the event streams, enabling immediate responses to data inputs.
- **Retrospective Analysis:** Allows for retrospective processing and analysis of event streams.
- **Data Routing:** Involves routing event streams to different destinations and technologies as required.
- **Continuous Data Flow and Interpretation:** Ensures a consistent flow and interpretation of data, ensuring information is timely and accurately placed.

Data Engineering – An Example

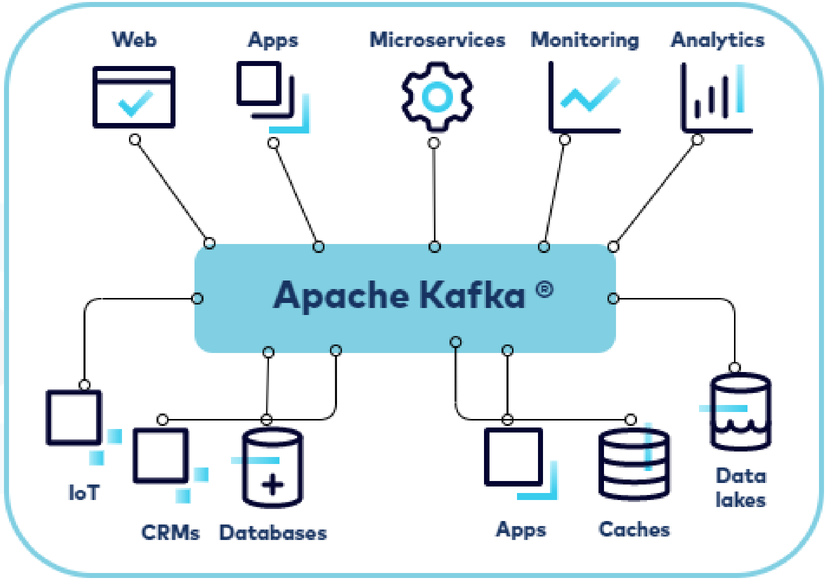
Event Streaming – What for?

- **Financial Transactions:** Used for real-time processing of payments and transactions in sectors like stock exchanges, banking, and insurance.
- **Logistics Tracking:** Applied to track and monitor vehicles in real time, such as cars, trucks, fleets, and shipments, particularly in logistics and the automotive industry.
- **IoT and Sensor Data Analysis:** Utilized to continuously capture and analyze data from IoT devices or equipment in settings like factories and wind parks.
- **Customer Interaction in Retail:** Helps in collecting and responding immediately to customer interactions and orders in retail, hotels, travel industry, and mobile apps.

Data Engineering – An Example

Event Streaming – What for?

- **Healthcare Monitoring:** Employed in hospitals to monitor patients and predict changes in condition for timely treatment in emergencies.
- **Corporate Data Integration**: Used to connect, store, and make data available from different divisions within a company.
- **Foundation for Tech Architecture:** Serves as a base for building data platforms, event-driven architectures, and microservices.



Source: https://docs.confluent.io/kafka/introduction.html