Business intelligence
Systèmes décisionnels
Informatique décisionnelle

Introduction to Big Data

Big Data

Big data refers to data that is so large, fast or complex that it's difficult or impossible to process using traditional methods (the size or type is beyond the ability of traditional relational databases to capture, manage and process the data with low latency). The act of accessing and storing large amounts of information for analytics has been around for a long time. But the concept of big data gained momentum in the early 2000s when industry analyst Doug Laney articulated the now-mainstream definition of big data as the three V's:

- Volume. Organizations collect data from a variety of sources, including transactions, smart (IoT) devices, industrial equipment, videos, images, audio, social media and more. In the past, storing all that data would have been too costly – but cheaper storage using data lakes, Hadoop and the cloud have eased the burden.
- Velocity. With the growth in the Internet of Things, data streams into businesses at an unprecedented speed and must be handled in a timely manner. RFID tags, sensors and smart meters are driving the need to deal with these torrents of data in near-real time.
- Variety. Data comes in all types of formats – from structured, numeric data in traditional databases to unstructured text documents, emails, videos, audios, stock ticker data and financial transactions.

Big Data may come from Web logs, RFID, GPS systems, sensor networks, social networks, Internet-based text documents, Internet search indexes, detailed call records, astronomy, atmospheric science, biological, genomics, nuclear physics, biochemical experiments, medical records, scientific research, military surveillance, photography archives, video archives, and large-scale ecommerce practices.

The importance of big data does not simply revolve around how much data you have. The value lies in how you use it. By taking data from any source and analyzing it.

Big data systems are typically designed to handle large values of any or all of these three parameters, by implementing the following set of core features:

- Distributed processing and storage: A big data system will spread physical storage across several networked locations.
- Data stored without complex structure: Unlike relatively low-scale, (such as relational databases), big data imposes no rigorous schemas or normalization.
- Indefinite scaling: Elastic responses to increased volume, velocity, or variety of data are necessary for any big data system. If there is a scale limit to the system it is essentially failing.

Name	Symbol	Value
Kilobyte	kB	10^3
Megabyte	MB	10^6
Gigabyte	GB	10^9
Terabyte	TB	10^{12}
Petabyte	PB	10^{15}
Exabyte	EB	10^{18}
Zettabyte	ZB	10^{21}
Yottabyte	YB	10^{24}
Brontobyte*	BB	10^{27}
Gegobyte*	GeB	10^{30}

*Not an official SI (International System of Units) name/symbol, yet.

BIG DATA ANALYTICS

Big Data typically refers to data that is arriving in many different forms, be they structured, unstructured, or in a stream. Major sources of such data are clickstreams from Web sites, postings on social media sites such as Facebook, or data from traffic, sensors, or weather. A Web search engine like Google needs to search and index billions of Web pages in order to give you relevant search results in a fraction of a second. Although this is not done in real time, generating an index of all the Web pages on the Internet is not an easy task. Luckily for Google, it was able to solve this problem. Among other tools, it has employed Big Data analytical techniques.

There are two aspects to managing data on this scale: storing and processing. If we could purchase an extremely expensive storage solution to store all the data at one place on one unit, making this unit fault tolerant would involve major expense. An ingenious solution was proposed that involved storing this data in chunks on different machines connected by a network, putting a copy or two of this chunk in different locations on the network, both logically and physically. It was originally used at Google (then called Google File System) and later

developed and released as an Apache project as the Hadoop Distributed File System (HDFS).
However, storing this data is only half the problem. Data is worthless if it does not provide business value, and for it to provide business value, it has to be analyzed. How are such vast amounts of data analyzed? Passing all computation to one powerful computer does not work; this scale would create a huge overhead on such a powerful computer. Another ingenious solution was proposed: Push computation to the data, instead of pushing data to a computing node. This was a new paradigm, and it gave rise to a whole new way of processing data. This is what we know today as the MapReduce programming paradigm, which made processing Big Data a reality. MapReduce was originally developed at Google, and a subsequent version was released by the Apache project called Hadoop MapReduce.
Big Data is associated with almost any kind of large data that has the characteristics of volume, velocity, and variety.

Analytics

- Analytics is the process of discovering, interpreting, and communicating significant patterns in data. . Quite simply, analytics helps us see insights and meaningful data that we might not otherwise detect. **Business analytics** focuses on using insights derived from data to make more informed decisions that will help organizations increase sales, reduce costs, and make other business improvements.
- Some experts use business analytics as a term to describe a set of predictive tools used within the realm of business intelligence.

Descriptive Analytics

Descriptive or reporting analytics refers to knowing what is happening in the organization and understanding some underlying trends and causes of such occurrences. This involves, first of all, consolidation of data sources and availability of all relevant data in a form that enables appropriate reporting and analysis. Usually development of this data infrastructure is part of data warehouses. From this data infrastructure we can develop appropriate reports, queries, alerts, and trends using various reporting tools and techniques. A significant technology that has become a key player in this area is visualization.
Using the latest visualization tools in the marketplace, we can now develop powerful insights into the operations of our organization.

Predictive analytics

Predictive analytics aims to determine what is likely to happen in the future. This analysis is based on statistical techniques as well as other more recently developed techniques that fall under the general category of data mining. The goal of these techniques is to be able to predict if the customer is likely to switch to a competitor

("churn"), what the customer is likely to buy next and how much, what promotion a customer would respond to, or whether this customer is a creditworthy risk. A number of techniques are used in developing predictive analytical applications, including various classification algorithms.

Prescriptive analytics

The third category of analytics is termed prescriptive analytics. The goal of prescriptive analytics is to recognize what is going on as well as the likely forecast and make decisions to achieve the best performance possible. This group of techniques has historically been studied under the umbrella of operations research or management sciences and has generally been aimed at optimizing the performance of a system. The goal here is to provide a decision or a recommendation for a specific action. These recommendations can be in the forms of a specific yes/no decision for a problem, a specific amount (say, price for a specific item or airfare to charge), or a complete set of production plans. The decisions may be presented to a decision maker in a report or may directly be used in an automated decision rules system (e.g., in airline pricing systems). Thus, these types of analytics can also be termed **decision or normative analytics**.

Visual Analytics

Visual analytics is the use of sophisticated tools and processes to analyze datasets using visual representations of the data. Visualizing the data in graphs, charts, and maps helps users identify patterns and thereby develop actionable insights. These insights help organizations make better, data-driven decisions.

Sometimes confused with data visualization, visual analytics isn't simply a matter of representing data graphically. Modern, interactive visual analytics makes it easy to combine data from multiple sources and deeply analyze the data directly within the visualization itself. Plus, AI and machine learning algorithms can offer recommendations to help guide your exploration. Ultimately, visual analytics helps you turn massive data sets into business insights which can have a major positive impact on your organization.

Overview of some of the most popular big data technologies:

Big data technologies are the software tools used to manage all types of datasets and transform them into business insights.

- **Hadoop** is designed to distribute and process vast amounts of data across a cluster of commodity hardware. Its core architecture consists of two primary components: Hadoop Distributed File System (HDFS) and the MapReduce programming model.
    - **Hadoop Distributed File System** (HDFS): HDFS is a distributed file system that provides high availability, fault tolerance, and

scalability. It divides large files into smaller blocks and replicates them across nodes in the cluster, ensuring data redundancy and reliability.

  - **MapReduce**: MapReduce is a programming model and processing engine that enables distributed processing of data across the cluster. It breaks down complex tasks into smaller sub-tasks, which are processed in parallel by map and reduce functions. This approach facilitates efficient computation and analysis of large datasets.
  - **The Hadoop ecosystem** consists of a wide array of tools and frameworks that extend its capabilities for various data processing and analytics tasks.
    - SQL on Hadoop (Hive and Pig)
    - computational models (MapReduce and Tez)
    - real-time processing tools (Storm and Spark Streaming)
    - databases (HBase and Cassandra)
    - streaming ingestion tools (Kafka and Flume)
    - data integration tools (Sqoop and Talend)
    - workflow coordination tools (Oozie and Control M for Hadoop).
    - distributed service coordination tools (Zookeeper)
    - cluster administration tools (Ranger and Sentry)
    - user interface tools (Hue and Jupyter)
    - content indexing tools (ElasticSearch and Splunk)
    - distributed file systems (HDFS)
    - resource managers (YARN and MESOS)
- **Spark** is a fast and general-purpose cluster computing system. Spark provides an interactive shell that can be used for ad-hoc data analysis, as well as APIs for programming in Java, Python, and Scala. Spark also supports SQL queries and machine learning algorithms.
- **NoSQL databases** are designed for scalability and flexibility, making them well-suited for storing big data. The most popular NoSQL database systems include MongoDB, Cassandra, and HBase.
- **Data warehouses** are traditional relational database management systems (RDBMS) that have been enhanced with architectural changes and added functionality to support big data analytics. The two most popular data warehouse systems are Teradata and Oracle Exadata.
- **Databricks** is a cloud-based platform specializing in analyzing data at scale regardless of its location. It is a data and analytics platform that helps enterprises extract business intelligence from the data. It also provides a complete data science workspace with its machine learning runtime,

Managed MLflow, and collaborative notebooks. The platform has support for multiple languages.

Features of HDFS

There are several features that make HDFS particularly useful, including:

- **Data replication**. This is used to ensure that the data is always available and prevents data loss. For example, when a node crashes or there is a hardware failure, replicated data can be pulled from elsewhere within a cluster, so processing continues while data is recovered.
- **Fault tolerance and reliability**. HDFS' ability to replicate file blocks and store them across nodes in a large cluster ensures fault tolerance and reliability.
- **High availability**. As mentioned earlier, because of replication across notes, data is available even if the NameNode or a DataNode fails.
- **Scalability**. Because HDFS stores data on various nodes in the cluster, as requirements increase, a cluster can scale to hundreds of nodes.
- **High throughput**. Because HDFS stores data in a distributed manner, the data can be processed in parallel on a cluster of nodes. This, plus data locality (see next bullet), cut the processing time and enable high throughput.
- **Data locality**. With HDFS, computation happens on the DataNodes where the data resides, rather than having the data move to where the computational unit is. By minimizing the distance between the data and the computing process, this approach decreases network congestion and boosts a system's overall throughput.

Data lakehouse vs. data lake vs. data warehouse [24]

- **Data warehouses: Data** warehouses provide fast access to data and SQL compatibility for business users that need to generate reports and insights for decision-making. All data must go through ETL (extract, transform, load) phase. This means it is optimized in a specific format, or schema, based on the use case before it is loaded to support high-performance queries and data integrity. However, this approach limits the flexibility of access to the data and creates additional costs if data needs to be moved around for future use.
- **Data lakes: Data** lakes store large amounts of unstructured and structured data in its native format. Unlike data warehouses, data is processed, cleaned

up, and transformed during analysis to enable faster loading speeds, making them ideal for big data processing, machine learning, or predictive analytics. However, they require expertise in data science, which limits the set of people who can use the data, and if they're not properly maintained, data quality can deteriorate over time. Data lakes also make it more challenging to get real-time queries as the data is unprocessed, so it still potentially needs to be cleaned, processed, ingested, and integrated before it can be used.

- **Data lakehouse :** A data lakehouse is a modern data architecture that creates a single platform by combining the key benefits of data lakes (large repositories of raw data in its original form) and data warehouses (organized sets of structured data). A data lakehouse merges these two approaches to create a single structure that allows you to access and leverage data for many different purposes, from BI to data science to machine learning. In other words, a data lakehouse captures all of your organization's unstructured, structured, and semi-structured data and stores it on low-cost storage while providing the capabilities for all users to organize and explore data according to their needs.

Batch processing vs stream processing

Batch processing and stream processing are two very different models for processing data. While batch processing is about processing large volumes of data at scheduled intervals, stream processing is all about handling data on-the-fly, in real time, or near-real-time.

Batch processing

Involves processing large volumes of data at once, at scheduled intervals. In contrast, stream processing involves continuously processing data in real time as it arrives.

Batch processing is a data processing technique where data is collected, processed, and stored in predefined chunks or batches over a period of time. Instead of handling data immediately as it arrives, batch processing waits for a certain amount of data or for a scheduled time to process it all at once.

Stream processing is a data processing paradigm that involves handling data in real-time as it arrives or is generated. Instead of waiting for data to accumulate in batches, stream processing systems continuously process data as it flows, enabling immediate insights and actions.

This approach is well-suited for tasks that require real-time analytics, monitoring, and decision-making, such as fraud detection, live dashboard updates, social media sentiment analysis, and IoT (Internet of Things) data processing.
Stream processing systems are designed to handle high-velocity data streams, ensuring low latency and rapid processing.

Stream processing flips the data processing paradigm – from store then process (as in batch process) to process then store if needed.

Examples of streaming data [22]

- Sensors in transportation vehicles, industrial equipment, and farm machinery send data to a streaming application. The application monitors performance, detects any potential defects in advance, and places a spare part order automatically preventing equipment down time.
- A financial institution tracks changes in the stock market in real time, computes value-at-risk, and automatically rebalances portfolios based on stock price movements.
- A real-estate website tracks a subset of data from consumers' mobile devices and makes real-time property recommendations of properties to visit based on their geo-location.
- A solar power company has to maintain power throughput for its customers, or pay penalties. It implemented a streaming data application that monitors of all of panels in the field, and schedules service in real time, thereby minimizing the periods of low throughput from each panel and the associated penalty payouts.
- A media publisher streams billions of clickstream records from its online properties, aggregates and enriches the data with demographic information about users, and optimizes content placement on its site, delivering relevancy and better experience to its audience.
- An online gaming company collects streaming data about player-game interactions, and feeds the data into its gaming platform. It then analyzes the data in real-time, offers incentives and dynamic experiences to engage its players.

Tools for Processing Streaming Data
Kafka, Apache Spark, Flink, Samza, Storm, Apex, Flume

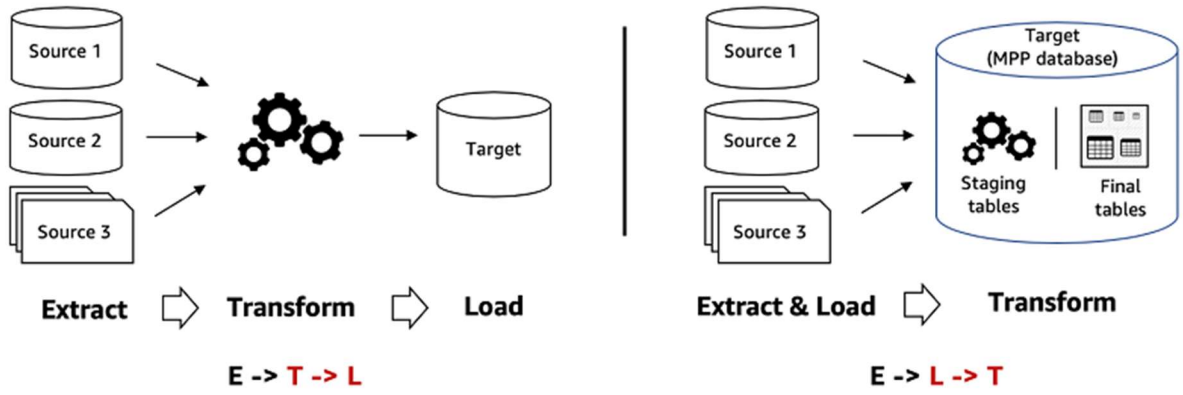
Data pipeline [21]

A data pipeline is a method in which raw data is ingested from various data sources and then ported to data store, like a data lake or data warehouse, for

analysis. This dataflow can also incorporate data transformation, such as filtering, masking, and aggregations, which ensure appropriate data integration and standardization, to convert your raw data into a required format suitable for storing and analysis.
ETL process is a subset of a data pipeline. A data pipeline might not include the transformation phase that an ETL pipeline usually includes before storing the data in the destination.
Depending on requirement and the kind of data, there are different types of data pipelines, including batch processing, streaming

- Batch Processing Pipelines: These types of data pipelines are used to transfer a large amount of data in batches, recurring regular periods. These types of pipelines are generally used for traditional analytics—using gathered data (historical data) for decision-making.
- Real-Time/Streaming Data Pipelines: These types of data pipelines are used to draw insights from constantly flowing data in near real-time. a streaming pipeline continuously ingests changing data, and updates metrics, reports, and summary statistics in response to each available event. Such types of data pipelines are generally used to deal with live-streaming or fluctuating data, like in financial stock markets.

ELT (extract, load and transform)



ETL has three steps:

- Extract raw data from various sources
- Use a secondary processing server to transform that data
- Load that data into a target database

The transformation stage ensures compliance with the target database's structural requirements. You only move the data once it is transformed and ready.

ELT process has three steps:

- Extract raw data from various sources
- Load it in its natural state into a data warehouse or data lake
- Transform it as needed while in the target system

With ELT, all data cleansing, transformation, and enrichment occur within the data warehouse. You can interact with and transform the **raw** data as many times as needed.

The ELT loads **raw** data directly into the target data warehouse. Once there, you can transform the data whenever you need it into the format you require.

This enables **faster** loading times. Since the data is not processed on entry to the data lake, the query and schema do **not need** to be defined a priori (although often the schema will be available during load since many data sources are extracts from databases or similar structured data systems and hence have an associated schema).

ELT Tools
Hevo Data, Luigi, Blendo, Matillion, Talend, Stitch, DBT, Xplenty, ...etc.

NoSQL Database

Traditional relational databases were not designed to cope with the scalability and flexibility challenges facing modern applications.
NoSQL Database is a non-relational Data Management System that does not require a fixed schema. They scale out horizontally, often across commodity servers. Data is partitioned and replicated across these machines, or nodes, providing redundancy and fault tolerance. These types of databases are optimized specifically for applications (Twitter, Facebook and Google) that require flexible data models, large data volume, and low latency, which are achieved by relaxing some of the data consistency restrictions of relational databases.
NoSQL databases provide a **new paradigm for storing big data** by addressing some of these limitations.

- Schema Flexibility: Unlike relational databases, which require a predefined schema, NoSQL databases are schema-less or schema-flexible. This means you can store data without needing to define a rigid structure beforehand. This is particularly beneficial when dealing with unstructured or semi-structured data.

- Scalability: NoSQL databases are designed to scale out horizontally, which means you can add more servers to your database cluster as your data grows. This allows for seamless scalability to handle big data without significant performance degradation.
- High Availability and Fault Tolerance: NoSQL databases are built to ensure high availability and fault tolerance. They often use distributed architectures and replication to prevent data loss in case of hardware failures.
- Query Performance: NoSQL databases are optimized for specific use cases and types of queries. For certain workloads, like real-time analytics or geospatial data, NoSQL databases can outperform traditional relational databases.

NoSQL databases use a variety of data models for managing and accessing the data. The common NoSQL database categories:

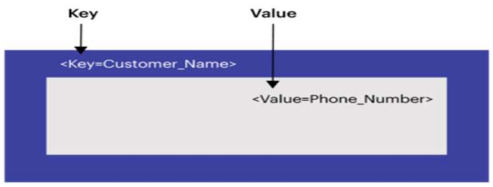
- Key-value pair
- Document-oriented
- Wide Column
- Graph-based

Time seriesOver time, **four** major types of NoSQL databases emerged: document databases, key-value databases , wide-column stores, and graph databases.

- Key-value Database : Examples: Redis, Amazon Dynamo DB
  This type of NoSQL database implements a hash table to store unique keys along with the pointers to the corresponding data values. The values can be of scalar data types such as integers or complex structures such as JSON, lists, BLOB, and so on.

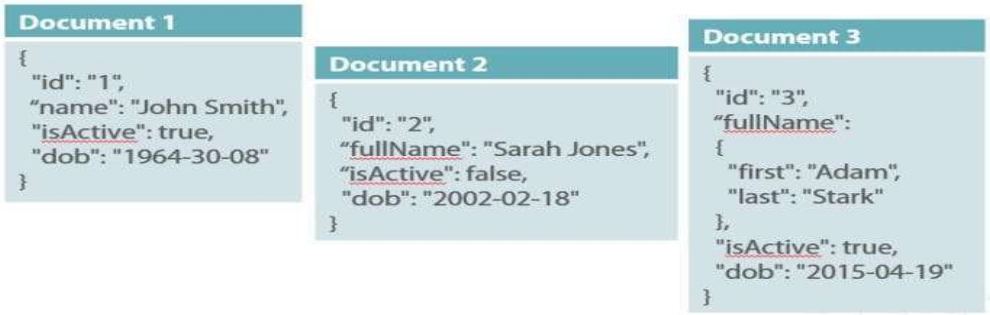
Phone directory

Key	Value
Paul	(091) 9786453778
Greg	(091) 9686154559
Marco	(091) 9868564334

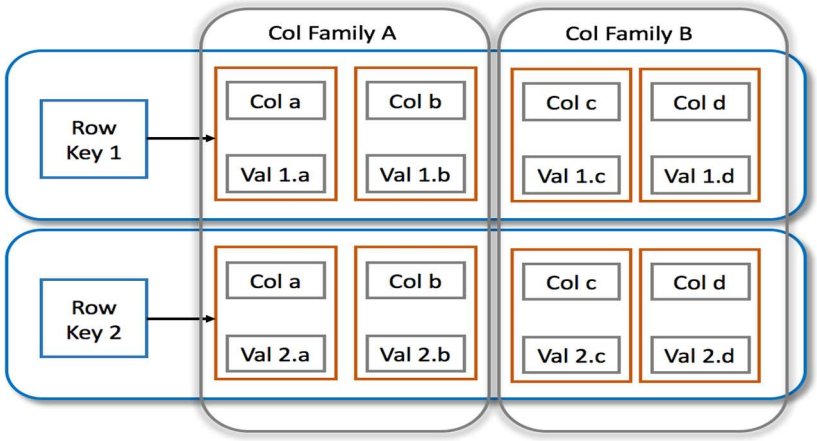


- Document-oriented database: Examples: Mongo DB, Apache SOLR, Elasticsearch
  A document is a record in a document database. A document typically stores information about one object and any of its related metadata.

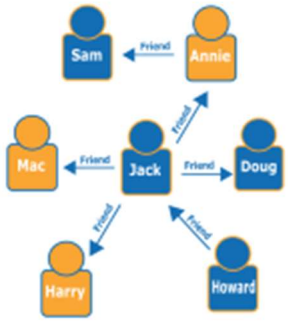
Documents store data in field-value pairs. The values can be a variety of types and structures, including strings, numbers, dates, arrays, or objects. Documents can be stored in formats like JSON, BSON, and XML.



- Wide-column database: Example: Google's BigTable, Cassandra, HBase.
  Data is **stored in columns** instead of rows as in a conventional relational database management system (RDBMS). Related columns can be grouped together to form column families. Each column family typically contains multiple columns that are used together, like traditional RDBMS tables. Within a given column family, all data is stored in a row-by-row fashion, such that the columns for a given row are stored together, rather than each column being stored separately. Each row in the column family has a unique identifier key, and then as many columns as it needs to hold the information. There is no requirement for every row to have the same columns



- Graph database
  A typical NoSQL graph database is made up of four essential components: nodes, edges, properties and labels.



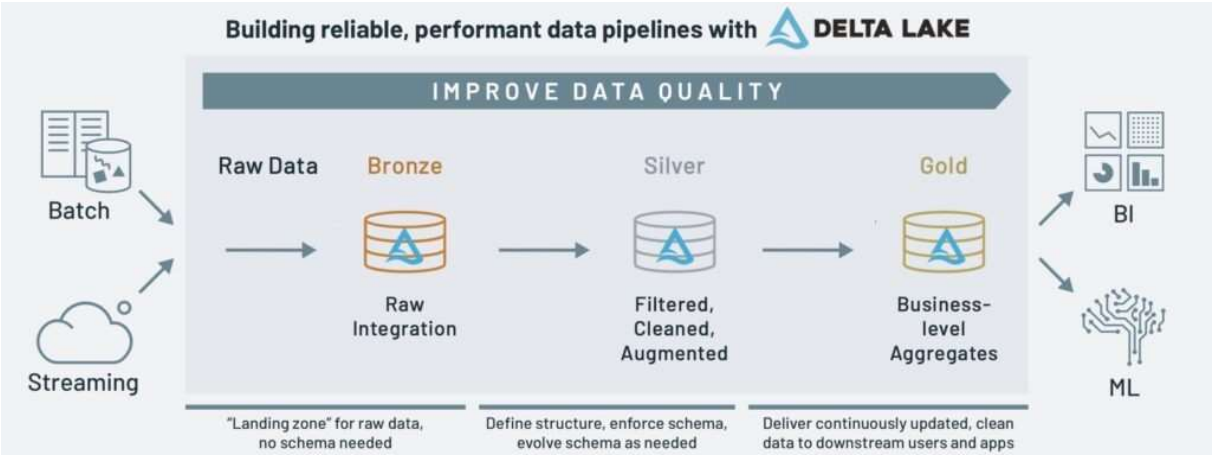
Data Processing Architectures

Data processing architectures are systems designed to efficiently handle the ingestion, processing, and storage of large amounts of data. These architectures play a crucial role in modern businesses. They allow organizations to analyze and extract valuable insights from their data, which can be used to improve decision-making, optimize operations, and drive growth.

Medallion Architecture

A medallion architecture is a data design pattern, coined by Databricks, used to logically organize data in a lakehouse, with the goal of incrementally improving the quality of data as it flows through various layers.

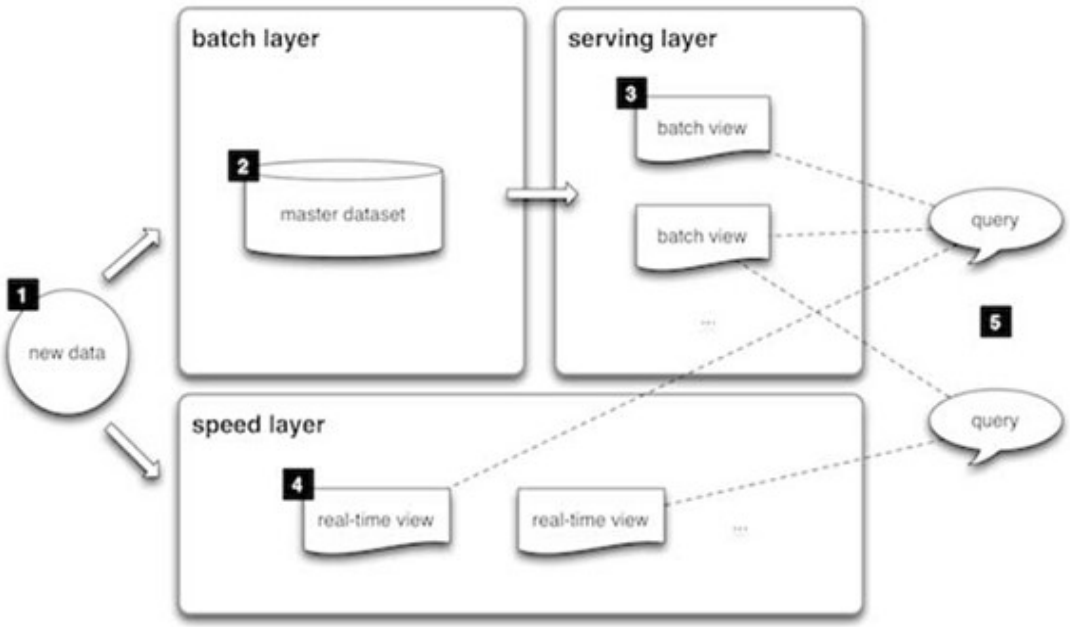
This architecture consists of three distinct layers – bronze (raw), silver (validated) and gold (enriched) – each representing progressively higher levels of the quality of data stored in the lakehouse. Medallion architectures are sometimes referred to as "multi-hop" architectures.



Lambda Architecture

Lambda architecture is a way of processing massive quantities of data (i.e. "Big Data") that provides access to batch-processing and stream-processing

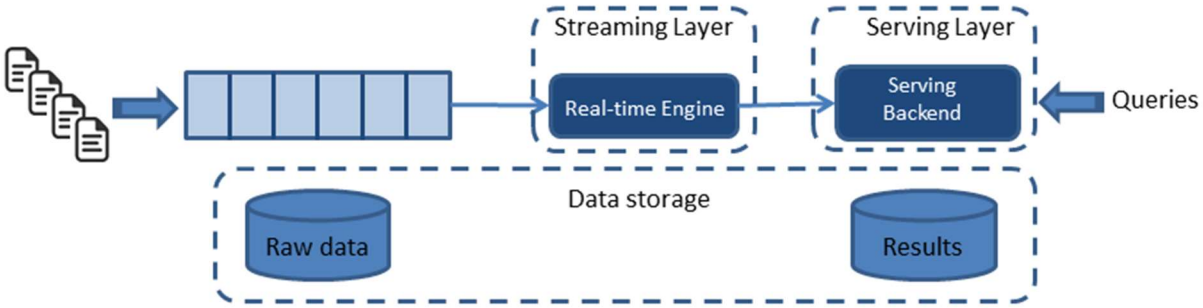
methods with a hybrid approach. The lambda architecture itself is composed of 3 layers:



Lambda architecture is a software architecture deployment pattern where incoming data is fed both to batch and streaming (speed) layers in parallel. The batch layer feeds the data into the data lake and data warehouse, applies the compute logic, and delivers it to the serving layer for consumption. The streaming layer makes use of the previous insights that are derived in the batch layer for processing new incoming data. It is important to note that Lambda architecture requires a separate batch layer along with a streaming layer (or fast layer) before the data is being delivered to the serving layer.

Kappa architecture

The Kappa Architecture is event-based and only has the Streaming Layer and the Serving Layer, so every kind of data that needs to be processed will be handled by a single technology stack. This paradigm solves the redundancy in Lambda architectures, avoiding the maintenance of two different code bases to feed the layers, as well as the multi-layered structure. Kappa architecture is not a substitute for Lambda architecture. It is, in fact, an alternative approach for data management within the organization.


Streaming Layer
Serving Layer
Real-time Engine
Serving Backend
Queries
Data storage
Raw data
Results