LECTURE NOTES

UNIT – I

Big Data Analytics: Define Big Data, Affects of Big Data on our daily lives, Data Sizes, Source of BigData, Challenges of Big Data, 5 V's of Big Data, Types of Digital Data, Structured/Unstructured data-Advantages and Sources, Business Intelligence, Comparison business intelligence from traditional big data,Architecture of data warehouse, Analytical tools used for big data analytics, Hadoop environment and itscomponents**.**

Define Big Data

Big Data refers to vast and rapidly growing volumes of data that are too large and complex for traditional data processing tools to manage. This data comes in many forms structured (e.g., tables), semi-structured (e.g., JSON, XML), and unstructured (e.g., text, images, video). With the explosion of devices, sensors, online services, and digital platforms, data is now generated at an unprecedented rate. This growth makes it essential for organizations to adopt advanced tools and technologies to capture, store, analyze, and utilize this data effectively.

Affects of Big Data on our daily lives

Healthcare: A life-saving application

In healthcare, big data is a game-changer. Hospitals and research facilities analyze vast amounts of data from medical records, wearables, and genomics to personalize patient care. This approach, known as precision medicine, tailors treatment to the individual's genetic makeup, lifestyle, and environment. Studies have shown that using big data in healthcare can significantly improve outcomes by identifying risk factors, predicting diseases, and customizing treatment plans.

Retail: Understanding consumer behavior

The retail industry thrives on understanding consumer behavior, and big data provides that insight on a silver platter. Retail giants analyze customer data to forecast trends, manage inventory, and tailor marketing strategies. This data-driven approach leads to personalized shopping experiences with recommendations that seem to read your mind. The results? Increased customer satisfaction and loyalty.

Banking: Security and personalization

In banking, big data enhances security and offers personalized financial advice. Banks can detect fraudulent activity by analyzing transaction data, preventing potential losses. Additionally, they use big data to understand customer behavior, offering personalized financial products that meet individual needs. This improves customer experience and boosts the bank's efficiency and profitability.

Transportation: Smoothing your commute

The transportation sector uses big data to optimize routes, reduce traffic congestion, and improve safety. Navigation apps analyze real-time traffic data, suggesting the fastest routes to your destination. Public transportation systems use big data to predict peak times and adjust

schedules accordingly. This makes your commute smoother and helps reduce carbon emissions by optimizing public transport usage.

Agriculture: Feeding the future

In agriculture, big data is revolutionizing the way we grow food. Farmers use data from satellites, drones, and sensors to make informed decisions about planting, watering, and harvesting. This technology-driven approach, known as precision agriculture, increases crop yields and reduces waste, ensuring a more sustainable food supply for the growing global population.

Entertainment: Curating your content

Streaming services like Netflix and Spotify use big data to curate content tailored to your preferences. By analyzing your viewing or listening history, these platforms recommend shows, movies, and music you'll likely enjoy. This personalization enhances your experience, making it seem like the service knows you better than you know yourself.

Smart Cities: Enhancing urban living

Smart cities employ big data to improve infrastructure, public services, and urban living. Sensors collect data on traffic, air quality, and energy usage, which is then used to optimize public transportation, reduce pollution, and make cities more energy-efficient. This improves the quality of life for residents and helps cities reduce their environmental footprint.

Education: Personalized learning experiences

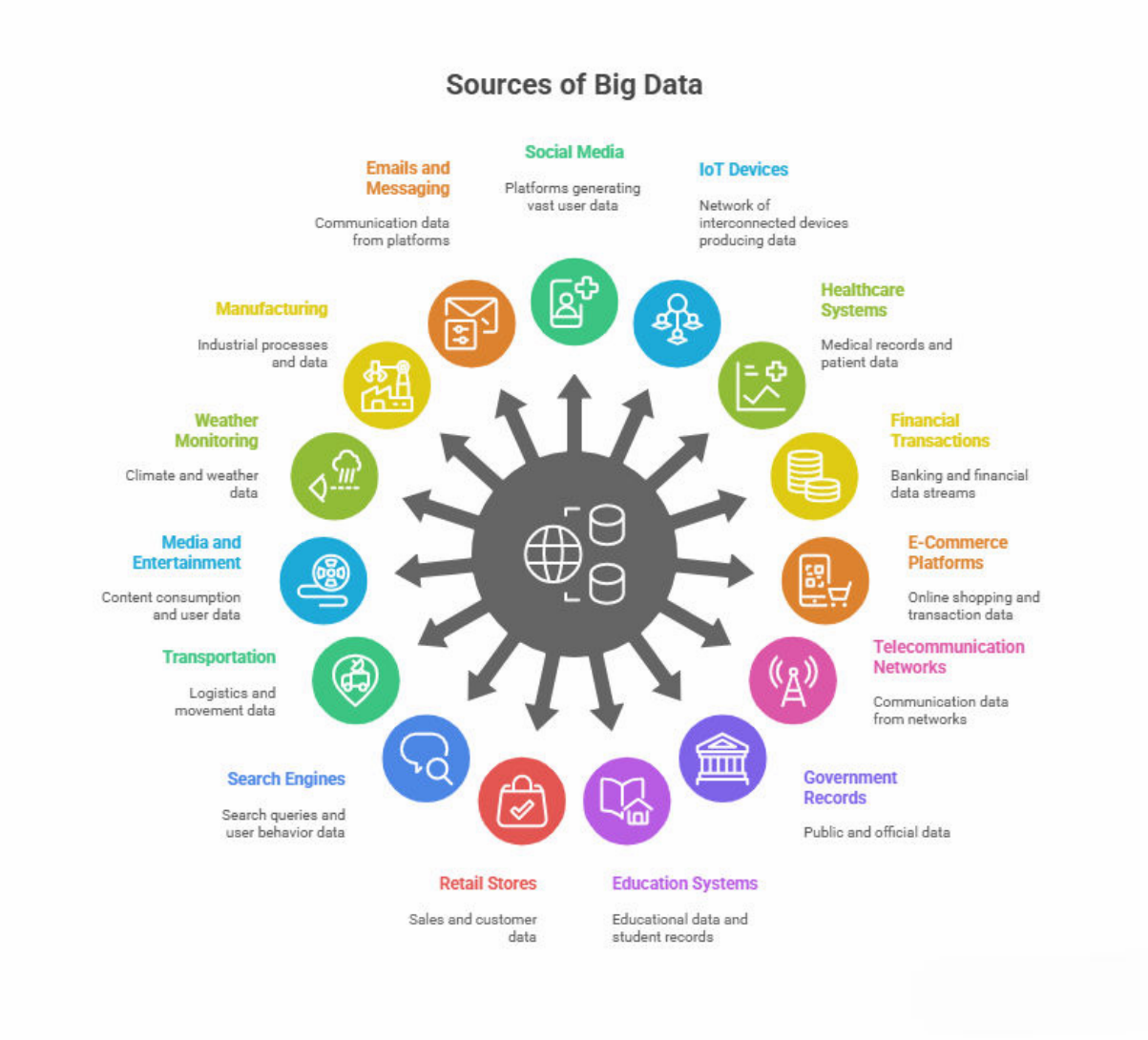
In education, big data is being used to create personalized learning experiences. By analyzing data on students' learning habits and performance, educators can tailor instruction to meet the needs of each student. This approach has been shown to improve student outcomes, as it addresses individual learning styles and paces

Data Size

Data can be defined as a representation of facts, concepts, or instructions in a formalized manner

Accuracy	Is the information correct in every detail?
Completeness	How comprehensive is the information?
Reliability	Does the information contradict other trusted resources?
Relevance	Do you really need this information?
Timeliness	How up- to-date is information? Can it be used for real-time reporting?

Source of Big data:



1. **Social Media Platforms**

Social media platforms generate massive amounts of data continuously. Platforms like Facebook, Instagram, Twitter, LinkedIn, and TikTok record posts, likes, shares, comments, videos, and user profiles.

Example: Over 500 million tweets are posted daily. Businesses analyze this to understand trends, customer behavior, and preferences. Social media is one of the most prominent sources of big data today.

2. **Internet of Things (IoT) Devices**

Connected devices like smartwatches, fitness trackers, smart TVs, home assistants, and connected cars generate constant streams of data. Sensors track locations, activities, and device performance.

Example: A smart fridge monitors food inventory and usage patterns. With billions of connected devices worldwide, IoT is among the world's biggest sources of big data due to its real-time updates.

3. **Healthcare Systems**

Hospitals, clinics, and wearable devices generate terabytes of medical data every day. This includes patient records, diagnostic images, and device readings.

Example: MRI scans and ECG readings are recorded for millions of patients. Healthcare relies on these sources of big data for disease prediction, personalized treatment, and research.

4. **Financial Transactions**

Every online purchase, card swipe, or stock trade produces high-frequency data. Banks and financial institutions track this information for security and analysis.

Example: Visa, Mastercard, and Paytm record millions of transactions daily. Financial data is one of the world's biggest sources of big data because of continuous, high-speed activity.

5. E-Commerce Platforms

Online marketplaces track clicks, searches, reviews, and purchases. This helps businesses offer personalized recommendations.

Example: Amazon monitors which products users view and their browsing time. E-commerce platforms remain key sources of big data for understanding customer behavior.

6. Telecommunication Networks

Telecom providers gather call records, SMS logs, and internet usage data.

Example: Providers track network usage to identify dropped calls and optimize coverage. Telecom data is an important source of big data for infrastructure planning.

7. Government and Public Records

Governments generate huge datasets, including census information, tax records, and vehicle registrations.

Example: India's Aadhaar system records biometric and demographic details for over a billion people. Such records are critical sources of big data for policy-making and planning.

8. Education Systems

Student performance, online course engagement, and learning platform activity generate data continuously.

Example: MOOCs record user progress and participation. Education data is a key source of big data for improving teaching and learning experiences.

9. Retail Stores and Point-of-Sale Data

Retail stores collect information from barcode scans, loyalty programs, and customer footfall.

Example: Walmart processes over 1 million transactions every hour. Retail data is an important source of big data for inventory management and sales prediction.

10. Search Engines

Search engines record billions of queries daily, reflecting user interests and trends.

Example: Google processes over 3.5 billion searches every day. Search data is one of the world's biggest sources of big data due to its volume, speed, and global coverage.

11. Transportation and Logistics

Data from GPS tracking, ride-hailing apps, and airline bookings provide insights for route optimization.

Example: Uber collects ride and location data from millions of users daily. Transport data is a vital source of big data for operational efficiency.

Challenges of Big Data:

The challenges of Big Data are the real implementation hurdles that require immediate attention and need to be addressed to avoid the technology's failure. If not properly handled, these challenges can lead to inefficient data management, poor decision-making, and missed opportunities. Let's discuss some of the most critical challenges related to Big Data.

Data Volume: Managing and Storing Massive Amounts of Data

- Challenge: The most apparent challenge with Big Data is the sheer volume of data being generated. Organizations are now dealing with petabytes or even exabytes of data, making traditional storage solutions inadequate. This vast amount of data requires advanced storage infrastructure, which can be costly and complex to maintain.
- Solution: Adopting scalable cloud storage solutions, such as *Amazon S3, Google Cloud Storage, or Microsoft Azure,* can help manage large volumes of data. These platforms offer flexible storage options that can grow with your data needs. Additionally, implementing data compression and deduplication techniques can reduce storage costs and optimize the use of available storage space.

Data Variety: Handling Diverse Data Types

- Challenge: Big Data encompasses a wide variety of data types, including structured data (e.g., databases), semi-structured data (e.g., XML, JSON), and unstructured data (e.g., text, images, videos). The diversity of data types can make it difficult to integrate, analyze, and extract meaningful insights.
- Solution: To address the challenge of data variety, organizations can employ data integration platforms and tools like Apache Nifi, Talend, or Informatica. These tools help in consolidating disparate data sources into a unified data model. Moreover, adopting schema-on-read approaches, as opposed to traditional schema-on-write, allows for more flexibility in handling diverse data types.

Data Velocity: Processing Data in Real-Time

- Challenge: The speed at which data is generated and needs to be processed is another significant challenge. For instance, IoT devices, social media platforms, and financial markets produce data streams that require real-time or near-real-time processing. Delays in processing can lead to missed opportunities and inefficiencies.
- Solution: To handle high-velocity data, organizations can implement real-time data processing frameworks such as Apache Kafka, Apache Flink, or Apache Storm. These frameworks are designed to handle high-throughput, low-latency data processing, enabling businesses to react to events as they happen. Additionally, leveraging edge computing can help process data closer to its source, reducing latency and improving real-time decision-making.

Data Veracity: Ensuring Data Quality and Accuracy

- Challenge: With Big Data, ensuring the quality, accuracy, and reliability of data—referred to as data veracity—becomes increasingly difficult. Inaccurate or low-quality data can lead to misleading insights and poor decision-making. Data veracity issues can arise from various sources, including data entry errors, inconsistencies, and incomplete data.
- Solution: Implementing robust data governance frameworks is crucial for maintaining data veracity. This includes establishing data quality standards, performing regular data audits, and employing data cleansing techniques. Tools like Trifacta, Talend Data

Quality, and Apache Griffin can help automate and streamline data quality management processes.

What are the Five "Vs" of Big Data?

Traditionally, we've recognized big data by three characteristics: variety, volume, and velocity, also known as the "three Vs." However, two additional Vs have emerged over the past few years: value and veracity.

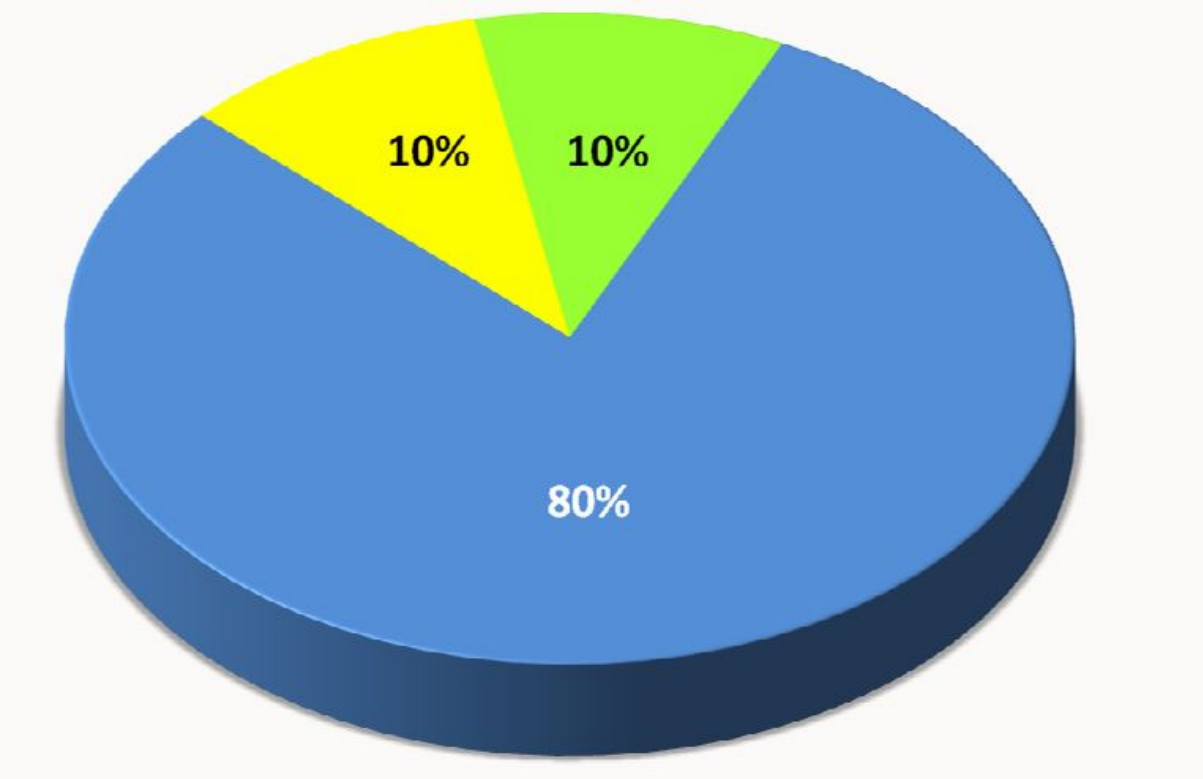
Those additions make sense because today, data has become capital. Think of some of the world's biggest tech companies. Many of the products they offer are based on their data, which they're constantly analyzing to produce more efficiency and develop new initiatives. Success depends on all five Vs.

- **Volume.** The amount of data matters. With big data, you'll have to process high volumes of low-density, unstructured data. This can be data of unknown value, such as X (formerly Twitter) data feeds, clickstreams on a web page or a mobile app, or sensor-enabled equipment. For some organizations, this might be tens of terabytes of data. For others, it may be hundreds of petabytes.
- **Velocity**. Velocity is the fast rate at which data is received and (perhaps) acted on. Normally, the highest velocity of data streams directly into memory versus being written to disk. Some internet-enabled smart products operate in real time or near real time and will require real-time evaluation and action.
- **Variety.** Variety refers to the many types of data that are available. Traditional data types were structured and fit neatly in a relational database. With the rise of big data, data comes in new unstructured data types. Unstructured and semistructured data types, such as text, audio, and video, require additional preprocessing to derive meaning and support metadata.
- **Veracity.** How truthful is your data—and how much can you rely on it? The idea of veracity in data is tied to other functional concepts, such as data quality and data integrity. Ultimately, these all overlap and steward the organization to a data repository that delivers high-quality, accurate, and reliable data to power insights and decisions.
- **Value**. Data has intrinsic value in business. But it's of no use until that value is discovered. Because big data assembles both breadth and depth of insights, somewhere within all of that information lies insights that can benefit your organization. This value can be internal, such as operational processes that might be optimized, or external, such as customer profile suggestions that can maximize engagement.

Digital Data

In fact, the computer and Internet duo has imparted the digital form to data. Digital data can be classified into three forms:– Unstructured– Semi-structured– Structured

Formats of Digital Data

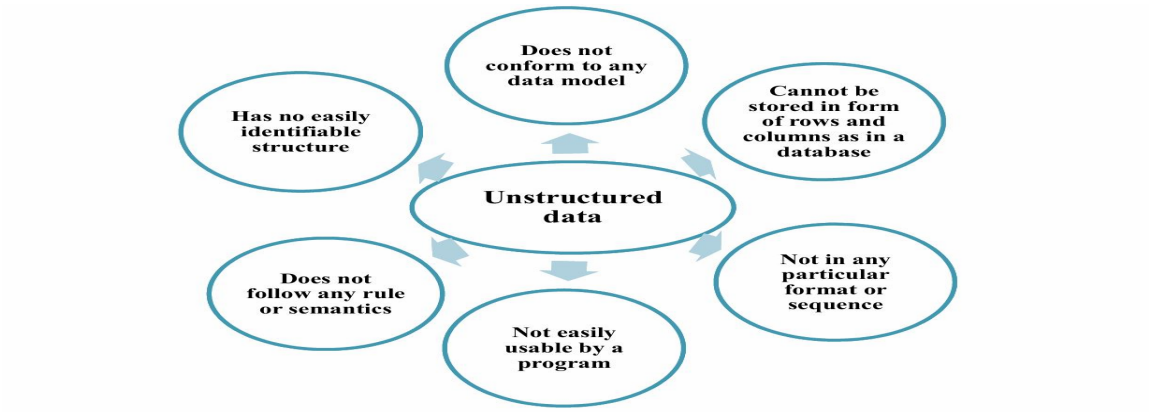


Unstructured data: This is the data which does not conform to a data model or is not in a form which can be used easily by a computer program. About 80—90% data of an organization is in this format; for example, memos, chat rooms, PowerPoint presentations, images, videos, letters, researches, white papers, body of an email, etc.

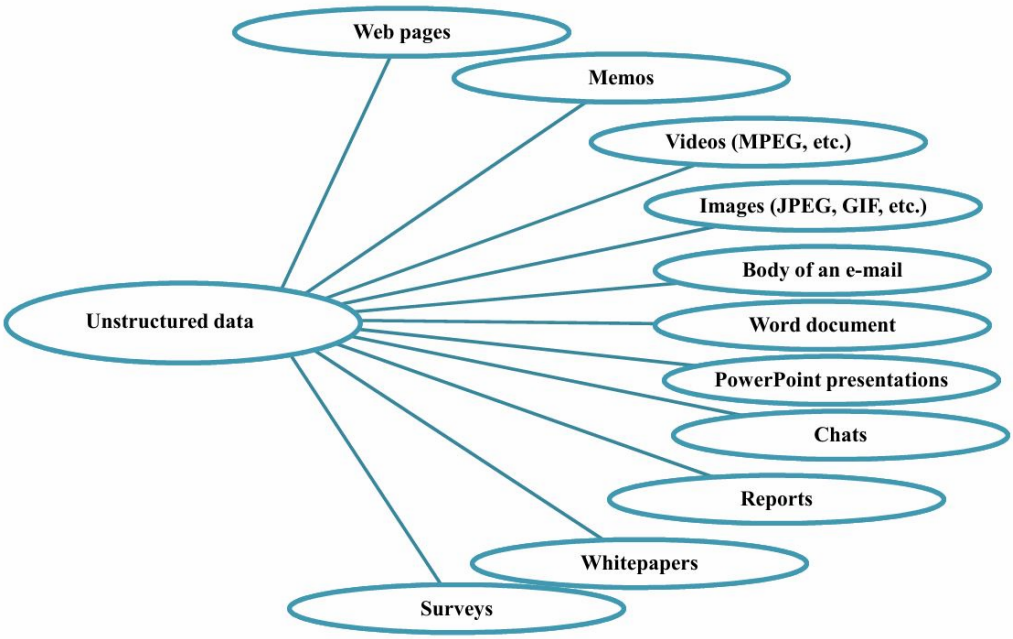
Semi-structured data: This is the data which does not conform to a data model but has some structure. However, it is not in a form which can be used easily by a computer program; for example, emails, XML, markup languages like HTML, etc. Metadata for this

data is available but is not sufficient.
Structured data: This is the data which is in an organized form (e.g., in rows and columns) and can be easily used by a computer program. Relationships exist between entities of data, such as classes and their objects. Data stored in databases is an example of structured data

Characteristics of Unstructured Data



Where does Unstructured Data Come from?



How to Manage Unstructured Data?

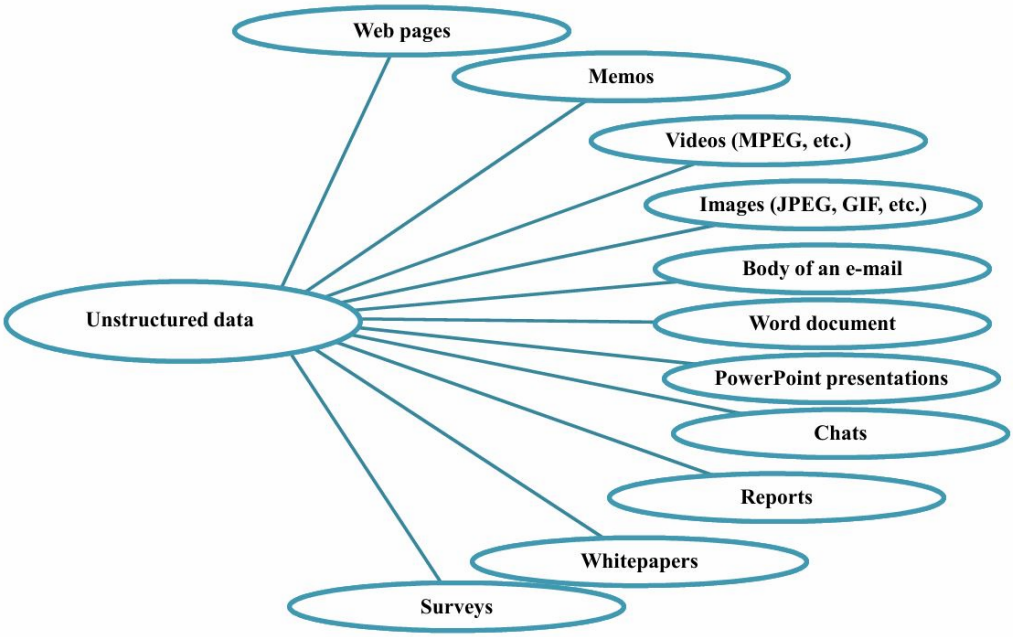
Let us look at a few generic tasks to be performed to enable storage and search of unstructured data:

Indexing: Let us go back to our understanding of the Relational Database Management System(RDBMS). In this system, data is indexed to enable faster search and retrieval. On the basis of some value in the data, index is defined which is nothing but an identifier and represents the large record in the data set. In the absence of an index, the whole data set/ document will be scanned for retrieving the desired information. In the case of unstructured data too, indexing helps in searching and retrieval. Based on text or some other attributes, e.g. file name, the unstructured data is indexed. Indexing in unstructured data is difficult because neither does this data have any predefined attributes nor does it follow any pattern or naming conventions. Text can be indexed based on a text string but in case of non-text based files, e.g. audio/video, etc., indexing depends on file names. This becomes a hindrance when naming conventions are not being followed.

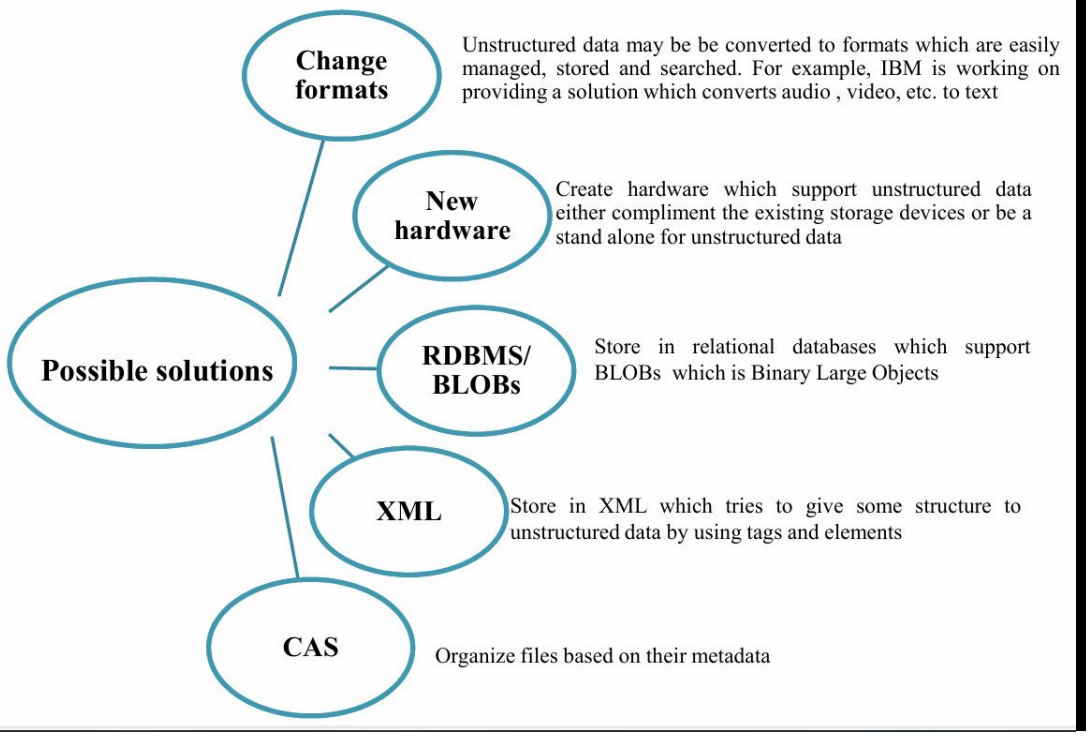
Tags/Metadata: Using metadata, data in a document, etc. can be tagged. This enables search and : Using metadata, data in a document, etc. can be tagged. This enables search and retrieval. But in unstructured data, this is difficult as little or no metadata is available. Structure of data has to be determined which is very difficult as the data itself has no particular format and is coming from more than one source.

Classification/Taxonomy: Taxonomy is classifying data on the basis of the relationships that exist between data. Data can be arranged in groups and placed in hierarchies based on the taxonomy prevalent in an organization. However, classifying unstructured data is difficult asidentifying relationships between data is not an easy task. In the absence of any structure ormetadata or schema, identifying accurate relationships and classifying is not easy. Since the datais unstructured, naming conventions or standards are not consistent across an organization, thusmaking it difficult to classify data.CAS (Content Addressable Storage): It stores data based on their metadata. It assigns 2 uniquename to every object stored in it. The object is retrieved based on its content and not its location.It is used extensively to store emails, etc

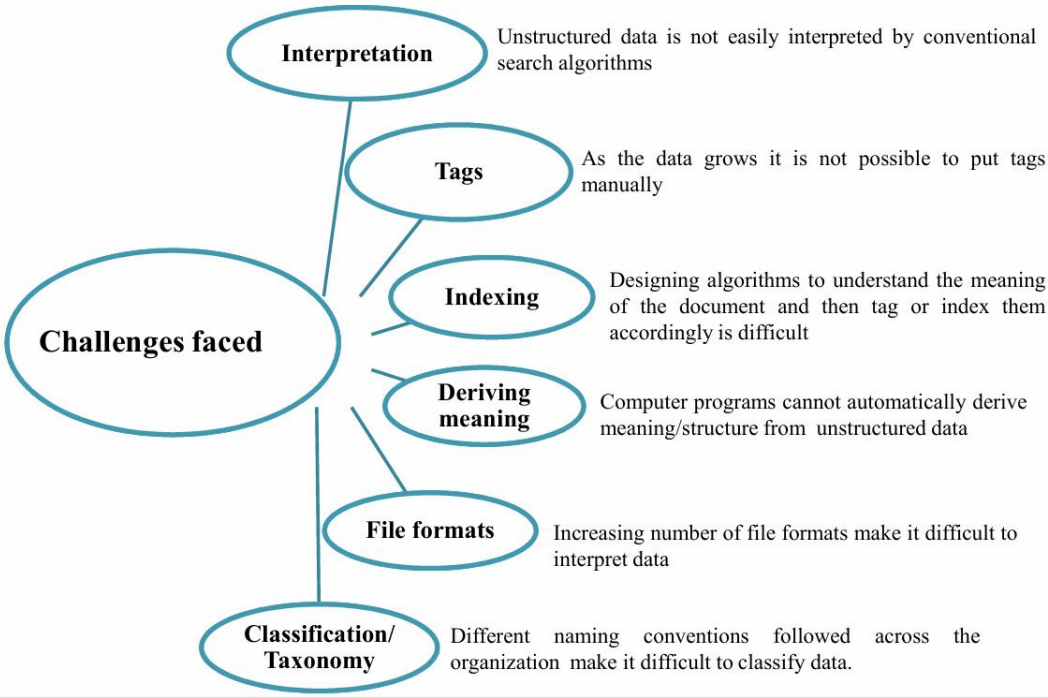
Where does Unstructured Data Come from?



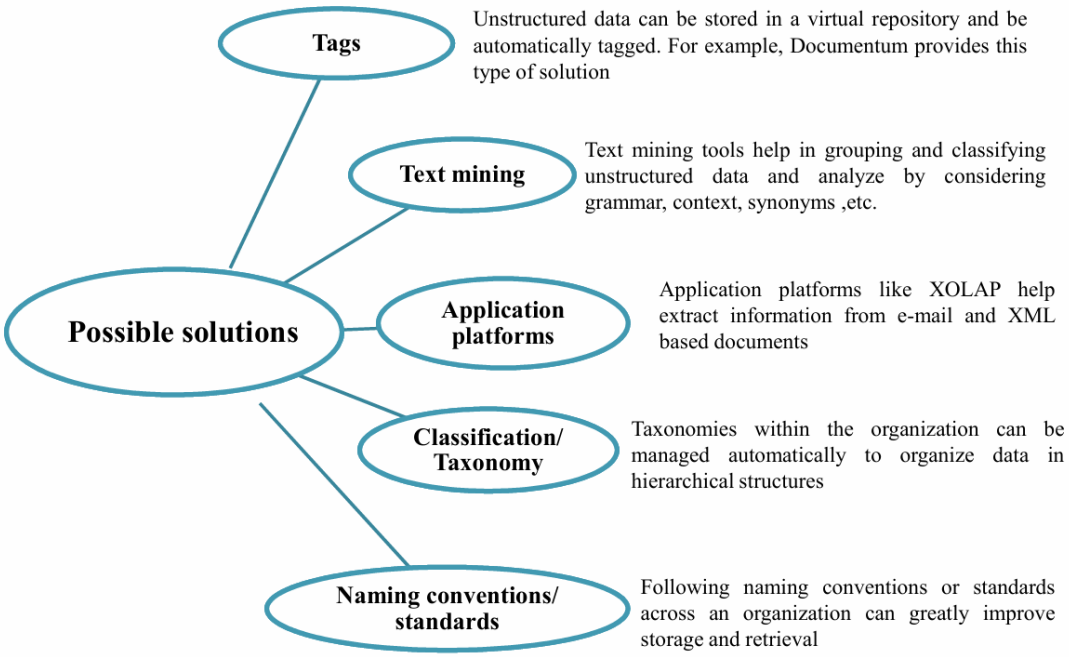
How to Store Unstructured Data?



How to Extract Information from Unstructured Data?



How to Extract Information from Unstructured Data?



UIMA(Unstructured Information Management Architecture)

It is an open source platform from IBM which integrates different kinds of analysis engines to provide a complete solution for edge discovery from unstructured data.

In UIMA, the analysis engines integration and analysis of unstructured information and bridge the gap between structured and unstructured data.

UIMAstores information in a structured format. The structured resources can be mined, searched, and put to other uses.

The information obtained from structured sources is also for sub-sequent analysis of unstructured from structured sources is also for sub-sequent analysis of unstructured data.

Various analysis engines analyze unstructured data in different ways such as:– Breaking up of documents into separate words.

– Grouping and classifying according to taxonomy.

– Detecting parts of speech, grammar, and synonyms.

– Detecting events and times.¢ Detecting relationships between various elements.

Semi-structured Data

• Semi-structured data does not conform to any data model i.e. it is difficult to determine the meaning of data neither can data be stored in rows and columns as in a database but semi-structured data has tags and markers which help to group data and describe how data is stored, giving some metadata but it is not sufficient for management and automation of data.

• Similar entities in the data are grouped and organized in a hierarchy. The attributes or the properties within a group may or may not be the same. For example two addresses may or may not contain the same number of properties as in

Address 1

<house number><street name><area name><city>

Address 2

<house number><street name><city>

• For example an e-mail follows a standard format To:

To: <Name>

From: <Name>

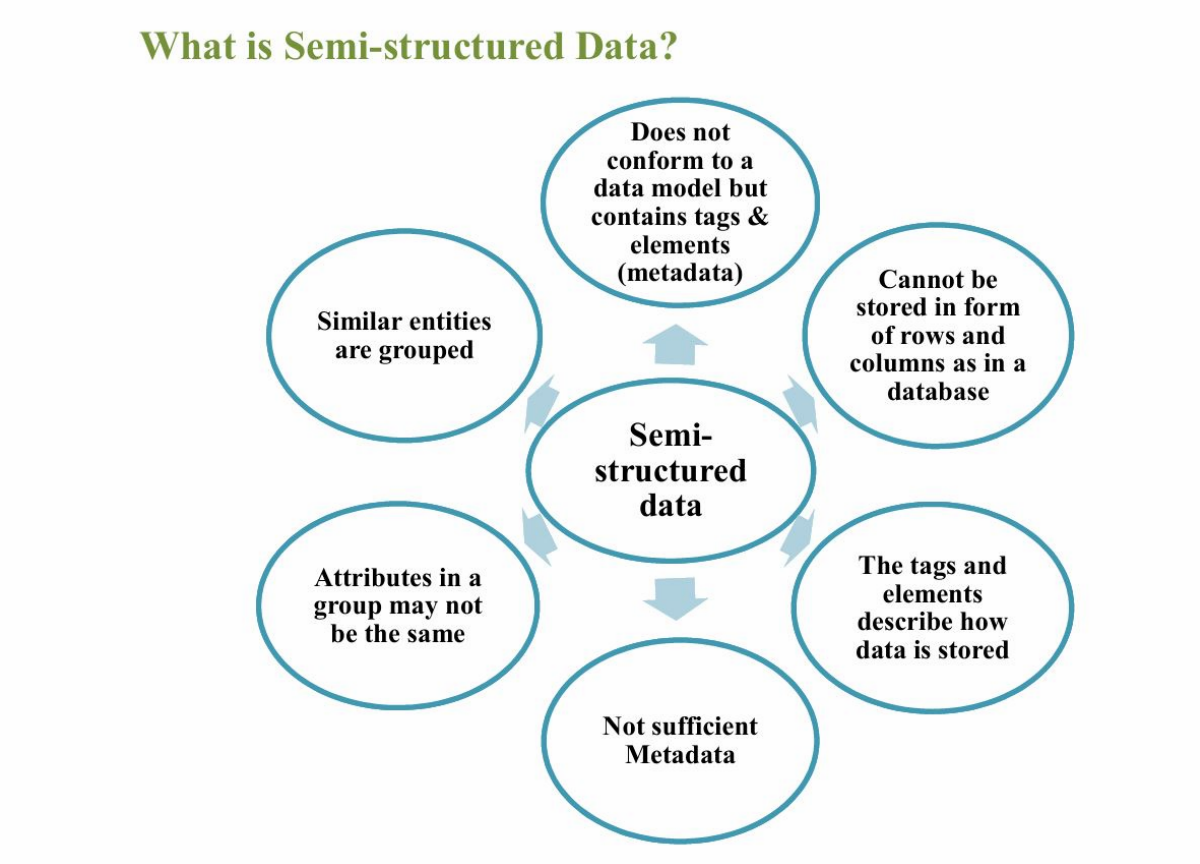
Subject: <Text>

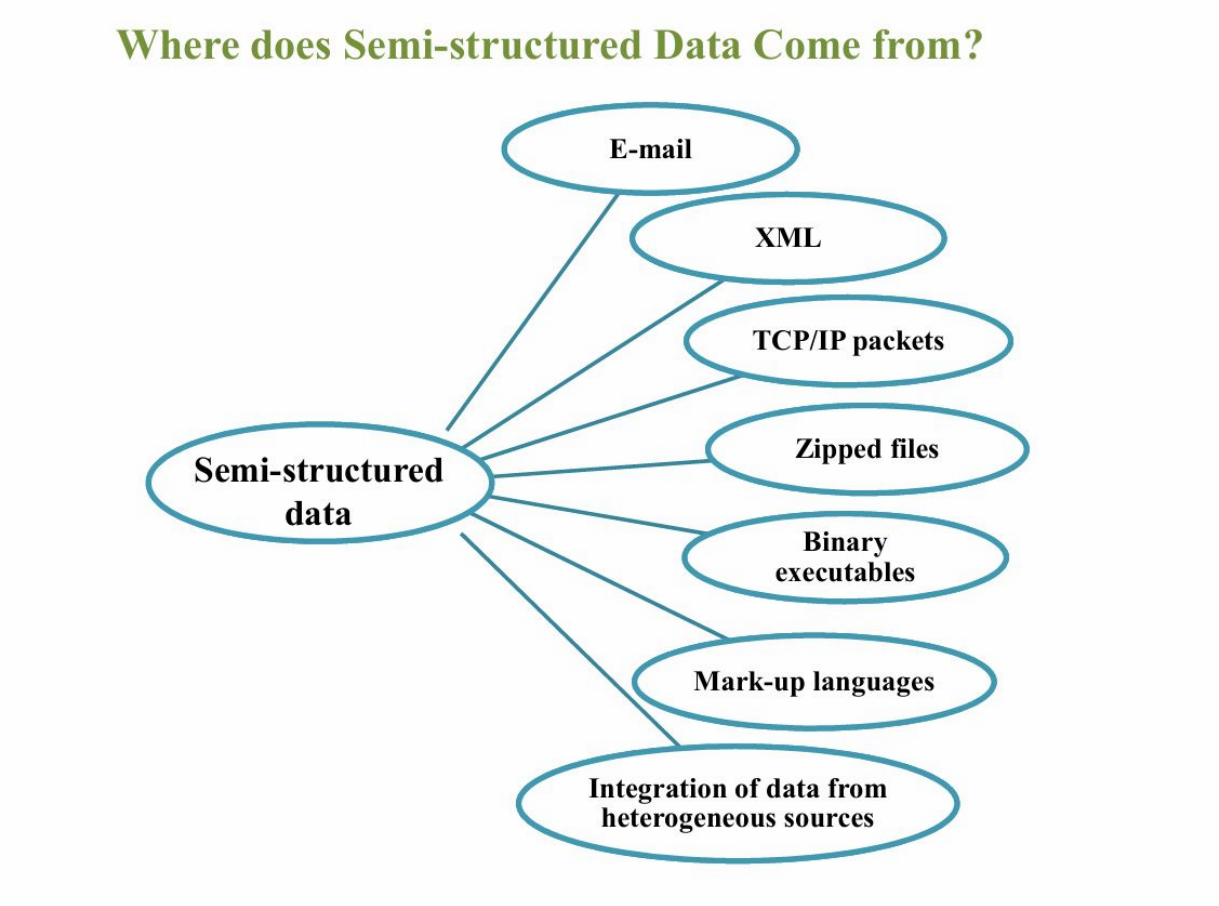
CC: <Name>

Body: <Text, Graphics, Images etc. >

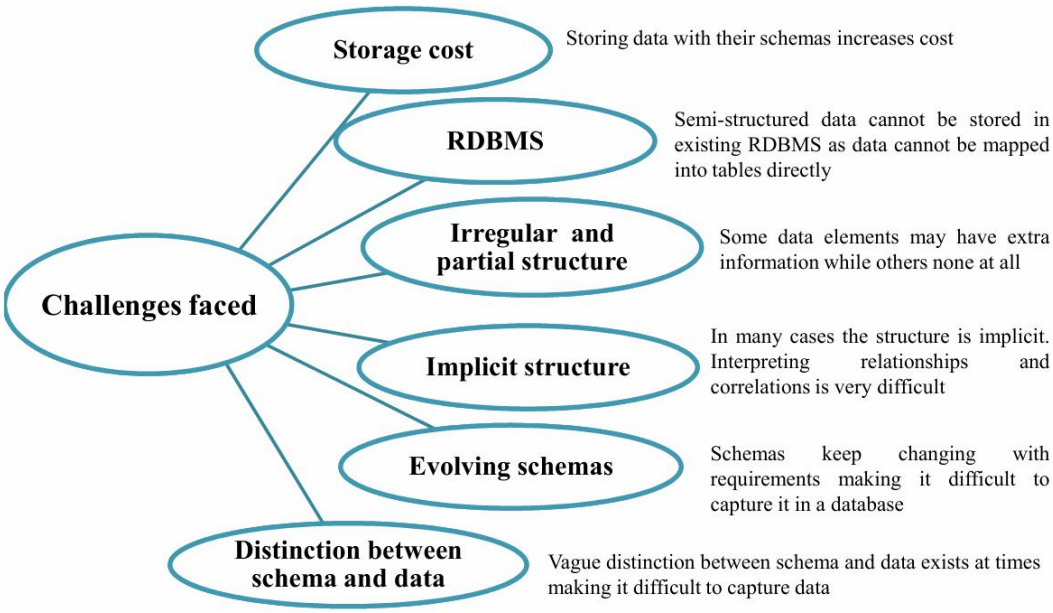
• The tags give us some metadata but the body of the e-mail contains no format neither is such which conveys meaning of the data it contains.

• There is very fine line between unstructured and semi-structured data.




Where does Semi-structured Data Come from?
E-mail
XML
TCP/IP packets
Zipped files
Semi-structured data
Binary executables
Mark-up languages
Integration of data from heterogeneous sources

How to Store Semi-structured Data?

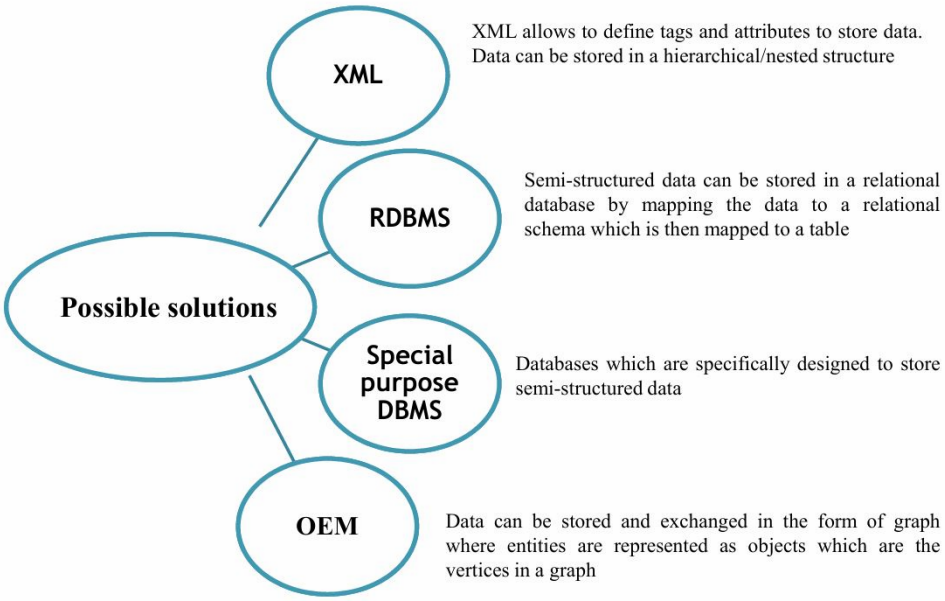


How to Manage Semi-structured Data?

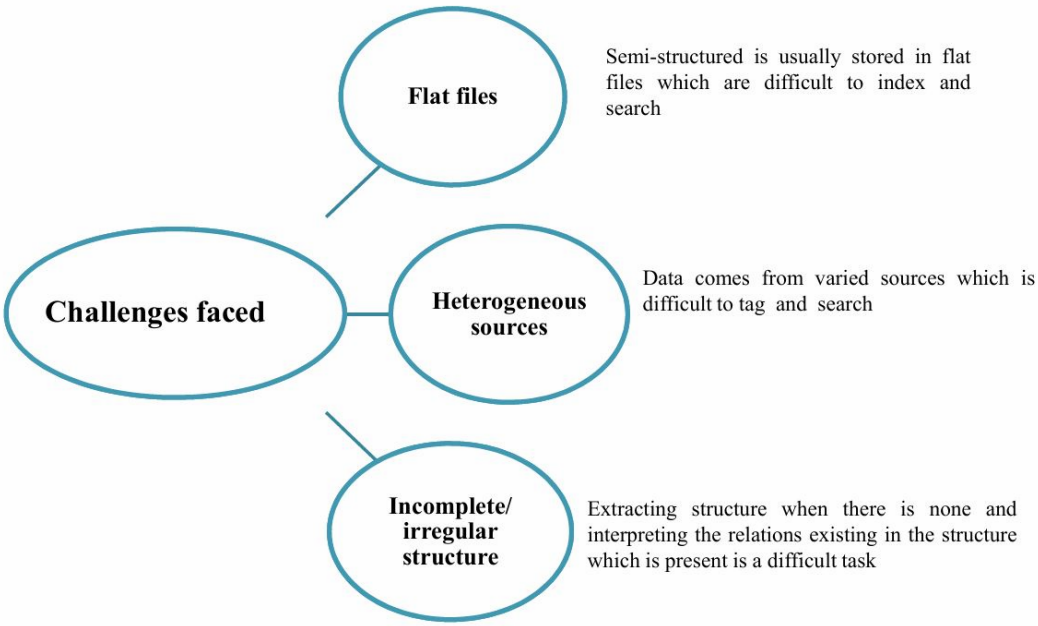
Some ways in which semi-structured data is managed and stored

Schemas	Graph-based data models	XML
• Describe the structure and content of data to some extent • Assign meaning to data hence allowing automatic search and indexing	• Contain data on the leaves of the graph. Also known as 'schema less' • Used for data exchange among heterogeneous sources	• Models the data using tags and elements • Schemas are not tightly coupled to data

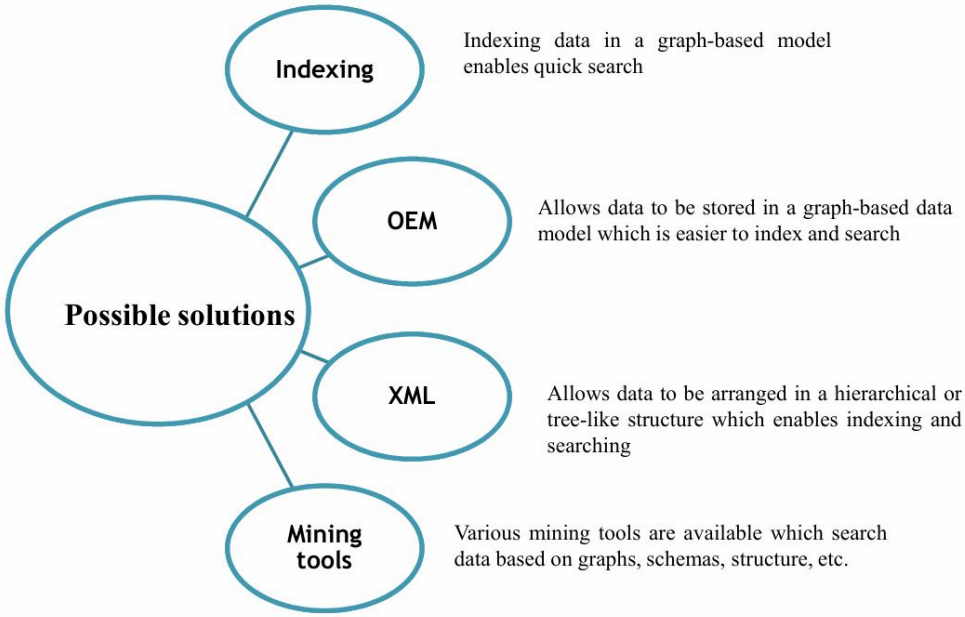
How to Store Semi-structured Data?



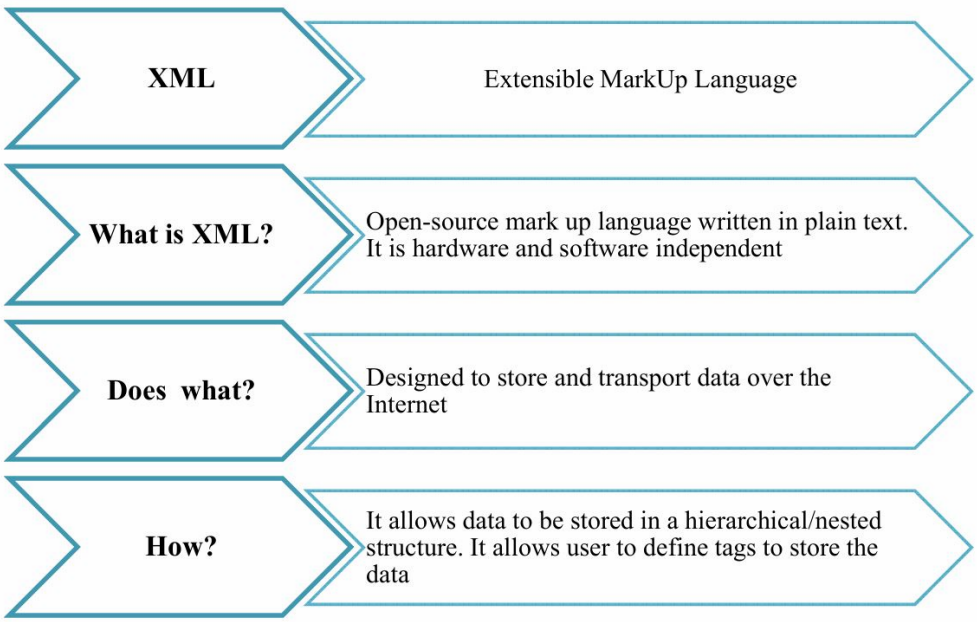
How to Extract Information from Semi-structured Data?



How to Extract Information from Semi-structured Data?



XML – A Solution for Semi-structured Data Management



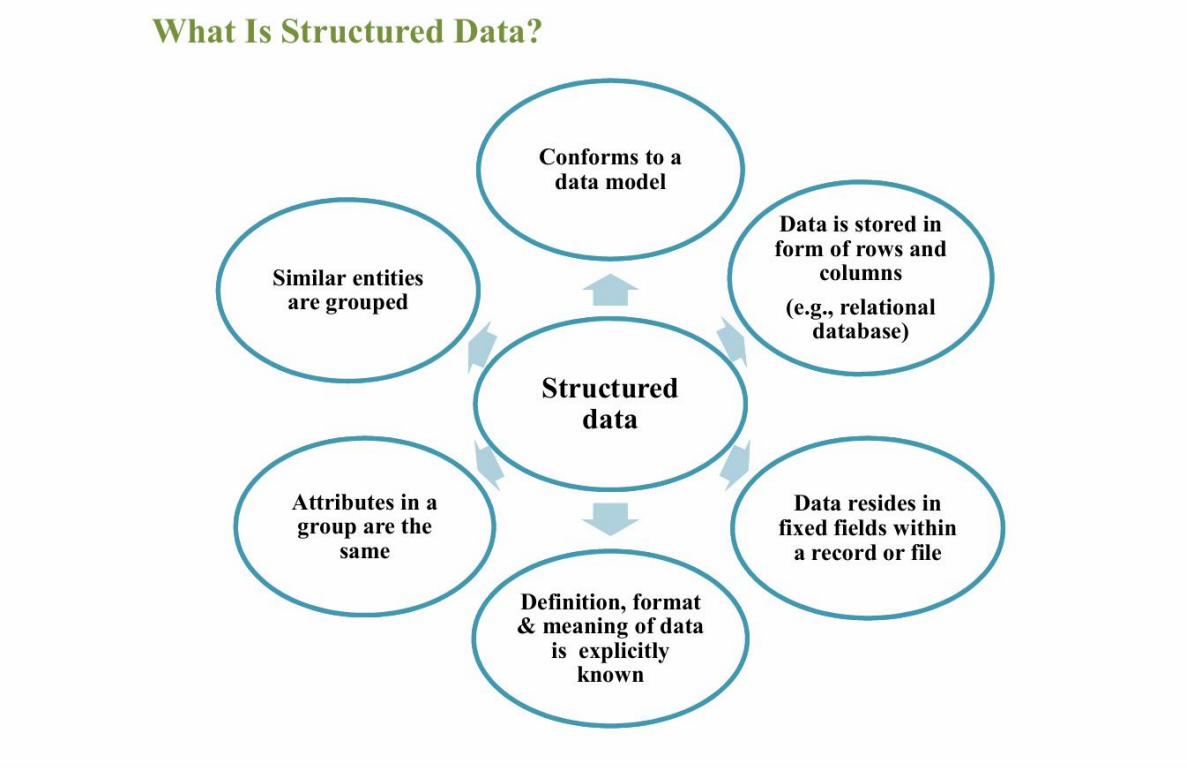
Structured Data

- Structured data is organized in semantic chunks (entities)
- Similar entities are grouped together (relations or classes)
- Entities in the same group have the same descriptions (attributes)

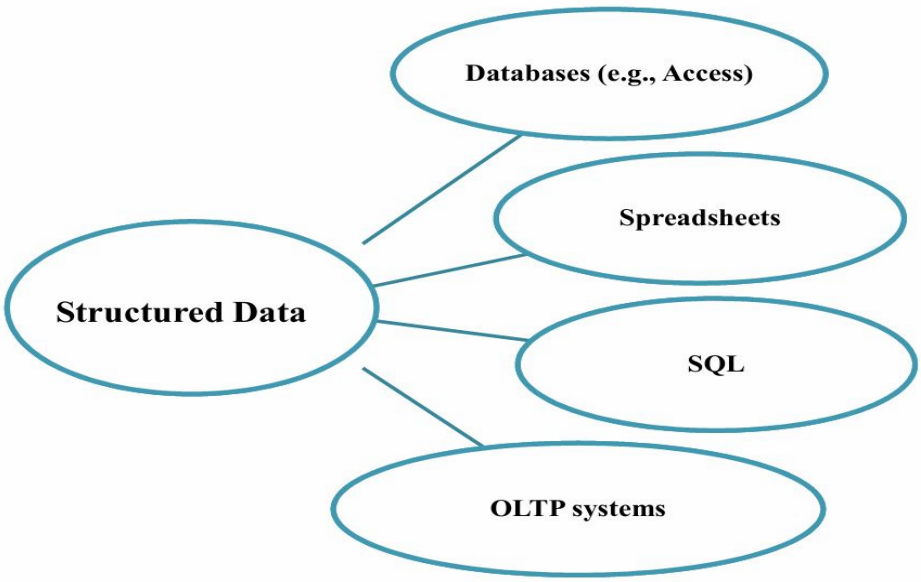
• Descriptions for all entities in a group (schema)

have the same defined format

have a predefined length are all present and follow the same order



Where does Structured Data Come from?



Structured Data: Everything in its Place

- Fully described datasets
- Clearly defined categories and sub-categories
- Data neatly placed in rows and columns
- Data that goes into the records is regulated by a well-defined structure
- Indexing can be easily done either by the DBMS itself or manually

Structured Data

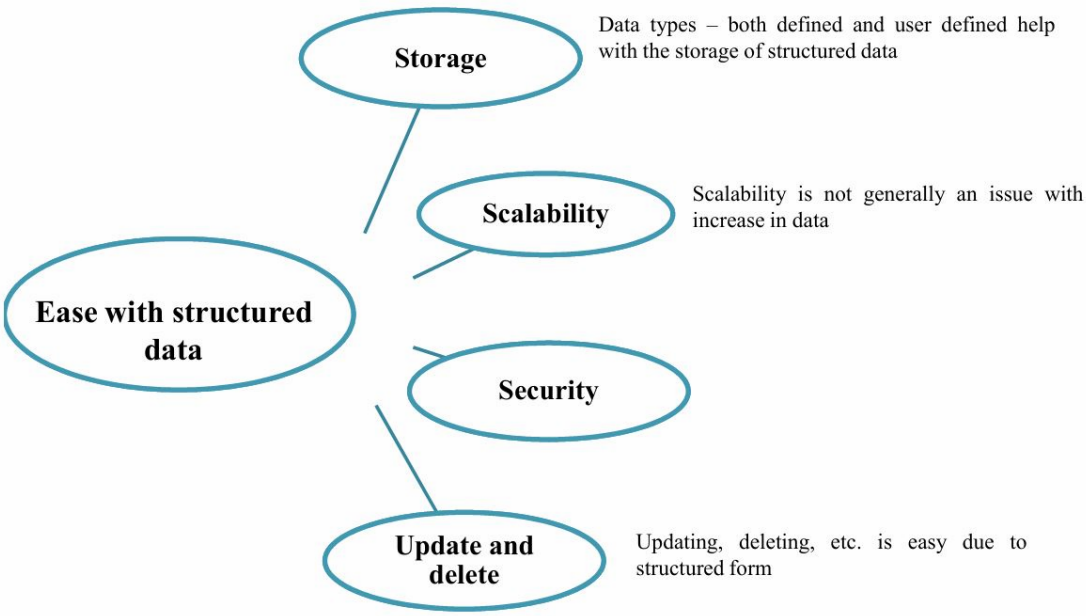
Semi-structured

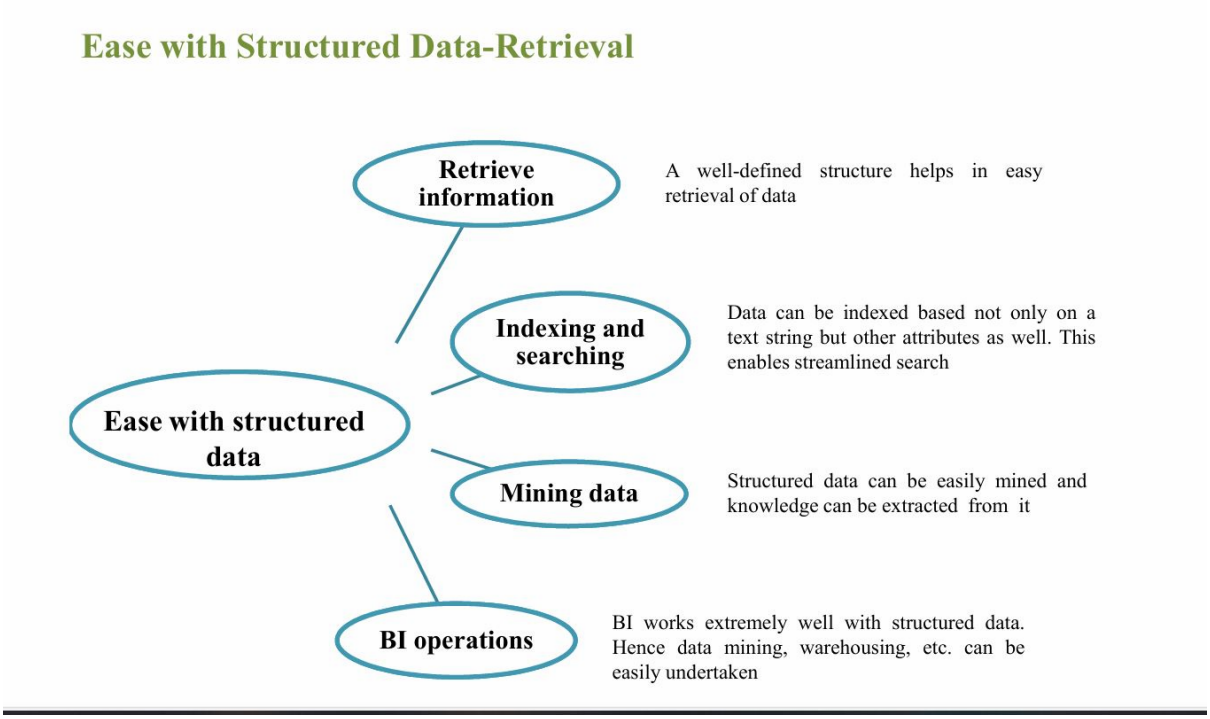
Name	E-mail
Patrick Wood	ptw@dcs.abc.ac.uk, p.wood@ymail.uk
First name: Mark Last name: Taylor	MarkT@dcs.ymail.ac.uk
Alex Bourdoo	AlexBourdoo@dcs.ymail.ac.uk

Structured

First Name	Last Name	E-mail Id	Alternate E-mail Id
Patrick	Wood	ptw@dcs.abc.ac.uk	p.wood@ymail.uk
Mark	Taylor	MarkT@dcs.ymail.ac.uk	
Alex	Bourdoo	AlexBourdoo@dcs.ymail.ac.uk	

Ease with Structured Data-Storage





Business Intelligence, Comparison business intelligence from traditional big data,

Business Intelligence (BI) and traditional Big Data approaches differ primarily in the type of data they handle, processing techniques, analysis goals, and use cases.

Business Intelligence (BI)

- Focuses on structured data mostly from internal systems such as ERP, CRM, and transactional databases.
- Deals with moderate to large datasets usually stored in centralized data warehouses or marts.
- Employs traditional SQL-based queries, OLAP, and visualization tools for historical and descriptive analysis.
- Provides insights to answer "what" and "where" questions, supporting routine reporting, operational decision-making, and business performance tracking.
- Requires moderate technical skills in querying and reporting structured data.
- BI solutions prioritize accuracy, simplicity, and stable schema environments.

Big Data (Traditional vs Business Intelligence Comparison)

- Handles massive volumes of structured, semi-structured, and unstructured data from diverse internal and external sources like social media, IoT devices, logs, and multimedia.
- Uses distributed computing frameworks (like Hadoop, Spark), NoSQL databases, and machine learning algorithms to process and analyze data at high velocity and scale.

- Focuses on both historical and real-time data for predictive and prescriptive analytics, addressing "why" and "how" questions about business trends and customer behavior.
- Employs advanced analytics techniques such as natural language processing and AI for deeper insights.
- Requires complex, multidisciplinary skills including data science and advanced analytics.
- Enables uncovering new patterns, real-time insights, and innovation opportunities beyond routine BI capabilities.

Key Comparison

Aspect	Business Intelligence	Big Data
Data Type	Structured data from internal systems	Structured, semi-structured, unstructured data from various sources
Data Volume	Moderate to large	Massive volumes (terabytes to exabytes)
Data Storage	Centralized data warehouses/marts	Distributed file systems and NoSQL databases
Processing Techniques	SQL, OLAP, BI tools	Hadoop, Spark, advanced machine learning
Analysis Focus	Historical, descriptive	Real-time, predictive, prescriptive
Insight Types	What happened, where	Why it happened, how to optimize
Technical Skills	Moderate technical querying skills	Data science, advanced analytics expertise
Use Cases	Reporting, performance tracking	Trend prediction, customer behavior, innovation

In summary, Business Intelligence provides reliable insights from structured historical data mainly for operational and strategic decision-making, while Big Data expands the scope to include vast, varied datasets and advanced analytics for predictive and innovative outcomes.

Many organizations now combine both approaches for comprehensive data-driven decision-making

What is Data Warehouse Architecture?

Data warehouse architecture consists of planning, designing, constructing, and managing daily operational processes for how data is used for organizational intelligence and decision support. A data warehouse architecture helps create a single source of truth for large volumes of data derived from various and different data sources. Data is then transformed into information and information is transformed into knowledge for analytics within the data warehouse architecture.

The data lifecycle includes data collection from identified sources, data integrity management and reconciliation, data storage, data transfer, and continuous improvement of data relative to organizational maturity, analytics, and decision needs. The data warehouse architecture must support these activities and other aspects of data lifecycle management.

Data warehouse architectures are usually designed to be stakeholder-oriented such as for sales, marketing, and others. Although using common data, each stakeholder has different modeling and data analysis needs for their decisions. This includes people using various tools as well as how technologies or applications consume data for translating the data to information and decisions.

Type of Data Warehouse Architectures

It is not a good practice to support analytical processing with a transactional database because of performance challenges. Transactional databases are optimized for processing huge volumes of transactions in real-time while analytical databases are optimized for long-running, resource-intensive queries. For this reason, transactional data should be an input to the data warehouse database rather than supporting both transactional and analytic needs.

There are different data warehouse models such as:

Basic Data Warehouse Architecture – Single Tier

This architecture minimizes the amount of data stored and data redundancies. It is not commonly used but may meet the needs of some small organizations that do not require enterprise access to data. Performance issues often occur when analytical and transaction processing are not separate.

Data Warehouse Architecture With a Centralized Repository – Two Tier

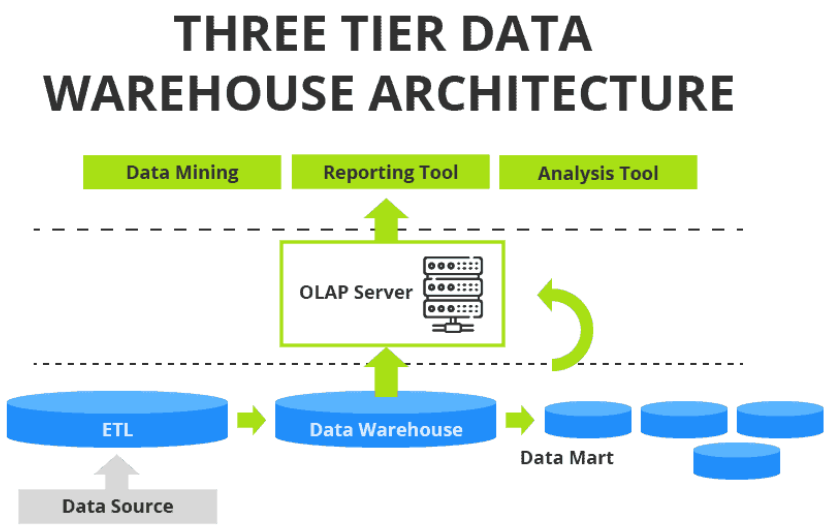
This architecture uses staging to extract specific data, transform data for usage, and load it into a data warehouse. This process is called Extraction, Transformation, and Loading (ETL). One of the extraction sources can be from a transactional database. Information is saved to one logically centralized individual repository, a data warehouse, that is paired with analytical tools. Data marts may be included in a two-tier data warehouse architecture to deliver focused business user applications.

Data Warehouse Architecture With a Centralized Repository, and an OLAP Server – Three Tier

This architecture adds an On-Line Analytical Processing (OLAP) Server to the two-tier design. This middle tier provides an abstracted view of the database for the end-user and helps with system scalability and performance.

In each data warehouse architecture listed, there is always room for additional optimization, such as using clusters to decentralize how data is managed and processed. This could be useful for challenges relative to data governance, locally or internationally. Data warehouse architectures could include bus, hub- and-spoke, and federated models to solve specific needs.

The following diagram shows a three-tier data warehouse architecture. The data warehouse structure can be modified at each level to fit more like components, such as an increase in the number of data marts to support additional functional units in the organization



Data Warehouse Infrastructure Diagram

The main components of a data warehouse architecture are:

- Data Sources – databases and other files, including a transactional database.
- The Data Warehouse itself.
- Data Marts – for specific stakeholder analytical capabilities.
- OLAP Server – enables fast, flexible multidimensional data analysis.
- Tools that stakeholder uses to access analytics (applications).

One of the values of architecture in data warehousing is simplicity. An organization can start with a basic structure using few components and add more later into various parts of the

architecture as the data strategy evolves. Basically, keep the design structure and expand the specific elements such as data sources to add depth and breadth to the solution.

Properties of Data Warehouse Architectures

Data warehouse architectures should focus on analytical processing. Transactional processing should be done separately using a different database. A transactional processing database should be a data source for the more extensive data warehouse.

Other properties of the data warehouse should include:

- The ability to scale the use of data for analytics quickly. This can be an essential factor for the prevalence of derived analytics incorporating the most recent data for the specific decisions that need to be made.
- The architecture should easily support additional data without redesigning the entire system.
- The data must be adequately secured. The data warehouse contains data about the entire organization. Compromise here is risky and could be very costly.
- Extraction, Transformation, and Loading tools should support different data sources.
- The architecture management should not be overly complicated and should be simplified for ease of use and better analytical outcomes.
- The data warehouse architecture in data mining applications should use trusted data that has been adequately extracted, transformed, and loaded into the data warehouse. Data mining tools that do not have good data will only return inaccurate results.
- As the organization matures and understands how to use data, the data warehouse solution should have the ability to transform quickly to accommodate changes.

Data warehouse architectures should also deliver a level of warranty relative to availability, security, capacity, and continuity of usage. These elements of service warranty for the data warehouse should also include usability and performance.

The data warehouse should easily support tools and applications such as reporting, data mining, and application development tools.

Analytical tools used for big data analytics:

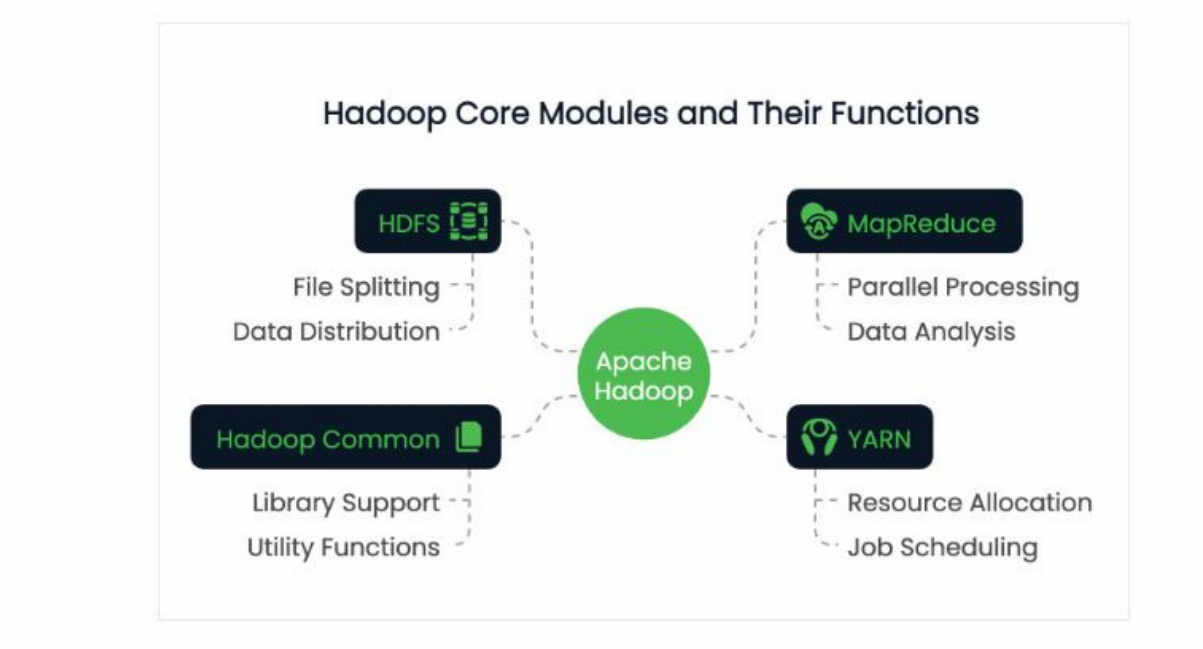
tegory	Tool	Key Features	Best For	Implementation Tip
Core Processing	Hadoop	Distributed storage, parallel processing	Batch processing for data lakes (e.g., analyzing historical sales data).	Start with a small cluster and scale as needed
	Apache Spark	In-memory processing, stream handling	Real-time analytics (e.g., processing GPS data for live route optimization).	Use Spark's MLlib for predictive tasks
Cloud Solutions	Google BigQuery	Serverless architecture, automatic scaling	Ad-hoc analysis (e.g., running queries on massive marketing datasets).	Use partitioned tables to optimize costs
	Amazon Redshift	Columnar storage, parallel processing	Enterprise BI (e.g., analyzing clinical trial data for research insights).	Enable automatic workload management
Visualization	Tableau	Drag-and-drop interface, interactive dashboards	Interactive reporting (e.g., visualizing inventory trends for decision-making).	Use data extracts for faster dashboard performance
	Power BI	Microsoft ecosystem integration, AI insights	Dynamic reports (e.g., monitoring real-time traffic data for operations).	Leverage DirectQuery for live data needs

Specialized Tools	TensorFlow	Deep learning, neural networks, GPU support	Advanced AI (e.g., building predictive models for user behavior).	Use pre-built models for common ML tasks
	Acceldata	Data observability, quality monitoring, real-time performance optimization	Ensuring clean, reliable pipelines (e.g., identifying and fixing bottlenecks in data operations).	Leverage Acceldata for monitoring across multiple systems to enhance analytics reliability.

Components of Hadoop

Hadoop is a framework that uses distributed storage and parallel processing to store and manage Big Data. It is the most commonly used software to handle Big Data. There are three components of Hadoop.

1. Hadoop HDFS - Hadoop Distributed File System (HDFS) is the storage unit of Hadoop.
2. Hadoop MapReduce - Hadoop MapReduce is the processing unit of Hadoop.
3. Hadoop YARN - Hadoop YARN is a resource management unit of Hadoop.



Hadoop HDFS

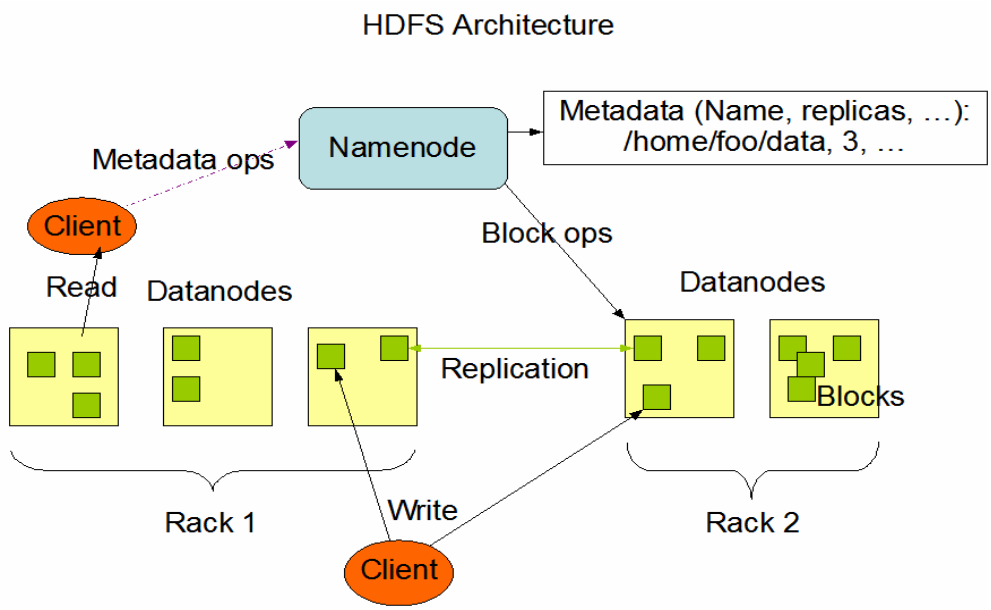
HDFS is Hadoop's primary storage system. It is designed to reliably store the vast amounts of data across a cluster of machines that we discussed earlier. Its architecture is set up for this type of access to large datasets and is optimized for fault tolerance, scalability, and data locality.

Architecture and components

The HDFS architecture revolves around a **master-slave model**. At the top sits the NameNode, which manages metadata— essentially, the file system's directory tree and information about each file's location. It doesn't store the actual data.

The DataNodes are the workhorses. They manage storage attached to the nodes and serve read/write requests from clients. Each DataNode regularly reports back to the NameNode with a heartbeat and block reports to ensure consistent state tracking.

Finally, HDFS includes a Secondary NameNode, not to be confused with a failover node. Instead, it periodically checkpoints the NameNode's metadata to reduce startup time and memory overhead.



How data is stored in HDFS

When a file is written to HDFS, it's broken into fixed-size **64MB blocks**. Each block is then distributed across different DataNodes. This distribution strategy supports parallelism during data access and boosts fault tolerance.

Data in HDFS isn't just stored once. Each block is replicated (the default is three copies) and spread across the cluster. During a read operation, the system pulls data from the nearest available replica to maximize throughput and minimize latency. Writes go to one replica first and then propagate to the others, ensuring durability without immediate bottlenecks.

This block-level design also allows HDFS to scale horizontally. New nodes can be added to the cluster, and the system will rebalance data to use additional capacity.

Fault tolerance in HDFS

HDFS is **built with failure in mind**. In large distributed systems, node failures are expected, not exceptional. HDFS ensures data availability through its replication mechanism. If a DataNode goes down, the NameNode detects the failure via missed heartbeats and schedules the replication of lost blocks to healthy nodes.

Plus, the system constantly monitors block health and initiates replication as needed. The redundant storage strategy, combined with proactive monitoring, ensures that no single point of failure can compromise data integrity or access.

Hadoop MapReduce

Next up is Hadoop's processing engine, MapReduce. It allows for distributed computation across large datasets by breaking down tasks into smaller, independent operations that can be executed in parallel.

The model simplifies complex data transformations and is especially suited for batch processing in a distributed environment.

Overview of MapReduce

MapReduce follows a two-phase programming model: the **Map** and **Reduce** phases.

During the Map phase, the input dataset is divided into smaller chunks, and each chunk is processed to produce key-value pairs. These intermediate results are then shuffled and sorted before being passed to the Reduce phase, where they are aggregated or transformed into final outputs.

This model is highly effective for tasks like counting word frequencies, filtering logs, or joining datasets at scale.

Developers implement custom logic through map() and reduce() functions, which the framework orchestrates across the cluster.

Execution flow of a MapReduce job

When a MapReduce job is submitted, the input data is first split into blocks. A JobTracker (or ResourceManager in YARN-enabled versions) then coordinates the distribution of these blocks across available TaskTrackers or NodeManagers.

The Map tasks execute in parallel across nodes, reading input splits and emitting intermediate key-value pairs. These pairs are then shuffled and sorted automatically. Once this phase is complete, the Reduce tasks begin, pulling in grouped data and applying aggregation logic to produce the final output.

The process guarantees data locality by assigning tasks close to where the data resides, minimizing network congestion and improving throughput. Task failures are detected and reassigned automatically, contributing to system resilience.

MapReduce benefits and limitations

MapReduce offers several clear advantages:

- Excellent parallelism for large-scale batch processing
- Scalability across thousands of nodes
- Strong fault tolerance through task re-execution and checkpointing

But, it also comes with trade-offs:

- The model is latency-heavy, so it's less ideal for real-time processing
- Writing custom MapReduce jobs can be a big lift and difficult to debug
- It lacks the interactivity and flexibility of newer tools like **Apache Spark**.

YARN: Yet Another Resource Negotiator

YARN serves as Hadoop's resource management layer. It separates job scheduling and resource allocation from the processing model, helping Hadoop support multiple data processing engines beyond MapReduce.

Role in Hadoop architecture

YARN acts as the central platform for managing compute resources across the cluster. It allocates available system resources to several applications and coordinates execution. This separation of concerns helps Hadoop scale efficiently and opens the door to support other frameworks like Apache Spark, Hive, and Tez.

YARN key components

YARN introduces three components:

1. **ResourceManager:** The master daemon (background program) that manages all resources and schedules applications.
2. **NodeManager:** A per-node agent that monitors resource usage and reports back to the ResourceManager.
3. **ApplicationMaster:** A job-specific operation that uses resources from the ResourceManager and coordinates execution with NodeManagers.

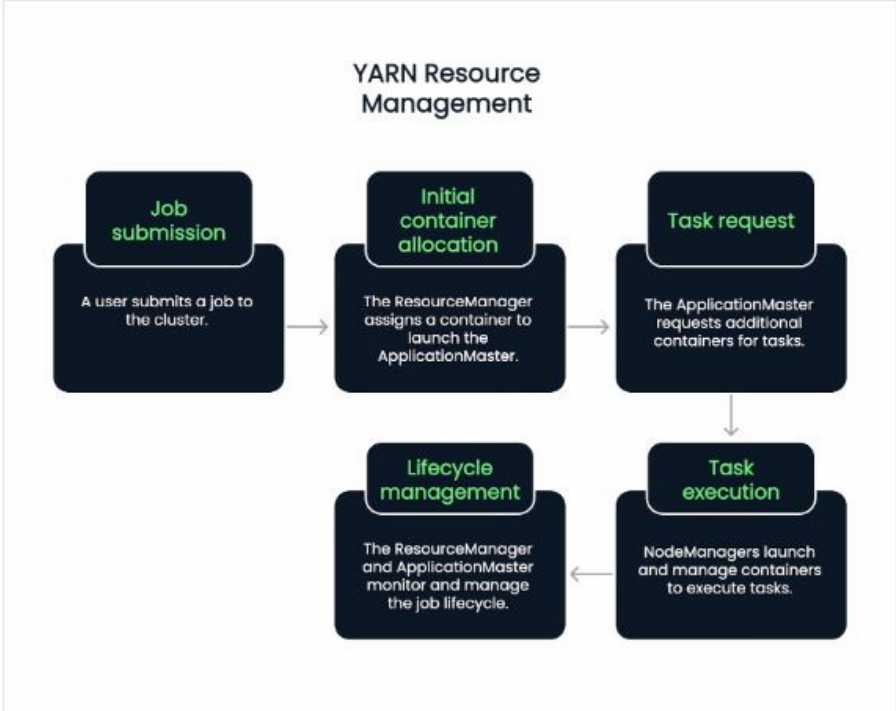
Each application (like a MapReduce job) has its own ApplicationMaster, which allows for better fault isolation and job tracking.

YARN resource management process

The flow begins when a user submits a job. The ResourceManager accepts the job and assigns a container to start the ApplicationMaster. From there, the ApplicationMaster requests more containers to execute tasks, based on the current workload and availability.

NodeManagers launch and manage these containers, which run individual tasks. Throughout the job lifecycle, the ResourceManager and ApplicationMaster handle monitoring, failure recovery, and completion reporting. So, it's about a five-step process.

1. **Job submission:** A user submits a job to the cluster.
2. **Initial container allocation:** The ResourceManager accepts the job and assigns a container to launch the ApplicationMaster.
3. **Task request:** The ApplicationMaster requests additional containers to run specific tasks, based on workload and available resources.
4. **Task execution:** NodeManagers launch and manage these containers, which carry out the tasks.
5. **Lifecycle management:** The ResourceManager and ApplicationMaster work together to monitor job progress, handle failures, and confirm task completion.Throughout the job lifecycle, the ResourceManager and ApplicationMaster handle monitoring, failure recovery, and completion reporting.


YARN Resource Management
Job submission
A user submits a job to the cluster.
Initial container allocation
The ResourceManager assigns a container to launch the ApplicationMaster.
Task request
The ApplicationMaster requests additional containers for tasks.
Task execution
NodeManagers launch and manage containers to execute tasks.
Lifecycle management
The ResourceManager and ApplicationMaster monitor and manage the job lifecycle.