BDA

1.1 Introduction to Big Data - Big Data characteristics and Types of Big Data

- We all are surrounded by huge data. People upload/download videos, audios, images from variety of devices. Sending
  text messages, multimedia messages, updating their Facebook, Whatsapp, Twitter status, comments, online shopping,
  online advertising etc. generates huge data.
- As a result of multiple processing machines have to generate and keep huge data too. Due to this exponential growth
  of data, Data analysis becomes very much required task for day to day operations.
- The term 'Big Data' means huge volume, high velocity and a variety of data.
- This big data is increasing tremendously day by day.
- Traditional data management systems and existing tools are facing difficulties to process such a Big Data.
- The term 'Big Data' means huge volume, high velocity and a variety of data. This big data is increasing tremendously
  day by day. Traditional data management systems and existing tools are facing difficulties to process such a Big Data.
- Big data is the most important technologies in modern world. It is really critical to store and manage it. Big data is a collection of large datasets that cannot be processed using traditional computing techniques.
- Big Data includes huge volume, high velocity and extensible variety of data. The data in it may be structured data,
  Semi Structured data or unstructured data. Big data also involves various tools, techniques and frameworks.

Characteristics of Big data / FOUR imp V's of Big data

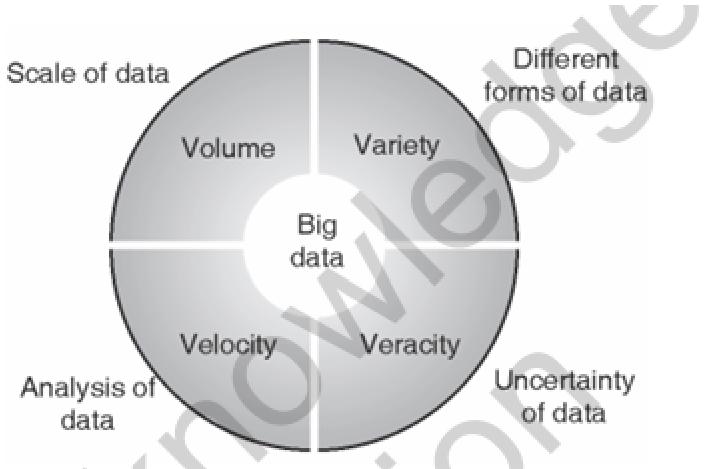


Fig. 1.3.1 : Big data characteristics

1. Volume:

- The name ‘Big Data’ itself is related to a size which is enormous.
- Volume is a huge amount of data.
- To determine the value of data, size of data plays a very crucial role. If the volume of data is very large, then it is actually considered as a ‘Big Data’. This means whether a particular data can actually be considered as a Big Data or not, is dependent upon the volume of data.
- Hence while dealing with Big Data it is necessary to consider a characteristic ‘Volume’.
- *Example:* In the year 2016, the estimated global mobile traffic was 6.2 Exabytes (6.2 billion GB) per month. Also, by the year 2020 we will have almost 40000 Exabytes of data.

2. Velocity:

- Velocity refers to the high speed of accumulation of data.
- In Big Data velocity data flows in from sources like machines, networks, social media, mobile phones etc.
- There is a massive and continuous flow of data. This determines the potential of data that how fast the data is generated and processed to meet the demands.
- Sampling data can help in dealing with the issue like ‘velocity’.
- *Example:* There are more than 3.5 billion searches per day are made on Google. Also, Facebook users are increasing by 22%(Approx.) year by year.

3. Variety:

- It refers to nature of data that is structured, semi-structured and unstructured data.
- It also refers to heterogeneous sources.
- Variety is basically the arrival of data from new sources that are both inside and outside of an enterprise. It can be structured, semi-structured and unstructured.
    - **Structured data**: This data is basically an organized data. It generally refers to data that has defined the length and format of data.
    - **Semi- Structured data**: This data is basically a semi-organised data. It is generally a form of data that do not conform to the formal structure of data. Log files are the examples of this type of data.
    - **Unstructured data**: This data basically refers to unorganized data. It generally refers to data that doesn't fit neatly into the traditional row and column structure of the relational database. Texts, pictures, videos etc. are the examples of unstructured data which can't be stored in the form of rows and columns.

4. Veracity:

- It refers to inconsistencies and uncertainty in data, that is data which is available can sometimes get messy and quality and accuracy are difficult to control.
- Big Data is also variable because of the multitude of data dimensions resulting from multiple disparate data types and sources.
- *Example:* Data in bulk could create confusion whereas less amount of data could convey half or Incomplete Information.

1.2 Traditional vs. Big Data business approach

Sr no	Dimension	Traditional	Big data
1	Data source	Mainly internal.	Both inside and outside organization, including traditional.

Sr no	Dimension	Traditional	Big data
2	Data structure	Pre-defined structure.	Unstructured in nature.
3	Data relationship	By default, stable and interrelationship.	Unknown relationship.
4	Data location	Centralized.	Physically highly distributed.
5	Data analysis	After the complete build.	Intermediate analysis, as you go.
6	Data reporting	Mostly canned with limited and pre-defined interaction paths.	Reporting in all possible directions across the data in real-time mode.
7	Cost factor	Specialized high-end hardware and software.	Inexpensive commodity boxes in cluster mode.
8	CAP theorem	Consistency - Top priority.	Availability - Top priority.

Types of Big data

All data cannot be stored in the same way. The methods for data storage can be accurately evaluated after the type of data has been identified. A Cloud Service, like Microsoft Azure, is a one-stop destination for storing all kinds of data; blobs, queues, files, tables, disks, and applications data. However, even within the Cloud, there are special services to deal with specific sub-categories of data.

For example

, Azure Cloud Services like Azure SQL and Azure Cosmos DB help in handling and managing sparsely varied kinds of data.

Structured Data

- Structured data can be crudely defined as the data that resides in a fixed field within a record.
- It is type of data most familiar to our everyday lives. for ex: birthday,address

- A certain schema binds it, so all the data has the same set of properties. Structured data is also called relational data. It is split into multiple tables to enhance the integrity of the data by creating a single record to depict an entity. Relationships are enforced by the application of table constraints.
- The business value of structured data lies within how well an organization can utilize its existing systems and processes for analysis purposes.

Examples of structured data include numbers, dates, strings, etc. The business data of an e-commerce website can be considered to be structured data.

Cons of Structured Data

1. Structured data can only be leveraged in cases of predefined functionalities. This means that structured data has limited flexibility and is suitable for certain specific use cases only.
2. Structured data is stored in a data warehouse with rigid constraints and a definite schema. Any change in requirements would mean updating all of that structured data to meet the new needs. This is a massive drawback in terms of resource and time management.

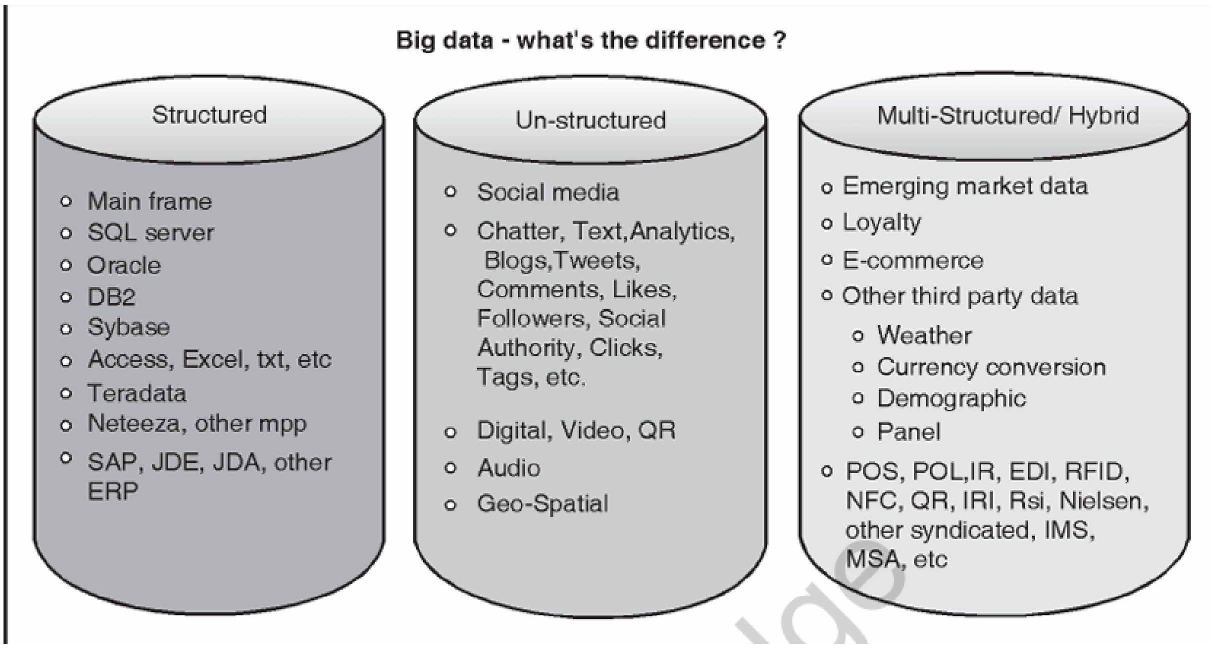
Semi-Structured Data

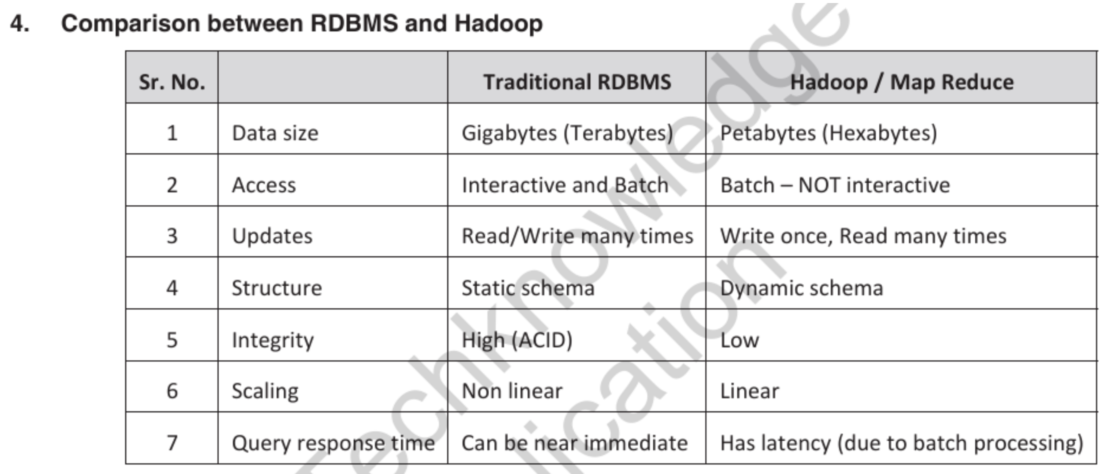
- Semi-structured data is not bound by any rigid schema for data storage and handling. The data is not in the relational format and is not neatly organized into rows and columns like that in a spreadsheet. However, there are some features like key-value pairs that help in discerning the different entities from each other.
- Since semi-structured data doesn’t need a structured query language, it is commonly called *NoSQL data*.
- A data serialization language is used to exchange semi-structured data across systems that may even have varied underlying infrastructure.
- Semi-structured content is often used to store metadata about a business process but it can also include files containing machine instructions for computer programs.
- This type of information typically comes from external sources such as social media platforms or other web-based data feeds.

A product catalog organized by tags is an example of semi-structured data.

Unstructured Data

- Unstructured data is the kind of data that doesn’t adhere to any definite schema or set of rules. Its arrangement is unplanned and haphazard.
- Photos, videos, text documents, and log files can be generally considered unstructured data. Even though the metadata accompanying an image or a video may be semi-structured, the actual data being dealt with is unstructured.
- Additionally, Unstructured data is also known as “dark data” because it cannot be analyzed without the proper software tools.
-





4. Comparison between RDBMS and Hadoop

Sr. No.		Traditional RDBMS	Hadoop / Map Reduce
1	Data size	Gigabytes (Terabytes)	Petabytes (Hexabytes)
2	Access	Interactive and Batch	Batch – NOT interactive
3	Updates	Read/Write many times	Write once, Read many times
4	Structure	Static schema	Dynamic schema
5	Integrity	High (ACID)	Low
6	Scaling	Non linear	Linear
7	Query response time	Can be near immediate	Has latency (due to batch processing)

1.3 Case Study of Big Data Solutions

1. **Healthcare and Public Health Industry**
    - The computing power of big data analytics enables us to predict diseases, discover new cures, and better understand and predict disease patterns.
    - Entire DNA sequences can be decoded within minutes.
    - Smartwatches can be utilized to predict symptoms for various diseases.
    - Big data techniques are already being used to monitor babies in specialized premature and sick baby units. By recording and analyzing every heartbeat and breathing pattern of each baby, algorithms have been developed to predict infections 24 hours before any physical symptoms appear.
    - Big data analytics allows us to monitor and predict the development of epidemics and diseases. Integrating data from medical records with social media analytics enables us to monitor certain diseases.
2. **Sports**
    - All popular sports perform analysis using big data analytics.
    - In cricket and football, many predictions are observed to be accurate most of the time.
    - An example is the IBM SlamTracker tool used in tennis tournaments.
    - Video analytics can track the performance of every player in a cricket game, and sensor technology in sports equipment, such as basketballs, allows feedback even via smartphones.
    - Many elite sports teams also track athletes outside of the sporting environment using smart technology to monitor nutrition, sleep, and even social media conversations.
3. **Science and Research**
    - Science and research are being transformed by the new possibilities offered by big data.
    - Experiments often involve testing numerous possibilities and generate vast amounts of data.

- Many advanced labs use the computing power of thousands of computers distributed across data centers worldwide to analyze data.
- Big data is leveraged to enhance the performance areas of science and research.

4. **Security Enforcement**
    - Big data is applied to improve national security enforcement.
    - The National Security Agency (NSA) in the U.S. uses big data analytics to thwart terrorist plots.
    - Big data techniques are used to detect and prevent cyber attacks.
    - Police forces use big data tools to catch criminals and even predict criminal activities, while credit card companies use big data to detect fraudulent transactions.
5. **Financial Trading**
    - Automated Trading and High-Frequency Trading (HFT) are new areas where big data plays a significant role.
    - Big data algorithms can be used to make trading decisions.
    - Today, the majority of equity trading is conducted via data algorithms that increasingly consider signals from social media networks and news websites to make buy and sell decisions in split seconds.

1.4 Concept of Hadoop, Core Hadoop Components; Hadoop Ecosystem

Hadoop is an open-source, big data storage and processing software framework. Hadoop stores and processes big data in a distributed fashion on large clusters of commodity hardware. Massive data storage and faster processing are the two important aspects of Hadoop.

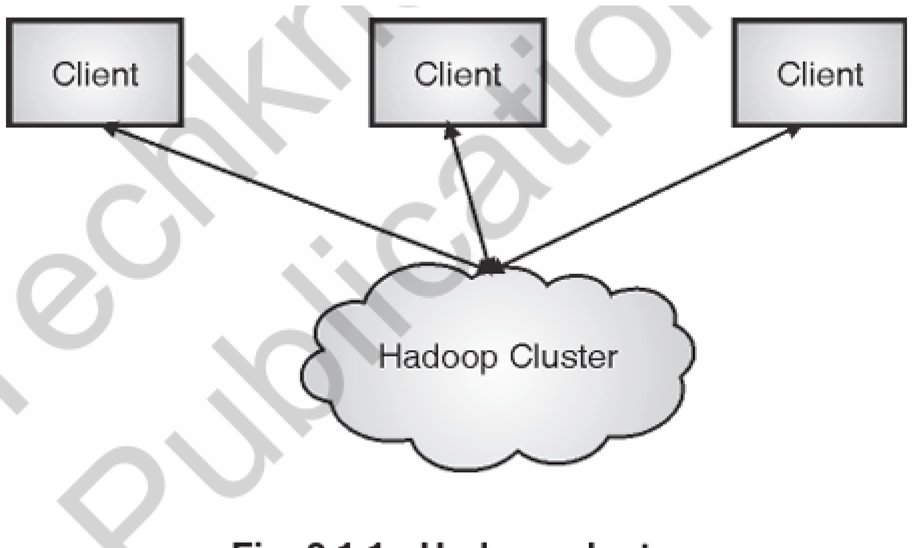


Fig. 2.1.1 : Hadoop cluster

Fig. 2.1.1: Hadoop cluster

- As shown in Fig. 2.1.1, a Hadoop cluster is a set of commodity machines networked together in one location, i.e., the cloud.
- These cloud machines are then used for data storage and processing. From individual clients, users can submit their jobs to the cluster. These clients may be present at some remote locations from the Hadoop cluster.
- Hadoop runs applications on systems with thousands of nodes involving huge storage capabilities. As a distributed file system is used by Hadoop, data transfer rates among nodes are very fast.
- As thousands of machines are there in a cluster, users can get uninterrupted service, and node failure is not a big issue in Hadoop, even if a large number of nodes become inoperative.
- Hadoop uses distributed storage and transfers code to data. This code is tiny and consumes less memory also.
- This code executes with data there itself. Thus, the time to get data and again restore results is saved as the data is locally available. Thus, interprocess communication time is saved, which makes for faster processing.
- The redundancy of data is an important feature of Hadoop, due to which node failures are easily handled.
- In Hadoop, users do not need to worry about partitioning the data, data and task assignment to nodes, or communication between nodes. As Hadoop

handles it all, users can concentrate on data and operations on that data.

2.1.1 Hadoop - Features

1. **Low Cost**
    - As Hadoop is an open-source framework, it is free. It uses commodity hardware to store and process huge data, making it cost-effective.
2. **High Computing Power**
    - Hadoop uses a distributed computing model, allowing tasks to be distributed among different nodes and processed quickly. Clusters with thousands of nodes provide Hadoop with high computing capability.
3. **Scalability**
    - Nodes can be easily added or removed, and failed nodes can be detected with minimal administration required.
4. **Huge and Flexible Storage**
    - Massive data storage is available due to the thousands of nodes in the cluster. Hadoop supports both structured and unstructured data, with no need for preprocessing before storage.
5. **Fault Tolerance and Data Protection**
    - If any node fails, the tasks are automatically redirected to other nodes. Multiple copies of all data are automatically stored, ensuring that data remains available even if a node fails.

2.3 Hadoop Physical Architecture

Hadoop's Physical Architecture involves running a set of resident programs, known as daemons, which can be executed on the same server or different servers within the network. These daemons are responsible for specific functionalities within the Hadoop ecosystem. Below is a breakdown of the key components:

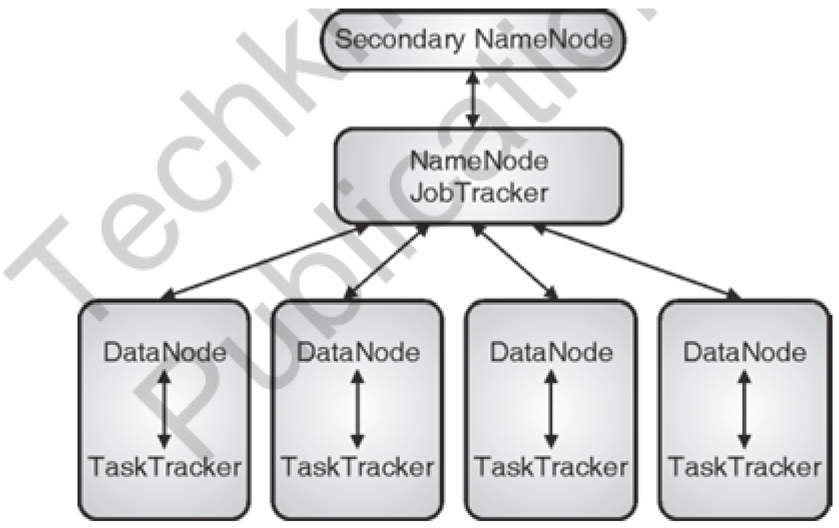


Fig. 2.3.1 : Hadoop cluster topology

1. NameNode

- **Master of HDFS**: The NameNode is the master of the Hadoop Distributed File System (HDFS).
- **DataNode as Slave**: It manages DataNodes, which serve as the slaves of HDFS.
- **JobTracker Role**: The NameNode also has a JobTracker that keeps track of files distributed across DataNodes.
- **Low-Level I/O Tasks**: It directs DataNodes regarding low-level input/output tasks.
- **Single Point of Failure**: The NameNode is the only single point of failure within the cluster.

2. DataNode

- **Slave of HDFS**: The DataNode operates as the slave node in HDFS.
- **Client Block Management**: It takes client block addresses from NameNodes and handles local data storage and processing tasks.
- **Replication and Local Updates**: DataNodes communicate with other DataNodes for replication and share local updates with the NameNode.
- **Task Execution**: It receives instructions for task execution from the NameNode and manages data storage across the cluster.

3. Secondary NameNode (SNN)

- **State Monitoring**: The SNN monitors the state of the HDFS cluster and resides on a separate machine.
- **Cluster Snapshot**: It takes periodic snapshots of the HDFS metadata by communicating with the NameNode to maintain data integrity.
- **Single Instance**: There is typically one SNN per cluster, which prevents the overload of NameNode.

4. JobTracker

- **Task Management**: The JobTracker assigns files to process, manages node assignments for different tasks, and monitors the progress of jobs.
- **Single Instance**: There is only one JobTracker daemon per Hadoop cluster.
- **Master Node**: It runs on the server as the master node of the cluster.

5. TaskTracker

- **Task Execution**: The TaskTracker is responsible for executing individual tasks assigned by the JobTracker.
- **Multiple JVMs**: It can handle multiple tasks simultaneously using different Java Virtual Machines (JVMs).
- **Continuous Communication**: The TaskTracker constantly communicates with the JobTracker. If the TaskTracker fails to respond within a specified time, it is assumed to have crashed, and tasks are rescheduled to other nodes within the cluster.

This architecture ensures high availability, fault tolerance, and efficient processing of large-scale data across the Hadoop cluster.

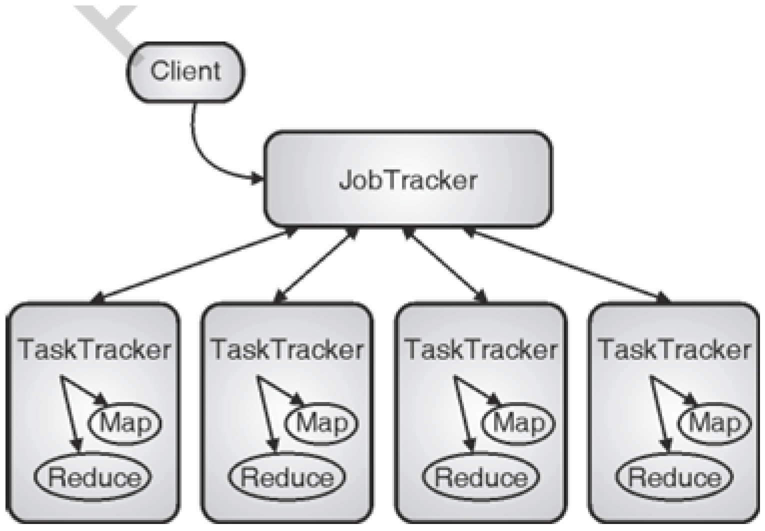


Fig. 2.3.2 : Jobtracker and tasktracker interaction

HDFS

With growing data velocity the data size easily outgrows the storage limit of a machine. A solution would be to store the data across a network of machines. Such filesystems are called *distributed filesystems*. Since data is stored across a network all the complications of a network come in.

This is where Hadoop comes in. It provides one of the most reliable filesystems. HDFS (Hadoop Distributed File System) is a unique design that provides storage for *extremely large files* with streaming data access pattern and it runs on *commodity hardware*. Let's elaborate the terms:

- ***Extremely large files***: Here we are talking about the data in range of petabytes(1000 TB).
- ***Streaming Data Access Pattern***: HDFS is designed on principle of *write-once and read-many-times*. Once data is written large portions of dataset can be processed any number times.
- ***Commodity hardware:*** Hardware that is inexpensive and easily available in the market. This is one of feature which specially distinguishes HDFS from other file system.

Nodes: Master-slave nodes typically forms the HDFS cluster.

1. **NameNode(MasterNode):**
    - Manages all the slave nodes and assign work to them.
    - It executes filesystem namespace operations like opening, closing, renaming files and directories.
    - It should be deployed on reliable hardware which has the high config. not on commodity hardware.
2. **DataNode(SlaveNode):**
    - Actual worker nodes, who do the actual work like reading, writing, processing etc.
    - They also perform creation, deletion, and replication upon instruction from the master.
    - They can be deployed on commodity hardware.

HDFS daemons: Daemons are the processes running in background.

- **Namenodes:**
    - Run on the master node.
    - Store metadata (data about data) like file path, the number of blocks, block Ids. etc.
    - Require high amount of RAM.
    - Store meta-data in RAM for fast retrieval i.e to reduce seek time. Though a persistent copy of it is kept on disk.
- **DataNodes:**
    - Run on slave nodes.
    - Require high memory as data is actually stored here.

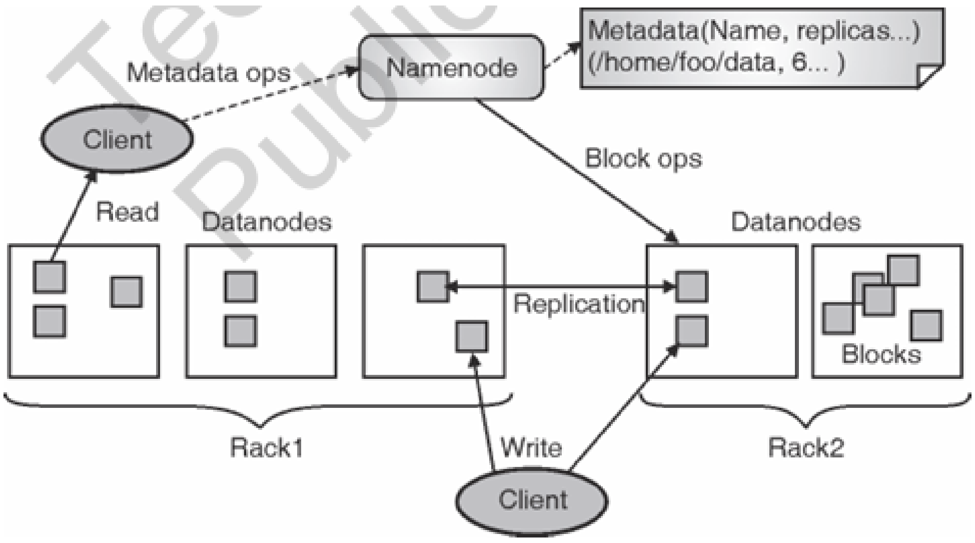


Fig. 2.4.2 : HDFS architecture

Mapreduce

MapReduce is a framework that breaks down applications into smaller parts (blocks or fragments) that can be processed in parallel across a cluster of machines. It manages data processing and scaling automatically. The two key phases are:

- **Mapping**: Filters and transforms input data into key/value pairs.
- **Reducing**: Aggregates and processes the outputs from the mapper to produce the final result.

Core functions:

1. **Read Input**
   - Divides input into small parts or blocks, which are then assigned to a Map function.
2. **Function Mapping**
   - Converts file data into smaller, intermediate `<key, value>` pairs.
3. **Partition, Compare, and Sort**
   - **Partition Function**: Determines the correct reducer for a given key and number of reducers.

- **Compare Function**: Sorts the intermediate outputs according to this function.

4. **Function Reducing**
    - Reduces intermediate values to smaller solutions and provides the final output.
5. **Write Output**
    - Produces the file output.

Hadoop ecosystem

Hadoop Distributed File System is the core component.
HDFS stores different types of large data sets (i.e. structured, unstructured and semi structured data).
It helps us in storing our data across various nodes and maintaining the log file about the stored data (metadata).
HDFS has two core components, i.e. NameNode and DataNode
The NameNode is the main node and it doesn't store the actual data. It contains metadata, just like a log file Therefore, it requires less storage and high computational resources.
All your data is stored on the DataNodes and hence it requires more storage resources. These DataNodes are commodity hardware (like your laptops and desktops).That's the reason, why Hadoop solutions are very cost effective.
You always communicate to the NameNode while writing the data. Then, it internally sends a request to the client to store and replicate data on various DataNodes.

HBase is an open source, non-relational distributed database (NoSQL database).
It supports all types of data and that is why, it's capable of handling anything and everything inside a Hadoop ecosystem.
The HBase was designed to run on top of HDFS and provides BigTable like capabilities.

The HBase is written in Java,

For better understanding, let us take an example.
You have billions of customer emails and you need to find out the number of customers who has used the word complaint in their emails.
The request needs to be processed quickly (i.e. at real time). So, here we are

handling a large data set while retrieving a small amount of data.
For solving these kind of problems, HBase was designed.

■MapReduce is a core component of Hadoop Ecosystem.

■It is a software framework which processes large data sets using distributed and parallel algorithms.

■In a MapReduce program, Map() and Reduce() are two functions.

1.*The Map function performs actions like* ***filtering, grouping and sorting.***

2.*While Reduce function* ***aggregates and summarizes the result*** *produced by map function.*

3.*The result generated by the Map function is a* ***key value pair (K, V)*** *which acts as the input for Reduce function.*

■PIG has two parts:

1.*Pig Latin, the language and*

2.*the pig runtime, for the execution environment.*

■You can better understand it as Java and JVM.

■It supports pig Latin language, which has SQL like command structure.

10 line of pig Latin = approx. 200 lines of Map-Reduce Java code.

■The compiler internally converts pig Latin to MapReduce.

■It produces a sequential set of MapReduce jobs.

■PIG was initially developed by Yahoo.

■It gives you a platform for building data flow for ETL (Extract, Transform and Load), processing and analyzing huge data sets.

■Facebook created HIVE for people who are fluent with SQL.

■HIVE is a data warehousing component which performs reading, writing and managing large data sets in a distributed environment using SQL-like interface.

■HIVE + SQL = HQL

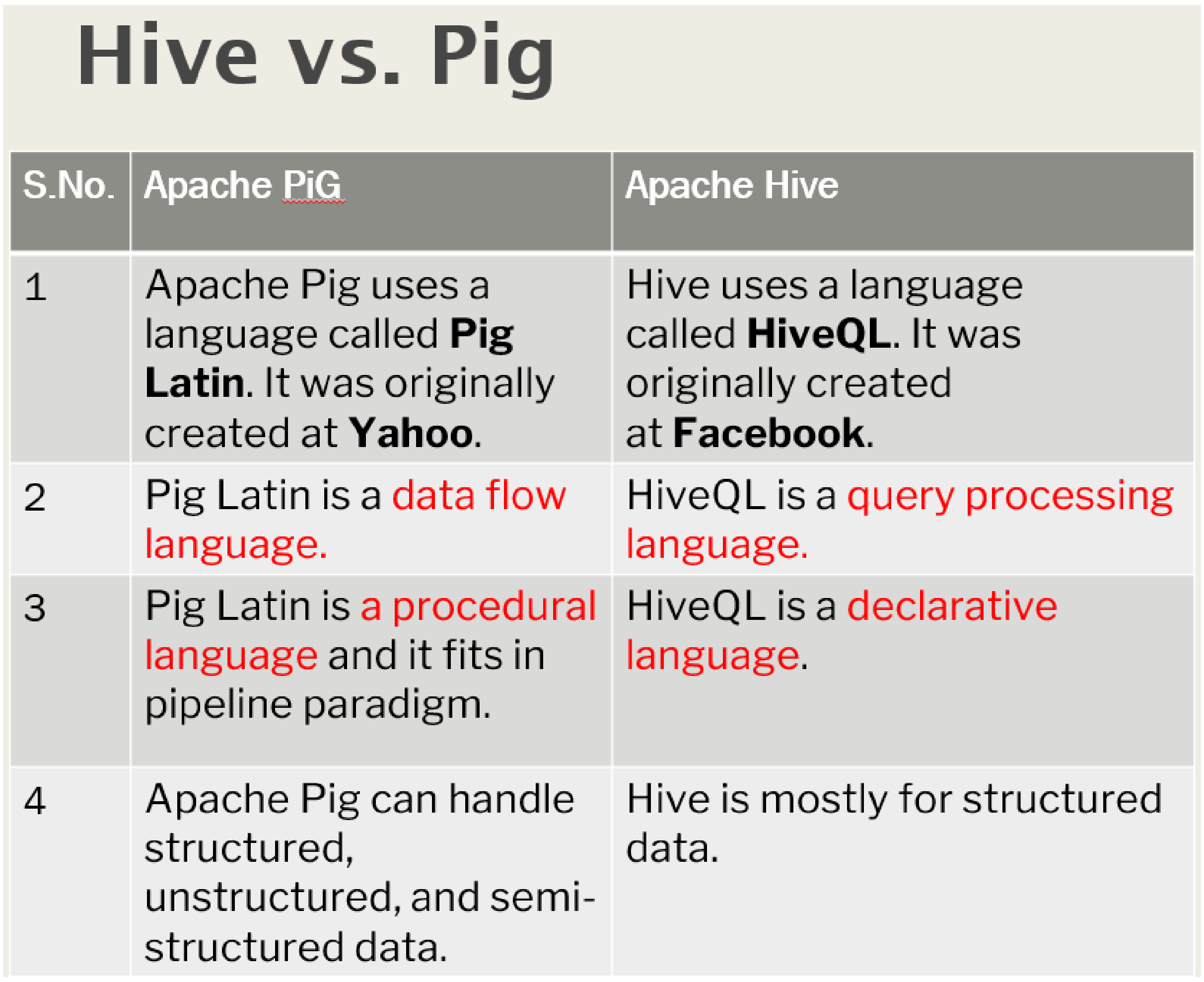
■The query language of Hive is called Hive Query Language(HQL), which is very similar like SQL.

■It has 2 basic components: Hive Command Line and JDBC/ODBC driver.

■ The Hive Command line interface is used to execute HQL commands.

■While, Java Database Connectivity (JDBC) and Object Database Connectivity (ODBC) is used to establish connection from data storage.

■Hive is highly scalable. As, it can serve both the purposes, i.e. large data set processing (i.e. Batch query processing) and real time processing (i.e. Interactive query processing).



Hive vs. Pig

S.No.	Apache PiG	Apache Hive
1	Apache Pig uses a language called **Pig Latin**. It was originally created at **Yahoo**.	Hive uses a language called **HiveQL**. It was originally created at **Facebook**.
2	Pig Latin is a data flow language.	HiveQL is a query processing language.
3	Pig Latin is a procedural language and it fits in pipeline paradigm.	HiveQL is a declarative language.
4	Apache Pig can handle structured, unstructured, and semi-structured data.	Hive is mostly for structured data.

▪ Apache Mahout is **a library of scalable machine-learning algorithms**, implemented on top of Apache Hadoop.

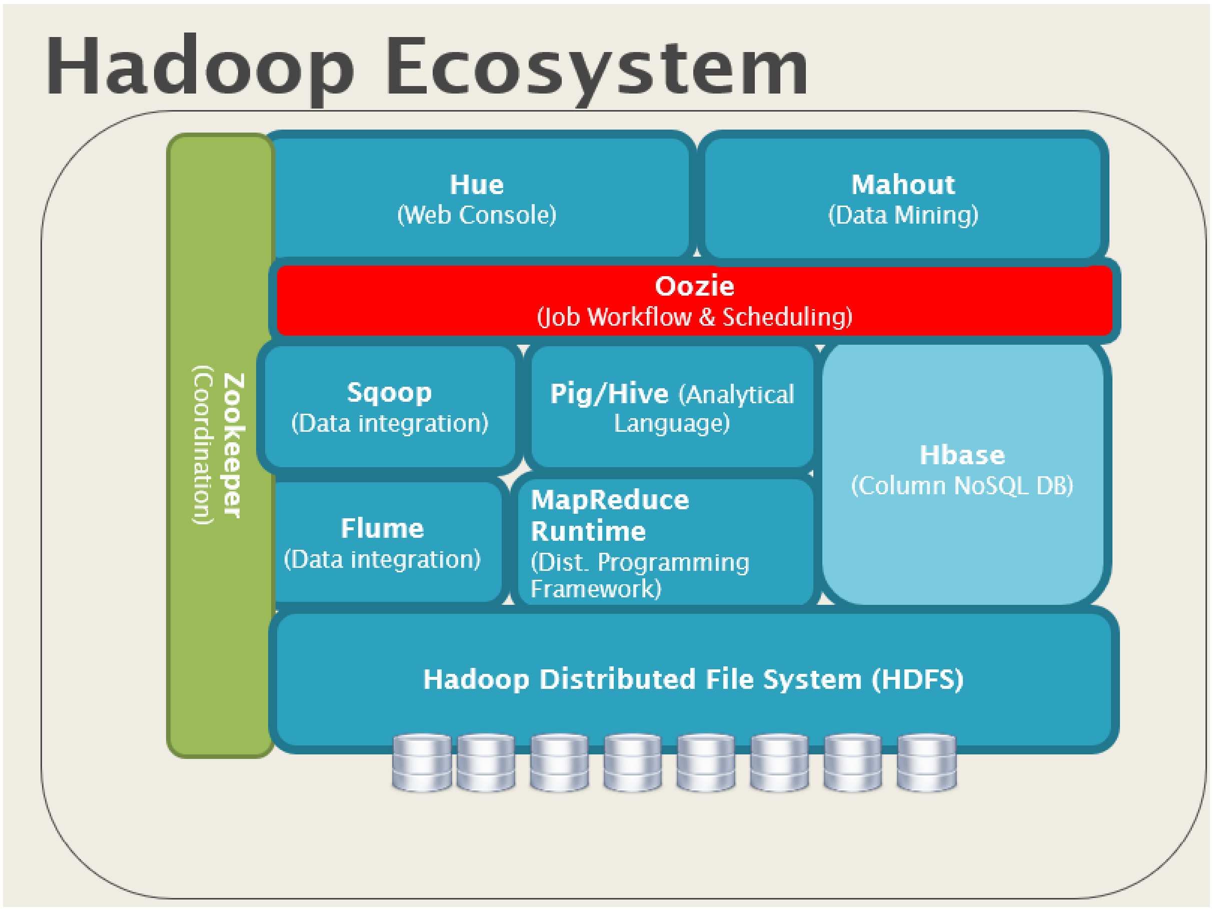
▪What Mahout does?

▪Introduction to apache mahout

▪

▪It performs collaborative filtering, clustering and classification. Some people also consider frequent item set missing as Mahout's function.

- **Collaborative filtering:** mines user behavior and makes product recommendations (e.g. Amazon recommendations)
- **Clustering:** It organizes a similar group of data together like articles can contain blogs, news, research papers etc.



Sqoop is a tool designed to transfer data between Hadoop and relational database servers.
It is used to import data from relational databases such as MySQL, Oracle to Hadoop HDFS, and export from Hadoop file system to relational databases.

■Apache Zookeeper coordinates with various services in a distributed environment.

■The services earlier had many problems with interactions like common configuration while synchronizing data. Even if the services are configured, changes in the configurations of the services

■Zookeeper performs **synchronization, configuration maintenance, grouping and naming.**

Hadoop limitations

- There is a single NameNode which is thus the single point of failure.
- It works with only MapReduce.
- MapReduce work as resource manager.
- It is only suitable for Batch Processing of Huge amount of Data, which is already in Hadoop System
- It is not suitable for Real-time Data Processing.
- It is not suitable for Data Streaming.
- It supports upto 4000 Nodes per Cluster.

Why does HDFS (Hadoop Distributed File System) employ data replication, and how does this feature contribute to the system's overall reliability and performance?

In HDFS (Hadoop Distributed File System), data replication is a key feature designed to ensure fault tolerance and high availability, rather than to cause data redundancy or duplication in a problematic sense. Here's why data replication is pursued in HDFS:

1. Fault Tolerance

- **Objective**: The primary purpose of data replication in HDFS is to provide fault tolerance. By replicating each block of data across multiple nodes, HDFS ensures that if a node fails, the data is still accessible from other nodes.
- **Example**: If a node storing a replica of a block fails, the system can still retrieve that block from another replica, ensuring that data is not lost.

2. High Availability

- **Objective**: Data replication enhances the availability of data. It ensures that data is available even if some nodes are down or unreachable.

- **Example**: During hardware failures or maintenance, data remains accessible because there are multiple replicas across different nodes.

3. Improved Read Performance

- **Objective**: Replication can also improve read performance by allowing data to be read from the nearest replica. This can reduce the load on any single node and speed up data retrieval.
- **Example**: If data is replicated across different nodes, a read request can be served by the replica that is closest to the requestor, reducing network latency.

4. Data Integrity

- **Objective**: By replicating data, HDFS can verify and maintain data integrity. If a replica becomes corrupted or inconsistent, the system can replace it with a correct version from another replica.
- **Example**: HDFS periodically checks the integrity of data and repairs corrupted replicas using the copies stored on other nodes.

5. System Scalability

- **Objective**: Replication supports the scaling of storage and computing resources. As the system grows, new nodes can be added, and data replication can be adjusted to balance the load and ensure redundancy.
- **Example**: When new nodes are added to the cluster, HDFS can automatically adjust replication levels to optimize resource utilization and data protection.

Trade-Offs

- **Data Redundancy**: While data replication does create redundancy, this is a deliberate trade-off. The cost of additional storage is outweighed by the benefits of data durability, availability, and reliability.
- **Storage Overhead**: Each replica increases storage requirements, but this overhead is considered acceptable given the benefits of ensuring that data is always available and resilient to failures.

In summary, data replication in HDFS is pursued to enhance fault tolerance, ensure high availability, improve read performance, maintain data integrity, and support system scalability. Although it introduces redundancy, the benefits of data protection and reliability make it a crucial feature of HDFS.

NOSQL

What is NoSQL? (2 Marks)

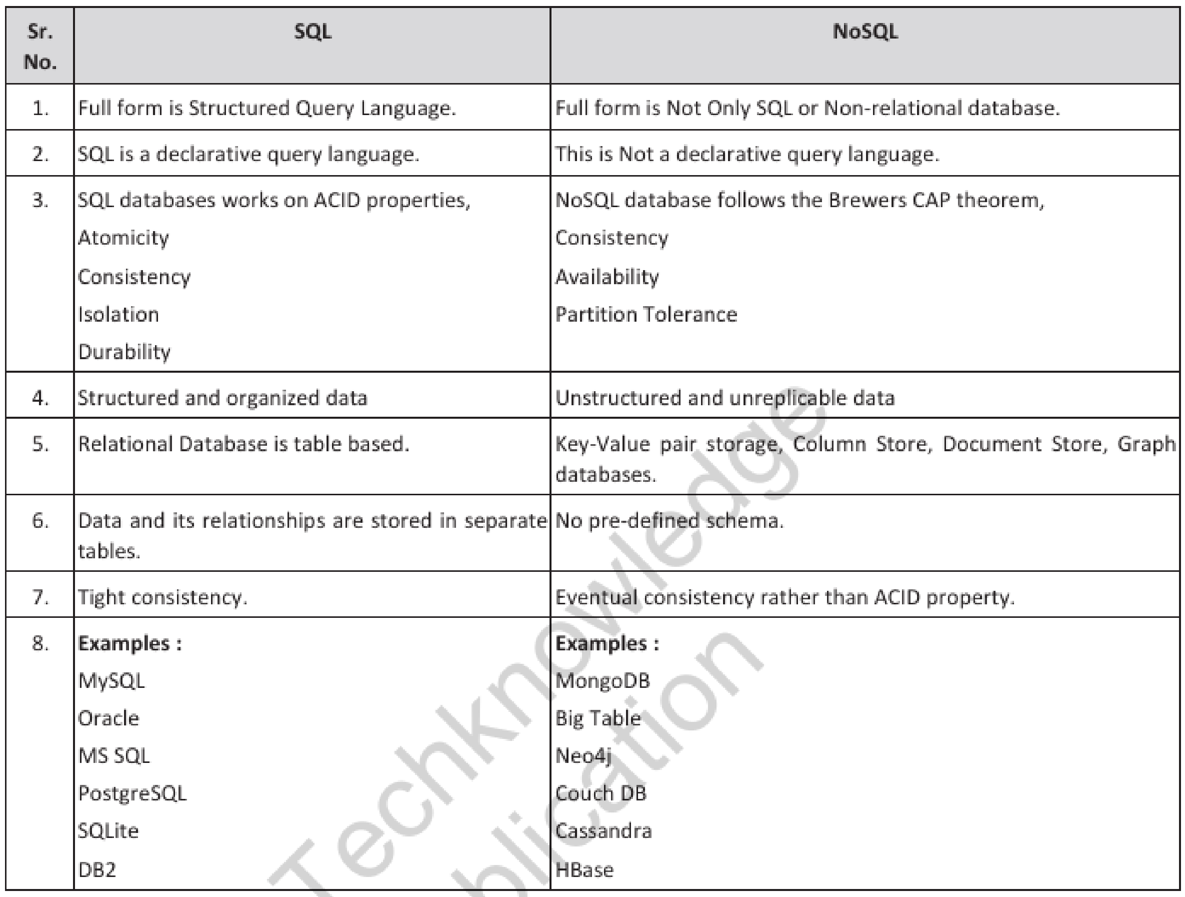
Answer:

1. **Definition and Overview**:
    - NoSQL refers to a category of database systems that do not use SQL as their primary query language. Unlike traditional relational databases, NoSQL databases are designed to handle large volumes of distributed data without the constraints of table schemas.
    - They are optimized for specific use cases, such as handling large amounts of unstructured or semi-structured data, and typically avoid join operations.
2. **History**:
    - The term "NoSQL" was first coined by Carlo Strozzi in 1998 to describe a database system that did not support SQL queries.
    - The concept gained broader attention and adoption in the early 2000s, particularly around 2009, when NoSQL databases started being actively discussed and implemented at technology conferences.
3. **Advantages**:
    - **Good Resource Scalability**: NoSQL databases can scale horizontally by adding more servers to handle increased load.
    - **Lower Operational Cost**: They often use commodity hardware and are less expensive to maintain.
    - **Supports Semi-Structured Data**: Capable of handling data that does not fit neatly into tables, such as JSON or XML.
    - **No Static Schema**: Allows for flexible data models without requiring predefined schemas.

- **Supports Distributed Computing**: Designed to work across multiple servers or data centers.
- **Faster Data Processing**: Often optimized for quick read and write operations.
- **No Complicated Relationships**: Avoids the complexities of relational data and joins.
- **Relatively Simple Data Models**: Offers straightforward ways to model data, such as key-value pairs, documents, or columns.

4. **Disadvantages**:
   - **Not a Defined Standard**: NoSQL lacks a universal standard, which can lead to inconsistencies between different systems.
   - **Limited Query Capabilities**: Often provides fewer querying features compared to traditional SQL databases.
5. **Companies Using NoSQL**:
   - **Google**
   - **Facebook**
   - **LinkedIn**
   - **McGraw-Hill Education**

Comparison between SQL and NoSQL



Sr. No.	SQL	NoSQL
1.	Full form is Structured Query Language.	Full form is Not Only SQL or Non-relational database.
2.	SQL is a declarative query language.	This is Not a declarative query language.
3.	SQL databases works on ACID properties, Atomicity Consistency Isolation Durability	NoSQL database follows the Brewers CAP theorem, Consistency Availability Partition Tolerance
4.	Structured and organized data	Unstructured and unreplicable data
5.	Relational Database is table based.	Key-Value pair storage, Column Store, Document Store, Graph databases.
6.	Data and its relationships are stored in separate tables.	No pre-defined schema.
7.	Tight consistency.	Eventual consistency rather than ACID property.
8.	**Examples :** MySQL Oracle MS SQL PostgreSQL SQLite DB2	**Examples :** MongoDB Big Table Neo4j Couch DB Cassandra HBase

CAP Theorem (Brewer's Theorem)

The CAP theorem states that a distributed database system can only achieve two out of the following three properties simultaneously:

1. **Consistency (C)**
    - The database remains in a consistent state after any operation. All nodes see the same data at the same time, ensuring that any read operation returns the most recent write.
2. **Availability (A)**
    - The database is always available for read and write operations, even if some nodes are down. Every request receives a response, regardless of the state of other nodes.
3. **Partition Tolerance (P)**

- The system continues to function and process requests even if there is a network partition or communication failure between nodes. The system can handle network splits and still operate.

Note: It is challenging to achieve all three properties at the same time, so distributed databases must make trade-offs. Here are the possible combinations:

- **CA (Consistency + Availability)**:
  - Typically suitable for a single-site cluster where all nodes are always in contact.
  - Partitioning issues can block the system or make it unavailable during network splits.
- **CP (Consistency + Partition Tolerance)**:
  - The system may not always be available during network partitions, but it ensures data consistency. Some data may be temporarily inaccessible but accurate once it becomes available.
- **AP (Availability + Partition Tolerance)**:
  - The system remains available even during network partitions, but some data might be inconsistent. Eventually, the system will reach a consistent state.

BASE Model

The BASE model is an alternative to the ACID model used in relational databases. It is designed to handle the characteristics of NoSQL databases:

1. **Basic Availability**
   - The system remains available for most operations, even under failure conditions.
2. **Soft State**
   - The state of the system may change over time, even without new input, due to eventual consistency.
3. **Eventual Consistency**
   - The system guarantees that, given enough time, all updates will propagate and all nodes will eventually converge to a consistent state.

Data Storage in NoSQL

- **Key/Value Store**: NoSQL databases often use a key/value store model, where each key is associated with a value. This allows for efficient data retrieval without schema restrictions.
- **Schema Flexibility**: NoSQL databases do not enforce a fixed schema, allowing for flexible data storage where different keys can have different associated values.

Redundancy and Scalability

- **Redundancy**: To ensure data reliability, NoSQL databases often employ replication, where data is duplicated across multiple nodes. Adding duplicate nodes and configuring replication provides redundancy.
- **Scalability**: NoSQL databases can scale horizontally by adding additional nodes to handle increased load. Data is distributed across nodes using hash functions or other partitioning methods to balance the load.

In summary, NoSQL databases are designed to handle large volumes of distributed data with flexible schema requirements, while making trade-offs in consistency, availability, and partition tolerance as defined by the CAP theorem. They use the BASE model to manage database transactions and rely on techniques like replication and horizontal scaling to achieve redundancy and scalability.

NoSQL Data Architecture Patterns

Architecture Pattern is a logical way of categorizing data that will be stored on the Database. **NoSQL** is a type of database which helps to perform operations on big data and store it in a valid format. It is widely used because of its flexibility and a wide variety of services.

Architecture Patterns of NoSQL:

The data is stored in NoSQL in any of the following four data architecture patterns.

```
1. Key-Value Store Database
2. Column Store Database
```

```
3. Document Database
4. Graph Database
```

These are explained as following below.

1. Key-Value Store Database:

This model is one of the most basic models of NoSQL databases. As the name suggests, the data is stored in form of Key-Value Pairs. The key is usually a sequence of strings, integers or characters but can also be a more advanced data type. The value is typically linked or co-related to the key. The key-value pair storage databases generally store data as a hash table where each key is unique. The value can be of any type (JSON, BLOB(Binary Large Object), strings, etc). This type of pattern is usually used in shopping websites or e-commerce applications.

Advantages:

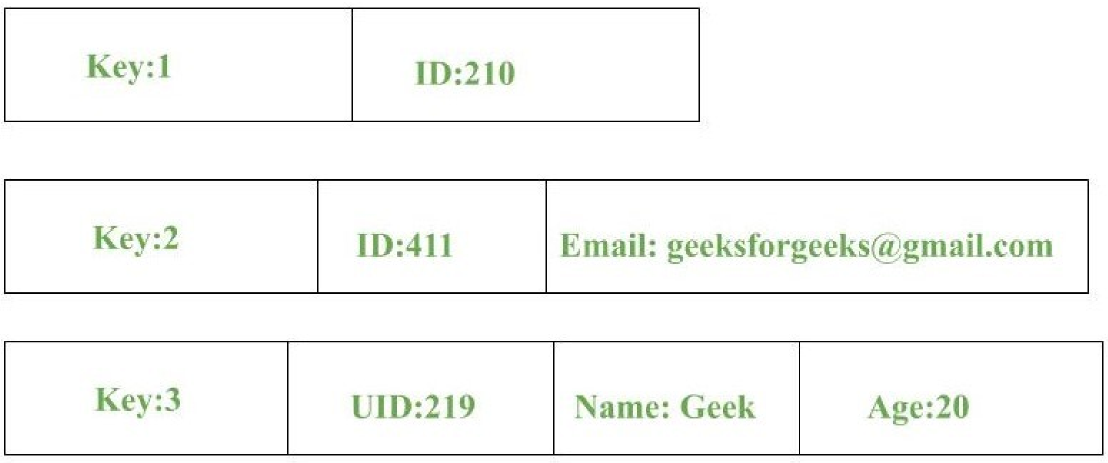
- Can handle large amounts of data and heavy load,
- Easy retrieval of data by keys.

Limitations:

- Complex queries may attempt to involve multiple key-value pairs which may delay performance.
- Data can be involving many-to-many relationships which may collide.

Examples:

- DynamoDB
- Berkeley DB



2. Column Store Database:

Rather than storing data in relational tuples, the data is stored in individual cells which are further grouped into columns. Column-oriented databases work only on columns. They store large amounts of data into columns together. Format and titles of the columns can diverge from one row to other. Every column is treated separately. But still, each individual column may contain multiple other columns like traditional databases.

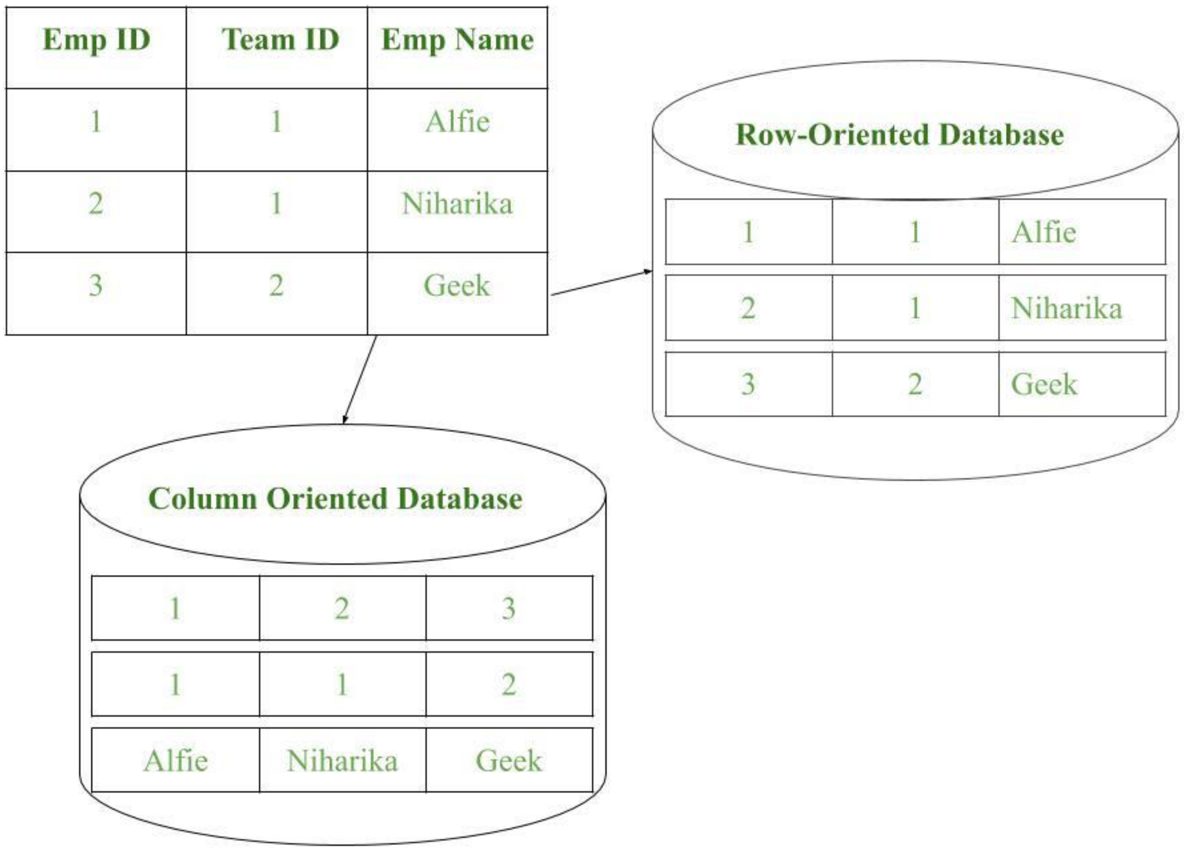
Basically, columns are mode of storage in this type.

Advantages:

- Data is readily available
- Queries like SUM, AVERAGE, COUNT can be easily performed on columns.

Examples:

- HBase
- Bigtable by Google
- Cassandra



3. Document Database:

The document database fetches and accumulates data in form of key-value pairs but here, the values are called as Documents. Document can be stated as a complex data structure. Document here can be a form of text, arrays, strings, JSON, XML or any such format. The use of nested documents is also very common. It is very effective as most of the data created is usually in form of JSONs and is unstructured.

Advantages:

- This type of format is very useful and apt for semi-structured data.
- Storage retrieval and managing of documents is easy.

Limitations:

- Handling multiple documents is challenging
- Aggregation operations may not work accurately.

Examples:

- MongoDB
- CouchDB

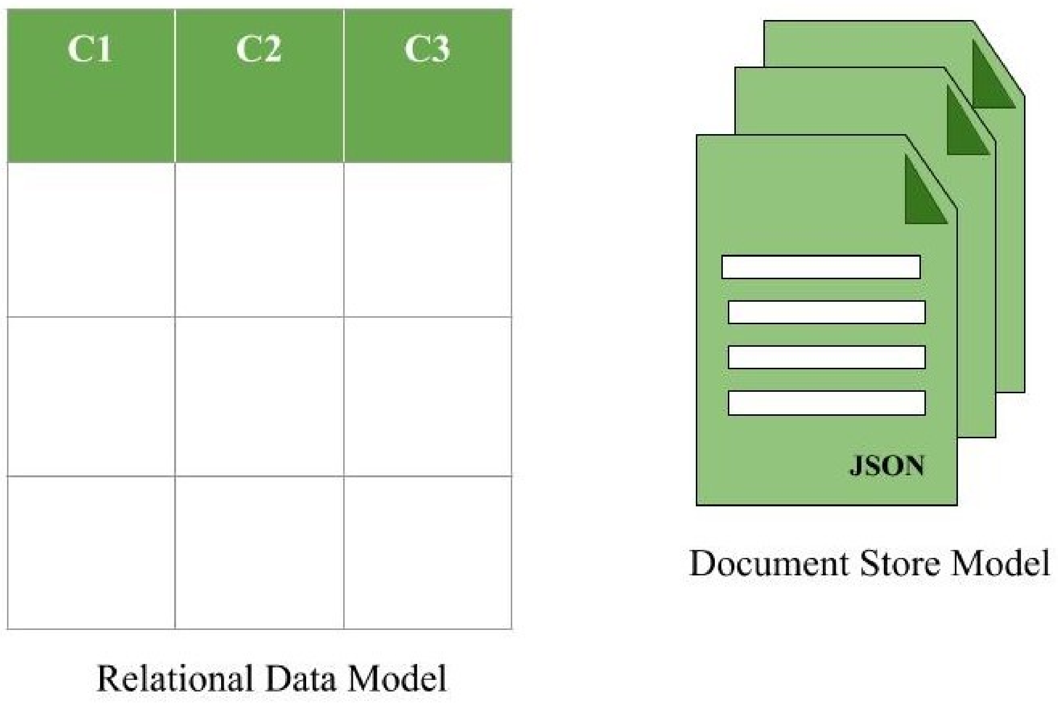


Figure – Document Store Model in form of JSON documents

4. Graph Databases:

Clearly, this architecture pattern deals with the storage and management of data in graphs. Graphs are basically structures that depict connections between two or more objects in some data. The objects or entities are called as nodes and are joined together by relationships called Edges. Each edge has a unique identifier. Each node serves as a point of contact for the graph. This pattern is very commonly used in social networks where there are a large number of entities and each entity has one or many characteristics which are connected by edges. The relational database pattern has tables that are loosely connected, whereas graphs are often very strong and rigid in nature.

Advantages:

- Fastest traversal because of connections.
- Spatial data can be easily handled.

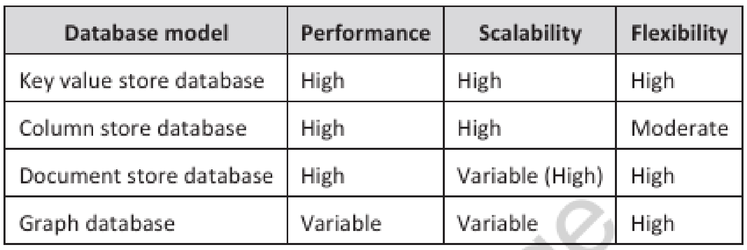
Limitations:

Wrong connections may lead to infinite loops.

Examples:

- Neo4J
- FlockDB(Used by Twitter)

5. Comparison of NoSQL variations



Database model	Performance	Scalability	Flexibility
Key value store database	High	High	High
Column store database	High	High	Moderate
Document store database	High	Variable (High)	High
Graph database	Variable	Variable	High

Benefits of NoSQL

1. **Scalability**:
    - **High**: NoSQL databases are designed to scale horizontally, which means they can handle large volumes of data by adding more servers. This scalability is crucial for handling big data and high-traffic applications.

2. **Flexibility**:
    - **Variable**: NoSQL databases do not enforce a fixed schema, allowing for flexible data models. This adaptability is beneficial for managing unstructured or semi-structured data and evolving application requirements.
3. **High Performance**:
    - **High**: NoSQL databases often provide faster read and write operations due to their distributed nature and optimized data access patterns. This performance is essential for real-time applications and large-scale data processing.
4. **High Availability**:
    - **High**: NoSQL databases are designed to work in distributed environments, ensuring that data is available across multiple servers. This availability reduces the risk of downtime and ensures continuous access to data.

Big Data Analytics

- **Promotion of NoSQL**: The rise of big data has significantly contributed to the growth and popularity of NoSQL databases. NoSQL systems are well-suited for handling the volume, velocity, and variety of big data.
- **Handling Big Data**: NoSQL databases provide robust mechanisms for managing large datasets, making them ideal for applications requiring real-time analysis and processing of big data.

Location Independence

- **Global Access**: NoSQL databases can operate independently of the physical location of the data. This means that data can be read from and written to the database regardless of where the database server is located, supporting global and distributed applications.

Choosing Distribution Models: Master-Slave vs. Peer-to-Peer

Master-Slave Model

- **Description**:

    - In the Master-Slave distribution model, one node (the master) controls the database operations and decision-making processes. This master node is responsible for task assignments, data storage, data retrieval, and manipulation.
    - The master node delegates tasks to other nodes (slaves) within the cluster. The slave nodes execute these tasks and report back to the master node.
- **Advantages**:
    - **Centralized Control**: The master node has full control over job assignments and data management, which simplifies coordination and management.
    - **Consistency**: Since the master node governs all changes, it helps in maintaining consistency and reducing conflicts.
- **Disadvantages**:
    - **Single Point of Failure**: The master node is a critical point in the system. If the master fails, the entire system might be impacted unless there is a failover mechanism.
    - **Scalability Limits**: The master node may become a bottleneck as the load increases, potentially affecting performance and scalability.
- **Use Case**:
    - Databases like **HBase** use the master-slave model, where the master node handles administrative tasks and the slave nodes perform the actual data operations.

Peer-to-Peer Model

- **Description**:
    - In the Peer-to-Peer distribution model, all nodes have equal roles and responsibilities. Each node can perform all operations, including Create, Read, Update, Delete, and configuration tasks.
    - There is no central authority; each node communicates directly with other nodes. This model often includes mechanisms for data replication to ensure reliability and fault tolerance.
- **Advantages**:

    - **Fault Tolerance**: Since there is no single point of failure, the system can continue to operate even if one or more nodes fail. Data replication across nodes helps prevent data loss.
    - **Scalability**: Adding more nodes to the system increases its capacity and performance. The load is distributed among all nodes, avoiding bottlenecks.
- **Disadvantages**:
    - **Complexity**: Managing and coordinating operations can be more complex in a peer-to-peer model due to the lack of centralized control.
    - **Consistency Challenges**: Ensuring consistency across all nodes can be more challenging, especially in systems with high write volumes or frequent data changes.
- **Use Case**:
    - Databases like **Cassandra** use the peer-to-peer model, where all nodes are equal and can handle any request. This model is suitable for distributed environments with high availability and fault tolerance requirements.

Aspect	**Master-Slave Model**	**Peer-to-Peer Model**
Architecture	One master node and multiple slave nodes.	All nodes are peers with equal roles.
Node Roles	Master node handles all administrative and coordination tasks; slave nodes execute tasks assigned by the master.	All nodes can perform any task, including administrative tasks.
Control	Centralized control by the master node.	Decentralized control; no single point of authority.
Fault Tolerance	Single point of failure (the master node) may affect the system; requires failover mechanisms.	High fault tolerance; the system continues to operate even if some nodes fail.

Scalability	Limited scalability due to potential bottleneck at the master node.	High scalability as load is distributed across all nodes.
Consistency	Easier to maintain consistency due to centralized control.	More complex to ensure consistency across all nodes.
Data Replication	Not inherently built into the model; typically handled by the master node if required.	Data is replicated across multiple nodes to ensure fault tolerance.
Performance	Performance can be affected if the master node becomes a bottleneck.	Performance is distributed; can handle high loads better.
Configuration	The master node manages configuration and coordination.	Configuration and coordination are distributed across nodes.
Use Cases	Suitable for systems where centralized control is desired and where consistency is critical.	Ideal for distributed systems requiring high availability and fault tolerance.
Example Databases	**HBase**, **MySQL** (with replication setup).	**Cassandra**, **MongoDB** (in certain configurations).

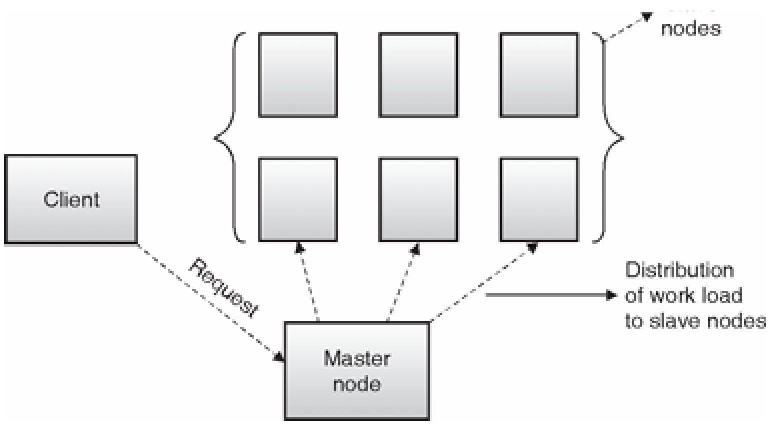


Fig. 4.11.1 : Master slave model

ig. 4.11.2 shows Peer to Peer model.

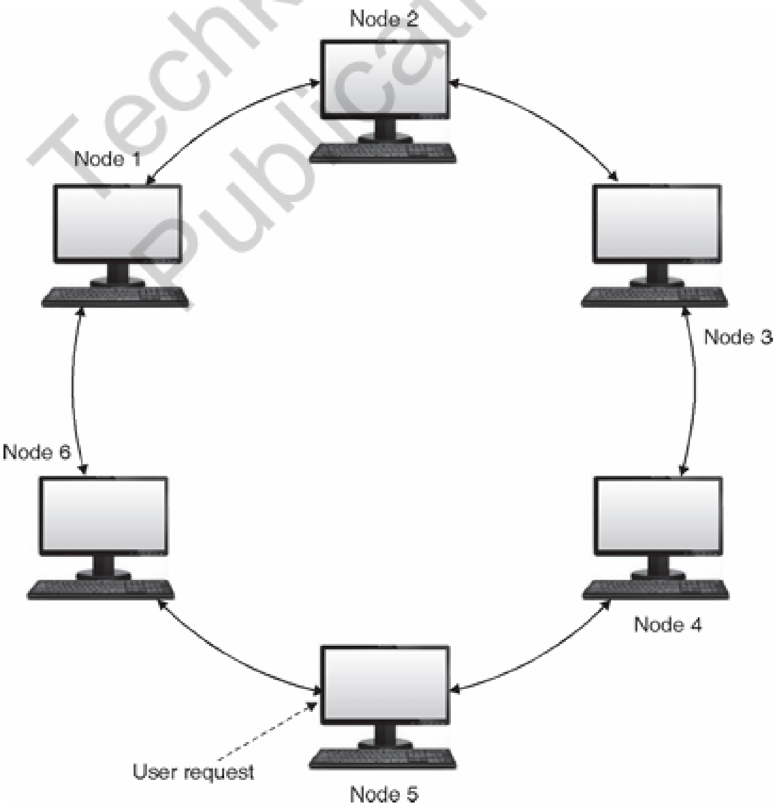


Fig. 4.11.2 : Peer to peer model