# BDAV [1,2,3]

02 October 2023 11:54

**UNIT - 1**

---

## Big Data

**Introduction**

Big data refers to extremely large and complex sets of data that cannot be managed, processed, or analyzed with traditional data processing tools.
Normally we work on data of size MB(Word Doc ,Excel) or maximum GB(Movies, Codes) but data in Peta bytes i.e. 10^15 byte size is called Big Data.

Let us quickly understand about resources of Big Data.

- Social Networking Sites
- E-Commerce Sites
- Weather Station
- Telecom Company

**Issues of Big Data**

Huge amount of unstructured data which needs to be stored, processed and analyzed.

**Solution**

**Storage:** This huge amount of data, Hadoop uses HDFS (Hadoop Distributed File System) which uses commodity hardware to form clusters and store data in a distributed fashion. It works on Write once, read many times principle.
**Processing:** Map Reduce paradigm is applied to data distributed over network to find the required output.
**Analyze:** Pig, Hive can be used to analyze the data.
**Cost:** Hadoop is open source so the cost is no more an issue.

**Characterstics of Big Data:**

- Volume:
  Big data hota hai large and complex dataset jisko traditional methods se manage karna mushkil hota hai. Isme hum large volume of data ke sath deal karte hai.

- Velocity:
  It is generated at high speed in real time . Data can be continuously streaming in from various sources, such as social media , sensors etc.

- Variety:
  It includes different types of data from different resources such as structured , semi-structured and unstructured data. Ye different type of data ko analyze karna mushkil ho jata hai.

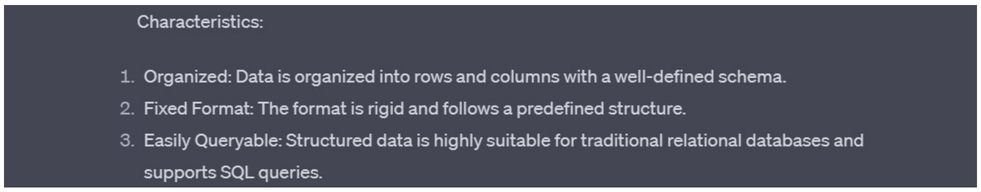
- Veracity:
  This refers to the quality and reliability of data. bhut bar large amount of data mai noisy data bhi aa jata hai.

- Value:
  the ultimate goal of working with large amount of data is to extract the value from it. isme different data analytics techniques use hoti hai jaise ki machine learning , data mining etc.

**Types of Big Data**

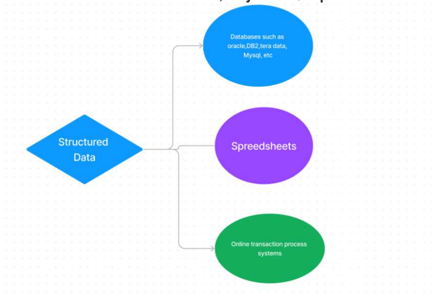
Structured Data:

- Data which is organized in fixed format with predefined schema is structured data.
- Ye data highly organized hota hai and retrieval bhot fast hota as data is stored in structured way.
- Ye data mostly relational databases mai store hota hai in the form of rows and columns which follows the structure of table.


Characteristics:

1. Organized: Data is organized into rows and columns with a well-defined schema.
2. Fixed Format: The format is rigid and follows a predefined structure.
3. Easily Queryable: Structured data is highly suitable for traditional relational databases and supports SQL queries.

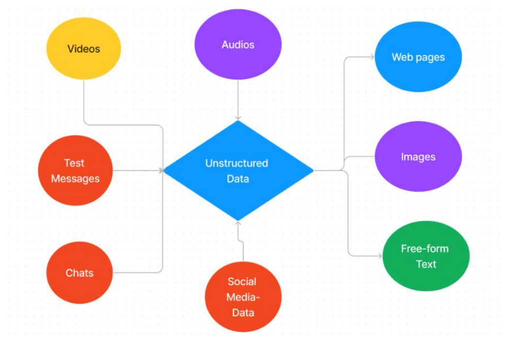
Example/Sources of Structured Data:
Database such as oracle, MySQL | Spreadsheets | Online transaction process system

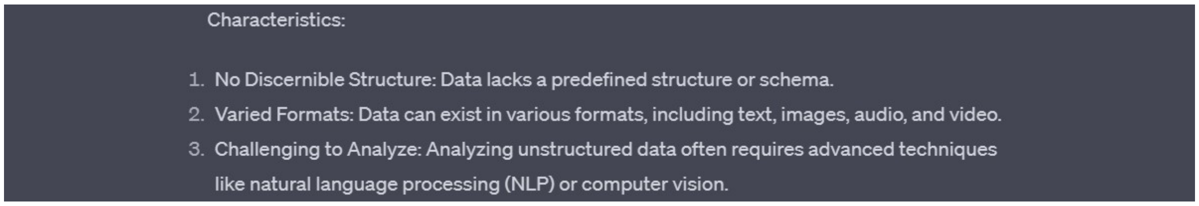

Structured Data
Databases such as oracle,DB2,tera data, Mysql, etc
Spreedsheets
Online transaction process systems

Unstructured Data

- Data which does not have predefined structure is called unstructured data.
- They are in the form of text, images, audio, video etc.
- Unstructured data is challenging to analyze using traditional data and searching particular data is also difficult in such type of unstructured data.

Example: Social Media Feed | Customer Reviews | Emails

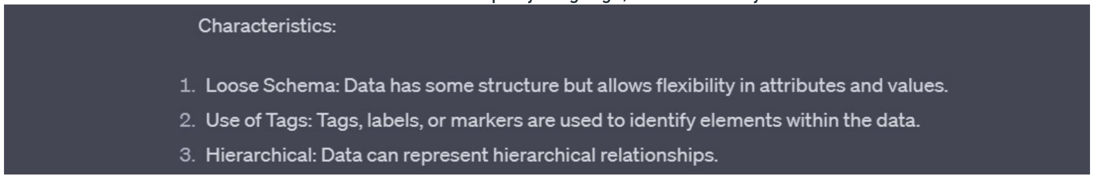



Characteristics:

1. No Discernible Structure: Data lacks a predefined structure or schema.
2. Varied Formats: Data can exist in various formats, including text, images, audio, and video.
3. Challenging to Analyze: Analyzing unstructured data often requires advanced techniques like natural language processing (NLP) or computer vision.

Semi-Structured Data

- It lies between structured and unstructured 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*.


Characteristics:

1. Loose Schema: Data has some structure but allows flexibility in attributes and values.
2. Use of Tags: Tags, labels, or markers are used to identify elements within the data.
3. Hierarchical: Data can represent hierarchical relationships.

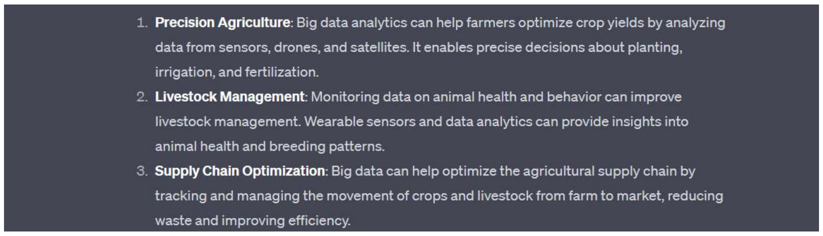
Example: XML files , JSON Documents , log files

## Traditional Data VS Big Data

| Traditional Data | Big Data |
|---|---|
| It is usually a small amount of data that can be collected and analyzed using traditional methods easily. | It is usually a big amount of data that cannot be processed and analyzed easily using traditional methods. |
| It is usually structured data and can be stored in spreadsheets, databases, etc. | It includes semi-structured, unstructured, and structured data. |
| It often collects data manually. | It collects information automatically with the use of automated systems. |
| It usually comes from internal systems. | It comes from various sources such as mobile devices, social media, etc. |
| It consists of data such as customer information, financial transactions, etc. | It consists of data such as images, videos, etc. |
| Analysis of traditional data can be done with the use of primary statistical methods. | Analysis of big data needs advanced analytics methods such as machine learning, data mining, etc. |
| Traditional methods to analyze data are slow and gradual. | Methods to analyze big data are fast and instant. |
| It generates data after the happening of an event. | It generates data every second. |
| It is typically processed in batches. | It is developed and processed in real-time. |
| It is limited in its value and insights. | It provides valuable insights and patterns for good decision-making. |
| It contains reliable and accurate data. | It may contain unreliable, inconsistent, or inaccurate data because of its size and complexity. |
| It is used for simple and small business processes. | It is used for complex and big business processes. |
| It does not provide in-depth insights. | It provides in-depth insights. |
| It is easy to secure and protect than big data because of its small size and simplicity. | It is harder to secure and protect than traditional data because of its size and complexity. |
| It requires less time and money to store traditional data. | It requires more time and money to store big data. |
| It can be stored on a single computer or server. | It requires distributed storage across numerous systems. |
| It is less efficient than big data. | It is more efficient than traditional data. |
| It can be managed in a centralized structure easily. | It requires a decentralized infrastructure to manage the data. |

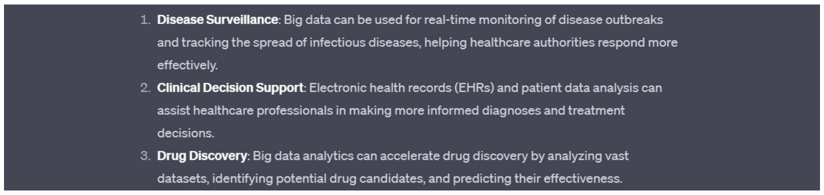
## Application of Big Data

### 1. Agriculture



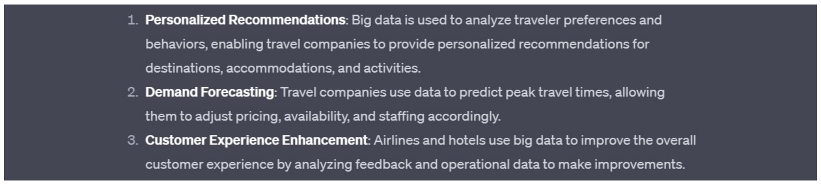
1. **Precision Agriculture**: Big data analytics can help farmers optimize crop yields by analyzing data from sensors, drones, and satellites. It enables precise decisions about planting, irrigation, and fertilization.
2. **Livestock Management**: Monitoring data on animal health and behavior can improve livestock management. Wearable sensors and data analytics can provide insights into animal health and breeding patterns.
3. **Supply Chain Optimization**: Big data can help optimize the agricultural supply chain by tracking and managing the movement of crops and livestock from farm to market, reducing waste and improving efficiency.

### 2. Healthcare



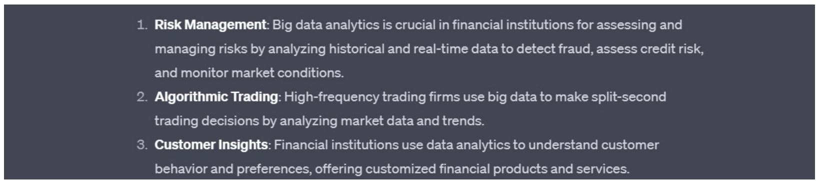
1. **Disease Surveillance**: Big data can be used for real-time monitoring of disease outbreaks and tracking the spread of infectious diseases, helping healthcare authorities respond more effectively.
2. **Clinical Decision Support**: Electronic health records (EHRs) and patient data analysis can assist healthcare professionals in making more informed diagnoses and treatment decisions.
3. **Drug Discovery**: Big data analytics can accelerate drug discovery by analyzing vast datasets, identifying potential drug candidates, and predicting their effectiveness.

### 3. Travel & Tourism



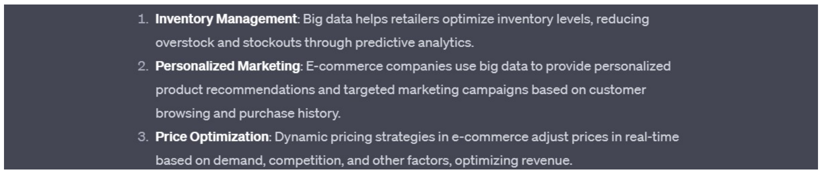
1. **Personalized Recommendations**: Big data is used to analyze traveler preferences and behaviors, enabling travel companies to provide personalized recommendations for destinations, accommodations, and activities.
2. **Demand Forecasting**: Travel companies use data to predict peak travel times, allowing them to adjust pricing, availability, and staffing accordingly.
3. **Customer Experience Enhancement**: Airlines and hotels use big data to improve the overall customer experience by analyzing feedback and operational data to make improvements.

### 4. Finance



1. **Risk Management**: Big data analytics is crucial in financial institutions for assessing and managing risks by analyzing historical and real-time data to detect fraud, assess credit risk, and monitor market conditions.
2. **Algorithmic Trading**: High-frequency trading firms use big data to make split-second trading decisions by analyzing market data and trends.
3. **Customer Insights**: Financial institutions use data analytics to understand customer behavior and preferences, offering customized financial products and services.

### 5. Retail & E-Commerce

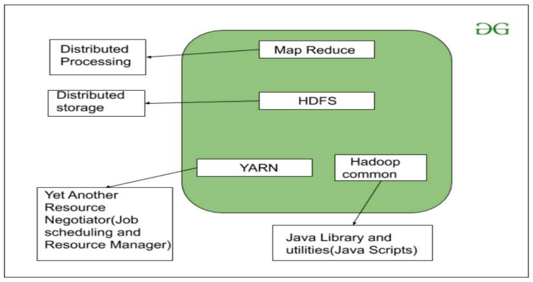


1. **Inventory Management**: Big data helps retailers optimize inventory levels, reducing overstock and stockouts through predictive analytics.
2. **Personalized Marketing**: E-commerce companies use big data to provide personalized product recommendations and targeted marketing campaigns based on customer browsing and purchase history.
3. **Price Optimization**: Dynamic pricing strategies in e-commerce adjust prices in real-time based on demand, competition, and other factors, optimizing revenue.

## HADOOP Architecture

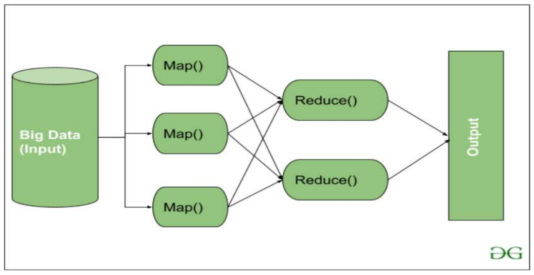
- Hadoop is an open source framework from Apache and is used to store process and analyze data which are very huge in volume. Hadoop is written in Java and is not OLAP (online analytical processing).
- It provides scalable, reliable and cost-effective solution for handling big data.
- Today lots of companies are using Hadoop in their organization to deal with big data. Ex: Facebook, Yahoo, Google, Twitter, LinkedIn and many more.

**Mainly 4 Components:**
Map Reduce | HDFS | YARN | Hadoop Common


Distributed Processing
Distributed storage
Map Reduce
HDFS
YARN
Hadoop common
Yet Another Resource Negotiator(Job scheduling and Resource Manager)
Java Library and utilities(Java Scripts)

**Map Reduce:**

- ◻ MapReduce is an algorithm that is based on YARN framework.
- ◻ The major feature of MapReduce is to perform the distributed processing in parallel in a Hadoop cluster which Makes Hadoop working so fast. When you are dealing with Big Data, serial processing is no more of any use.
- ◻ MapReduce has mainly 2 tasks which are divided phase-wise



*Here, we can see that the Input is provided to the Map() function then it's output is used as an input to the Reduce function and after that, we receive our final output.*

Map ko input as data diya jata hai . Map() use tuples mai break karta hai jo ki key value pair hote hai . Again who key value pair as a input diya jata hai reduce() ko. Then ye function sare key value pairs ko combine karta ha based on key value and tuple ka set create hota hai and uspe then operations perform hota hai jaise sorting , summation etc.

**HDFS:**

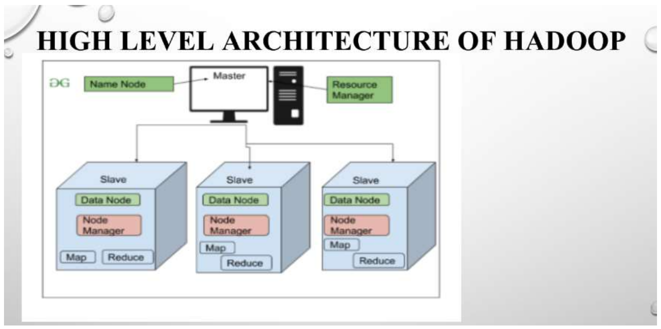
- ◇ HDFS stands for Hadoop distributed file system.
- ◇ Ye ek primary storage system hai Hadoop ka.
- ◇ HDFS divides large files into smaller blocks and copy of these send to different computers to ensure fault tolerance.
- ◇ It follows master slave architecture.

**YARN:**

- ◇ Ye ek framework hai jispe map reducer kam karta hai.
- ◇ Ye two operations perform karta hai job scheduling and resource management.
- ◇ The purpose of job schedular is to divide a big task into small jobs so that each job can be assign to particular slave.
- ◇ Job schedular also keep track of which job is important, which job has more priority.
- ◇ Resource manager is used to manage all resources that are made available for running a Hadoop cluster.

**Hadoop Common:**

- ◇ Hadoop common are java library and java files that we need for all the components present in Hadoop cluster.
- ◇ These utilities are used by HDFS , YARN and MapReduce for running the cluster.



--------------------------------------------------------------------------------------------------------------------------------

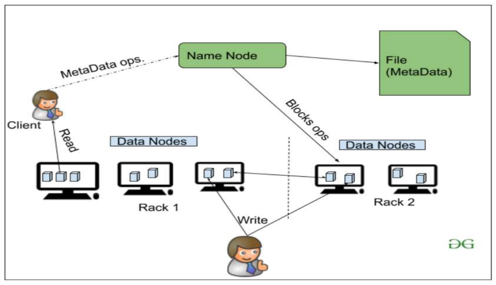
UNIT - 2

**HDFS**

The Hadoop Distributed File System (HDFS) is a distributed file system for Hadoop. It contains a master/slave architecture. This architecture consist of a single NameNode performs the role of master, and multiple DataNodes performs the role of a slave.

**Architecture:**

- HDFS ye ek primary storage system hai Hadoop ka , ye ek open source framework hai for distributed data processing.
- HDFS use hota hai to store and manage large data sets across the cluster of commodity hardware.
- Its architecture is built with the goal of providing fault tolerance , high availability and scalability of big data processing.



NameNode(Master) | DataNodes | File Block in HDFS | Replication in HDFS | Rack Awareness

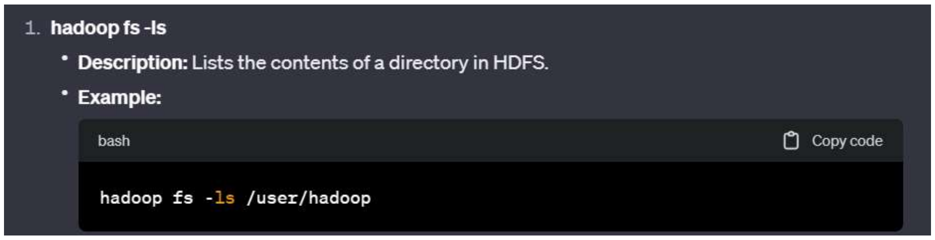
## NameNode

- Ye Master ke jaise kam karta hai and ye guide karta hai DataNodes ko in Hadoop.
- NameNode basically used to store metadata i.e. data about data.
- Meta data matlab jaise transaction logs that keep track of users activity in a Hadoop cluster.
- Meta data can also be the name of the file , size and the location of the data node that name node stores to find closest DataNodes for fast communication.
- NameNode instructs the DataNodes with the operations delete, create ,Replicate etc.

## DataNodes

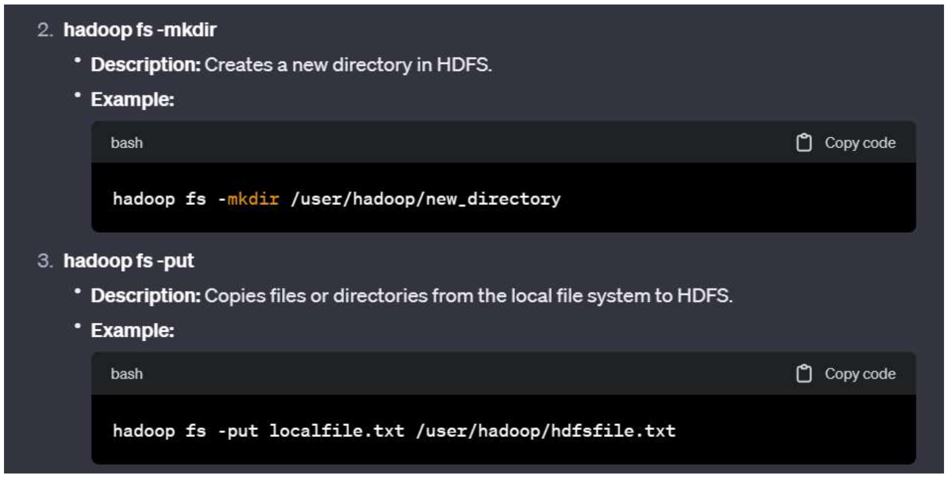
- DataNode work as a slave that listens to the NameNode.
- The number of datanodes can be 1 to 500 or even more than that.
- Jitne jyada DataNode honge utna jyada data Hadoop store kar payega.
- Data node ki storage capacity jyada honi chahiye to store large number of file blocks.

HDFS commands:



1. **hadoop fs -ls**
   - **Description:** Lists the contents of a directory in HDFS.
   - **Example:**

```bash
hadoop fs -ls /user/hadoop
```

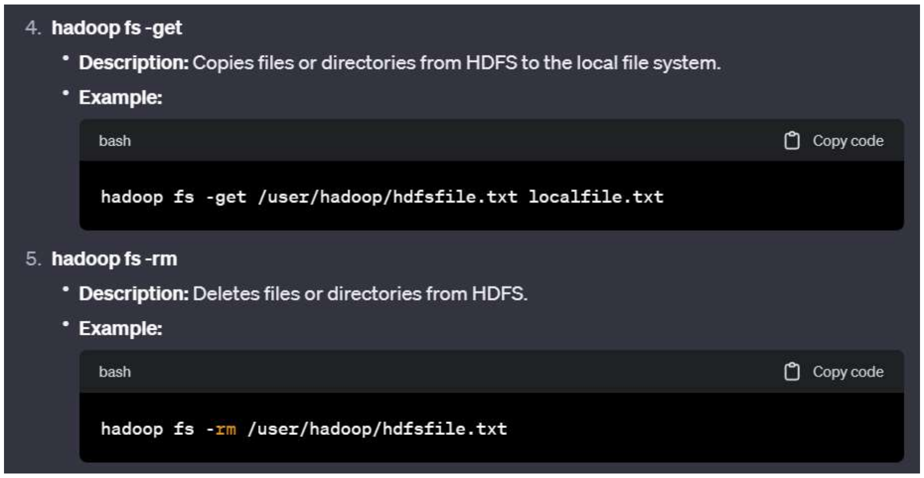


2. **hadoop fs -mkdir**
   - **Description:** Creates a new directory in HDFS.
   - **Example:**

```bash
hadoop fs -mkdir /user/hadoop/new_directory
```

3. **hadoop fs -put**
   - **Description:** Copies files or directories from the local file system to HDFS.
   - **Example:**

```bash
hadoop fs -put localfile.txt /user/hadoop/hdfsfile.txt
```


4. **hadoop fs -get**
   - **Description:** Copies files or directories from HDFS to the local file system.
   - **Example:**

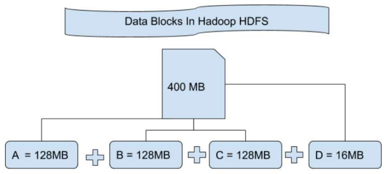
```bash
hadoop fs -get /user/hadoop/hdfsfile.txt localfile.txt
```

5. **hadoop fs -rm**
   - **Description:** Deletes files or directories from HDFS.
   - **Example:**

```bash
hadoop fs -rm /user/hadoop/hdfsfile.txt
```

## File Block in HDFS

- Data in HDFS always stores in the form of blocks . Single block of data is divided into multiple blocks of size 128MB which is default and you can change it manually.



- Suppose we have uploaded one file of size 400MB to your HDFS then who file 4 blocks mai divide hoti hai 128 ke 3 and ek 16 MB ka . HDFS doesn't care about what data is stored in these block and final block ko wo partial consider karta hai.

## Replication in HDFS

- Replication means copy of something.
- The number of times you make copy of something it can be expressed as its replication factor.
- By default replication factor for Hadoop is set to 3 you can change it manually.
- In above example we have made 4 blocks which means 3 replica or copy of each block is made means total 4* 3=12 blocks are made for backup purpose

## Rack Awareness

- The rack is just the physical collection of nodes in our Hadoop Cluster.
- A large Hadoop cluster consist of so many Racks.
- With the help of Rack the NameNode choose the closet data node to achieve maximum performance and perform read write operations.

## **Features of HDFS**

1) Highly Scalable:   HDFS is highly Scalable as it can scale hundreds of nodes in a single cluster. As the vloume of data increase it increases its size.

2) Replication: Due to some unfavorable conditions , the node containing the data may be loss. So, to overcome such problem , HDFS alwas maintain the copy of  data on different machine.

3) Fault Tolerance :  HDFS provides fault tolerance . In any machine fails , the other machine containing the copy of that data automatically became active.

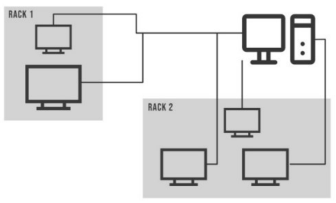
4) Distributed data storage : It is one of the most important feature of HDFS that make Hadoop powerful. Here data is divided into multiple blocks and stored into nodes.

5) Portable: HDFS is designed in such a way that it can be portable from one platform to other.

6) Security: HDFS offeres high security features likes access control lists and file permissions . It can integrate with Kerberos for authentication and secure communication.

## Rack Awareness

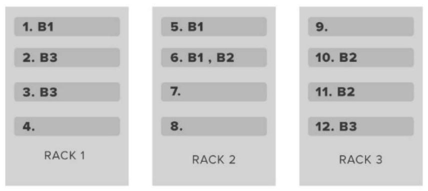
- The rack is a physical collection of nodes in our Hadoop.
- A large Hadoop cluster consists of large number of nodes.
- With the help of rack information, NameNode chooses the closet DataNode to increase the performance and to perform read write operations which reduces the network traffic.
- A rack can have multiple nodes that stores file blocks and their replica.
- Rack particular file block ko 2 different nodes mai write karta hai. And agar hame ek file ko 2 or jyada nodes mai store karna hai that also we can do.



- ✧ In large Hadoop cluster multiple racks are present , in each rack there are lots of data nodes are available.
- ✧ Communication between data nodes is faster which are present in same rack rather than which are present in different rack.
- ✧ The NameNode holds the id of every Rack which is present to find the closed DataNode present.
- ✧ The concept of choosing closest data node for serving purpose is Rack Awareness

Example

==> **Block 1 , Block 2 , Block 3**



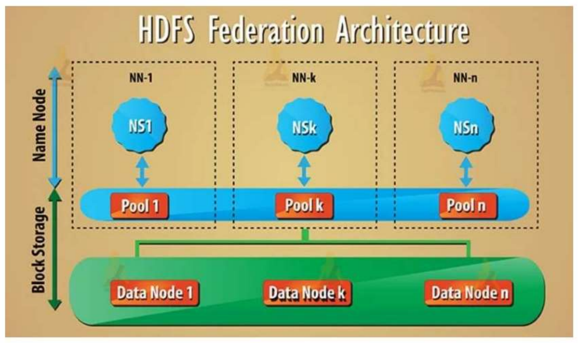
- ✧ Yaha hamare pass 3 different rack hai and har rack mai 4 datanodes hai.
- ✧ Now suppose hamare pass 3 blocks hai B1, B2 , B3 and we want to put in datanodes.
- ✧ As HDFS replica banata hai har file block ka to increase availability and fault tolerance. By default hum 3 replicas bana sakte hai , so hadoop un replicas ko rack mai aise place karta hai ki we get good network band width.
- ✧ For that it follow some network awareness policies as follows
  - There should not be more than one replica in same data node.
  - More than 2 replicas of single block is not allowed on the same rack.
  - The number of racks used inside Hadoop must smaller than the number of replicas.

**Benefits of Rack awareness:**

- ✧ We can store data in different racks so no way to lose our data.
- ✧ Helps to maximize network bandwidth.
- ✧ Improves cluster performance and high data availability.

## HDFS Federation:

- ❖ Federation enhances and existing Hadoop HDFS architecture.
- ❖ Pahle HDFS architecture allows single namespace for entire cluster. Jaha single NameNode manage karta tha name space ko.
- ❖ Agar namespace fail hota toh pura cluster out of service jata tha, jata Tak NameNode restart nahi hota who available nahi rehta.
- ❖ To overcome this limitation HDFS federation is introduced.
- ❖ Federation uses many independent NameNode to scale the name service horizontally. In federation architecture at the bottom data nodes are present . And ye data nodes common storage hai for all namenodes.
- ❖ Each data node is registered with all the name nodes present in the cluster.



**Benefits of HDFS Federation:**

1. Isolation : Each namespace is independent , with its own file system hierarchy.
2. Improved Scalability: It manages multiple namespaces , by distributing the namespace load across the multiple name nodes. Which make easier to handle large files and dictionaries.
3. Increases throughput: with multiple namespace and namenodes HDFS can handle the large volume of data which helps to get the value data and analytics.

## MAP REDUCE

- ▸ A MapReduce is a data Processing tool which is used to process data parallelly in a distributed form.
- ▸ It has two phase the mapper phase and the reducer phase.
- ▸ In Mapper input is given in the key value pair. Then jo bhi output milta hai is given as input to the Reducer.
- ▸ Reducer runs only after the Mapper is over.

- Reducer bhi key value format mai input leta hai and uska output final output hota hai.
  [Map Reduce Wala Diagram]

**Steps in map Reduce:**

- Map takes data in key value pair and returns a list of pairs. Isme keys unique nahi hongi.
- Phir jo bhi Map ka output hai uspe Hadoop sort and shuffle apply karta hai and unique keys send karta hai and list create karta hai of values which are associated with unique keys.
- Then jo sort and shuffle ka output hai who send kiya jata hai reducer ko . Then reducer uspar operations perform karta hai and final output milta hai.
  Input -----> Map --------> Shuffle --------> Reduce Input ----> Reduce Output------->Output.

**Map Task**

- Record Reader : - The purpose of record reader is to break the record. Ye key value pair provide karta hai Map() function ko. - Key its locational information and value is data associated with it.
- Map : - Map is one user defined function whose work to process the data obtained from record reader. - Map() function does not generate any key value pair.
- Combiner: - Combiner is used used for grouping the data in the map workflow. Ye similar hai local reducer ke . The intermediate key value which are generated with map() is combined with the help of combiner. - - Its use is optional.
- Partitioner: - Ye responsible hota hai to fetch key-value pairs generated in the mapper phase. - Hash code of each key is also fetched by the partitioner.

**Reduce Task**:

- Shuffle and sort: - Sort shuffle occurs on the Output of mapper and before the reducer. - When the Mapper task is complete, the results are sorted by key, partitioned if there are multiple reducers. - Using the input from each Mapper , we collect all the values for each unique key k2. - This output from the shuffle phase in the form of is sent as input to reducer phase.
- Reduce : The main function of Reduce is to gather the tuple generated from Map and then perform some sorting and aggregation function.
- Output Format: - A once all the operations are performed the key value pairs written into the file with the help of record writer.

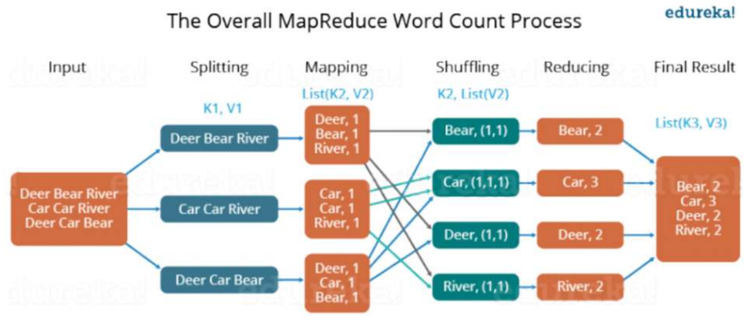
**Partitioner and Combiner**

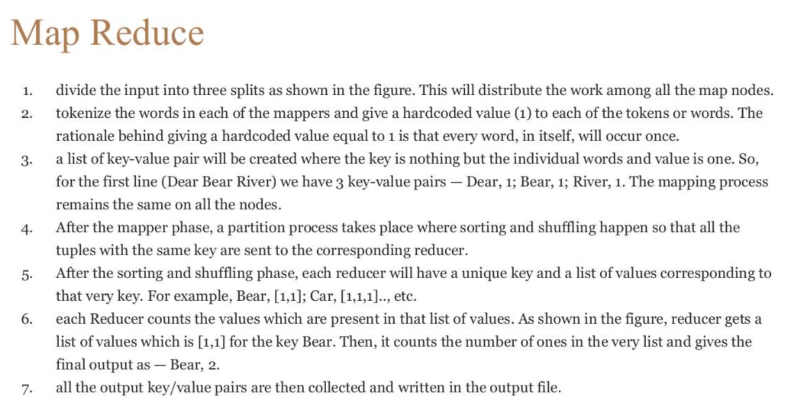
Combiner:  Ise semi reducer bhi bolte hai , ye map class se input accept karta and pass karta hai reducer phase ko. - Iska main function hota hai to summarize the map output records with the same key. Combiner class is used in between the map class and the reduce class to reduce the volume of the data transfer between map and reduce.

How Combiner works - Combiner does not have a predefined interface and it must implement the reducer reduce method(). - A combiner operates on each map output key.it must have same output key-value ass the Reducer class. A combiner can produce summary information from large dataset because i9it replaces the original map output.

Partitioner - A partitioner partitions the key-value pairs of intermediate Map-outputs. - It partitions the data using a user-defined condition, which works like a hash function. - The total number of partitions is same as

Word Count:





## Map Reduce

1. divide the input into three splits as shown in the figure. This will distribute the work among all the map nodes.
2. tokenize the words in each of the mappers and give a hardcoded value (1) to each of the tokens or words. The rationale behind giving a hardcoded value equal to 1 is that every word, in itself, will occur once.
3. a list of key-value pair will be created where the key is nothing but the individual words and value is one. So, for the first line (Dear Bear River) we have 3 key-value pairs — Dear, 1; Bear, 1; River, 1. The mapping process remains the same on all the nodes.
4. After the mapper phase, a partition process takes place where sorting and shuffling happen so that all the tuples with the same key are sent to the corresponding reducer.
5. After the sorting and shuffling phase, each reducer will have a unique key and a list of values corresponding to that very key. For example, Bear, [1,1]; Car, [1,1,1].., etc.
6. each Reducer counts the values which are present in that list of values. As shown in the figure, reducer gets a list of values which is [1,1] for the key Bear. Then, it counts the number of ones in the very list and gives the final output as — Bear, 2.
7. all the output key/value pairs are then collected and written in the output file.

## Unit 3 -------------------------------------------------------------------------------------------------------------------------------------------------

### NoSQL Business Drivers:

1. **Volume - Dealing with Big Data:**
   - Explanation: Companies have so much data that the traditional way of handling it, using one powerful computer, is becoming too slow.
   - Example: Imagine having so many customers and transactions that a regular computer can't keep up. NoSQL helps by using many smaller computers together to handle this massive amount of data.

2. **Velocity - Handling Data in Real-Time:**
   - Explanation: Some businesses need to deal with data really quickly, like when lots of people are using a website at the same time. Traditional systems struggle to keep up with this speed.
   - Example: Think of a website during a sale - many people are buying things at once. NoSQL helps make sure the website stays fast even when lots of people are using it at the same time.

3. **Variability - Dealing with Different Types of Data:**
   - Explanation: Businesses often have different types of information, and fitting all of it into neat tables can be hard. NoSQL is more flexible, allowing for different types of data to be stored more easily.
   - Example: In a traditional system, if you want to add a new type of information about customers, you might have to stop everything. With NoSQL, you can add new info without causing a big interruption.

4. **Agility - Adapting Quickly to Changes:**
   - Explanation: Changing or adding things to a traditional system can be slow and complicated. NoSQL makes it easier to adapt and make changes quickly.
   - Example: Imagine you want to add a new feature to an app. With NoSQL, it's like adding a new room to your house without having to rebuild the entire house. It's more flexible and adaptable.

### NoSQL Architecture Pattern

**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.

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

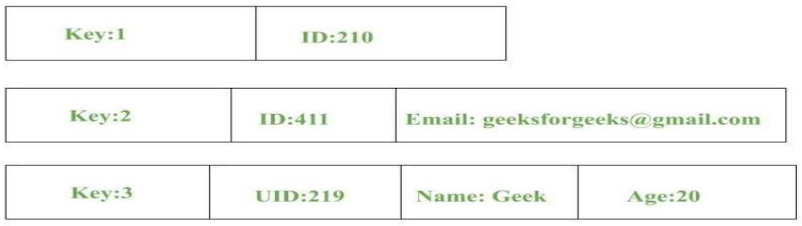
1. Key-Value Store Database
2. Column Store Database
3. Document Database
4. Graph Database

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.

Examples:
- DynamoDB
- Berkeley DB



**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.

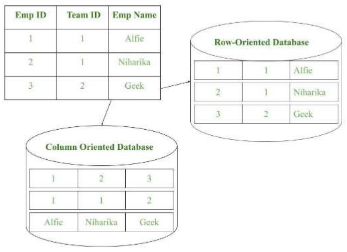
Basically, columns are mode of storage in this type.

Advantages:

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

Examples:

- HBase
- Bigtable by Google
- Cassandra



**Document Database:**

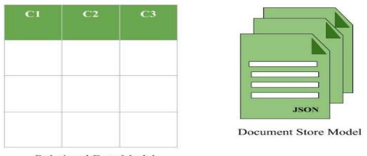
The document database fetches and accumulates data in form of key-value pairs but here, the values are called as Documents. Document can be stated as a complex data structure. Document here can be a form of text, arrays, strings, JSON, XML or any such format. The use of nested documents is also very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

Examples:

- MongoDB
- CouchDB



**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.

Examples:

- Neo4J
- FlockDB( Used by Twitter)

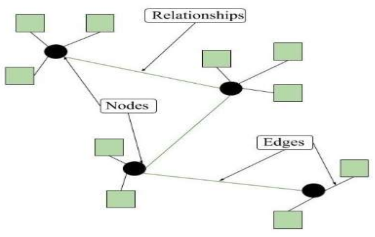
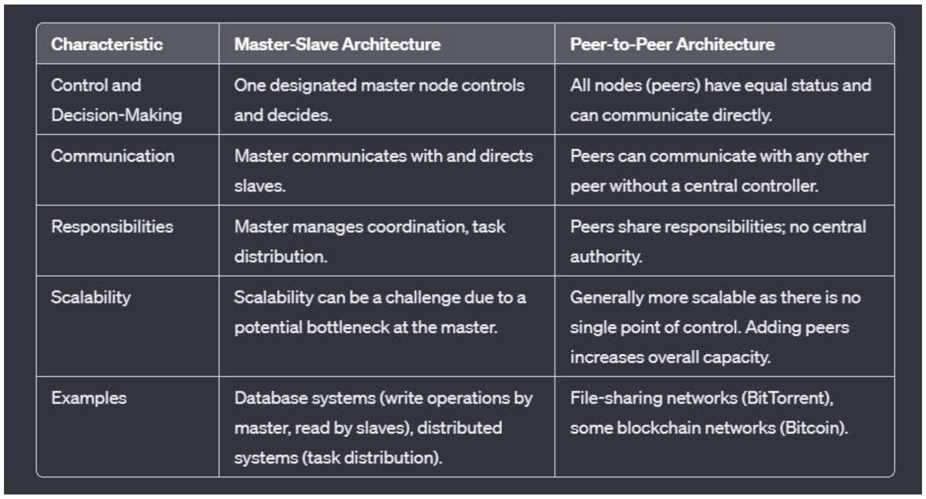


**Figure** – Graph model format of NoSQL Databases

**Master-Slave Vs Peer-to-Peer**



| Characteristic | Master-Slave Architecture | Peer-to-Peer Architecture |
|---|---|---|
| Control and Decision-Making | One designated master node controls and decides. | All nodes (peers) have equal status and can communicate directly. |
| Communication | Master communicates with and directs slaves. | Peers can communicate with any other peer without a central controller. |
| Responsibilities | Master manages coordination, task distribution. | Peers share responsibilities; no central authority. |
| Scalability | Scalability can be a challenge due to a potential bottleneck at the master. | Generally more scalable as there is no single point of control. Adding peers increases overall capacity. |
| Examples | Database systems (write operations by master, read by slaves), distributed systems (task distribution). | File-sharing networks (BitTorrent), some blockchain networks (Bitcoin). |

## HBase

HBase is like a giant, organized storage system for big amounts of data. It's part of the Hadoop family, designed to handle massive amounts of information in a way that's fast and reliable.

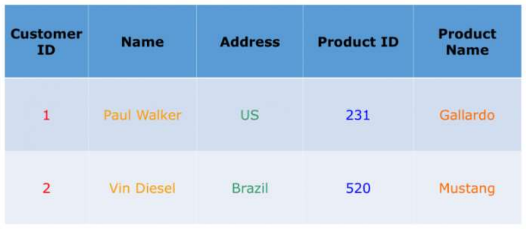
How Does HBase Work?

1. Think of Rows and Columns:
   - HBase organizes data like a giant table with rows and columns, just like a spreadsheet. Each row has a unique ID.
2. Scattered Storage:
   - Instead of storing everything in one place, HBase scatters the data across many computers. This makes it faster to find and use the information, especially when there's a lot of it.
3. Quick Access:
   - HBase is super quick at finding specific pieces of data, making it great for applications that need to grab information in a flash.

**HBase Architecture: HBase Data Model**
As we know, HBase is a column-oriented NoSQL database. Although it looks similar to a relational database which contains rows and columns, but it is not a relational database. Relational databases are row oriented while HBase is column-oriented. So, let us first understand the difference between Column-oriented and Row-oriented databases:
*Row-oriented vs column-oriented Databases:*

- Row-oriented databases store table records in a sequence of rows. Whereas column-oriented databases store table records in a sequence of columns, i.e. the entries in a column are stored in contiguous locations on disks.

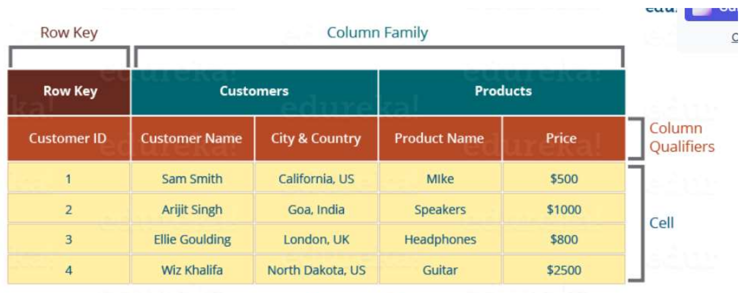
To better understand it, let us take an example and consider the table below.



| Customer ID | Name | Address | Product ID | Product Name |
|---|---|---|---|---|
| 1 | Paul Walker | US | 231 | Gallardo |
| 2 | Vin Diesel | Brazil | 520 | Mustang |

If this table is stored in a row-oriented database. It will store the records as shown below:
**1, Paul Walker, US, 231, Gallardo,**
**2, Vin Diesel, Brazil, 520, Mustang**
In row-oriented databases data is stored on the basis of rows or tuples as you  can see above.
While the column-oriented databases store this data as:
**1,2, Paul Walker, Vin Diesel, US, Brazil, 231, 520, Gallardo, Mustang**
In a column-oriented databases, all the column values are stored together like first column values will be stored together, then the second column values will be stored together and data in other columns are stored in a similar manner.

- When the amount of data is very huge, like in terms of petabytes or exabytes, we use column-oriented approach, because the data of a single column is stored together and can be accessed faster.
- While row-oriented approach comparatively handles less number of rows and columns efficiently, as row-oriented database stores data is a structured format.
- When we need to process and analyze a large set of semi-structured or unstructured data, we use column oriented approach. Such as applications dealing with **Online Analytical Processing** like data mining, data warehousing, applications including analytics, etc.
- Whereas, **Online Transactional Processing** such as banking and finance domains which handle structured data and require transactional properties (ACID properties) use row-oriented approach.


Row Key
Column Family
Column Qualifiers
Cell

| Row Key | Customers | | Products | |
|---|---|---|---|---|
| Customer ID | Customer Name | City & Country | Product Name | Price |
| 1 | Sam Smith | California, US | Mike | $500 |
| 2 | Arijit Singh | Goa, India | Speakers | $1000 |
| 3 | Ellie Goulding | London, UK | Headphones | $800 |
| 4 | Wiz Khalifa | North Dakota, US | Guitar | $2500 |

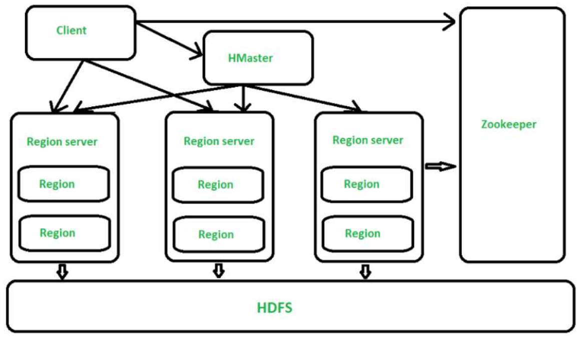
*Figure: HBase Table*

- **Tables**: Data is stored in a table format in HBase. But here tables are in column-oriented format.
- **Row Key**: Row keys are used to search records which make searches fast. You would be curious to know how? I will explain it in the architecture part moving ahead in this blog.
- **Column Families**: Various columns are combined in a column family. These column families are stored together which makes the searching process faster because data belonging to same column family can be accessed together in a single seek.
- **Column Qualifiers**: Each column's name is known as its column qualifier.
- **Cell**: Data is stored in cells. The data is dumped into cells which are specifically identified by rowkey and column qualifiers.
- **Timestamp**: Timestamp is a combination of date and time. Whenever data is stored, it is stored with its timestamp. This makes easy to search for a particular version of data.

In a more simple and understanding way, we can say HBase consists of:

- Set of tables
- Each table with column families and rows
- Row key acts as a Primary key in HBase.
- Any access to HBase tables uses this Primary Key
- Each column qualifier present in HBase denotes attribute corresponding to the object which resides in the cell.

HBase Architecture:



1. **HMaster: The Organizer**

- Role: HMaster is like the boss or organizer in HBase.
- Job: It decides where different parts of data (called regions) should go and handles important tasks like creating or deleting tables.
- Tasks: It keeps an eye on all the worker servers (Region Servers) to make sure everything is running smoothly.
- Superpowers: HMaster can do things like balancing the workload and taking charge if one worker server stops working.

2. **Region Server: The Worker**

- Role: Region Server is like the hardworking employee in HBase.
- Job: It takes care of specific parts of the data (regions) and manages tasks like reading and writing information.
- Place of Work: Region Server works on the computers where the actual data is stored (HDFS DataNode in Hadoop).
- Superpowers: It's responsible for making sure your data operations (like reading and writing) happen smoothly.

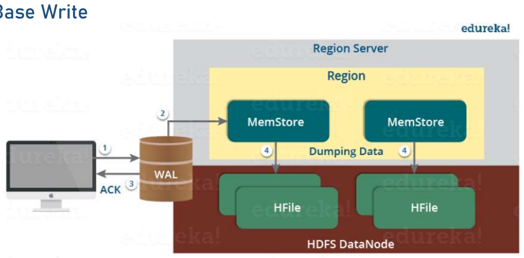
3. **Zookeeper: The Coordinator**

- Role: Zookeeper is like the coordinator that helps everyone work together in HBase.
- Job: It does things like keeping track of where data is stored, making sure everyone knows the current plan, and telling servers about any issues.
- Services: Zookeeper provides services that help servers communicate and stay organized.
- Superpowers: It helps in managing the behind-the-scenes details so that everything in HBase runs smoothly.

Putting it Together:

- Imagine HMaster as the boss deciding what needs to be done.
- Think of Region Server as the workers doing the actual tasks on specific pieces of data.
- Zookeeper is like the coordinator making sure everyone is on the same page and helping them work together effectively.

HBase Write



The write mechanism goes through the following process sequentially (refer to the above image):
*Step 1:* Whenever the client has a write request, the client writes the data to the WAL (Write Ahead Log).
- The edits are then appended at the end of the WAL file.
- This WAL file is maintained in every Region Server and Region Server uses it to recover data which is not committed to the disk.

*Step 2:* Once data is written to the WAL, then it is copied to the MemStore.
*Step 3:* Once the data is placed in MemStore, then the client receives the acknowledgment.
*Step 4:* When the MemStore reaches the threshold, it dumps or commits the data into a HFile.

The MemStore always updates the data stored in it, in a lexicographical order (sequentially in a dictionary manner) as sorted KeyValues. There is one MemStore for each column family, and thus the updates are stored in a sorted manner for each column family.

**Read Mechanism**
As discussed in our search mechanism, first the client retrieves the location of the Region Server from .META Server if the client does not have it in its cache memory. Then it goes through the sequential steps as follows:
- For reading the data, the scanner first looks for the Row cell in Block cache. Here all the recently read key value pairs are stored.
- If Scanner fails to find the required result, it moves to the MemStore, as we know this is the write cache memory. There, it searches for the most recently written files, which has not been dumped yet in HFile.
- At last, it will use bloom filters and block cache to load the data from HFile.

------------------------------------------------------------------------------------------------------------------------------------------------

UNIT 6

Data visualization is a powerful tool in the field of data analysis and communication, offering numerous benefits for both individuals and organizations. Here are some of the key importance and benefits of data visualization:
1. Enhances Understanding:
   - Importance: Humans process visual information faster than text or numbers. Data visualization helps to convert complex datasets into visual representations, making it easier for individuals to comprehend and interpret information.
2. Identifies Patterns and Trends:
   - Importance: Visualizations highlight patterns, trends, and insights within data that may not be apparent in raw numbers or textual formats. This can lead to a deeper understanding of the underlying data and facilitate more informed decision-making.
3. Aids in Decision-Making:
   - Importance: Visualizing data allows decision-makers to quickly grasp the key information they need. It supports more informed and timely decision-making by presenting data in a format that is easy to understand.
4. Facilitates Communication:
   - Importance: Visualizations are effective tools for conveying complex information to a diverse audience. Whether in business meetings or academic settings, visualizations make it easier to communicate findings and insights to stakeholders with varying levels of expertise.
5. Improves Data Quality:
   - Importance: Visual representations of data can reveal outliers, errors, or inconsistencies more easily than examining raw datasets. This helps in identifying and addressing data quality issues promptly.
6. Supports Storytelling:
   - Importance: Data visualizations can be used to tell a compelling story. By combining data points into a narrative, visualizations engage audiences and help them connect with the information on a more emotional level.
7. Enhances Exploration and Discovery:
   - Importance: Interactive data visualizations allow users to explore datasets and discover insights by interacting with the data. This fosters a sense of exploration and encourages curiosity about the underlying information.
8. Increases Efficiency:
   - Importance: Visualizations simplify the understanding of data, reducing the time required to comprehend complex datasets. This increased efficiency is particularly valuable in environments where quick decisions are crucial.
9. Encourages Data-Driven Culture:
   - Importance: Implementing data visualization tools fosters a data-driven culture within organizations. When individuals can easily access and understand data, they are more likely to rely on data in their decision-making processes.
10. Monitors Performance:
    - Importance: Visual dashboards and charts enable real-time monitoring of key performance indicators (KPIs) and metrics. This allows organizations to track progress, identify areas for improvement, and make data-driven adjustments.

**Types of Data Visualization Techniques:**
1. Bar Charts:
   - Description: Uses bars of varying lengths to represent data values. Suitable for comparing quantities across different categories.
   - Example: Comparing sales figures for different products.
2. Line Charts:
   - Description: Connects data points with lines to show trends over a continuous interval.
   - Example: Showing the change in temperature over a week.
3. Pie Charts:

- Description: Divides a circle into slices to represent parts of a whole.
- Example: Displaying the percentage distribution of expenses.

4. Scatter Plots:
    - Description: Represents individual data points on a two-dimensional graph, helpful for identifying relationships between variables.
    - Example: Analyzing the correlation between hours of study and exam scores.
5. Heatmaps:
    - Description: Uses color to represent values in a matrix, making it easy to identify patterns and trends.
    - Example: Displaying website traffic by time of day and day of the week.
6. Histograms:
    - Description: Visualizes the distribution of a dataset by grouping data into bins.
    - Example: Analyzing the distribution of ages in a population.
7. Treemaps:
    - Description: Hierarchically visualizes data using nested rectangles within rectangles.
    - Example: Displaying the breakdown of expenses in a budget hierarchy.
8. Bubble Charts:
    - Description: Similar to scatter plots but includes a third dimension, using the size of bubbles to represent a variable.
    - Example: Showing the correlation between GDP, population, and education spending for different countries.
9. Word Clouds:
    - Description: Represents words or phrases with varying sizes based on their frequency or importance.
    - Example: Analyzing the most common words in customer reviews.
10. Choropleth Maps:
    - Description: Uses colour variations to represent data values in different geographical regions.
    - Example: Visualizing population density across countries.

**Data Visualization Tools:**

1. Tableau:
    - Description: A popular and user-friendly tool for creating interactive and shareable visualizations.
2. Microsoft Power BI:
    - Description: Integrates with Microsoft products and allows users to create dynamic visualizations.
3. Google Data Studio:
    - Description: A free tool that enables users to create interactive reports and dashboards.
4. D3.js:
    - Description: A JavaScript library for creating dynamic, interactive data visualizations in web browsers.
5. Matplotlib:
    - Description: A Python library for creating static, animated, and interactive visualizations.
6. Plotly:
    - Description: A versatile tool that supports various programming languages and allows for interactive visualizations.
7. QlikView:
    - Description: Offers associative data modeling for exploring relationships in data.
8. Excel (Microsoft):
    - Description: Widely used spreadsheet software that includes basic data visualization tools.
9. R Studio:
    - Description: An integrated development environment (IDE) for the R programming language, commonly used for statistical analysis and data visualization.
10. Infogram:
    - Description: Focuses on creating infographics and interactive charts for storytelling.

Dashboards play a crucial role in data visualization by providing a consolidated and interactive view of key metrics, trends, and insights. Here are some important uses of dashboards in data visualization:

1. Centralized Information:
    - Use: Dashboards serve as a centralized hub where various data visualizations and metrics are aggregated in one place.
    - Benefit: Users can quickly access and comprehend a wide range of information without navigating through multiple reports or datasets.
2. Real-Time Monitoring:
    - Use: Dashboards often include real-time or near-real-time data updates.
    - Benefit: Decision-makers can monitor current performance, track trends, and respond promptly to changes, contributing to more agile and proactive decision-making.
3. Performance Measurement:
    - Use: Dashboards showcase key performance indicators (KPIs) and metrics relevant to an organization's goals.
    - Benefit: Users can easily evaluate how well they are performing against targets and objectives, enabling effective performance management.
4. Trend Analysis:
    - Use: Dashboards visualize trends and patterns over time.
    - Benefit: Users can identify historical patterns, seasonal variations, or emerging trends, facilitating data-driven insights for strategic planning.
5. Data Exploration:
    - Use: Dashboards often provide interactive elements for data exploration.
    - Benefit: Users can drill down into specific data points, filter information, and gain deeper insights into the underlying data, fostering a more thorough understanding.
6. Cross-Functional Collaboration:
    - Use: Dashboards are shared across teams or departments.
    - Benefit: Facilitates collaboration by ensuring that everyone has access to the same visualizations, fostering a shared understanding and alignment toward common goals.
7. Quick Decision-Making:
    - Use: Dashboards present information in a visually compelling and digestible format.
    - Benefit: Decision-makers can quickly grasp the current state of affairs, leading to faster and more informed decisions.
8. Alerts and Notifications:
    - Use: Dashboards may include alert mechanisms for exceptional situations.
    - Benefit: Users can be notified when specific thresholds or conditions are met, allowing for proactive intervention in critical situations.
9. Customization for User Roles:
    - Use: Dashboards can be customized based on user roles and responsibilities.
    - Benefit: Ensures that each user sees the most relevant information for their specific role, streamlining their decision-making process.

10. Enhanced Communication:
    - Use: Dashboards enable clear and concise communication of complex data.
    - Benefit: Supports effective storytelling, helping to convey data-driven insights to stakeholders and fostering a shared understanding of organizational performance.

---

UNIT 5

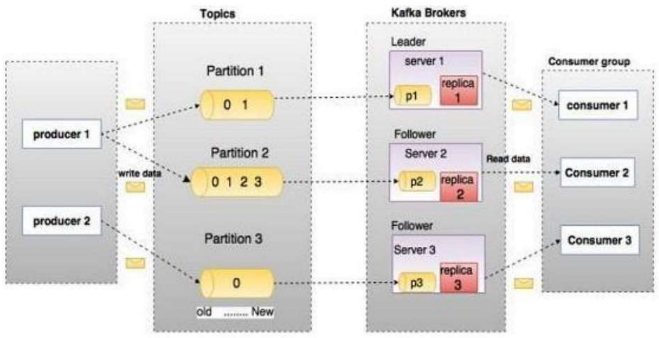
## Kafka

Apache Kafka is a distributed publish-subscribe messaging system and a robust queue that can handle a high volume of data and enables you to pass messages from one endpoint to another. Kafka is suitable for both offline and online message consumption.

Kafka is built on top of the Zookeeper synchronization service. Apache Kafka is an open-source distributed event streaming platform used for building real-time data pipelines and streaming applications.

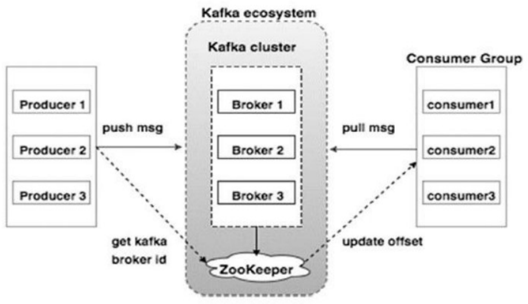
**Benefits:**
Reliability: Kafka is distributed, partitioned, replicated and fault tolerance.
Scalability: Kafka messaging system scales easily without down time.
Durability: Kafka uses Distributed commit log which means messages persists on disk as fast as possible, hence it is durable.
Performance: Kafka has high throughput for both publishing and subscribing messages.
It maintains stable performance even many TB of messages are stored. Kafka is very fast and guarantees zero downtime and zero data loss.

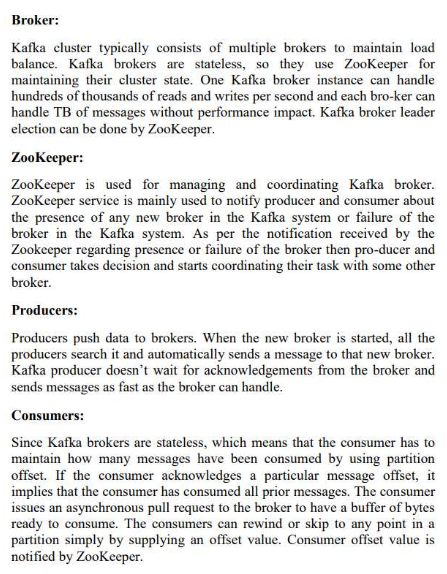
Fundamentals:



Topics: A stream of messages belonging to a particular category is called a topic. Data is stored in topics.
Partition: Topics may have many partitions
Partition offset: Each partitioned message has a unique sequence id called as offset.
Replica of partition: Replicas are nothing but backups of a partition. Replicas are never read or write data. They are used to prevent data loss
Brokers: Brokers are simple system responsible for maintaining the published data.
Kafka Clusters: Kafka's having more than one broker are called as Kafka cluster.
Producers: Producers are the publisher of messages to one or more Kafka topics. Producers send data to Kafka brokers. Every time a producer pub-lishes a message to a broker, the broker simply appends the message to the last segment file.
Consumers: Consumers read data from brokers. Consumers subscribes to one or more topics and consume published messages by pulling data from the brokers.
Leader: Leader is the node responsible for all reads and writes for the given partition. Every partition has one server acting as a leader.
Follower: Node which follows leader instructions are called as follower.

Architecture:




**Broker:**

Kafka cluster typically consists of multiple brokers to maintain load balance. Kafka brokers are stateless, so they use ZooKeeper for maintaining their cluster state. One Kafka broker instance can handle hundreds of thousands of reads and writes per second and each bro-ker can handle TB of messages without performance impact. Kafka broker leader election can be done by ZooKeeper.

**ZooKeeper:**

ZooKeeper is used for managing and coordinating Kafka broker. ZooKeeper service is mainly used to notify producer and consumer about the presence of any new broker in the Kafka system or failure of the broker in the Kafka system. As per the notification received by the Zookeeper regarding presence or failure of the broker then pro-ducer and consumer takes decision and starts coordinating their task with some other broker.

**Producers:**

Producers push data to brokers. When the new broker is started, all the producers search it and automatically sends a message to that new broker. Kafka producer doesn't wait for acknowledgements from the broker and sends messages as fast as the broker can handle.

**Consumers:**

Since Kafka brokers are stateless, which means that the consumer has to maintain how many messages have been consumed by using partition offset. If the consumer acknowledges a particular message offset, it implies that the consumer has consumed all prior messages. The consumer issues an asynchronous pull request to the broker to have a buffer of bytes ready to consume. The consumers can rewind or skip to any point in a partition simply by supplying an offset value. Consumer offset value is notified by ZooKeeper.

## Apache Spark

Apache Spark is an open-source, distributed computing system that's designed to process large amounts of data quickly and efficiently. Imagine you have a massive dataset, and you want to perform complex operations or analysis on it. Spark is like a super-smart assistant that helps you do that in a fast and organized way.
Here are some key features of Apache Spark explained in easy words:

1. Speed:
   - Spark is really fast! It can process data much quicker than traditional methods. This is because it stores data in memory (RAM) and performs multiple operations in one go, reducing the need to read and write to disk.
2. Ease of Use:
   - Spark is designed to be user-friendly. You can write programs in languages like Scala, Java, Python, or even use SQL-like queries. This makes it easier for developers to work with.
3. Versatility:
   - Spark is like a Swiss Army knife for data processing. It can handle various tasks like batch processing (working with large sets of data), real-time stream processing (dealing with continuous data streams), machine learning, and graph processing.
4. In-Memory Processing:
   - Instead of reading data from disk every time it's needed, Spark keeps frequently used data in memory. This makes repeated operations much faster.
5. Fault Tolerance:
   - Spark is robust. If a part of your computation fails, it can recover from that failure and continue processing the data from where it left off.
6. Scalability:
   - Spark can scale up to handle really large datasets. If you have more data or more computing power, Spark can use them efficiently to get the job done.

RDD
Resilient Distributed Datasets (RDD) is a fundamental data structure of Spark. It is an immutable distributed collection of objects.

Resilient:
- ❖ RDDs are "resilient" because they can recover from failures. If a part of your data or computation is lost due to a node failure, RDDs have the information needed to rebuild that lost data.
- ❖ Example: Imagine you have a big jigsaw puzzle. Your puzzle is "resilient" because even if a few pieces go missing, you can still complete the puzzle.

Distributed:
- ❖ RDDs are "distributed" because they are spread across multiple nodes (computers) in a cluster. This distribution allows them to handle large-scale data processing tasks by dividing the data into partitions that can be processed in parallel on different machines.
- ❖ Example: Now, imagine you have friends helping you solve the puzzle. Each friend has some pieces, and you work together to complete it.

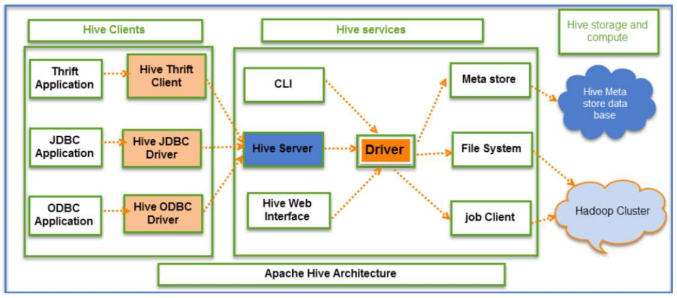
Datasets:
- ❖ RDDs represent a collection of objects distributed across a cluster. These objects can be anything - from simple data types to complex objects. RDDs allow you to perform parallel processing on these distributed datasets.
- ❖ Example: The puzzle pieces are your "dataset." RDDs deal with lots of data, just like your puzzle has many pieces. But RDDs organize and process this data in a way that makes it easy to work with, even if it's spread across different computers.

--------------------------------------------------------------------------------------------------------------------------------------------

Hive
Apache Hive is a data warehousing and SQL-like query language system built on top of Hadoop. It provides a way to query and analyze large datasets stored in Hadoop's distributed file system (HDFS) and other compatible distributed storage systems.

Characterstics of Hive

Hive Architecture



**HIVE Clients:**
Hive offers a variety of drivers for interacting with various types of applications. It will provide a Thrift client for communication in Thrift based applications. JDBC Drivers are available for Java-based applications. ODBC drivers are available for any sort of application. In the Hive services, these Clients and Drivers communicate with the Hive server.

**HIVE Services:**
Hive Services can be used by clients to interact with Hive. If a client wishes to do any query-related actions in Hive, it must use Hive Services to do so. The command line interface (CLI) is used by Hive to perform DDL (Data Definition Language) operations. As indicated in the architectural diagram above, all drivers communicate with Hive server and the main driver in Hive services. The main driver is present in Hive services, and it interfaces with all types of JDBC, ODBC, and other client specific applications. Driver will process requests from various apps and send them to the meta store and field systems for processing.

**Storage and Computing:**
Hive services like Meta store, File system, and Job Client communicate with Hive storage and carry out the following tasks.
- The Hive "Meta storage database" stores the metadata of tables generated in Hive.
- Query results and data loaded into tables will be stored on HDFS in a Hadoop cluster.

**Job execution flow**
Step-1: Execute Query –
Interface of the Hive such as Command Line or Web user interface delivers query to the driver to execute. In this, UI calls the execute interface to the driver such as ODBC or JDBC.

Step-2: Get Plan –
Driver designs a session handle for the query and transfer the query to the compiler to make execution plan. In other words, driver interacts with the compiler.

Step-3: Get Metadata –
In this, the compiler transfers the metadata request to any database and the compiler gets the necessary metadata from the metastore.

Step-4: Send Metadata –
Metastore transfers metadata as an acknowledgment to the compiler.

Step-5: Send Plan –
Compiler communicating with driver with the execution plan made by the compiler to execute the query.

Step-6: Execute Plan –
Execute plan is sent to the execution engine by the driver. Execute Job Job Done Dfs operation (Metadata Operation)

Step-7: Fetch Results –
Fetching results from the driver to the user interface (UI).

Step-8: Send Results –
Result is transferred to the execution engine from the driver. Sending results to Execution engine. When the result is retrieved from data nodes to the execution engine, it returns the result to the driver and to user interface (UI).

**Advantages of Hive Architecture:**

**Scalability:** Hive is a distributed system that can easily scale to handle large volumes of data by adding more nodes to the cluster.
**Data Accessibility:** Hive allows users to access data stored in Hadoop without the need for complex programming skills. SQL-like language is used for queries and HiveQL is based on SQL syntax.
**Data Integration:** Hive integrates easily with other tools and systems in the Hadoop ecosystem such as Pig, HBase, and MapReduce.
**Flexibility:** Hive can handle both structured and unstructured data, and supports various data formats including CSV, JSON, and Parquet.
**Security:** Hive provides security features such as authentication, authorization, and encryption to ensure data privacy.

## Hive Built In Function

**Round() function:**

```
hive> SELECT round(2.6) from temp;
```

On successful execution of query, you get to see the following response:

```
3.0
```

**Floor() function:**

```
hive> SELECT floor(2.6) from temp;
```

On successful execution of the query, you get to see the following response:

```
2.0
```

**Ceil() function:**

```
hive> SELECT ceil(2.6) from temp;
```

On successful execution of the query, you get to see the following response:

```
3.0
```

| | | |
|---|---|---|
| BIGINT | count(*), count(expr), | count(*) - Returns the total number of retrieved rows. |
| DOUBLE | sum(col), sum(DISTINCT col) | It returns the sum of the elements in the group or the sum of the distinct values of the column in the group. |
| DOUBLE | avg(col), avg(DISTINCT col) | It returns the average of the elements in the group or the average of the distinct values of the column in the group. |
| DOUBLE | min(col) | It returns the minimum value of the column in the group. |
| DOUBLE | max(col) | It returns the maximum value of the column in the group. |