1. Illustrate Hadoop ecosystem with its Architecture in detail.

The Hadoop ecosystem is a collection of open-source software tools and frameworks that work together to process, store, and analyze big data. The core components of the Hadoop ecosystem include the Hadoop Distributed File System (HDFS) for storage and the MapReduce programming model for distributed processing. Here's an overview of the key components and architecture of the Hadoop ecosystem:

Core Components:
Hadoop Distributed File System (HDFS):

Description: A distributed file system that provides scalable and reliable storage for large amounts of data across multiple nodes. HDFS breaks large files into smaller blocks (default size is 128 MB or 256 MB) and distributes them across the cluster.
MapReduce:

Description: A programming model and processing engine for distributed computing. MapReduce divides a processing task into two phases—Map and Reduce—and executes them in parallel across the nodes of a Hadoop cluster.
Additional Components:
Hadoop YARN (Yet Another Resource Negotiator):

Description: A resource management layer that enables multiple data processing engines to share resources in a Hadoop cluster. YARN separates the resource management functionality from the processing logic, allowing different applications to run simultaneously.
Apache Hive:

Description: A data warehouse infrastructure built on top of Hadoop that provides a SQL-like query language called HiveQL. Hive enables users to query and analyze data stored in HDFS using familiar SQL syntax.
Apache HBase:

Description: A distributed, scalable, and NoSQL database that is designed to handle large volumes of sparse data. HBase is suitable for random, real-time read/write access to big data.
Apache Pig:

Description: A high-level scripting language for expressing data analysis programs. Pig scripts are translated into a series of MapReduce jobs that can be executed on a Hadoop cluster.
Apache Spark:

Description: A fast and general-purpose cluster computing system that supports in-memory data processing. Spark can be used for various tasks, including batch processing, interactive queries, and machine learning.
Apache Sqoop:

Description: A tool for efficiently transferring bulk data between Apache Hadoop and structured data stores such as relational databases. Sqoop simplifies the process of importing and exporting data.

Apache Flume:

Description: A distributed, reliable, and available service for efficiently collecting, aggregating, and moving large amounts of log data or event data to HDFS.
Apache Oozie:

Description: A workflow scheduler system used to manage Apache Hadoop jobs. Oozie allows the coordination of various Hadoop jobs to create complex data processing workflows.
Apache ZooKeeper:

Description: A distributed coordination service that provides distributed synchronization and configuration management. ZooKeeper helps in maintaining configuration information, naming, and providing distributed locks.
Hadoop Ecosystem Architecture:
The architecture of the Hadoop ecosystem is based on the master-slave model, where a master node manages the overall coordination, and multiple slave nodes handle the distributed processing of data.

NameNode (Master): Manages the metadata of the file system, including information about the structure of the HDFS (e.g., file names, permissions, and block locations).

DataNodes (Slaves): Store the actual data in the form of blocks and are responsible for reading and writing data as instructed by the NameNode.

ResourceManager (Master): Part of YARN, manages the allocation of resources in the cluster and schedules applications.

NodeManagers (Slaves): Part of YARN, execute tasks on individual nodes and monitor their resource usage.

JobTracker (Deprecated): In older versions, JobTracker was responsible for managing and scheduling MapReduce jobs. In newer versions, this role is handled by ResourceManager and per-job ApplicationMaster.

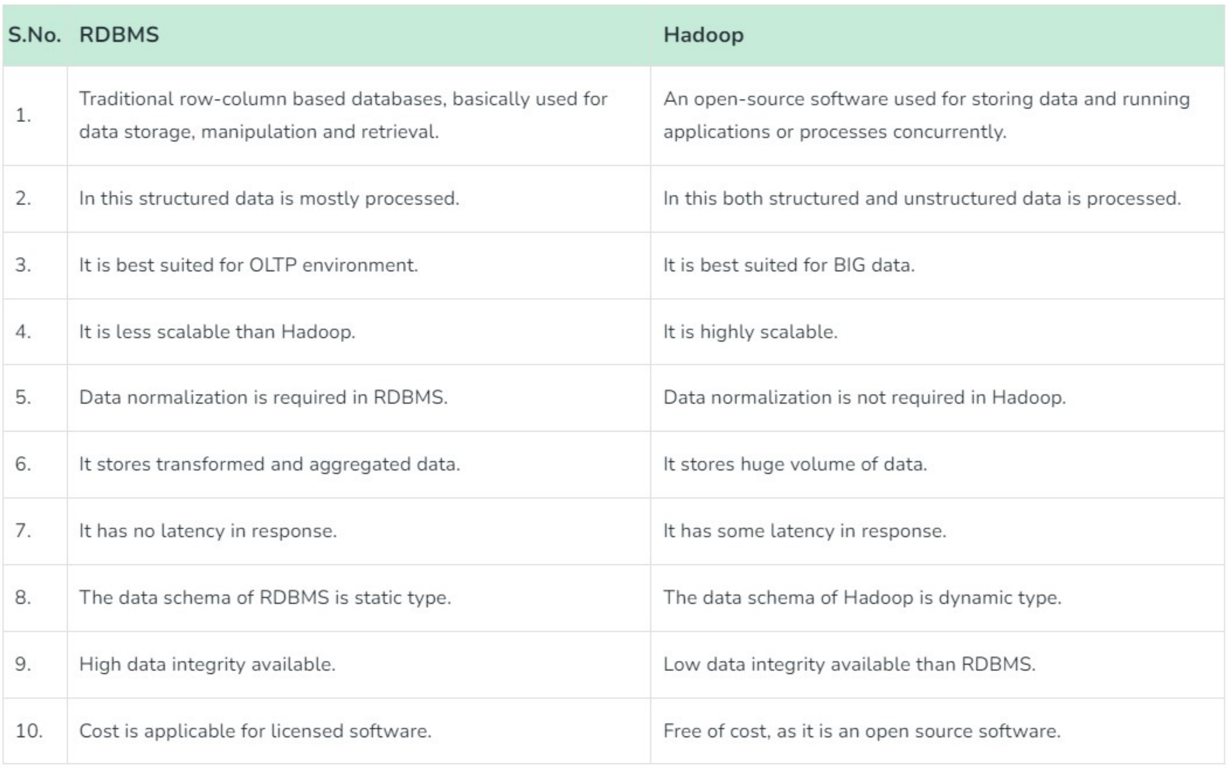
TaskTrackers (Deprecated): In older versions, TaskTrackers were responsible for executing MapReduce tasks. In newer versions, this role is handled by NodeManagers.

This architecture enables the distributed storage and processing of data across the Hadoop cluster, providing scalability and fault tolerance.

Refer Text Book Page No :68

2. Compare and contrast RDBMS and Hadoop/HDFS.

RDMS (Relational Database Management System): RDBMS is an information management system, which is based on a data model.In RDBMS tables are used for information storage. Each row of the table represents a record and column represents an attribute of data. Organization of data and their manipulation processes are different in RDBMS from other databases. RDBMS ensures ACID (atomicity, consistency, integrity, durability) properties required for designing a database. The purpose of RDBMS is to store, manage, and retrieve data as quickly and reliably as possible.
Hadoop: It is an open-source software framework used for storing data and running applications on a group of commodity hardware. It has large storage capacity and high processing power. It can manage multiple concurrent processes at the same time. It is used in predictive analysis, data mining and machine learning. It can handle both structured and unstructured form of data. It is more flexible in storing, processing, and managing data than traditional RDBMS. Unlike traditional systems, Hadoop enables multiple analytical processes on the same data at the same time. It supports scalability very flexibly.



S.No.	RDBMS	Hadoop
1.	Traditional row-column based databases, basically used for data storage, manipulation and retrieval.	An open-source software used for storing data and running applications or processes concurrently.
2.	In this structured data is mostly processed.	In this both structured and unstructured data is processed.
3.	It is best suited for OLTP environment.	It is best suited for BIG data.
4.	It is less scalable than Hadoop.	It is highly scalable.
5.	Data normalization is required in RDBMS.	Data normalization is not required in Hadoop.
6.	It stores transformed and aggregated data.	It stores huge volume of data.
7.	It has no latency in response.	It has some latency in response.
8.	The data schema of RDBMS is static type.	The data schema of Hadoop is dynamic type.
9.	High data integrity available.	Low data integrity available than RDBMS.
10.	Cost is applicable for licensed software.	Free of cost, as it is an open source software.

Refer Text Book Page No :72

3. What are the various distributed system challenges and explain in brief how Hadoop tried to address them.

Distributed systems, including those used in big data processing frameworks like Hadoop, face various challenges due to their distributed and parallel nature. Here are some common challenges and how Hadoop attempts to address them:

Data Distribution and Replication:

Challenge: Ensuring efficient and balanced data distribution across nodes while avoiding hotspots.
Hadoop's Approach: Hadoop's HDFS addresses this challenge by splitting large files into smaller blocks and distributing them across the cluster. Additionally, HDFS employs data replication to ensure fault tolerance and availability.

Fault Tolerance:

Challenge: Nodes in a distributed system may fail, leading to data loss or processing interruptions.
Hadoop's Approach: Hadoop uses data replication to store multiple copies of each block across different nodes. If a node fails, the system can retrieve the data from one of the replicated copies, ensuring fault tolerance.

Data Consistency:

Challenge: Maintaining consistency across distributed data copies, especially in the presence of failures or concurrent updates.
Hadoop's Approach: Hadoop prioritizes fault tolerance over strict consistency. In the HDFS architecture, eventual consistency is maintained by asynchronously replicating data, allowing for system availability even during temporary inconsistencies.

Node Communication Overhead:

Challenge: High communication overhead between nodes in a distributed system can lead to performance bottlenecks.
Hadoop's Approach: Hadoop minimizes inter-node communication overhead by using a master-slave architecture and leveraging mechanisms like HDFS block replication and MapReduce task distribution to efficiently manage data and computation across the cluster.

Task Scheduling and Load Balancing:

Challenge: Efficiently distributing tasks among nodes and balancing the workload to avoid stragglers.
Hadoop's Approach: Hadoop's YARN (Yet Another Resource Negotiator) component addresses task scheduling and load balancing. YARN manages resources in the cluster, allocates them to applications, and schedules tasks, ensuring optimal resource utilization.

Scalability:

Challenge: Adding nodes to the system should result in proportional increases in performance.
Hadoop's Approach: Hadoop's architecture is designed for horizontal scalability. New nodes can be added to the cluster, and the system can scale out to handle larger datasets and processing loads.

Data Locality:

Challenge: Minimizing data movement across the network by executing tasks on nodes where the data resides.
Hadoop's Approach: Hadoop's MapReduce paradigm emphasizes data locality. It schedules tasks to run on nodes where the data is stored, reducing the need for extensive data transfer across the network and improving performance.

Security:

Challenge: Ensuring the security of data and resources in a distributed environment.

Hadoop's Approach: Hadoop provides security features such as authentication, authorization, and encryption. Components like HDFS and MapReduce have security mechanisms, and projects like Apache Knox and Apache Ranger enhance security in Hadoop ecosystems.

Complexity of Development and Debugging:
Challenge: Developing and debugging applications for a distributed environment can be complex.
Hadoop's Approach: Hadoop provides high-level abstractions like MapReduce and tools like Apache Pig and Apache Hive, which simplify the development process. Additionally, debugging tools and logs help identify and troubleshoot issues in distributed applications.

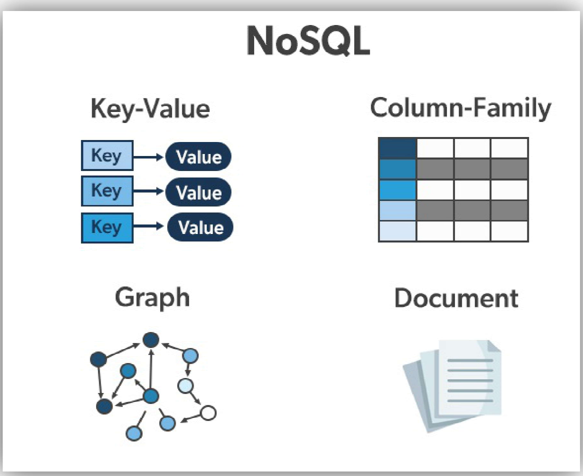
Refer Text Book Page No : 83-84

4. Identify various types of NoSQL Databases.

A database is a collection of structured data or information which is stored in a computer system and can be accessed easily. A database is usually managed by a Database Management System (DBMS).

NoSQL is a non-relational database that is used to store the data in the nontabular form. NoSQL stands for Not only SQL. The main types are documents, key-value, wide-column, and graphs.
Types of NoSQL Database:

- Document-based databases
- Key-value stores
- Column-oriented databases
- Graph-based databases



Document-Based Database:
The document-based database is a nonrelational database. Instead of storing the data in rows and columns (tables), it uses the documents to store the data in the database. A document database stores data in JSON, BSON, or XML documents.

Documents can be stored and retrieved in a form that is much closer to the data objects used in applications which means less translation is required to use these data in the applications. In the Document database, the particular elements can be accessed by using the index value that is assigned for faster querying.

Collections are the group of documents that store documents that have similar contents. Not all the documents are in any collection as they require a similar schema because document databases have a flexible schema.

Key features of documents database:

Flexible schema: Documents in the database has a flexible schema. It means the documents in the database need not be the same schema.
Faster creation and maintenance: the creation of documents is easy and minimal maintenance is required once we create the document.
No foreign keys: There is no dynamic relationship between two documents so documents can be independent of one another. So, there is no requirement for a foreign key in a document database.
Open formats: To build a document we use XML, JSON, and others.
Key-Value Stores:
A key-value store is a nonrelational database. The simplest form of a NoSQL database is a key-value store. Every data element in the database is stored in key-value pairs. The data can be retrieved by using a unique key allotted to each element in the database. The values can be simple data types like strings and numbers or complex objects.

A key-value store is like a relational database with only two columns which is the key and the value.
Key features of the key-value store:
Simplicity.
Scalability.
Speed.
Column Oriented Databases:
A column-oriented database is a non-relational database that stores the data in columns instead of rows. That means when we want to run analytics on a small number of columns, you can read those columns directly without consuming memory with the unwanted data.

Columnar databases are designed to read data more efficiently and retrieve the data with greater speed. A columnar database is used to store a large amount of data. Key features of columnar oriented database:
Scalability.
Compression.
Very responsive.
Graph-Based databases:
Graph-based databases focus on the relationship between the elements. It stores the data in the form of nodes in the database. The connections between the nodes are called links or relationships.
Key features of graph database:
In a graph-based database, it is easy to identify the relationship between the data by using the links.
The Query's output is real-time results.
The speed depends upon the number of relationships among the database elements.
Updating data is also easy, as adding a new node or edge to a graph database is a straightforward task that does not require significant schema changes.
Refer Text Book Page No : 59-60

5. Explain HDFS in detail along with its commands.

Listing Files and Directories:

Command: **hadoop fs -ls /path**
Description: Lists files and directories in the specified path.

Creating a Directory:
Command: **hadoop fs -mkdir /new_directory**
Description: Creates a new directory in the HDFS file system.

Copying from Local to HDFS:
Command: **hadoop fs -copyFromLocal local_path hdfs_path**
Description: Copies a file or directory from the local file system to HDFS.

Copying from HDFS to Local:
Command: **hadoop fs -copyToLocal hdfs_path local_path**
Description: Copies a file or directory from HDFS to the local file system.

Moving or Renaming a File/Directory:
Command: **hadoop fs -mv old_path new_path**
Description: Moves or renames a file or directory in HDFS.

Removing a File or Directory:
Command: **hadoop fs -rm /path**
Description: Removes a file in HDFS.

Displaying File Content:
Command: **hadoop fs -cat /file**
Description: Displays the content of a file in the HDFS.
Refer Text Book Page No : 93-95

6. Illustrate various components to interact with Hadoop Ecosystem.

Following are the components that collectively form a Hadoop ecosystem:

HDFS: Hadoop Distributed File System
YARN: Yet Another Resource Negotiator
MapReduce: Programming based Data Processing
Spark: In-Memory data processing
PIG, HIVE: Query based processing of data services
HBase: NoSQL Database
Mahout, Spark MLLib: Machine Learning algorithm libraries
Solar, Lucene: Searching and Indexing
Zookeeper: Managing cluster
Oozie: Job Scheduling
key components in the Hadoop ecosystem. Here's a brief summary of each component:

HDFS (Hadoop Distributed File System):

Description: A distributed file system that provides scalable and reliable storage for large datasets across multiple nodes. It breaks large files into smaller blocks and distributes them across the cluster.

YARN (Yet Another Resource Negotiator):

Description: A resource management layer that allows multiple data processing engines to share resources in a Hadoop cluster. YARN separates resource management from application logic.

MapReduce:

Description: A programming model and processing engine for distributed computing. MapReduce divides tasks into Map and Reduce phases and processes them in parallel across the nodes of a Hadoop cluster.

Spark:

Description: An open-source, in-memory data processing engine for large-scale data processing. Spark provides APIs for various programming languages and supports diverse workloads, including batch processing, interactive queries, and machine learning.

PIG and HIVE:

PIG Description: A high-level scripting language for processing and analyzing large datasets. Pig scripts are translated into MapReduce jobs.
HIVE Description: A data warehousing infrastructure built on top of Hadoop. Hive provides a SQL-like query language (HiveQL) for querying and analyzing data in Hadoop.

HBase:

Description: A distributed, scalable, and NoSQL database that provides real-time read/write access to large datasets. HBase is suitable for sparse data with low-latency requirements.

Mahout and Spark MLLib:

Mahout Description: A library of scalable machine learning algorithms. Mahout is designed for distributed computing and integration with Hadoop.
Spark MLLib Description: A machine learning library for Spark that provides scalable implementations of various machine learning algorithms.

Solr and Lucene:

Solr Description: An open-source search platform that provides full-text search capabilities. Solr is often used for building search applications.
Lucene Description: A high-performance, full-featured text search engine library that is used as the core search technology in Solr.

Zookeeper:

Description: A distributed coordination service that provides primitives for building distributed applications. Zookeeper is used for managing configuration information, synchronization, and group services in a Hadoop cluster.

Oozie:

Description: A workflow scheduler system for managing and scheduling Hadoop jobs. Oozie allows users to define and execute workflows that consist of multiple Hadoop jobs and other processing tasks.

This comprehensive set of components addresses various aspects of big data processing, storage, analysis, and coordination within the Hadoop ecosystem. Each component plays a specific role in enabling organizations to build scalable and distributed data processing solutions.

Refer Text Book Page No : 104-105

7. List the various hadoop distributions available in the market.

Hadoop distributions were available in the market, each offering a set of tools, services, and support for big data processing and analytics. It's essential to note that the landscape may have evolved since then, and new distributions or changes to existing ones may have occurred. As of my last update, some prominent Hadoop distributions include:

Cloudera Distribution for Hadoop (CDH):

Description: Cloudera is one of the leading providers of Hadoop-based solutions. CDH includes a comprehensive set of Hadoop ecosystem components, management tools, and additional features for enterprise-grade big data processing.

Hortonworks Data Platform (HDP):

Description: Hortonworks, now part of Cloudera, offered the Hortonworks Data Platform. HDP was an open-source Hadoop distribution that included various Apache projects for data processing, storage, and management.

MapR Distribution:

Description: MapR provided a distribution that focused on delivering performance, reliability, and ease of use for big data and analytics. MapR featured its own file system, MapR-FS, and a range of additional components.

Apache Hadoop (Community Distribution):

Description: Apache Hadoop itself is an open-source project, and users can download and configure the core Hadoop components directly from the Apache Software Foundation. This is often referred to as the "community distribution."

IBM Open Platform (IOP):

Description: IBM offered the Open Platform for Apache Hadoop (IOP), which was an open-source Hadoop distribution that included various Apache projects along with additional IBM tools and support.

Amazon EMR (Elastic MapReduce):

Description: Amazon EMR is a cloud-based Hadoop distribution provided by Amazon Web Services (AWS). It allows users to launch and manage Hadoop clusters on the AWS cloud, leveraging resources on-demand.

Microsoft Azure HDInsight:

Description: HDInsight is a cloud-based Hadoop distribution provided by Microsoft Azure. It allows users to deploy and manage Hadoop clusters on the Azure cloud, integrating with other Azure services.

Pivotal HD:

Description: Pivotal HD was an Hadoop distribution offered by Pivotal Software, which focused on providing a high-performance and easy-to-use platform for big data analytics. Pivotal has undergone changes, and its assets were acquired by VMware in 2020.

8. List and explain the important features of Hadoop

Several Hadoop features simplify the handling of large amounts of data. Let us explore the range of fundamental features of Hadoop that make it preferred by professionals in big data and various industries. They are as follows –

1. Open-Source Framework

Hadoop is a free and open-source framework. That means the source code is freely available online to everyone. This code can be customized to meet the needs of businesses.

2. Cost-Effectiveness

Hadoop is a cost-effective model because it makes use of open-source software and low-cost commodity hardware. Numerous nodes make up the Hadoop cluster. These nodes are a collection of commodity hardware (servers or physical workstations). They are also reasonably priced and offer a practical way to store and process large amounts of data.

3. High-Level Scalability

The Hadoop cluster is both horizontally and vertically scalable. Scalability refers to –

Horizontal Scalability: This means the addition of any number of nodes in the cluster.
Vertical Scalability: This means an increase in the hardware capacity (data storage capacity) of the nodes.
This high-level and flexible scalability of Hadoop offers powerful processing capabilities.

4. Fault Tolerance

Hadoop uses a replication method to store copies of data in each block on different machines (nodes). This replication mechanism makes Hadoop a fault-tolerant framework because if any of the machines fail or crash, a replica copy of the same data can be accessed on other machines. With the more recent Hadoop 3 version, the fault tolerance feature has been enhanced. It employs a replication mechanism called ‘Erasure Coding’ to provide fault tolerance while using less space, with a storage overhead of no more than 50 percent.

5. High-Availability of Data

The high availability of data and fault tolerance features are complementary to each other. The fault tolerance feature provides data availability at any time if any of the DataNode or NameNode fails or crashes. A copy of the same data is available on either of these three platforms. Let us see how –

Case 1: Availability of Data in Case of a DataNode Failure
Even if a data node fails, the cluster is still accessible to users in the following ways –

HDFS (Hadoop Distributed File System) stores copies of files on different nodes, and by default, the DataNodes send NameNode heartbeat signals every three seconds.
The DataNode is considered dead by the NameNode if it does not receive a heartbeat signal after a specified time (10 minutes).
Replication of data begins when NameNodes instruct DataNodes that already have a copy of the data to duplicate it on additional DataNodes.
When a user seeks access, NameNode sends it the IP address of the nearest DataNode that has the requested data.
If the requested DataNode cannot be accessed, NameNode directs the user to another DataNode that has the same data.
Case 2: Availability of Data in Case of NameNode Failure
To use the file system in the cluster, access to NameNode is required. There are two NameNode configurations – active and passive. An active NameNode is a running node, whereas the passive NameNode is the standby node in the cluster. The passive NameNode assumes responsibility for providing uninterrupted client service if the currently active NameNode fails.

6. Data-Reliability

Data reliability in Hadoop is because of the –

Replication mechanism in HDFS (Hadoop Distributed File System), which creates a duplicate of each block, HDFS reliably stores data on the nodes.
Block Scanner, Volume Scanner, Disk Checker, and Directory Scanner are built-in mechanisms provided by the framework itself to ensure data reliability.
7. Faster Data Processing
Hadoop has overcome traditional data processing challenges which were slow and sluggish. Enough resources were not available to process the data smoothly and fast. The distributive storage property of the Hadoop cluster of nodes enables faster data processing of huge amounts of data at lightning speed.

8. Data Locality

This is one of the most distinctive features of Hadoop. The MapReduce component has a feature called data locality. This attribute helps by placing calculation logic close to where the actual data is located on the node. As a result, network congestion is reduced and system performance is improved overall.

9. Possibility of Processing All Types of Data

The Hadoop framework can process all types of data – structured, semi-structured, and unstructured that includes databases, images, videos, audio, graphics, etc. It makes it possible for the users to work with any kind of data for analysis and interpretation irrespective of their format or size. In the Hadoop ecosystem, various tools like Apache Hive, Pig, Sqoop, Zookeeper, GraphX, etc., can process the data.

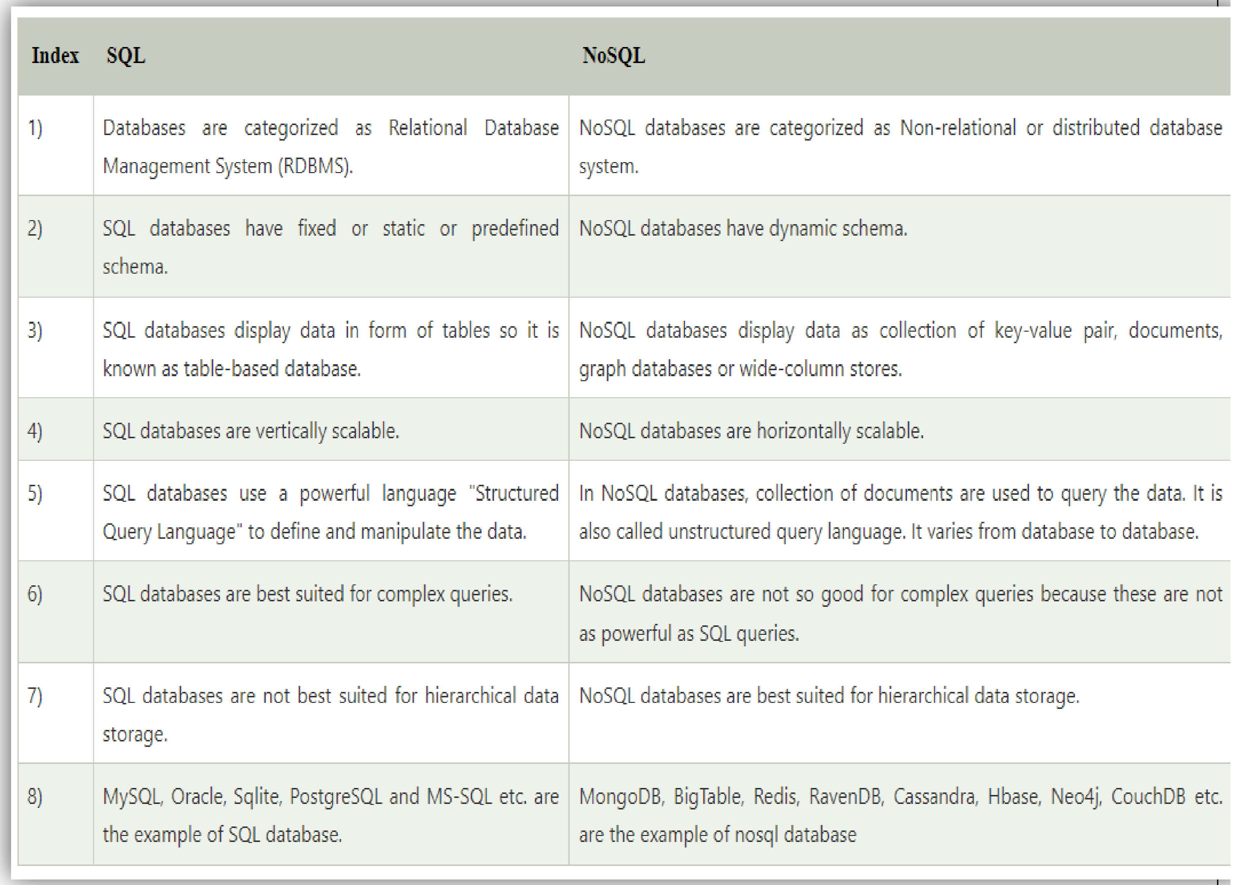
10. Easy Operability

Hadoop is an open-source software framework written in Java. To work on it, users must be familiar with programming languages such as SQL, Java, and others. The framework handles the complete data processing and storage dissemination process. Therefore, it is easy to operate on it.

Refer Text Book Page No : 65

9. Compare between SQL versus NoSQL databases.

There are a lot of databases used today in the industry. Some are SQL databases, some are NoSQL databases. The conventional database is SQL database system that uses tabular relational model to represent data and their relationship. The NoSQL database is the newer one database that provides a mechanism for storage and retrieval of data other than tabular relations model used in relational databases.
Following is a list of differences between SQL and NoSQL database:



Index	SQL	NoSQL
1)	Databases are categorized as Relational Database Management System (RDBMS).	NoSQL databases are categorized as Non-relational or distributed database system.
2)	SQL databases have fixed or static or predefined schema.	NoSQL databases have dynamic schema.
3)	SQL databases display data in form of tables so it is known as table-based database.	NoSQL databases display data as collection of key-value pair, documents, graph databases or wide-column stores.
4)	SQL databases are vertically scalable.	NoSQL databases are horizontally scalable.
5)	SQL databases use a powerful language "Structured Query Language" to define and manipulate the data.	In NoSQL databases, collection of documents are used to query the data. It is also called unstructured query language. It varies from database to database.
6)	SQL databases are best suited for complex queries.	NoSQL databases are not so good for complex queries because these are not as powerful as SQL queries.
7)	SQL databases are not best suited for hierarchical data storage.	NoSQL databases are best suited for hierarchical data storage.
8)	MySQL, Oracle, Sqlite, PostgreSQL and MS-SQL etc. are the example of SQL database.	MongoDB, BigTable, Redis, RavenDB, Cassandra, Hbase, Neo4j, CouchDB etc. are the example of nosql database

10. What are the advantages of Hadoop? Explain Hadoop Architecture and its Components with proper diagram.

Hadoop, an open-source framework for distributed storage and processing of large datasets, offers several advantages that make it a popular choice for big data analytics. Here are some key advantages of Hadoop:

Scalability:

Description: Hadoop is designed to scale horizontally, allowing organizations to easily add more nodes to their cluster as data volume and processing needs grow. This scalability makes it well-suited for handling vast amounts of data.

Cost-Effectiveness:

Description: Hadoop runs on commodity hardware, eliminating the need for expensive specialized hardware. It leverages the economics of scale, making it a cost-effective solution for storing and processing large datasets.

Fault Tolerance:

Description: Hadoop provides fault tolerance by replicating data across multiple nodes in the cluster. In the event of node failures, the system can recover by accessing replicated copies of the data from other nodes, ensuring data integrity and availability.

Flexibility with Data Variety:

Description: Hadoop can handle diverse types of data, including structured, semi-structured, and unstructured data. This flexibility allows organizations to store and analyze data in various formats, such as text, JSON, XML, and more.

Distributed Storage (HDFS):

Description: The Hadoop Distributed File System (HDFS) allows for the distributed storage of large files across multiple nodes. It breaks files into smaller blocks and distributes them across the cluster, enabling efficient storage and retrieval of data.

Parallel Processing (MapReduce):

Description: Hadoop uses the MapReduce programming model for distributed processing. It divides tasks into smaller sub-tasks and processes them in parallel across nodes, resulting in faster data processing for large-scale analytics.

Data Locality:

Description: Hadoop optimizes data processing by executing tasks on nodes where the data is stored. This minimizes data movement across the network, reducing latency and improving overall processing efficiency.

Community Support:

Description: Hadoop benefits from a large and active open-source community. This community support ensures continuous development, improvement, and innovation in the Hadoop ecosystem. Users can access a wealth of resources, documentation, and forums for assistance.

Compatibility with Various Tools:

Description: Hadoop is compatible with a wide range of tools and technologies in the big data ecosystem. This compatibility allows organizations to integrate Hadoop with other tools such as Hive, Pig, Spark, and more, enhancing its capabilities for data processing and analytics.

Adaptability to Changing Workloads:

Description: Hadoop is well-suited for batch processing of large datasets, but it can also adapt to changing workloads. The ecosystem includes tools like Apache Spark that support real-time and iterative processing, providing versatility for different analytical requirements.

Security Features:

Description: Hadoop has implemented security features to protect data and resources. This includes authentication, authorization, and encryption mechanisms. Components like Apache Ranger contribute to fine-grained access control and auditing.

Support for Multi-Tenancy:

Description: Hadoop can support multi-tenancy, allowing multiple users or applications to share the same cluster resources while maintaining isolation. This is beneficial in scenarios where different departments or projects utilize the same Hadoop infrastructure.

Support for Structured and Semi-Structured Data:

Description: Hadoop ecosystem components like Apache Hive provide a SQL-like query language, making it easier for users familiar with relational databases to analyze structured and semi-structured data stored in HDFS.

These advantages make Hadoop a powerful and versatile solution for organizations dealing with large-scale data processing and analytics.