**CHAPTER 2**

1) Explain the 5 V’s present in the BDA (Big Data Analytics) .

Certainly! The 5 V's of Big Data provide a comprehensive framework for understanding the key characteristics of large and complex datasets. Here's a broad explanation of each V:

**1. \*\*Volume:\*\***

- \*Definition:\* Refers to the sheer quantity of data generated and collected. Big Data involves massive amounts of information, often ranging from terabytes to petabytes and beyond.

- \*Example:\* Social media posts, sensor readings, financial transactions.

**2. \*\*Variety:\*\***

- \*Definition:\* Encompasses the diverse types and formats of data. Big Data includes structured, semi-structured, and unstructured data from various sources and in different formats.

- \*Example:\* Text, images, videos, log files, emails, sensor data.

**3. \*\*Velocity:\*\***

- \*Definition:\* Describes the speed at which data is generated, processed, and updated. Big Data often involves high-velocity data streams, requiring real-time or near-real-time processing.

- \*Example:\* Social media updates, streaming data from sensors, online transactions.

**4. \*\*Veracity:\*\***

- \*Definition:\* Focuses on the reliability and accuracy of data. Deals with data quality issues, including noise, errors, and inconsistencies.

- \*Example:\* Incomplete or inaccurate data, data from unreliable sources.

**5. \*\*Value:\*\***

- \*Definition:\* Represents the ability to turn data into meaningful insights and business value. It emphasizes the ultimate goal of deriving actionable information from Big Data.

- \*Example:\* Extracting customer preferences from purchase data, optimizing business processes based on data insights.

These V's collectively highlight the complexity of Big Data and the challenges associated with managing, processing, and extracting value from large and diverse datasets. Organizations need advanced technologies and strategies to address these characteristics and harness the potential insights for informed decision-making and business success.

**2) Explain big data analytics ? Explain second school of thoughts of BDA .**

Big Data Analytics (BDA) refers to the process of examining large and complex datasets, often referred to as "big data," to uncover hidden patterns, correlations, and other valuable insights. The goal of big data analytics is to extract meaningful information from vast amounts of data that traditional data processing applications may not be able to handle efficiently.

There are two main schools of thought in Big Data Analytics:

**1. \*\*Classical School of Thought:\*\***

- \*\*Batch Processing:\*\* This school of thought focuses on batch processing, where large volumes of data are collected over a period of time, stored, and then processed in batches. Hadoop, a popular open-source framework, is often associated with this approach. It divides large datasets into smaller chunks and processes them in parallel across a distributed cluster of computers.

- \*\*MapReduce Paradigm:\*\* This paradigm, popularized by Google and implemented in frameworks like Apache Hadoop, involves breaking down complex tasks into smaller sub-tasks that can be processed independently. The MapReduce programming model is designed to process large-scale data in a parallel and distributed fashion.

- \*\*Scalability:\*\* The classical school emphasizes scalability in terms of handling large volumes of data and distributing processing across multiple nodes to achieve high performance.

**2. \*\*Modern School of Thought:\*\***

- \*\*Real-time Processing:\*\* The modern school of thought focuses on real-time or near-real-time processing of data. Instead of storing data first and processing it later in batches, this approach involves analyzing data as it is generated, allowing organizations to make quicker and more informed decisions.

- \*\*Stream Processing:\*\* Technologies like Apache Kafka and Apache Flink are associated with this school of thought. Stream processing enables the analysis of data in motion, providing insights into rapidly changing datasets.

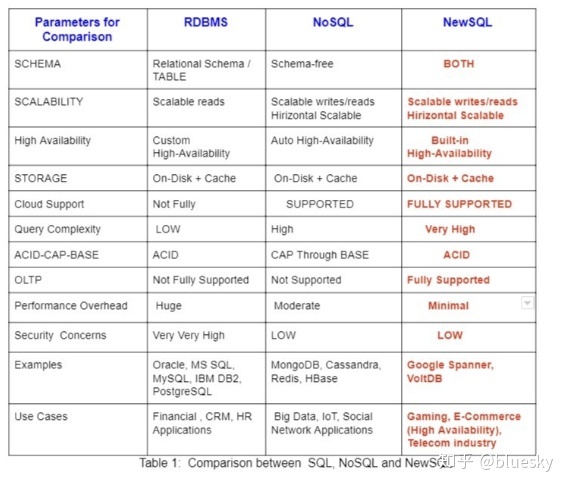
- \*\*In-Memory Computing:\*\* This involves processing data in the computer's random-access memory (RAM) rather than reading and writing to disk, which can significantly improve processing speed. Technologies like Apache Spark leverage in-memory computing for faster analytics.

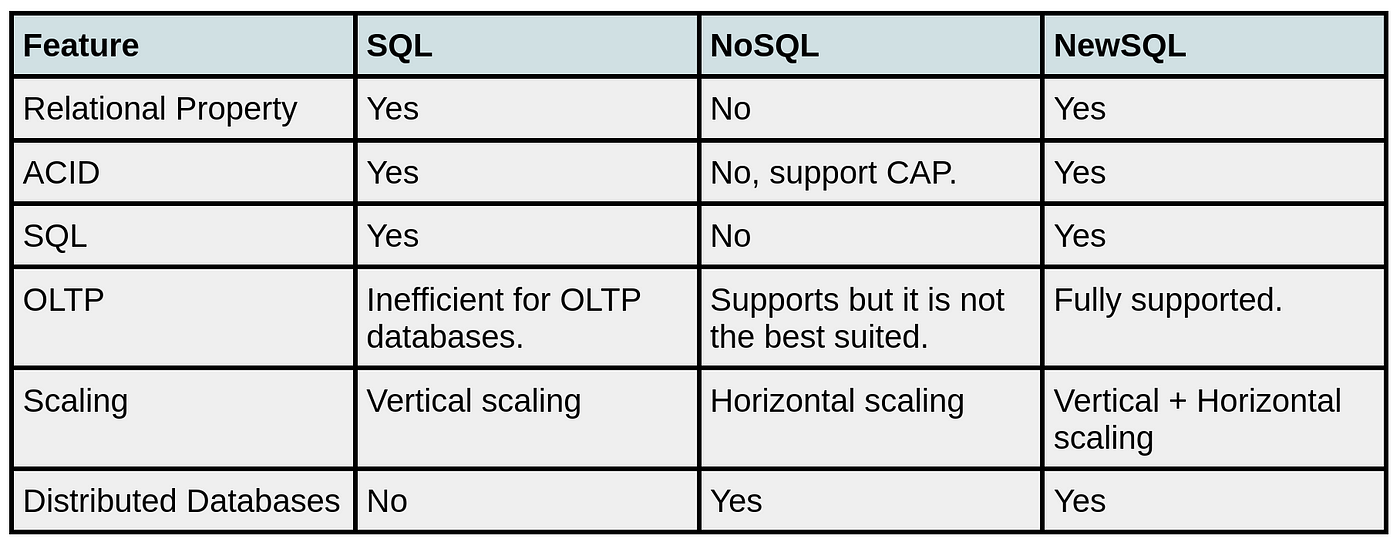
- \*\*Machine Learning Integration:\*\* The modern school often integrates machine learning algorithms into the analytics process to enable predictive modeling, pattern recognition, and automated decision-making.

Both schools of thought have their strengths and weaknesses, and the choice between them depends on the specific requirements and nature of the data being analyzed. Organizations often adopt a hybrid approach that combines elements from both schools to create a flexible and effective big data analytics strategy.

**CHAPTER 3**

**1) Explain Difference between SQL , NoSQL and NewSQL .**





**2) What is Big Data Analytics? Explain the classification of BDA ?**

Big Data Analytics (BDA) is the process of examining and interpreting large and complex datasets, commonly known as big data, to uncover valuable insights, patterns, trends, and other meaningful information. The goal is to extract knowledge from massive volumes of structured and unstructured data that traditional data processing tools and methods might struggle to handle efficiently.

The classification of Big Data Analytics can be broadly categorized into four types, often referred to as the "Four V's" of big data:

1. \*\*Volume:\*\*

- \*\*Description:\*\* Volume refers to the sheer size of the data. Big data involves datasets that are too large to be processed by traditional database systems.

- \*\*Challenge:\*\* The challenge is to efficiently store, process, and analyze massive volumes of data. Technologies like Hadoop Distributed File System (HDFS) are used for distributed storage, and batch processing frameworks like Apache Hadoop handle large-scale data processing.

2. \*\*Velocity:\*\*

- \*\*Description:\*\* Velocity represents the speed at which data is generated, processed, and analyzed. Big data applications often deal with real-time or near-real-time data streams.

- \*\*Challenge:\*\* The challenge is to process and analyze data as it is generated to make timely decisions. Stream processing frameworks like Apache Kafka and Apache Flink are employed for real-time analytics.

3. \*\*Variety:\*\*

- \*\*Description:\*\* Variety refers to the diverse types of data, including structured, semi-structured, and unstructured data. This can include text, images, videos, social media posts, and more.

- \*\*Challenge:\*\* The challenge is to handle and analyze data in various formats. NoSQL databases and technologies like Apache Spark support the processing of diverse data types.

4. \*\*Veracity:\*\*

- \*\*Description:\*\* Veracity focuses on the reliability and accuracy of the data. Big data analytics deals with data from various sources, and ensuring data quality is crucial.

- \*\*Challenge:\*\* The challenge is to deal with uncertain or unreliable data. Data cleansing, validation, and quality assurance processes are essential in addressing veracity issues.

In addition to these "Four V's," a fifth "V" is sometimes added:

5. \*\*Value:\*\*

- \*\*Description:\*\* Value represents the importance of deriving actionable insights and value from the big data analysis process.

- \*\*Challenge:\*\* The challenge is to ensure that the insights gained from big data analytics contribute meaningfully to business objectives and decision-making.

These classifications help organizations understand and address the specific challenges associated with big data, enabling them to implement effective strategies for extracting valuable insights from their data assets.

Certainly! Let's break down Analytics 1.0, 2.0, and 3.0 in simple terms:

**1. \*\*Analytics 1.0:\*\***

- \*\*What it is:\*\* Basic analysis and reporting of data.

- \*\*Imagine:\*\* Looking at numbers like how many people visited a website.

- \*\*Problem:\*\* Sometimes, people just report numbers without really understanding or improving anything.

**2. \*\*Analytics 2.0:\*\***

- \*\*What it is:\*\* Smarter analysis that started around the 2000s.

- \*\*Imagine:\*\* Not just counting visitors but predicting future trends and understanding more about what's happening.

- \*\*Tools:\*\* Used technologies like big data (handling lots of data) and predictive models.

**3. \*\*Analytics 3.0:\*\***

- \*\*What it is:\*\* Connecting data from everywhere, especially from things like smart devices.

- \*\*Imagine:\*\* Now, it's not just about websites. It's about understanding and making decisions from data all around us.

- \*\*Benefit:\*\* Helps companies make quicker and better decisions using data from different sources.

**4. \*\*Future Trend - Quantum Analytics:\*\***

- \*\*What it might be:\*\* Using super-advanced computers to analyze data crazy fast.

- \*\*Imagine:\*\* Computers that can solve problems much quicker than today's computers.

- \*\*Status:\*\* It's like a cool idea for the future, but we're still figuring out how to make it work.

So, Analytics 1.0 is like looking at basic numbers, Analytics 2.0 is about being smarter with data, Analytics 3.0 is connecting data from everywhere, and Quantum Analytics is a futuristic idea about using super-fast computers for even better analysis.

**3) Explain massively parallel processing terminologies used in big data.**

Massively Parallel Processing (MPP) is a key concept in big data analytics, particularly when dealing with large-scale datasets that require significant computational power. MPP involves the simultaneous processing of data across multiple processors or nodes, allowing for efficient and parallel execution of complex queries and analyses. Here are some terminologies commonly associated with Massively Parallel Processing in the context of big data:

**1. \*\*Parallel Computing:\*\***

- \*\*Definition:\*\* Parallel computing is a type of computation in which many calculations or processes are carried out simultaneously. It contrasts with serial computing, where instructions are executed one at a time.

- \*\*Application:\*\* In big data analytics, parallel computing is essential for handling large volumes of data efficiently. MPP systems divide tasks into smaller sub-tasks that can be processed in parallel, improving overall performance.

**2. \*\*Node:\*\***

- \*\*Definition:\*\* In the context of MPP, a node refers to an individual computational unit within a parallel processing system. Each node typically has its own processors, memory, and storage.

- \*\*Application:\*\* Nodes work together to process data in parallel. In a distributed computing environment, nodes communicate and coordinate to perform complex tasks.

**3. \*\*Cluster:\*\***

- \*\*Definition:\*\* A cluster is a group of interconnected computers or nodes that work together to perform parallel processing tasks. Clusters are a common architecture in MPP systems.

- \*\*Application:\*\* In big data analytics, a cluster enables the distribution of data and computation across multiple nodes, allowing for parallelization of tasks and improved scalability.

**4. \*\*Partitioning:\*\***

- \*\*Definition:\*\* Partitioning involves dividing a large dataset into smaller, manageable partitions that can be processed independently in parallel.

- \*\*Application:\*\* In MPP databases, partitioning is used to distribute data across different nodes or servers. Each node processes its assigned partition of data, contributing to overall parallel processing efficiency.

**5. \*\*Shuffling:\*\***

- \*\*Definition:\*\* Shuffling refers to the process of redistributing and exchanging data between nodes in a parallel processing system.

- \*\*Application:\*\* During certain operations, such as joins or aggregations, data needs to be shuffled between nodes to ensure that relevant information is available for further processing. Efficient shuffling is crucial for optimizing MPP performance.

**6. \*\*Query Parallelization:\*\***

- \*\*Definition:\*\* Query parallelization involves breaking down a query into smaller sub-queries that can be executed in parallel across multiple nodes.

- \*\*Application:\*\* MPP databases use query parallelization to distribute the workload and accelerate query execution. Each node processes its portion of the query independently.

**7. \*\*Distributed File System:\*\***

- \*\*Definition:\*\* A distributed file system is a file system that spans multiple nodes or servers in a cluster, providing access to a shared pool of storage.

- \*\*Application:\*\* Distributed file systems, such as Hadoop Distributed File System (HDFS), play a crucial role in MPP environments by enabling distributed storage and access to large datasets across multiple nodes.

**4) Explain various terminology used in big data environments**

Certainly! Here are the top six terminologies in the context of big data environments:

**1. \*\*Big Data:\*\***

- \*\*Definition:\*\* Extremely large and complex datasets that traditional data processing applications may struggle to handle.

- \*\*Importance:\*\* Signifies the volume, velocity, variety, and veracity challenges associated with large-scale data.

**2. \*\*Hadoop:\*\***

- \*\*Definition:\*\* An open-source framework for distributed storage and processing of big data.

- \*\*Importance:\*\* Provides scalable and distributed computing capabilities, including the Hadoop Distributed File System (HDFS) and MapReduce.

**3. \*\*NoSQL:\*\***

- \*\*Definition:\*\* Stands for "Not Only SQL," referring to a class of databases that do not strictly use traditional SQL relational database management systems.

- \*\*Purpose:\*\* Used for handling large volumes of unstructured or semi-structured data in a flexible manner.

**4. \*\*MapReduce:\*\***

- \*\*Definition: \*\* A programming model and processing engine for distributed computing of large datasets.

- \*\*Role:\*\* Divides tasks into smaller sub-tasks processed in parallel across a distributed cluster.

**5. \*\*Machine Learning:\*\***

- \*\*Definition:\*\* A subset of artificial intelligence that involves the development of algorithms allowing computers to learn and make predictions from data.

- \*\*Application:\*\* Used in big data environments for predictive analytics and pattern recognition.

**6. \*\*Data Lake:\*\***

- \*\*Definition:\*\* A centralized repository that allows for the storage of structured and unstructured data at any scale.

- \*\*Purpose:\*\* Enables organizations to store diverse data types for analytics and reporting in a flexible and cost-effective manner.

Certainly! Here are two additional terminologies in the context of big data environments:

**7. \*\*Data Warehouse:\*\***

- \*\*Definition:\*\* A centralized repository for storing and managing structured data from different sources.

- \*\*Purpose:\*\* Facilitates business intelligence and reporting by providing a consolidated view of structured data for analysis.

**8. \*\*Streaming Analytics:\*\***

- \*\*Definition:\*\* Involves the real-time analysis of streaming data.

- \*\*Purpose:\*\* Enables immediate insights and decision-making based on data as it is generated, allowing organizations to respond swiftly to changing conditions.

**CHAPTER 4**

**1) Give different HDFS commands**

Certainly! Here are eight top HDFS commands along with explanations:

**1. \*\*List Files and Directories (`-ls`):\*\***

hadoop fs -ls <path>

- \*\*Explanation:\*\* Lists files and directories in the specified HDFS path. It provides details such as permissions, owner, size, and modification date.

**2. \*\*Create Directory (`-mkdir`):\*\***

hadoop fs -mkdir <directory>

```

- \*\*Explanation:\*\* Creates a new directory in HDFS at the specified path.

**3. \*\*Copy From Local (`-copyFromLocal`):\*\***

hadoop fs -copyFromLocal <local-source> <hdfs-destination>

```

- \*\*Explanation:\*\* Copies a file or directory from the local file system to HDFS.

**4. \*\*Copy To Local (`-copyToLocal`):\*\***

hadoop fs -copyToLocal <hdfs-source> <local-destination>

```

- \*\*Explanation:\*\* Copies a file or directory from HDFS to the local file system.

**5. \*\*Remove File or Directory (`-rm`):\*\***

hadoop fs -rm <hdfs-path>

```

- \*\*Explanation:\*\* Removes a file or directory from HDFS.

**6. \*\*Move or Rename File (`-mv`):\*\***

hadoop fs -mv <source> <destination>

```

- \*\*Explanation:\*\* Moves or renames a file or directory in HDFS.

**7. \*\*Display File Content (`-cat`):\*\***

hadoop fs -cat <hdfs-file>

```

- \*\*Explanation:\*\* Displays the content of a file in HDFS.

**8. \*\*Check Cluster Health (`dfsadmin -report`):\*\***

hadoop dfsadmin -report

```

- \*\*Explanation:\*\* Provides a report on the health and status of the HDFS cluster, including information about the number of DataNodes, overall capacity, and usage.

These commands cover fundamental operations for navigating, managing, and interacting with files and directories in the Hadoop Distributed File System (HDFS).

**2) What are the different key aspects of hadoop**

The key aspects of Hadoop can be summarized by the following attributes:

**1. \*\*Open-Source:\*\***

- \*\*Definition:\*\* Hadoop is an open-source framework, meaning that its source code is freely available to the public. This encourages collaboration, innovation, and a large community of developers contributing to its improvement.

**2. \*\*Framework:\*\***

- \*\*Definition:\*\* Hadoop is not a standalone application but rather a framework or ecosystem of tools and components. It provides a structure for distributed storage and processing of large datasets.

**3. \*\*Distributed:\*\***

- \*\*Definition:\*\* Hadoop operates in a distributed computing environment, meaning that it processes and stores data across multiple machines (nodes) in a cluster. This enables parallel and scalable processing of large datasets.

**4. \*\*Massive Storage:\*\***

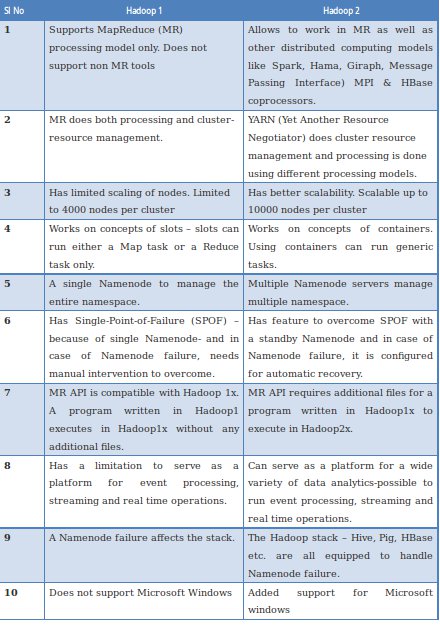
- \*\*Definition:\*\* Hadoop is designed to handle massive amounts of data. It uses the Hadoop Distributed File System (HDFS) to distribute and store data across the nodes in a Hadoop cluster. This allows for the efficient storage of petabytes or more of data.

**5. \*\*Faster Processing:\*\***

- \*\*Definition:\*\* Hadoop facilitates faster processing of large datasets through its parallel processing model. The MapReduce programming paradigm, a core component of Hadoop, breaks down tasks into smaller sub-tasks that can be processed concurrently on different nodes, leading to faster overall computation.

These aspects collectively make Hadoop a powerful and flexible solution for big data processing and analysis. The open-source nature encourages collaboration and innovation, the framework provides a structure for organizing and managing tasks, the distributed architecture allows for scalability, massive storage capability supports the handling of large datasets, and the parallel processing model contributes to faster data processing.

**3) Give the difference between Hadoop 1.0 and Hadoop 2.0**



**5) Explain high level architecture of Hadoop with its components**

**6) Describe YARN architecture in detail**

**CHATPER 5**

**1) Explain different CRUD operations in MongoDB.**

MongoDB is a NoSQL database that stores data in a flexible, JSON-like format called BSON (Binary JSON). CRUD operations stand for Create, Read, Update, and Delete, which are the fundamental operations for interacting with data in a database. Here's an explanation of how CRUD operations work in MongoDB:

1. \*\*Create (Insert):\*\*

- MongoDB uses the `insert` operation to create new documents in a collection. A document in MongoDB is similar to a record in a relational database. The basic syntax for inserting a document is as follows:

```javascript

db.collectionName.insert({ field1: value1, field2: value2, ... });

```

Example:

```javascript

db.users.insert({ name: "John Doe", age: 30, email: "john@example.com" });

```

2. \*\*Read (Query):\*\*

- MongoDB provides the `find` operation for reading or querying documents in a collection. The basic syntax for a query is as follows:

```javascript

db.collectionName.find({ /\* query criteria \*/ });

```

Example:

```javascript

// Find all documents in the 'users' collection

db.users.find();

// Find users with age greater than 25

db.users.find({ age: { $gt: 25 } });

```

3. \*\*Update:\*\*

- The `update` operation is used to modify existing documents in a collection. The basic syntax for an update is as follows:

```javascript

db.collectionName.update({ /\* query criteria \*/ }, { $set: { /\* update fields \*/ } });

```

Example:

```javascript

// Update the age of the user with email "john@example.com"

db.users.update({ email: "john@example.com" }, { $set: { age: 31 } });

```

It's important to note that by default, the `update` method only updates the first matching document. To update multiple documents, you can use the `updateMany` method.

4. \*\*Delete:\*\*

- MongoDB uses the `remove` operation to delete documents from a collection. The basic syntax for removing documents is as follows:

```javascript

db.collectionName.remove({ /\* query criteria \*/ });

```

Example:

```javascript

// Delete the user with email "john@example.com"

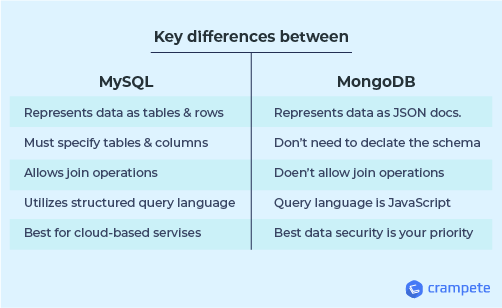
db.users.remove({ email: "john@example.com" });

```

Alternatively, you can use the `deleteOne` or `deleteMany` methods for more control over deletion.

These CRUD operations form the foundation for interacting with data in MongoDB. It's important to note that MongoDB also provides additional features and operators for more complex queries, indexing, and aggregation. Additionally, in recent versions of MongoDB, the `update` method has been deprecated in favor of `updateOne` and `updateMany` for better clarity and control over updates.

**2) Write difference between MongoDB and SQL**



**3) What is MongoDB ?**

MongoDB is a popular open-source NoSQL (non-relational) database management system. It falls under the category of document-oriented databases, which means it stores and retrieves data in a flexible, JSON-like format called BSON (Binary JSON). MongoDB is designed to handle large volumes of unstructured or semi-structured data, making it well-suited for a variety of applications and use cases.

Key features and characteristics of MongoDB include:

**1. \*\*Document-Oriented:\*\***

- MongoDB stores data in documents, which are JSON-like BSON objects. Each document represents a record and can have a flexible schema, meaning different documents in the same collection can have different fields.

**2. \*\*Scalability:\*\***

- MongoDB is designed to scale horizontally by distributing data across multiple servers. This makes it suitable for handling large amounts of data and traffic.

**3. \*\*Schema Flexibility:\*\***

- Unlike traditional relational databases, MongoDB does not require a predefined schema. Fields can be added or removed from documents without affecting other documents in the same collection.

**4. \*\*Query Language:\*\***

- MongoDB uses a rich query language that supports a wide range of queries and operations. It includes support for filtering, sorting, and aggregation.

**5. \*\*Indexing:\*\***

- MongoDB supports the creation of indexes on fields, which can significantly improve query performance. Indexes can be created on single fields or compound indexes on multiple fields.

**6. \*\*High Performance:\*\***

- MongoDB is designed for high-performance reads and writes. It uses techniques like memory-mapped files for efficient storage and retrieval of data.

**7. \*\*Replication and High Availability:\*\***

- MongoDB supports automatic data replication, allowing data to be mirrored across multiple servers. This provides fault tolerance and high availability in case of server failures.

**8. \*\*Sharding:\*\***

- Sharding is the process of distributing data across multiple servers. MongoDB supports sharding to horizontally scale databases and improve performance.

**9. \*\*Aggregation Framework:\*\***

- MongoDB provides a powerful aggregation framework for performing data transformations and computations on the server side. It supports a variety of aggregation stages and operators.

**10. \*\*Geospatial Indexing:\*\***

- MongoDB includes geospatial indexing, allowing for the efficient storage and retrieval of geospatial data.

MongoDB is widely used in various applications, including content management systems, e-commerce platforms, real-time analytics, and more. Its flexibility, scalability, and ease of use make it a popular choice for developers working on projects with dynamic or evolving data requirements.

**4) Explain Map reduce programming in MongoDB with example.**

MapReduce is a programming model and processing technique used for large-scale data processing and computation. MongoDB provides support for MapReduce as a way to perform complex aggregations and transformations on data stored in collections. The MapReduce process in MongoDB involves two main functions: the map function and the reduce function.

Here's an overview of how MapReduce works in MongoDB:

1. \*\*Map Function:\*\*

- The map function is applied to each document in the collection, producing a set of key-value pairs as output. The map function is responsible for extracting and emitting the relevant data for further processing.

2. \*\*Reduce Function:\*\*

- The reduce function processes the output of the map function, grouping data by key and performing aggregation or computation on each group. The result is a reduced set of key-value pairs.

3. \*\*Finalize Function (Optional):\*\*

- An optional finalize function can be applied after the reduce function to perform additional processing on the results.

4. \*\*Output:\*\*

- The final output is stored in a separate collection or returned as the result of the MapReduce operation.

Here's a simple example to illustrate the MapReduce process in MongoDB. Suppose we have a collection called "orders" with documents representing sales orders, and we want to calculate the total sales amount for each product:

```javascript

// Sample document in the 'orders' collection

/\*

{

"\_id": ObjectId("..."),

"product": "Laptop",

"quantity": 2,

"price": 1000

}

\*/

// Map function

var mapFunction = function() {

emit(this.product, this.quantity \* this.price);

};

// Reduce function

var reduceFunction = function(key, values) {

return Array.sum(values);

};

// Run MapReduce

db.orders.mapReduce(

mapFunction,

reduceFunction,

{ out: "product\_totals" } // Output collection name

);

// Query the result

db.product\_totals.find();

```

In this example:

- The map function emits key-value pairs where the key is the product name, and the value is the calculated sales amount for that order.

- The reduce function takes these key-value pairs and sums up the sales amounts for each product.

- The output is stored in a new collection named "product\_totals."

Note that MapReduce in MongoDB can be less efficient than using the aggregation framework for certain tasks. The aggregation framework provides a more expressive and performant way to perform data transformations and aggregations. MapReduce is typically used for complex tasks that cannot be easily achieved with the aggregation framework.