TYBSC-IT

Next Generation Technology (NGT)

SEM 5 (NOV 2022) **Q.P.CODE :10171**

1.Attempt any three of the following:

a. Explain the three aspects of the data :i) Data at rest ii) Data at motion iii) Data in many forms. (5)

Ans : Data plays a crucial role, and understanding its various aspects is essential. The three aspects of data in the next generation technology are data at rest, which is stored information, data in motion, which is actively transmitted or transferred data, and data in many forms, which encompasses the diverse types and formats in which data exists.

1. **Data at Rest:**
- Data at rest refers to information that is stored and not actively being transmitted or processed.
- It typically resides in databases, data warehouses, or other data storage systems.
- This static data can include files, documents, databases, and any other structured or unstructured data stored in a dormant state.

2. **Data in Motion:**
   - Data in motion refers to information that is actively being transmitted or transferred from one location or system to another.
   - This data is in transit, moving across networks or communication channels.
   - Examples of data in motion include emails, file transfers, streaming media, real-time sensor data, and network communication.

3. **Data in Many Forms:**
   - Data in many forms refers to the diverse types and formats in which data exists.
   - With advancements in technology, data can be generated and collected from various sources, such as text documents, images, audio recordings, videos, social media posts.
   - This data can be structured (e.g., databases) or unstructured (e.g., natural language text), and it may also include semi-structured data (e.g., XML or JSON).

b. List the Big Data sources and also explain the challenges of Big Data. (5)

Ans: Common sources of Big Data :

1. **Social media data:** Platforms like Facebook, Twitter, and Instagram generate vast amounts of data through user interactions, posts, and comments.

2. **Sensor data:** Internet of Things (IoT) devices, such as temperature sensors, GPS trackers, and fitness wearables, produce continuous streams of data.
3. **Machine-generated data:** This includes data generated by machines and systems, such as log files, server data, and network traffic.
4. **Transactional data:** Financial transactions, online purchases, and customer interactions generate significant amounts of data in various industries.
5. **Web data:** Data collected from website visits, online searches, and user behavior on websites can provide insights into user preferences and trends.
6. **Geospatial and location data:** GPS data, satellite imagery, and geolocation data contribute to the growing pool of Big Data, enabling applications like mapping and navigation.
7. **Publicly available data:** Open data initiatives, government records, and research databases provide valuable sources of data for analysis.

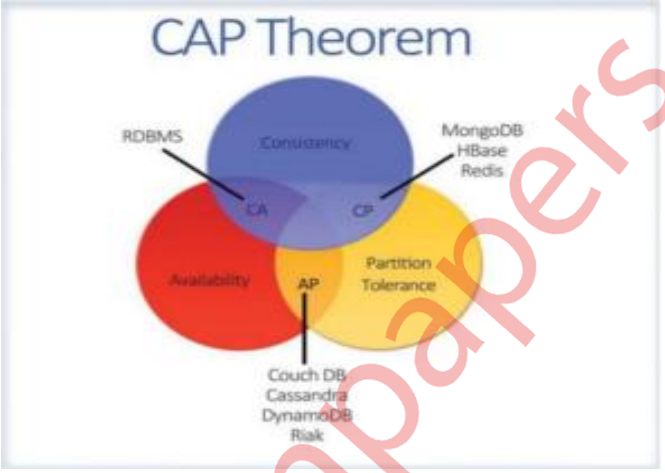
Challenges of Big Data :

1. **Volume:** Managing and storing large volumes of data is a significant challenge. Big Data requires scalable and distributed systems capable of handling massive amounts of information.
2. **Velocity:** The speed at which data is generated can overwhelm traditional data processing systems. Real-time or near-real-time analysis is often necessary to derive meaningful insights from high-velocity data streams.
3. **Variety:** Big Data comes in various formats, including structured, unstructured, and semi-structured data. Processing and integrating these diverse data types can be complex and require advanced techniques.
4. **Complexity:** Big Data often involves complex data relationships and correlations. Analysing such data requires sophisticated algorithms and analytics techniques to uncover hidden patterns and insights.
5. **Privacy and security:** The large-scale collection and analysis of personal data raise privacy concerns.
6. **Scalability and infrastructure:** Processing and analysing Big Data require powerful computing resources and scalable infrastructure capable of handling the data volume and computational demands.

c. Explain Brewer's theorem along with the neat diagram. (5)

Ans :

- Brewer's theorem is also known as CAP theorem, which is a fundamental concept in distributed systems.
- Brewer's theorem, proposed by Eric Brewer in 2000.
- It states that it is impossible for a distributed computer system to simultaneously provide consistency (C), availability (A), and partition tolerance (P).
- The CAP theorem states that in a distributed system, you can only achieve two out of the three properties (C, A, and P) simultaneously. This means that when a network partition occurs, a system must choose between either consistency or availability.



1. **Consistency (C):** Consistency refers to all nodes in a distributed system having the same data at the same time. - data remains consistent after any operation is performed that changes the data, and that all users or clients accessing the application see the same updated data.
2. **Availability (A):** Availability means that the system remains operational and responsive to user requests, even in the presence of failures. It means that the system is available. An available system ensures that clients can always read and write data.
3. **Partition Tolerance (P):** Partition tolerance refers to the system's ability to continue operating and providing availability despite the occurrence of network partitions, which means the network is divided into separate partitions that cannot communicate with each other. It means system will continue to function even if it is partitioned into groups of servers that re not able to communicate with one another.

d. How consistency implemented at both read and write operation levels explain. **(5)**

Ans : Consistency is achieved by combining various techniques such as atomicity, transactions, consensus algorithms, and synchronization mechanisms. In next-generation technology, ensuring consistency at both read and write operation levels is a crucial aspect of designing robust and reliable systems.

- **Consistency at the Write Operation Level:**

A. **Atomicity:** Atomicity ensures that a write operation is treated as a single, indivisible unit. If a write operation involves multiple data updates, either all the updates are applied successfully, or none of them are applied at all. This prevents partial updates and maintains consistency.

B. **Transactional Support:** Transactions provide a way to group multiple write operations together as a single logical unit. ACID (Atomicity, Consistency, Isolation, Durability) properties are often employed to ensure that transactions are executed reliably and consistently. If any part of a transaction fails, the entire transaction is rolled back, ensuring data consistency.

C. **Consensus Algorithms:** Consensus algorithms are used in distributed systems to achieve consistency. These algorithms allow multiple nodes to agree on the order of write operations and ensure that all replicas of data are updated in a consistent manner.

- **Consistency at the Read Operation Level:**

a. Read-after-Write Consistency: This technique ensures that a read operation always returns the most recent value written by a preceding write operation. To achieve this, systems may use various mechanisms, such as maintaining write timestamps or employing synchronization protocols, to guarantee that read operations are performed on the latest consistent state of the data.

b. Replication and Synchronization: In distributed systems, data is often replicated across multiple nodes to enhance performance and availability. To maintain consistency, synchronization mechanisms like quorums or consensus protocols are employed. These ensure that all replicas eventually receive the updates and converge to a consistent state, allowing read operations to provide consistent results.

e. List the categories of NoSQL databases. Also explain the ways in which MongoDB is different from SQL. (5)

Ans : Categories of NoSQL Databases.

Category	Description	Example
Document-Based	These databases store and retrieve data in the form of documents, typically using JSON or XML formats.	MongoDB
Key- value pair based	This type of database stores data as key-value pairs, where each value is associated with a unique key.	CouchDB
Column-based	These databases organize data into columns instead of rows, where each column contains related data. They are ideal for storing and processing large amounts of data across distributed systems.	Cassandra
Graph based	These databases are designed to represent and store data in the form of nodes, edges, and properties, enabling efficient traversal of complex relationships.	Orient DB

- **How MongoDB differs from traditional SQL databases:**

1. **Data model:** MongoDB is a document-oriented database, while SQL databases follow a tabular data model. In MongoDB, data is stored in flexible, self-describing JSON-like documents, allowing dynamic and evolving schemas. SQL databases, on the other hand, require predefined schemas with fixed tables and columns.
2. **Scalability:** MongoDB provides horizontal scalability by employing sharding, which involves distributing data across multiple machines. It can handle massive amounts of data and high read/write workloads efficiently. SQL databases often require complex and time-consuming scaling techniques to achieve similar levels of scalability.
3. **Querying:** MongoDB uses a flexible query language called the MongoDB Query Language (MQL). SQL databases use Structured Query Language (SQL) for querying, which is primarily based on set operations and relational algebra.
4. **Replication and High Availability:** MongoDB provides built-in replication, allowing for automatic failover and data redundancy. It uses replica sets to maintain multiple copies of data across different servers. SQL databases typically require additional configuration and setup for achieving high availability.

f. What is NoSQL. Explain the advantages of NoSQL Databases. **(5)**

Ans : NoSQL, which stands for "Not Only SQL," is a type of database management system (DBMS) that provides an alternative approach to data storage and retrieval compared to traditional relational databases. NoSQL databases are designed to handle large-scale, unstructured, and semi-structured data, making them well-suited for modern applications with high scalability and performance requirements.

- **Advantages of NoSQL databases include:**

1. **Scalability:** NoSQL databases are built to scale horizontally, meaning they can handle increased workloads by adding more servers to distribute the data and processing.
2. **Flexibility with Data Models:** Unlike relational databases that require a predefined schema, NoSQL databases offer flexibility in handling unstructured and evolving data. They can store and retrieve different data types, such as documents, key-value pairs, graph data, and wide-column stores.
3. **High Performance:** NoSQL databases often provide faster read and write operations compared to traditional relational databases. By optimizing data access patterns and using distributed architectures, NoSQL databases can deliver low latency and high throughput, especially for scenarios with heavy read or write workloads.
4. **Horizontal Scalability:** NoSQL databases can scale horizontally by distributing data across multiple servers or nodes. This architecture providing increased storage capacity and improved performance without sacrificing data availability.
5. **High Availability:** NoSQL databases are designed with fault-tolerant mechanisms to ensure high availability of data. They typically replicate data across multiple nodes, allowing for automatic failover and data redundancy.
6. **Cost-Effective:** NoSQL databases often leverage commodity hardware and can run on clusters of low-cost servers. Additionally, their ability to handle large-scale data without requiring extensive schema design can reduce development and maintenance costs.

2.Attempt any three of the following:

a.Explain _id, capped collection, and Binary Javascript Object Notation(BSON). **(5)**

Ans :

1. **_id :**

- _Id field as the primary key for the collection.
- _id field contains a unique ObjectId value.
- Each document can be uniquely identified in the collection.

- When inserting documents in the collection, if you don't add a field name with the _id in the Field name, then mongodb will automatically add an object id field.
- Specifies the _id field:
  >**db.users.insert({"_id":10, "name": "explicit id"})**
- When explicitly creating an id field, it needs to be created with _id in its name.

2. **Capped Collection :**

- It stores the documents in the collection in the inserted order.
- In this, As the collection reaches its limit, the documents will be removed from the collection in FIFO(Fisrt In First Out) order.
- In capped collection, least recently inserted documents will be removed first.
- It is good for use cases where the order of insertion is required to be maintained automatically
- Capped collection are best for storing log information, cache data, or any other high volume Data.
- In this, fixed-size circular collections that follows the insertion order to support high performance for create, read, and delete operations.
- It uses capped collections for maintaining its replication logs.

3. **BSON(Binary Javascript Object Notation):**
   - MongoDB stores the JSON document in a binary-encoded format termed as BSON. This is termed as BSON.
   - The BSON data model is an extended form of the JSON data model.
   - MongoDB's implementation of a BSON document is fast, highly traversable, and lightweight.
   - It supports embedding of arrays and objects within other arrays, and also enables MongoDB to reach inside the objects to build indexes and match objects.

---

b. What is polymorphic schema? Explain the various reasons for using the polymorphic schema. **(5)**

Ans :

- A polymorphic schema refers to a database schema design that allows different types of data entities to be represented within a single table or collection.
- In other words, it allows for storing heterogeneous or diverse data within a unified structure.
- Traditionally, databases are designed with a fixed schema where each table represents a specific entity or data type, and the columns within the table are predetermined.

- In certain scenarios where the data being stored can vary significantly in terms of structure or attributes, a polymorphic schema provides more flexibility.
- **Reasons for using the polymorphic schema:**

1. **Flexibility and Adaptability:** Polymorphic schemas enable the storage of different types of data in a single table. This allows for greater flexibility in accommodating evolving data requirements.
2. **Simplified Data Model:** With a polymorphic schema, there is no need to create multiple tables to represent different data types. This simplifies the overall data model and reduces the complexity of the database structure.
3. **Efficient Data Storage:** By combining different data types in a single table, a polymorphic schema can reduce the storage overhead associated with maintaining separate tables for each data type. This can be particularly advantageous when dealing with large volumes of data.
4. **Easier Data Integration:** When data from multiple sources needs to be integrated into a unified database, a polymorphic schema can provide a convenient solution. Since different data types can be stored in the same table, it becomes easier to consolidate and merge data from various systems or applications.
5. **Simplified Data Access and Analysis:** A polymorphic schema allows for simpler data access and analysis since all relevant data can be queried using a single table structure.
6. **Support for Dynamic or User-Defined Data:** Polymorphic schemas are particularly useful when dealing with data that has variable attributes or user-defined fields. They provide a mechanism to store such data without explicitly defining the schema for every possible variation.

---

c. How can you create a collection explicitly? Explain about selector and projector with example. (5)

Ans: MongoDB, a collection is a grouping of MongoDB documents. It is analogous to a table in a relational database. You can create a collection explicitly in MongoDB by using the createCollection() method.

- **db.createCollection("collectionName")**

Selectors and projectors in MongoDB :

1. **Selectors:**

- Selectors are used in MongoDB queries to specify criteria for document selection. They allow you to filter documents based on certain conditions.
- In MongoDB, selectors are typically used with the find() method to retrieve documents from a collection that match specific criteria.
- For Example : **db.users.find({ age: 25, city: "New York" })**

2. **Projectors:**

- Projectors allow you to specify which fields should be included or excluded in the result set of a MongoDB query.
- They provide control over the fields that are returned, which can be useful for reducing the amount of data transmitted over the network.
- For Example : **db.users.find({ age: 25, city: "New York" }, { name: 1 })**
  In this above query, the projector { name: 1 } is provided as the second argument to the find() method. It indicates that only the "name" field should be included in the result set. The value of 1 specifies inclusion, while 0 would indicate exclusion.

d. What is the use of FindOne() method? Briefly explain about explain() function. **(5)**

Ans:

1. **findOne() Method:**

- The findOne() method is a query method provided by MongoDB that is used to retrieve a single document from a collection that matches the specified query criteria.
- It returns the first document that satisfies the query conditions and is often used when you only need to retrieve a single result or when you're not concerned about multiple matching documents.
- For Example : **const user = db.users.findOne({ name: "John" });**
  In this example, the findOne() method is used to find a document from the "users" collection where the name field matches the value "John". It returns the entire document that matches the query criteria.

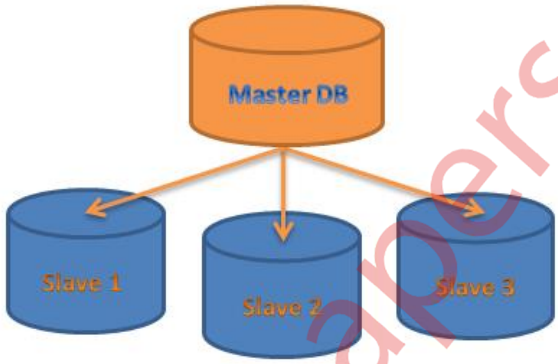
2. **explain() Function:**

- The explain() function in MongoDB is a powerful tool used for query optimization and performance analysis.
- It provides detailed information about how a specific query is executed, including the query plan, index usage, execution time, and more.
- This information can help you optimize your queries, and improve the overall performance of your MongoDB database.
- **const explainResult = db.users.find({ age: { $gte: 25 } }).explain();**
  printjson(explainResult);
  In this example, the explain() function is used to analyze the query execution for a find() operation on the "users" collection. The query selects all documents where the age is greater than or equal to 25.

e. Explain Master Slave Replication with a neat diagram. **(5)**

Ans :

- Master-slave replication enables data from one database server (the master) to be replicated to one or more other database servers (the slaves).
- The master logs the updates, which then ripple through to the slaves.
- The slave outputs a message stating that it has received the update successfully, thus allowing the sending of subsequent updates.
- Master-slave replication can be either synchronous or asynchronous.
- The difference is simply the timing of propagation of changes. If the changes are made to the master and slave at the same time, it is synchronous. If changes are queued up and written later, it is asynchronous.



- In this diagram, there is a single master database and three slave databases. The master database receives write operations and updates from clients or applications. It stores the authoritative version of the data.
- The replication process ensures that changes made to the master database are propagated to the slave databases. This is usually achieved by sending replication logs or data updates from the master to the slaves. The slaves then apply these updates to their own copies of the database, keeping them synchronized with the master.
- The slaves can handle read operations, which means that clients or applications can query the slave databases for read-only operations without impacting the master. This helps distribute the read workload and improve performance.

f. Explain the components of a sharded cluster. **(5)**

Ans :

- A sharded cluster is a mechanism for horizontally scaling databases, allowing them to handle large amounts of data and high traffic loads.
- It achieves this by partitioning data across multiple machines called shards.
- Each shard contains a subset of the data, and together they form a unified, scalable database.

- The components of a sharded cluster typically include the following:
  1. **Shards:** These are individual database instances that store a portion of the overall data set. Each shard operates as a separate database server, managing its subset of data and executing queries independently.
  2. **Shard Key:** The shard key is a unique identifier or attribute that determines how data is distributed among the shards. It helps ensure that related data is stored on the same shard, allowing for efficient query execution and data retrieval.
  3. **Query Router:** The query router, also known as the shard router, acts as an intermediary between the application and the sharded cluster. It receives incoming queries from the application and determines the appropriate shard(s) to process the query based on the shard key. The query router ensures that queries are directed to the relevant shards and merges the results if necessary.
  4. **Config Servers:** Config servers store metadata about the sharded cluster, such as the shard key ranges, mappings of chunks to shards, and cluster configurations. They serve as the control plane for the cluster, managing the distribution of data across shards and handling shard metadata updates.
  5. **Balancer:** The balancer is responsible for ensuring an even distribution of data across the shards in the cluster. It periodically monitors the shard utilization and moves data between shards as needed to balance the workload and prevent hotspots.

---

3. Attempt any three of the following:

a. Write a short note on WiredTiger Storage Engine. **(5)**

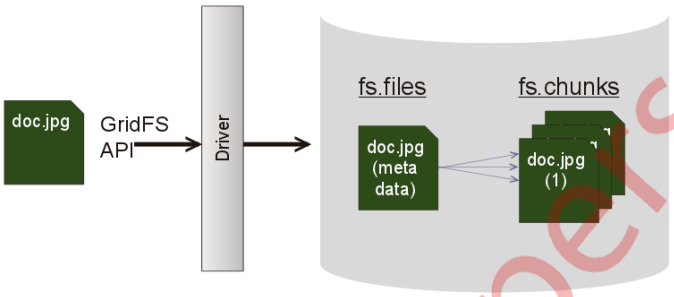
Ans :

- WiredTiger is a storage engine introduced in version 3.0 of the technology. A storage engine is responsible for managing the storage and retrieval of data in a database system.
- WiredTiger uses document-level locking. This means that when multiple clients access the same collection (a group of related documents), the lock is acquired at the document level. This allows for better concurrency, as multiple clients can access different documents simultaneously without blocking each other.
- With WiredTiger's document-level locking, multiple clients can access the same collection concurrently. Since the lock is acquired for each specific document, different clients can work on different documents within the same collection simultaneously.
- WiredTiger offers data compression capabilities using two methods: **Snappy compression** and **Zlib.**

- Data compression reduces the storage space required for storing data, improves I/O performance, and reduces network transfer time when exchanging data.
- WiredTiger can take advantage of multithreading, allowing it to utilize multiple cores in a CPU.
- This enables better performance by distributing workload across multiple threads and leveraging the processing power of modern multi-core processors.

b. Explain the concept of GridFS – The MongoDB File System. (5)

Ans :



- GridFS is a file system in MongoDB designed for storing and retrieving large files such as images, audio files, video files, and more.
- It allows you to store files in MongoDB by dividing them into smaller chunks and storing each chunk as a separate document.
- In GridFS, there are two collections: **fs.files** and **fs.chunks.**
- The fs.files collection stores the metadata of the file, while the fs.chunks collection stores the actual chunks of the file's data.
- The fs.files collection acts as a parent document and stores information such as:
    1. **_id:** The unique identifier for the file.
    2. **filename:** The name of the file.
    3. **length:** The total size of the file.
    4. **chunkSize:** The size of each chunk.
    5. **uploadDate:** The date and time the file was uploaded.
- The fs.chunks collection stores the chunks of data that make up the file. Each chunk is identified by its unique _id field and is associated with a specific files_id, which links it back to the fs.files collection.
- To store an mp3 file using GridFS, you would use the put command. The file would be divided into chunks, and each chunk would be stored as a separate document in the fs.chunks collection.
- The metadata of the file would be stored in a document in the fs.files collection.

c. What is sharding? List and Explain the sharding limitations. (5)

Ans:

- Sharding is a technique used in database management systems and distributed systems to horizontally partition data across multiple servers or nodes.
- It involves breaking down a large database or dataset into smaller, more manageable pieces called shards.
- Each shard contains a subset of the data, and multiple shards are distributed across different nodes or servers in a network.
- The primary goal of sharding is to improve performance and scalability by allowing data to be distributed and processed in parallel across multiple machines.

Limitations associated with sharding:

- **Complexity:** Sharding introduces additional complexity to the system architecture and data management. It requires a sophisticated strategy to determine how to partition the data, distribute the shards across nodes, and handle data consistency and synchronization.
- **Data Integrity:** Maintaining data integrity can be challenging in a sharded environment.
- **Joins and Relationships:** Sharding can cause difficulties when dealing with queries that involve joining data from multiple shards. Performing joins across shards requires coordination and additional processing, which can impact performance.
- **Shard Key Selection:** Choosing an appropriate shard key is crucial for efficient sharding. The shard key determines how data is distributed across shards. Poorly chosen shard keys may result in data imbalances, leading to performance issues or overloaded shards.
- **Shard Failure:** The failure of a single shard can impact the availability and performance of the overall system. Implementing fault tolerance mechanisms and ensuring high availability becomes essential but adds complexity to the system.
- **Data Migration:** When the data distribution strategy or the number of shards needs to be modified, migrating data between shards can be a time-consuming and resource-intensive process, impacting system availability and performance.

d. Explain MongoDB limitations from security perspective. Also give an overview about Read and Write limitations. (5)

Ans :

Security Limitations:

1) **Authentication and Authorization:** MongoDB provides authentication mechanisms to control access to the database, but it lacks built-in fine-grained access control at the document level.
2) **Encryption:** MongoDB supports data encryption at rest and in transit, but encryption is not enabled by default. Administrators need to configure and manage encryption settings manually to ensure data confidentiality.
3) **Auditing and Compliance:** MongoDB lacks comprehensive auditing features out of the box. While you can implement custom auditing mechanisms, it requires additional development and maintenance efforts to meet specific compliance requirement.
4) **Default Configuration:** MongoDB's default configuration is designed for ease of use and development rather than security.

Read Limitations:

a. Data Consistency: MongoDB, by default, provides eventual consistency, meaning that changes made to data may not be immediately reflected across all replicas.

b. Lack of Joins: MongoDB does not support traditional table joins like relational databases. While it provides a flexible document model, performing complex joins across multiple collections may require additional application logic.

c. Limited Transactions: MongoDB introduced multi-document ACID transactions in recent versions, but transactions are not supported across multiple shards in a sharded cluster.

Write Limitations:

a. Single-Point of Failure: In a single replica set configuration, MongoDB operates with a single primary node that handles write operations.

b. Write Performance: MongoDB ensures durability by syncing data to disk after each write operation. This process can impact write performance, especially when handling a high volume of write-intensive workloads.

c. Scalability Challenges: MongoDB's write scalability can become a challenge when dealing with highly concurrent write operations.

e. Write a short note on Deployment in MongoDB. **(5)**

Ans :

- Deployment refers to the process of setting up and configuring a MongoDB database system to make it accessible and operational for applications and users.
- It involves deploying MongoDB across various servers or nodes, configuring replication and sharding for scalability and high availability, and managing the overall database infrastructure.
- Aspects of deployment in MongoDB:
  1) **Server Configuration:** The deployment process begins with configuring the MongoDB servers. These configurations ensure that the servers are appropriately tuned for performance and meet the specific requirements of the deployment.
  2) **Replica Sets:** MongoDB supports high availability through replica sets. A replica set consists of multiple MongoDB instances, where one instance acts as the primary and others function as secondary replicas.
  3) **Sharding:** Sharding is a technique used to horizontally scale MongoDB across multiple servers or shards. In sharded deployments, MongoDB divides data into smaller chunks called shards and distributes them across different servers.
  4) **Security and Access Control:** Deployment in MongoDB also involves implementing security measures to protect the data and control access. This includes enabling authentication to require users to provide valid credentials before accessing the database.
  5) **Monitoring and Management:** MongoDB provides various tools and features for monitoring and managing deployments. These tools help track the performance of the database system, identify bottlenecks, and optimize resource utilization.

---

f. What are the tips need to be considered when coding with the MongoDB database. **(5)**

Ans : When coding with the MongoDB database, there are several tips and best practices to consider to ensure efficient and effective development.

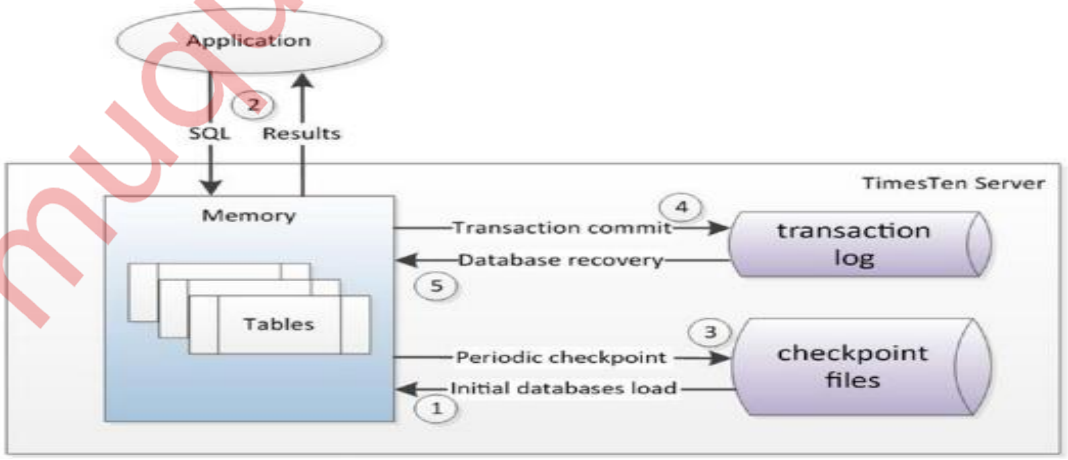
1. **Define a Clear Data Model:** Before starting the development process, carefully design and define your data model. This includes identifying the collections, fields, relationships, and indexes you will need.
2. **Normalize or Denormalize Data:** MongoDB provides flexibility in data modeling, allowing you to choose between normalization and denormalization based on your application's requirements.

3. **Optimize Queries with Indexes:** Indexes play a crucial role in query performance. Identify the queries that are frequently executed and create appropriate indexes on the fields used in those queries.
4. **Use Bulk Operations:** MongoDB offers bulk write operations, such as insertMany, updateMany, and deleteMany, which allow you to perform multiple operations in a single request. Leveraging bulk operations can significantly improve the performance of your application, especially when dealing with large datasets.
5. **Utilize Projection and Aggregation:** When querying data, utilize projection to retrieve only the necessary fields instead of fetching the entire document. This helps reduce network overhead and improves query performance.
6. **Handle Error Conditions:** Proper error handling is crucial in any application. When coding with MongoDB, make sure to handle potential error conditions, such as network failures, connection timeouts, and duplicate key errors.
7. **Secure Your Database:** MongoDB offers various security features, including authentication, authorization, and encryption. Ensure that you properly secure your MongoDB deployment by enabling authentication, creating strong user credentials, and configuring appropriate access controls.
8. **Monitor and Optimize Performance:** Continuously monitor the performance of your MongoDB deployment using tools like MongoDB Compass, mongostat, and MongoDB's Performance Advisor. Analyze slow queries, identify bottlenecks, and optimize your database configuration accordingly.
9. **MongoDB Documentation and Community:** MongoDB has comprehensive documentation, including tutorials, guides, and API references. Familiarize yourself with the documentation and utilize it as a valuable resource during development.

4. Attempt any three of the following:

a. Explain about TimesTen In-Memory Database with a neat diagram. (5)

Ans :



- TimesTen In-Memory Database (TimesTen) is a high-performance, memory-optimized relational database management system (RDBMS) designed to provide real-time, in-memory data management for demanding applications.
- It is specifically built to cater to industries that require extremely low latency and high throughput, such as telecommunications, finance, and real-time analytics.
- At its core, TimesTen operates by storing the entire database in the main memory (RAM) of the server, which allows for faster data access and manipulation compared to traditional disk-based databases.
- **Architecture of TimesTen :**
    1. **Application:** Represents the application or software that interacts with the TimesTen database using supported APIs like ODBC (Open Database Connectivity), JDBC (Java Database Connectivity), or .NET.
    2. **TimesTen In-Memory DBMS:** This layer comprises the TimesTen database management system, responsible for storing, managing, and processing the database in-memory. It handles tasks like query execution, transaction management, and data caching.
    3. **TimesTen Instances**: The TimesTen database can be partitioned into multiple instances, where each instance resides in the main memory of a server. This allows for parallel processing and increased scalability. In the diagram, two instances (Instance 1 and Instance 2) are shown.
    4. **Cache Replication:** TimesTen supports cache replication to ensure high availability and fault tolerance. It replicates data changes from one instance to another, providing redundancy and failover capabilities.

b. Explain about Solid State Disk. **(5)**

Ans :

- Solid State Disk (SSD) is a type of data storage device that uses solid-state memory to store and retrieve data.
- Unlike traditional hard disk drives (HDDs), which use mechanical components like spinning disks and read/write heads, SSDs are built with non-volatile flash memory chips, similar to those found in USB flash drives or memory cards.
- The main advantage of SSDs over HDDs is their speed and performance.
- SSDs don't rely on moving parts, they can access data much faster and with lower latency.
- Features of SSD :
    I. **Faster Read/Write Speeds:** SSDs offer significantly faster read and write speeds than traditional hard disk drives (HDDs), which use spinning disks and mechanical read/write heads. This translates into faster boot times, application loading times, and overall system performance.

II. **Lower Power Consumption:** SSDs consume less power than HDDs, making them ideal for use in laptops, tablets, and other mobile devices that rely on battery power.

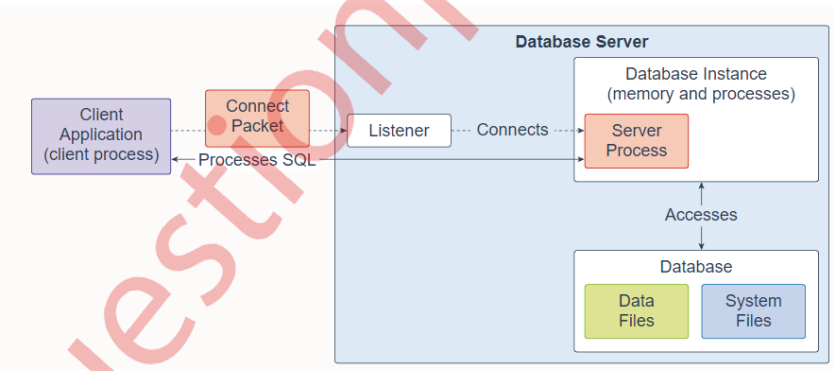
III. **Higher Reliability:** Because SSDs have no moving parts, they are generally more reliable than HDDs, which are subject to mechanical wear and tear. SSDs also have a lower rate of data loss due to read/write errors.

IV. **Higher Cost per GB:** SSDs are typically more expensive than HDDs on a cost-per-GB basis, although prices have come down in recent years. This makes SSDs less practical for use in systems that require large amounts of storage.

c. Discuss the Oracle 12c In-Memory Database architecture with a neat diagram. (5)

Ans :

- Oracle Database In-Memory is a feature introduced in Oracle Database 12c that allows for efficient storage and retrieval of data in memory to accelerate performance.
- It is designed to take advantage of the increasing availability of large memory configurations in modern servers.
- The architecture of the Oracle 12c In-Memory Database involves the following components:



i. **In-Memory Column Store:** This is the main component of the In-Memory Database architecture. The column store uses a compression algorithm optimized for in-memory storage to minimize memory usage. Data stored in the column store is organized into In-Memory Compression Units (IMCUs).

ii. **In-Memory Compression Units (IMCUs):** IMCUs are the smallest units of data stored in the column store. Each IMCU corresponds to a subset of columns from one or more database tables or partitions. IMCUs are compressed and stored in memory, allowing for efficient retrieval and processing of columnar data.

iii. **In-Memory Aggregation:** The In-Memory Database supports various aggregation capabilities that accelerate queries by performing calculations directly on the compressed columnar data. This allows for faster response times for analytics and reporting queries.

iv. **In-Memory Scan:** When a SQL query requests data that is present in the In-Memory Column Store, the In-Memory Scan feature retrieves the required data from memory, bypassing the need to access disk-based data.

v. **Automatic Data Population:** Oracle's In-Memory Database is designed to automatically determine which database objects and columns to populate in memory based on usage patterns

vi. **In-Memory Priority:** The In-Memory Database allows administrators to prioritize certain database objects or columns for population in memory.

vii. **In-Memory Query Optimizations:** The query optimizer in Oracle Database 12c is enhanced to take advantage of the In-Memory Database.

---

d. What is jQuery? Explain jQuery element selector, id selector and class selector with example. (5)

Ans :

- jQuery is a popular JavaScript library that simplifies the process of interacting with HTML documents, handling events, animating elements, and making Ajax requests.
- It provides a concise and powerful set of tools that help developers write code more efficiently and effectively.
- Types of selector :
    1. **jQuery Element Selector:**
        - The jQuery element selector allows you to select and manipulate HTML elements on a web page.
        - It uses CSS-like syntax to target elements based on their tag names, attributes, or relationship to other elements.
        - The basic syntax for the element selector is : **$('element')**
    2. **jQuery ID Selector:**
        - The jQuery ID selector allows you to select and manipulate HTML elements based on their unique identifier, known as the "id" attribute.
        - The ID selector uses the "#" symbol followed by the ID value. The syntax for the ID selector is : **$('#idValue')**

3. **jQuery Class Selector:**
    - The jQuery class selector allows you to select and manipulate HTML elements based on their class attribute.
    - The class selector uses the "." symbol followed by the class name.
    - The syntax for the class selector is : **$('.className')**

e. What is an Event ? Explain with syntax fadeIn() and fadeOut() jQuery methods. **(5)**

Ans :

- All the different visitors' actions that a web page can respond to are called events.
- An event represents the precise moment when something happens.

1. **fadeIn():**
    - This method is used to gradually fade in selected elements, making them visible by increasing their opacity over a specified duration.
    - **Syntax : $(selector).fadeIn(speed, easing, callback);**
    - **selector:** Specifies the element(s) to be faded in.
    - **speed (optional):** Defines the duration of the fade-in effect. It can be specified in milliseconds (e.g., 1000 for one second) or using predefined values like "slow" or "fast".
    - **easing (optional):** Determines the easing function to be used for the animation. It can be a predefined easing function or a custom one.
    - **callback (optional):** Specifies a function to be executed once the fade-in animation is complete.
2. **fadeOut():**
    - This method is used to gradually fade out selected elements, making them invisible by decreasing their opacity over a specified duration.
    - **Syntax: $(selector).fadeOut(speed, easing, callback);**
    - **selector:** Specifies the element(s) to be faded out.
    - **speed (optional):** Defines the duration of the fade-out effect. It can be specified in milliseconds or using predefined values like "slow" or "fast".
    - **easing (optional):** Determines the easing function to be used for the animation.
    - **callback (optional):** Specifies a function to be executed once the fade-out animation is complete.

f. Explain the features supported by jQuery. **(5)**

Ans :

1. **DOM manipulation :**

The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called Sizzle.

2. **Event handling :**

The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on alink, without the need to clutter the HTML code itself with event handlers.

3. **AJAX Support :**

The jQuery helps you a lot to develop a responsive and feature rich site using AJAX technology.

4. **Animations :**

The jQuery comes with plenty of built-in animation effects which you can use in your websites.

5. **Lightweight :**

The jQuery is very lightweight library - about 19KB in size (Minified and zipped).

6. **Cross Browser Support :**

The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+

7. **Latest Technology :**

The jQuery supports CSS3 selectors and basic XPath syntax.

---

5. Attempt any three of the following:

a. What is the use of Stringify method? Explain the Syntax. **(5)**

Ans :

1. A common use of JSON is to exchange data to/from a web server. When sending data to a web server, the data has to be a string.

2. We can Convert a JavaScript object into a string with **JSON.stringify(). Stringify** a JavaScript Object. Imagine we have this object in JavaScript:

var obj = { name: "John", age: 30, city: "New York" };

Use the JavaScript function **JSON.stringify()** to convert it into a string.

var myJSON = JSON.stringify(obj);

3. Syntax of the JSON stringify Method

JSON.stringify(value[, replacer [, space]]);

4. The value parameter of the stringify method is the only required parameter of the three outlined by the signature.The argument supplied to the method represents the JavaScript value intended to be serialized.This can be that of any object, primitive, or even a composite of the two.

5. The optional replacer parameter is either a function that alters the way objects and arrays are stringified or an array of strings and numbers that acts as a white list for selecting the object properties that will be stringified.

6. The third parameter, space, is also optional and allows you to specify the amount of padding that separates each value from one another within the produced JSON text.

7. Code:

```
<html>
<head>
<title>JSON programs </title>
</head>
<body>
<script type="text/javascript">
var data={
"Bname":"JSON",
"Publisher": "TataMcgraw",
"author": "Smith",
"price":250,
"ISBN":"1256897912345"
};
document.writeln(JSON.stringify(data));
document.writeln(JSON.stringify(data,["Bname","author","price"]));
document.write(JSON.stringify(data,["Bname","author","price"],5));
</script></body></html>
```

b. Explain the six members of the web storage Interface. **(5)**

Ans :

setItem(key, value): This method allows you to store a key-value pair in the web storage. The key is a string that acts as an identifier for the data, and the value can be any data that can be represented as a string.

getItem(key): This method retrieves the value associated with the specified key. It takes the key as a parameter and returns the corresponding value if it exists; otherwise, it returns null.

removeItem(key): With this method, you can remove a specific key-value pair from the web storage. When called with a key, it will delete the corresponding data.

clear(): This method clears all data stored in the web storage, regardless of the key-value pairs. It effectively resets the storage to an empty state.

key(index): This method allows you to retrieve the key at a specific index in the storage. The index is a numeric value indicating the position of the key-value pair in the storage. This method is helpful when you want to iterate through all the stored keys.

length: The length property returns the number of key-value pairs currently stored in the web storage. It is useful for determining the size of the storage and checking if it contains any data.

---

c. Explain the structure of HyperText Transfer Protocol (HTTP) –Request. **(5)**

Ans : HTTP (HyperText Transfer Protocol) is the foundation of data communication on the World Wide Web. It defines how clients (such as web browsers) request resources from servers and how servers respond to those requests. An HTTP request consists of several components, which are sent by the client to the server when the client wants to retrieve a resource, like a web page or an image.

- **The basic structure of an HTTP request is as follows:**

1. **Request Line:**
   - This is the first line of the HTTP request and contains three important components:
   - **HTTP Method**: This indicates the action the client wants to perform on the resource. Common HTTP methods include:
   - **GET:** Used to request data from the server (typically used to retrieve web pages).
   - **POST:** Used to send data to the server (often used for form submissions or data uploads).
   - **PUT:** Used to update a resource on the server.
   - **DELETE:** Used to delete a resource from the server, etc.

2. **Request Headers:**
   - After the request line, there can be additional headers that provide more information about the request or the client making the request. Headers are key-value pairs that are separated by a colon (:).
   - Some common request headers include:
   - **Host:** Specifies the domain name of the server that the client is requesting the resource from.
   - **User-Agent:** Contains information about the client application or web browser making the request.
   - **Accept**: Indicates the media types that the client can handle in the response.
   - **Content-Type:** Specifies the format of the data being sent in the request (applicable to certain HTTP methods like POST and PUT).
   - **Authorization**: Used to send authentication credentials to access protected resources, etc.
3. **Request Body (Optional):**
   - Some HTTP methods, like POST or PUT, may include a request body that contains data to be sent to the server.
   - This is often used for submitting form data or uploading files.
   - After composing the HTTP request with the appropriate method, headers, and optional body, the client sends the request to the server using the underlying TCP/IP (or other transport) connection.
   - The server then processes the request and responds with an HTTP response, containing the requested resource or an error message.

d. Explain the JSON Grammer. **(5)**

Ans :

- JSON, in a nutshell, is a textual representation defined by a small set of governing rules in which data is structured.
- The JSON specification states that data can be structured in either of the two following compositions:

  1. A collection of name/value pairs

  2. An ordered list of values

- JSON stem from the ECMAScript standardization, the implementations of the two structures are represented in the forms of the object and array.
- a collection begins with the use of the opening brace ({), and ends with the use of the closing brace (}). The content of the collection can be composed of any of the following possible three designated paths:

  1. The top path illustrates that the collection can remain devoid of any string/value pairs.

2. The middle path illustrates that our collection can be that of a single string/value pair.

3. The bottom path illustrates that after a single string/value pair is supplied, the collection needn't end but, rather, allow for any number of string/value pairs, before reaching the end.

- Each string/value pair possessed by the collection must be delimited or separated from one another by way of a comma (,).

e. Give an overview of JavaScript Object Notation(JSON). Also Explain about JSON Tokens. (5)

Ans:

Overview of JSON:

- JavaScript Object Notation, commonly known as JSON, is a lightweight data interchange format. It is easy for humans to read and write and straightforward for machines to parse and generate. JSON is often used to transmit data between a server and a web application, serving as an alternative to XML in many cases.
- JSON is based on two fundamental data structures:

1. **Objects:** An object is an unordered collection of key-value pairs, where keys are strings, and values can be any valid JSON data type (string, number, boolean, null, array, or another object). Objects are enclosed within curly braces {}.
   Example :
   {
   "name": "John Doe",
   "age": 30,
   "isStudent": true
   }
2. **Arrays:** An array is an ordered list of values, where each value can be any valid JSON data type. Arrays are enclosed within square brackets []
   Example : ["apple", "orange", "banana"]

JSON Tokens:

JSON is composed of various elements called tokens. Each token represents a specific element within the JSON structure. Here are the different JSON tokens:

- Object Token: The object token { indicates the start of a JSON object.
- End Object Token: The end object token } indicates the end of a JSON object.
- Array Token: The array token [indicates the start of a JSON array.
- End Array Token: The end array token] indicates the end of a JSON array.

- Key Token: The key token is a string that represents the name of a property within a JSON object. It is always followed by a colon : to separate it from its corresponding value.
- Value Token: The value token represents the data associated with a key within a JSON object. It can be a string, number, boolean, null, array, or another object.
- String Token: The string token represents a sequence of characters enclosed within double quotes ". Strings are used for representing textual data.
- Number Token: The number token represents a numeric value, which can be an integer or a floating-point number.
- Boolean Token: The boolean token represents either true or false.
- Null Token: The null token represents an empty value or absence of data.

f. Explain about JSON parsing with Syntax. **(5)**

Ans :

- Parsing is the process of analyzing a string of symbols, either in natural language or in computer languages, according to the rules of a formal grammar.
- As the grammar of JSON is a subset of JavaScript, the analysis of its tokens by the parser occurs indifferently from how the Engine parses source code. Because of this, the data produced from the analysis of the JSON grammar will be that of objects, arrays, strings, and numbers.
- Additionally, the three literals true, false, and null are produced as well.
- JSON.parse
- In a nutshell, JSON.parse converts serialized JSON into usable JavaScript values.
- Syntax of the JSON.parse() Method :
- JSON.parse(text [, reviver]);
- JSON.parse can accept two parameters, text and reviver.
- The name of the parameter text is indicative of the value it expects to receive.
- The parameter reviver is used similarly to the replacer parameter of stringify, in that it offers the ability for custom logic to be supplied for necessary parsing that would otherwise not be possible by default.
- The parameter text implies the JavaScript value, which should be supplied.
- This is a rather important aspect, because any invalid argument will automatically result in a parse error.
- Eg : Invalid JSON Grammar Throws a Syntax Error :

var str = JSON.parse("abc123"); //SyntaxError: JSON.parse: unexpectedcharacter

- Throws an error because it was provided a string literal and not serialized JSON.

- Therefore, "abc123" must be escaped and wrapped with an additional set of quotation marks.

• Valid JSON Grammer Is Successfully Parsed

```
var str = JSON.parse( "\"abc123\"" ); //valid JSON string value

console.log(str) //abc123;

console.log(typeof str) //string;
```