

TYBSC IT

Next Generation Technology (NGT)

SEM 5 (NOV-2018)

Q.P.Code:57843

1. Attempt any three of the following:

a. What is Big Data? What are the different sources of Big Data? (5)

Ans: Big data is a term used to describe data that has massive volume, comes in a variety of structures, and is generated at high velocity. This kind of data poses challenges to the traditional RDBMS systems used for storing and processing data. Big data is paving way for newer approaches of processing and storing data.

The different sources of Big Data are:

- Enterprises, which are collecting data with more granularities now, attaching more details with every transaction in order to understand consumer behaviour.
- Increase in multimedia usage across industries such as health care, product companies, etc.
- Increased popularity of social media sites such as Facebook, Twitter, etc.
- Rapid adoption of smartphones, which enable users to actively use social media sites and other Internet applications.
- Increased usage of sensors and devices in the day-to-day world, which are connected by networks to computing resources.

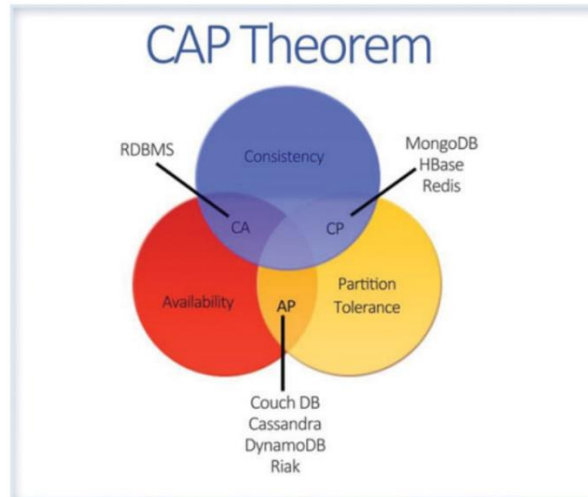
b. Compare ACID Vs BASE. (5)

Ans. The RDBMS systems are based on the concept of ACID transactions. ACID stands for Atomic, Consistent, Isolated, and Durable, where

1. Atomic implies either all changes of a transaction are applied completely or not applied at all.
 2. Consistent means that the data is in a consistent state after the transaction is applied. This means after a transaction is committed, the queries fetching a particular data will see the same result.
 3. Isolated means the transactions that are applied to the same set of data are independent of each other. Thus, one transaction will not interfere with another transaction.
 4. Durable means the changes are permanent in the system and will not be lost in case of any failures. Whereas ACID defines the key characteristics required for reliable transaction processing, the NoSQL world requires different characteristics to enable flexibility and scalability.
- These opposing characteristics are cleverly captured in the acronym BASE. BASE can be explained as
 1. Basically Available means the system will be available in terms of the CAP theorem.
 2. Soft state indicates that even if no input is provided to the system, the state will change over time. This is in accordance to eventual consistency.
 3. Eventual consistency means the system will attain consistency in the long run, provided no input is sent to the system during that time.

c. With the help of a neat diagram explain, the CAP theorem. (5)

Ans.



- CAP theorem also called as brewers theorem.
- Eric brewer outline the cap theorem in 2000
- The theorem states that when designing an application in a distributed environment namely consistent, availability and partition tolerance.
 - a. **Consistency** means that the 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.
 - b. **Availability** means that the system is always available.
 - c. **Partition Tolerance** means that the system will continue to function even if it is partitioned into groups of servers that are not able to communicate with one another.

Eric Brewer coined the BASE acronym . BASE can be explained as

- Basically Available means the system will be available in terms of the CAP theorem.
- Soft state indicates that even if no input is provided to the system, the state will change over time. This is in accordance to eventual consistency.
- Eventual consistency means the system will attain consistency in the long run, provided no input is sent to the system during that time.

d. What are the advantages and disadvantages of NoSQL databases? (5)

Ans.

➤ Advantages of NoSQL :

• **High scalability :**

This scaling up approach fails when the transaction rates and fast response requirements increase. In contrast to this, the new generation of NoSQL databases is designed to scale out (i.e. to expand horizontally using low-end commodity servers).

• **Manageability and administration :**

NoSQL databases are designed to mostly work with automated repairs, distributed data, and simpler data models, leading to low manageability and administration.

- **Low cost :**

NoSQL databases are typically designed to work with a cluster of cheap commodity servers, enabling the users to store and process more data at a low cost.

- **Flexible data models :**

NoSQL databases have a very flexible data model, enabling them to work with any type of data; they don't comply with the rigid RDBMS data models. As a result, any application changes that involve updating the database schema can be easily implemented.

➤ Disadvantages of NoSQL :

- **Maturity :**

Most NoSQL databases are pre-production versions with key features that are still to be implemented. Thus, when deciding on a NoSQL database, you should analyze the product properly to ensure the features are fully implemented and not still on the To-do list.

- **Support :**

Support is one limitation that you need to consider. Most NoSQL databases are from start-ups which were open sourced. As a result, support is very minimal as compared to the enterprise software companies and may not have global reach or support resources.

- **Limited Query Capabilities :**

Since NoSQL databases are generally developed to meet the scaling requirement of the web-scale applications, they provide limited querying capabilities. A simple querying requirement may involve significant programming expertise.

- **Administration :**

Although NoSQL is designed to provide a no-admin solution, it still requires skill and effort for installing and maintaining the solution.

- **Expertise :**

Since NoSQL is an evolving area, expertise on the technology is limited in the developer and administrator community.

e. What are the different categories of NoSQL database? Explain each with an example. (5)

Ans.

- The NoSQL databases are categorized on the basis of how the data is stored.
- NoSQL mostly follows a horizontal structure because of the need to provide curated information from large volumes, generally in near real-time.
- They are optimized for insert and retrieve operations on a large scale with built-in capabilities for replication and clustering.

Category	Brief Description	For E.g.
Document-based	Data is stored in form of documents. For instance, {Name="Test User", Address="Address1", Age:8}	MongoDB
XML database	XML is used for storing data.	MarkLogic
Graph databases	Data is stored as node collections. The nodes are connected via edges. A node is comparable to an object in a programming language.	GraphDB
Key-value store	Stores data as key-value pairs.	Cassandra, Redis, memcached

➤ Document Databases

- A document database stores data in JSON, BSON , or XML documents (not Word documents or Google docs, of course).
- In a document database, documents can be nested. Particular elements can be indexed for faster querying.
- 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 the data in an application.
- The most widely adopted document databases are usually implemented with a scale-out architecture, providing a clear path to scalability of both data volumes and traffic.

➤ Key-Value Stores

- The simplest type of NoSQL database is a key-value store .
- Every data element in the database is stored as a key value pair consisting of an attribute name (or "key") and a value.
- In a sense, a key-value store is like a relational database with only two columns: the key or attribute name (such as state) and the value (such as Alaska).

➤ Graph Databases

- A graph database focuses on the relationship between data elements.
- Each element is stored as a node (such as a person in a social media graph).
- The connections between elements are called links or relationships.
- In a graph database, connections are first-class elements of the database, stored directly.

f. What are the different challenges Big Data poses?

(5)

Ans. Big data poses challenges which are as follows:

- 1) Policies and Procedure.
- 2) Access to Data.
- 3) Technology and Techniques.

1) Policies and Procedure:

- As more and more data is gathered, digitalized, and moved around the globe, the policy and compliance issues become increasingly important.

- Data privacy, security, intellectual property, and protection are of immense important to organization.
- Compliance with various statutory and legal requirements poses a challenge in data handling.
- Issues around ownership and liabilities around data are important legal aspects that need to deal in case of big data.

2) Access to Data:

- Accessing data for consumption is a challenge for big data projects.
- Some of the data may be available to third parties, and gaining access can be legal contractual challenge.
- Data about a product or service is available on Facebook, Twitter feeds , reviews and blogs so how does the product owner access this data from various sources owned by various providers.
- Contractual clause and economic incentives for accessing big data need to be tied in to enable the availability od data by the consumer.

3) Technology and Techniques:

- Flexible data models that easily adopt to changing requirements.
- Elastic scalability for growing in lockstep with system demand.
- High performance in terms of throughput and latency.
- Parallel processing is also the challenge for changing requirements.

2. Attempt any three of the following:

a. Consider a Collection users containing the following fields

```
{
  id: ObjectID(),
  FName: "First Name",
  LName: "Last Name",
  Age: 30, Gender: "M",
  Country: "Country"
}
```

Where Gender value can be either "M" or "F" or "Other". Country can be either "UK" or "India" or "USA".

Based on above information write the MongoDB query for the following.

- Update the country to UK for all female users.
- Add the new field company to all the documents.
- Delete all the documents where Gender = 'M'.
- Find out a count of female users who stay in either India or USA.
- Display the first name and age of all female employees.

(5)

Ans.

- Update the country to UK for all female users.

```
db.users.update({"Gender":"F"}, {$set:{"Country":"UK"}})
```

- Add the new field company to all the documents.

```
db.users.update({}, {$set:{"Company":"TestComp"}}, {multi:true})
```

iii. Delete all the documents where Gender = 'M'.

```
db.users.remove({"Gender":"M"})
```

iv. Find out a count of female users who stay in either India or USA.

```
db.users.find({"Gender":"F", $or:[{"Country":"India"}, {"Country":"USA"}]}).count()
```

v. Display the first name and age of all female employees.

```
db.users.find({"Gender":"F"}, {"Name":1,"Age":1})
```

b. Write the Mongo dB command to create the following with an example:

(5)

(i) Database (ii) Collection (iii) Document

(iv) Drop Collection (v) Drop Database (vi) Index

Ans.

(i) Database :

- MongoDB use DATABASE_NAME is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

Syntax :

- Basic syntax of use DATABASE statement is as follows :

use DATABASE_NAME

Example :

- If you want to use a database with name , then use DATABASE statement would be as follows :

use mydb

(ii) Collection :

- MongoDB db.createCollection(name, options) is used to create collection.

Syntax :

- Basic syntax of createCollection() command is as follows :

db.createCollection(name, options)

- In the command, name is name of collection to be created. Options is a document and is used to specify configuration of collection.

- Basic syntax of createCollection() method without options is as follows :

- switched to db test :

use test

db.createCollection("mycollection")

(iii) Document :

- To create a document into MongoDB collection, you need to use MongoDB's **insert()** or **save()** method.

Syntax :

- The basic syntax of insert() command is as follows – **db.COLLECTION_NAME.insert(document)**

Example :

db.mycol.insert({

_id: ObjectId(7df78ad8902c),

title: 'MongoDB Overview',

description: 'MongoDB is no sql database',

```
url: 'http://www.MongoDB.com',
tags: ['mongodb', 'database', 'NoSQL'],
likes: 100
})
```

(iv) Drop Collection :

- MongoDB's **db.collection.drop()** is used to drop a collection from the database.

Syntax :

- Basic syntax of drop() command is as follows :

db.COLLECTION_NAME.drop()

Example :

db.mycollection.drop()

(v) Drop Database :

- MongoDB db.dropDatabase() command is used to drop a existing database.

Syntax :

- Basic syntax of dropDatabase() command is as follows :

db.dropDatabase()

(vi) Index :

- To create an index you need to use ensureIndex() method of MongoDB.

Syntax :

- The basic syntax of ensureIndex() method is as follows(). **db.COLLECTION_NAME.ensureIndex({KEY:1})**

Example :

- **db.mycol.ensureIndex({"title":1})**

c. Write a short note on Capped collection.

(5)

Ans.

- MongoDB has a concept of capping the collection.
 - This means it stores the documents in the collection in the inserted order.
 - As the collection reaches its limit, the documents will be removed from the collection in FIFO (first in, first out) order.
 - This means that the least recently inserted documents will be removed first.
 - This is good for use cases where the order of insertion is required to be maintained automatically.
 - Deletion of records after a fixed size is required.
- Eg : Log files -which get automatically truncated after a certain size.
- Capped collections are fixed-size circular collections that follows the insertion order to support high performance for create, read, and delete operations.
 - Capped collection are best for storing log information, cache data, or any other high volume data.
 - MongoDB itself uses capped collections for maintaining its replication logs.

Important points regarding capped collections :

- We cannot delete documents from a capped collection.

- There are no default indexes present in a capped collection, not even on `_id` field.
- While inserting a new document, MongoDB does not have to actually look for a place to accommodate new document on the disk. It can blindly insert the new document at the tail of the collection. This makes insert operations in capped collections very fast.
- Similarly, while reading documents MongoDB returns the documents in the same order as present on disk. This makes the read operation very fast.

d. List and explain the different conditional operators in MongoDB.

(5)

Ans. The different conditional operators in MongoDB are:

1. `$lt` and `$lte` :

They stand for “less than” and “less than or equal to,” respectively.

If you want to find all students who are younger than 25 (`Age < 25`), you can execute the following find with a selector:

```
db.students.find({"Age":{"$lt":25}})
```

2. `$gt` and `$gte` :

The `$gt` and `$gte` operators stand for “greater than” and “greater than or equal to,” respectively.

Let’s find out all of the students with `Age > 25`. This can be achieved by executing the following command:

```
db.students.find({"Age":{"$gt":25}})
```

3. `$in` and `$nin` :

Let’s find all students who belong to either class `C1` or `C2`.

The command for the same is :

```
db.students.find({"Class":{"$in":["C1","C2"]}})
```

The inverse of this can be returned by using `$nin`.

To find students who don’t belong to class `C1` or `C2`. The command is

```
db.students.find({"Class":{"$nin":["C1","C2"]}})
```

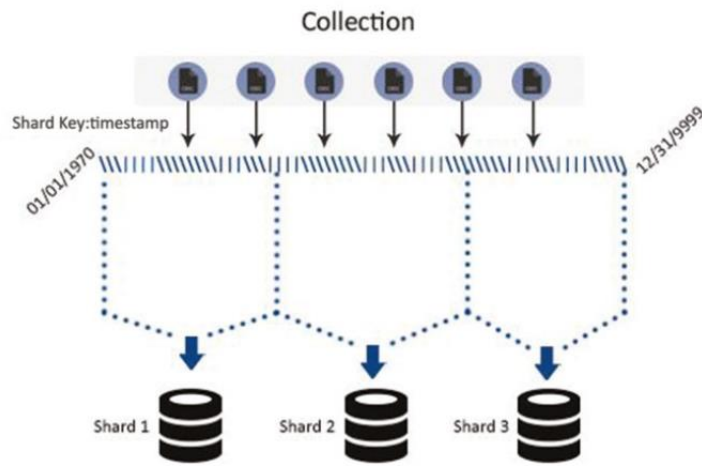
e. Explain the two ways MongoDB enables distribution of the data in Sharding.

(5)

Ans.

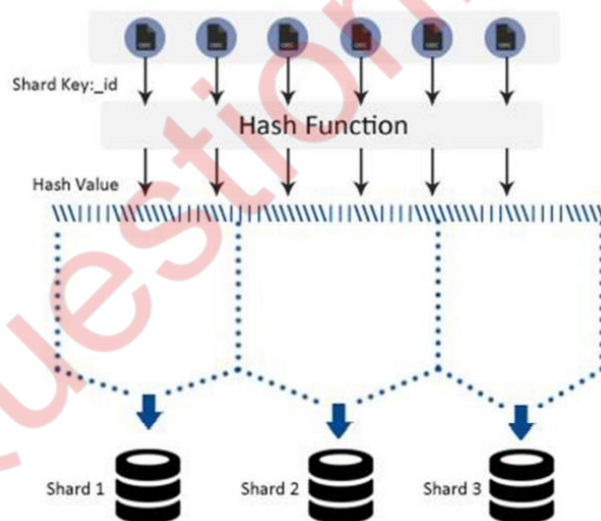
- In MongoDB, the data is sharded or distributed at the collection level.
- The collection is partitioned by the shard key.
- There are two ways MongoDB enables distribution of the data:
 - I. Range-based partitioning
 - II. Hash-based partitioning.

1. Range-based partitioning :



- In range-based partitioning , the shard key values are divided into ranges.
- The values are considered as a straight line starting from a Min value to Max value where Min is the starting period (say, 01/01/1970) and Max is the end period (say, 12/31/9999).
- Every document in the collection will have timestamp value within this range only, and it will represent some point on the line.
- Based on the number of shards available, the line will be divided into ranges, and documents will be distributed based on them.
- The documents where the values of the shard key are nearby are likely to fall on the same shard. This can significantly improve the performance of the range queries.

2. Hash-based partitioning:



- In hash-based partitioning , the data is distributed on the basis of the hash value of the shard field.
- If selected, this will lead to a more random distribution compared to range-based partitioning.
- It's unlikely that the documents with close shard key will be part of the same chunk.
- For example, for ranges based on the hash of the `_id` field, there will be a straight line of hash values, which will again be partitioned on basis of the number of shards.

f. List and explain the 3 core components in the MongoDB package.

(5)

Ans.

The core components in the MongoDB package are :

- mongod : which is the core database process .
- mongos : which is the controller and query router for sharded clusters .
- mongo : which is the interactive MongoDB shell .

Mongod:

- The primary daemon in a MongoDB system is known as mongod . This daemon handles all the data requests, manages the data format, and performs operations for background management.
- When a mongod is run without any arguments, it connects to the default data directory.
- It's important to ensure that the data directory exists and you have write permissions to the directory before the mongod process is started.
- If the directory doesn't exist or you don't have write permissions on the directory, the start of this process will fail.

Mongos:

- Mongos is used in MongoDB sharding.
- It acts as a routing service that processes queries from the application layer and determines where in the sharded cluster the requested data is located.
- mongos is the process that routes the queries to the correct server holding the data.

Mongo:

- Mongo provides an interactive JavaScript interface for the developer to test queries and operations directly on the database and for the system administrators to manage the database. This is all done via the command line.
- As a developer or administrator you need to change the database from test to your database post the first connection is made. You can do this by using .

3. Attempt any three of the following:

a. List and explain the limitations of Sharding.

(5)

Ans.

- Sharding is the mechanism of splitting data across shards.
- The following sections talk about the limitations that you need to be aware of when dealing with sharding.

• Shard Early to Avoid Any Issues

Using the shard key, the data is split into chunks, which are then automatically distributed amongst the shards. However, if sharding is implemented late, it can cause slowdowns of the servers because the splitting and migration of chunks takes time and resources. A simple solution is to monitor your MongoDB instance capacity using tools such as MongoDB Cloud Manager (flush time, lock percentages, queue lengths, and faults are good measures) and shard before reaching 80% of the estimated capacity.

- **Shard Key Can't Be Updated**

The shard key can't be updated once the document is inserted in the collection because MongoDB uses shard keys to determine to which shard the document should be routed. If you want to change the shard key of a document, the suggested solution is to remove the document and reinsert the document when the change has been made.

- **Shard Collection Limit**

The collection should be sharded before it reaches 256GB.

- **Select the Correct Shard Key**

It's very important to choose a correct shard key because once the key is chosen it's not easy to correct it.

b. What is Data Storage engine? Which is the default storage engine in MongoDB? Also compare MMAP and Wired Tiger storage engines. (5)

Ans.

1. The data storage engine is the component of the database that is responsible for managing how data is stored, both in memory and on disk.
2. MongoDB supports multiple storage engines, as different engines perform better for specific workloads.
3. MongoDB uses MMAP as its default storage engine.
4. This engine works with memory-mapped files .
5. Memory-mapped files are data files.

Difference between MMAP and Wired storage engine:

MMAP:

- i. Memory Mapping(MMAP) has been the only storage engine.
- ii. MMAP uses Collection-level locking.
- iii. if a client acquires a lock on a document to modify its content, then no other client can access the collection.
- iv. MMAP uses journaling to recover from failure.
- v. Data compression facilities is not present in MMAP storage engine.
- vi. MMAP, increasing the size of the RAM decreases the number of page faults which in turn increases the performance.

WiredTiger:

- i. But in version 3.0 and more, another storage engine, WiredTiger has been introduced which has its own benefits.
- ii. WiredTiger uses Document-Level Locking.
- iii. Multiple clients can access the same collection, since the lock is acquired for that particular document.
- iv. In WiredTiger, a consistent view of the data is given by means of checkpoints. so that in case of failure it can rollback to the previous checkpoint.
- v. In WiredTiger Storage Engine. Data compression is achieved using two methods

- a. 1.Snappy compression
 - b. 2.Zlib
- vi. Wired Tiger can make use of multithreading and hence multi core CPU's can be made use of it.

c. "With the rise of the Smartphone, it's becoming very common to query for things near a current location". Explain the different indexes used by MongoDB to support such location-based queries. (5)

Ans.

- With the rise of the smartphone, it's becoming very common to query for things near a current location.
- In order to support such location-based queries, MongoDB provides geospatial indexes.
- To create a geospatial index, a coordinate pair in the following forms must exist in the documents:

1. Either an array with two elements
2. Or an embedded document with two keys (the key names can be anything).

➤ The following are valid examples:

```
{ "userloc" : [ 0, 90 ] }  
{ "loc" : { "x" : 30, "y" : -30 } }  
{ "loc" : { "latitude" : -30, "longitude" : 180 } }  
{ "loc" : { "a1" : 0, "b1" : 1 } }.
```

➤ The following can be used to create a geospatial index on the userloc field:

```
db.userplaces.ensureIndex( { userloc : "2d" } )
```

- A geospatial index assumes that the values will range from -180 to 180 by default.
- If this needs to be changed, it can be specified along with ensureIndex as follows:

```
db.userplaces.ensureIndex({ "userloc" : "2d", { "min" : -1000, "max" : 1000 } }
```
- Any documents with values beyond the maximum and the minimum values will be rejected.

➤ You can also create compound geospatial indexes.

➤ Examples:

```
{ "loc": [0,100], "desc": "coffeeshop" }  
{ "loc": [0,1], "desc": "pizzashop" }
```

If the query of a user is to find all coffee shops near her location, the following compound index can help:

```
db.ensureIndex({ "userloc" : "2d", "desc" : 1 } )
```

- Geohaystack indexes are bucket-based geospatial indexes (also called geospatial haystack indexes).
- They are useful for queries that need to find out locations in a small area and also need to be filtered along another dimension, such as finding documents with coordinates within 10 miles.
- While defining the index, it's mandatory to specify the bucketSize parameter as it determines the haystack index granularity.
- For example,

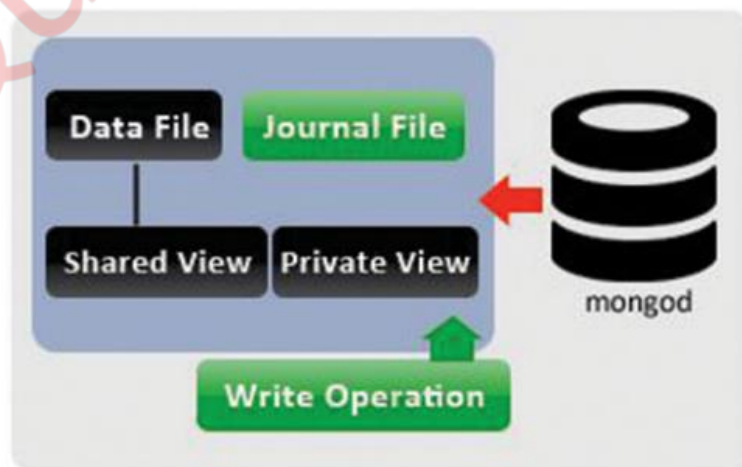
```
db.userplaces.ensureIndex({ userpos : "geoHaystack", type : 1 }, { bucketSize : 1 })
```

- You can also include an additional category in the index.
- If your use case typically searches for "nearby" locations (i.e. "restaurants within 25 miles"), a haystack index can be more efficient.
- The matches for the additional indexed field (e.g. category) can be found and counted within each bucket.

d. What is Journaling? Explain the importance of Journaling with the help of a neat diagram. (5)

Ans.

- Journalling is another file system where we keep records which are not yet updated or committed to file system's main part.
- Records are known as a journal which is another circular records or logs.
- In journaling, all the records or journals written in consecutive tracks (similar as an arrays data structure) so that seek time will be very less as compared to random access. this is the main benefit of journalling.
- If we write data to file system's main part then it will be slow and slows down overall processing.
- In this process, a write operation occurs in mongod, which then creates changes in private view.
- The first block is memory and the second block is 'my disc'.
- After a specified interval, which is called a 'journal commit interval', the private view writes those operations in journal directory (residing in the disc).
- Once the journal commit happens, mongod pushes data into shared view.
- As part of the process, it gets written to actual data directory from the shared view (as this process happens in background).
- The basic advantage is, we have a reduced cycle from 60 seconds to 200 milliseconds.
- In a scenario where an abruption occurs at any point of time or flash disc remains unavailable for last 59 seconds (keeping in mind the existing data in journal directory/write operations), then when the next time mongod starts, it basically replays all write operation logs and writes into the actual data directory.



e. Write a short note on Replication Lag.

(5)

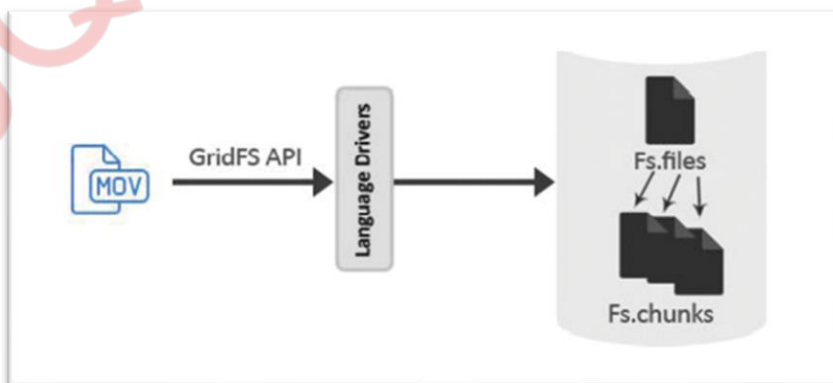
Ans.

- Replication lag is the primary administrative concern behind monitoring replica sets.
 - Replication lag for a given secondary is the difference in time when an operation is written in primary and the time when the same was replicated on the secondary.
 - The replication lag remedies itself and is transient.
 - If it remains high and continues to rise, there might be a problem with the system.
 - You might end up either shutting down the system until the problem is resolved, or it might require manual intervention for reconciling the mismatch, or you might even end up running the system with outdated data.
- The following command can be used to determine the current replication lag of the replica set:
testset:PRIMARY> rs.printSlaveReplicationInfo()
 - Further, you can use the **rs.printReplicationInfo()** command to fill in the missing piece:
testset:PRIMARY> rs.printReplicationInfo()
- MongoDB Cloud Manager can also be used to view recent and historical replication lag information.
 - Here are some tips to help reduce this time:
 - In scenarios with a heavy write load, you should have a secondary as powerful as the primary node so that it can keep up with the primary and the writes can be applied on the secondary at the same rate. Also, you should have enough network bandwidth so that the ops can be retrieved from the primary at the same rate at which they are getting created.
 - Adjust the application write concern.
 - If the secondary is used for index builds, this can be planned to be done when there are low write activities on the primary.
 - If the secondary is used for taking backups, consider taking backups without blocking.
 - Check for replication errors. Run **rs.status()** and check the **errmsg** field. Additionally, the secondary's log files can be checked for any existing error messages.

f. Explain “ GridFS – The MongoDB File System” with the help of a neat diagram.

(5)

Ans.



- **GridFS** is the MongoDB specification for storing and retrieving large files such as images, audio files, video files, etc. It is kind of a file system to store files but its data is stored within MongoDB collections. GridFS has the capability to store files even greater than its document size limit of 16MB.
 - GridFS divides a file into chunks and stores each chunk of data in a separate document, each of maximum size 255k.
 - GridFS by default uses two collections **fs.files** and **fs.chunks** to store the file's metadata and the chunks. Each chunk is identified by its unique `_id` ObjectId field.
 - The `fs.files` serves as a parent document. The **files_id** field in the `fs.chunks` document links the chunk to its parent.
- Example of `fs.files` collection –

```
{
  "filename": "test.txt",
  "chunkSize": NumberInt(261120),
  "uploadDate": ISODate("2014-04-13T11:32:33.557Z"),
  "md5": "7b762939321e146569b07f72c62cca4f",
  "length": NumberInt(646)
}
```

The document specifies the file name, chunk size, uploaded date, and length.

- Example of `fs.chunks` document –

```
{
  "files_id": ObjectId("534a75d19f54bfec8a2fe44b"),
  "n": NumberInt(0),
  "data": "Mongo Binary Data"
}
```

➤ Adding Files to GridFS

- Now, we will store an mp3 file using GridFS using the **put** command. For this, we will use the **mongofiles.exe** utility present in the bin folder of the MongoDB installation folder.
- Open your command prompt, navigate to the `mongofiles.exe` in the bin folder of MongoDB installation folder and type the following code –

```
>mongofiles.exe -d gridfs put song.mp3
```

- We can also see all the chunks present in `fs.chunks` collection related to the stored file with the following code, using the document id returned in the previous query –

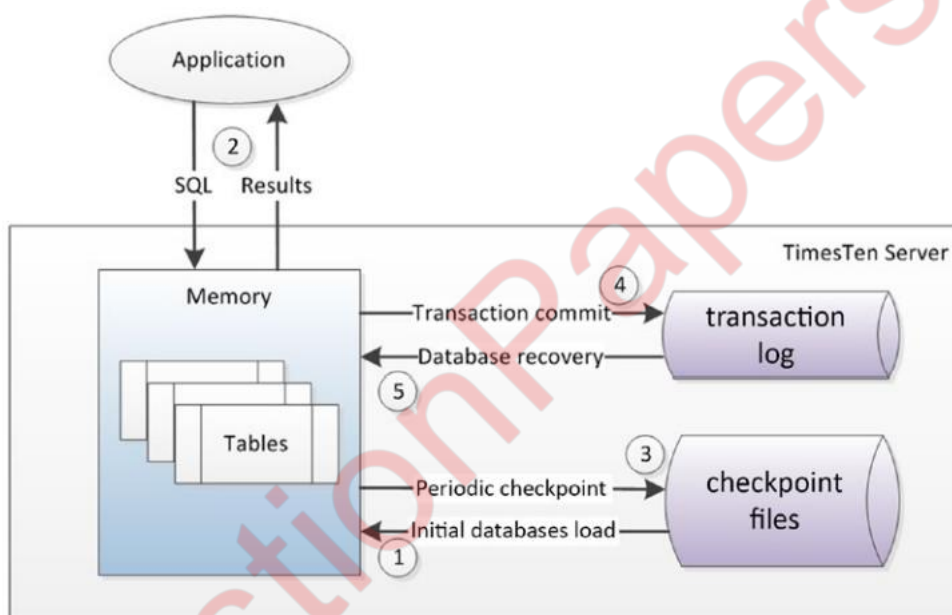
```
>db.fs.chunks.find({files_id:ObjectId('534a811bf8b4aa4d33fdf94d')})
```

4. Attempt any three of the following:

a. Explain TimesTen Architecture with the help of a neat diagram. (5)

Ans.

- TimesTen is a relatively early in-memory database system that aspires to support workloads similar to a traditional relational system.
- TimesTen was founded in 1995 and acquired by Oracle in 2005.
- TimesTen implements a fairly familiar SQL-based relational model.
- In a TimesTen database, all data is memory resident.
- Persistence is achieved by writing periodic snapshots of memory to disk, as well as writing to a disk-based transaction log following a transaction commit.
- If the power fails between the transaction commit and the time the transaction log is written, then data could be lost.



1. When the database is started, all data is loaded from checkpoint files into main memory.
2. The application interacts with TimesTen via SQL requests that are guaranteed to find all relevant data inside that main memory.
3. Periodically or when required database data is written to checkpoint files.
4. An application commit triggers a write to the transaction log.
5. The transaction log can be used to recover the database in the event of failure.

b. Write a jQuery code to change text contents of the elements on button click. (5)

Ans.

```
<!DOCTYPE html>
<html>
<head>
  <title> jQuery </title>
  <script src="jquery.js"></script>
```



```

<script>
$(document).ready(function()
{
    $("button").click(function()
    {
        document.write("hello world");
    }
    );
}
);
</script>
</head>
<body>
<p>Hello! Welcome in JQuery Language!!</p>
<button>Click me</button>
</body>
</html>

```

Output:

Hello! Welcome in JQuery Language!!

Click me

Click on the button **click me**.

hello world

c. Explain how we can create our own custom event in jQuery with an example.

(5)

Ans. Custom event means you can create your own events in jQuery.

Example :

Create a custom event to fire an alert box on pressing any key on the keyboard.

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.2.1/jquery.min.js"></script>
```

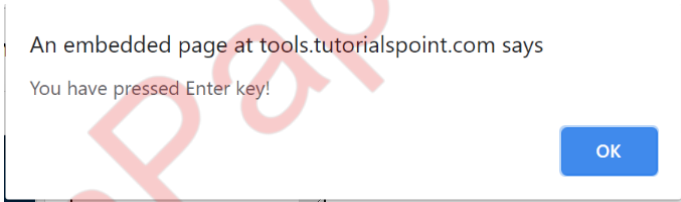
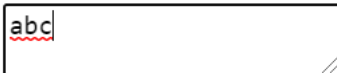
```
<script>
```

```
$(document).ready(function(){
```

```
    $('textarea').bind("enterKey",function(e){
```

```
    alert("You have pressed Enter key!");
  });
  $('textarea').keyup(function(e){
    if(e.keyCode == 13)
    {
      $(this).trigger("enterKey");
    }
  });
});
</script>
</head>
<body>
<textarea rows="2" cols="20">
</textarea>
</body>
</html>
```

Output:



d. What is Ajax? What is the use of Ajax? Explain how Ajax can be used with jQuery.

(5)

Ans.

- AJAX is an acronym standing for Asynchronous JavaScript and XML and this technology helps us to load data from the server without a browser page refresh.
- Examples of applications using AJAX: Gmail, Google Maps, Youtube, and Facebook tabs.
- JQuery is a great tool which provides a rich set of AJAX methods to develop next generation web application.
- jquery provides several methods for AJAX functionality.
- With the jquery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post - And you can load the external data directly into the selected HTML elements of your web page

Writing regular AJAX code can be a bit tricky, because different browsers have different syntax for AJAX implementation. This means that you will have to write extra code to test for different browsers. However, the jquery team has taken care of this for us, so that we can write AJAX functionality with only one single line of code.

➤ For example jquery load() Method :

The jQuery load() method loads data from the server and place the returned HTML into the selected element. This method provides a simple way to load data asynchronous from a web server. The basic syntax of this method can be given with: \$(selector).load(URL, data, complete);

The parameters of the load() method has the following meaning:

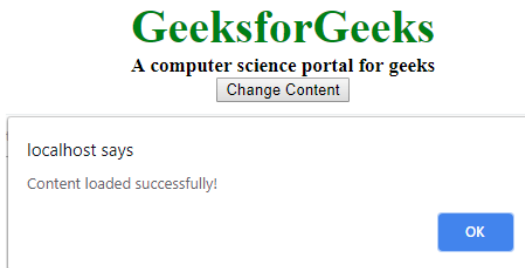
- 1) The required URL parameter specifies the URL of the file you want to load.
- 2) The optional data parameter specifies a set of query string (i.e. key/value pairs) that is sent to the web server along with the request.
- 3) The optional complete parameter is basically a callback function that is executed when the request completes. The callback is fired once for each selected element.

```
<!DOCTYPE html>
<html>
  <head>
    <script src=
      "https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/jquery.min.js">
    </script>
    <script>
      $(document).ready(function(){
        $("button").click(function(){
          $("#div_content").load("gfg.txt");
        });
      });
    </script>
    <style>
      body {
        text-align: center;
      }
      .gfg {
        font-size:40px;
        font-weight: bold;
        color: green;
      }
      .geeks {
        font-size:17px;
        color: black;
      }
      #div_content {
        font-size: 40px;
        font-weight: bold;
        color: green;
      }
    </style>
  </head>
  <body>
    <div id="div_content">
      <div class = "gfg">GeeksforGeeks</div>
      <div class = "geeks">A computer science portal for geeks</div>
    </div>
    <button>Change Content</button>
```

```
</body>
</html>
```

Output:

Before:



After click on change content button:



e. Explain how to add and remove elements to DOM in jQuery with an example.

(5)

Ans. Adding Elements to DOM

- jQuery provides several methods that allows us to insert new content inside an existing element.
- There are 3 ways to insert elements in the DOM. Inserting into the DOM.

- DOM Insertion, Around: These methods let you insert elements around existing ones.(wrap(),wrapAll(),wrapInner())
- The wrap() method wraps specified HTML element(s) around each selected element.

Example

Wrap a <div> element around each <p> element:

```
$("#button").click(function(){
$("#p").wrap("<div></div>");
});
```

- DOM Insertion, Inside: These methods let you insert elements within existing ones.(append(),appendTo(), html(),prepend(),prependTo(),text())
- The append() method inserts specified content at the end of the selected elements.

Example

Insert content at the end of all <p> elements:

```
$("#button").click(function(){
$("#p").append("<b>Appended text</b>");
});
```

- The prepend() method inserts specified content at the beginning of the selected elements.
- The html() method sets or returns the content (innerHTML) of the selected elements.
- DOM Insertion, Outside: These methods let you insert elements outside existing ones that are completely separate(after(),before(),insertAfter(),insertBefore())
- The after() method inserts specified content after the selected elements.

Example

Insert content after each <p> element:

```
$("#button").click(function(){
  $("#p").after("<p>Hello world!</p>");
});
```

The before() method inserts specified content in front of (before) the selected elements.

Example

Insert content before each <p> element:

```
$("#button").click(function(){
  $("#p").before("<p>Hello world!</p>");
});
```

Removing Elements to DOM

- 2) jQuery provides handful of methods, such as empty(), remove(), unwrap() etc. to remove existing HTML elements or contents from the document.
- The empty() method removes all child nodes and content from the selected elements

Example

Remove the content of all <div> elements:

```
$("#button").click(function(){
  $("#div").empty();
});
```

- The remove() method removes the selected elements, including all text and child nodes.
- This method also removes data and events of the selected elements.

Example

Remove all <p> elements:

```
$("#button").click(function(){
  $("#p").remove();
});
```

f. Write a jQuery code to add a CSS class to the HTML elements.

(5)

Ans.

```
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="utf-8">
<title>Adding a Class to the Elements in jQuery</title>
<style>
  .page-header{
    color: red;
    text-transform: uppercase;
  }
  .highlight{
    background: yellow;
  }
  .hint{
    font-style: italic;
  }
</style>
```

```

<script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("h1").addClass("page-header");
    $("p.hint").addClass("highlight");
  });
});
</script>
</head>
<body>
  <h1>Demo Text</h1>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit...</p>
  <p class="hint"><strong>Tip:</strong> Lorem Ipsum is dummy text.</p>
  <button type="button">Add Class</button>
</body>
</html>

```

Output:

Demo Text

Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Tip: Lorem Ipsum is dummy text.

Add Class

DEMO TEXT

Lorem ipsum dolor sit amet, consectetur adipiscing elit...

Tip: Lorem Ipsum is dummy text.

Add Class

5. Attempt any three of the following:

a. Explain the use of `json_encode` and `json_decode` function with an example.

(5)

Ans.

- A common use of JSON is to read data from a web server, and display the data in a web page.
- We can exchange JSON data between the client and a PHP server. PHP has some built-in functions to handle JSON.
- Objects in PHP can be converted into JSON by using the PHP function `json_encode()`.

Example:

```

<?php
$myObj->name = "John";
$myObj->age = 30;
$myObj->city = "New York";
$myJSON = json_encode($myObj);
echo $myJSON;
?>

```

- PHP's `json_decode` function takes a JSON string and converts it into a PHP variable. Typically, the JSON data will represent a JavaScript array or object literal which `json_decode` will convert into a PHP
- array or object.
- The following two examples demonstrate, first with an array, then with an object:

Example 1:

```
$json = ['apple','orange','banana','strawberry'];
$ar = json_decode($json);
// access first element of $ar array
echo $ar[0]; // apple
```

Example 2:

```
$json = {
  "title": "JavaScript: The Definitive Guide",
  "author": "David Flanagan",
  "edition": 6
};
$book = json_decode($json);
// access title of $book object
echo $book->title; // JavaScript: The Definitive Guide
```

b. List and explain any 5 XMLHttpRequest Event Handlers used for Monitoring the Progress of the HTTP Request. (5)

Ans.

Event Handlers	Event Handler Event Type
<code>onloadstart *</code>	<code>loadstart *</code>
<code>onprogress</code>	<code>progress</code>
<code>onload</code>	<code>load</code>
<code>onloadend *</code>	<code>loadended *</code>
<code>onerror</code>	<code>error</code>
<code>ontimeout</code>	<code>timeout</code>
<code>onabort *</code>	<code>abort *</code>
<code>onreadystatechange</code>	<code>readystatechange</code>

- The event handlers will alert our application to a variety of notifications pertaining to the state of the request.
- Furthermore, they can be utilized in one of two possible implementations.
- The first is that we can remain object-oriented and register the event of the state to which we choose to listen.
- For each event to which we listen, we can assign a particular function to be triggered upon notification.

– As different browsers implement various ways to register an event, it is necessary to make use of a cross-browser solution.

– The Registration for Event Listeners Belonging to the xhr object for Each Notification of State :

```
var xhr = new XMLHttpRequest();
addListener(xhr, 'loadstart', function() { alert("load-start"); });
addListener(xhr, 'progress', function() { alert("progress"); });
addListener(xhr, 'load', function() { alert("load"); });
addListener(xhr, 'loadended', function() { alert("loadended"); });
addListener(xhr, 'timeout', function() { alert("timeout"); });
addListener(xhr, 'abort', function() { alert("abort"); });
addListener(xhr, 'readystatechange', function() { alert("readystatechange"); });
```

```
//cross browser addListener
function addListener(elem, eventName, handler) {
  if (elem) {
    elem.addEventListener(eventName, handler, false);
  } else if (elem.attachEvent) {
    elem.attachEvent('on' + eventName, handler);
  } else {
    elem['on' + eventName] = handler;
  }
}
```

c. What is the use of Stringify function? What are the different parameters that can be passed in Stringify function? Explain with an example. (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);
```

The result will be a string following the JSON notation.

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. This padding provides an added layer of readability to the produced string.

7. Code:

```
<html>
<head>
<title>JSON programs </title>
</head>
<body>
<script>
  var value = {
    name: "Logan",
    age: 21,
    location: "London"
  };
  var result = JSON.stringify(value);
  document.write("value of result = " + result + "<br>");
  document.write("type of result = " + typeof result);
</script>
</body>
</html>
```

Output:

```
value of result = {"name":"Logan", "age":21, "location":"London"}
type of result = string
```

d. Write a short note on JSON Arrays.

(5)

Ans.

- Arrays in JSON are almost the same as arrays in JavaScript.
- In JSON, array values must be of type string, number, object, array, boolean or null.
- In JavaScript, array values can be all of the above, plus any other valid JavaScript expression, including functions, dates, and undefined.

Arrays in JSON Objects:

- Arrays can be values of an object property

Example

```
{
  "name":"John",
  "age":30,
  "cars":[ "Ford", "BMW", "Fiat" ]
```

```
}
```

Accessing Array Values :

- You access the array values by using the index number

Example

```
x = myObj.cars[0];
```

Looping Through an Array

You can access array values by using a for-in loop:

Example

```
for (i in myObj.cars) {
```

```
x += myObj.cars[i];
```

```
}
```

Or you can use a for loop:

Example

```
for (i = 0; i < myObj.cars.length; i++) {
```

```
x += myObj.cars[i];
```

```
}
```

Nested Arrays in JSON Objects :

- Values in an array can also be another array, or even another JSON object

Example

```
myObj = {
```

```
"name": "John",
```

```
"age": 30,
```

```
"cars": [
```

```
{ "name": "Ford", "models": [ "Fiesta", "Focus", "Mustang" ] },
```

```
{ "name": "BMW", "models": [ "320", "X3", "X5" ] },
```

```
{ "name": "Fiat", "models": [ "500", "Panda" ] }
```

```
]
```

```
}
```

Modify Array Values :

- Use the index number to modify an array

Example

```
myObj.cars[1] = "Mercedes";
```

Delete Array Items :

- Use the delete keyword to delete items from an array

Example

```
delete myObj.cars[1];
```

Ans.

Methods	Description
<code>create();</code>	Used to create a database
<code>exists(callback);</code>	Used to determine if a database currently exists
<code>get(id[,id] , [object], callback);</code>	Used to fetch a particular document
<code>view(id, [object] ,callback);</code>	Used to query an existing view
<code>save([id], object , callback);</code>	Used to save a document to the current database. This can be used to save either a view or an entry.

create

- The first method that we will review is the create method.
- Use of the create method provides our Node application with the ability to initialize a database within CouchDB.
- Use of the method is as simple as invoking the method upon a database.
 - Invoking the Creation of Our Database Reference


```

          //..truncated code
          var gbDataBase = couchDB.database('guestbook');
          gbDataBase.create();
          
```

exists

- The exists method is an asynchronous method used to determine if a database currently exists.
- The advantage of such a method is to determine whether a database already exists, lest we insert values to a table
- we did not intend to.
- As an asynchronous method, the invocation of the exists call must be provided with a callback function.
 - Callback Signature of the exists Method


```

          function(err, exists);
          
```

get

- The get method, as you may suspect, initiates HTTP requests utilizing the GET request method.
- The get method is used to obtain documents that are associated with the targeted database in an asynchronous fashion.
- The arguments represent the document by its ID, an object, and a callback function.
- The first parameter, id, can be provided either as a singular identifier or as an array of multiple document IDs supplied as an array.
- The second parameter of our get method is that of an object.

f. "JSON is better than XML".Comment.

(5)

Ans.

1. JSON is Unlike XML Because

- JSON doesn't use end tag
- JSON is shorter
- JSON is quicker to read and write
- JSON can use arrays

2. The biggest difference is: XML has to be parsed with an XML parser. JSON can be parsed by a standard JavaScript function.XML is much more difficult to parse than JSON.

3. JSON is parsed into a ready-to-use JavaScript object.For AJAX applications, JSON is faster and easier than XML:

4. Using XML

- Fetch an XML document
- Use the XML DOM to loop through the document
- Extract values and store in variables

5. Using JSON

- Fetch a JSON string
- JSON.Parse the JSON string.

6. The following JSON and XML examples both define an employees object, with an array of 3 employees:

JSON Example

```
{"employees":[
  { "firstName":"John", "lastName":"Doe" },
  { "firstName":"Anna", "lastName":"Smith" },
  { "firstName":"Peter", "lastName":"Jones" }
]}
```

XML Example

```
<employees>
  <employee>
    <firstName>John</firstName> <lastName>Doe</lastName>
  </employee>
  <employee>
    <firstName>Anna</firstName> <lastName>Smith</lastName>
  </employee>
  <employee>
    <firstName>Peter</firstName> <lastName>Jones</lastName>
  </employee>
</employees>
```