TYBSC IT

Next Generation Technology (NGT)

SEM 5 (APR-2019) Q.P.Code:57844

1.Attempt any three of the following:

a. Explain the three V's of Big Data. (5)

Ans: The three V's of Big Data are as follows **:**

Volume:

a. Volume in big data means the size of the data.
b. as businesses are becoming more transaction-oriented, we see ever increasing numbers of transactions; more devices are getting connected to the Internet, which is adding to the volume
c. This huge volume of data is the biggest challenge for big data technologies.
d. The storage and processing power needed to store, process, and make accessible the data in a timely and cost effective manner is massive.

Variety:

a. The data generated from various devices and sources follows no fixed format or structure.
b. Variety of big data are text files, log files, streaming videos, photos, meter readings, stock ticker data, PDFs, audio, and various other unstructured formats.
c. There is no control over the structure of the data these days.
d. New sources and structures of data are being created at a rapid pace.
e. Example: to provide alternate routes for commuters, a traffic analysis application needs data feeds from millions of smartphones and sensors to provide accurate analytics on traffic conditions and alternate routes.

Velocity:

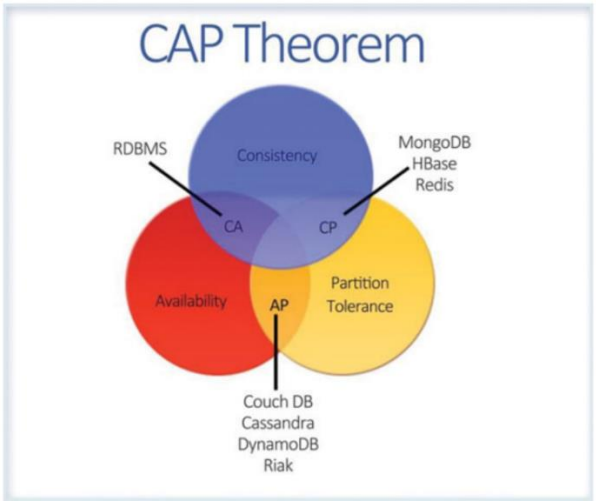
a. Velocity in big data is the speed at which data is created and the speed at which it is required to be processed.
b. If data cannot be processed at the required speed, it loses its significance.
c. Due to data streaming in from social media sites, sensors, tickers, metering, and monitoring, it is important for the organizations to speedily process data both when it is on move and when it is static.
d. Reacting and processing quickly enough to deal with the velocity of data is one more challenge for big data technology.

---

b. Briefly 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.

- The CAP theorem states that at any point in time a distributed system can fulfil only two of the above three guarantees.

---

c. What is MongoDB Design Philosophy? Explain. (5)

Ans.

1. **Speed, Scalability, and Agility :**
   a. MongoDB was to create a database that was fast, massively scalable, and easy to use.
   b. To achieve speed and horizontal scalability in a partitioned database, as explained in the CAP theorem.
   c. As per CAP theorem, MongoDB provides high availability, scalability, and partitioning at the cost of consistency and transactional support.
   d. MongoDB uses documents to make it flexible, scalable, and fast.

2. **Non-Relational Approach Traditional :**
   a. MongoDB stores its data in BSON documents where all the related data is placed together, which means everything is in one place.
   b. The queries in MongoDB are based on keys in the document, so the documents can be spread across multiple servers.
   c. Non-Relational Approach means no table approach i.e, table should not link to each other.
   d. Scalable storage of data.
   e. MongoDB has a primary-secondary replication where the primary accepts the write requests.

3. **JSON-Based Document Store :**
   a. MongoDB uses a JSON-based (JavaScript Object Notation) document store for the data.
   b. JSON/BSON offers a schema-less model, which provides flexibility in terms of database design. Changes can be done to the schema seamlessly.
   c. This design also makes for high performance by providing for grouping of relevant data together internally and making it easily searchable.
   d. A document contains data in form of key-value pairs.

```
Eg: {
    "Name": "ABC",
    "Phone": ["1111111",
     ........"222222" ........],
     "Fax":..
     }
```

4. Performance vs. Features :

a. MongoDB is a document-oriented DBMS where data is stored as documents.
b. It does not support JOINs, and it does not have fully generalized transactions.
c. It does provide support for secondary indexes, it enables users to query using query documents, and it provides support for atomic updates at a per document level.
d. It provides a replica set, which is a form of master-slave replication.

5. Running the Database Anywhere :

a. In order to make MongoDB fast and high performance, certain features commonly available in RDBMS system are not available in MongoDB.
b. One of the main design decisions was the ability to run the database from anywhere, which means it runs on servers, VMs, or even on the cloud using the pay-for-what-you-use services.
c. The language used for implementing MongoDB is C++, which enables MongoDB to achieve this goal. MongoDB to run on almost any type of machine.

---

d. Differentiate between SQL and NoSQL Databases. (5)

Ans.

SQL	NOSQL
1. SQL was developed in 1970.	**1.** NoSQL was developed in 2000.
2. Data is stored in rows and columns in a table.	2. data is stored as a key-value pair for key-value stores.
3. These databases have fixed or static or predefined schema.	3. They have dynamic schema.
4. These databases are not suited for hierarchical data storage.	4. These databases are best suited for hierarchical data storage.
5. These databases are best suited for complex queries	5. These databases are not so good for complex queries
6. Maturity have been around for a long time.	6. Some of them are mature; others are evolving.
7. SQL databses are vertically Scalable.	7. NoSQL databses are horizontally scalable.
8. Strong consistency.	8. Consistency dependent on the product.
9. Supports ACID and transactions.	9. Supports BASE, partitioning and availability.
10. Examples : SQL Server, Oracle, MySQL.	10. Examples : MongoDB, HBase, Cassandra.

---

e. Write a short note on Non-Relational Approach. **(5)**

Ans.

- Non-Relational databases are any type of databases that does not follow the relational database model.
- They are also known as NoSQL databases and are growing in popularity as a result of the rise of big data and the need to handle the great volumes, variety and velocity of data.
- Traditional RDBMS platforms provide scalability using a scale-up approach, which requires a faster server to increase performance.

The following issues in RDBMS systems led to why MongoDB and other NoSQL databases were designed the way they are designed:

- In order to scale out, the RDBMS database needs to link the data available in two or more systems in order to report back the result. This is difficult to achieve in RDBMS systems since they are designed to work when all the data is available for computation together. Thus the data has to be available for processing at a single location.

- In case of multiple Active-Active servers, when both are getting updated from multiple sources there is a challenge in determining which update is correct.

- When an application tries to read data from the second server, and the information has been updated on the first server but has yet to be synchronized with the second server, the information returned may be stale.

- The MongoDB team decided to take a non-relational approach to solving these problems.
- MongoDB stores its data in BSON documents where all the related data is placed together, which means everything is in one place.
- The queries in MongoDB are based on keys in the document, so the documents can be spread across multiple servers.
- Querying each server means it will check its own set of documents and return the result. This enables linear scalability and improved performance.
- MongoDB has a primary-secondary replication where the primary accepts the write requests.
- If the write performance needs to be improved, then sharding can be used; this splits the data across multiple machines and enables these multiple machines to update different parts of the datasets.
- Sharding is automatic in MongoDB; as more machines are added, data is distributed automatically.

---

f. Discuss the various applications of Big Data. **(5)**

Ans. The various applications of Big Data some of them are as follows:

1. **Big Data in Healthcare**

   a. Big Data and healthcare are an ideal match. It complements the healthcare industry better than anything ever will. The amount of data the healthcare industry has to deal with is unimaginable.
   b. Identifying unusual patterns of certain medicines to discover ways for developing more economical solutions is a common practice these days.

2. **Big Data in Education**

    a. Big Data is the key to shaping the future of the people and has the power to transform the education system for better.
    b. Big Data is providing assistance in evaluating the performances of both the teachers as well as the students.
    c. Some of the top universities are using Big Data as a tool to renovate their academic curriculum.

3. **Big Data in E-commerce**

    a. Some of the biggest E-commerce companies of the world like Amazon, Flipkart, Alibaba, and many more are now bound to Big Data and analytics is itself an evidence of the level of popularity Big Data has gained in recent times.
    b. Big Data's recommendation engine is one of the most amazing applications the Big Data world has ever witnessed.
    c. It furnishes the companies with a 360-degree view of its customers.

4. **Big Data in Media and Entertainment**

    a. Media and Entertainment industry is all about art and employing Big Data in it is a sheer piece of art.
    b. Earlier the companies broadcasted the Ads randomly without any kind of analysis.
    c. But after the advent of Big Data analytics in the industry, companies now are aware of the kind of Ads that attracts a customer and the most appropriate time to broadcast it for seeking maximum attention.

5. **Big Data in Finance**

    a. The functioning of any financial organization depends heavily on its data and to safeguard that data is one of the toughest challenges any financial firm faces. Data has been the second most important commodity for them after money.
    b. Big Data is bossing the key areas of financial firms such as fraud detection**,** risk analysis**,** algorithmic trading**,** and customer contentment.

6. **Big Data in Travel Industry**

    a. Big Data and analytics, travel companies are now able to offer more customized traveling experience. They are now able to understand their customer's requirements in a much-enhanced way.
    b. From providing them with the best offers to be able to make suggestions in real-time, Big Data is certainly a perfect guide for any traveller.

2.Attempt any three of the following:

a. Explain the 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.

---

b. What are the various tools available in MongoDB Explain. (5)

Ans. There are various tools that are available as part of the MongoDB installation:

1. mongodump :
   This utility is used as part of an effective backup strategy. It creates a binary export of the database contents.
2. mongorestore :
   The binary database dump created by the mongodump utility is imported to a new or an existing database using the mongorestore utility.
3. bsondump :
   This utility converts the BSON files into human-readable formats such as JSON and CSV. For example, this utility can be used to read the output file generated by mongodump.
4. mongoimport , mongoexport :
   mongoimport provides a method for taking data in JSON, CSV, or TSV formats and importing it into a mongod instance. mongoexport provides a method to export data from a mongod instance into JSON, CSV, or TSV formats.
5. mongostat , mongotop , mongosniff:
   These utilities provide diagnostic information related to the current operation of a mongod instance.

---

c. Discuss the points to be considered while Importing data in Shared environment. (5)

Ans. The points to be considered while Importing data in Shared environment.

- Pre-Splitting of the Data :
  - Instead of leaving the choice of chunks creation with MongoDB, you can tell MongoDB how to do so using the following command:
  - db.runCommand({ split : "practicalmongodb.mycollection" , middle : { shardkey : value } });
    Post this you can also let MongoDB know which chunks goes to which node. For all this you will need knowledge of the data you will be imported to the database.

- Deciding on the Chunk Size :
    - You need to keep the following points in mind when deciding on the chunk size :
    - If the size is too small, the data will be distributed evenly but it will end up having more frequent migrations, which will be an expensive operation at the mongos layer.
    - If the size is large, it will lead to less migration, reducing the expense at the mongos layer, but you will end up with uneven data distribution.

- Choosing a Good Shard Key
    - It's very essential to pick a good shard key for good distribution of data among nodes of the shard cluster .

- Monitoring for Sharding :
    - The sharding cluster requires an additional monitoring to ensure that all its operations are functioning appropriately and the data is distributed effectively among the nodes.

- Monitoring the Config Servers :
    - The config server stores the metadata of the sharded cluster.

The mongos caches the data and routes the request to the respective shards. If the config server goes down but there's a running mongos instance, there's no immediate impact on the shard cluster and it will remain available for a while. However, you won't be able to perform operations like chunk migration or restart a new mongos.

---

d. Explain the concept Inserting by Explicitly Specifying_id. (5)

Ans.

- In MongoDB, _id field as the primary key for the collection so that each document can be uniquely identified in the collection. The _id field contains a unique ObjectID value.

- By default 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.

- In the following example, you will see how to explicitly specify the _id field when inserting the documents within a collection.
- While explicitly specifying the _id field, you have to keep in mind the uniqueness of the field; otherwise the insert will fail.
    - The following command explicitly specifies the _id field:

```
>db.users.insert({"_id":10, "Name": "explicit id"})
```

    - The insert operation creates the following document in the users collection:

```
{ "_id" : 10, "Name" : "explicit id" }
```

    - This can be confirmed by issuing the following command:

```
>db.users.find()
```

- When you query the documents in a collection, you can see the ObjectId for each document in the collection.

- If you want to ensure that MongoDB does not create the _id Field when the collection is created and if you want to specify your own id as the _id of the collection, then you need to explicitly define this while creating the collection.
- When explicitly creating an id field, it needs to be created with _id in its name.

• Example :

```
db.Employee.insert({_id:10, "EmployeeName" : "Smith"})
```

---

e. Discuss Indexes and its types. **(5)**

Ans. The different types of indexes that are available in MongoDB.

- _id index :
  - This is the default index that is created on the _id field. This index cannot be deleted.

- Secondary Indexes :
  - All indexes that are user created using ensureIndex() in MongoDB are termed as secondary indexes.
  - These indexes can be created on any field in the document or the sub document.
  - These indexes can be created on a field that is holding a sub-document.
  - These indexes can either be created on a single field or a set of fields. When created with set of fields, it's also termed a compound index .
  - If the index is created on a field that holds an array as its value, then a multikey index is used for indexing each value of the array separately.
  - Multikey compound indexes can also be created. However, at any point, only one field of the compound index can be of the array type.

- Indexes with Keys Ordering :
  - MongoDB indexes maintain references to the fields. The refernces are maintained in either an ascending order or descending order.
  - This is done by specifying a number with the key when creating an index. This number indicates the index direction.
  - This index contains references to the documents that are sorted in the following manner:
  - First by the username field in ascending order.
  - Then for each username sorted by the timestamp field in the descending order.

- Unique Indexes :
  - When you create an index, you need to ensure uniqueness of the values being stored in the indexed field.
  - you can create indexes with the Unique property set to true.
  - The following command can be run to create the unique index:
    db.payroll.ensureIndex({ "userid": 1 }, { unique: true })

- Sparse Indexes :
  - A sparse index is an index that holds entries of the documents within a collection that has the fields on which the index is created.
  - The index is said to be sparse because this contains documents with the indexes field and miss the documents when the fields are missing.
  - Null value is stored in case the fields are missing.

- Geospatial Indexes :
    - MongoDB provides geospatial indexes . To create a geospatial index, a coordinate pair in the following forms must exist in the documents:
    - Either an array with two elements
    - Or an embedded document with two keys (the key names can be anything).

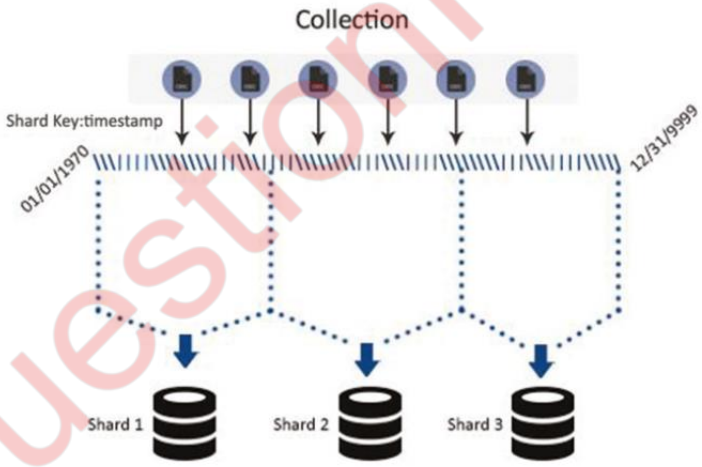
- Geohaystack Indexes :
    - Geohaystack indexes are bucket-based geospatial indexes (also called geospatial haystack indexes).
    - They are useful for queries that need to find out locations in a small area and also need to be filtered along another dimension, such as finding documents with coordinates within 10 miles and a type field value as restaurant.

---

f. Write a short note on Data Distribution Process. **(5)**

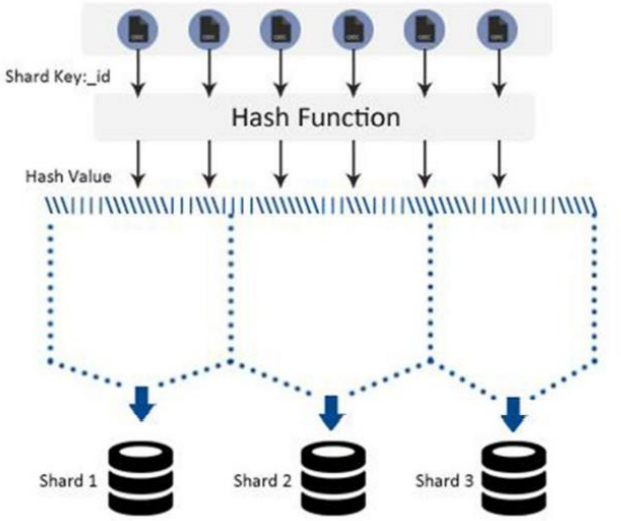
– **Ans.** In MongoDB, the data is sharded or distributed at the collection level.
– The collection is partitioned by the shard key.
– There are two ways MongoDB enables distribution of the data:
  I. Range-based partitioning
  II. Hash-based partitioning.

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.

---

3. Attempt any three of the following:

a. What is Wired Tiger Storage Engine? (5)

Ans.

- When the storage option selected is WiredTiger, data, journals, and indexes are compressed on disk.
- The compression is done based on the compression algorithm specified when starting the mongod.
- Snappy is the default compression option.
- Under the data directory, there are separate compressed wt files corresponding to each collection and indexes.
- Journals have their own folder under the data directory.
- The compressed files are actually created when data is inserted in the collection (the files are allocated at write time, no preallocation).

  For example : if you create collection called users , it will be stored in collection-0—2259994602858926461 files and the associated index will be stored in index-1—2259994602858926461 , index-2—2259994602858926461 , and so on.

- WiredTiger uses the traditional B+ tree structure for storing and managing data but that's where the similarity ends.
- Unlike B+ tree, it doesn't support in-place updates. WiredTiger cache is used for any read/write operations on the data.
- The trees in cache are optimized for in-memory access.

- Advantages of the WiredTiger Storage Engine
  - Efficient storage due to a multiple of compression technologies such as the Snapp, gzip and prefix compressions.
  - It is highly scalable with concurrent reads and writes. This in the end improves on the throughput and general database performance.
  - Assure data durability with write-ahead log and usage of checkpoints.
  - Optimal memory usage. The WiredTiger uses both the internal cache and file system cache.

---

b. List and explain the MongoDB limitation. **(5)**

Ans. The MongoDB limitation are :

- Max document size: 16 MB.
- Max document nesting level: 100 (documents inside documents inside documents…)
- Namespace is limited to ~123 chars (namespace is db_name + collection_name (or index_name)).
- DB name is limited to 64 chars.
- Default .ns file can store about 24000 namespaces (again, a namespace is referring to a collection or an index)
- If you index some field, that field can't contain more than 1024 bytes
- Max 64 indexes per collection
- Max 31 fields in a compound index
- If you set a limit of documents in a capped collection, this limit can't be more than 2**32. Otherwise, number of documents is unlimited.
- On linux, one mongod instance can't store more than 64 TB of data
- On windows, mongod can't store more than 4 TB of data (8 TB without journal)
- Max 12 nodes in a replica set
- Max 7 voting nodes in a replica set
- You can't automatically rollback more than 300 MB of data. If you have more than this, manual invervention is needed.
- You can't refer `db` object in `$where` functions.
- If you want to shard a collection, it must be smaller than 256 GB, or else it will likely fail to shard.
- Max 512 bytes for shard key values.

---

c. Explain the hardware requirements for MongoDB. **(5)**

Ans. The actual hardware configuration depends on your data, availability requirement, queries, performance criteria, and the selected hardware components ' capabilities.

• Memory :

a. Since memory is used extensively by MongoDB for a better performance, the more memory, the better the performance.

• Storage :

a. MongoDB can use SSDs (solid state drives) or local attached storage.
b. MongoDB's disk access patterns don't have sequential properties, SSDs usage can enable customers to experience substantial performance gains.

c. Another benefit of using a SSD is if the working set no longer fits in memory, they provide a gentle degradation of performance.
d. Most MongoDB deployments should use RAID-10. When using the WiredTiger storage engine, the use of a XFS file system is highly recommended due to performance issues.

• CPU :
a. Since MongoDB with a MMAPv1 storage engine rarely encounters workloads needing a large number of cores, it's preferable to use servers with a faster clock speed than the ones with multiple cores but slower clock speed.
b. The WiredTiger storage engine is CPU bound, so using a server with multiple cores will offer a significant performance improvement.

---

d. Write a short note on GridFS. **(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
```

- To see the file's document in database, you can use find query −

```
>db.fs.files.find()
```

- 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 −

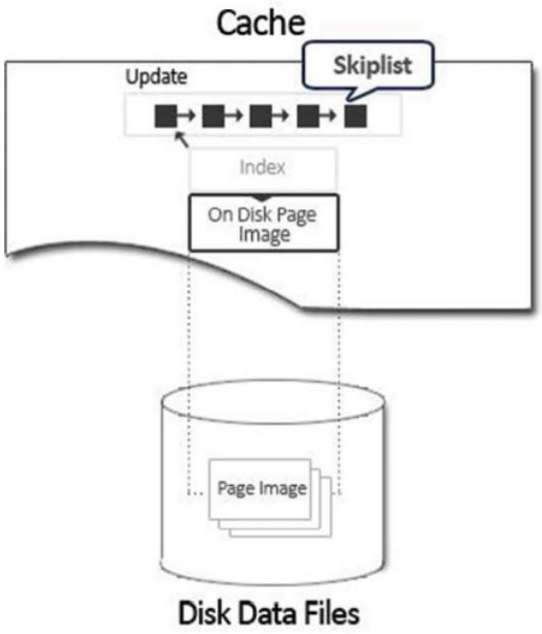
```
>db.fs.chunks.find({files_id:ObjectId('534a811bf8b4aa4d33fdf94d')})
```

---

e. How are read and write operations performed in MongoDB? (5)

Ans.

- When MongoDB updates and reads from the DB, it is actually reading and writing to memory.
- If a modification operation in the MongoDB MMAPv1 storage engine increases the record size bigger then the space allocated for it, then the entire record will be moved to a much bigger space with extra padding bytes.
- By default, MongoDB uses power of 2-sized allocations so that every document in MongoDB is stored in a record that contains the document itself and extra space (padding).
- Once the record is moved, the space that was originally occupied is freed up and will be tracked as free lists of different size.
- Every 60 seconds the files in RAM are flushed to disk.
- To prevent data loss in the event of a power failure, the default is to run with journaling switched on.
- The behaviour of journal is dependent on the configured storage engine.
- The write operation in WiredTiger never updates in-place.
- Whenever an operation is issued to WiredTiger, internally it’s broken into multiple transactions wherein each transaction works within the context of an in-memory snapshot.
- Writers can create new versions concurrent with the readers.
- The write operations do not change the page; instead the updates are layered on top of the page.
- A skipList data structure is used to maintain all the updates, where the most recent update is on the top.
- Whenever a user reads/writes the data, the index checks whether a skiplist exists.
- If skiplist exists, the data at the head of the list is returned to the threads, which then update the data.



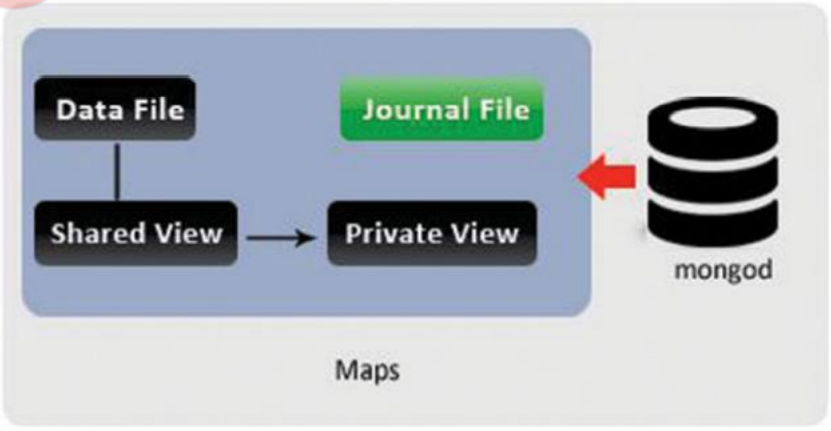
---

f. Discuss how data is written using Journaling. **(5)**

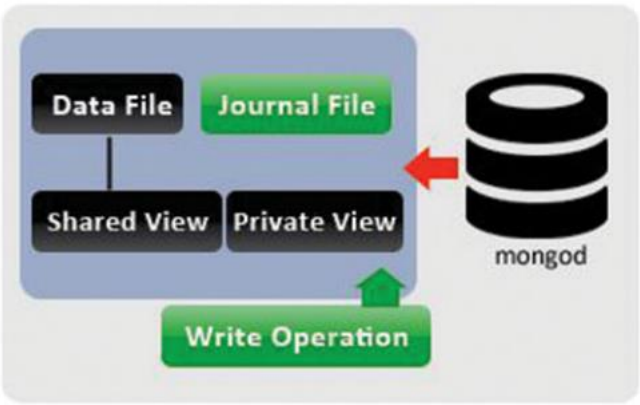
Ans.

- In the MongoDB system, mongod is the primary daemon process.
- The disk has the data files and the journal files.
- When the mongod is started, the data files are mapped to a shared view i.e, virtual address space.

- If a data file is 2000 Bytes on a memory address ranges from 1,000,000 - 1,002,000.
- Data is not actually loaded it is just map.
- In this scenario, the journaling is not yet enabled.
- When journaling is enabled, a second mapping is made to a private view by the mongod.



- The data file is not directly connected to the private view, so the changes will not be flushed from the private view to the disk by the OS.
- When a write operation is initiated it, first it writes to the private view.
- The journal keeps appending the change description as and when it gets the change.



- If the mongod fails at this point, the journal can replay all the changes even if the data file is not yet modified.

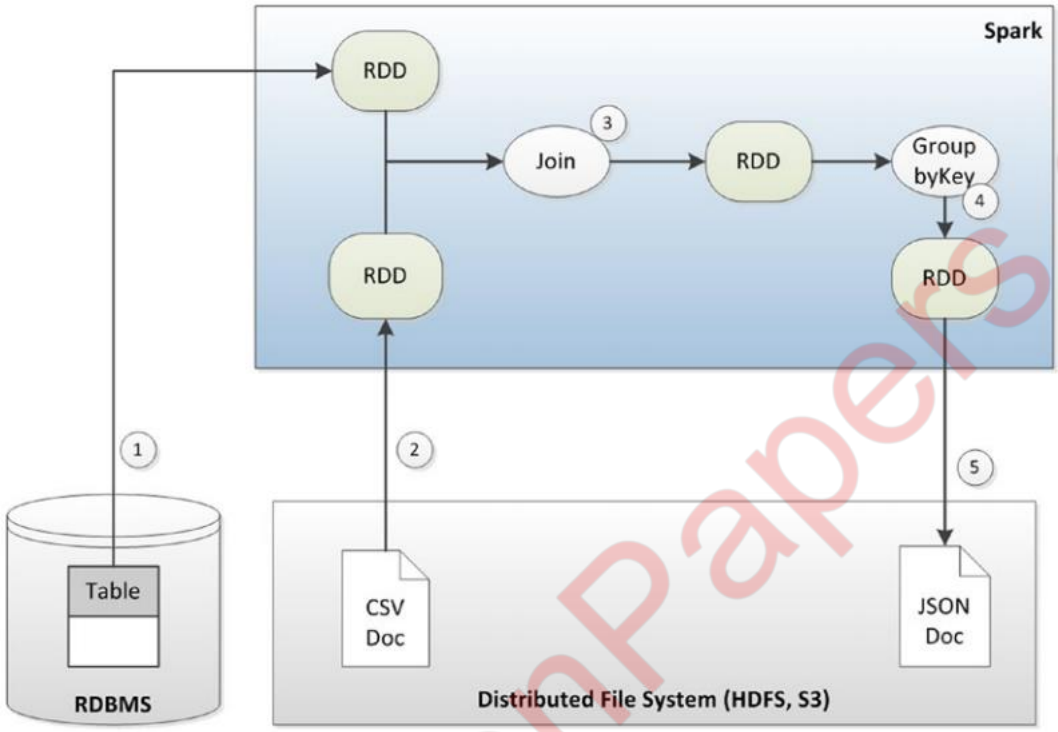
---

4. Attempt any three of the following:

a. Diagrammatically explain the Spark architecture. (5)

Ans.

- In Spark, data is represented as resilient distributed datasets (RDD).
- RDDs are collections of objects that can be partitioned across multiple nodes of the cluster.
- Spark operations on RDDs return new RDDs, rather than modifying the original RDD.



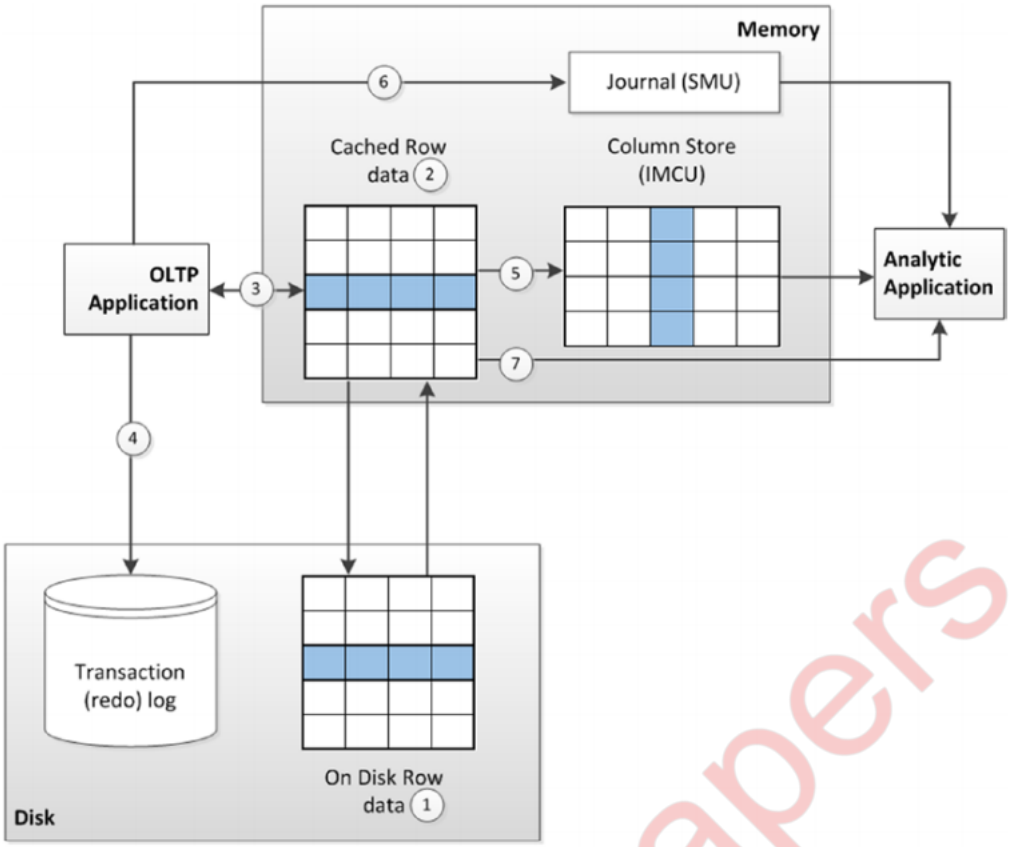
1. Data can be loaded into resilient distributed datasets (RDD) from external sources including relational databases.
2. or a distributed file system such as HDFS.
3. Spark provides high-level methods for operating on RDDs and which output new RDDs. These operations include joins.
4. or aggregations.
5. Spark data may be persisted to disk in a variety of formats.

---

b. Explain the concept of Oracle 12c "in-Memory Databases". (5)

Ans.

- Oracle RDBMS version 12.1 introduced the "Oracle database in-memory" feature.
- This wording is potentially misleading, since the database as a whole is not held in memory.
- Rather, Oracle has implemented an in-memory column store to supplement its disk-based row store.



1. OLTP applications work with the database in the usual manner. Data is maintained in disk files.
2. but cached in memory.
3. An OLTP application primarily reads and writes from memory (3),
4. but any committed transactions are written immediately to the transaction log on disk.
5. When required or as configured, row data is loaded into a columnar representation for use by analytic applications.
6. Any transactions that are committed once the data is loaded into columnar format are recorded in a journal.
7. and analytic queries will consult the journal to determine if they need to read updated data from the row store or possibly rebuild the columnar structure.

---

c. What is jQuery? Explain its features. **(5)**

Ans.

- jQuery is a lightweight, "write less, do more", JavaScript library.
- The purpose of jQuery is to make it much easier to use JavaScript on your website.
- jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.
- jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The features of jQuery are as follows:

i. **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.

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

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

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

v. **Lightweight :**
The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).

vi. **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+

vii. **Latest Technology** :
The jQuery supports CSS3 selectors and basic XPath syntax.

---

d. Discuss the concept of Disk Economics. **(5)**

Ans.

- The promise of SSD has led some to anticipate a day when all magnetic disks are replaced by SSD.
- While this day may come, in the near term the economics of storage and the economics of I/O are at odds: magnetic disk technology provides a more economical medium per unit of storage, while flash technology provides a more economical medium for delivering high I/O rates and low latencies.
- And although the cost of a solid state disk is dropping rapidly, so is the cost of a magnetic disk. An examination of the trend for the cost per gigabyte for SSD and HDD.

- While SSD continues to be an increasingly economic solution for small databases or for performance-critical systems, it is unlikely to become a universal solution for massive databases, especially for data that is infrequently accessed.
- We are, therefore, likely to see combinations of solid state disk, traditional hard drives, and memory providing the foundation for next-generation databases.

---

e. Explain the jQuery DOM Filter Methods. **(5)**

Ans.

- The jQuery is a very powerful tool which helps us to incorporate a variety of DOM traversal methods to select elements in a document randomly or in sequential order.
- Most of the DOM traversal methods do not modify the elements whereas they filter them out upon the given conditions.

- **The filter()** method is used to filter out all the elements that do not match the selected criteria and those matches will be returned.

- **Syntax:**
  $(selector).filter(criteria, function(index))

- **Parameters:**

  criteria : It specifies a selector expression, a jQuery object or one or more elements to be returned from a group of selected elements.
  To specify more than one criteria, use a comma

  function(index) : It specifies a function to run for each element in the set. If the function returns true, the element is kept. Otherwise, it is removed.

  index : The index position of the element in the set.

Example:

```
<html>
  <head>
    <title>The JQuery Example</title>
  </head>

  <body>
    <div>
      <ul>
        <li>list item 1</li>
        <li>list item 2</li>
        <li>list item 3</li>
        <li>list item 4</li>
        <li>list item 5</li>
        <li>list item 6</li>
      </ul>
    </div>
  </body>
</html>
```

f. Write a short note on jQuery Event Handling. **(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.

Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element

The term **"fires/fired"** is often used with events. Example: "The keypress event is fired, the moment you press a key".

The different type of events are:

Commonly Used jQuery Event Methods:

Mouse Events	**Keyboard Events**	**Form Events**	**Document/ Window Events**
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

- **click()**:The function is executed when the user clicks on the HTML element.
  - **Example :**
```
$("p").click(function(){
    $(this).hide();
  });
```

- **dblclick()** : The function is executed when the user double-clicks on the HTML element.
  - **Example :**
```
$("p").dblclick(function(){
    $(this).hide();
  });
```

- **mouseenter()** : The function is executed when the mouse pointer enters the HTML element.
  - **Example :**
```
$("#p1").mouseenter(function(){
    alert("You entered p1!");
  });
```

- **mouseleave()** : The function is executed when the mouse pointer leaves the HTML element.
  - **Example :**
```
$("#p1").mouseleave(function(){
    alert("Bye! You now leave p1!");
  });
```

- **mousedown()** : The function is executed, when the left, middle or right mouse button is pressed down, while the mouse is over the HTML element.

- **Example :** $("#p1").mousedown(function(){
  alert("Mouse down over p1!");
  });

---

5. Attempt any three of the following:

a. Explain JSON datatypes. **(5)**

Ans. In JSON, values must be one of the following data types:

- a string
- a number
- an object (JSON object)
- an array
- a boolean
- *null*

string, number, Boolean, null are simple datatype or primitive datatypes whereas object and array are referred as complex or structure datatypes.

JSON values cannot be one of the following data types:

- a function
- a date
- *undefined*

- **JSON Strings :**
  - Strings in JSON must be written in double quotes.
  - Example **:** { "name":"John" }

- **JSON Numbers :**
  - Numbers in JSON must be an integer or a floating point.
  - Example **:** { "age":30 }

- **JSON Objects :**
  - Values in JSON can be objects.
  - It is set of name or value pairs inserted between { } (curly braces).
  - Example **:** {
    "employee":{ "name":"John", "age":30, "city":"New York" }
    }
  - Objects as values in JSON must follow the same rules as JSON objects.

- **JSON Arrays :**
  - It is an ordered collection of values and begins with [(left bracket) and ends with] (right bracket).
  - The values of array are separated by ,(comma).

- Example : {
  "employees":["John", "Anna", "Peter"]
  }

- **JSON Booleans :**
  - This datatype can be either true/false.
  - Example : { "sale":true }

- **JSON null :**
  - It is just a define nullable value.
  - Example : { "middlename":null }

---

b. Discuss JSON schema with validation libraries. **(5)**

Ans. JSON Schema is a specification for JSON based format for defining the structure of JSON data. It was written under IETF draft which expired in 2011. JSON Schema −

- Describes your existing data format.
- Clear, human- and machine-readable documentation.
- Complete structural validation, useful for automated testing.
- Complete structural validation, validating client-submitted data.

JSON Schema Validation Libraries :

There are several validators currently available for different programming languages. Currently the most complete and compliant JSON Schema validator available is JSV.

JSON Schema Example :

```
{
  "$schema": "http://json-schema.org/draft-04/schema#",
  "title": "Product",
  "description": "A product from Acme's catalog",
  "type": "object",

  "properties": {

   "id": {
     "description": "The unique identifier for a product",
     "type": "integer"
   },

    "name": {
      "description": "Name of the product",
      "type": "string"
    },
```

```
    "price": {
      "type": "number",
      "minimum": 0,
      "exclusiveMinimum": true
    }
  },

  "required": ["id", "name", "price"]
}
```

Various important keywords.

- $schema : The $schema keyword states that this schema is written according to the draft v4 specification.
- title : You will use this to give a title to your schema.
- description : A little description of the schema.
- type : The type keyword defines the first constraint on our JSON data: it has to be a JSON Object.
- properties : Defines various keys and their value types, minimum and maximum values to be used in JSON file.

---

c. Differentiate between XML and JSON. **(5)**

Ans.

JSON	XML
1. It is JavaScript Object Notation.	1. It is Extensible markup language.
2. It is based on JavaScript language.	2. It is based on JavaScript language.
3. It is a way of representing objects.	3. It is a markup language and uses tag structure to represent data items.
4. It does not provides any support for namespaces.	4. It supports namespaces.
5. It supports array.	5. It doesn't supports array.
6. Its files are very easy to read as compared to XML.	6. Its documents are comparatively difficult to read and interpret.
7. It doesn't use end tag.	7. It has start and end tags.
8. It is less secured.	8. It is more secured than JSON.
9. It doesn't supports comments.	9. It supports comments.
10. It supports only UTF-8 encoding.	10. It supports various encoding.

---

d. How do we do encoding and decoding JSON in Python. **(5)**

Ans.

➢ **Encoding JSON in Python (encode):**

– Python encode() function encodes the Python object into a JSON string representation.

Syntax

demjson.encode(self, obj, nest_level=0)

Example

```
import demjson

data = [ { 'a' : 1, 'b' : 2, 'c' : 3, 'd' : 4, 'e' : 5 } ]

json = demjson.encode(data)
print json
```

➢ **Decoding JSON in Python (decode):**

– Python can use demjson.decode() function for decoding JSON. This function returns the value decoded from json to an appropriate Python type.

Syntax

demjson.decode(self, txt)

Example

```
import demjson

json = '{"a":1,"b":2,"c":3,"d":4,"e":5}';

text = demjson.decode(json)
print  text
```

---

e. What is JSON Grammar. Explain **(5)**

Ans. JSON syntax is basically considered as a subset of JavaScript syntax; it includes the following:

- Data is represented in name/value pairs.
- Curly braces hold objects and each name is followed by ':'(colon), the name/value pairs are separated by , (comma).
- Square brackets hold arrays and values are separated by ,(comma).

```
{
        "book":
          [
         {
           "id": "01",
           "language": "Java",
           "edition": "third",
```

```
    "author": "Herbert Schildt"
  },

  {
    "id": "07",
    "language": "C++",
    "edition": "second",
    "author": "E.Balagurusamy"
  }
    ]
}
```

JSON supports the following two data structures −

- **Collection of name/value pairs** − This Data Structure is supported by different programming languages.
- **Ordered list of values** − It includes array, list, vector or sequence etc.

– The implementations of the two structures are represented in the forms of the object and array.

Crockford outlines the two structural representations of JSON through a series of syntax diagrams

---

f. Write a short note on Persisting JSON. (5)

Ans.

– Over the years, many a developer has needed to be able to string together the isolated requests of a common server, in order to facilitate things such as shopping carts for e-commerce.
– One of the technologies that was forged from this requirement brought forth a technique that we will leverage in order to achieve the persistence of JSON. That technology is the HTTP cookie.

– The HTTP cookie, or cookie for short, was created as a means to string together the actions taken by the user per “isolated” request and provide a convenient way to persist the state of one page into that of another.
– The cookie is simply a chunk of data that the browser has been notified to retain. Furthermore, the browser will have to supply, per subsequent request, the retained cookie to the server for the domain that set it, thereby providing state to a stateless protocol.
– The cookie can be utilized on the client side of an application with JavaScript.
– Additionally, it is available to the server, supplied within the header of each request made by the browser.
– The header can be parsed for any cookies and made available to server-side code. Cookies provide both front-end and back-end technologies the ability to collaborate and reflect the captured state, in order to properly handle each page view or request accordingly.
– The ability to continue to progress the state from one page to another allows each action to no longer be isolated and, instead, occur within the entirety of the user’s interaction with a web site.

Syntax:

– The cookie is simply a string of ASCII encoded characters composed of one or more attribute-value pairs, separated by a semicolon (;) token.
– Key/Value Pairs Intended to Be Persisted As a Cookie Must Both Be Valid ASCII Characters.
