paulfranco / **MongoDB Interview Questions.md**

Last active 2 months ago



MongoDB Interview Questions.md

# Interview Questions

## MongoDB

### Q1: Explain what is MongoDB? ☆

**Answer:** MongoDB is an open-source document database that provides high performance, high availability, and automatic scaling. It's Key Features are:

- Document Oriented and NoSQL database.
- Supports Aggregation
- Uses BSON format
- Sharding (Helps in Horizontal Scalability)
- Supports Ad Hoc Queries
- Schema Less
- Capped Collection
- Indexing (Any field in MongoDB can be indexed)
- MongoDB Replica Set (Provides high availability)
- Supports Multiple Storage Engines

**Source:** *mongodb.com*

### Q2: What is “Namespace” in MongoDB? ☆☆

**Answer:** MongoDB stores BSON (Binary Interchange and Structure Object Notation) objects in the collection. The concatenation of the collection name and database name is called a namespace

**Source:** *medium.com/@hub4tech*

### Q3: What do you understand by NoSQL databases? Explain. ☆☆☆

**Answer:** At the present time, the internet is loaded with big data, big users, big complexity etc. and also becoming more complex day by day. NoSQL is answer of all these problems; It is not a traditional database management system, not even a relational database management system (RDBMS). NoSQL stands for “Not Only SQL”. NoSQL is a type of database that can handle and sort all type of unstructured, messy and complicated data. It is just a new way to think about the database.

**Source:** *medium.com/@hub4tech*

### Q4: What is the difference between MongoDB and MySQL? ☆☆☆

**Answer:** Although MongoDB and MySQL both are free and open source databases, there is a lot of difference between them in the term of data representation, relationship, transaction, querying data, schema design and definition, performance speed, normalization and many more. To compare MySQL with MongoDB is like a comparison between Relational and Non-relational databases.

**Source:** *medium.com/@hub4tech*

### Q5: What is the difference b/w MongoDB and CouchDB? ☆☆☆

**Answer:** MongoDB and CouchDB both are the great example of open source NoSQL database. Both are document oriented databases. Although both stores data but there is a lot of difference between them in terms of implementation of their data models, interfaces, object storage and replication methods etc.

**Source:** *medium.com/@hub4tech*

### Q6: Why does Profiler use in MongoDB? ☆☆

**Answer:** MongoDB uses a database profiler to perform characteristics of each operation against the database. You can use a profiler to find queries and write operations

**Source:** *medium.com/@hub4tech*

### Q7: If you remove an object attribute, is it deleted from the database? ☆☆

**Answer:** Yes, it be. Remove the attribute and then re-save () the object.

**Source:** *medium.com/@hub4tech*

### Q8: Does MongoDB need a lot space of Random Access Memory (RAM)? ☆☆

**Answer:** No. MongoDB can be run on small free space of RAM.

**Source:** *medium.com/@hub4tech*

### Q9: What is a replica set? ☆☆

**Answer:** It is a group of mongo instances that maintain same data set. Replica sets provide redundancy and high availability, and are the basis for all production deployments.

**Source:** *interviewbubble.com*

### Q10: Does Mongodb Support Foreign Key Constraints? ☆

**Answer:** No. MongoDB does not support such relationships. The database does not apply any constraints to the system (i.e.: foreign key constraints), so there are no "cascading deletes" or "cascading updates". Basically, in a NoSQL database it is up to you to decide how to organise the data and its relations if there are any.

**Source:** *interviewbubble.com*

### Q11: What Is Replication In MongoDB? ☆☆

**Answer: Replication** is the process of synchronizing data across multiple servers. Replication provides redundancy and increases data availability. With multiple copies of data on different database servers, replication protects a database from the loss of a single server. Replication also allows you to recover from hardware failure and service interruptions.

**Source:** *interviewbubble.com*

### Q12: Which are the most important features of MongoDB? ☆

**Answer:**

- Flexible data model in form of documents

- Agile and highly scalable database
- Faster than traditional databases
- Expressive query language

**Source:** *tutorialspoint.com*

### Q13: Compare SQL databases and MongoDB at a high level. ☆☆

**Answer:** SQL databases store data in form of tables, rows, columns and records. This data is stored in a pre-defined data model which is not very much flexible for today's real-world highly growing applications. MongoDB in contrast uses a flexible structure which can be easily modified and extended.

**Source:** *tutorialspoint.com*

### Q14: How is data stored in MongoDB? ☆☆

**Answer:** Data in MongoDB is stored in BSON documents – JSON-style data structures. Documents contain one or more fields, and each field contains a value of a specific data type, including arrays, binary data and sub-documents. Documents that tend to share a similar structure are organized as collections. It may be helpful to think of documents as analogous to rows in a relational database, fields as similar to columns, and collections as similar to tables.

The advantages of using documents are:

- Documents (i.e. objects) correspond to native data types in many programming languages.
- Embedded documents and arrays reduce need for expensive joins.
- Dynamic schema supports fluent polymorphism.

**Source:** *mongodb.com*

### Q15: Mention the command to insert a document in a database called school and collection called persons. ☆☆

**Answer:**

```
use school;
db.persons.insert( { name: "kadhir", dept: "CSE" } )
```

**Source:** *tutorialspoint.com*

### Q16: What are Indexes in MongoDB? ☆☆

**Answer:** Indexes support the efficient execution of queries in MongoDB. Without indexes, MongoDB must perform a collection scan, i.e. scan every document in a collection, to select those documents that match the query statement. If an appropriate index exists for a query, MongoDB can use the index to limit the number of documents it must inspect.

**Source:** *tutorialspoint.com*

### Q17: How many indexes does MongoDB create by default for a new collection? ☆

**Answer:** By default, MongoDB created the _id collection for every collection.

**Source:** *tutorialspoint.com*

### Q18: Can you create an index on an array field in MongoDB? If yes, what happens in this case? ☆☆

**Answer:** Yes. An array field can be indexed in MongoDB. In this case, MongoDB would index each value of the array so you can query for individual items:

```
> db.col1.save({'colors': ['red','blue']})
> db.col1.ensureIndex({'colors':1})

> db.col1.find({'colors': 'red'})
{ "_id" : ObjectId("4ccc78f97cf9bdc2a2e54ee9"), "colors" : [ "red", "blue" ] }
> db.col1.find({'colors': 'blue'})
{ "_id" : ObjectId("4ccc78f97cf9bdc2a2e54ee9"), "colors" : [ "red", "blue" ] }
```

**Source:** *stackoverflow.com*

### Q19: When should we embed one document within another in MongoDB? ☆☆

**Answer:** You should consider embedding documents for:

- *contains* relationships between entities
- One-to-many relationships
- Performance reasons

**Source:** *tutorialspoint.com*

### Q20: What is BSON in MongoDB? ☆☆

**Answer: BSON** is a binary serialization format used to store documents and make remote procedure calls in MongoDB. BSON extends the JSON model to provide additional data types, ordered fields, and to be efficient for encoding and decoding within different languages.

**Source:** *mongodb.com*

### Q1: Explain the structure of ObjectID in MongoDB ☆☆☆

**Answer: ObjectIds** are small, likely unique, fast to generate, and ordered. ObjectId values consist of 12 bytes, where the first four bytes are a timestamp that reflect the ObjectId's creation. Specifically:

- a 4-byte value representing the seconds since the Unix epoch,
- a 5-byte random value, and
- a 3-byte counter, starting with a random value. In MongoDB, each document stored in a collection requires a unique _id field that acts as a primary key. If an inserted document omits the _id field, the MongoDB driver automatically generates an ObjectId for the _id field.

**Source:** *mongodb.com*

### Q2: What is sharding? ☆☆☆

**Answer:** Sharding means to store the data on the multiple machines.

**Source:** *interviewbubble.com*

### Q3: What are NoSQL databases? What are the different types of NoSQL databases? ☆☆☆

**Answer:** A NoSQL database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases (like SQL, Oracle, etc.).

Types of NoSQL databases:

- Document Oriented
- Key Value
- Graph
- Column Oriented

**Source:** *interviewbubble.com*

### Q4: How is MongoDB better than other SQL databases? ☆☆☆

**Answer:** MongoDB allows a highly flexible and scalable document structure. For e.g. one data document in MongoDB can have five columns and the other one in the same collection can have ten columns. Also, MongoDB database are faster as compared to SQL databases due to efficient indexing and storage techniques.

**Source:** *tutorialspoint.com*

### Q5: What does MongoDB not being ACID compliant really mean? ☆☆☆

**Answer:** It's actually not correct that MongoDB is not ACID-compliant. On the contrary, MongoDB is ACID-compilant at the document level.

Any update to a single document is

- **Atomic**: it either fully completes or it does not
- **Consistent**: no reader will see a "partially applied" update
- **Isolated**: again, no reader will see a "dirty" read
- **Durable**: (with the appropriate write concern)

What MongoDB doesn't have is *transactions* - that is, multiple-document updates that can be rolled back and are ACID-compliant.

Multi-document transactions, scheduled for MongoDB 4.0 in Summer 2018*, will feel just like the transactions developers are familiar with from relational databases – multi-statement, similar syntax, and easy to add to any application.

**Source:** *stackoverflow.com*

### Q6: How can you achieve primary key - foreign key relationships in MongoDB? ☆☆☆

**Answer:** By default MongoDB does not support such primary key - foreign key relationships. However, we can achieve this concept by embedding one document inside another (aka subdocuments). Foe e.g. an address document can be embedded inside customer document.

**Source:** *tutorialspoint.com*

### Q7: Does MongoDB pushes the writes to disk immediately or lazily? ☆☆☆

**Answer:** MongoDB pushes the data to disk lazily. It updates the immediately written to the journal but writing the data from journal to disk happens lazily.

**Source:** *tutorialspoint.com*

### Q8: If you remove a document from database, does MongoDB remove it from disk? ☆☆☆

**Answer:** Yes. Removing a document from database removes it from disk too.

**Source:** *tutorialspoint.com*

### Q9: What is a covered query in MongoDB? ☆☆☆

**Answer:** A covered query is the one in which:

- fields used in the query are part of an index used in the query, and
- the fields returned in the results are in the same index

**Source:** *tutorialspoint.com*

### Q10: How can you achieve transaction and locking in MongoDB? ☆☆☆

**Answer:** To achieve concepts of transaction and locking in MongoDB, we can use the nesting of documents, also called embedded (or sub) documents. MongoDB supports atomic operations within a single document.

**Source:** *tutorialspoint.com*

### Q11: What is Aggregation in MongoDB? ☆☆☆

**Answer:** *Aggregations* operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. MongoDB provides three ways to perform aggregation:

- the aggregation pipeline,
- the map-reduce function,
- and single purpose aggregation methods and commands.

**Source:** *tutorialspoint.com*

### Q12: What is Sharding in MongoDB? Explain. ☆☆☆

**Answer:** *Sharding* is a method for storing data across multiple machines. MongoDB uses sharding to support deployments with very large data sets and high throughput operations.

**Source:** *tutorialspoint.com*

### Q13: Why are MongoDB data files large in size? ☆☆☆

**Answer:** MongoDB preallocates data files to reserve space and avoid file system fragmentation when you setup the server.

**Source:** *tutorialspoint.com*

### Q14: Why MongoDB is not preferred over a 32-bit system? ☆☆☆

**Answer:** When running a 32-bit build of MongoDB, the total storage size for the server, including data and indexes, is 2 gigabytes. For this reason, do not deploy MongoDB to production on 32-bit machines.

If you're running a 64-bit build of MongoDB, there's virtually no limit to storage size.

**Source:** *tutorialspoint.com*

### Q15: Mention the command to check whether you are on the master server or not. ☆☆☆

**Answer:**

```
db.isMaster()
```

**Source:** *tutorialspoint.com*

### Q16: Can one MongoDB operation lock more than one databases? If yes, how? ☆☆☆

**Answer:** Yes. Operations like `copyDatabase()`, `repairDatabase()`, etc. can lock more than one databases involved.

**Source:** *tutorialspoint.com*

### Q17: What is oplog? ☆☆☆

**Answer:** The *oplog* (operations log) is a special capped collection that keeps a rolling record of all operations that modify the data stored in your databases. MongoDB applies database operations on the primary and then records the operations on the primary's oplog. The secondary members then copy and apply these operations in an asynchronous process.

**Source:** *tutorialspoint.com*

### Q18: Is there an "upsert" option in the mongodb insert command? ☆☆☆

**Answer:** Since upsert is defined as operation that *creates a new document when no document matches the query criteria* there is no place for upserts in insert command. It is an option for the update command.

```
db.collection.update(query, update, {upsert: true})
```

**Source:** *stackoverflow.com*

### Q19: How to query MongoDB with "like"? ☆☆☆

**Answer:** I want to query something as SQL's like query:

```
select *
from users
where name like '%m%'
```

How to do the same in MongoDB?

**Answer**

```
db.users.find({"name": /.*m.*/})
// or
db.users.find({"name": /m/})
```

You're looking for something that contains "m" somewhere (SQL's `'%'` operator is equivalent to Regexp's `'.*'`), not something that has "m" anchored to the beginning of the string.

**Source:** *stackoverflow.com*

### Q20: Find objects between two dates MongoDB ☆☆☆

**Answer:**

```
db.CollectionName.find({"whenCreated": {
    '$gte': ISODate("2018-03-06T13:10:40.294Z"),
    '$lt': ISODate("2018-05-06T13:10:40.294Z")
}});
```

**Source:** *stackoverflow.com*

### Q1: How replication works in MongoDB? ☆☆☆☆

**Answer:** A replica set consists of a primary node and a secondary node too. With the help of a replica set, all the data from primary node to the secondary node replicates. Replication is a process of synchronizing the data. Replication provides redundancy and it also increases the availability of data with the help of multiple copies of data on the different database server. It also protects the database from the loss of a single server.

**Source:** *interviewbubble.com*

### Q2: What are alternatives to MongoDB? ☆☆☆☆

**Answer:** Cassandra, CouchDB, Redis, Riak, Hbase are a few good alternatives.

**Source:** *interviewbubble.com*

### Q3: Does MongoDB support ACID transaction management and locking functionalities? ☆☆☆

**Answer:** ACID stands that any update is:

- **Atomic**: it either fully completes or it does not
- **Consistent**: no reader will see a "partially applied" update
- **Isolated**: no reader will see a "dirty" read
- **Durable**: (with the appropriate write concern)

Historically MongoDB does not support default multi-document ACID transactions (multiple-document updates that can be rolled back and are ACID-compliant). However, MongoDB provides atomic operation on a single document. MongoDB 4.0 **will add support for multi-document transactions**, making it the only database to combine the speed, flexibility, and power of the document model with ACID data integrity guarantees.

**Source:** *tutorialspoint.com*

### Q4: Where can I run MongoDB? ☆☆☆☆

**Answer:** MongoDB can be run anywhere, providing you complete freedom from platform lock-in.

MongoDB Atlas provides you with a complete, pay-as-you-go fully managed service for MongoDB in AWS, Azure, and Google Cloud Platform.

You can download MongoDB and run it yourself anywhere. MongoDB Ops Manager is the best way to run MongoDB on your own infrastructure – whether that lives on-premise or in a public cloud – making it fast and easy for operations teams to deploy, monitor, backup and scale MongoDB. Many of the capabilities of Ops Manager are also available in the MongoDB Cloud Manager service.

**Source:** *mongodb.com*

### Q5: How does MongoDB ensure high availability? ☆☆☆☆

**Answer:** MongoDB automatically maintains replica sets, multiple copies of data that are distributed across servers, racks and data centers. Replica sets help prevent database downtime using native replication and automatic failover.

A replica set consists of multiple replica set members. At any given time one member acts as the primary member, and the other members act as secondary members. If the primary member fails for any reason (e.g., hardware failure), one of the secondary members is automatically elected to primary and begins to process all reads and writes.

**Source:** *mongodb.com*

### Q6: Is MongoDB schema-less? ☆☆☆☆

**Answer:** No. In MongoDB schema design is still important. MongoDB's document model does, however, employ a different schema paradigm than traditional relational databases.

In MongoDB, documents are self-describing; there is no central catalog where schemas are declared and maintained. The schema can vary across documents, and the schema can evolve quickly without requiring the modification of existing data.

MongoDB's dynamic schema also makes it easier to represent semi-structured and polymorphic data, as documents do not all need to have exactly the same fields. For example, a collection of financial trading positions might have positions that are equity positions, and some that are bonds, and some that are cash. All may have some fields in common, but some fields ('ticker', “number_of_shares”) do not apply to all position types.

**Source:** *mongodb.com*

### Q7: Should I normalize my data before storing it in MongoDB? ☆☆☆

**Answer:** It depends from your goals. Normalization will provide an *update efficient data representation*. Denormalization will make data *reading efficient*.

In general, use embedded data models (denormalization) when:

- you have “contains” relationships between entities.
- you have one-to-many relationships between entities. In these relationships the “many” or child documents always appear with or are viewed in the context of the “one” or parent documents.

In general, use normalized data models:

- when embedding would result in duplication of data but would not provide sufficient read performance advantages to outweigh the implications of the duplication.
- to represent more complex many-to-many relationships.
- to model large hierarchical data sets.

Also normalizing your data like you would with a relational database is usually not a good idea in MongoDB. Normalization in relational databases is only feasible under the premise that JOINs between tables are relatively cheap. The $lookup aggregation operator provides some limited JOIN functionality, but it doesn't work with sharded collections. So joins often need to be emulated by the application through multiple subsequent database queries, which is very slow (see question MongoDB and JOINs for more information).

**Source:** *stackoverflow.com*

### Q8: Why is a covered query important? ☆☆☆☆

**Answer:** Since all the fields are covered in the index itself, MongoDB can match the query condition as well as return the result fields using the same index without looking inside the documents. Since indexes are stored in RAM or sequentially located on disk, such access is a lot faster.

**Source:** *tutorialspoint.com*

### Q9: Does MongoDB provide a facility to do text searches? How? ☆☆☆☆

**Answer:** Yes. MongoDB supports creating text indexes to support text search inside string content. This was a new feature which can introduced in version 2.6.

**Source:** *tutorialspoint.com*

### Q10: What happens if an index does not fit into RAM? ☆☆☆☆

**Answer:** If the indexes do not fit into RAM, MongoDB reads data from disk which is relatively very much slower than reading from RAM.

**Source:** *tutorialspoint.com*

### Q11: Mention the command to list all the indexes on a particular collection. ☆☆☆☆

**Answer:**

```
db.collection.getIndexes()
```

**Source:** *tutorialspoint.com*

### Q12: At what interval does MongoDB write updates to the disk? ☆☆☆☆

**Answer:** By default configuration, MongoDB writes updates to the disk every 60 seconds. However, this is configurable with the `commitIntervalMs` and `syncPeriodSecs` options.

**Source:** *tutorialspoint.com*

### Q13: What are Primary and Secondary Replica sets? ☆☆☆☆

**Answer:**

- **Primary** and master nodes are the nodes that can accept writes. MongoDB's replication is 'single-master:' only one node can accept write operations at a time.
- **Secondary** and slave nodes are read-only nodes that replicate from the primary.

**Source:** *tutorialspoint.com*

### Q14: By default, MongoDB writes and reads data from both primary and secondary replica sets. True or False. ☆☆☆☆

**Answer:** False. MongoDB writes data only to the primary replica set.

**Source:** *tutorialspoint.com*

### Q15: How does Journaling work in MongoDB? ☆☆☆☆

**Answer:** When running with *journaling*, MongoDB stores and applies write operations in memory and in the on-disk journal before the changes are present in the data files on disk. Writes to the journal are atomic, ensuring the consistency of the on-disk journal files. With journaling enabled, MongoDB creates a journal subdirectory within the directory defined by dbPath, which is /data/db by default.

**Source:** *tutorialspoint.com*

### Q16: How can I combine data from multiple collections into one collection? ☆☆☆

**Answer:** MongoDB 3.2 allows one to combine data from multiple collections into one through the `$lookup` aggregation stage.

```
db.books.aggregate([{
    $lookup: {
            from: "books_selling_data",
            localField: "isbn",
            foreignField: "isbn",
            as: "copies_sold"
        }
}])
```

**Source:** *stackoverflow.com*

### Q17: When to use MongoDB or other document oriented database systems? ☆☆☆

**Answer:** MongoDB is best suitable to store *unstructured data*. And this can organize your data into document format. These data stores don't enforce the ACID properties, and any schemas. This doesn't provide any transaction abilities. So this can scale big and we can achieve faster access (both read and write). If you wanted to work with structured data you can go ahead with RDBM.

**Source:** *stackoverflow.com*

### Q18: How do I perform the SQL JOIN equivalent in MongoDB? ☆☆☆

**Answer:** Mongo is not a relational database, and the devs are being careful to recommend specific use cases for $lookup, but at least as of 3.2 doing join is now possible with MongoDB. The new $lookup operator added to the aggregation pipeline is essentially identical to a left outer join:

```
{
   $lookup:
     {
       from: <collection to join>,
       localField: <field from the input documents>,
```

```
        foreignField: <field from the documents of the "from" collection>,
        as: <output array field>
      }
}
```

**Source:** *stackoverflow.com*

### Q19: How to query MongoDB with %like%? ☆☆☆

**Details:** I want to query something as SQL's like query:

```
select *
from users
where name like '%m%'
```

How to do the same in MongoDB?

**Answer:**

```
db.users.find({name: /a/})  //like '%a%'
db.users.find({name: /^pa/}) //like 'pa%'
db.users.find({name: /ro$/}) //like '%ro'
```

Or using Mongoose:

```
db.users.find({'name': {'$regex': 'sometext'}})
```

**Source:** *stackoverflow.com*

### Q60: What is use of capped collection in MongoDB? ☆☆☆

**Answer: Capped collections** allow you to define a fix length/size collection. After the size/no of documents have been reached it will override the oldest document to accommodate new documents. It is like a circular buffer. Capped collections support high-throughput operations that insert and retrieve documents based on insertion order.

Consider the following potential use cases for capped collections:

- Store log information generated by high-volume systems. Inserting documents in a capped collection without an index is close to the speed of writing log information directly to a file system.
- Cache small amounts of data in a capped collections.

**Source:** *stackoverflow.com*

### Q1: What are the differences between MongoDB and MySQL? ☆☆☆☆☆

**Answer:** The Major Differences between MongoDB and MySQL are:

1. There is a difference in the representation of data in the two databases. In MongoDB, data represents in a collection of JSON documents while in MySQL, data is in tables and rows. JSON documents can compare to associative arrays when using PHP and directory objects when using Python.
2. When it comes to querying, you have to put a string in the query language that the DB system parses. The query language is called Structured Query Language, or SQL,from where MySQL gets its name. This exposes your DB susceptible to SQL injectionattacks. On the other hand, MongoDB's querying is object-oriented, which means you pass MongoDB a document explaining what you are querying. There is no parsing whatsoever, which will take some time getting used to if you already use SQL.
3. One of the greatest benefits of relational databases like MySQL is the JOIN operation. The operation allows for the querying across several tables. Although MongoDB doesn't support joints, it supports multi-dimensional data types like other documents and arrays.

4. With MySQL, you can have one document inside another (embedding). You would have to create one table for comments and another for posts if you are using MySQL to create a blog. In MongoDB, you will only have one array of comments and one collection of posts within a post.
5. MySQL supports atomic transactions. You can have several operations within a transaction and you can roll back as if you have a single operation. There is no support for transactions in MongoDB and the single operation is atomic.
6. One of the best things about MongoDB is that you are not responsible for defining the schema. All you need to do is drop in documents. Any 2 documents in a collection need not be in the same field. You have to define the tables and columns before storage in MySQL. All rows in a table share the same columns.
7. MongoDB's performance is better than that of MySQL and other relational DBs. This is because MongoDB sacrifices JOINS and other things and has excellent performance analysis tools. Note that you still have to index the data and the data in most applications is not enough for them to see a difference. MySQL is criticized for poor performance, especially in ORM application. However, you are unlikely to have an issue if you do proper data indexing and you are using a database wrapper.
8. One advantage of MySQL over NoSQL like MongoDB is that the community in MySQL is much better than NoSQL. This is mostly because NoSQL is relatively new while MySQL has been around for several years.
9. There are no reporting tools with MongoDB, meaning performance testing and analysis is not always possible. With MySQL, you can get several reporting tools that help you rove the validity of your applications.
10. RDBSs function on a paradigm called ACID, which is an acronym for (Atomicity, Consistency, Isolation, and Durability). This is not present in MongoDB database.
11. MongoDB has a Map Reduce feature that allows for easier scalability. This means you can get the full functionality of MongoDB database even if you are using low-cost hardware.
12. You do not have to come up with a detailed DB model with MongoDB because of is non-relational. A DB architect can quickly create a DB without a fine-grained DB model, thereby saving on development time and cost.

**Source:** *analyticbridge.datasciencecentral.com*

### Q2: What's the advantage of the backup features in Ops Manager versus traditional backup strategies? ☆☆☆☆☆

**Answer:** Ops Manager offers a lot of advantages, including:

- Point-in-Time-Recovery, Scheduled Backups. You can generate a restore image from an exact time in the past, which can be very helpful for restoring operations just prior to a catastrophic event.
- Continuous, Incremental Backup. Data is backed up continuously as the cluster updates. Your backups are always up-to-date.
- Sharded Cluster Backup. Backing up sharded clusters can be hard. Ops Manager Backup automates this for you.
- Queryable Backup. Query backups directly without having to restore them. Find the backup you need or examine how data has changed over time.

Ops Manager is included with MongoDB Enterprise Advanced, and provides continuous backup and point-in-time restore for MongoDB. Those interested in a cloud-based backup solution should consider MongoDB Cloud Manager, which provides continuous, online backup and point-in-time restore for MongoDB as a fully managed service.

**Source:** *mongodb.com*

### Q3: What is a Storage Engine in MongoDB ☆☆☆☆☆

**Answer:** A *storage engine* is the part of a database that is responsible for managing how data is stored on disk. For example, one storage engine might offer better performance for read-heavy workloads, and another might support a higher-throughput for write operations.

**Source:** *tutorialspoint.com*

### Q4: Which are the two storage engines used by MongoDB? ☆☆☆☆☆

**Answer:** MongoDB uses MMAPv1 and WiredTiger.

**Source:** *tutorialspoint.com*

### Q5: How does MongoDB provide concurrency? ☆☆☆☆

**Answer:** MongoDB uses reader-writer locks that allow concurrent readers shared access to a resource, such as a database or collection, but give exclusive access to a single write operation.

**Source:** *tutorialspoint.com*

### Q6: How can you isolate your cursors from intervening with the write operations? ☆☆☆☆

**Answer:** You can use the `snapshot()` method on a cursor to isolate the operation for a very specific case. `snapshot()` traverses the index on the `_id` field and guarantees that the query will return each document no more than once.

**Source:** *tutorialspoint.com*

### Q7: What is splitting in mongodb? ☆☆☆☆☆

**Answer:** It is a background process that is used to keep chunks from growing too large.

**Source:** *intellipaat.com*

### Q8: Update MongoDB field using value of another field ☆☆☆☆

**Answer:** Consider SQL command:

```
UPDATE Person SET Name = FirstName + ' ' + LastName
```

In Mongo, is it possible to update the value of a field using the value from another field?

**Answer**

You cannot refer to the document itself in an update (yet). You'll need to iterate through the documents and update each document using a function like:

```
db.person.find().snapshot().forEach(
    function (elem) {
        db.person.update(
            {
                _id: elem._id
            },
            {
                $set: {
                    name: elem.firstname + ' ' + elem.lastname
                }
            }
        );
    }
);
```

**Source:** *stackoverflow.com*

### Q9: How to remove a field completely from a MongoDB document? ☆☆☆☆

**Answer:** Suppose this is a document.

```
{
    name: 'book',
    tags: {
        words: ['abc','123'], // <-- remove it comletely
        lat: 33,
        long: 22
    }
}
```

How do I remove `words` completely from all the documents in this collection?

**Answer**

```
db.example.update({}, {$unset: {words:1}}, false, true);
```

**Source:** *stackoverflow.com*

### Q10: How to condense large volumes of data in Mongo? ☆☆☆☆☆

**Answer:** *Map-reduce* is a data processing paradigm for condensing large volumes of data into useful aggregated results. For map-reduce operations, MongoDB provides the `mapReduce` database command.

**Source:** *docs.mongodb.com*

### Q11: What are three primary concerns when choosing a data management system? ☆☆☆☆☆

**Answer:** There are three primary concerns you must balance when choosing a data management system: consistency, availability, and partition tolerance.

- **Consistency** - means that each client always has the same view of the data.
- **Availability** - means that all clients can always read and write.
- **Partition tolerance** - means that the system works well across physical network partitions.

According to the CAP Theorem, you can only pick two.

In addition to CAP configurations, another significant way data management systems vary is by the data model they use: relational, key-value, column-oriented, or document-oriented (there are others, but these are the main ones).

- *Relational* systems are the databases we've been using for a while now. RDBMSs and systems that support ACIDity and joins are considered relational.
- *Key-value* systems basically support get, put, and delete operations based on a primary key.
- *Column-oriented* systems still use tables but have no joins (joins must be handled within your application). Obviously, they store data by column as opposed to traditional row-oriented databases. This makes aggregations much easier.
- *Document-oriented* systems store structured "documents" such as JSON or XML but have no joins (joins must be handled within your application). It's very easy to map data from object-oriented software to these systems.

**Source:** *stackoverflow.com*

### Q12: When to Redis or MongoDB? ☆☆☆☆

**Answer:**

- **Use MongoDB if you don't know yet how you're going to query your data or what schema to stick with.** MongoDB is suited for Hackathons, startups or every time you don't know how you'll query the data you inserted. MongoDB does not make any assumptions on your underlying schema. While MongoDB is schemaless and non-relational, this does not mean that there is no schema at all. It simply means that your schema needs to be defined in your app (e.g. using Mongoose). Besides that, MongoDB is great for prototyping or trying things out. Its performance is not that great and can't be compared to Redis.
- **Use Redis in order to speed up your existing application.** It is very uncommon to use Redis as a standalone database system (some people prefer referring to it as a "key-value"-store).

**Source:** *stackoverflow.com*

### Q13: MongoDB relationships. What to use - embed or reference? ☆☆☆☆

**Details:** I want to design a question structure with some comments, but I don't know which relationship to use for comments: embed or reference? Explain me pros and cons of both solutions?

**Answer:** In general,

- **embed** is good if you have one-to-one or one-to-many relationships between entities, and
- **reference** is good if you have many-to-many relationships.

Also consider as a general rule, if you have a lot of [child documents] or if they are large, a separate collection might be best. Smaller and/or fewer documents tend to be a natural fit for embedding.

**Source:** *stackoverflow.com*

### Q14: How to get the last N records from find? ☆☆☆☆

**Answer:** Use `sort` and `limit` in chain:

```
db.foo.find().sort({_id:1}).limit(50);
```

**Source:** *stackoverflow.com*

### Q15: How to find MongoDB records where array field is not empty? ☆☆☆☆

**Answer:** Consider some variants:

```
collection.find({ pictures: { $exists: true, $not: { $size: 0 } } })
collection.find({ pictures: { $exists: true, $ne: [] } })
collection.find({ pictures: { $gt: [] } }) // since MongoDB 2.6
collection.find({'pictures.0': {$exists: true}}); // beware if performance is important - it'll do a fu
```

**Source:** *stackoverflow.com*

### Q16: How to find document with array that contains a specific value? ☆☆☆☆☆

**Details:** You have this schema:

```
person = {
    name : String,
    favoriteFoods : Array
}
```

where the `favoriteFoods` array is populated with strings. How can I find all persons that have `sushi` as their favorite food using MongoDB?

**Answer:** Consider:

```
PersonModel.find({ favouriteFoods: "sushi" }, ...);
PersonModel.find({ favouriteFoods: { "$in" : ["sushi"]} }, ...);
```

**Source:** *docs.mongodb.com*

### Q17: How to check if a field contains a substring? ☆☆☆☆

**Answer:** Consider:

```
db.users.findOne({"username" : {$regex : ".*subsctring.*"}});
```

**Source:** *stackoverflow.com*

### Q18: Is it possible to update MongoDB field using value of another field? ☆☆☆☆☆

**Details:** In SQL we will use:

```
UPDATE Person SET Name = FirstName + ' ' + LastName
```

Is it possible with MongoDB?

**Answer:** You cannot refer to the document itself in an update (yet). You'll need to iterate through the documents and update each document using a function. So consider:

```
db.person.find().snapshot().forEach(
  function(elem) {
    db.person.update({
      _id: elem._id
    }, {
      $set: {
        name: elem.firstname + ' ' + elem.lastname
      }
    });
  }
);
```

**Source:** *stackoverflow.com*

### Q19: Explain what is horizontal scalability? ☆☆☆☆☆

**Answer: Horizontal scalability** is the ability to increase capacity by connecting multiple hardware or software entities so that they work as a single logical unit. MongoDB provides horizontal scalability as part of its core functionality.

**Source:** *searchcio.techtarget.com*