MongoDB Interview Questions and Answers From Basic To Advanced Level

. What are some of the key features of MongoDB?

- MongoDB is a document-oriented database that allows for flexible and dynamic schema design.
- It supports a wide range of data types, including arrays, embedded documents, and geospatial data.
- MongoDB provides horizontal scaling through sharding and automatic failover through replica sets.
- It has powerful query and aggregation capabilities, with support for indexing and text search.
- MongoDB supports ACID transactions with multi-document transactions in version 4.0 and later.

Q 2. Can you explain the difference between a replica set and a sharded cluster in MongoDB?

- A replica set is a group of MongoDB instances that replicate data across multiple servers for redundancy and high availability.
- A sharded cluster is a collection of replica sets that distribute data across multiple servers for horizontal scaling.

Q 3. What is the purpose of the WiredTiger storage engine in MongoDB?

- WiredTiger is the default storage engine for MongoDB as of version 3.2.
- It provides efficient compression, allowing for greater data density and reduced disk space requirements.
- It also supports document-level concurrency control, allowing for multiple reads and writes to occur simultaneously.

Q 4. How do you monitor the performance of a MongoDB database?

- MongoDB provides several tools for monitoring performance, including the mongostat and mongotop command-line utilities.
- MongoDB also supports integration with third-party monitoring tools like Datadog and New Relic.
- Important metrics to monitor include CPU usage, memory usage, and disk I/O.

Q 5. What is indexing in MongoDB? How can you create an index?

- Indexing is the process of creating data structures that allow for efficient retrieval of data based on certain criteria.
- In MongoDB, indexes can be created on any field or combination of fields in a collection using the createIndex() method.
- Indexes can be unique or non-unique, and can be created as ascending or descending.

Q 6. Can you explain the difference between a primary key and a secondary key in MongoDB?

- In MongoDB, the primary key is automatically created on the _id field, which is unique to each document in a collection.
- Secondary keys can be created on any field or combination of fields in a collection, and are used to optimize queries that filter, sort, or aggregate data.

Q 7. What are some of the best practices for backup and recovery in MongoDB?

- MongoDB recommends taking regular backups and storing them off-site.
- Backups can be taken using the mongodump or mongorestore utilities, or by using a third-party backup tool.
- It's also important to test backups regularly to ensure they can be restored successfully.

Q 8. How do you handle concurrency in MongoDB?

- MongoDB uses a document-level locking mechanism to handle concurrency.

- This means that only the document being modified is locked, allowing for multiple reads and writes to occur simultaneously.
- MongoDB also supports optimistic locking through the use of the $set operator with the $inc modifier.

Q 9. Can you explain the difference between a collection and a document in MongoDB?

- A collection is a group of related documents in MongoDB.
- Each document in a collection represents a single instance of a data object, and can have its own unique structure.

Q 10. What are some of the security features available in MongoDB?

- MongoDB supports role-based access control (RBAC), which allows for granular control over user privileges.
- It also supports SSL/TLS encryption for data in transit, as well as authentication mechanisms like Kerberos and LDAP.
- MongoDB Enterprise Edition includes additional security features like field-level encryption and auditing.

MongoDB Query Interview Questions 2023

MongoDB Query is a way to retrieve data from a MongoDB database by specifying certain criteria or conditions that the data must meet. MongoDB is a NoSQL document-oriented database, which means that data is stored in flexible, semi-structured documents in JSON-like format. Below are questions related to MongoDB Query.

Q 11. What is a MongoDB query?

A MongoDB query is a request for data retrieval from a MongoDB database. It is a command or set of commands that retrieves data from one or more MongoDB collections based on specified criteria.

Q 12. What are the different types of MongoDB queries?

The different types of MongoDB queries are:

- find(): Retrieves documents from a collection based on specified criteria.
- aggregate(): Performs aggregation operations such as grouping, sorting, and filtering data.
- count(): Counts the number of documents in a collection that match specified criteria.
- distinct(): Returns a list of distinct values for a given field in a collection.
- text(): Performs a text search on a collection.
- geoNear(): Returns documents near a specified location in a collection.
- mapReduce(): Performs map-reduce operations on a collection.

Q 13. How do you perform a query in MongoDB?

To perform a query in MongoDB, you can use the find() method. For example, the following command retrieves all documents from a collection named "users":

db.users.find()

You can also pass a query document to the find() method to retrieve specific documents that match specified criteria. For example, the following command retrieves all documents from the "users" collection where the age is greater than or equal to 18:

db.users.find({ age: { $gte: 18 } })

Q 14. What is an index in MongoDB?

An index in MongoDB is a data structure that improves the speed of data retrieval operations by providing a quick lookup of documents based on specified criteria. MongoDB supports various types of indexes, including single-field indexes, compound indexes, and text indexes.

Q 15. How do you create an index in MongoDB?

You can create an index in MongoDB using the createIndex() method. For example, the following command creates a single-field index on the "name" field of a collection named "users":

db.users.createIndex({ name: 1 })

The value "1" specifies that the index is in ascending order. You can also create a compound index by specifying multiple fields in the index document. For example, the following command creates a compound index on the "name" and "age" fields:

db.users.createIndex({ name: 1, age: -1 })

The value "-1" specifies that the "age" field is in descending order.

Q 16. What is sharding in MongoDB?

Sharding in MongoDB is a process of partitioning data across multiple servers to improve the scalability and performance of the database system. It involves distributing data across multiple nodes in a cluster, where each node stores a subset of data.

Q 17. How do you perform a sharded query in MongoDB?

To perform a sharded query in MongoDB, you can use the shardCollection() method to enable sharding on a collection. For example, the following command enables sharding on a collection named "users" and shards it based on the "name" field:

sh.shardCollection("test.users", { name: 1 })

After sharding is enabled, you can perform a query on the collection using the standard MongoDB query syntax. The query will automatically be distributed across all shards in the cluster.

Q 18. What is the explain() method in MongoDB?

The explain() method in MongoDB is used to analyze the performance of a query and provide information on how the query is executed. It returns a document that contains details on the query execution plan, including the number of documents scanned, the index used, and the execution time.

Now it is time for some advanced mongoDB interview questions.

MongoDB Technical Interview Questions With Answers 2023

While appearing for MongoDB interview candidates can be asked questions from any of these topics - MongoDB basics, data modeling, indexing, aggregation, replication and sharding, security, performance tuning, integration, backup and recovery, scaling. Check top MongoDB Technical Interview Questions With Answers below.

Q 19. What are the advantages of using MongoDB?

MongoDB offers several advantages, including:

- Flexibility and scalability
- High performance and speed
- No need for a schema definition
- Easy integration with other databases
- Support for a wide variety of data types and structures

Q 20. What is sharding in MongoDB?

Sharding is a technique used in MongoDB to horizontally partition data across multiple servers or nodes in a cluster. This allows for high scalability and performance when working with large volumes of data.

Q 21. What is the difference between MongoDB and MySQL?

MongoDB is a NoSQL document-oriented database, while MySQL is a relational database management system. MongoDB does not require a predefined schema and stores data in flexible JSON-like documents, whereas MySQL requires a predefined schema and stores data in tables with fixed columns and rows.

Q 22. What is a replica set in MongoDB?

A replica set in MongoDB is a group of MongoDB servers that maintain the same data set. This provides high availability and fault tolerance by allowing automatic failover in the event of a primary node failure.

Q 23. How does MongoDB ensure high availability and fault tolerance?

MongoDB ensures high availability and fault tolerance through several mechanisms, including:

- Replica sets for automatic failover and data redundancy
- Sharding for horizontal scaling and load balancing
- Write concern and read concern settings for controlling data consistency and availability

Q 24. What is indexing in MongoDB?

Indexing in MongoDB is the process of creating a data structure that improves the efficiency of data retrieval operations. Indexes allow for faster query execution by reducing the number of documents that need to be scanned.

Q 25. What is the syntax for inserting a document in MongoDB?

To insert a document in MongoDB, use the insertOne() method with the following syntax:

db.collection.insertOne({<field1>: <value1>, <field2>: <value2>, ...})

Q 26. What is the syntax for updating a document in MongoDB?

To update a document in MongoDB, use the updateOne() or updateMany() method with the following syntax:

db.collection.updateOne({<filter>}, {<update>})
db.collection.updateMany({<filter>}, {<update>})

Q 27. What is the syntax for deleting a document in MongoDB?

To delete a document in MongoDB, use the deleteOne() or deleteMany() method with the following syntax:

db.collection.deleteOne({<filter>}) db.collection.deleteMany({<filter>})

Q 28. What is MongoDB?

MongoDB is a document-oriented NoSQL database that stores data in flexible JSON-like documents.

Q 29. What is a replica set?

A replica set is a group of MongoDB instances that maintain the same data set, providing redundancy and high availability.

Q 30. What is sharding in MongoDB?

Sharding is the process of distributing data across multiple machines to improve performance and scalability.

Q 31. What is the syntax to create a collection in MongoDB?

The syntax to create a collection in MongoDB is:

db.createCollection(name, options)

Q 32.What is the difference between MongoDB and MySQL?

MongoDB is a NoSQL document-oriented database while MySQL is a relational database.

Q 33.What are the advantages of MongoDB?

MongoDB offers flexible schema design, high availability and scalability, and fast query performance.

Q 34. What are the disadvantages of MongoDB?

MongoDB may have higher storage requirements due to the flexible schema design, and may require more RAM compared to traditional databases.

Q 35. What is an index in MongoDB?

An index is a data structure that helps MongoDB efficiently retrieve data.

Q 36. What are the different types of indexes in MongoDB?

The different types of indexes in MongoDB include single-field indexes, compound indexes, multikey indexes, and text indexes.

Q 37. What is the syntax to create an index in MongoDB?

The syntax to create an index in MongoDB is:

db.collection.createIndex(keys, options)

Q 38. What is aggregation in MongoDB?

Aggregation is the process of grouping data together and performing calculations on groups of data.

Q 39. What is the syntax to perform aggregation in MongoDB?

The syntax to perform aggregation in MongoDB is:

db.collection.aggregate(pipeline)

Q 40. What is a MongoDB namespace?

A MongoDB namespace is the combination of a database name and collection name.

Q 41. What is a covered query in MongoDB?

A covered query is a query where all of the fields required to satisfy the query are contained within an index.

Q 42. What is a capped collection in MongoDB?

A capped collection is a fixed-size collection that automatically overwrites the oldest documents when the collection reaches its maximum size.

Q 43. What is the syntax to create a capped collection in MongoDB?

The syntax to create a capped collection in MongoDB is:

db.createCollection(name, { capped: true, size: max_size })

Q 44.What is the difference between findOne() and find() in MongoDB?

findOne() returns the first document that matches the query while find() returns a cursor to the documents that match the query.

Q 45.What is the purpose of the $text operator in MongoDB?

The $text operator is used to perform text search on MongoDB collections.

Q 46. What is the syntax to use the $text operator in MongoDB?

The syntax to use the $text operator in MongoDB is:

db.collection.find({ $text: { $search: "search term" } })

Q 47. What is the purpose of the $lookup operator in MongoDB?

The $lookup operator is used to perform a left outer join between two collections in MongoDB.

Q 48. What is the difference between updateOne() and updateMany() in MongoDB?

updateOne() updates the first document that matches the query, while updateMany() updates all documents that match the query.

Q 49. What is a TTL index in MongoDB and how is it used?

A TTL (time to live) index in MongoDB automatically removes documents from a collection after a specified amount of time has passed since the indexed field's value.

Q 50. How does MongoDB handle transactions?

MongoDB supports multi-document transactions in versions 4.0 and later, allowing developers to atomically write to multiple collections or shards.

Q 51. What is the use of explain() method in MongoDB?

The explain() method in MongoDB provides information on how a query is executed, including which indexes are used and how many documents are examined.

Q 52. What is a GridFS in MongoDB?

GridFS is a specification for storing and retrieving large files, such as images or videos, in MongoDB.

Q 53. What is the purpose of the $group operator in MongoDB?

The $group operator in MongoDB is used to group documents by a specified field and perform aggregation operations on the grouped data, such as summing or averaging.

Q 54. What is a geospatial index in MongoDB?

A geospatial index in MongoDB is used to efficiently query and manipulate data that represents locations on the earth's surface, such as latitude and longitude coordinates.

Q 55. What is the difference between aggregation pipeline and MapReduce in MongoDB?

Aggregation pipeline in MongoDB is used to perform aggregation operations on data, while MapReduce is used for more complex data processing operations.

Q 56. How does MongoDB handle data consistency?

MongoDB uses a write-ahead log (WAL) and a process called journaling to ensure data consistency in the event of a crash or system failure.

Q 57. What is the purpose of the $push operator in MongoDB?

The $push operator in MongoDB is used to add an element to an array field in a document. It can also create an array field if it does not already exist.

MongoDB Interview Questions For Experienced 2023

Below we have shared MongoDB interview questions for 2 years experienced candidates.

Q 58. Define BSON.

BSON stands for Binary JSON, which is a binary-encoded serialization of JSON-like documents used in MongoDB. BSON extends the JSON model by adding additional data types and optimizing for efficient data storage and retrieval.

Q 59. How does MongoDB store the data?

MongoDB stores data in a binary format called BSON (Binary JSON) which represents documents as a sequence of fields and values. MongoDB uses a flexible schema model, which means that documents in a collection can have different fields and structures.

Q 60. Does MongoDB support ACID Transaction? Define ACID Transaction?

Yes, MongoDB supports ACID transactions. ACID stands for Atomicity, Consistency, Isolation, and Durability. ACID transactions are a set of

properties that guarantee that database transactions are processed reliably.

Q 61. Explain Composing elements or Structure of ObjectID in MongoDB?

ObjectID in MongoDB is a 12-byte unique identifier that consists of a timestamp, a machine identifier, a process identifier, and a random counter.

Q 62. How do we find array elements with multiple criteria?

We can use the $elemMatch operator in MongoDB to find array elements with multiple criteria. The $elemMatch operator matches documents that contain an array field with at least one element that matches all the specified criteria.

Q 63. How can we sort the user-defined function? For example, x and y are integers, and how do we calculate “x-y”?

We can use the sort() method in MongoDB to sort the user-defined function. For example, to calculate "x-y" and sort the result in ascending order, we can use the following code:

```
db.collection.find().sort(function(a, b) { return (a.x - a.y) - (b.x - b.y); })
```

Q 64. Upto what extent does the data expand to multi-slice?

In MongoDB, the maximum size of a single document is 16 megabytes. However, there is no limit to the number of slices (i.e., documents) that can be stored in a single collection or database.

Q 65. How do we retrieve MongoDB databases in Javascript Array?

We can retrieve MongoDB databases in a JavaScript array using the find() method. For example:

```
db.collection.find().toArray(function(err, result) { if (err) throw err; console.log(result); });
```

This code retrieves all the documents in the collection and returns them as an array.

Q 66. How do we update the object in the Nested Array?

We can update an object in a nested array using the $ positional operator in MongoDB. The $ operator identifies the element in the array that matches the query condition. For example:

db.collection.update({ "arrayField.objectField": "value" }, { $set: { "arrayField.$.objectField": "new_value" } })

This code updates the "objectField" value in the first element of the "arrayField" array that matches the query condition.

Q 67. How do we retrieve a particular embedded document in a MongoDB collection?

We can retrieve a particular embedded document in a MongoDB collection using the dot notation. For example:

db.collection.find({ "embeddedDocument.field": "value" })

This code retrieves all the documents that have an embedded document with a field that has the value "value".

Q 68. How do we query a nested Join?

In MongoDB, there is no support for nested joins. Instead, we can use the $lookup aggregation pipeline stage to perform a left outer join operation between two collections. The $lookup stage can be used to join two collections based on a common field, and it can also be used to join nested arrays.

For example, consider two collections: orders and items. Each document in the orders collection contains an array of item_ids that correspond to the _id field in the items collection. To join these collections based on the item_ids field, we can use the following aggregation pipeline:

db.orders.aggregate([{ $lookup: { from: "items", localField: "item_ids", foreignField: "_id", as: "items" } }])

This pipeline stage will perform a left outer join between the orders and items collections based on the item_ids field. The resulting documents will contain a new field called items, which will be an array of all the matching documents from the items collection.

Q 69. Can we run more than one Javascript Operation in one MongoDB instance?

Yes, MongoDB supports running multiple JavaScript operations in a single instance. MongoDB allows JavaScript to be executed on the server-side, using the eval method or by storing JavaScript functions as strings in the database.

For example, we can define a JavaScript function that performs some operations on a collection, and then execute that function using the eval method:

function updateCollection() { db.myCollection.updateMany({ status: "active" }, { $set: { status: "inactive" } }); db.myCollection.updateMany({ name: "John" }, { $set: { age: 30 } }); } db.eval(updateCollection);

This code will execute two update operations on the myCollection collection, setting the status of all documents with a status of "active" to "inactive" and setting the age of all documents with a name of "John" to 30. The eval method is used to execute the updateCollection function on the server-side.

Alternatively, we can store the JavaScript function as a string in the database, and then execute it using the eval method:

db.system.js.save({ _id: "updateCollection", value: function() { db.myCollection.updateMany({ status: "active" }, { $set: { status: "inactive" } }); db.myCollection.updateMany({ name: "John" }, { $set: { age: 30 } }); } }); db.eval("updateCollection()");

This code will save the updateCollection function as a string in the system.js collection, and then execute it using the eval method.

Other Important MongoDB Interview Questions and Answers

Below-mentioned interview questions on MongoDB might be asked in the interview.

Q 70. Can you differentiate between MongoDB and CouchDB?

MongoDB is a document-oriented NoSQL database that stores data in JSON-like documents and supports dynamic schema design, whereas CouchDB is a database that uses a semi-structured document-oriented approach and supports ACID transactions.

Q 71. What is a Capped Collection and how does it work in MongoDB?

A Capped Collection is a fixed-size collection in MongoDB that maintains insertion order, and if the collection reaches its maximum size, it overwrites the oldest documents. It works by automatically removing older documents as new ones are inserted.

Q 72. How can we perform Join operations in MongoDB?

MongoDB does not support traditional Join operations like SQL databases, but it provides an alternative using the $lookup aggregation pipeline stage to perform left outer joins between two collections based on a specified condition.

Q 73. What are the different storage engines used by MongoDB?

MongoDB currently has two storage engines: MMAPv1 and WiredTiger. MMAPv1 is the default storage engine for MongoDB versions before 3.2, while WiredTiger is the default storage engine for MongoDB versions 3.2 and later.

Q 74. How can we configure the cache size in MongoDB?

We can configure the cache size in MongoDB by setting the "storage.wiredTiger.engineConfig.cacheSizeGB" parameter in the configuration file or via the command line during startup.

Q 75. What are the methods to control MongoDB performance?

Some methods to control MongoDB performance include indexing, sharding, configuring the cache size, optimizing queries, and monitoring server metrics.

Q 76. What is MongoDB and its types?

MongoDB is a document-oriented NoSQL database that stores data in flexible JSON-like documents. MongoDB supports various data types such as string, integer, boolean, double, date, array, object, and binary data.

Q 77. What are the aggregate functions available in MongoDB?

Some aggregate functions available in MongoDB include $sum, $avg, $min, $max, $group, and $project.

Q 78. What are the CRUD operations in MongoDB?

CRUD stands for Create, Read, Update, and Delete operations. In MongoDB, the CRUD operations are performed using the insert(), find(), update(), and remove() methods, respectively.

Q 79. What are the datatypes supported by MongoDB?

MongoDB supports various data types such as string, integer, boolean, double, date, array, object, and binary data.