MongoDB Interview Question With Answer

January 31, 2024

What is MongoDB?

MongoDB is an open-source document-based database management tool that stores data in JSON-like formats. It is a highly scalable, flexible, and distributed NoSQL database.

What are Indexes in MongoDB?

In MongoDB, indexes are data structures that improve the speed of data retrieval operations by providing an efficient way to locate documents within a collection. They store a small portion of the data set in an easy-to-traverse form, allowing queries to quickly locate data without scanning every document in a collection.

How many indexes does MongoDB create by default for a new collection?

By default, MongoDB creates one index for a new collection, automatically linked to the `_id` field for efficient document retrieval.

What is "Namespace" in MongoDB?

In MongoDB, a namespace refers to the combination of a database name and a collection name, uniquely identifying a collection within a database.

What is Replication in MongoDB?

Replication in MongoDB is the process of synchronizing data across multiple servers to ensure redundancy, fault tolerance, and high availability. It involves copying data from a primary node to one or more secondary nodes, allowing for failover and data recovery in case of node failure.

How can you achieve primary key - foreign key relationships in MongoDB?

In MongoDB, you can create relationships by storing the ID of one document inside another, just like linking primary and foreign keys.

When should we embed one document within another in MongoDB?

Embedding documents in MongoDB is advantageous when the nested data is frequently accessed together, doesn't change frequently, and fits within the document size limit.

Explain the limitations of MongoDB Transactions.

1) Joins are not supported by MongoDB. Even if you are using joins you have to code manually for it and its execution is too slow.

2) Due to not having Joins functionality, it creates data redundancy. Due to this thing, it increases unnecessary memory usage.

3) Document size limit is up to 16 MB only.

4) Nesting of documents can be done up to 100 levels only.

5) MongoDB doesn't have any transaction support.

6) The MongoDB collection cannot have more than 64 indexes.

7) Length of the collection cannot be more than 125 characters.

Is there an "upsert" option in the MongoDB insert command?

Yes, MongoDB's `insert()` command includes an "upsert" option. When set to true, it inserts a new document if no matching document is found based on the query criteria, otherwise updates the existing document.

What are some of the advantages of MongoDB?

Some advantages of MongoDB include its flexibility to handle unstructured data, scalability to accommodate growing data needs, high performance for read and write operations, support for replication and sharding for high availability and horizontal scaling, and its ability to integrate with modern development technologies and frameworks. Additionally, MongoDB's flexible schema design allows for iterative development and agile workflows.

What is a Document in MongoDB?

A record in MongoDB is a document, which is a data structure composed of field and value pairs. MongoDB documents are similar to JSON objects. The values of fields may include other documents, arrays, and arrays of documents.

How to add data in MongoDB?

To add data in MongoDB, you can use the `insertOne()` method to add a single document or the `insertMany()` method to add multiple documents at once. Here's a basic example in MongoDB's JavaScript shell:

```
  Insert a single document
db.collection.insertOne({ key: value });

// Insert multiple documents
db.collection.insertMany([
 { key1: value1 },
 { key2: value2 },
 { key3: value3 }
]);
```

Replace `collection` with the name of your MongoDB collection and `key`, `value` with the field-value pairs you want to add.

How do you Update a Document?

To update a document in MongoDB, you can use the `updateOne()` or `updateMany()` method to update one or multiple documents respectively. Here's a basic example in MongoDB's JavaScript shell:

```
// Update a single document
db.collection.updateOne(
  { <filter> },
  { $set: { <update> } }
);

// Update multiple documents
db.collection.updateMany(
  { <filter> },
  { $set: { <update> } }
);
```

Replace `collection` with the name of your MongoDB collection, `<filter>` with the criteria to select the document(s) to update, and `<update>` with the new values or modifications you want to apply to the selected document(s).

What are the data types in MongoDB?

In MongoDB, the data types include:

String: UTF-8 encoded string.
Integer: 32-bit signed integer or 64-bit signed integer.
Double: 64-bit floating-point number.
Boolean: True or false value.
Date: UNIX timestamp format.
Array: List of values.
Object: Embedded document.
Null: Null value.
Binary data: Binary data.
ObjectId: Unique identifier.
Timestamp: BSON timestamp.
Decimal128: 128-bit decimal-based floating-point number.
Regular Expression: Pattern matching using regular expressions.

When to use MongoDB

You might choose MongoDB when your project requires flexible schema design, scalability for handling large volumes of data, real-time analytics, geospatial indexing, or when working with unstructured or semi-structured data. It's also suitable for applications needing high availability and horizontal scaling.

How is Querying done in MongoDB?

In MongoDB, querying is done using the `find()` method. You can specify query criteria as well as projection, sorting, and limiting options. For example:

```
// Basic query
db.collection.find({ field: value });

// Query with projection (selecting specific
  .collection.find({ field: value }, { _id: 0, field1: 1, field2: 1 });

// Query with sorting and limiting
db.collection.find().sort({ field: 1 }).limit(10);
```

Replace `collection` with the name of your MongoDB collection, `field` with the field you want to query, and `value` with the value to match.

How to perform queries in MongoDB?

In MongoDB, you perform queries using the `find()` method. Here's a basic overview:

Basic Query:

```
db.collection.find({ field: value });
```

Projection (selecting specific fields):

```
db.collection.find({ field: value }, { _id: 0, field1: 1, field2: 1 });
```

Sorting and Limiting:

```
db.collection.find().sort({ field: 1 }).limit(10);
```

You can combine these options as needed to retrieve the desired data from your MongoDB collection.

How do you Delete a Document?

To delete a document in MongoDB, you can use the `deleteOne()` method to delete a single document or the `deleteMany()` method to delete multiple documents. Here's a basic example in MongoDB's JavaScript shell:

```
// Delete a single document
db.collection.deleteOne({ <filter> });

// Delete multiple documents
db.collection.deleteMany({ <filter> });
```

Replace `collection` with the name of your MongoDB collection, and `<filter>` with the criteria to select the document(s) to delete.

What are Geospatial Indexes in MongoDB?

Geospatial indexes in MongoDB are special indexes used to support efficient queries on geospatial data. They enable MongoDB to perform spatial queries, such as finding nearby locations or determining if a point is within a specified area, by indexing and querying geometries stored in documents.

Explain the SET Modifier in MongoDB?

In MongoDB, the `$set` modifier is used in update operations to set or update the value of a field within a document. It can be used to add new fields if they don't exist or update existing fields. When combined with the `update()` method, `$set` ensures that only specified fields are modified, leaving other fields unchanged.

Explain the process of Sharding.

Sharding in MongoDB involves distributing data across multiple machines to improve scalability and performance. The process includes:

Shard Key Selection: Choose a field to act as the shard key, which MongoDB uses to partition data across shards.
Shard Cluster Setup: Configure a sharded cluster consisting of multiple servers (shards), each responsible for storing a subset of the data.
Data Distribution: MongoDB automatically distributes data across shards based on the shard key. Each shard holds a distinct range of data determined by the shard key values.
Query Routing: When querying data, MongoDB's router (mongos) determines which shard(s) to query based on the shard key values specified in the query.
Data Balancing: MongoDB continuously rebalances data across shards to ensure even distribution and optimal performance.

Sharding enables horizontal scaling, allowing MongoDB to handle large datasets and high query loads by distributing data and query processing across multiple servers.

What do you mean by Transactions?

Transactions in MongoDB refer to a set of operations that are executed as a single, atomic unit of work. This means that either all the operations within the transaction are successfully completed, or none of them are. Transactions ensure data integrity and consistency by guaranteeing that changes made by the transaction are isolated from other operations until the transaction is committed. If any part of the transaction fails, MongoDB rolls back all changes, maintaining the integrity of the database.

What are MongoDB Charts?

MongoDB Charts is a feature of MongoDB that allows users to visualize their MongoDB data without needing to export it to another tool. It enables users to create, share, and embed visualizations of their MongoDB data directly within their MongoDB Atlas environment or on-premises deployments. MongoDB Charts supports a variety of chart types, filtering options, and aggregation pipelines, making it easy to explore and understand data stored in MongoDB databases.

What is the Aggregation Framework in MongoDB?

The Aggregation Framework in MongoDB is a powerful tool for performing data processing tasks such as filtering, grouping, and transforming data within a collection. It allows users to run complex queries and perform analytics operations on large datasets. The Aggregation Framework uses a pipeline-based approach, where documents are processed through a series of stages, each stage performing a specific operation on the input documents and passing the results to the next stage in the pipeline. This allows for flexible and efficient data aggregation and analysis directly within MongoDB.

What is a Replica Set in MongoDB?

A replica set in MongoDB is a group of MongoDB servers that maintain the same data set, providing redundancy and high availability. It consists of multiple nodes: one primary node for all write operations and one or more secondary nodes that replicate data from the primary node. If the primary node fails, one of the secondary nodes automatically becomes the new primary node, ensuring continuous operation and data availability. Replica sets are essential for fault tolerance, data durability, and disaster recovery in MongoDB deployments.