Frequently asked: MongoDB Interview Questions and Answers

Vigo Webs Follow
Sep 24, 2018 · 7 min read

Q1. What is MongoDB and Why we need MongoDB?

MongoDB is the most famous NoSQL open source database management system. It is written in C++ language and developed by MongoDB Inc.

MongoDB is a document oriented database which means it stores the data in BSON format which is a binary representation of JSON and it contains more data types than JSON.

```
{
  item: "Book",
  qty: 25,
  status: "A",
  size: {
    h: 14,
    w: 21,
    unit: "cm"
  },
  tags: [ "fiction" ]
}
```

In MongoDB, tables are called as collection, rows are called as document, columns are called as field, joins are called as linking.

MongoDB is widely supported by most of the languages like, JavaScript, Java, C# and runs on all the available OS.

Q2. What is difference between MongoDB and SQL or NoSQL between SQL?

In SQL, data is stored in Table format whereas in MongoDB, it stores in JSON format.

In MongoDB, development is simplified as its documents map naturally to modern, object-oriented programming languages. In SQL, we need object-relational mapping (ORM) layer that translates objects in code to relational tables.

The MongoDB provides high performance, high availability, easy scalability rather than SQL Server.

In MongoDB, each document can store data with different attributes from other documents. With JSON documents, we can add new attributes when we need to, without having to alter a centralized database schema. But in a relational database, this causes downtime and significant performance overhead

Having all the data for an object in one place also makes it easier for developers to understand and optimize query performance.

When using JSON data, MySQL drivers do not have the capability to properly and precisely convert JSON into a useful native data type used by the application. This includes different types of numeric values (e.g. floating points, 64-bit integers, decimals) timestamps, and dates, or a Map or List in Java or a Dictionary or List in Python. Instead developers have to manually convert text-based JSON in their application, losing the ability to have fields that can take on multiple data types in different documents (polymorphism) and making the computation, sorting and comparison of values difficult and error-prone. But MongoDB and its drivers supports advanced data types not supported by regular text-based JSON.

Q3. What is BSON in MongoDB?

MongoDB stores data as BSON documents. BSON is a binary representation of JSON documents, though it contains more data types than JSON. Some of the supported data types are: Double, String, Object, Array, Binary Data, RegEx, Boolean, Date, Null, JavaScript, Timestamp. ObjectId.

Some of the features of BSON:

- BSON is lightweight and is an important feature for any data representation format, especially when used over the network.
- BSON is designed to be traversed easily.
- It is efficient. Encoding data to BSON and decoding from BSON can be performed very quickly in most languages

Q4. What is `_id` Field in MongoDB?

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.

The _id field has the following behavior and constraints:

- By default, MongoDB creates a unique index on the `_id` field during the creation of a collection.
- The `_id` field is always the first field in the documents. If the server receives a document that does not have the `_id` field first, then the server will move the field to the beginning.
- The `_id` field may contain values of any BSON data type, other than an array.

While storing values for `_id` , we can follow these options: Use an ObjectId, Use a natural unique identifier, Generate an auto-incrementing number, Generate a UUID in our application code.

Most MongoDB driver clients will include the `_id` field and generate an `ObjectId` before sending the insert operation to MongoDB; however, if the client sends a document without an `_id` field, the `mongod` will add the `_id` field and generate the `ObjectId` .

Q5. How do you create or select a database in MongoDB?

In MongoDB, databases hold collections of documents. To select a database to use issue the `use <db>` statement, as in the following example:

```
use myDB
```

If a database does not exist, MongoDB creates the database when we first store data for that database. Some points to be noted when creating database in MongoDB are:

- Database Name Case Sensitivity
- For MongoDB deployments running on Windows, database names cannot contain any of the following characters: `/\. "$*<>:|?`
- For MongoDB deployments running on Unix and Linux systems, database names cannot contain any of the following characters: `/\. "$`
- Database names cannot be empty and must have fewer than 64 characters.

Q6. What is collection in MongoDB?

A *collection* is a group of documents. If a document is the MongoDB analog of a row in a relational database, then collection is the analog to a table.

Collections have *dynamic schemas*. This means that the documents within a single collection can have any number of different shapes. For example, both the following documents can be stored in a single collection.

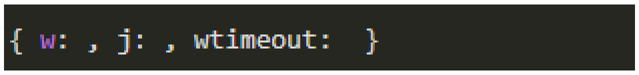
```
{ "greeting": "Hello world!" }
{ "message": "Learn MongoDB" }
```

Note that the previous documents not only have different types for their values (string versus integer) but also have entirely different keys.

Because in MongoDB any document can be put into any collection.

Q7. What is `writeConcern` ?

Whenever we query the database to `insert()` , `insertOne()` or `insertMany()` , we can pass an additional parameter which is `writeConcern` . It is a document describes the level of acknowledgement requested from MongoDB for write operations. Write concern can include the following fields:



```
{ w: , j: , wtimeout:  }
```

- the `w` option to request acknowledgement that the write operation has propagated to a specified number of mongod instances or to `mongod` instances with specified tags.
- the `j` option to request acknowledgement that the write operation has been written to the journal, and
- the `wtimeout` option to specify a time limit to prevent write operations from blocking indefinitely.

Q8. What is `upsert` ?

`upsert` is one of the option that we can pass to the `update` operation as third parameter.

If `upsert` is true and no document matches the query criteria, `update()` inserts a *single* document. The update creates the new document with either:

- The fields and values of the `update` parameter if the `update` parameter is a replacement document (i.e., contains only field and value pairs). If neither the `query` nor the `update` document specifies an `_id` field, MongoDB adds the `_id` field with an `ObjectId` value.
- The fields and values of both the `query` and `update` parameters if the `update` parameter contains update operator expressions (such as `$inc` , `$set` , `$min` , `$max`). The update creates a base document from the equality clauses in the parameter, and then

applies the update expressions from the `update` parameter. Comparison operations from the `query` will not be included in the new document.

Also if we want to update multiple documents we can pass `multi: true` as another option.

Q9. Explain some of the Evaluation Query Operators.

- **`$expr`** : `$expr` can build query expressions that compare fields from the same document in a `$match` stage. Example: `db.users.find( { $expr: { $gt: [ "$age", "$requiredAge" ] } } )` . This query will finds documents where the `age` is greater than the `requiredAge` .
- **`$jsonSchema`** : `$jsonSchema` can be used in a document validator, which enforces that inserted or updated documents are valid against the schema.
- **`$mod`** : Select documents where the value of a field divided by a divisor has the specified remainder. It is performing a modulo operation. Example: `db.users.find( { items: { $mod: [ 4, 0 ] } } )`
- **`$regex`** : Provides regular expression capabilities for pattern matching strings in queries. MongoDB uses Perl compatible regular expressions. Example: `db.users.find( { name: { $regex: /^John/ } } )`
- **`$text`** : `$text` performs a text search on the content of the fields indexed with a text index. Example: `db.users.find( { $text: { $search: "john" } } )`
- **`$where`** : Use the `$where` operator to pass either a string containing a JavaScript expression or a full JavaScript function to the query system.

Q10. What is Storage Engine? What are all the Storage Engines available in MongoDB?

The storage engine is the component of the database that is responsible for managing how data is stored, both in memory and on disk.

MongoDB supports multiple storage engines, as different engines perform better for specific workloads.

- **WiredTiger Storage Engine**: This is the default engine. It is well-suited for most workloads and is recommended for new deployments. WiredTiger provides a document-level concurrency model, checkpointing, and compression, among other features.
- **In-Memory Storage Engine**: In-Memory Storage Engine is available in MongoDB Enterprise. Rather than storing documents on-disk, it retains them in-memory for more predictable data latencies.
- **MMAPv1 Storage Engine**: This is deprecated as of MongoDB v4.0 and is the default storage engine for MongoDB versions 3.0 and earlier.

For more Interview Questions and answer from Full stack development try our Android App:

https://play.google.com/store/apps/details?id=com.vigowebs.interviewquestions