AIMERZ.ai
Aim.Act.Achieve

mongoDB

INTERVIEW

Questions & Answers

MongoDB

MongoDB is a NoSQL database that is ideal for developers who want flexibility, scalability, and high performance. Strong, scalable, and incredibly flexible, MongoDB leads to a NoSQL database and offers excellent performance even with massive data volumes. The data is kept in BSON format by MongoDB.

1. How does MongoDB differ from relational databases?

MongoDB differs from relational databases in several key ways:

- The MongoDB database stores information in documents that are flexible and resemble BSON, whereas relational databases store information in tables that are structured and have rows and columns.
- Relational databases have a preset schema with fixed structures, but MongoDB uses a dynamic schema that allows fields to change between documents.
- **Scalability:** Relational databases usually scale vertically, i.e., by adding more resources to a single server, whereas MongoDB is designed to scale horizontally, i.e., across multiple servers.
- **Joins:** While relational databases use joins to combine data from multiple tables, MongoDB avoids complex joins and encourages data to be embedded in a single document.
- **Transactions:** MongoDB has historically supported eventual consistency, but it now allows multi-document ACID transactions. Relational databases, on the other hand, offer robust ACID-compliant transactions across multiple operations.

2. What do you understand about documents in the context of Databases?

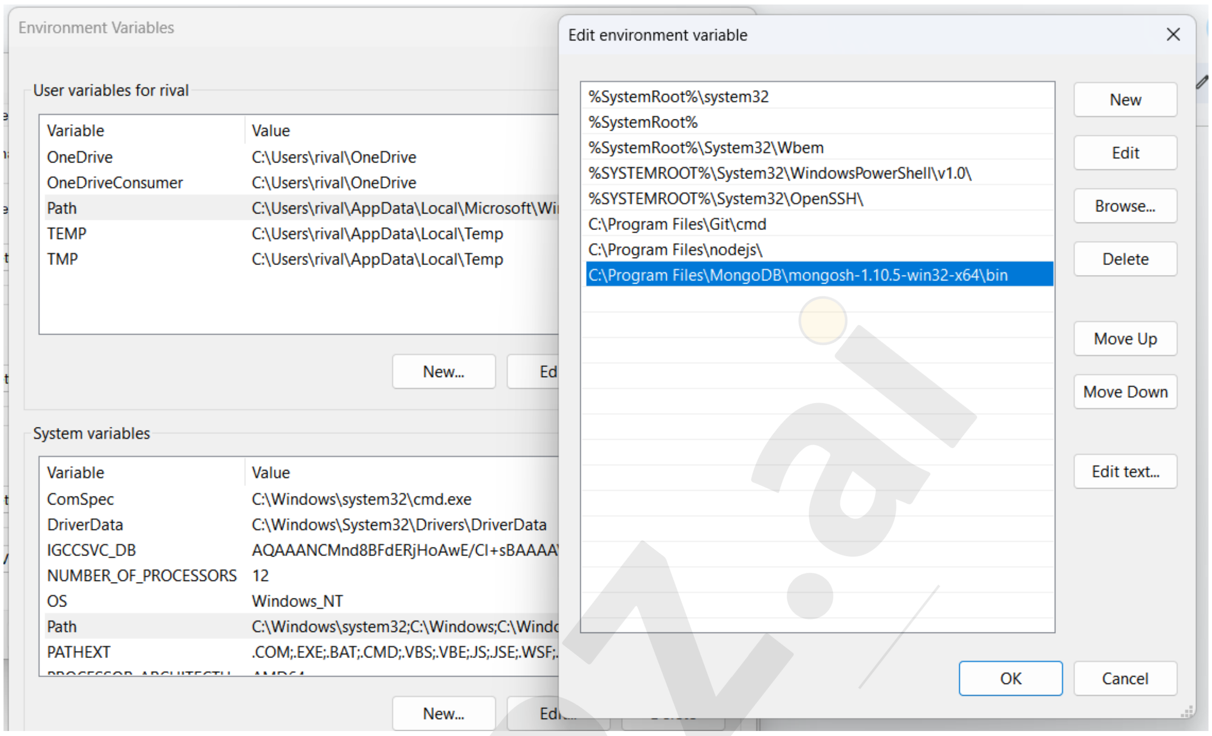
The core of MongoDB is the document. To put it simply, the document consists of an ordered list of keys and their corresponding values. It is comparable to the rows in tables found in RDBMS systems. It always keeps its scheme dynamic, negating the need for any fields or predefined structure.

3. What is the process to setup mongoDB on a local machine ?

Here are a few steps to setup mongoDB in your local machine

- Goto mongoDB products page and Download the mongoDB community server setup and from tools section install mongo Shell as per your OS requirements.
- Install it on your machine and set up it properly
- After Installation ensure the path of bin directory of mongo shell is already set inside **Environment variables.**

Unset
```
C:\Program Files\MongoDB\mongosh-<version name>\bin
```



Variable	Value
OneDrive	C:\Users\rival\OneDrive
OneDriveConsumer	C:\Users\rival\OneDrive
Path	C:\Users\rival\AppData\Local\Microsoft\Wi
TEMP	C:\Users\rival\AppData\Local\Temp
TMP	C:\Users\rival\AppData\Local\Temp

Variable	Value
ComSpec	C:\Windows\system32\cmd.exe
DriverData	C:\Windows\System32\Drivers\DriverData
IGCCSVC_DB	AQAAANCMnd8BFdERjHoAwE/Cl+sBAAAA
NUMBER_OF_PROCESSORS	12
OS	Windows_NT
Path	C:\Windows\system32;C:\Windows;C:\Windo
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;

```
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files\Git\cmd
C:\Program Files\nodejs\
C:\Program Files\MongoDB\mongosh-1.10.5-win32-x64\bin
```

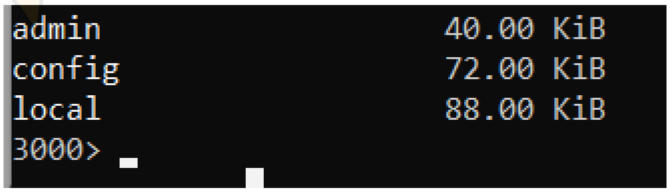
- Now search mongo shell from the start menu, and start using it.

4. What is the command to launch mongosh ?

Firstly, open windows cmd and hit command , **mongosh**

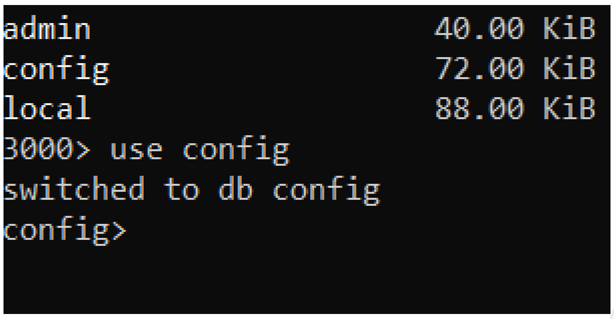
5. How can you use a specific database?

- Firstly, open the mongo shell.
- Run command, > show dbs



```
admin                40.00 KiB
config               72.00 KiB
local                88.00 KiB
3000> _
```

- As shown above you will see a list of all the available databases.
- Now Run command use <db name>



```
admin                40.00 KiB
config               72.00 KiB
local                88.00 KiB
3000> use config
switched to db config
config>
```

As shown above you will be switched to the specific database.

6. What is a Collection and how is a Collection made up?

In MongoDB, a collection is a set of documents. A collection can be compared to a table if a document is the MongoDB equivalent of a row in an RDBMS.

A collection of documents can have any number of distinct collections and have dynamic schemas.

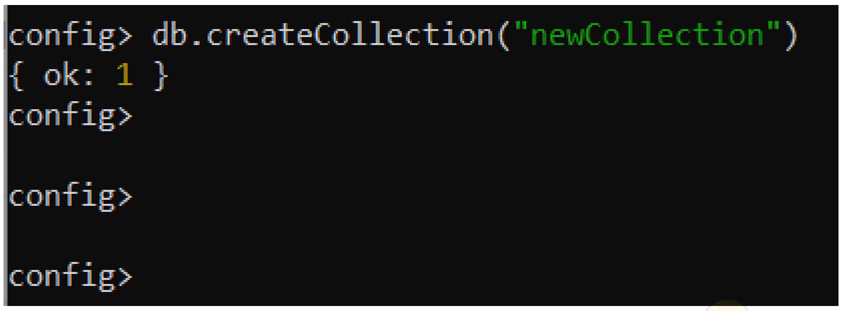
NOTE:

- All the collection names in MongoDB are case sensitive,
- Field names in the documents are also case sensitive.

Unset

```
db.createCollection(name,options)
```

Above command will give below response:



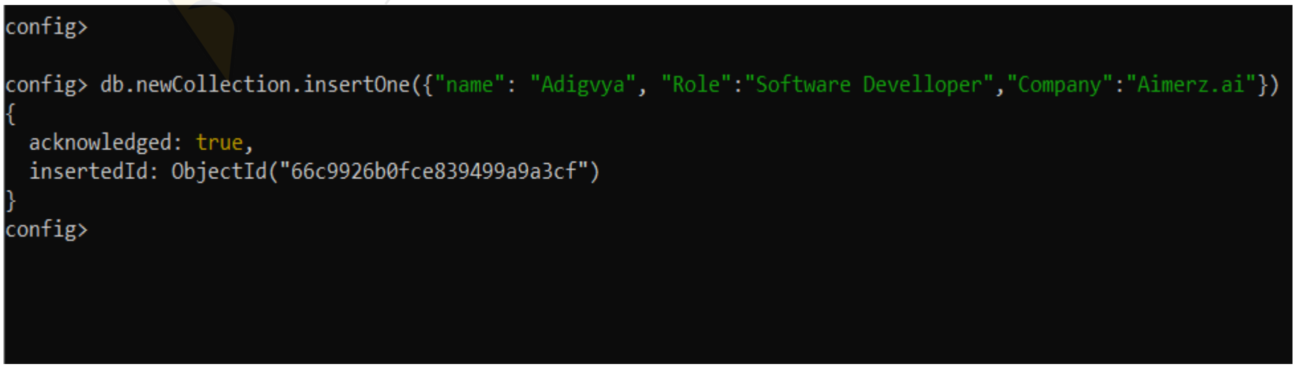
```
config> db.createCollection("newCollection")
{ ok: 1 }
config>

config>

config>
```

7. How to insert entries in a database inside a collection with MongoSHELL local database.

We can use command:

```
db.newCollection.insertOne(
      {     "name":"Adigva",
            "Role":"Software Developer",
            "Company":"Aimerz.ai"
      }
)
```

The response on the console must be acknowledge:true , as mentioned below:



```
config>

config> db.newCollection.insertOne({"name": "Adigvya", "Role":"Software Develloper","Company":"Aimerz.ai"})
{
  acknowledged: true,
  insertedId: ObjectId("66c9926b0fce839499a9a3cf")
}
config>
```

In similar way if we want to insert many entries at the same time we can use:

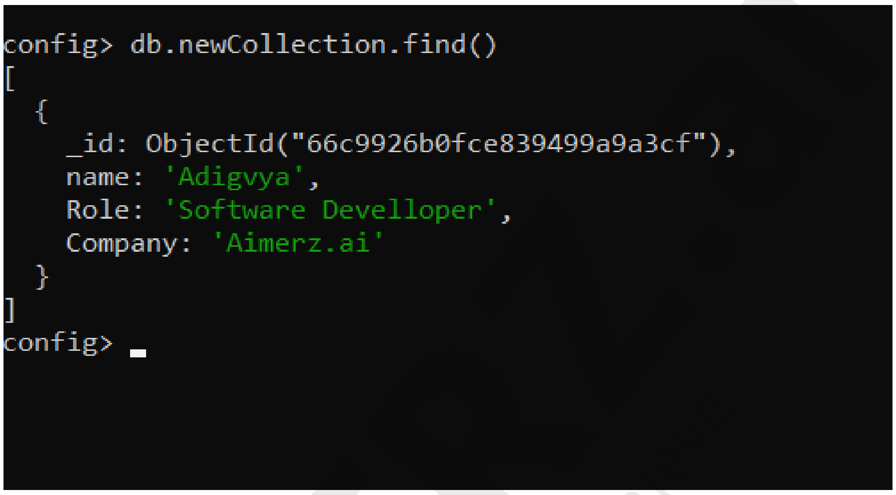
```
Unset
db.newCollection.insertMany(
     {    "name":"Adigva",
          "Role":"Software Developer",
          "Company":"Aimerz.ai"
     },
     {    "name":"Saksham",
          "Role":"Software Developer",
          "Company":"Google"
     }

     {    "name":"Tripathi Ji",
          "Role":"Software Developer",
          "Company":"Tower Research"
     }
)
```

8. How to find entries in a database inside a collection with MongoSHELL local database.

For finding the entries in any database inside a collection just type command as followed:

```
Unset
db.newCollection.find()


// Also we can use another commands with arguments inside
db.newCollection.find({'name':'Adigvya'})
```

```
// Below command will get us the 1 top most occuring
entity
db.newCollection.findOne({'name':'Adigvya'})
```



```
config> db.newCollection.find()
[
  {
    _id: ObjectId("66c9926b0fce839499a9a3cf"),
    name: 'Adigvya',
    Role: 'Software Develloper',
    Company: 'Aimerz.ai'
  }
]
config> _
```

9. How you can delete a collection in mongoDB.

Use command > *db.<collection-name>.drop( );*

The above command will delete the whole collection permanently.

10. What are the Ordered and Unordered Inserts ?

We know that if we are about to work in production or processing in BULK we need to take advantage of the ordered clause in mongoDB.
Here, by default the behaviour is ordered but mongoDB stops on the occurrence of its first error.
We can use :

```
Unset
db.<collection name>.insertMany([part1, part2, part3,
part4]);
```

Whether in the unordered clause even if the error is occurred the processing is continued.

```
Unset
db.<collection name>.insertMany([part1, part2, part3,
part4], { ordered: false});
```

11. What do you understand about comparison operators in the context of mongoDB?

There are various types of comparison operators :

- `$eq`: It matches values that are equal to a specified value.
- `$ne`: It matches values that are not equal to a specified value.
- `$gt`: It matches values greater than a specified value.
- `$gte`: It matches values greater than or equal to a specified value.
- `$lt`: It matches values less than a specified value.
- `$lte`: It matches values less than or equal to a specified value.
- `$in`: It matches any of the values specified in an array.
- `$nin`: It matches none of the values specified in an array.

We can use one or more then one combination of above comparison operators herein as followed:

```
Unset
db.< collection name >.find({ 'price': {$eq: 699}  });
db.<collection name>.find({'price':{$nin:[249,129,39] });

...
```

12. List out some logical operators used in mongoDB.

There are various types of comparison operators :

- `$and`: Joins query clauses with a logical AND.
- `$or`: Joins query clauses with a logical OR.
- `$not`: Inverts the effect of a query expression.
- `$nor`: Joins query clauses with a logical NOR (none must match).

To find all the entities with Field 1 greater than 25 and field 2 is set to active.

```
Unset
db.<collection name>.find({
  $and: [
    { <field 1>: { $gte: 25 } },
    { <field 2>: "active" }
  ]
})
```

13. What are the various cursor methods available?

MongoDB provides several cursor methods that allow you to manipulate and iterate through query.

- **count():** Gives back the total number of records that fit the search.
- **`limit()`:** Restricts the number of documents returned by the query.
- **`skip()`:** Skips a specified number of documents in the result set.
- **next():** Gets the document that's next in the cursor.
- **Sort():** Arranges the result set's documents according to a specified field.

```
Unset
// Implementation for Cursor methods

db.<collection name>.find({
price:{$gt:250}}).count();

db.<collection name>.find({
price:{$gt:250}}).skip(2);

db.<collection name>.find({ price:{$gt:
250}}).limit(2).sort({ price: 1});
```

14. What do you understand by projection in databases?

When we encounter a situation where we need to include specific fields and or remove some fields we use the Projection in databases.

- Herein if we want to include specific projection, we use 1 with their fields.
- Also if we want to exclude any projection we use 0 along with its field.

NOTE: We cannot use both exclude and include projection at the same time in mongoDB.

```
Unset
// Include
db.collection.find({{   },  <field 1>:1, <field 2>:1 })

// Exclude
db.collection.find({{   },  <field 1>:0, <field 2>:0 })
```

15. What are the update operations used in mongoDB ?

There are mainly two kinds of update operations in mongoDB:

(a) Update specific entry in the database.

```
Unset
db.<collection name>.updateOne(
           {filter},{$set: {<existingField>:<updated value>,
<New   field (optional)>: <new value >}
})
```

(b)Update multiple fields in the database.

```
Unset
db.<collection name>.updateMany(
           {filter},{$set: {<existingField>:<updated value>
....<and soo onn>}
})
```

16. What are the Removing and Renaming operations used in mongoDB ?

There Removing and Renaming operations in mongoDB are described below:

(a) Update specific field name in the database.
(b) Removing specific fields from the database.

```
Unset

//Update specific field name in the database

db.< collection name >.updateOne(
     {filter},
     { $rename:{oldFieldName:"newFieldName"}}
);

//Remove field name from the database
db.< collection name >.updateOne(
     {filter},
     { $unset:{fieldName:1}}
);
```

17. How do you delete the entries in a database ?

For deleting any specific entry or entries in a document database life mongoDB there are mainly two methods:

a. deleteOne( )
b. deleteMany( )

```
Unset
//For deletion of first occuringng entry in the
//database.
db.< collection name >.deleteOne({filter});

//For multiple entries deletion fromm the //database.
db.< collection name >.deleteMany({name:<Any specific
name>});
```

18. Explain Indexes concept in Database ?

Indexes are unique data structures in MongoDB that provide easy-to-access storage for a small percentage of a collection's data. Instead of scanning every document in a collection, indexes enable the relational database to quickly locate and access the data, improving query performance. They resemble relational database indexes.

19. Describe "explain()" clause in terms of Query optimization.

In general, we use explain() to understand the query execution in detail.
Also we can measure the time taken to execute a query.

```
Unset
//The below query will execute and all the execution will
be explained as we //have added .explain() after it.
```

```
db.<collection name>.find({name:
<aimerz.ai>}).explain("executionStatus")
// Here in this query as an argument is passed inside
//the explain() block. i.e. it will only tell the //total
timme taken in execution of this query.
```

20. Describe something about Mongo Atlas. How is it different from DB in your local machine?

MongoDB Atlas is a fully managed cloud database solution. With its features, which include security management, monitoring, and automated backups, users can concentrate on developing applications rather than managing infrastructure. Major cloud platforms such as AWS, Azure, and Google Cloud offer MongoDB Atlas.

1. **Deployment and Management:** While local requires manual setup and maintenance, Atlas simplifies both deployment and management.
2. **Scalability:** While local scaling necessitates manual server configuration, Atlas provides simple scaling with just a few clicks.
3. **Backups and Recovery:** Local backup setup necessitates manual effort, but Atlas offers automated backups.
4. **Security:** While local security needs to be configured by the user, Atlas comes with built-in security features.
5. **Monitoring and Alerts:** While local requires external tools for monitoring, Atlas offers integrated monitoring and alerts.
6. **High Availability:** Whereas local requires manual replica set configuration, Atlas guarantees automatic high availability.

21. How can you connect mongoDB Atlas with node.js ?

1. **Open an account on the Web with Atlas:** Register with MongoDB Atlas and start a fresh cluster.
2. Select the cluster tier, region, and cloud provider when configuring the cluster.
3. Create a database user and set a password under the Database Access tab after setup.
4. **Permission Entry:** Whitelist your IP address (or permit access from any location) under the Network Access tab.
5. **Obtain Connection String:** Click Connect under the Clusters tab, then select "Connect your application”.

Unset
```
// Connection String
mongodb+srv://\username>:\password>@cluster0.
mongodb.net/?retryWrites=true&w=majority,
substitute that string for the connection
string.
```

6. Ensure the setup of mongoDB in your local machine Use command (npm install mongodb).
7. **Connect to MongoDB Atlas:** Use the connection string in your node.js application.

JavaScript

```javascript
const { MongoClient } = require('mongodb');

const { MongoClient } = require('mongodb');


const uri = "<-your-mongodb-connection-string->";

const client = new MongoClient(uri, { useNewUrlParser: true,
useUnifiedTopology: true });


async function run() {

  try {

    await client.connect();

    console.log("Connected to MongoDB Atlas");

  } finally {

    await client.close();

  }

}

run().catch(console.dir);
```

8. Another way to connect to mongoDB Atlas is by using Mongoose ODM, it will be further discussed in the next (node.js) module.

22. What are the multiple languages supported by MongoDB?

MongoDB officially supports the following languages: C, C++, C#, Java, Node.js, Perl, PHP, Python, Ruby, Scala, Go, and Erlang. We can use MongoDB with any of the languages mentioned above. There are several other community-supported drivers available, but MongoDB provides the ones described above.

23. What is the aggregation framework in MongoDB?

Aggregation framework is an advanced functional feature of MongoDB, which is used to process data and compile information from documents in a collection. With the help of this feature, you can accomplish complex data processing and aggregation outcomes by transforming and combining data in a series of steps that build upon each other using a staged, pipeline model.

24. What does a MongoDB document's _id field do, and is it customizable?

The primary key in a MongoDB document that gives each document in a collection its unique identity is the "_id" field. Although "_id" values are automatically generated by MongoDB, you have the option to modify this field if you require different values or would prefer to use a different field as the primary key.

25. In MongoDB, what is a cursor and how is it used?

A cursor is an iterator for navigating through query results in MongoDB. It is a powerful concept and tool. Upon the execution of a find() operation, MongoDB returns a cursor to the documents that match rather than the complete set of results right away. This cursor efficiently retrieves and manipulates data, either batch-processed or one document at a time, by pointing to the query results.

When working with large datasets that might require too much memory to load and process all at once, cursors come in handy. Cursors allow MongoDB to retrieve documents in smaller, more manageable batches, which lessens the strain on the database and application memory.

26. How can a case-insensitive search be carried out in MongoDB?

Use regular expressions with the $regex operator to do a case-insensitive search, and set the $options to "i" to indicate case-insensitive matching. As an illustration,

```
        db.collectionName.find({
            field: {
                $regex: "searchTerm", $options: "i"
                }
            }).
```

27. What is the MongoDB $push operator and how does it operate?

In MongoDB, an array field within a document can have elements added to it using the $push operator. It adds the designated value or values to the array. As an illustration:

```
    db.collectionName.update({
        _id: ObjectId("documentId")
        },
        {
            $push: {
                fieldName: valueToPush
                }  })
```

28. What is the default port number for MongoDB and how can you change it?

The default port number for MongoDB is 27017. You can change it by specifying a different port number in the MongoDB configuration file or using the --port option when starting the MongoDB server.

29. How can a MongoDB database backup be made?

A MongoDB database can be backed up using file system-level backups or by using tools such as mongodump. The built-in MongoDB tool mongodump exports the database as a binary file that can be imported back into the database using mongorestore.

When performing file system-level backups, the whole database directory is copied while the database is in a consistent state.

30. Why is MongoDB known as the best NoSQL database?

MongoDb is the best NoSQL database because, it is:

- Document Oriented
- Rich Query language
- High Performance
- Highly Available
- Easily Scalable

THANK YOU

AIMERZ.ai

Aim.Act.Achieve