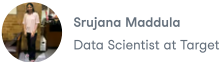


Home > Blog > MongoDB

Top 25 MongoDB Interview Questions and Answers for 2025

This guide covers essential MongoDB interview questions, from basics to advanced, with coding examples, real-world scenarios, and expert tips to help you succeed!

Dec 13, 2024 · 25 min read



TOPICS

- MongoDB
- NoSQL

To date, I've given many data science and database interviews. Companies are increasingly looking for professionals with expertise in NoSQL databases. **MongoDB** is one of the most flexible databases, capable of supporting scalable modern applications with minimal downtime.

In this article, I've compiled a list of interview questions I've come across, along with insights I collected from my colleagues' interview experiences.

What is MongoDB?

MongoDB is a **No-SQL database** that stores data in a flexible, schema-less architecture. Unlike traditional tables, it uses **documents** and **collections** to save records in a JSON-like format called BSON. This format allows MongoDB to store a variety of data types in a hierarchical model.

Since MongoDB doesn't have a fixed schema, it allows any type of storage, making it a good choice for real-time analytics and data streaming. Moreover, modern applications often experience rapid growth or unpredictable traffic, which MongoDB is uniquely equipped to handle. For instance, MongoDB supports horizontal scaling, allowing you to add extra servers to handle increasing load.

Get certified in your dream Data Analyst role

Our certification programs help you stand out and prove your skills are job-ready to potential employers.

Get your Certification



Basic MongoDB Interview Questions

In this section, we will focus on foundational questions often asked in MongoDB interviews.

Explain the BSON storage format.

BSON stands for binary Javascript object notation. It stores serialized JSON documents in a binary-encoded format and extends JSON capabilities by supporting additional data types like Date, ObjectId, and regular expressions.

A BSON document typically contains three components: the size of the document, field elements(such as data type, name, and value), and a null terminator, all encoded in binary format.

The following table illustrates the differences between BSON and JSON, especially since BSON extends JSON capabilities:

Feature	JSON	BSON
Encoding Format	Text-based	Binary
Data Types Supported	String, Number, Boolean, etc.	JSON types + Date, ObjectId, Regular Expressions
Readability	Human-readable	Machine-readable
Use Case	Data interchange	Data storage in MongoDB

What is a collection in MongoDB?

A MongoDB collection is a group of documents stored in a single database. It is similar to a table in a relational database, where each document in a collection represents a row in a table. However, the collection doesn't have a fixed schema, meaning the documents within a collection can be of different data types.

How to query a document in MongoDB?

In MongoDB, you can query documents using the `find()` method. To query all documents in a collection, use `db.collection_name.find()`. The find method has two input parameters: `query` and `projection`. The `query` parameter is used to filter documents that match a specific condition.

Syntax for query parameter:

```
db.collection_name.find({condition})
```

POWERED BY datacamp

The second is a projection parameter that indicates the columns to include or exclude in the output. Assign 1 to the columns that you want to fetch. Here is the syntax:

```
db.collection_name.find({}, {column1: 1, column2: 1})
```

POWERED BY datacamp

What is the difference between find() and findOne()?

The `find()` method returns a cursor to multiple documents that match the query criteria. This works by iterating over all the results and fetching the matching documents. On the other hand, the `findOne()` method returns the first matched document.

What is the _id field in MongoDB?

Each document stored in a collection requires a unique identifier. This `_id` field acts as a primary key to uniquely identify documents in a collection.

Intermediate MongoDB Interview Questions

Now, let's move on to some core MongoDB concepts that the interviewer might expect you to know.

How does MongoDB store large images and videos?

Traditionally, MongoDB doesn't allow the storage of documents larger than 16 MB. However, it has a special specification called GridFS for storing and retrieving files greater than 16 MB. It divides the file into smaller, equal chunks and stores them as separate documents internally.

The table below summarizes how MongoDB stores large files using GridFS:

Step	Description
File Chunking	Splits files into smaller chunks of 255 KB or less
Metadata Storage	Stores metadata about the file in a <code>files</code> collection
Chunk Storage	Stores file chunks in a <code>chunks</code> collection
Retrieval	Reconstructs the file from stored chunks

How does MongoDB ensure high availability?

MongoDB achieves high availability through replication. Replica sets store different copies of data across nodes so that if one node fails, another can take over.

What is sharding in MongoDB?

Sharding enables horizontal scaling in MongoDB. When a single instance can't manage a large dataset, MongoDB splits the data into smaller chunks and distributes them across multiple servers, known as shards.

The following table clarifies the differences between replication and sharding:

Feature	Replication	Sharding
Purpose	High availability	Scalability for large datasets
Implementation	Replica sets (multiple copies)	Data partitioned across shards
Use Case	Fault tolerance	Load balancing for large databases
Data Distribution	All nodes store the same data	Data is distributed across nodes

What is a replica set in MongoDB?

A replica set in MongoDB is a group of instances that maintain the same dataset. They are deployed in applications that require high availability because if one instance troubles, the system automatically shifts to the next available node in the replica set.

Explain the concept of aggregation in MongoDB.

MongoDB aggregates data from multiple documents and processes it to return a single result. This involves an aggregation pipeline, where documents pass through several stages —each stage's output becomes the input for the next. A typical pipeline might include stages like match, group, and sort:

- Match: filters documents based on the given criteria.
- Group: Performs aggregation operation.
- Sort: Sorts the final results the way we need.

What is a capped collection in MongoDB?

A capped collection has a fixed size and a limit for the number of documents. When the limit is reached, it automatically overwrites the oldest document and stores the latest information. This concept makes it suitable for use cases like logging and caching.

Advanced MongoDB Interview Questions

In this section, we take a look at some popular advanced MongoDB interview questions and answers.

Does MongoDB support ACID transactions?

Until version 4.0, MongoDB supported ACID transactions only for single documents. With its multi-document ACID transactions, developers can now ensure **ACID properties** across multiple documents within a collection.

What is map-reduce in MongoDB?

Map-reduce is a data processing paradigm that performs operations on large datasets and outputs aggregated results. MongoDB offers a built-in `mapReduce()` function consisting of two stages: map and reduce.

During the map phase, the function processes each document in the collection and generates key-value pairs. These key-value pairs are aggregated in the reduce phase, and operations are performed.

For example, if you have a collection of text documents, the map function would convert each word into a key and assign a value of 1 to it. The reduce function then sums the values of each key to count the occurrences of each word across the entire collection.

Explain TTL indexes in MongoDB.

Generated data should be consistently reviewed and removed when not required; otherwise, you will run out of resources to accommodate the latest information.

MongoDB provides Time-to-Live (TTL) indexes, which streamline the deletion of expired documents. All you need to do is specify how long a document should be retained, and TTL will automatically remove it once the specified time period has passed.

The following table explains the types of indexes available in MongoDB and their use cases:

Index Type	Description	Example Use Case
Single Field	Index on a single field	Indexing email for faster lookup
Compound	Index on multiple fields	Sorting by <code>lastName</code> and <code>age</code>
Text	Full-text search on string fields	Searching a blog post by keywords
TTL	Automatically deletes expired documents	Log cleanup after a specific time

Top 25 MongoDB Interview Questions and Answers for 2025 | DataCamp

Geospatial

Supports location-based queries

Finding nearby restaurants

Does MongoDB feature backup and recovery?

MongoDB allows data backup through the `mongodump` utility. This tool creates binary backups of your data, which you can import whenever you need to. Another option is to use third-party cloud solutions or [MongoDB Atlas](#)(cloud service) to automate the backup process.

MongoDB provides the `mongorestore` utility to import data from backed-up BSON files. Additionally, third-party cloud solutions offer automated restoration capabilities, minimizing downtime.

How can you optimize MongoDB queries?

Here are a few solutions that can be applied to optimize your MongoDB queries:

- Indexes store information about documents, which helps to locate the right data quickly. So, creating indexes can improve query performance.
- If you know which columns you need, use projection methods to return only those fields for better performance.
- Avoid expensive operations like regular expressions; use prefix searches or indexed fields instead.
- Choose the right shard key, especially when working on read-heavy workloads.

Explain journaling in MongoDB.

When a write operation is performed, MongoDB records it to the [journal files](#) before they are etched into the database files. These logs ensure that committed write operations can be quickly recovered in case of system failures or crashes.

MongoDB Coding Interview Questions


Coding interview questions often focus on your ability to implement MongoDB concepts through code. They test your syntax, coding practices, and how efficiently you can use MongoDB tools and functions.

How to create an index in MongoDB?

MongoDB has a `createIndex()` function to create various types of indexes, such as single-field indexes, text indexes, and 2D indexes. The method has two input parameters: keys defining the columns to index and other options.

Syntax:


```
db.collection.createIndex(keys, options)
```

POWERED BY  datacamp

- Keys: { field1: 1, field2: -1, ... }, 1 for ascending order and -1 for descending order
- Options: {unique: true}, {sparse: true}, { expireAfterSeconds: 3600 }

Example:

```
db.users.createIndex({ email: 1 }, { unique: true });
```

POWERED BY  datacamp


How to implement aggregation in MongoDB?

Aggregation typically contains three stages: match, group, and sort. Let's see how we can implement these in code.

Top 25 MongoDB Interview Questions and Answers for 2025 | DataCamp

Example “products” document:

```
[
  { "_id": 1, "product_id": "t2409", "amount": 250, "status": "done" },
  { "_id": 2, "product_id": "t2009", "amount": 300, "status": "done" },
  { "_id": 3, "product_id": "t1309", "amount": 150, "status": "pending" },
  { "_id": 4, "product_id": "t1919", "amount": 480, "status": "done" },
  { "_id": 5, "product_id": "t5459", "amount": 120, "status": "pending" },
  { "_id": 6, "product_id": "t3829", "amount": 280, "status": "done" }
]
```

POWERED BY  datacamp

- `$match` : To filter documents based on a condition
- `$group` : This groups the data and applies aggregation operation
- `$sort` : Order the output documents as you need

Example:

```
db.products.aggregate([
  { $match: { status: "completed" } },
  { $group: { _id: "$product_id", totalAmount: { $sum: "$amount" },
  { $sort
  });
```


POWERED BY  datacamp

How do you perform SQL join equivalent in MongoDB?

MongoDB provides aggregation operators like `$lookup` to perform SQL equivalent joins.

Syntax:

```
db.collection_1_name.aggregate([
  {
    $lookup: {
      from: "collection_2_name", // The other collection to join with
      localField: "field_in_collection_1", // The field on which you want to join
      foreignField: "field_in_collection_2", // The field from the second collect
      as: "result_field" // The name of the new field to store the joined result
    }
  }
])
```


POWERED BY  datacamp

Example:

Say you have order and product collections with data as follows:

“Orders” collection:

```
[
  { "_id": 1, "product_id": 101, "order_amount": 250 },
  { "_id": 2, "product_id": 102, "order_amount": 300 },
  { "_id": 3, "product_id": 101, "order_amount": 150 }
]
```


POWERED BY  datacamp

“Products” collection:

```
[
  { "_id": 3789, "product_id": 102, "product_price": 100},
  { "_id": 3970, "product_id": 103, "product_price": 297},
]
```

POWERED BY  datacamp

Top 25 MongoDB Interview Questions and Answers for 2025 | DataCamp

POWERED BY  datalab

Join operation:

```
db.orders.aggregate([
  {
    $lookup: {
      from: "products",
      localField: "customer_id",
      foreignField: "_id",
      as: "customer_info"
    }
  }
])
```

POWERED BY  datalab

How do you model a one-to-many relationship in MongoDB?


You can create a data model that uses embedded documents to describe a one-to-many relationship. For instance, a single team can have multiple employees, so you can embed them as follows:

Data:

```
// team details
{
  _id: "DataScience"
  company_name: "DataCamp"
  team_name: "Data leaders"
}

// employee one
{
  name: "John"
  employee_id: "t009456"
  email: johnsmith@datacamp.com
}


// employee two
{
  name: "Emily"
  employee_id: "t0068ms"
  email: emilyjones@datacamp.com
}
```

POWERED BY  datalab

One-to-many embedded document:

```
{
  "_id": "DataScience",
  "company_name": "DataCamp",
  "team_name": "Data leaders",
  "employees": [
    {
      "name": "John",
      "employee_id": "t009456",
      "email": "johnsmith@datacamp.com"
    },
    {
      "name": "Emily",
      "employee_id": "t0068ms",
      "email": "emilyjones@datacamp.com"
    }
  ]
}
```

Top 25 MongoDB Interview Questions and Answers for 2025 | DataCamp


POWERED BY  datalab

MongoDB Scenario-Based Interview Questions for DBAs

Interviewers will put you in challenging situations and assess how you would approach them. This helps them understand your ability to handle real-time problems while working with MongoDB.


Imagine you are performing a search operation on the database “orders” as follows. How do you debug why the query is slow?

```
df.orders.find({
  customer_id: 'yop89'
  ordered_items: {
    product_id: 'toi45'
    product_id: 'tac87'
  }
});
```

POWERED BY  datalab


First, you should set the profiling level 1 to select only slow-running queries:

```
db.setProfilingLevel(1, { slowms: 100 }); // Logs queries slower than 100ms
```

POWERED BY  datalab

Now, the following code gives us extra details like the type of operation, time taken, and keys or documents scanned. This information helps you find the slow-running queries:

```
db.system.profile.find({ millis: { $gt: 100 } }).sort({ millis: -1 }).limit(1000);
```

POWERED BY  datalab


In the above scenario, how do you improve the performance of slow-running queries?

From the above code's output, if the query scans the entire collection, create indexes on customer_id and ordered_items. Indexes can reduce the number of documents scanned, improving query execution time.

Build data and AI skills

50%

Sale ends in 1d 05h 52m 46s

POWERED BY  datalab

In sharding, if one shard is overloaded and the others remain idle. How do you balance this?

Two potential issues could be the inappropriate selection of the shard key and uneven data distribution across shards.

Fix 1: Choose the right shard key

- Choose high cardinality columns as shard keys. That is, a shard key should have many unique values.
- If your workload is write-heavy, ensure your shard key doesn't direct all writes to a single shard.
- Choose a shard key that aligns with your most frequent queries. For instance, if your queries often join on a specific field, that field could be an effective shard key.

Fix 2: Rebalance the uneven distribution

This command below provides the overview of data distributed across shards. If the chunks are not well distributed, enable the balancer.

```
sh.status()
```



POWERED BY datalab

The command below gets the balancer's state. If it's disabled, use the later command to enable it.

```
sh.getBalancerState()  
sh.enableBalancer("db_name.collection_name")
```



POWERED BY datalab

What challenges might you face while migrating from RDBMS to MongoDB?

Mapping relational database operations to their counterparts in NoSQL can be challenging.

- For instance, SQL joins are not straightforward in MongoDB; instead, you need to use the aggregation framework to achieve similar functionality.
- Another challenge is the data stored in structured tables will need transformation to fit into MongoDB's BSON storage format.
- Additionally, while RDBMS provides robust ACID compliance, MongoDB only offers document-level ACID properties, which means complex ACID transactions may require extra attention.

Tips for Preparing for a MongoDB Interview

Cracking a MongoDB interview requires a deep understanding of concepts and practical expertise in the subject. It's essential to gear up your knowledge of [NoSQL databases](#), master [MongoDB syntax](#), and have exposure to multiple MongoDB failure scenarios and learn to resolve them.

Here are my best tips to help you pass the next MongoDB interview:

- **Be good at fundamentals:** Whether you apply to an entry-level role or a senior position, fundamentals are always tested. So, understand the MongoDB architecture, storage type, syntax, and supported operations.
- **Advanced topics:** You should know how advanced concepts like replication, aggregation, sharding, and indexing work in MongoDB and be able to implement them if required.
- **Real-world scenarios:** Interviewers often test how you manage MongoDB databases in various workplace scenarios. For instance, you may be asked how to transition an existing RDBMS database to MongoDB. In more advanced interviews, you might be given a scenario and required to write code to address it.
- **Read past experiences:** Learning about previous interview experiences at the company gives you an idea of the type of questions to expect and their interview patterns, allowing you to prepare accordingly. I personally use platforms like Glassdoor, Blind, YouTube, and LinkedIn for previous interview experiences.
- **Mock interviews:** Mock interviews help you assess your strengths and weaknesses. They also prepare you for the interview environment, which helps you feel confident and comfortable during the actual interview.
- **Soft skills:** The ability to communicate complex topics is essential to impress the interviewer. So, work on your communication and presentation skills.
- **Certifications:** MongoDB certifications serve as proof of your expertise. They deepen your understanding of the subject and enhance your chances of getting hired. Here is [a complete guide to getting a MongoDB certification](#).