MongoDB Tutorial for Beginners

Introduction

MongoDB is presently the most popular NoSQL document database and is an essential tool for developing web and mobile apps. MongoDB lacks a fixed schema, which makes it an ideal solution for scaling up quickly when dealing with enormous datasets. Education in MongoDB is a necessity for someone looking to pursue a career in Data Science, Web Development, or Cloud Computing.

Ready to take on the future of data storage? See our entire **MongoDB course syllabus** here to learn all you can do with this powerful technology!

Why Students or Freshers Learn MongoDB

MongoDB knowledge can be a major plus in today's tech environment. Here's why it is essential in your profession:

- **Industry Demand:** MongoDB is a leading NoSQL database with very high demand in roles such as Full Stack Development, Data Science, and DevOps in large companies.
- **Flexible Data Modeling:** Its document-oriented structure is based on a JSON-like format called “BSON,” which corresponds one-to-one with objects in languages such as JavaScript and Python, making development time quicker.
- **Scalability/Performance:** Offers support for horizontal scaling/sharding and high availability/replica sets, which are critical capabilities for developing apps with large datasets and/or a heavy volume of traffic.
- **Simplified Learning Curve:** The non-structural schema and basic CRUD operations make it simpler for someone learning databases to grasp compared to learning SQL.
- **Modern Ecosystem:** It can be easily integrated with MERN/MEAN stacks and cloud platforms such as MongoDB Atlas, which puts you on the right path to a cloud-native development career.

Ace Your Tech Screening! Ace your tech screening with our compilation of must-know **MongoDB interview questions and answers** to get your dream job!

Step-by-Step MongoDB Tutorial for Beginners

MongoDB is a very flexible and scaleable NoSQL database which holds information in a JSON-like format called BSON documents. In this tutorial, you will see everything from install to basic manipulation of your data.

Step 1: Installation & Setup

To get started with MongoDB, you have to install both MongoDB Database Server and MongoDB Shell/Compass. For this tutorial, I will describe how to install MongoDB Community Server Edition, which is available for free.

1.1. Downloading MongoDB Community Server

- **Go to the Official Site:** MongoDB Download Center: MongoDB Website
- **Select Version:** Select the current Stable version, such as 7.0 or 6.0.
- **Select Platform:** Select your platform (Windows, macOS, or Linux).
- **Select Package:** Select the Windows MSI package.
- **Download:** Press the Download button.

1.2. Installing MongoDB on Windows using MSI

1. **Running the Installer:** Double-click on the downloaded .msi file.
2. **Setup Type:** Select “Custom” to pick individual components or “Complete” for a standard install. “Complete” is recommended for newbies.
3. **Service Configuration:** Crucially, on “Service Configuration” screen:
    - Make sure “Run service as Network Service user” is checked.
    - It is important to notice the Data Directory and Log Directory. pour quarterback, Steve Sarkis
4. **Installing MongoDB Compass(Optional but Recommended):** Select the box to install MongoDB Compass, which will allow you to work with MongoDB using a comfortable graphical interface.
5. **Finish:** Enter your name in the box and click Install and finish.

1.3. Environment Path Configuration Setup (Optional but Very Helpful)

To execute MongoDB commands from any terminal window, you have to include your MongoDB bin directory in your system Environment Variables:

1. **Find Bin Folder Location:** Find your Bin Folder with these steps: Go to your installation directory. Common paths include: *C:\Program Files\MongoDB\Server\<Version>\bin.*
2. **Copy Path:** A complete path to a directory can be copied using this function.
3. **Modify Environment Variables:** Type “Environment Variables” in the Windows search box and click “Edit the system environment variables”.
4. **Edit Path:** In the Properties of System box, click Environment Variables. In the System Variables box, click Path and click Edit.
5. **Add New Path:** Click on ‘New’ and enter the path you have copied for MongoDB bin. Click ‘OK’ in each window.

1.4. Starting the Server and Shell

1. **Server Starting Service (Windows Service):** The installation process will start MongoDB Server automatically with Windows Service Tools. To check if it is started successfully or not, search for “Services” with “MongoDB Server (Version)” in Services.
2. **Accessing the Shell:** Open a command prompt or terminal. Type in:

Mongosh

This starts up the MongoDB Shell instance, “mongosh”, and attaches to a local MongoDB instance running on port 27017. A prompt such as “test>” or “>” will appear in your shell.

Step 2. MongoDB Core Concepts

Now you can start to learn more about MongoDB by learning its core concepts.

SQL Concept	MongoDB Concept	Description

Database	Database	A physical container for collections.
Table	Collection	A group of MongoDB documents. Equivalent to an SQL table.
Row	Document	A single record, stored in BSON (JSON-like) format. Equivalent to an SQL row.
Column	Field	A key-value pair within a document. Equivalent to an SQL column.
JOIN	Embedded/Reference	Data relationships are handled by nesting documents or using references.

2.1. Documents (The Core Data Unit)

A document is considered the basic unit of data. They are elastic, with different documents in a given collection being able to have different fields.

// Example Document (JSON Format)

{

```
    "_id": ObjectId("60d5ec4981180b5e28e67a1c"),
    "name": "Alice Johnson",
    "age": 30,
    "city": "New York",
    "interests": ["reading", "hiking", "coding"],
    "address": {
        "street": "123 Main St",
        "zip": "10001"
    }
}
```

_id Field: Every MongoDB document must have a unique _id field. In cases where you do not specify an _id field, MongoDB will automatically create a unique id using an ObjectId.

Step 3: Important MongoDB Shell Commands (CRUD Operations)

The following commands are performed in a mongosh terminal.

3.1 Database Operations

Command	Description	Example

show dbs	Lists all databases on the server.	show dbs
use <dbname>	Creates and/or switches to a specified database.	use userDB
db	Displays the name of the currently selected database.	db
db.dropDatabase()	Permanently deletes the current database.	db.dropDatabase()

3.2 Collection Operations

Command	**Description**	**Example**
show collections	Lists all collections in the current database.	show collections
db.createCollection()	Explicitly creates a new collection.	db.createCollection(“products”)

3.3. Creating Documents (CREATE)

The insertOne() and insertMany() methods can be used to insert new documents into a MongoDB database.

```
// Insert a single document
db.users.insertOne({
    name: “John Doe”,
    age: 25,
    status: “active”
});
// Insert multiple documents
db.products.insertMany([
    { name: “Laptop”, price: 1200, category: “Electronics” },
    { name: “T-shirt”, price: 25, category: “Apparel” }
]);
```

3.4. Reading Documents(READ)

The main query function in MongoDB is find() with a query filter parameter.

Method	Description	Example
db.collection.find()	Returns all documents in the collection.	db.users.find()

db.collection.find(<query>)	Returns documents that match the query filter.	db.users.find({ age: 25 })
db.collection.findOne()	Returns the first document that matches the criteria.	db.users.findOne({ status: “active” })
db.collection.find().pretty()	Displays results in a readable JSON format.	db.users.find().pretty()

Using Query Operators: Very frequently you will have to execute more complicated queries using operators ($).

Operator	**Description**	**Example**
$eq	Equal to (default if no operator is used)	{ age: { $eq: 30 } }
$gt	Greater than	{ price: { $gt: 100 } }
$lt	Less than	{ price: { $lt: 50 } }
$ne	Not equal to	{ status: { $ne: “inactive” } }

$in	Value is in an array	{ category: { $in: [“Apparel”, “Home Goods”] } }

Example: Find products priced greater than 50.

db.products.find({ price: { $gt: 50 } })

3.5. Updating Documents (UPDATE)

The updateOne() and updateMany() methods update existing documents in a MongoDB database. They need three parameters:

1. Filter (which documents need updating)
2. Update Operator (how to update them)
3. Options (optional).

The most common Update Operator is $set.

Operator	**Description**	**Example Use**
$set	Sets the value of a field. Creates the field if it doesn’t exist.	{ $set: { status: “inactive” } }
$inc	Increments the value of a field by a specified amount.	{ $inc: { quantity: 1 } }

$unset	Removes a specified field from a document.	{ $unset: { address: “” } }

Example: Update John Doe’s status to ‘pending’.

db.users.updateOne(

{ name: “John Doe” }, // Filter (find John Doe)

{ $set: { status: “pending”, lastUpdated: new Date() } } // Update (set new status and date)

);

3.6. Deleting Documents (DELETE)

The deleteOne() and deleteMany() methods can be used to remove documents from a collection.

Method	**Description**	**Example**
db.collection.deleteOne(<filter>)	Deletes the first document matching the filter.	db.users.deleteOne({ name: “John Doe” })
db.collection.deleteMany(<filter>)	Deletes all documents matching the filter.	db.users.deleteMany({ status: “inactive” })

db.collection.deleteMany({})	CAUTION: Deletes ALL documents in the collection.	db.products.deleteMany({})

Step 4: Advanced Data Operations

As you gain experience with CRUD, these are some advanced concepts which are essential in real-world usage.

4.1. Indexing for Performance

Indexes are specialized storage structures that hold a small amount of information from a collection in a format that can easily be traversed. They greatly improve query performance but have a slight overhead cost during writing operations.

- **Extending an Index:** Use extendIndex(). A value of 1 represents increasing order.

// Create an index on the 'name' field

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

- **Checking Performance:** To check MongoDB query performance, a query can be explained using db.collection.explain().

db.users.find({ name: "Alice Johnson" }).explain("executionStats")

4.2. Aggregation Framework

The Aggregation Framework is a MongoDB mechanism for a variety of complex operations on data such as grouping, filtering, and calculating averages, very much like GROUP BY in SQL statements. The Aggregation Framework operates through a "pipeline" which consists of a series of "stages".

Common Aggregation Stages

- **$match:** Filters documents (similar to *find()*)
- **$group:** Groups documents by a key and computes aggregate values.
- **$sort:** Orders the documents.
- **$project:** Transforming documents into a specified format.

Example: Calculating the Average Price Per Category

```
db.products.aggregate([

  // 1. Group documents by 'category'

  { $group: {

    _id: "$category", // Group by the category field

    avgPrice: { $avg: "$price" }, // Calculate the average price

    totalItems: { $sum: 1 } // Count the number of items

  }},

  // 2. Sort the results by average price descending

  { $sort: { avgPrice: -1 } }

]);
```

4.3. Data Relationships: Embedding and Referencing

Because MongoDB is schemaless, it lacks primary/foreign key constraints. The relationships in MongoDB are facilitated by:

- **Embedding:** Storing all related information in one document. (Relatively good for one-to-one or one-to-few relationships.)

{ “user”: “Bob”, “address”: { “street”: “456 Oak”, “zip”: “90210” } }

- **Referencing:** Storing the _id of the related document. (Good for one-to-many or many-to-many relationships.)

{ “orderId”: 101, “userId”: ObjectId(“…”) }

MongoDB has been successfully installed, and you have learned basic concepts such as Documents, Collections, and Databases. Additionally, you have learned basic CRUD operations and touched on advanced subjects such as Indexing and Aggregations. Check out our **MongoDB Challenges and Solutions** to learn deeper.

Real Time Examples for MongoDB Tutorial for Beginners

Practical application of MongoDB knowledge with real-world examples can help reinforce your grasp of NoSQL principles and dynamic schema design. Here are a few examples:

Creating a Product Catalog (E-commerce)

The problem: An online shopping platform requires a very flexible and scaleable system to store different product information such as clothes, electronics, and books where every type of product possesses different characteristics.

The Solution: Use one MongoDB collection named products. The learners will practice:

- Inserting documents with varying fields. For example, an insert statement with a document containing information about ‘color’ and ‘size’, and another with ‘CPU’ and ‘RAM’.
- Applying $gt and $lt query operators for filtering products based on a price range.

- Using the Aggregation Framework ($group and $avg) to calculate average price per product category.

Skill Learned: Handling flexible schema, querying, and basic aggregation.

Logging Application Events (User Activity)

The Problem: A social media application must record all actions performed by each of its users with high insert throughput and under minimal schema checking.

The Solution: Create an events collection.

- insertOne() is used quite frequently in this case, since a very small amount of information needs to be recorded in a MongoDB collection, which is no more than the action type.
- Indexing a timestamp field to support quick time-domain queries.
- Application of $match and $group aggregation operations in identifying unique login entries on a daily basis.

Skill Learned: High-speed writes, time series index, and summarization of data using an aggregate function.

User Profile & Embedded Data (Social Media)

The Problem: Implement a system where a user's information and access settings such as notifications and themes are obtained in a single call.

The Solution: Embedding can be achieved using the Embedding data model in the "users" collection.

- Learners will be able Storing main information of a user and an array of addresses or preferences object inside the user document.

- Applying the $push operator to append a new element in an embedded array without fetching the whole document when a new address or preference is added.

Skill Learned: Modeling of embedded data, single query retrieval, and array manipulation.

Want to get started with implementation? Have a look at our interesting **MongoDB project ideas** to get your hands on your newfound NoSQL knowledge.

FAQs About MongoDB Tutorial for Beginners

1. What is MongoDB used for?

MongoDB can be used in a variety of applications such as content management systems, ecommerce product listings, social media sites, real time logging of data, and mobile apps where high scalability and fast iteration are desired.

2. Is MongoDB a SQL or NoSQL?

MongoDB is a NoSQL (Not Only SQL) database. It uses a document model to store data, contrasting with the relational structure of traditional SQL databases.

3. What is MongoDB vs. MySQL?

MongoDB is a NoSQL, document-oriented DB with a flexible schema, suitable for unstructured or dynamic datasets. MySQL is a relational/SQL DB with a fixed, inflexible schema, suitable for very structured, analytical datasets with complex joins.

4. Is MongoDB a DB or DBMS?

MongoDB is a DBMS. MongoDB is a NoSQL document database management system. MongoDB is used for managing the database (DB) in which all the data is being controlled.

5. What are the 4 types of database?

The four most common kinds of databases include: A) Relational/SQL database (SQL Server, PostgreSQL) B) Key/Value store database (Redis) C) Document-oriented database (MongoDB) D) Graph database (Neo4j)

6. Can I write SQL in MongoDB?

No, you can't write SQL in MongoDB. Instead, you'll make use of MongoDB Query Language (MQL), which is composed of JSON-like commands and methods such as "find," "aggregate," and so on, written in JavaScript.

7. What language is used in MongoDB?

MongoDB stores documents in a format called Binary JSON, which is an extension of JSON. MongoDB Query Language, or simply MQL, is executed with a series of JavaScript commands within Mongo Shell, which is referred to as "mongosh" in MongoDB versions 1.

8. Is MongoDB hard to learn?

MongoDB is not very difficult to learn when compared to a full SQL database because a MongoDB document relates directly to an object in programming languages such as Python or JavaScript.

9. Does MongoDB require coding?

Yes, because in order to interact with MongoDB, one must have command knowledge of MongoDB Query Language, which is a form of coding.

10. What is the salary in MongoDB?

The average **MongoDB salary for freshers in India** is approximately ₹11 lakhs per year, but it differs greatly depending on experience, a particular role such as Developer/DBA, and location.

Conclusion

You have successfully walked through the basic concepts of MongoDB, the adaptable, elastic, and fast database technology behind the most innovative applications in the world. With your expertise in document modeling and basic MQL operations, you have gained a skillset which is critical in web application development, research, and cloud technology. The future of data storage is non-relational, and you are all set to create solutions.

Want to become a certified expert? Join our complete **MongoDB course in Chennai** and learn advanced topics such as sharding, replication, and performance optimization.