GUVI | HCL

Skill Up. Level Up

MongoDB for Beginners

Your Guide to Modern Databases

GUVI | HCL
Skill Up. Level Up

MongoDB

MongoDB is a powerful NoSQL database that offers flexibility, scalability, and high performance for modern applications. Unlike traditional relational databases, MongoDB uses a document-oriented data model, allowing developers to store data in flexible, JSON-like structures. This approach makes it an excellent choice for handling large volumes of unstructured or semi-structured data, making it popular among developers building web applications, mobile apps, and cloud-based solutions.

In this beginner-friendly ebook, you'll learn the fundamentals of MongoDB, including how to set up a database, store and query data, and optimize performance. Whether you're a developer looking to integrate MongoDB into your project or someone exploring NoSQL databases for the first time, this guide will provide you with the essential knowledge to get started quickly and efficiently.

GUVI | HCL
Skill Up. Level Up

Table of Contents

1. Introduction

What is MongoDB?

MongoDB is a NoSQL database that stores data in a flexible, document-based format instead of traditional tables and rows. Unlike relational databases, which rely on structured schemas, MongoDB uses JSON-like documents to store data, making it more adaptable to changing application needs. It's widely used in modern web applications, big data solutions, and real-time analytics due to its scalability, performance, and ease of use.

Why Learn MongoDB?

MongoDB is one of the most popular databases for modern applications because it offers several advantages over traditional relational databases:

- **Flexibility:** Since MongoDB uses a schema-less structure, you can store different types of data without strict formatting.
- **Scalability:** It is designed to handle large amounts of data efficiently and can scale horizontally across multiple servers.
- **Speed:** Querying and retrieving data in MongoDB is often faster due to its document-based model.
- **Use Cases:** MongoDB is widely used in real-time applications, e-commerce platforms, IoT systems, content management, and mobile apps. Many companies, including Netflix, Uber, and Facebook, use MongoDB for its performance and adaptability.

Tools You'll Need

To work with MongoDB, you'll need one of the following tools:

- **MongoDB Compass** – A graphical user interface (GUI) that allows you to visualize, analyze, and manage your database without using code.

- **MongoDB Atlas** – A cloud-based platform that lets you host and manage MongoDB databases without setting up a local server.

- **MongoDB CLI** – A command-line tool for advanced users who prefer to interact with MongoDB through commands and scripts.

These tools make it easy to start using MongoDB, whether you're a beginner exploring databases or a developer building scalable applications.

2. MongoDB Basics

Core Concepts: Documents, Collections, and Databases

MongoDB follows a document-oriented approach, which differs from the traditional table-based structure of SQL databases. The three core components in MongoDB are:

- **Documents:** The fundamental data units in MongoDB, similar to rows in SQL databases. They are stored in a JSON-like format and can hold various types of data, including nested objects.

- **Collections:** A group of related documents, similar to tables in relational databases. However, unlike SQL tables, collections in MongoDB do not enforce a fixed schema, allowing flexibility in data storage.

- **Databases:** A container for collections, similar to how SQL databases store multiple tables. MongoDB allows multiple databases to be managed within a single instance.

This structure allows MongoDB to store complex and varied data efficiently while maintaining high performance and scalability.

JSON (JavaScript Object Notation): A lightweight, human-readable format used for data exchange. Example:

```
{
  "name": "Alice",
  "age": 25,
  "hobbies": ["reading", "gaming"]
}
```

BSON (Binary JSON): An optimized binary format that supports additional data types such as dates and integers. BSON is more efficient for storage and retrieval.

MongoDB internally converts JSON data into BSON, making queries faster while still allowing developers to work with familiar JSON-like syntax.

MongoDB Architecture (Replica Sets and Sharding)

MongoDB is designed for scalability and high availability through its distributed architecture. Two key features of MongoDB's architecture are:

- **Replica Sets:** A group of MongoDB instances that maintain the same data. One node acts as the primary (handling read/write operations), while the others are secondary replicas (used for failover and read scaling). If the primary fails, one of the secondaries is automatically promoted.

- **Sharding:** A method of distributing large datasets across multiple servers to handle high-traffic applications. Instead of storing all data on a single machine, MongoDB divides it into smaller parts (shards) and distributes them, improving performance and scalability.

These features make MongoDB an excellent choice for applications that require fault tolerance, scalability, and high-speed data access.

3. Setting Up MongoDB

Installing MongoDB Locally

To use MongoDB on your computer, you need to install it manually. Follow these steps:

1. **Download MongoDB** – Visit the official MongoDB website and download the Community Edition for your operating system.

2. **Install MongoDB** – Run the installer and follow the setup instructions. On Windows, you may need to add MongoDB to your system's PATH for easy access via the command line.

Start MongoDB – Once installed, start the MongoDB service using the command: mongod

3. This runs the MongoDB server, allowing you to interact with databases and collections.

After installation, you can use the MongoDB Shell (mongosh) or other tools like Compass to interact with your database.

Introduction to MongoDB Atlas (Cloud Setup)

MongoDB Atlas is a cloud-based database service that allows you to create and manage MongoDB databases without installing anything locally. It provides built-in scalability, security, and backup features.

To get started:

1. Sign up – Go to MongoDB Atlas and create a free account.

2. Create a Cluster – Choose a free-tier cluster, select a cloud provider, and configure your settings.

3. Connect to Your Database – Use MongoDB Compass, the shell, or your application to connect using the connection string provided in Atlas.

Atlas is a great option for deploying production-ready applications without worrying about infrastructure management.

Setting Up MongoDB Compass for GUI-Based Interactions

MongoDB Compass is a graphical user interface (GUI) that makes it easier to interact with your MongoDB databases. Instead of writing queries in the shell, you can use Compass to visualize and manage data.
To set it up:

1. Download and Install Compass – Get it from the official MongoDB website.

Connect to MongoDB – If you're using a local database, connect using the default connection string:
arduino
mongodb://localhost:27017

2. For Atlas, use the connection string provided in your Atlas dashboard.

3. Explore Your Database – Use Compass to create, modify, and delete databases, collections, and documents visually.

Compass is useful for beginners who prefer a no-code approach to managing MongoDB.

First Steps: Creating a Database and a Collection

Once MongoDB is set up, you can start working with databases and collections. Here's how to do it:

Using MongoDB Shell (CLI):

- Open the shell by running: mongosh
- Create a new database: use myDatabase
- Create a collection and insert a document: db.users.insertOne({ name: "Alice", age: 25 })

Using MongoDB Compass:

1. Click "Create Database", enter a name, and add a collection.
2. Click "Insert Document" to add data in JSON format.

4. CRUD Operations

CRUD (Create, Read, Update, Delete) are the four fundamental operations used to manage data in MongoDB. These operations allow you to **add, retrieve, modify, and remove documents** in a database. Let's go through each one with real-life examples.

a. Create: Adding Documents

In MongoDB, data is stored as documents inside collections. You can insert a single document or multiple documents at once.

Example: Storing User Data

Imagine you're building a user registration system for a website. To add a new user, you can use the following command:

```
db.users.insertOne({
  name: "Alice",
  age: 25,
  email: "alice@example.com",
  created_at: new Date()
})
```

To insert multiple users at once:

```
db.users.insertMany([
  { name: "Bob", age: 30, email: "bob@example.com" },
  { name: "Charlie", age: 28, email: "charlie@example.com" }
])
```

Real-life Use Case: User sign-ups on a website or app.

b. Read: Querying Data (Basic Queries with Filters)

You can retrieve data using the find() method, which allows you to filter and fetch specific documents.

Example: Finding a User by Name

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

This returns:

{ "_id": ObjectId("..."), "name": "Alice", "age": 25, "email": "alice@example.com", "created_at": "..." }

Example: Filtering Users by Age Greater Than 25

db.users.find({ age: { $gt: 25 } })

Real-life Use Case: Searching for users based on their age, location, or preferences in a web application.

c. Update: Modifying Documents

To update existing documents, MongoDB provides the updateOne() and updateMany() methods.

Example: Updating a User's Email

db.users.updateOne(
  { name: "Alice" },

```
{ $set: { email: "alice.new@example.com" } }
)
```

If you need to update multiple users at once:

```
db.users.updateMany(
  { age: { $gt: 25 } },
  { $set: { status: "premium user" } }
)
```

Real-life Use Case: Updating user profiles, modifying order statuses, or changing subscription plans.

d. Delete: Removing Documents

To delete records, use deleteOne() or deleteMany().

Example: Deleting a User by Name

```
db.users.deleteOne({ name: "Alice" })
```

To delete all users older than 50:

```
db.users.deleteMany({ age: { $gt: 50 } })
```

Real-life Use Case: Removing inactive accounts, deleting expired promotions, or clearing outdated logs.

5. Data Modeling

Basics of schema design in MongoDB

MongoDB follows a schema-less approach, meaning documents in a collection do not have to follow a strict structure like SQL databases. However, designing an efficient schema is crucial for performance and scalability.

Key Principles of Schema Design in MongoDB:

- Design for queries, not just storage – Structure your data based on how it will be retrieved.
- Avoid unnecessary nesting – While MongoDB supports embedded documents, excessive nesting can make updates complex.
- Use indexes wisely – Indexing fields speeds up queries but increases storage overhead.
- Choose between embedding and referencing – Use embedded documents for fast access and references for scalability.

Relationships: Embedded vs. Referenced

MongoDB does not enforce foreign key relationships like SQL databases, but you can define relationships in two ways:

1. Embedded Documents (Denormalization)

- Stores related data inside the same document.
- Best for: Data that is frequently accessed together.

Example: Storing an order and its items together

```
{
  "orderId": 123,
  "customer": "Alice",
  "items": [
    { "product": "Laptop", "price": 1000 },
    { "product": "Mouse", "price": 50 }
  ],
  "total": 1050
}
```

Pros: Faster reads, fewer queries.
Cons: Can lead to large documents if nested too deeply.

2. Referenced Documents (Normalization)

- Stores related data in separate collections and links them using an ID.
- Best for: Data that needs to be updated frequently or shared across multiple documents.

6. Conclusion

Recap of what was learned

To give you a better understanding, we have talked about the basics of MongoDB, some of the top tools you should know while handling complex databases. Next, we read about the steps required to set up a MongoDB environment.

We also read about CRUD operations, which plays a vital role in creating, reading, updating, and deleting data, i.e., performing operations in databases. We also read about data modeling, schema design and its key principles.

Next steps for learning (e.g., advanced aggregation, analytics, and sharding)

Now, if you want to explore advanced concepts of MongoDB, you must go for an advanced course which not only clears your theoretical knowledge, but gives you a practical understanding of it.

Check out for Mastering MongoDB course by GUVI which explains in-depth concepts with practical examples.

If you're looking for a complete guide on FullStack Development, do check our Zen Class - Full Stack Development Program which talks about the fundamentals of JavaScript, MongoDB, ExpressJS, NodeJS, ReactJS, AWS, and the core tools and technologies you must know. You'll also learn working on real-world projects, and get placement guidance.

About Zen Class

Zen Class is GUVI's specially curated learning platform that incorporates all the Advanced Tech Career Courses like Full Stack Web Development, Data Science, Automation Testing, Big Data & Cloud Analytics, UI/UX Designing, and more. Zen Class provides the best industry-led curriculum programs with assured placement guidance

Zen Class mentors are experts from leading companies like **Google, Microsoft, Flipkart, Zoho & Freshworks.** Partnered with 600+ tech companies, your chances of getting a high-paying tech job at top companies increases.

Why choose Zen Class?

- Assured Placement Guidance
- IIT-M Pravartak Certification for Advanced Programming
- Ease of Learning in native languages such as Hindi & Tamil
- A rich portfolio of real-world Projects
- Self-paced Online Classes
- Industry-led Curriculum
- 600+ Hiring Partners
- Unlimited access to course resources
- Excellent Mentor Support
- Easy EMI options

Now, let's know a bit about GUVI.

About GUVI

GUVI (Grab Ur Vernacular Imprint), an IIT-Madras Incubated Company is First Vernacular Ed-Tech Learning Platform. Introduced by Ex-PayPal Employees Mr. Arun Prakash (CEO) and Mr. SP Balamurugan, and late Sridevi Arun Prakash (co-founders) and driven by leading Industry Experts, GUVI empowers students to master on-demand technologies, tools, and programming skills in the comfort of their native languages like Hindi, Tamil, Telugu, English etc.

Personalized Solutions

End-to-end personalized solutions in online learning, upskilling, and recruitment.

Empowering Learners

Empowering learners with tech skills in native languages.

Gamified Learning

Gamified, bite-sized videos for an interactive learning experience.

Accreditations & Partnerships

Want to know more about GUVI? Visit our website today!

GUVI | HCL
Skill Up. Level Up

Thank You

www.guvi.in