DAY-13

Introduction to Databases and Basics of MongoDB

Databases *are like storage spaces where we keep and manage data. Think of them as digital filing cabinets, where each piece of information is stored in a way that makes it easy to access and organize. Now,* ***MongoDB*** *is a popular type of database that stores data in a more flexible way compared to traditional databases.*

- ***What is a Database?***
  - A database is a collection of data organized in a way that allows you to easily store, retrieve, and manage it. There are two main types of databases:

    1. ***Relational Databases (like MySQL, PostgreSQL):*** These organize data into tables with rows and columns. It's like an Excel sheet where each row is a piece of information, and the columns define the types of data (like name, age, etc.).

    2. ***Non-relational Databases (NoSQL) (like MongoDB):*** These are more flexible. Instead of rows and columns, data can be stored in different forms (like documents, key-value pairs, etc.), and they don't need to follow a strict structure.

- ***Why Use MongoDB?***
  - MongoDB is one of the most popular NoSQL databases. Unlike traditional databases, it stores data in a more flexible format, which is great for applications that need to handle different types of data.

  - ***Here's why MongoDB is popular:***

    - ***Flexible Schema:*** You don't have to define exactly what data will look like before storing it. You can change the structure of your data as your application evolves.
    - ***Scalability****:* MongoDB can handle huge amounts of data and traffic, which makes it great for large applications.
    - ***Easy to Use:*** It's simple to get started with MongoDB, and you don't need to worry too much about complicated configurations.

DAY-13

MongoDB Data Structure

MongoDB stores data in two main things:

1. ***Database:*** A MongoDB server can hold many databases. Think of a database like a folder where related information is stored.

2. ***Collection:*** A collection is like a table in a traditional database. It holds documents (pieces of data).

3. ***Document:*** A document is like a row in a traditional table. It stores individual pieces of data. These documents are in a format called BSON (similar to JSON, a popular data format).

Example of a MongoDB Document:

A document might look like this:

json:
```json
{
  "_id": "1",
  "name": "Het Dabhi",
  "age": 20,
  "email": "hetdabhi@gmail.com",
  "hobbies": ["gaming", "traveling"]
}
```

- **_id:** A unique identifier for each document. MongoDB generates this automatically for you.
- **Other fields:** These are custom pieces of data, like the person's name, age, email, etc.

DAY-13

How to Get Started with MongoDB

Here's how you can start working with MongoDB:

1. ***Install MongoDB:*** You can install it on your computer, or you can use a cloud version called MongoDB Atlas. The installation is simple, and MongoDB provides guides for setting it up.

2. ***MongoDB Shell:*** Once installed, you can use the MongoDB shell to interact with your database. It's a command-line tool where you can run commands to manage your data.

Basic MongoDB Operations

Let's look at some simple commands to get started with MongoDB:

1. ***Create a Database***
   - To create a new database, you can use the use command:

     ```
     use myDatabase
     ```
   - This creates a new database called myDatabase.

2. ***Create a Collection***
   - A collection is where you store your data. You can create one like this:

     ```
     db.createCollection("users")
     ```
   - MongoDB will create a collection automatically the first time you add data to it.
3. ***Insert Data***
   - To add data (or a document) to a collection, you use insertOne() or insertMany():

     ```
             db.users.insertOne({
              name: "Het Dabhi",
              age: 20,
              email: "hetdabhi@gmail.com"
     })
     ```
   - This inserts a new document into the users collection.

DAY-13

4. ***Query Data***
   - To look at the data you've added, you can use find(). This retrieves all documents from the collection:

```
db.users.find()
```

   - To find a specific user by their name:

```
db.users.find({ name: "Het Dabhi" })
```

5. ***Update Data***
   - If you want to update someone's details, use updateOne():

```
          db.users.updateOne(
           { name: "Het Dabhi" },
           { $set: { age: 20 } }
)
```

   - This will change Het Dabhi age to 20.

6. ***Delete Data***
   - If you need to delete a user, you can use deleteOne():

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

   - This deletes the first user with the name "John Doe."