Introduction to MongoDB

Presented By

Lekshmi M B

K B A

Understanding Web Applications

Data Needs and Interactions

Web apps rely heavily on databases.

K B A

What is a Database?

Understanding data storage and management

A database is an organized collection of data that allows for efficient storage, retrieval, and management of information, crucial for various software applications and systems to function effectively.

Example: Student records, banking systems, e-commerce websites.

K B A

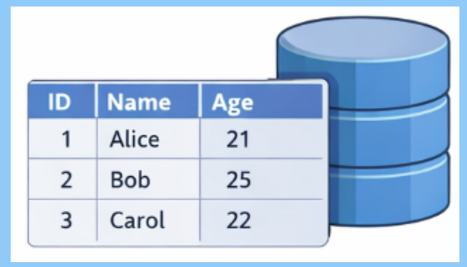
Types of Databases

Understanding various database categories

K B A

RELATIONAL DATABASES

RDBMS store data in structured tables with predefined relationships, ensuring data integrity and enabling complex queries.

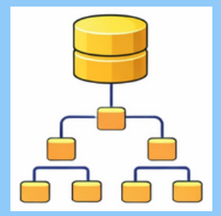


NOSQL DATABASES

NoSQL databases provide flexible data models, allowing unstructured data storage and horizontal scaling for high-performance applications.

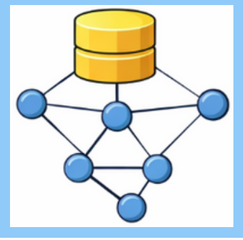
HIERARCHICAL DATABASES

Data stored in a tree-like structure (parent-child)



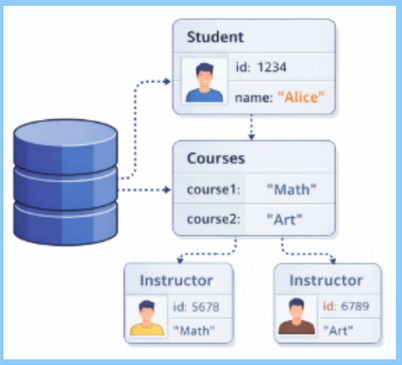
NETWORK DATABASES

Data stored as a graph-like structure with multiple relationships



OBJECT-ORIENTED DATABASES

Data stored as objects



RDBMS vs NoSQL

Key distinctions between database types

DATA MODEL

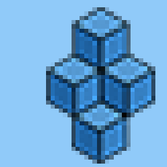
RDBMS uses structured tables with rows and columns, while NoSQL employs flexible documents or key-value pairs.

SCHEMA RIGIDITY

RDBMS requires a fixed schema, whereas NoSQL allows for dynamic schemas, adapting easily to changing data requirements.

SCALABILITY

RDBMS typically scales vertically, while NoSQL databases can scale horizontally, distributing data across multiple servers efficiently.



Examples

SQL

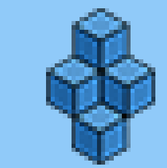
- MySQL
- PostgreSQL
- Oracle
- Microsoft SQL Server

ID	Name	Age
1	Alice	21
2	Bob	25
3	Carol	22

NOSQL

- MongoDB
- CouchDB
- Redis

```
{
  "name": "Alice",
  "age": 21,
  "course": "MERN"
}
```



What is MongoDB?

A NoSQL document database explained

MongoDB is a **document-oriented NoSQL database** designed for flexibility and scalability. It stores data in JSON-like documents, enabling efficient handling of complex data structures and rapid development.

MongoDB structure:

Database → Collection → Document

RDBMS structure:

Database → Table → Record

Example Document:

{ name: 'Alice', age: 21, course: 'MERN' }

SQL vs MongoDB

SAMPLE SQL QUERIES (RDBMS)

- **Create Table:**

CREATE TABLE Students(id INT, name VARCHAR(50));

- **Insert:**

INSERT INTO Students VALUES (1,'Alice');

- **Read:**

SELECT * FROM Students;

SAMPLE MONGODB QUERIES (NOSQL)

- **Create Database:**

use collegeDB

- **Insert Document:**

db.students.insertOne({name:'Alice', age:21})

- **Find**:

db.students.find()

MongoDB Shell Commands

Essential commands for managing databases

SHOW DATABASES

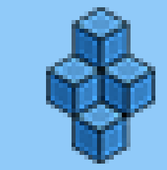
This command lists all databases available in the MongoDB server, helping users identify existing databases.

USE DATABASE

The use command switches the context to a specific database, preparing it for further operations and queries.

SHOW COLLECTIONS

This command displays all collections within the currently selected database, providing insight into the data structure.



MongoDB Shell Commands

CRUD Commands

CREATE

```
db.myCollection.insertOne({name:'Alice'})
```

READ

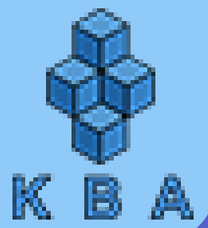
```
db.myCollection.find()
db.myCollection.findOne({name:'Alice'})
```

UPDATE

```
db.myCollection.updateOne({name:'Alice'}, {$set:{age:22}})
```

DELETE

```
db.myCollection.deleteOne({name:'Alice'})
```



MongoDB Shell (mongosh)

mongosh is the interactive command-line interface for MongoDB

USES:

- Run queries directly in terminal
- Manage databases and collections
- Perform CRUD operations quickly

K B A

MongoDB Compass

Exploring Data Visually

MongoDB Compass is the official GUI tool for MongoDB

FEATURES:

- Visualize databases, collections, and documents
- Build queries using a user-friendly interface
- Monitor performance and indexes
- Import/export data easily
- Best for beginners and database administrators.

K B A

Ways to Use MongoDB

Flexible options for deployment and usage

LOCAL INSTALLATION

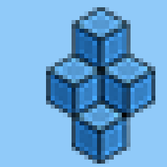
Installing MongoDB locally gives developers complete control and allows for a hands-on experience with database management.

MONGODB ATLAS

MongoDB Atlas is a cloud service that provides a managed database solution, simplifying deployment and scaling for applications.

DOCKER CONTAINER

Using MongoDB within Docker containers streamlines the setup process and ensures consistent environments across development and production stages.



Next Steps

Actionable Ideas for Learning

INSTALL MONGODB

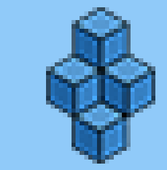
Download and install MongoDB locally to familiarize yourself with its features and functionalities for your projects.

EXPLORE MONGOSH

Practice using mongosh commands to efficiently interact with your databases and manage data effortlessly.

BUILD WITH MONGOOSE

Start developing applications using Mongoose to take advantage of its schema enforcement and streamlined MongoDB interactions.



Introduction to Mongoose

Object Document Mapper for Node.js

Mongoose is a Node.js library that simplifies interactions with MongoDB. It enables developers to define schemas, enforce data validation, and perform CRUD operations seamlessly in Node.js applications.

Connecting MongoDB

Understanding Connection URL Structure

To connect MongoDB with Express, you need a connection string that includes parameters such as username, password, and cluster information, ensuring secure and efficient access to your database.

```
mongoose.connect('mongodb://127.0.0.1:27017/test')
 .then(() => console.log('Connected to MongoDB!'))
 .catch(err => console.error('Could not connect to MongoDB...', err));
```

Mongoose Schema

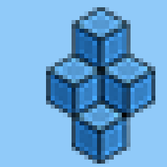
Defining Structure and Data Types

A **Mongoose schema** outlines the structure of documents within a collection, including fields and their data types, providing a clear framework for data validation and enforcement in applications.

```
// Define Schema (Structure of Document)
const studentSchema = new mongoose.Schema({
  name: { type: String, required: true },
  age: { type: Number },
  email: { type: String, unique: true },
  grade: { type: String, enum: ["A", "B", "C"] }
});

// Create Model (Collection)
const Student = mongoose.model("Student", studentSchema);
```

- **Schema** defines fields, data types, and validation rules
- **Model** creates a MongoDB collection and allows CRUD operations



Mongoose Methods

Key Functions for CRUD Operations

SAVE()

The **save() method** is used to create or update documents in the database, ensuring changes are stored persistently.

```
await student.save();
```

FIND()

The **find() method** retrieves documents from a collection based on specified criteria, returning all matching results for further processing.

```
await Student.find()
```

DELETE

The **findOneAndDelete() method** finds the first document matching a condition and deletes it from the collection in a single operation.

```
await
Student.findOneAndDelete(
{ name: req.params.name });
```

K B A

Multer

File Upload

Multer is a Node.js middleware used with Express to handle file uploads.

It processes data sent using **multipart/form-data,** which is the format browsers use when uploading files (images, PDFs, videos, etc.).

- **multer.memoryStorage()** creates a storage engine. It tells Multer to store uploaded files in RAM (memory) instead of saving them to the disk.
- Uploaded files are kept as Buffer objects in memory.
- You can access the file like this: **req.file.buffer**

Sharp is a Node.js image processing library used to resize, compress, crop, or convert images on the server.

Any Queries????

KBA

THANK YOU

Let’s move on to the Hands-on Session!!!