MongoDB

Laiba Imran

What is MongoDB?

- **Definition:** MongoDB is a popular, open-source NoSQL database that uses a document-oriented model.

- **Type:** It’s part of the NoSQL family, storing data in flexible, JSON-like documents instead of tables.

- **Use Cases:** Ideal for applications requiring fast development, scalability, and flexible data structures.

SQL vs. NoSQL

- SQL (Relational Database): Structured schema, uses tables and rows, rigid relationships.
- NoSQL (Non-relational Database): Flexible schema, uses collections and documents, supports dynamic data models.
- MongoDB: A NoSQL database that provides high flexibility and scalability.

MongoDB Key Concepts

- Database: Holds collections of documents.
- Collection: Equivalent to a table in relational databases but without a predefined schema, has dynamic schema.
- Document: Equivalent to a row in relational databases. A single data record in BSON (Binary JSON) format, similar to JSON.

BSON vs JSON

- **Format:**
  - BSON: Binary format
  - JSON: Text format
- **Data Types:**
  - BSON: Supports additional data types (e.g., Date, ObjectId)
  - JSON: Limited to basic data types (e.g., String, Number, Boolean)
- **Size:**
  - BSON: Generally larger due to additional metadata
  - JSON: More compact, especially for simple data
- **Encoding:**
  - BSON: Uses a binary encoding, not human-readable
  - JSON: Text-based encoding, human-readable
- **Speed:**
  - BSON: Faster to encode and decode in binary form
  - JSON: Slower to parse and serialize due to text processing
- **Use Cases:**
  - BSON: Primarily used in MongoDB for storage and transmission
  - JSON: Widely used for APIs and data interchange

MongoDB Operations

db.createcollection(“testcollection”);

insert(): Adds one or more documents to a collection.
Example: db.collection.insert({ name: "John Doe", age: 30 })

insertOne(): Inserts a single document into a collection.
Example: db.collection.insertOne({ name: "John Doe", age: 30 });

insertMany(): Inserts multiple documents into a collection in a single operation.
Example: db.collection.insertMany([{ name: "Alice", age: 25 }, { name: "Bob", age: 28 }]);

MongoDB Operations

find(): Retrieves documents from a collection based on specified query criteria.
Example: db.collection.find({ age: { $gt: 25 } })

findOne(): Retrieves a single document that matches specified criteria.
Example: db.collection.findOne({ name: "John Doe" })

replaceOne(): Replaces a single document that matches the specified filter with a new document.
Example: db.collection.replaceOne({ name: "John Doe" }, { name: "Jane Doe", age: 25 })

MongoDB Operations

update(): Modifies existing documents in a collection.
Example: db.collection.update({ name: "John Doe" }, { $set: { age: 31 } })

updateOne(): Updates a single document that matches the specified filter.
Example: db.collection.updateOne({ name: "John Doe" }, { $set: { age: 31 } })

updateMany(): Updates multiple documents that match the specified filter.
Example: db.collection.updateMany({ age: { $lt: 30 } }, { $set: { status: "young" } })

MongoDB Operations

delete(): Removes documents from a collection that match specified criteria.
Example: db.collection.delete({ name: "John Doe" })

deleteOne(): Deletes a single document that matches the specified filter.
Example: db.collection.deleteOne({ name: "John Doe" })

deleteMany(): Deletes multiple documents that match the specified filter.
Example: db.collection.deleteMany({ age: { $lt: 25 } })

Resources

Go over https://www.mongodb.com/docs/manual/crud/