Analytics career connect

MongoDB MATERIAL

Analytics career connect

TABLE OF CONTENTS:

1.Introduction to MongoDB

Definition:

MongoDB is a **NoSQL document-oriented database** that stores information as JSON-like documents. It is designed for scalability, flexibility, and high performance. Unlike SQL databases, it does not require a predefined schema.

Key Features:

- Schema-less (flexible structure).
- Stores data in BSON (Binary JSON).
- Provides replication and sharding for scalability.
- Supports aggregation and indexing.
- Widely used in big data, IoT, e-commerce, and analytics.

Examples:

1. Social media user profile storage.
2. IoT sensor data collection.
3. Real-time shopping cart management

2.MongoDB Basics

Core Components:

- **Database:** Container of collections.
- **Collection:** Group of related documents (like a SQL table).
- **Document:** JSON-like record.
- **Field:** Key-value pair inside a document.

Example Document:

```
{
  "_id": 1,
  "name": "Ganesh",
  "age": 22,
  "skills": ["Python", "SQL", "MongoDB"]
}
```

Examples:

1. A users collection with personal details.
2. An orders collection for e-commerce.
3. A products collection with stock and pricing.

3.Installation & Tools

Options to Install:

1. **MongoDB Community Edition** – Local installation.
2. **MongoDB Compass** – GUI for visualization.
3. **mongosh** – Command line shell.
4. **MongoDB Atlas** – Cloud-based service.

Steps:

- Install MongoDB binaries.
- Start mongod service.
- Use mongosh for queries.

Examples:

1. Install on Windows/Linux using MSI or tar files.
2. Use **MongoDB Compass** to connect and explore.
3. Create a free **MongoDB Atlas cluster** online.

4.MongoDB Shell Commands

Database Commands:

show dbs
use mydb
db.dropDatabase()

Collection Commands:

db.createCollection("users")
show collections
db.users.drop()

CRUD Operations:

- Insert: db.users.insertOne({name:"Ganesh", age:22})
- Read: db.users.find({age:{$gt:20}})
- Update: db.users.updateOne({name:"Ganesh"}, {$set:{age:23}})
- Delete: db.users.deleteOne({name:"Ganesh"})

Examples:

1. Add new users to users collection.
2. Find users aged above 20.
3. Update or remove specific records.

5.Operators in MongoDB

Comparison Operators: $eq, $ne, $gt, $lt, $in, $nin

Logical Operators: $and, $or, $not, $nor

Element Operators: $exists, $type

Update Operators: $set, $unset, $inc, $push, $pull, $rename

Examples:

1. Find users: { age: { $gt: 25 } }
2. Combine filters: { $or:[{city:"Hyd"},{age:22}] }
3. Update user: { $set:{city:"Hyderabad"} }

6.Indexing

Definition: Indexes improve query speed by reducing the number of documents MongoDB scans.

Types of Indexes:

- Single field index.
- Compound index.
- Text index.
- Geospatial index.

Examples:

1. db.users.createIndex({name:1})
2. db.users.createIndex({age:-1})
3. db.users.createIndex({city:1, age:1})

7.Aggregation Framework

Definition: Aggregation is used to transform and analyze data.

Key Stages: $match, $group, $project, $sort, $limit, $lookup

Examples:

1. Grouping Sales:

```
db.sales.aggregate([
 { $match:{status:"A"} },
 { $group:{_id:"$cust_id", total:{$sum:"$amount"}} }
])
```

2. Sorting: { $sort:{total:-1} }
3. Joining: $lookup to combine users and orders.

8.Relationships in MongoDB

Types:

- **Embedded Documents:** Store related data inside a document.
- **References:** Use ObjectId to link documents.

Examples:

1. Embedded:

```
{
 "name":"Ganesh",
 "address": { "city":"Hyd", "pin":500001 }
}
```

2. Reference:

```
{
 "name":"Ganesh",
 "address_id": ObjectId("654321...")
}
```

3. One-to-Many with arrays.

9.Advanced Topics

Replication:
Creates multiple copies of data across servers for fault tolerance.
Sharding:
Splits large datasets into smaller parts across multiple machines.
Transactions:
Support ACID properties for multiple-document updates.
Security:
Authentication, authorization, and role-based access.
Examples:

1. Replica set with 3 MongoDB nodes.
2. Sharding a collection by user_id.
3. Using transactions for banking transfers.

10.MongoDB vs SQL

Feature	**SQL (RDBMS)**	**MongoDB (NoSQL)**
Data Model	Tables & Rows	Collections & Documents
Schema	Fixed	Flexible
Query Language	SQL	MQL
Joins	Yes	Limited ($lookup)
Transactions	Strong	Supported (from 4.0)
Scaling	Vertical	Horizontal (Sharding)

Examples:

1. SQL: SELECT * FROM users WHERE age>20;
   MongoDB: db.users.find({age:{$gt:20}})
2. SQL: Data normalization with joins.
   MongoDB: Embedded documents.
3. SQL: Vertical scaling (more CPU/RAM).
   MongoDB: Horizontal scaling with shards.

Quick Recap:

MongoDB Fundamentals

- **MongoDB Definition:** A NoSQL, document-oriented database that stores data as JSON-like documents (BSON).
- **Key Features:**
  - Schema-less (flexible structure)
  - Scalable (replication & sharding)
  - High performance
  - Supports indexing & aggregation
- **Core Components:**
  - Database → container of collections
  - Collection → group of related documents (like SQL table)
  - Document → JSON-like record
  - Field → key-value pair
- **Installation & Tools:**
  - Local: MongoDB Community Edition
  - GUI: MongoDB Compass
  - CLI: mongosh
  - Cloud: MongoDB Atlas
- **Shell Commands:**
  - Database: show dbs, use mydb
  - Collection: db.createCollection("users")
  - CRUD: insertOne(), find(), updateOne(), deleteOne()

Working with Data

- **Operators:**
  - Comparison: $eq, $gt, $lt
  - Logical: $and, $or
  - Element: $exists, $type
  - Update: $set, $inc, $push
- **Indexing:**
  - Improves query speed
  - Types: Single field, compound, text, geospatial
  - Example: db.users.createIndex({name:1})

- **Aggregation Framework:**
    - Stages: $match, $group, $project, $sort, $limit, $lookup
    - Used for grouping, filtering, sorting, joining data
- **Relationships:**
    - Embedded documents → nested inside parent
    - References → linked with ObjectId
    - Supports one-to-one, one-to-many, many-to-many

Advanced Features & Comparison

- **Advanced Features:**
    - Replication → multiple copies for fault tolerance
    - Sharding → distribute data across servers for scale
    - Transactions → multi-document ACID compliance
    - Security → authentication & role-based access
- **MongoDB vs SQL:**
    - Data Model: Tables (SQL) vs Documents (MongoDB)
    - Schema: Fixed (SQL) vs Flexible (MongoDB)
    - Joins: Strong in SQL, limited in MongoDB ($lookup)
    - Scaling: Vertical (SQL) vs Horizontal (MongoDB)
    - Transactions: Supported in both (MongoDB from v4.0)

MongoDB Q&A

1. **What is MongoDB?**
   → A NoSQL document-oriented database that stores data as JSON-like documents (BSON) with a flexible schema.
2. **What is the difference between SQL and MongoDB?**
   → SQL uses tables, rows, and a fixed schema; MongoDB uses collections, documents, and a flexible schema.
3. **What is a Collection in MongoDB?**
   → A group of related documents (like a table in SQL).
4. **What is a Document?**
   → A JSON-like object containing key-value pairs. Example:
   { "name": "Ganesh", "age": 22 }
5. **What is BSON?**
   → Binary JSON – the format MongoDB uses internally to store documents.
6. **What are CRUD operations in MongoDB?**
   → Create, Read, Update, Delete. Example:
   - insertOne() → Create
   - find() → Read
   - updateOne() → Update
   - deleteOne() → Delete
7. **What is the difference between find() and findOne()?**
   → find() returns multiple documents (cursor). findOne() returns only the first matching document.
8. **What are indexes in MongoDB? Why are they important?**
   → Indexes improve query performance by avoiding full collection scans.
   Example:
   db.users.createIndex({name:1})
9. **What is the Aggregation Framework?**
   → A pipeline-based system for transforming/analyzing documents (like SQL GROUP BY).
   Example:
   db.sales.aggregate([{ $group: { _id: "$cust_id", total: { $sum: "$amount" } } }])

10. **How is a primary key represented in MongoDB?**
→ Every document has a unique _id field (ObjectId by default).
11. **What is Sharding in MongoDB?**
→ Distributing data across multiple servers for horizontal scaling.
12. **What is Replication in MongoDB?**
→ Creating multiple copies of data (Replica Set) for fault tolerance and high availability.
13. **What is the difference between Embedded and Referenced relationships?**
    - **Embedded:** Store related data inside the same document.
    - **Referenced:** Store ObjectId references to other collections.
14. **How are transactions handled in MongoDB?**
→ Since version 4.0, MongoDB supports multi-document ACID transactions.
15. **What is $lookup in MongoDB?**
→ An aggregation stage used to perform a join between collections.

ARISETTY GANESH

DATA ANALYTICS INTERN

ganesha.data.analytics@gmail.com