No-SQL MongoDB

1. Introduction to NoSQL

NoSQL databases (AKA "not only SQL") store data differently than relational tables. NoSQL databases come in a variety of types based on their data model. The main types are document , key-value, wide-column, and graph. They provide flexible schemas and scale easily with large amounts of big data and high user loads.

1. What is NoSQL?

- **NoSQL** stands for **“Not Only SQL”**
- A type of database that stores and retrieves data in formats other than the traditional **row-column** structure.
- Designed for **flexibility, scalability, and performance** for modern applications.

Key features:

- Schema-less (dynamic structure)
- Handles unstructured/semi-structured data
- Distributed & horizontally scalable
- Supports big data & real-time processing

Types of NoSQL Databases

1. **Document-based** (e.g., MongoDB)
2. **Key-Value stores** (e.g., Redis)
3. **Column-family stores** (e.g., Cassandra)

4. **Graph databases** (e.g., Neo4j)

Real-world Example:

- Instagram stores user profiles and posts as **JSON documents** for flexibility.
- Netflix uses NoSQL for fast video recommendations.

---

2. NoSQL vs RDBMS

Feature	NoSQL (MongoDB)	RDBMS (MySQL, Oracle)
Schema	Dynamic	Fixed (predefined)
Scalability	Horizontal	Vertical
Data Format	JSON/BSON	Tables (rows/columns)
Joins	Limited	Supported
Best Use Case	Big data, real-time apps	Complex transactions
Speed with large scale	High	May degrade

Unique Tip:

Think of RDBMS as a **structured warehouse** with fixed boxes,
while NoSQL is a **flexible storeroom** where you can store boxes, bags, or even loose items.

---

3. JSON Introduction

JSON = JavaScript Object Notation

- Lightweight data format
- Human-readable & machine-parseable

- Key-Value pairs, arrays, nested data

Example:

```
{
  "name": "Dani Krishnamurthi",
  "age": 25,
  "skills": ["Java", "Python", "MongoDB"],
  "address": {
    "city": "Bengauru",
    "country": "India"
  }
}
```

In MongoDB: JSON is stored as **BSON** (Binary JSON) for faster processing.

---

4. MongoDB Installation (macOS)

```
brew tap mongodb/brew
brew install mongodb-community
brew services start mongodb-community
mongosh  # open MongoDB shell
```

MongoDB Installation (Windows)

```
# 1. Download MongoDB Community Server
```

```
# Go to:
https://www.mongodb.com/try/download/community
# 2. Install MongoDB (choose "Complete" setup)
# During installation, check the option "Install
MongoDB as a Service"

# 3. Start MongoDB service
net start MongoDB

# 4. Open MongoDB Shell (mongosh)
mongosh
```

Check version:

```
mongod --version
```

---

5. CRUD Operations in Mongo Shell

Create

```
db.students.insertOne({ name: "Priya", age: 22,
course: "B.Tech" })

db.students.insertMany([
  { name: "Ravi", age: 24, course: "MCA" },
  { name: "Anjali", age: 23, course: "MBA" }
])
```

Read

```
db.students.find()              // all documents
db.students.find({ age: { $gt: 23 } }) // filter
```

Update

```
db.students.updateOne(
  { name: "Priya" },
  { $set: { course: "M.Tech" } }
)
```

Delete

```
db.students.deleteOne({ name: "Ravi" })
```

---

6. Query Operators

Operator	Description	Example
$gt	Greater than	{ age: { $gt: 25 } }
$lt	Less than	{ age: { $lt: 30 } }
$in	Matches any value in array	{ course: { $in: ["MBA","MCA"] } }
$and	Logical AND	{ $and: [{ age: { $gt: 20 } }, { course: "MCA" }] }

---

7. Data Modeling

Best Practices:

- Embed related data if always accessed together

- Reference data if reused across collections
- Avoid deep nesting (max 2–3 levels)
- Use meaningful field names

Example: Embedded:

```
{
  name: "Ravi",
  orders: [
    { product: "Laptop", price: 50000 },
    { product: "Mouse", price: 500 }
  ]
}
```

---

8. Storage Classes

MongoDB stores data in:

- **WiredTiger storage engine** (default)
- Supports **compression** for efficiency
- Journaling for crash recovery

---

9. Indexing & Performance

Create Index:

```
db.students.createIndex({ name: 1 })
```

Check Indexes:

```
db.students.getIndexes()
```

Unique Tip: Always index fields used in frequent queries; avoid indexing rarely used fields.

---

10. Aggregation Framework

Used for analytics and data transformation.

Example: Count students per course:

```
db.students.aggregate([
  { $group: { _id: "$course", total: { $sum: 1 }
} }
])
```

---

11. MongoDB Replication

1. What is Replication?

- Replication in MongoDB is the process of **synchronizing data across multiple servers**.

- It provides **high availability**, **fault tolerance**, and **data redundancy**.
- A group of MongoDB servers that maintain the same data set is called a **Replica Set**.

---

2. Why Replication is Important?

Protects against **hardware failures**

Ensures **high availability** of data

Enables **automatic failover**

Helps in **backup and disaster recovery**

Supports **scalability for read operations**

---

3. Replica Set Components

A replica set usually has **3 or more nodes**:

1. **Primary**
    - Receives **all write operations**.
    - Replicates changes to secondary nodes.
2. **Secondary**
    - Copies data from the primary's oplog (operation log).
    - Can be used for read operations (depending on read preferences).
3. **Arbiter** (optional)
    - Does **not store data**.
    - Only participates in elections to decide a new primary.
    - Useful when you want an odd number of voting members without extra storage cost.

Replication = multiple copies of data for **high availability**.

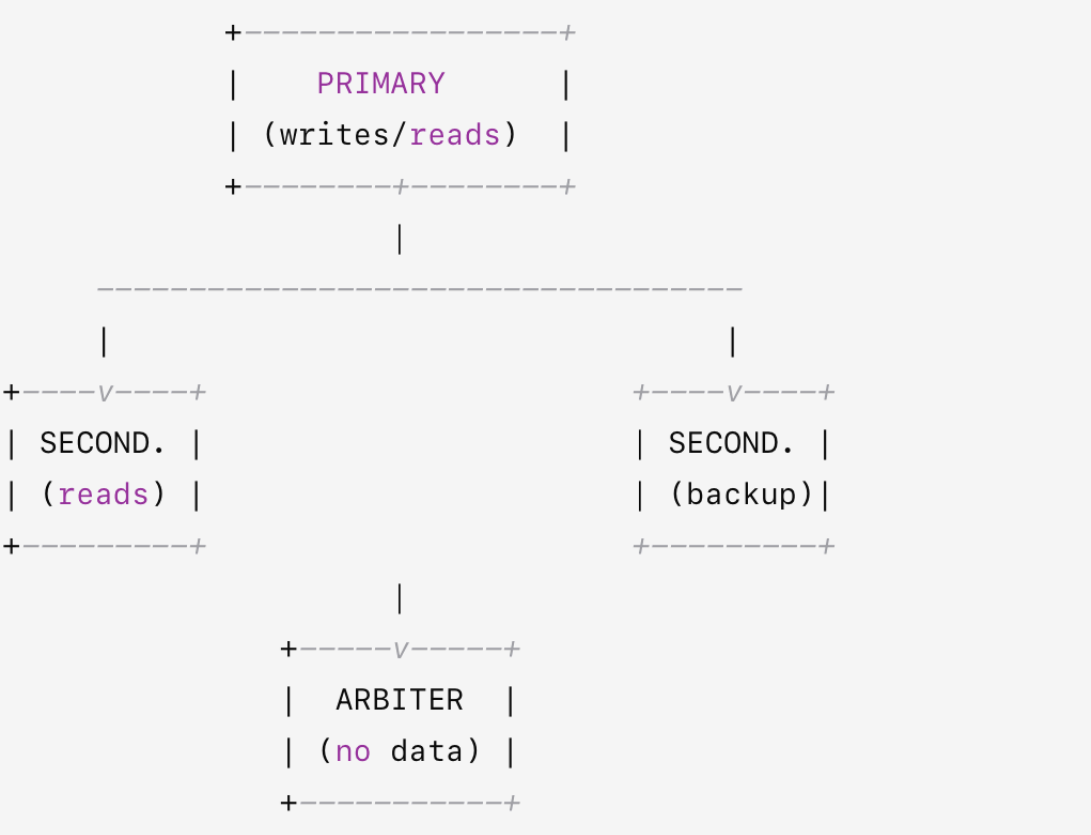
- **Primary**: Accepts writes
- **Secondary**: Read-only copies
- **Arbiter**: Helps in election

Start a replica set (development):

```
mongod --replSet "rs0"
```

In shell:

```
rs.initiate()
```



---

6. Basic Commands

Start Replica Set

mongod --replSet "rs0" --port 27017 --dbpath /data/db1 --bind_ip localhost

mongod --replSet "rs0" --port 27018 --dbpath /data/db2 --bind_ip localhost

mongod --replSet "rs0" --port 27019 --dbpath /data/db3 --bind_ip localhost

Connect to Mongo Shell

mongo --port 27017

Initiate Replica Set

```
rs.initiate({
  _id: "rs0",
  members: [
    { _id: 0, host: "localhost:27017" },
    { _id: 1, host: "localhost:27018" },
    { _id: 2, host: "localhost:27019" }
```

```
  ]
})
```

Check Replica Set Status

```
rs.status()
```

---

7. Reading from Replica Set

- By default → all reads & writes go to primary.
  If you want to allow reads from secondaries:
- db.getMongo().setReadPref("secondary")

---

8. Advantages

- Automatic failover
- Redundancy (multiple copies of data)
- Horizontal read scaling (read from secondaries)

9. Limitations

- Writes can only go to primary.
- Replication lag may cause secondaries to be slightly behind.
- More servers = more hardware cost.

In summary:
MongoDB Replication = Primary + Secondaries + Arbiter, ensuring high availability and automatic failover with minimal downtime.

Unique & Informative Add-ons

- **Tip for Non-IT learners:** Compare MongoDB queries to Google search:
  - `db.students.find({ name: "Priya" })` = Searching for exact match in a document.
- **Real-life analogy for Replication:** Like Google Docs auto-save across devices — one change syncs everywhere.
- **For IT learners:** Explain BSON advantages (speed, type support like Date, Decimal128).

Quiz

1. NoSQL databases are always schema-less. (T/F)
2. Which MongoDB command inserts multiple documents?
3. What's the difference between `$gt` and `$gte`?
4. Give an example of embedding data vs referencing in MongoDB.
5. What is the role of an Arbiter in replication?