

1. Introduction to MongoDB

◆ What is MongoDB?

- MongoDB is an **open-source, NoSQL, document-oriented** database.
 - It stores data in the form of **documents** (similar to JSON).
 - Each document is a **collection of key-value pairs**.
 - MongoDB is designed for **scalability, flexibility, and high performance**.
-

◆ Key Features:

1. Document-oriented storage

- Stores data as documents instead of tables and rows.
- Example document:

```
{  
  "_id": 1,  
  "name": "Rahul",  
  "course": "Java",  
  "score": 95  
}
```

2. Schema-less

- Documents in the same collection can have different fields.
- Example:

```
{ "name": "Rahul", "age": 22 }  
{ "name": "Meera", "city": "Bangalore", "course": "Python" }
```

3. Collection-based

- Documents are grouped into **collections** (similar to tables in SQL).

4. Horizontal scalability

- Data can be distributed across multiple servers (**Sharding**).

5. High performance

- Uses **indexes** for faster query performance.

6. Powerful query language

- Queries use a JSON-like syntax.

7. Integration with multiple languages

- MongoDB supports Java, Python, Node.js, etc.

MongoDB Architecture

Think of MongoDB as a **smart filing system** that stores and manages your data.



How MongoDB Works

When you use MongoDB:

- 👉 You (the **client**)
 - send requests to the **MongoDB server (mongod)**
 - the server stores or retrieves your data from **databases**.
-

◆ 1. Client

- The user or application (like Java, Python, Compass, or Mongo Shell).
 - It sends commands such as:
 - db.students.find()
 - The client just asks questions; the server does the work.
-

◆ 2. MongoDB Server (mongod)

- The main brain  of MongoDB.
 - It handles all:
 - Read & write requests
 - Data storage
 - Indexing
 - Replication
 - Security
-

◆ 3. Database

- A container where your data lives.
- You can create many databases in one MongoDB server.
- Example:
 - studentDB
 - hotelDB
 - projectDB

◆ 4. Collection

- Inside a database, data is grouped into **collections**.
(Think of them like folders inside your database.)
 - Example:
 - students, teachers, courses
-

◆ 5. Document

- Each **record** inside a collection.
- It's a JSON-like structure called **BSON**.

Example:

```
{  
  "_id": 1,  
  "name": "Rahul",  
  "age": 22,  
  "course": "Java"  
}
```

◆ 6. _id Field

- Each document gets a unique ID (like a roll number).
 - If you don't give one, MongoDB creates an **ObjectId** automatically.
-

◆ 7. Storage Engine

- The internal system that actually saves your data to disk.
 - Default: **WiredTiger** → fast and supports compression.
-

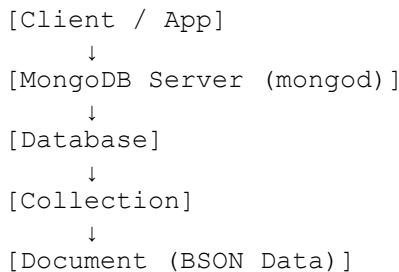
◆ 8. Replication

- Keeps **multiple copies** of your data on different servers.
 - If one server fails, another automatically takes over.
🌀 (This setup is called a **Replica Set**)
-

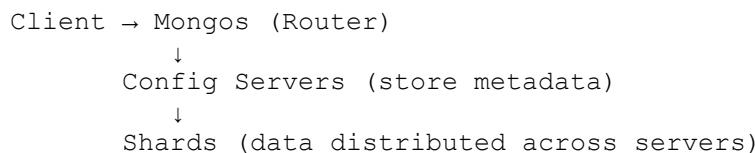
◆ 9. Sharding

- When data becomes too large, MongoDB splits it into parts (called **shards**) and stores them on multiple machines — making it faster and scalable.
-

In One Simple Diagram



Advanced Setup (When Scaled)



CRUD Operations in MongoDB

CRUD = **Create, Read, Update, Delete**

These are the four basic operations you perform on data in any database.

In MongoDB, all these operations are done on **documents inside collections**.

MongoDB CRUD Operations

Operation	Meaning	Command
C	Create (Insert)	insertOne(), insertMany()
R	Read (Find)	find(), findOne()
U	Update	updateOne(), updateMany(), replaceOne()
D	Delete	deleteOne(), deleteMany()

Example Setup

Let's say we are working with a **students** collection:

```
use studentDB
```

1. CREATE — Insert Documents

► Insert One Document

```
db.students.insertOne({  
  name: "Rahul",  
  age: 22,  
  course: "Java"  
})
```

Output:

```
{  
  "acknowledged": true,  
  "insertedId": ObjectId("67226eabcde123...")  
}
```

► Insert Multiple Documents

```
db.students.insertMany([  
  { name: "Meera", age: 21, course: "Python" },  
  { name: "Amit", age: 23, course: "Spring Boot" }  
])
```

Output:

```
{  
  "acknowledged": true,  
  "insertedIds": {  
    "0": ObjectId("67226eabcde124..."),  
    "1": ObjectId("67226eabcde125...")  
  }  
}
```

2. READ — Retrieve Documents

► Find All Documents

```
db.students.find()
```

- Shows all records in the collection.
-

► Find One Document

```
db.students.findOne({ name: "Rahul" })
```

- Returns the first matching document.
-

► Apply Filter (like WHERE in SQL)

```
db.students.find({ age: { $gt: 21 } })
```

- Returns all students with age > 21.
-

► Project Specific Fields

```
db.students.find({}, { name: 1, course: 1, _id: 0 })
```

- Shows only name and course fields (hides _id).
-

► Sort Documents

```
db.students.find().sort({ age: 1 }) // 1 = ascending, -1 = descending
```



3. UPDATE — Modify Documents

► Update One Document

```
db.students.updateOne(  
  { name: "Rahul" },  
  { $set: { course: "Advanced Java" } }  
)
```

- Updates only Rahul's course.
-

► Update Multiple Documents

```
db.students.updateMany(  
  { age: { $lt: 23 } },  
  { $set: { active: true } }  
)
```

- Adds "active": true for all students younger than 23.
-

► Replace Entire Document

```
db.students.replaceOne(  
  { name: "Meera" },  
  { name: "Meera", course: "Data Science", age: 22 }  
)
```

- Replaces the entire document (not just specific fields).
-

● 4. DELETE — Remove Documents

► Delete One Document

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

- Deletes the first matching document.
-

► Delete Multiple Documents

```
db.students.deleteMany({ age: { $lt: 21 } })
```

- Deletes all students with age less than 21.
-

► Delete All Documents (Clear Collection)

```
db.students.deleteMany({})
```

📊 Comparison with SQL

Operation SQL Command MongoDB Equivalent

Insert	INSERT INTO	insertOne() / insertMany()
Read	SELECT * FROM	find() / findOne()
Update	UPDATE ... SET	updateOne() / updateMany()
Delete	DELETE FROM	deleteOne() / deleteMany()

✓ Summary Notes

- insertOne() → Insert one document

- `insertMany()` → Insert multiple
 - `find()` → Fetch multiple documents
 - `findOne()` → Fetch one document
 - `updateOne()`, `updateMany()` → Modify existing documents
 - `replaceOne()` → Replace entire document
 - `deleteOne()`, `deleteMany()` → Remove documents
-

 **Tip:**

- MongoDB queries use **JSON-like syntax**.
 - All filters use **operators** like `$gt`, `$lt`, `$in`, `$and`, `$or`, etc.
-

Would you like me to continue your notes next with **MongoDB Query Operators and Filters** (like `$gt`, `$in`, `$and`, `$regex` etc.)?

You said:

Now On Operators

ChatGPT said:

Perfect  — now let's move to the **MongoDB Operators** section in your notes.

Operators make MongoDB queries powerful — they help filter, compare, and modify data.

 **5. MongoDB Operators**

Operators are **special keywords (starting with \$)** used to:

- Filter data while reading (`find()`)
 - Modify data while updating (`updateOne()`)
 - Perform calculations during aggregation
-

 **Types of MongoDB Operators**

Type	Description
1. Comparison Operators	Compare field values (<code>></code> , <code><</code> , <code>=</code> , etc.)
2. Logical Operators	Combine multiple conditions
3. Element Operators	Check if a field exists or its type

Type	Description
4. Evaluation Operators	Match values using regex or expressions
5. Array Operators	Work with arrays inside documents
6. Update Operators	Used in update queries to modify data

1. Comparison Operators

Operator Meaning	Example
\$eq Equal to	{ age: { \$eq: 22 } } → age = 22
\$ne Not equal to	{ age: { \$ne: 22 } }
\$gt Greater than	{ age: { \$gt: 20 } }
\$gte Greater than or equal	{ age: { \$gte: 21 } }
\$lt Less than	{ age: { \$lt: 25 } }
\$lte Less than or equal	{ age: { \$lte: 22 } }
\$in Matches any value in array { course: { \$in: ["Java", "Python"] } }	{ course: { \$in: ["Java", "Python"] } }
\$nin Not in	{ course: { \$nin: ["C", "C++"] } }

Example:

```
db.students.find({ age: { $gt: 21, $lt: 25 } })
```

→ Fetches students whose age is between 21 and 25.

2. Logical Operators

Operator Meaning	Example
\$and Both conditions true	{ \$and: [{ age: { \$gt: 21 } }, { course: "Java" }] }
\$or Either condition true	{ \$or: [{ course: "Java" }, { course: "Python" }] }
\$not Negates a condition	{ age: { \$not: { \$gt: 25 } } }
\$nor None of the conditions true { \$nor: [{ course: "Java" }, { course: "Python" }] }	{ \$nor: [{ course: "Java" }, { course: "Python" }] }

Example:

```
db.students.find({
```

```
  $or: [
```

```

    { course: "Java" },
    { course: "Python" }

]
})

```

3. Element Operators

Operator	Meaning	Example
----------	---------	---------

\$exists Checks if a field exists { phone: { \$exists: true } }

\$type Matches field data type { age: { \$type: "int" } }

 **Example:**

db.students.find({ city: { \$exists: false } })

→ Finds documents where city field is missing.

4. Evaluation Operators

Operator	Meaning	Example
\$regex	Matches a pattern using regular expressions	{ name: { \$regex: "^R" } } → names starting with R
\$expr	Use expressions inside queries	{ \$expr: { \$gt: ["\$marks", "\$attendance"] } }

 **Example:**

db.students.find({ name: { \$regex: "al\$", \$options: "i" } })

→ Finds names ending with “al” (case-insensitive).

5. Array Operators

Operator	Meaning	Example
\$all	Match all elements in array	{ skills: { \$all: ["Java", "Spring Boot"] } }
\$size	Match array size	{ skills: { \$size: 3 } }
\$elemMatch	Match specific element condition	{ scores: { \$elemMatch: { math: { \$gt: 80 }, science: { \$gt: 70 } } } }

 **Example:**

db.students.find({ skills: { \$all: ["Java", "Python"] } })

6. Update Operators

Used with updateOne() / updateMany() to modify documents.

Operator	Meaning	Example
\$set	Set a field value	{ \$set: { course: "Advanced Java" } }
\$unset	Remove a field	{ \$unset: { city: "" } }
\$inc	Increment numeric field	{ \$inc: { age: 1 } }
\$rename	Rename a field	{ \$rename: { course: "subject" } }
\$push	Add element to array	{ \$push: { skills: "MongoDB" } }
\$addToSet	Add only if not already present	{ \$addToSet: { skills: "Java" } }
\$pull	Remove element from array	{ \$pull: { skills: "C++" } }

Example:

```
db.students.updateOne(  
  { name: "Rahul" },  
  { $inc: { age: 1 }, $push: { skills: "MongoDB" } }  
)
```

→ Increases Rahul's age by 1 and adds "MongoDB" to his skills array.

Summary Table

Category	Examples
Comparison	\$eq, \$ne, \$gt, \$gte, \$lt, \$lte, \$in, \$nin
Logical	\$and, \$or, \$not, \$nor
Element	\$exists, \$type
Evaluation	\$regex, \$expr
Array	\$all, \$size, \$elemMatch
Update	\$set, \$unset, \$inc, \$push, \$pull, \$addToSet, \$rename