

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### Module 3

#### 3.1 Introduction to NoSQL

##### Introduction

Traditional Relational Database Management Systems (RDBMS) are effective for structured data but face challenges when dealing with the huge volume, velocity, and variety of Big Data. **NoSQL (Not Only SQL)** databases were introduced to address these limitations by providing flexible schemas, horizontal scalability, and high performance.

---

##### Definition

**NoSQL** is a class of non-relational databases designed to store, manage, and retrieve structured, semi-structured, and unstructured data efficiently.

---

##### Characteristics of NoSQL Databases

- Schema-less data model
- Horizontal scalability
- Distributed architecture
- High availability
- Fault tolerance
- Supports Big Data
- Handles structured and unstructured data
- Fast read/write operations

---

##### Need for NoSQL

Traditional databases have limitations such as:

- Fixed schema
- Vertical scaling only
- Difficulty handling massive datasets
- Limited support for semi-structured and unstructured data

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

NoSQL databases overcome these issues by using flexible document, key-value, column-family, and graph data models.

### Features of NoSQL

- Flexible schema
- Distributed storage
- Automatic replication
- High scalability
- High availability
- Fast query execution
- Cloud-friendly architecture

### Advantages of NoSQL

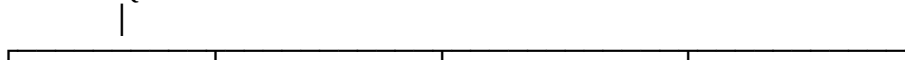
- Easy scalability
- High performance
- Supports Big Data
- Cost-effective
- Handles heterogeneous data
- Easy replication

### Limitations of NoSQL

- Limited support for complex joins
- Less standardized than SQL
- ACID compliance may be limited (depends on the database)
- Different query languages across products

### Types of NoSQL Databases

NoSQL Databases



## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

|   |   |   |  
Key-Value Database   Document Database   Column Family Database   Graph Database

### 1. Key-Value Database

Stores information as key-value pairs.

Examples:

- Redis
- Riak
- Amazon DynamoDB

Applications:

- Session management
- Shopping carts
- Caching

### 2. Document Database

Stores information as documents using formats such as **JSON** and **BSON**.

Examples:

- MongoDB
- CouchDB

Applications:

- E-commerce
- Content Management Systems
- Web applications

### 3. Column-Family Database

Stores data column-wise instead of row-wise.

Examples:

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Apache Cassandra
- Apache HBase

Applications:

- Data warehousing
- IoT
- Time-series data

---

### 4. Graph Database

Represents data as nodes and relationships.

Examples:

- Neo4j
- Amazon Neptune

Applications:

- Social networks
- Fraud detection
- Recommendation systems

---

## Comparison: SQL vs NoSQL

| SQL                      | NoSQL                                                       |
|--------------------------|-------------------------------------------------------------|
| Relational               | Non-relational                                              |
| Fixed schema             | Flexible schema                                             |
| Vertical scaling         | Horizontal scaling                                          |
| SQL language             | Product-specific query languages                            |
| Best for structured data | Supports structured, semi-structured, and unstructured data |
| Complex joins supported  | Limited join support                                        |

---

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

# MongoDB

## Introduction

**MongoDB** is an open-source **document-oriented NoSQL database** that stores data as **BSON (Binary JSON)** documents instead of rows and columns.

It is designed for applications requiring scalability, flexibility, and high performance.

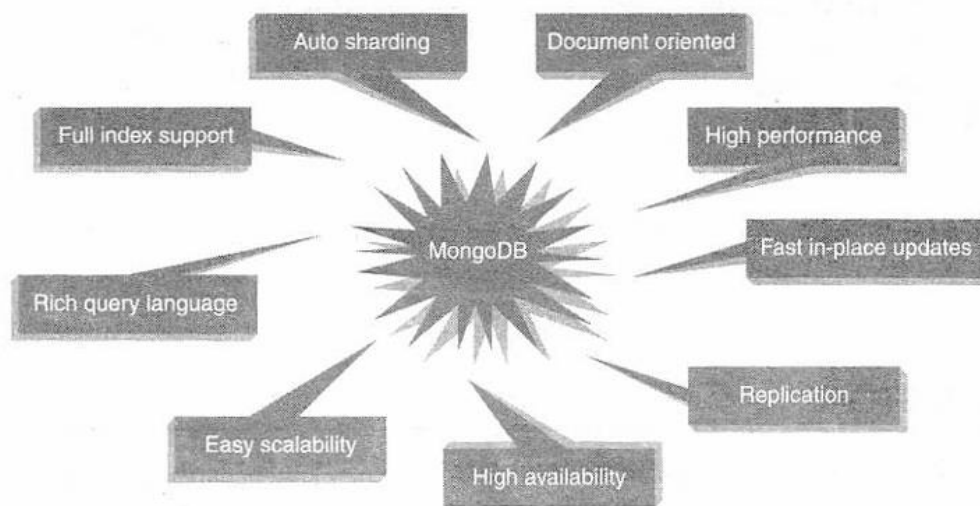


Figure 6.1 Why MongoDB?

## Definition

MongoDB is a document-oriented NoSQL database that stores data in BSON documents with dynamic schemas.

## Features of MongoDB

- Document-oriented storage
- Dynamic schema
- High availability
- Automatic replication
- Sharding support
- High performance

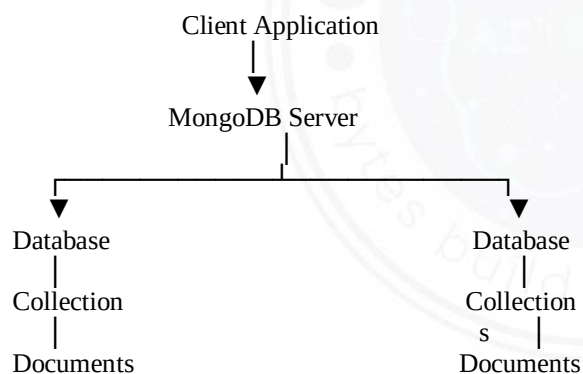
## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Indexing support
- Aggregation framework

## Advantages of MongoDB

- Flexible schema
- Easy scalability
- Fast data retrieval
- Cloud compatible
- High availability through replica sets
- Efficient handling of JSON-like documents

## MongoDB Architecture



## MongoDB Data Hierarchy

### Relational Database MongoDB

|          |            |
|----------|------------|
| Database | Database   |
| Table    | Collection |
| Row      | Document   |
| Column   | Field      |

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### BSON (Binary JSON)

MongoDB stores documents in **BSON**, a binary representation of JSON.

Advantages:

- Faster storage
- Efficient retrieval
- Supports additional data types
- Better indexing

### JSON Example

```
{
  "Name": "Rahul",
  "USN": "1RV21CS001",
  "Department": "CSE",
  "Marks": 88
}
```

### Applications of MongoDB

- E-Commerce
- Banking
- Healthcare
- Social Media
- Mobile Applications
- Content Management Systems
- IoT Applications
- Real-Time Analytics

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

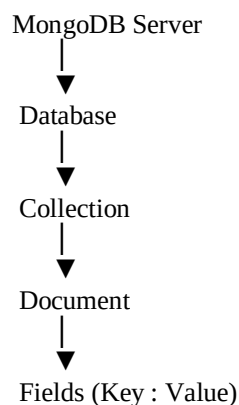
### 3.2 MongoDB Data Model

#### Introduction

MongoDB follows a **document-oriented data model** where information is stored as **documents** instead of rows and columns. Documents are grouped into **collections**, and collections are stored inside **databases**.

Unlike relational databases, MongoDB allows documents in the same collection to have different structures (dynamic schema).

#### MongoDB Data Hierarchy



#### Components

##### 1. Database

A database is a collection of related collections.

Example:

CollegeDB

##### 2. Collection

A collection stores related documents.

Equivalent to a **table** in an RDBMS.

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Example:

Student  
Employee  
Faculty

---

### 3. Document

A document stores data in **JSON/BSON** format.

Equivalent to a row in an RDBMS.

Example

```
{  
  "USN": "1RV21CS001",  
  "Name": "Rahul",  
  "Department": "CSE",  
  "Marks": 90  
}
```

---

### 4. Field

A field is a key-value pair inside a document.

Example

Name : Rahul  
Marks : 90

---

## MongoDB Document Structure

Student Collection

```
|  
|  
|— Document 1  
|   Name : Rahul  
|   Marks : 90  
|  
|  
|— Document 2
```

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Name : Anita

Marks : 95

Document 3

Name : Ravi

Marks : 80

### Advantages of Document Model

- Flexible schema
- Easy storage of nested objects
- Fast retrieval
- Supports hierarchical data
- No complex joins
- Better scalability

### JSON

#### Definition

**JSON (JavaScript Object Notation)** is a lightweight data interchange format used to store and exchange data.

#### Characteristics

- Human readable
- Text format
- Key-value pairs
- Language independent
- Lightweight

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### JSON Example

```
{  
  "Name": "Anita",  
  "Age": 20,  
  "Department": "ISE"  
}
```

### BSON

#### Definition

**BSON (Binary JSON)** is the binary representation of JSON used internally by MongoDB.

#### Advantages

- Faster storage
- Better indexing
- Supports additional data types
- Faster retrieval

### JSON vs BSON

| JSON                   | BSON                        |
|------------------------|-----------------------------|
| Text format            | Binary format               |
| Human readable         | Machine optimized           |
| Smaller feature set    | Supports more data types    |
| Slower parsing         | Faster parsing              |
| Used for data exchange | Used for storage in MongoDB |

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### CRUD Operations

CRUD stands for:

- **C** – Create
- **R** – Read
- **U** – Update
- **D** – Delete

These are the fundamental operations performed on MongoDB collections.

---

### Create Database

use CollegeDB

Creates (or switches to) the database.

---

### Create Collection

db.createCollection("Student")

---

### Insert One Document

db.Student.insertOne(

{

USN:"1RV21CS001",

Name:"Rahul",

Marks:90

}

)

---

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### Insert Multiple Documents

```
db.Student.insertMany(  
[  
{  
Name:"Anita",  
Marks:95  
},  
{  
Name:"Ravi",  
Marks:80  
}  
]  
)
```

### Read Operations

#### Display All Documents

```
db.Student.find()
```

#### Pretty Format

```
db.Student.find().pretty()
```

#### Display Selected Record

```
db.Student.find(  
{  
Name:"Rahul"  
}  
)
```

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### Query Operators

#### Greater Than

```
db.Student.find(  
  
  {  
  
    Marks:  
  
    {  
  
      $gt:80  
  
    }  
  
  }  
)
```

#### Less Than

```
db.Student.find(  
  
  {  
  
    Marks:  
  
    {  
  
      $lt:60  
  
    }  
  
  }  
)
```

#### Equal

```
db.Student.find(  
  
  {  
  
    Department:"CSE"  
  
  }  
)
```

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### Update Operations

#### Update One Record

```
db.Student.updateOne(
```

```
{
```

```
  Name:"Rahul"
```

```
},
```

```
{
```

```
  $set:
```

```
{
```

```
    Marks:95
```

```
  }
```

```
}
```

```
)
```

---

#### Update Multiple Records

```
db.Student.updateMany(
```

```
{
```

```
  Department:"CSE"
```

```
},
```

```
{
```

```
  $set:
```

```
{
```

```
    Semester:7
```

```
  }
```

```
}
```

```
)
```

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### Delete Operations

#### Delete One Record

```
db.Student.deleteOne(
```

```
{
```

```
  Name:"Rahul"
```

```
}
```

```
)
```

#### Delete Multiple Records

```
db.Student.deleteMany(
```

```
{
```

```
  Department:"ISE"
```

```
}
```

```
)
```

### Drop Collection

```
db.Student.drop()
```

Deletes the entire collection.

### Drop Database

```
db.dropDatabase()
```

Deletes the complete database.

### MongoDB Query Flow

Application

↓

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

MongoDB Driver



MongoDB Server



Database



Collection



Documents



Result

## MongoDB Data Types

MongoDB supports:

- String
- Integer
- Boolean
- Double
- Date
- Timestamp
- Object
- Array
- Null
- ObjectId

## ObjectId

Every MongoDB document contains a unique identifier called **ObjectId**.

Example

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

\_id :

64df23ab12998eac45f23cd8

## MongoDB Operators

### Operator Meaning

|       |                       |
|-------|-----------------------|
| \$gt  | Greater than          |
| \$lt  | Less than             |
| \$gte | Greater than or equal |
| \$lte | Less than or equal    |
| \$eq  | Equal                 |
| \$ne  | Not equal             |
| \$and | Logical AND           |
| \$or  | Logical OR            |
| \$in  | Value exists in list  |

## Applications

- Banking
- Healthcare
- Retail
- E-Commerce
- Social Networks
- IoT
- Mobile Apps
- Business Intelligence

## Advantages of MongoDB

- Dynamic schema

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- High performance
- Easy replication
- Horizontal scalability
- Distributed storage
- Fast indexing

### Limitations

- Limited joins
- Larger storage than normalized SQL databases
- Transactions may be less suitable for some workloads compared to traditional RDBMS.

## 3.3 MongoDB Indexing

### Introduction

When a MongoDB collection contains millions of documents, searching every document becomes slow. **Indexes** improve query performance by allowing MongoDB to locate documents quickly without scanning the entire collection.

### Definition

**Indexing** is the process of creating a special data structure that enables MongoDB to retrieve documents efficiently.

### Need for Indexing

Without Index

Search Query

↓

Entire Collection Scan

↓

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Required Document

With Index

Search Query

↓

Index

↓

Required Document

### Advantages

- Faster searching
- Improved query performance
- Reduced execution time
- Efficient sorting
- Better filtering

### Create Index

```
db.Student.createIndex(  
  { Name:1 }  
)
```

1 → Ascending order

### Descending Index

```
db.Student.createIndex(  
  { Marks:-1 }  
)
```

-1 → Descending order

### Display Indexes

```
db.Student.getIndexes()
```

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### Drop Index

```
db.Student.dropIndex(  
  {Name:1}  
)
```

---

### Types of Indexes

#### Single Field Index

Created on one field.

Example

```
{Name:1}
```

---

#### Compound Index

Created on multiple fields.

```
{Name:1,Marks:-1}
```

---

#### Text Index

Used for text searching.

```
db.Student.createIndex(  
  {Name:"text"}  
)
```

---

#### Unique Index

Prevents duplicate values.

```
db.Student.createIndex(  
  {USN:1},  
  {unique:true}  
)
```

---

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

# Replication

## Introduction

Replication improves data availability by maintaining multiple copies of the same database.

If one server fails, another server automatically takes over.

## Definition

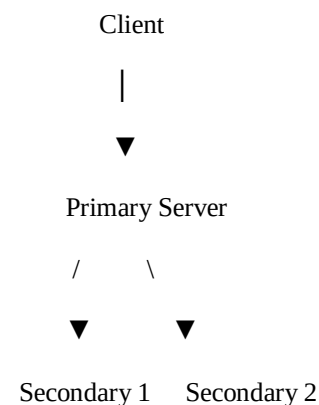
Replication is the process of maintaining identical copies of data on multiple servers.

## Replica Set

A Replica Set consists of:

- Primary Server
- Secondary Server(s)

## Replica Set Architecture



## Working

1. Client writes data to Primary.
2. Primary updates the database.

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

3. Secondary servers automatically copy the changes.
4. If Primary fails, one Secondary becomes the new Primary.

---

### Advantages

- High availability
- Fault tolerance
- Automatic backup
- Disaster recovery
- Continuous database service

---

### Limitations

- Additional storage required
- Increased network traffic
- Replica synchronization overhead

---

## Sharding

### Introduction

As data grows into terabytes or petabytes, a single server may not be sufficient.

MongoDB uses **Sharding** to distribute data across multiple servers.

---

### Definition

Sharding is the process of dividing a large database into smaller pieces called **shards**.

---

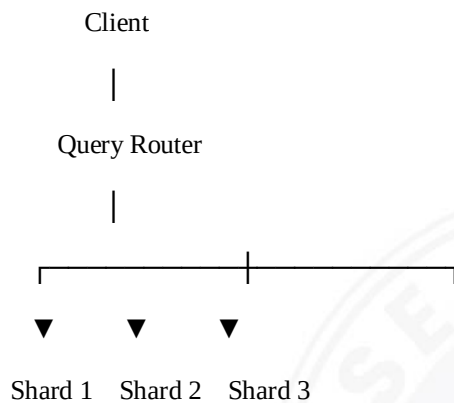
### Need for Sharding

- Massive datasets
- High traffic
- Faster processing

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Better scalability

### Sharding Architecture



### Components

#### Config Server

Stores metadata.

#### Query Router (mongos)

Receives client requests.

Routes queries to the correct shard.

#### Shards

Store actual data.

### Advantages

- Horizontal scalability
- Faster query processing

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Load balancing
- Better storage utilization

---

### Limitations

- Complex configuration
- Difficult administration
- Higher maintenance

---

## Replication vs Sharding

### Replication

Copies data

High availability

Improves reliability

Multiple copies

Backup solution

### Sharding

Splits data

Horizontal scaling

Improves performance

Different partitions

Storage solution

---

## Aggregation Framework

### Introduction

Aggregation processes multiple documents and returns summarized results.

It performs operations similar to SQL:

- GROUP BY
- COUNT
- SUM
- AVG

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### Definition

Aggregation is the process of transforming and summarizing MongoDB documents to produce analytical results.

### Aggregation Pipeline

MongoDB processes data through several stages.

Collection

↓

Match

↓

Group

↓

Sort

↓

Project

↓

Output

### Common Pipeline Stages

#### \$match

Filters documents.

#### Example

```
db.Student.aggregate(
```

```
[
```

```
{
```

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

\$match:

```
{  
  Department:"CSE"  
}  
}  
]  
)
```

\$group

Groups documents.

db.Student.aggregate(  
[  
{  
 \$group:  
{  
 \_id:"\$Department",  
 Average:  
{  
 \$avg:"\$Marks"  
 }  
 }  
}]  
)

\$sort

Sorts records.

```
{  
  $sort:
```

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

```
{
```

Marks:-1

```
}
```

```
}
```

### \$project

Displays selected fields.

```
{
```

\$project:

```
{
```

Name:1,

Marks:1

```
}
```

```
}
```

### \$limit

Limits output.

```
{
```

\$limit:5

```
}
```

## Aggregation Pipeline Example

Student Collection

↓

Filter CSE Students

↓

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Group by Department



Average Marks



Display Result

---

### Advantages of Aggregation

- High performance
  - Data summarization
  - Report generation
  - Statistical analysis
  - Business Intelligence
- 

### MongoDB Performance Optimization

Methods include:

- Create indexes
  - Use proper shard keys
  - Optimize queries
  - Avoid unnecessary fields
  - Use aggregation efficiently
  - Minimize document size
- 

### MongoDB Security Features

MongoDB provides:

- Authentication
- Authorization
- Role-based access control
- Data encryption
- SSL/TLS support

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Auditing

---

### Applications of MongoDB

- Banking
- Healthcare
- IoT
- Retail
- Social Media
- Mobile Apps
- Big Data Analytics
- Business Intelligence
- E-Commerce
- Government Applications

---

### Advantages of MongoDB

- Flexible schema
- High performance
- Horizontal scalability
- Easy replication
- Sharding support
- Fault tolerance

---

### Limitations

- Limited joins
- Large document size affects performance
- Complex sharding configuration
- Higher memory usage

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### 3.8 MongoDB Best Practices

#### Introduction

To improve the performance, scalability, and reliability of MongoDB applications, developers should follow recommended design and implementation practices.

#### Best Practices

##### 1. Create Appropriate Indexes

- Improve search performance
- Reduce query execution time
- Avoid unnecessary collection scans

##### 2. Design Small Documents

Large documents require more memory and increase query execution time.

Recommended:

- Store only necessary fields
- Avoid unnecessary nesting

##### 3. Use Embedding Carefully

Embed related data when:

- Relationship is one-to-one
- Relationship is one-to-few

#### Example

Student

↓

Address

↓

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

Phone Number

---

### 4. Use Referencing

Use references when:

- Relationship is one-to-many
- Documents become very large

Example

Student

↓

Department ID

↓

Department Collection

---

### 5. Optimize Queries

- Retrieve only required fields
  - Use indexes
  - Avoid unnecessary sorting
  - Minimize document scanning
- 

## MongoDB Transactions

### Definition

A **transaction** is a sequence of database operations executed as a single logical unit.

MongoDB supports **multi-document ACID transactions** in replica sets and sharded clusters.

---

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### ACID Properties

#### Atomicity

Either all operations execute or none execute.

---

#### Consistency

Database remains in a valid state.

---

#### Isolation

Transactions do not interfere with each other.

---

#### Durability

Committed data is permanently stored.

---

### MongoDB Backup and Restore

#### Backup

Creates a copy of database files for recovery.

Tool:

mongodump

---

#### Restore

Restores data from backup.

Tool:

mongorestore

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

### Backup Process

Database



mongodump



Backup File



mongorestore



Recovered Database

### MongoDB Shell Commands (Quick Reference)

#### Database Commands

Create/Switch Database

use CollegeDB

Display Databases

show dbs

Delete Database

db.dropDatabase()

#### Collection Commands

Create Collection

db.createCollection("Student")

Display Collections

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

show collections

Drop Collection

db.Student.drop()

---

### CRUD Commands

Insert

db.Student.insertOne({...})

Read

db.Student.find()

Update

db.Student.updateOne(...)

Delete

db.Student.deleteOne(...)

---

### Index Commands

Create Index

```
db.Student.createIndex(  
  {Name:1}  
)
```

Display Index

db.Student.getIndexes()

---

## MongoDB Command Summary

| Command  | Purpose         |
|----------|-----------------|
| use      | Select database |
| show dbs | List databases  |

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

| Command          | Purpose                   |
|------------------|---------------------------|
| show collections | List collections          |
| insertOne()      | Insert one document       |
| insertMany()     | Insert multiple documents |
| find()           | Retrieve documents        |
| updateOne()      | Update one document       |
| updateMany()     | Update multiple documents |
| deleteOne()      | Delete one document       |
| deleteMany()     | Delete multiple documents |
| createIndex()    | Create index              |
| aggregate()      | Perform aggregation       |

## SQL vs MongoDB

| SQL          | MongoDB                 |
|--------------|-------------------------|
| Database     | Database                |
| Table        | Collection              |
| Row          | Document                |
| Column       | Field                   |
| Primary Key  | ObjectId                |
| JOIN         | Embedding / Referencing |
| Fixed Schema | Dynamic Schema          |

## MongoDB Advantages (Revision)

- Flexible schema

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Easy scalability
- High availability
- Automatic replication
- Sharding support
- Fast indexing
- Distributed architecture
- High performance
- JSON/BSON document model

### MongoDB Limitations

- Limited join operations
- Increased storage for embedded documents
- Complex sharding administration
- Memory intensive for large working sets

### Complete MongoDB Workflow

Client Application



MongoDB Driver



MongoDB Server



Database



Collection



Document



CRUD Operations

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

↓

Indexing

↓

Aggregation

↓

Result

---

## Complete Module 3 Mind Map

MODULE 3

↓

NoSQL Database

↓

Types of NoSQL

↓

MongoDB

↓

Database

↓

Collection

↓

Document

↓

CRUD Operations

↓

JSON / BSON

↓

ObjectId

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML



Indexing



Replication



Sharding



Aggregation



Transactions



Backup & Restore



Business Intelligence Applications

---

## Applications of MongoDB

- Banking Systems
- Healthcare Analytics
- Retail Management
- Social Media
- E-Commerce
- IoT Applications
- Mobile Applications
- Content Management Systems
- Business Intelligence
- Big Data Analytics

---

## Advantages in Business Intelligence

MongoDB is widely used in BI because it:

## DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING - AI & ML

- Stores semi-structured and unstructured data
  - Supports real-time analytics
  - Scales horizontally
  - Integrates with Big Data platforms
  - Enables flexible reporting and dashboards
- 
- 

