**Day 3**: **21st May 2025**

**ACID Properties**:

ACID is a set of rules that help keep data safe and correct in databases, even when things go wrong like a power cut or system crash.

**Atomicity:**

Transaction is all or nothing

Either all operations complete or none do

**Consistency:**

Database remains in valid state before and after transaction

All rules/constraints are enforced

**Isolation:**

Transactions execute independently

Changes from one transaction don't affect others until committed

**Durability:**

Once committed, changes are permanent

Data survives system failures

These properties ensure data integrity and reliability in database systems.

**MongoDB - Simple Guide**

**What is MongoDB?**

MongoDB is a **NoSQL database** that stores data in **JSON-like documents** instead of tables (like SQL databases).

**Think of it like this:**

* **SQL Database** = Excel spreadsheet with rows and columns
* **MongoDB** = Collection of JSON files in folders

**Why Use MongoDB?**

✅ **Flexible** - No fixed structure required  
✅ **Fast** - Great for web applications  
✅ **Scalable** - Handles large amounts of data  
✅ **Developer-friendly** - Works naturally with JavaScript

**MongoDB Structure**

Database (like a folder)

├── Collection (like a file)

│ ├── Document (like a JSON object)

│ ├── Document

│ └── Document

└── Collection

├── Document

└── Document

**Example:**

ecommerce (database)

├── users (collection)

│ ├── {name: "John", age: 30}

│ └── {name: "Alice", age: 25}

└── products (collection)

├── {name: "Laptop", price: 999}

└── {name: "Phone", price: 699}

**Getting Started**

**1. Installation & Setup**

# Start MongoDB (after installation)

mongod # Start the database server

mongosh # Connect to database

**2. Basic Commands**

// See current database

db

// List all databases

show dbs

// Create/Switch to database

use mystore

// List collections in current database

show collections

**Basic Operations (CRUD)**

**Create - Adding Data**

// Add one document

db.users.insertOne({

name: "John Doe",

email: "john@email.com",

age: 30

})

// Add multiple documents

db.users.insertMany([

{name: "Alice", age: 25},

{name: "Bob", age: 35}

])

**Read - Finding Data**

// Find all users

db.users.find()

// Find specific user

db.users.find({name: "John Doe"})

// Find users older than 25

db.users.find({age: {$gt: 25}})

**Update - Changing Data**

// Update one user

db.users.updateOne(

{name: "John Doe"}, // find this user

{$set: {age: 31}} // change age to 31

)

// Update multiple users

db.users.updateMany(

{age: {$lt: 30}}, // find users under 30

{$set: {status: "young"}} // add status field

)

**Delete - Removing Data**

// Delete one user

db.users.deleteOne({name: "John Doe"})

// Delete multiple users

db.users.deleteMany({age: {$gt: 60}})

**Simple Query Examples**

**Finding Data**

// Find all

db.users.find()

// Find one

db.users.findOne({name: "Alice"})

// Find with conditions

db.users.find({age: 25}) // age equals 25

db.users.find({age: {$gt: 25}}) // age greater than 25

db.users.find({age: {$lt: 30}}) // age less than 30

**Common Operators**

| **Symbol** | **Meaning** | **Example** |
| --- | --- | --- |
| $gt | Greater than | {age: {$gt: 18}} |
| $lt | Less than | {age: {$lt: 65}} |
| $gte | Greater or equal | {age: {$gte: 18}} |
| $lte | Less or equal | {age: {$lte: 65}} |
| $ne | Not equal | {status: {$ne: "inactive"}} |

**Useful Methods**

**Counting & Limiting**

// Count documents

db.users.countDocuments()

db.users.countDocuments({age: {$gt: 25}})

// Limit results

db.users.find().limit(5) // Get first 5 users

// Sort results

db.users.find().sort({age: 1}) // Sort by age (ascending)

db.users.find().sort({age: -1}) // Sort by age (descending)

**Selecting Specific Fields**

// Only show name and email (hide \_id)

db.users.find({}, {name: 1, email: 1, \_id: 0})

// Hide specific fields

db.users.find({}, {password: 0})

**Real-World Example**

Let's create a simple blog database:

// Switch to blog database

use blog

// Add some posts

db.posts.insertMany([

{

title: "My First Post",

content: "Hello World!",

author: "John",

date: new Date(),

likes: 5

},

{

title: "Learning MongoDB",

content: "MongoDB is easy to learn",

author: "Alice",

date: new Date(),

likes: 12

}

])

// Find all posts

db.posts.find()

// Find posts by author

db.posts.find({author: "John"})

// Find popular posts (more than 10 likes)

db.posts.find({likes: {$gt: 10}})

// Update likes

db.posts.updateOne(

{title: "My First Post"},

{$inc: {likes: 1}} // Increase likes by 1

)

**Key Concepts to Remember**

**1. Documents are Flexible**

// These are all valid in the same collection:

db.users.insertOne({name: "John", age: 30})

db.users.insertOne({name: "Alice", age: 25, city: "NYC"})

db.users.insertOne({name: "Bob", skills: ["HTML", "CSS"]})

**2. Every Document has an \_id**

// MongoDB automatically adds \_id if you don't provide one

{

\_id: ObjectId("507f1f77bcf86cd799439011"),

name: "John",

age: 30

}

**3. Data Types**

* **String**: "Hello"
* **Number**: 25, 3.14
* **Boolean**: true, false
* **Array**: ["apple", "banana"]
* **Object**: {street: "123 Main St", city: "NYC"}
* **Date**: new Date()

**Common Mistakes to Avoid**

❌ **Wrong:**

db.users.find(name: "John") // Missing quotes around field

db.users.find({name = "John"}) // Using = instead of :

✅ **Correct:**

db.users.find({name: "John"}) // Proper syntax

**Quick Command Reference**

// Database

show dbs // List databases

use dbName // Switch database

// Collections

show collections // List collections

db.collection.drop() // Delete collection

// CRUD

.insertOne({}) // Add one document

.insertMany([]) // Add multiple documents

.find({}) // Find documents

.updateOne({}, {}) // Update one document

.deleteOne({}) // Delete one document

// Helpers

.countDocuments() // Count documents

.sort({field: 1}) // Sort (1=ascending, -1=descending)

.limit(5) // Limit results