INTERSKILL SOLUTIONS

MONGODB

DIWATE YASH
Interskill Solutions

LECTURE 1

WHAT IS A DATABASE?

- A DATABASE IS AN ORGANIZED COLLECTION OF DATA.
- EXAMPLE IN REAL LIFE:
- A COLLEGE DATABASE STORES STUDENT RECORDS (NAME, ROLL NO, MARKS).
- A BANK DATABASE STORES ACCOUNT INFO (BALANCE, TRANSACTIONS).

👉 WHY USE DATABASES?

- STORE DATA SAFELY
- RETRIEVE DATA QUICKLY
- MODIFY/UPDATE EFFICIENTLY

SQL VS NOSQL

INTERSKILL SOLUTIONS

Aspect	SQL (Relational DB)	NoSQL (MongoDB type)
Data Structure	Tables → Rows & Columns	Collections → Documents (JSON-like)
Schema	Fixed (must define table before inserting data)	Flexible (fields can vary from doc to doc)
Example Record	`INSERT INTO students VALUES (1, "Yash", 21, "CE")`	`{ "id": 1, "name": "Yash", "age": 21, "course": "CE" }`
Joins/Relations	Supports joins between multiple tables	Usually embedded/nested docs (less joins)
Transactions	Strong ACID compliance (Atomicity, Consistency, Isolation, Durability)	Eventual consistency, but flexible with scaling
Scaling	Vertical (buy a bigger server)	Horizontal (add more servers easily → sharding)
Best Suited For	Banking, Finance, ERP (structured data)	Social Media, IoT, Real-time apps (dynamic data)
Examples	MySQL, PostgreSQL, Oracle, SQL Server	MongoDB, Cassandra, CouchDB, Firebase

📍 **Analogy (Easy to Remember)**

- **SQL = Excel Sheet (rows & columns, fixed format).**
- **NoSQL = Diary (each page can have different details).**

WHY DO WE NEED DATABASES?

INTERSKILL SOLUTIONS

- Store large amounts of data safely
- Search & filter data quickly
- Update and delete easily
- Handle multiple users at the same time
- Prevent data loss

INTERSKILL SOLUTIONS

WHAT IS MONGODB?

- MongoDB = NoSQL Database
- Stores data in documents (JSON-like)
- Hierarchy:
- Database → contains collections
- Collection → contains documents
- Document → stores data

📍 Example:

```json
{
  "firstname": "Yash",
  "lastname": "Diwate",
  "age": 21,
  "course": "Computer Engineering"
}
```

MONGODB FEATURES

- Flexible schema (can add new fields anytime)
- High performance & scalability
- Stores unstructured, semi-structured data
- Widely used in modern web apps
- Works great with JavaScript (Node.js)

INSTALLING MONGODB

LOCAL SETUP

- DOWNLOAD MONGODB COMMUNITY EDITION
- INSTALL MONGODB COMPASS (GUI)

CLOUD SETUP

- USE MONGODB ATLAS (FREE ACCOUNT)
- EASY TO CONNECT WITH NODE.JS APPS

INTERSKILL SOLUTIONS

INTERSKILL SOLUTIONS

MONGODB

DIWATE YASH
Interskill Solutions

LECTURE 2

CRUD OPERATIONS IN MONGODB

- C → CREATE → ADD NEW DATA
- R → READ → RETRIEVE DATA
- U → UPDATE → MODIFY EXISTING DATA
- D → DELETE → REMOVE DATA

\📍 CRUD = HEART OF ANY DATABASE OPERATION

STARTING MONGODB

- OPEN MONGODB COMPASS (GUI) OR USE MONGO SHELL (CLI).

TO SEE ALL DATABASES

```
> show databases
< Employee          116.00 KiB
  Loginpage           8.00 KiB
  admin              40.00 KiB
  blogDB            112.00 KiB
  blogsite            8.00 KiB
  book               72.00 KiB
```

CREATE & SWITCH DATABASE

- MONGODB DOESN'T CREATE DB UNTIL DATA IS INSERTED!
- SWITCH TO / CREATE NEW DATABASE:

TO SEE ALL DATABASES

```
> use blogsite
< switched to db blogsite
> db
< blogsite
```

CREATE COLLECTION

- A COLLECTION = TABLE IN SQL
- CREATE COLLECTION STUDENTS:

```
< switched to db test
> db.createCollection("students")
< { ok: 1 }
```

- SHOW COLLECTIONS:

```
> show collections
< students
  users
```

INSERT DATA (CREATE)

INTERSKILL SOLUTIONS

INSERT ONE DOCUMENT

```
> db.students.insertOne(
    { name: "Yash", age: 21, course: "CE" }
  )
< {
    acknowledged: true,
    insertedId: ObjectId('68d5a95334cebc64716c8173')
  }
```

INSERT MANY DOCUMENTS

```
> db.students.insertMany([
    { name: "Sakshi", age: 20, course: "IT" },
    { name: "Prince", age: 22, course: "CS" },
    { name: "Anush", age: 23, course: "Mechanical" }
  ])
< {
    acknowledged: true,
    insertedIds: {
      '0': ObjectId('68d5a97734cebc64716c8174'),
      '1': ObjectId('68d5a97734cebc64716c8175'),
      '2': ObjectId('68d5a97734cebc64716c8176')
    }
  }
test>
```

: READ DATA (RETRIEVE)

INTERSKILL SOLUTIONS

FIND ALL DOCUMENTS

```
> db.students.find()
< {
    _id: ObjectId('68d5a95334cebc64716c8173'),
    name: 'Yash',
    age: 21,
    course: 'CE'
  }
  {
    _id: ObjectId('68d5a97734cebc64716c8174'),
    name: 'Sakshi',
    age: 20,
    course: 'IT'
  }
  {
    _id: ObjectId('68d5a97734cebc64716c8175'),
    name: 'Prince',
    age: 22,
    course: 'CS'
  }
  {
    _id: ObjectId('68d5a97734cebc64716c8176'),
    name: 'Anush',
    age: 23,
    course: 'Mechanical'
  }
```

FIND WITH CONDITION

```
> db.students.find({ course: "IT" })
< {
    _id: ObjectId('68d5a97734cebc64716c8174'),
    name: 'Sakshi',
    age: 20,
    course: 'IT'
  }
```

PRETTY PRINT

```
> db.students.find().pretty()
< {
    _id: ObjectId('68d5a95334cebc64716c8173'),
    name: 'Yash',
    age: 21,
    course: 'CE'
  }
  {
    _id: ObjectId('68d5a97734cebc64716c8174'),
    name: 'Sakshi',
    age: 20,
    course: 'IT'
  }
  {
    _id: ObjectId('68d5a97734cebc64716c8175'),
    name: 'Prince',
    age: 22,
    course: 'CS'
```

UPDATE DATA (MODIFY)

INTERSKILL SOLUTIONS

UPDATE ONE DOCUMENT

```
> db.students.updateOne(
    { name: "Yash" },
    { $set: { course: "AI & ML" } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
test>
```

UPDATE MANY DOCUMENTS

```
> db.students.updateMany(
    { course: "CS" },
    { $set: { course: "Computer Science" } }
  )
< {
    acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0
  }
test>
```

DELETE DATA (REMOVE)

INTERSKILL SOLUTIONS

DELETE ONE DOCUMENT

DELETE MANY DOCUMENTS

```
> db.students.deleteOne({ name: "Prince" })
< {
    acknowledged: true,
    deletedCount: 1
  }
> db.students.deleteMany({ course: "IT" })
< {
    acknowledged: true,
    deletedCount: 1
  }
```

📘 MONGODB – USEFUL QUERY OPERATORS

INTERSKILL SOLUTIONS

1. `$gt` → Greater Than

👉 Used to find documents where a field's value is **greater than** the given value.

```js
db.students.find({ age: { $gt: 20 } })
```

📌 Example: Returns all students whose **age > 20**.

---

2. `$lt` → Less Than

👉 Used to find documents where a field's value is **less than** the given value.

```js
db.students.find({ age: { $lt: 22 } })
```

📌 Example: Returns all students whose **age < 22**.

3. `$gte` → Greater Than or Equal To

👉 Includes values **greater than OR equal to** the given value.

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

📌 Example: Returns students aged **21 or older**.

---

4. `$lte` → Less Than or Equal To

👉 Includes values **less than OR equal to** the given value.

```js
db.students.find({ age: { $lte: 20 } })
```

📌 Example: Returns students aged **20 or younger**.

5. `$and` → Logical AND

👉 Combines multiple conditions – all must be true.

```js
db.students.find({
  $and: [
    { age: { $gt: 20 } },
    { course: "CE" }
  ]
})
```

📌 Example: Finds students **older than 20 AND in CE course**.

---

6. `$or` → Logical OR

👉 Matches documents where **at least one condition** is true.

```js
db.students.find({
  $or: [
    { course: "IT" },
    { age: { $lt: 21 } }
  ]
})
```

📌 Example: Returns students who are **in IT OR younger than 21**.

7. `$not` → Negation

👉 Excludes documents that match the condition.

```js
db.students.find({
  age: { $not: { $gt: 21 } }
})
```

📌 Example: Returns students where age is **NOT greater than 21** (so ≤ 21).

🔥 Extra Examples (for better practice)

Between Two Values

```js
db.students.find({ age: { $gte: 20, $lte: 22 } })
```

📌 Students aged **20, 21, or 22**.

ACTIVITY (HANDS-ON)

INTERSKILL SOLUTIONS

👉 STUDENTS MUST DO:
CREATE A NEW DB COLLEGE.
CREATE COLLECTION FACULTY.
INSERT AT LEAST 5 DOCUMENTS WITH FIELDS: NAME, SUBJECT, EXPERIENCE, AGE.
FIND ALL FACULTY TEACHING MATH.
UPDATE ONE FACULTY'S EXPERIENCE.
DELETE ONE FACULTY.
FIND FACULTY WITH AGE > 30.