DBMS

Types of DBMS

1. Hierarchical DB → like tree fashion
   {child - parent}
2. Network DB → [stores]
   [clerks] [customers]
3. Relational DB ⊛
   tabular form
   [Server] → Client
4. Object oriented DB
   ↳ uses oops concept
5. Graph DB
   ↳ based on graph
6. ER Model DB
7. Document DB
8. NoSQL DB. ⊛ non-relational DB

- non-relational DB

1. Document DB
2. Graph DB
3. wide column DB
4. key-value DB

SQL — NoSQL

MySQL — MongoDB

↳ 2009 feb → mongo launch

↳ non relational

↓

Document oriented

↓

BSON to query database

↓

Binary JSON

BSON is just binary JSON (a superset of JSON with some more data types, most importantly binary byte array

BSON Senior

JSON Junior

in SQL, MySQL } → Table

field

collection

in mongo

```
[
  { "username": "Darshan",
    { //data }
  },
]
```

Document

SQL		NoSQL
table	→	collection
Row	→	Document
column	→	field

in mongo ?
Flexibility & Scalability

Command
→ show dbs

⇒ use cscoaner } coection
↳ name

→ एक collection than & only that will show in db

collection add for adding 1 doc data

db. collectionname . insertOne ({ [ ] })

name : "ReactJS",
type : "Front End",
videos : 80
active : true
↓
field : value

code
db → active database
show collections → name suggest

document आये
d. b. collectionname . find ( )

exit ( ) → name suggest

drop database
db. dropDatabase()

Case Sensitive

CRUD Operation

Creat — insertOne({ })
Read — insertMany([{}, {}, {}])
update
Delete

Read

d.b.collection.find(query, projection)

Mongoose

Object Data Modeling [ODM]
↓
manage relationship between data
↓
small codes
↓
provide schema validation
↓
translate between object in code

CTRL + J → terminal

For Project

make file app.js

npm init -y → to add packag.json

than

npm i mongoose

folder name must be diff. from dependency

node version checking ⇒ node -v → in terminal

Scratch

npm init or npm init -y

↓

npm install express mongoose

↓

both will install

↓

run index.js → node index.js

CTRL + C → to stop server

required to restart server fer..

don't need to do manuall that

↓

nodemon → install

↓

npm i --save-dev nodemon

in package.json

add this

```
"scripts": {
    "start": "nodemon index.js",
    // bla bla - - -
}
```

↓

npm start

shortcut: rs → hit enter

// localhost:27017/

↓

give error

↓

connect ECONNREFUSED ::1:27017

↓

use this

// 0.0.0.0:27017/

Schema: defines the structure of your collection documents

Schema
↓
make folder along with package.json
↓
inside folder
↓
user.js or usermodel.js → folder name
↓
mongoose module required
↓
const schemaName = new mongoos.Schema

Model

model('collection_name', Schema);

(EGHCH) ← model → take schema
CRUD
op are require
↓
apply it to each document in its collection

db -> users -> documents

CRUD Op → routes required

routes: localhost:5000/user/create

Routers contain Routes + callback

if this is separate

called callback

controller

CRUD

creat () one or more document isertion

save () one document isertion

second arg. option arg. ⇒ Passing one or multiple ⇒ must be in form of array

Create (docs, option, callba)

Data must be follow schema

Pascal ⇒ Conversi

Camel: document

Update Query

```
const updateDocument = async() => {
    const result = await Playlist.updateOne (
        {_id}, {
        $set : {
            name : "Darshan"
        }
        });

    updateDocument ("64 ... ");
```

<-> Print all documents first

find() -> print up to 20 documents

Use ForEach Loop

Syntex: db.studentData.find().forEach((student) => {printjson(student)})