MongoDB

→ It is schemales Database management system and it is very useful

→ Relational data follow schema due to that it has very large redundancy, normalisation issue while mongodb is JSON object based and only need object to store data

→ mongodB is flexible

→ It also so much hard to delete a column (you have to remove it from each row

→ many thing in Relationdl data is null so it also not memory efficiant. also, such can used efficiantly in mongodB

⇒ NOSQL mean non Relationdl database

⇒ For using Dependent Structure of sql we have to use joins & many more operations so in order to avoid it we use NOSQL data

NOSQL ⇒ we save data in a key value pair

→ no concept of blank dataset (data leakage)

→ no need to do normalisation to remove redundancy

mongo ⇒ Shell to run mongoDB command

mongoD ⇒ Server which run mongoDB

Compass is GUI app for mongoDB

mongod commands

mongod

to run shell ⇒ mongo

&

⇒ sudo systemctl status mongod

→ to check whether installed successfully or not

⇒ sudo systemcntl start mongod

→ used to start mongodb server

⇒ sudo systemcntl restart mongod → to restart

⇒ sudo systemcntl stop mongod. ⇒ to stop mongoDB

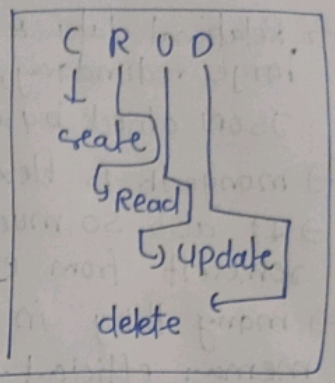
Basic of functions

Basic command

CRUD → Create / Update, Delete / Read

Analogies : Database operation

MongoDB		SQL
Database	→	Database
Collection	→	table
document	→	row
field	→	column



cls ⇒ to clear logs

show dbs; ⇒ show all databases in mongo

use dbname ⇒ switch to specific database

db ⇒ give current db

use db ⇒ can used to create new database or switch both

If use dbName dbName exist in show database it will switch else it create one

exit ⇒ to go out of mongo shell

⇒ In show dbs database is shown only if it have some collection (table) else it will not shown

⇒ to delete a database just go inside it

⇒ use database,

⇒ db.dropDatabase() ⇒ this will delete that database

always check database before drop

Such like that you can create database, switch it & delete it also. good!

Basic collection commands

database → collection → object data (hierarchy)

collection is like table inside database

⇒ So In database say school
so school ⇒ database & its part like student, class, courses, teacher, ⇒ all are containing in table
so that are collections inside mongoDB

⇒ Default database in mongo is (test)

→ show collections ⇒ this will show collection in a db

→ db.createCollection("provide Name") → command to create colln

{ok : 1} → responce mean collection created successfully

→ ~~db.DatabaseName.drop()~~ ⇒ this will delete all the ~~data~~ colln in database

↑ return true if done

- db.collectionName.drop() → we have to specify collection name

---

Create operations

(Documents ⇒ row data)

⇒ In mongodb we pass key value pairs to insert a document

fields (columns) ← take dict

⇒ db.collectionName.insert({"name": "grs", 'Gender': "male"})

↑ that will add single entry inside our collection

⇒ the NOSQL do not have specific schema.

writeResult({nInserted: 1}) → If insert done

⇒ db.collectionName.find() ⇒ command used to show content in collection

⇒ mongoDB also assign the id for each object automatically you can also pass own id with key value pair

"_id" : Id you want to give

⇒ ⇒ In mongodB it does matter that a field should have same universal datatype, for each document there can be different datatype for each field

#Inserting multiple entries at once ⇒ multiple docs insertion at once

`db.collectionName.insertMany([])` → you can pass array

⇒ insert multiple entries at once by insertMany

⇒ easy and useful ⭐

→ we get acknowledgment and Id

⇒ If we give many object we get objectId which have close serial name

⇒ InsertMany() ⇒ provide good efficiency

`db.collectionName.find().pretty()` → good visualization of data

pretty ~~pretty~~ `find().pretty()` good

→ In mongod we pass JSON objects

⭐ Basic update operation → used to rectify the wrongly inserted data

for each document it can have different schema

⭐ Update Document (single filter)

to update data you need a filter like a id, name or anything.

all field will replaced by this

⇒ db.collectionName.update({searchCriteria}, {updation})

we have to pass object at both place

easy

you can pass single filter or multiple filter

you ~~will get~~ {nmodified : count} ⇒ to how many mat

~~⇒ If yo~~ with update you only can modify existing field & will not able to add new fields

~~⇒ you also can't able to change object in search~~

⇒ so all modifiation will updated and old will deted (overwrite take place)

⇒ update: can't be used to so all data in replace block of update will ow

(See coding & you will know.)

~~upd~~ db.collectionName.update({search},{replace})

update will only change one object while (updateMany) will update multiple instan at one

db.collectionName.updateMany({search},{replace})

⇒ upsert this third paramter in mongodb

⇒ upsert will add that entry if no match found

⇒ db.collectionName.Update({search},{replace},{upsert:true})

(Imp) ⇒ so if update will not match it will create new object.

⇒ we can pass multiple search criteria also in same object

{Search? = {"criteria1":"val1", "criteria2":"val2"}}

Such like that we pass two criteria in search

Passing multiple values inside ~~database~~ search is called as multiple filtering

we can also "_id" to update record.

the search block all things are case sensitive

⇒ JSON objects are case sensitive (keep that in mind)

Say { 'abc': { "m": 1, "n": 2 } }
↑ abc
So to search for it

there are large no. of functions exist in mongoDB read it

db.collectionName.find({ "abc.m": 1 })

dot operator used to fetch inside object & should be inside same quote

⇒ you can try any search query inside find()

Read operations (do quiz)

⇒ we can used Read operations and with those we try to read database
⇒ can used to extract information
⇒ we use find to read data.

you can also do filters to search

⇒ db.collectionName.find() → to Read data

⇒ db.collectionName.find().pretty() → formatted fashion → this will find & display data in good elaborated format

Deletion in Mongo

we use remove keyword to delete & syntax is same as read

⇒ db.collectionName.remove() → you can do search

↑ you can't use directly you have to specify some

⇒ to remove whole collection

~~db.removeCollection~~

easy

db.collectionname.drop()

you can add multiple filters also