What is MongoDB?

Mongodb is database

Mongodb is about flexibity as opposed to MySQL, which there is no schema, for example one of the columns are missing.

JSON data format

\*\* show databases

Show dbs

\*\* connect to shop database

Use shop

Show collections (showing the collections of a database)

\*\* Write a new document into the database

db.products.insertOne({name:"A book", price: 12.99})

So in the database, and in the products collection, we added the this document

- db.products.find()

Cls to clear the window

\*\*\*\*\*\*\*\*\* ALT + left/right arrow to go faster in the word

CRUD: Creat, Read, Update, and Delete

MongoDB automatically add the id to the document, but we can have our own id by typing \_id and have our own id, either string or number or whatever.

1. INSERT:

- insertOne(data, options)

Db.products.insertOne({name: “Hamid”, price: 12, details:{cpu:”Intel”}})

- insertMany(data, options)

db.flightData.insertMany([

... {

... "departureAirport": "MUC",

... "arrivalAirport": "SFO",

... "aircraft": "Airbus A380",

... "distance": 12000,

... "intercontinental": true

... },

... {

... "departureAirport": "LHR",

... "arrivalAirport": "TXL",

... "aircraft": "Airbus A320",

... "distance": 950,

... "intercontinental": false

... }

... ]

... )

1. READ:

-find(filter, options)

-findOne(filter, options)

1. UPDATE:

-updateOne(filter, data, options)

db.fightData.updateOne({distance:12000},{$set:{marker:"delete"}})

--- multiple entry update

db.fightData.updateOne({distance:12000},{$set:{marker:"delete", age: 37}})

-updateMany(filter,data, options)

db.flightData.updateMany({}, {$set:{status: "on-time"}})

-replaceOne(filter, data, options)

1. DELETE

-deleteOne(filter, options)

db.fightData.deleteOne({departureAirport:"TXL"})

-deleteMany(filter, options)

Db.products.deleteMany({}): will delete all of the documents of products.

db.flightData.updateMany({}, {$set{marker:"todelete"}})

\*\*\*\*\*\*\*\*\*\*\*\*\*

FIND:

db.flightData.find().pretty()

db.flightData.find({name:"Max"})

db.flightData.find({ismale:true})

\*\*\*\*\*\*\* Greater Than

db.flightData.find({distance:{$gt: 11000}}).pretty()

Just the first maching element:

db.flightData.findOne({distance:{$gt: 11000}}).pretty()

Note: find() method does not give you the data itself, but it will give you the cursor to the data. In order to get the data, you may wanna do something like that: (find automatically get the first 20 elemnts)

Db.passengers.find().toArray()

See the pattern for having two elements for one file of a document:

db.passengers.updateOne({name: "Albert Twostone"}, {$set:{hobbies: ["sports", "cooking"]}})

We can even fetch the specific field:

db.passengers.findOne({name: "Albert Twostone"}).hobbies

db.flightData.find({"status.description":"on-time"}).pretty()

-- For the nested fileds, use quotation for the filed!!!

\*\*\*\*\*\*\*\*\*\*\* Projection

We wanna just grab name:

1 means include it, and 0 means do not include it:

db.passengers.find({},{name: 1}).pretty()

db.passengers.find({},{name: 1, \_id: 0}).pretty(): I wanna name, and I don’t wanna id. Note that, for others rather than \_id, the default value is 0.

\*\*\*\*\*\*\*\*\*\*\* In order to get rid of a database

Use DATBASE

Db.dropbDatabsae()

\*\*\*\*\*droping a collection

Db.orders.drop()

Dropping orders collection from the under study database

\*\*\*\*\*\*\*\*\*\*

Useful links:

<https://docs.mongodb.com/ecosystem/drivers/>

<https://docs.mongodb.com/manual/tutorial/getting-started/>

\*\*\*\*\*\*\*\* Null

Dbs.products.insertOne({name: “hamid”, price: 12.99, details: null})

This is used when you wanna use the SQL structure and give all of the documents the same schema!!!

\*\*\*\*\*\*\* statistics of a database

Db.stats()

\*\*\*\*\* MongoDB data types

<https://docs.mongodb.com/manual/reference/bson-types/>

\*\*\*\*\*\*\*\*\* Variable

Var dsid = db.patient.findOne().diseaseSummary

Db.diseaseSummary.findOne({\_id = dsid})

\*\*\*\* Joining the collections using lookup

Db.books.aggregate([{$lookup:{from: “Authors”, localField: “aths”, foreignField: “\_id”, as: “Creators” }}])

So from “Authors” collection, and using the “aths” filed, aggregate the two collections using the \_id and name it as Creators!