Mongo DB

-mongo-DB is document-based DB (noSQL) uses BSON(binary JSON) which have some advantages over json

Search in MongoDB for: msi(mongo community server(mongod))—shell(mongosh acts as a client)---compass(GUI)----services(to ensure for installation)

\*connection string: mongodb://127.0.0.1:27013 for: server://ip\_address:port

Or use ‘mongosh’ simply to run the default one.

\*Mongo shell commands:

-show dbs--> will show(admin, config, local) initially ,later it will show the whole created db

-use database\_name -->create new db if doesn’t exist else this command switch to this db.

-db.createCollection(“posts”)-->create table named posts

-db.posts.drop()🡪delete collection

-db.dropDatabase()

-db.posts.insertOne({id:1, UID:1, text:”first post”}) //object-id index will be created automatically

-db.posts.find()

-db.posts.insertMany([{id:2, UID:1, text:”second post”, comments:[{UID:1, content:”blblb”}], image:”img\_url”, likes:100}, { id:3, UID:2, text:”first post”, comments:[{UID:2, content:”ffff”}], image:”img\_url”, likes:150, location:{“Gov”:”Giza”, “city”:”gor\_city”}}]) 🡪insert array of objects(rows) in the collection.

-db.posts.find({UID:2})

-db.posts.find({“location.city”:”dishna”})

-db.posts.find({likes:{$lte:150}}) //lte:less than or equal – ne:not equal – gt:greater than – gte:greater than or equal

-db.posts.find({location:{$exists:true}})

-db.posts.find({$and:[{id:1},{UID:2}]})

-db.posts.find({$or:[{id:1},{UID:2}]})

-db.posts.find({likes:100},{text:”first post”, id:1}) 🡪the text property with set value “first post” will not effect, but id will be included with its own value, We can’t exclude property from including projection(either all 0 or all 1).

-db.posts.findOne()🡪return the first object

-db.updateOne({likes:100},{$set:{private:true}})🡪update the first object that match the condition in the first passed object and add or update the property private with the set new value

-db.updateMany({likes:{$lte:150}}, {$set:{reach:”low”}})

-db.updateMany({reach:{$exists:true}}, {$unset:{reach:”low”}})

-db.updateMany({UID:2}, {$push:{comments:{UID:3, content:”jjjs”}}})

-db.posts.updateMany({},{$inc:{likes:50}}) //increment, dec:decrement //will set this update to all objects

-db.posts.deleteOne({}); //delete the first object

-db.posts.findOneAndUpdate({UID:3},{$set:{likes:200},{returnDocument:”after”}) //there isn’t updateOne()

-db.posts.findOneAndDelete({UID:2},{returnDocument:”after”})

-db.posts.find({comments:{$size:3}})

Indexing

-db.posts.createIndex({id2:-1},{unique:true}) //create unique index with descending order

-db.posts.createIndex({id3:1}, {expireAfterSeconds:220}) //create a TTL(time-to-live) index with ascending order (must be always ascending) doesn’t appear when we retrieve documents

-db.posts.getIndexes()

-db.posts.find({id:2}).explain() //Shows how MongoDB executes this query. Useful for analyzing **performance** and **index usage**.

-db.posts.dropIndex(“id2\_-1”) //note:id2 is field-name, but id2\_-1 is index-name

-db.posts.getIndexes()