**MONGO DB lessons**

1. Mongo db is a collection based No SQL Database
2. Many databases in a mongo server, a database has many collections, a collection has many documents. A Document is a set of key, value pairs. Documents have a dynamic schema. Example:



**\_id** is a 12 bytes hexadecimal number which ensures uniqueness of every object. First 4 bytes for current timestamp, next 3 bytes for machine id, next 2 bytes for process id of MongoDb server and remaining 3 bytes are simple incremental value.

1. Download from <https://www.mongodb.org/downloads> and then install
2. Go to the mongo directory till bin and run commands:

mongod.exe --dbpath "d:\set up\mongodb\data"

mongo.exe (in different terminal)

1. Mongo db commands

db.test.save({a:1}) #save a new collection and new document in db test, test db is the default db in mongo

db.test.find() #get all collections

db.stats() # get all info

use db\_name #creates a new db or switches to existing

db.movie.insert({‘name’:’tutorial’}) #insert a document to see created db

db.movie.save({‘name’:’tutorial’}) #insert a document to see created db

db #gives the current db name

dbs #gives a list of all databases

db.dropDatabase() #drop an existing database

db.createCollection(collectionname, options) # options are optional

db.createCollection("mycol", { capped : true, autoIndexId : true, size : 6142800, max : 10000 } )

show collections

db.t1.insert({“name”:”doll”}) #creates t1 collection if not present. Inserts the document in t1 collection

db.t1.drop() #drops a database

1. Data types in Mongo db

* **String** − This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.
* **Integer** − This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
* **Boolean** − This type is used to store a boolean (true/ false) value.
* Double − This type is used to store floating point values.
* **Min/ Max keys** − This type is used to compare a value against the lowest and highest BSON elements.
* **Arrays** − This type is used to store arrays or list or multiple values into one key.
* **Timestamp** − ctimestamp. This can be handy for recording when a document has been modified or added.
* **Object** − This datatype is used for embedded documents.
* **Null** − This type is used to store a Null value.
* **Symbol** − This datatype is used identically to a string; however, it's generally reserved for languages that use a specific symbol type.
* **Date** − This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
* **Object** **ID** − This datatype is used to store the document’s ID.
* **Binary** **data** − This datatype is used to store binary data.
* **Code** − This datatype is used to store JavaScript code into the document.
* **Regular** **expression** − This datatype is used to store regular expression.

1. Insert multiple documents in an array

db.tutorial.insert([{"name":"t1","duration":"6","tags":["no sql","db"]},{"name":"t2","duration":3,"tags":["no sql","db"]}])

1. If we specify \_id in document in save(), then it will replace all the data of the document of that \_id else it will work same as insert()
2. To see all the documents in a collection, db.c1.find() or db.c1.find().pretty()

db.c1.findOne() returns first document

db.COLLECTION\_NAME.find({},{KEY:1})

find({}, {“title”:1,\_id:0})

1. Where clauses

|  |  |  |  |
| --- | --- | --- | --- |
| Operation | Syntax | Example | RDBMS Equivalent |
| Equality | {<key>:<value>} | db.mycol.find({"by":"tutorials point"}) | where by = 'tutorials point' |
| Less Than | {<key>:{$lt:<value>}} | db.mycol.find({"likes":{$lt:50}}) | where likes < 50 |
| Less Than Equals | {<key>:{$lte:<value>}} | db.mycol.find({"likes":{$lte:50}}) | where likes<=50 |
| Greater Than | {<key>:{$gt:<value>}} | db.mycol.find({"likes":{$gt:50}}) | Where likes>50 |
| Greater Than Equals | {<key>:{$gte:<value>}} | db.mycol.find({"likes":{$gte:50}}) | Where likes>=50 |
| Not Equals | {<key>:{$ne:<value>}} | db.mycol.find({"likes":{$ne:50}}) | Where likes != 50 |

1. In the find() method, if you pass multiple keys by separating them by ',' then MongoDB treats it as AND condition.

db.mycol.find({$and:[{"by":"tutorials point"},{"title": "MongoDB Overview"}]})

db.mycol.find({$or:[{"by":"tutorials point"},{"title": "MongoDB Overview"}]})

1. Update() updates into a collection and save() replaces whole document db.COLLECTION\_NAME.update(SELECTION\_CRITERIA, UPDATED\_DATA)

Eg. db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}})

Update mycol set title=”New MongoDB Tutorial” where title=”MongoDB Overview”

By default, mongo updates only one document, for updating multiple docs, set multi:true

Eg. db.mycol.update({'title':'MongoDB Overview'},{$set:{'title':'New MongoDB Tutorial'}}, {multi:true})

1. Method to remove documents from collection: db.col.remove(deletion criteria)

To remove just 1 doc: db.col.remove(deletion criteria, 1)

Eg. db.mycol.remove({'title':'MongoDB Overview'})

If no deletion criteria is given, then all documents will be removed from the collection

1. To limit the number of records, db.col.find().limit(20)