**Insersiones:**

db.collections.insert(document) –insert document in our database

db.collections.insert([documet]) –insert many document in our collections

**Selecciones**

db.collections.find({where},{filds\_show})

db.users.find( { nombre : “Test”, edad : { $ne : 22 } } , { edad : 1, edad : 1, \_id : 0} )

WHERE FIELDS\_SHOW

db.collections.find() – Select all documents of a collections

db.collections.findOne() – Select the first document of a collections

db.collections.find({nombre : “Javier”, edad : 28}) – el primer parametro de find method es el where, el cual es un objeto js

db.users.find( { edad:{ $ne : 22 } } ) – Get all documents of a collections, where edad !=22

db.users.find({nombre:"Test uno", edad : { $ne : 22} }) – select all documents where nombre=”Test uno” and edad !=22

db.users.find({ name : /vi/}) – devuelve los documentos q contengan en su campo name la subcadena vi

db.users.find({ email : /.com$/}) – find all documents thar ending with ‘.com’ in email field.

Db.users.find( { email : / test/}) – esto es lo contrario de lo anterior.

**Updates documents**

db.users.save(document) – este método busca si el object tiene \_id : ObjectId(xxxxx), si lo tiene y coincide con algún document lo actualiza, sino lo inserta.

db.users.update( { nombre : “Test” } , document) – primer parametro es el where, Segundo parametro el document q quiero insertar

db.users.update( { nombre : “Test” } , { $set : { nombre : “CF”} } ) – en este caso actualizamos un campo especificamente, buscamos todos los documentos con nombre “Test” en mi colección de usuarios y actualizamos el campo nombre. Si no ponemos el tercer parámetro { multi : true}, solo se actualiza el primer documento encontrado.

db.users.update({},{},{})

db.users.update( { nombre : “Test” } , { $set : { nombre : “CF”} }, { multi : true } ) – en este caso actualiza el campo nombre de todos documents encontrados con nombre Test

ex1: db.users.update({},{ $set:{nombre : "CF2", edad : 28, example : 56 }},{multi: true})

ex2: db.users.update({},{ $unset:{ example : 1 }},{multi: true}) – de esta manera elimino el campo example en la collections users

**delete documents**

db.users.remove({where})

db.users.remove({nombre: “Test”})—Elimina todos los documentos de la collections users con nombre “Test”

db.users.remove({}) -- Elimina todos los documentos de la collections users

db.users.drop() – Elimina la collection

db.dropDatabase() – Elimina mi db

**Between**

Operadores :

1. $gt – greater than
2. $gte – greate than equals
3. $lt – less than
4. $lte – less than equals.

db.collections.find( **{** **field : { $gt : number, $lt : number }** …..**field n : { $lte : Number }** **}** )

condiction 1 **and condiction n**

**where**

Ex:

db.books.find( { first\_edition : { $gt : 2000 } } ) find all books >= 2000 year

Db.books.find( { first\_edition : { $gt : 2000, $lt : 2018 }, sales\_number: { $lt : 1000 } } ) find all books where first\_edition is between 2000 and 2018 and sales\_number >1000

**Arrays**

var array = [1,2,3]

var usuario = { nombre : “TEST”, valores : array }

db.users.insert( usuario )

db.users.update( {where}, { $set : { valores : array }} ) – esto primero busca todos los documentos en la colections users, si en la busqueda encuentra el campo valores lo actualize con array, si no le pongo el tercer parametro a la function update { multi : true }, modificara solo el 1er documento.

db.users.update( {where}, { $addToSet : { valores : val }} ) – en este caso añadirá al attr valores los elementos val (esto nos sirve si queremos insertar varios elem de un golpe)

db.users.update( {where}, { $push : { valores : val }} ) – en este caso añadirá al attr valores los elementos val

db.users.update( {where}, { $push : { valores : { $each : [5,6] } }} ) – insertar varios elementos. En este caso se insertan en el campo valores todos los elementos q están en el each.

db.users.update( {where}, { $addToSet : { valores : { $each : [5,6] } }} ) – esto hace lo mismo solo q si el elem existe no lo inserta

db.users.update( {where}, { $push : { valores : { $each : [5,6], $position : 4 } }} ) – inserta los valores 5 y 6 en la pos 4 del array valores

db.users.update({}, { $push : { valores : { $each : [95,96], $sort :1 } } }) – esto inserta los elmts y ordena el array

db.users.update({nombre: "CF2"},{ $pull : { valores : 95 } }) – en este caso se elimina el valor 95 del array q esta en la colección de usuarios

db.users.update({nombre: "CF2"},{ $pull : { valores : { $gte : 50 } } }) – de esta manera eliminamos los valores =>50

db.users.update({nombre: "CF2"},{ $pullAll : { valores : [9,1,8] }}) – de esta manera eliminamos todos los elementos q están dentro del arreglo

**selectArray**

db.users.find( {},{ valores : { $slice : 3}}) – devuélvame todos los elementos de la colección de usuarios y solo muéstrame del campo array hasta la posición #3 en adelante.

db.users.find({},{ valores : { $slice : [1,3] }}) – devuelve todos los elementos en el rango pasado al slice, en este caso nos devuelve de la collections users todos los documentos y en el campo valores solo los elementos desde el 1 al 3

db.users.find({ valores : { $in : [ “Mongoose”, “Python”,”Django”,”Nodejs”,”Expressjs”] }}, { \_id : false, nombre : false, valores : true } ) – devuelve todos los documentos q en el attr valores(q en este caso es un array) se encuentren dentro del array pasado como valor de la llave $in, si ningún elemento del array valores se encuentra dentro del otro arreglo, esta consulta no devolverá nada.

db.users.find({ valores : { $nin : [ “Mongoose”, “Python”,”Django”,”Nodejs”,”Expressjs”] }} ) – en este caso es todo lo contrario

**Group by:**

db.users.aggregate([ { $group : { “$item”} } ])

db.item.aggregate(

[

{ $group : { \_id : "$item", "repetidos" : { $sum : 1 } } }

]

)

db.collection.aggregate**(pipeline, options)**