Mongo practiced screen:

Crud:

1)db.COLLECTION\_NAME.insert(document)

2)db.COLLECTION\_NAME.find()

2b) The basic syntax of **find()** method with projection is as follows −

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

Ex: >db.mycol.find({},{"title":1})

2c) Please note **\_id** field is always displayed while executing **find()** method, if you don't want this field, then you need to set it as 0.

>db.mycol.find({},{"title":1,\_id:0})

3) db.COLLECTION\_NAME.update(SELECTION\_CRITERIA, UPDATED\_DATA)

The update() method updates the values in the existing document.

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

3b) db.COLLECTION\_NAME.save({\_id:ObjectId(),NEW\_DATA})

The **save()** method replaces the existing document with the new document passed in the save() method.

Ex;

db.mycol.save(

{

"\_id" : ObjectId(5983548781331adf45ec7), "title":"Tutorials Point New Topic",

"by":"Tutorials Point"

}

)

4) db.COLLECTION\_NAME.remove(DELLETION\_CRITTERIA)

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

Note:

remove() method accepts two parameters. One is deletion criteria and second is justOne flag.

* **deletion criteria** − (Optional) deletion criteria according to documents will be removed.
* **justOne** − (Optional) if set to true or 1, then remove only one document.

## 4a) Remove Only One

If there are multiple records and you want to delete only the first record, then set **justOne** parameter in **remove()** method.

>db.COLLECTION\_NAME.remove(DELETION\_CRITERIA,1)

## 4b) Remove All Documents

If you don't specify deletion criteria, then MongoDB will delete whole documents from the collection. **This is equivalent of SQL's truncate command.**

>db.mycol.remove()

Crud extra pts ;

1)To insert the document you can use **db.post.save(document)** also. If you don't specify **\_id** in the document then **save()** method will work same as **insert()** method. If you specify \_id then it will replace whole data of document containing \_id as specified in save() method.

7) By default, MongoDB will update only a single document. To update multiple documents, you need to set a parameter 'multi' to true.

>db.mycol.update({'title':'MongoDB Overview'},

{$set:{'title':'New MongoDB Tutorial'}},{multi:true})

8)

2) db.COLLECTION\_NAME.findOne() - that returns only one document.

3) To display the results in a formatted way, you can use **pretty()** method.

**Syntax**

>db.mycol.find().pretty()

## 4) RDBMS Where Clause Equivalents in MongoDB

To query the document on the basis of some condition, you can use following operations.

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

## AND in MongoDB

>db.mycol.find(

{

$and: [

{key1: value1}, {key2:value2}

]

}

).pretty()

### Example

Following example will show all the tutorials written by 'tutorials point' and whose title is 'MongoDB Overview'.

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

"\_id": ObjectId(7df78ad8902c),

"title": "MongoDB Overview",

"description": "MongoDB is no sql database",

"by": "tutorials point",

"url": "http://www.tutorialspoint.com",

"tags": ["mongodb", "database", "NoSQL"],

"likes": "100"

}

where clause will be **' where by = 'tutorials point' AND title = 'MongoDB Overview' '**. You can pass any number of key, value pairs in find clause.

## OR in MongoDB

### Syntax

To query documents based on the OR condition, you need to use **$or** keyword. Following is the basic syntax of **OR** −

>db.mycol.find(

{

$or: [

{key1: value1}, {key2:value2}

]

}

).pretty()

Ex; ..

## Using AND and OR Together

### Example

The following example will show the documents that have likes greater than 10 and whose title is either 'MongoDB Overview' or by is 'tutorials point'. Equivalent SQL where clause is **'where likes>10 AND (by = 'tutorials point' OR title = 'MongoDB Overview')'**

>db.mycol.find({"likes": {$gt:10}, $or: [{"by": "tutorials point"},

{"title": "MongoDB Overview"}]}).pretty()

{

"\_id": ObjectId(7df78ad8902c),

"title": "MongoDB Overview",

"description": "MongoDB is no sql database",

"by": "tutorials point",

"url": "http://www.tutorialspoint.com",

"tags": ["mongodb", "database", "NoSQL"],

"likes": "100"

}

---

some easy commands:

1)>db.COLLECTION\_NAME.find().limit(NUMBER)

Ex: >db.mycol.find({},{"title":1,\_id:0}).limit(2)

2)>db.COLLECTION\_NAME.find().limit(NUMBER).skip(NUMBER)

Ex: >db.mycol.find({},{"title":1,\_id:0}).limit(1).skip(1)

3) >db.COLLECTION\_NAME.find().sort({KEY:1})

Ex: >db.mycol.find({},{"title":1,\_id:0}).sort({"title":-1})

4) **aggregate()**

>db.COLLECTION\_NAME.aggregate(AGGREGATE\_OPERATION)

$sum , $group, $project,… see in tp

--

To create an index you need to use ensureIndex() method of MongoDB.

### Syntax

The basic syntax of **ensureIndex()** method is as follows().

>db.COLLECTION\_NAME.ensureIndex({KEY:1})

Here key is the name of the field on which you want to create index and 1 is for ascending order. To create index in descending order you need to use -1.

### Example

>db.mycol.ensureIndex({"title":1})

>

In **ensureIndex()** method you can pass multiple fields, to create index on multiple fields.

>db.mycol.ensureIndex({"title":1,"description":-1})

--

**a)use DATABASE\_NAME** is used to create database

>use mydb

switched to db mydb

b) To check your currently selected database, use the command **db**

>db

mydb

c) If you want to check your databases list, use the command **show dbs**.

>show dbs

d) Your created database (mydb) is not present in list. To display database, you need to insert at least one document into it.

>db.movie.insert({"name":"tutorials point"})

>show dbs

e) db.dropDatabase()

f) Basic syntax of **createCollection()** command is as follows −

db.createCollection(name, options)

ex: f1) >db.createCollection("mycollection")

f2) db.createCollection("mycol", { capped : true, autoIndexId : true, size : 6142800, max : 10000 } )

ff)In MongoDB, you don't need to create collection. MongoDB creates collection automatically, when you insert some document.

>db.tutorialspoint.insert({"name" : "tutorialspoint"})

>show collections

mycol

mycollection

system.indexes

tutorialspoint

g) db.COLLECTION\_NAME.drop()

h)

1)db.getCollection('employee').find({})

2)db.employee.findOne()

Please note **\_id** field is always displayed while executing **find()** method, if you don't want this field, then you need to set it as 0.

The basic syntax of **find()** method with projection is as follows −

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

Ex: >db.mycol.find({},{"title":1,\_id:0})

o/p

{"title":"MongoDB Overview"}

{"title":"NoSQL Overview"}

{"title":"Tutorials Point Overview"}

>

--

db.address.insert(

{id:"2", info:"hyd"})

--

db.employee.insert(

{id:"2",

name:"raj",

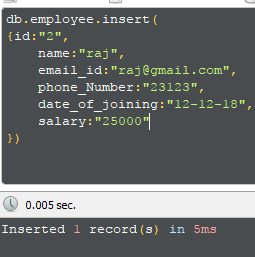
email\_id:"raj@gmail.com",

phone\_Number:"23123",

date\_of\_joining:"12-12-18",

salary:"25000"

})



db.employee.insertMany(

[{id:"3",

name:"ragu",

email\_id:"ragu@gmail.com",

phone\_Number:"23128",

date\_of\_joining:"11-12-18",

salary:"25000"

},

{id:"4",

name:"ram",

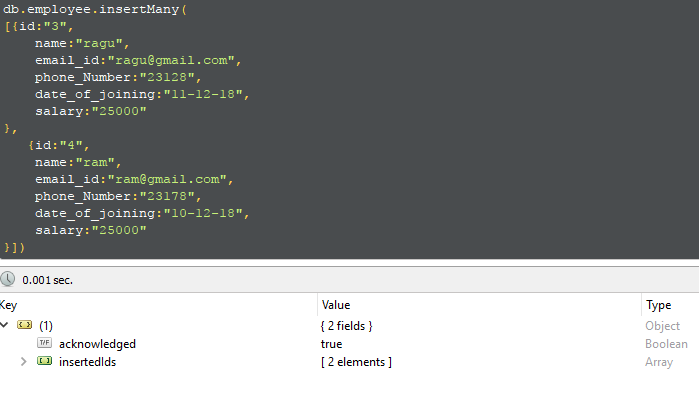
email\_id:"ram@gmail.com",

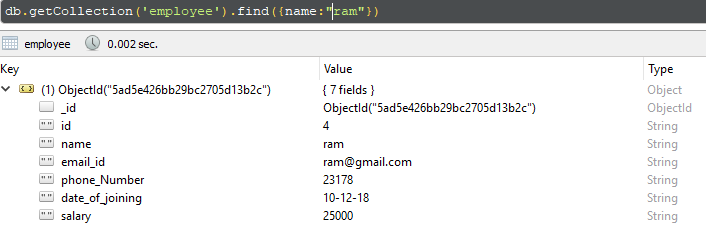
phone\_Number:"23178",

date\_of\_joining:"10-12-18",

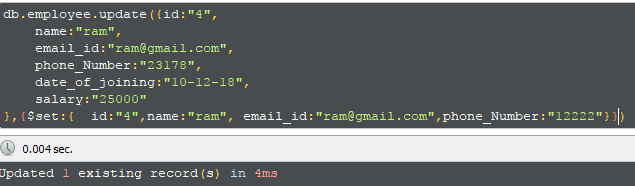
salary:"25000"

}])

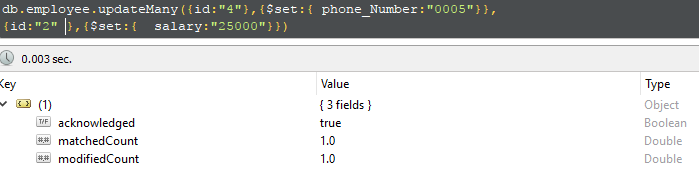


  
db.employee.updateMany({id:"4"},{$set:{ phone\_Number:"0005"}},

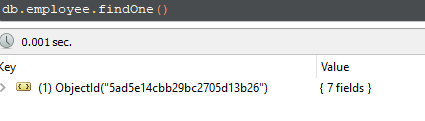
{id:"2" },{$set:{ salary:"25000"}})



Update Many doubt: Its updating only 1st value



Getting only 1st object



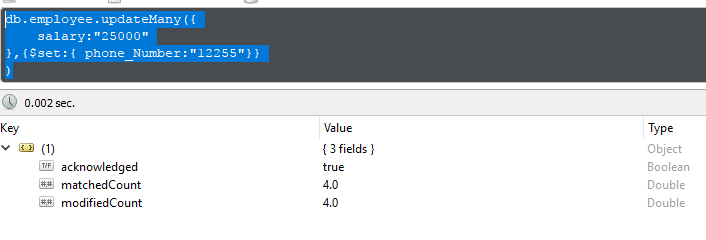
Update many got success:

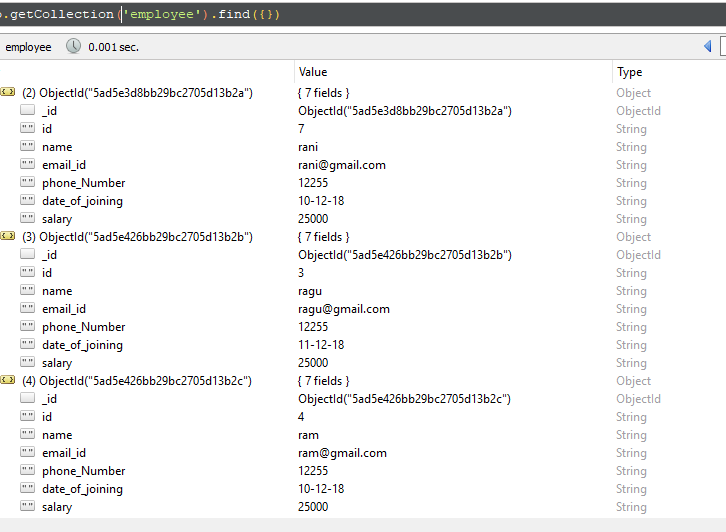
db.employee.updateMany({

salary:"25000"

},{$set:{ phone\_Number:"12255"}}

)

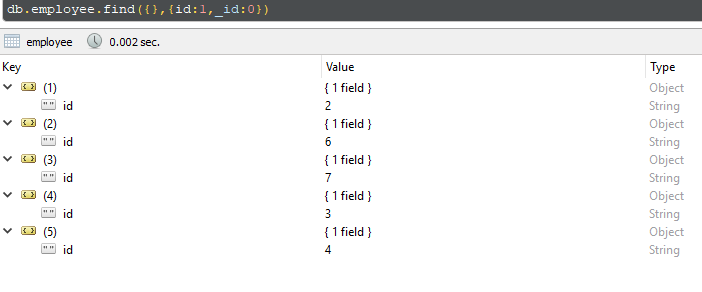




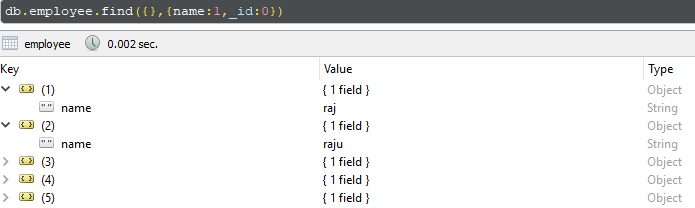
db.COLLECTION\_NAME.find({},{KEY:1})

project:

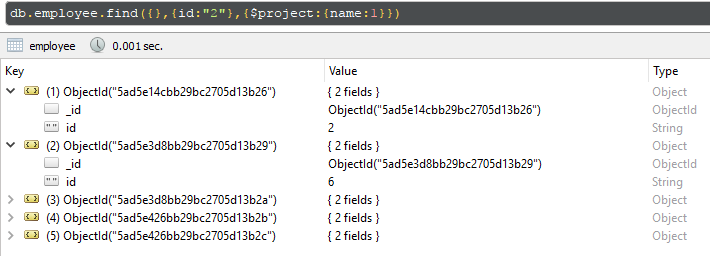
db.employee.find({},{id:1,\_id:0})



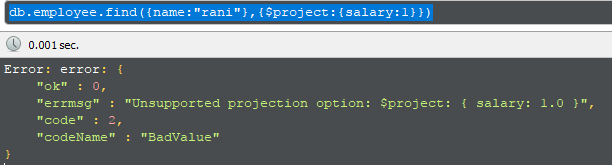
db.employee.find({},{name:1,\_id:0})



Project multidoc;

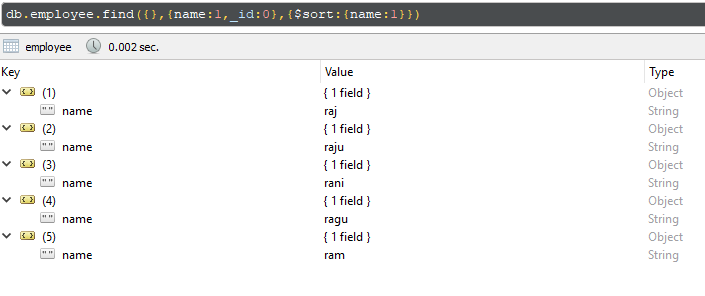


Project for single doc got error:



Sort: ( db.employee.find().sort({name:1}) -> it also works frm tp )

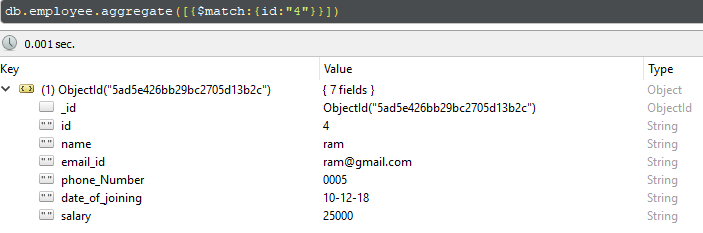
db.employee.find({},{name:1,\_id:0},{$sort:{name:1}})



--

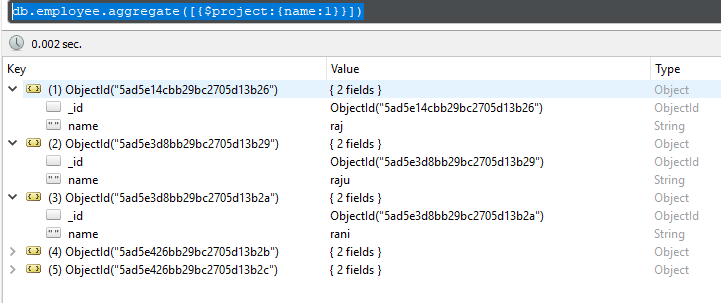
Aggragate: getting match record

db.employee.aggregate([{$match:{id:"4"}}])



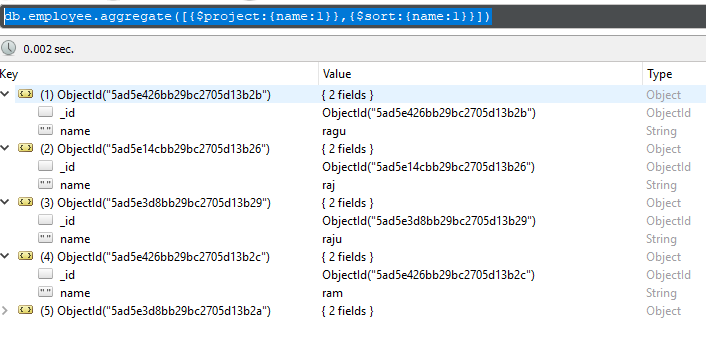
Project: using project multi doc retrieve

db.employee.aggregate([{$project:{name:1}}])



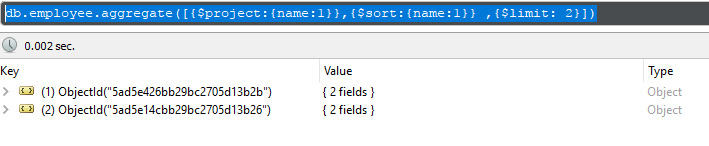
Sort using aggregation and project:

db.employee.aggregate([{$project:{name:1}},{$sort:{name:1}}])



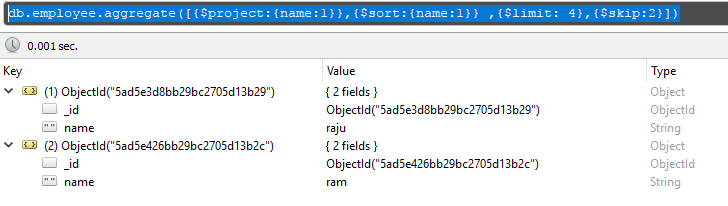
Limit:

db.employee.aggregate([{$project:{name:1}},{$sort:{name:1}} ,{$limit: 2}])



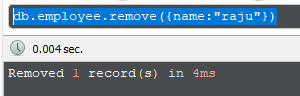
Limit and skip:

db.employee.aggregate([{$project:{name:1}},{$sort:{name:1}} ,{$limit: 4},{$skip:2}])



Remove;

db.employee.remove({name:"raju"})



Embed doc example:

db.employee.insert(

{id:"9",

name:"rajesh",

email\_id:"raje@gmail.com",

phone\_Number:"25555",

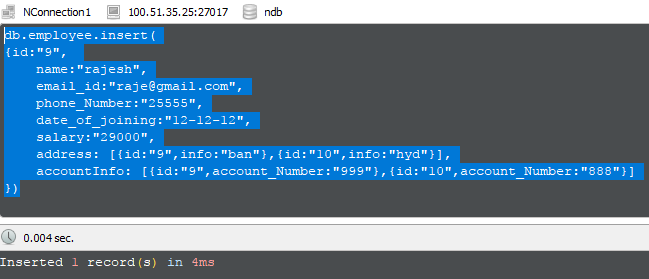
date\_of\_joining:"12-12-12",

salary:"29000",

address: [{id:"9",info:"ban"},{id:"10",info:"hyd"}],

accountInfo: [{id:"9",account\_Number:"999"},{id:"10",account\_Number:"888"}]

})



o/p:

