Update Documents

>db.collectionname.updateOne(<filter>,<update>,<options>);

>db.collectionname.updateMany(<filter>,<update>,<options>);

>db.collectionname.replaceOne(<filter>,<update>,<options>);

```
> db.emp.updateOne({"EID" : 1025},
                                {$set: {"ADDRESS" : "B302 PRAGYA
APARTMENTS, DWARKA, DELHI", "PHONE" : 9899245970} }
                        );
```

```
> db.salary.updateMany({$and: [{"DEPT" :"HR"},{"DESI" :
"ASSOCIATE"}]},{$set:{"DESI" : "SR. ASSOCIATE"}});
```

Update Documents

>db.collectionname.updateMany(<filter>,<update>,<options>);

```
> db.salary.updateMany(
                        {$and: [{"DEPT" :"HR"},
                                {"DESI" : "ASSOCIATE"}]},
                        {$set:{"DESI" : "SR. ASSOCIATE"}}
                        );
> db.stu2.updateMany(
                                {},
                                {"$set" : {"class" : "8th"}});
```

Update Documents

>db.collectionname.replaceOne(<filter>,<update>,<options>);

> db.stu.replaceOne(

{"rno" : 2},

{"rno" : 3 ,"name" : "ajay kumar",

"age" : 16}

);

Note: _id remains same the document has been replaced

- Create a database demo
- Create a EMP collection.
- Insert 8 documents containing in EMP collection containing eid, name, city, doj, dept, desi
- Display the documents in formatted manner.
- Show the documents containing “HR” dept.
- Show the documents for “OPS” managers.
- Promote all the associates as Sr.Associates

MongoDb CRUD Operations

CRUD operations create, read, update, and delete operations on documents

Create Operations – to create a new document

db.collection.insertOne()

db.collection.insertMany()

Read Operations – to retrieve the document form collection

db.collection.find()

db.collection.findOne()

MongoDb CRUD Operations

Update Operations – to modify existing documents in a collection.

db.collection.updateOne()

db.collection.updateMany()

db.collection.replaceOne()

Delete Operations – to remove documents from a collection.

db.collection.deleteOne()

db.collection.deleteMany()

Inserting Documents

db.collection.save() can also be used to insert document into a collection.

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.

```
> db.stu2.save({"rno" : 7, "name" : "inderjeet singh", "age" : 15});

> db.stu2.save({"_id" :
                        ObjectId("608fb168ea90e1f913d52714"),
                        "rno" : 8,
                        "name" : "Inder kumar singh",
                        "age" : 16}
              );
```

Logical Operator in MongoDB

To query documents based on the AND condition, we need to use $and keyword.

```
>db.collection.find({ $and: [ {<key1>:<value1>}, { <key2>:<value2>} ] })
```

```
> db.sal.find({
                $and:[
                        {"DEPT" :"HR"},
                        {"DESI" : "ASSOCIATE"}
                        ]
                });
```

Note: and is the default logical operator when working with multiple conditions.

```
.sal.find({"DEPT" :"HR","DESI" : "ASSOCIATE"});
```

Logical Operator in MongoDB

To query documents based on the OR condition, we need to use $or keyword.

```
>db.collection.find({ $or: [ {<key1>:<value1>}, { <key2>:<value2>} ] })
```

```
> db.sal.find(
                    {$or:[
                              {"DEPT" :"HR"},
                              {"DESI" : "ASSOCIATE"}
                              ]
                    });
```

Logical Operator in MongoDB

To query documents based on the NOT condition, we need to use $not keyword.

```
> db.salary.find(
                {"DEPT" :
                        {$not:
                                {$eq : "HR"}
                        }
                });
```

Logical Operator in MongoDB

NOR is the combination of NOT and OR, we need to use $nor keyword.

```
> db.salary.find({$nor:
                    [
                        { "DESI" : "ASSOCIATE"},
                        {"DEPT" : "HR"}
                    ]
                    });
```

Comparison Operator in MongoDb

$eq: Equality Operator same as where dept = ‘hr’ in SQL

Synatx : {<key>:{$eq;<value>}}

Eg1: > db.sal.find({"DEPT" :{$eq : "HR"}});

Eg2: > db.sal.find({"DEPT" : "HR"});

$ne: Not equal to Operator same as where dept <> ‘hr’ in SQL

Synatx : {<key>:{$ne;<value>}}

> db.sal.find({"DEPT" :{$ne : "HR"}});

Comparison Operator in MongoDb

$lt: less than Operator same as where salary< 100000 in SQL

Syntax: {<key>:{$lt:<value>}}

Eg: > db.sal.find({"SALARY" :{$lt : 50000}});

$lte: less than or equal to Operator same as where salary<= 50000 in SQL

Syntax: {<key>:{$lte:<value>}}

Eg: > db.sal.find({"SALARY" :{$lte : 50000}});

Comparison Operator in MongoDb

$gt: greater than or equal to Operator same as where salary>= 50000 in SQL

Syntax: {<key>:{$gt:<value>}}

Eg: > db.sal.find({"SALARY" :{$gt : 50000}});

$gte: greater than or equal to Operator same as where salary >= 50000 in SQL

Syntax: {<key>:{$gte:<value>}}

Eg: > db.sal.find({"SALARY" :{$gte : 50000}});

Comparison Operator in MongoDb

$in: represents values in an array same as where dept in (‘hr’, ‘it’, ’admin’)

Syntax: {<key>:{$in:[<value1>, <value2>,……<valueN>]}}

Eg: > db.sal.find({"DEPT" :
{$in :
["HR", "IT", "TEMP"]
}
});

$nin: represents values not in an array same as where NOT dept in (‘hr’, ‘it’, ’admin’)

Syntax: {<key>:{$nin:[<value1>, <value2>,……<valueN>]}}

Eg: > db.sal.find({"DEPT" :{$nin : ["HR", "IT", "TEMP“,”ADMIN”]}});

Operators Examples

> db.orders.find({"Category" : {"$eq" : "Technology"},"Sub-Category" : {"$eq" : "Phones"}}).count()

> db.salary.find({"$and" : [{"DEPT" : {"$eq" : "HR"}}, {"DESI" : {"$eq" : "ASSOCIATE"}}]})

> db.salary.find({"$and" : [{"DEPT" : "HR"}, {"DESI" : "ASSOCIATE"}]})

> db.salary.find({"$and" : [{"$or" : [{"DEPT" : "HR"}, {"DEPT" : "MIS"}]},{"$or" : [{"DESI" : "MANAGER"}, {"DESI" : "SR. ASSOCIATE"}]}]})

> db.salary.find({"SALARY" :{"$not" : {"$gt" : 100000}}})

Importing Data

For importing the data we need to download MongoDB Database Tools.

https://www.mongodb.com/try/download/database-tools

Download the Zip file

Unzip the downloaded file – Browse to the bin folder – Copy mongoimport application - paste it under the bin folder of MongoDB

C:\Program Files\MongoDB\Server\4.4\bin

Go to the command prompt and run mongoimport application

Importing Data

Flags used with mongoimport command.

-d: Specifies what database to use. We used the demo database.

-c: Specifies what collection to use. We used a sal collection.

--type: Specifies the type of file to import. json, csv, or tsv. We are using csv

--headerline: Specifies that the first row in our csv file should be the field names.

--drop: Specifies that we want to drop the collection before importing documents to avoid duplicate documents.

```
>mongoimport -d demo -c sal --type csv --file Salary.csv --headerline  --drop
```

```
>mongoimport -d demo2 -c sal --type csv --file C:\Users\Raj\Desktop\Salary.csv --headerline --drop
```