Name :- Rushikesh Karbhari Palve
Roll No. 31258

: Date :	Page No :
/ /20	

Assignment No. 9 [B1]

DOP :- 12-11-2021 DOS :- 30-11-2021

Title :- MongoDB Queries :

Problem Definition :-

Design and Develop MongoDB Queries using CRUD operations. (Use CRUD operations, SAVE method, logical operators).

Objectives :-

(i) Understand the concept of Binary JSON format.
(ii) Understand the concept of MongoDB document model -

Outcomes :-

After completion of assignment, students will be able to -
(i) Understand the concepts of Binary JSON format and MongoDB document model -
(ii) Design and Develop MongoDB queries using CRUD operations.

ROLL NO. 31258

Date: 1/20 | Page No:

Theory :-

MongoDB is a cross-platform, document oriented database that provides, high performance, high availability and easy scalability. MongoDB works on concept of collection and document.

Database :

Database is a physical container for collections. Each database gets it's own set of files on the file system. A single MongoDB server typically has multiple databases.

Collection :

collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

Document :

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure and common fields in a collection's document may hold different types of data.

The following table shows the relationship of RDBMS terminology with MongoDB.

ROLL NO. 31258

Date: / /20 | Page No.

RDBMS	MongoDB
Database	Database
Table	collection
Tuple / Row	Document
Column	Field
Table join	embeded Documents
primary key	primary key (Default key -id provided by mongodb)
Database server and client	
Mysqld / Oracle	mongod
mysql / sqlplus	mongo

CRUD is the basic operation of Mongodb, it stands CREATE, READ, UPDATE, DELETE.

MongoDB - 1. Create Collection -

The createCollection() Method

MongoDB db.createCollection(name, options) is used to create collection.

Basic syntax of createCollection() command is as follows:

```
db.createCollection(name, options)
```

In the command, name is the name of collection to be created. Options are a document and are used to specify configuration of collection.

Parameters	Type	Description
Name	string	Name of the collection to be created
Options	Document	(optional) specify options about memory size and indexing -

Options parameter is optional, so you need to specify only the name of the collection. Following is the list of options you can use:

Field	Type	Description
capped	Boolean	(Optional) if true, enables a capped collection. Capped collection is a fixed size collection that automatically overwrites its oldest entries when it reaches its maximum size. If you specify true, you need to specify size parameter also -
autoIndexID	Boolean	(Optional) if true, automatically create index on id field. Default value is false -
size	number	(Optional) specifies a maximum size in bytes for a capped collection. If capped is true, then you need to specify this field also.

Field	Type	Description.
max	number	(optional) Specifies the maximum number of documents allowed in the capped collection.

While inserting the document, MongoDB first checks size field of capped collection, then it checks max field.

Examples

Basic syntax of createCollection() method without options is as follows.

```
>use test
switched to db test
> db.createCollection("mycollection")
{"ok" : 1}
>
```

You can check the created collection using the command show collections.

```
> show collections
mycollection
system.indexes
```

2.] READ - The Find() Method

To query data from MongoDB collection, you need to use MongoDB's find() method.

Syntax

The basic syntax of find() method is as follows:

```
> db.COLLECTION_NAME.Find()
```

find() method will display all the documents in a non-structured way.

The pretty() Method

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

Syntax

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

Example

```
> db.mycol.find().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"
}
>
```

Apart from find() method, there is findOne() method, that returns only one document.

3.] UPDATE

. MongoDB's update() and save() methods are used to update document into a collection. The update() method updates the values in the existing document while the save() method replaces the existing document with the document passed in save() method.

ROLL NO. 31258 | Date: / /20 | Page No: | (7)

MongoDB update() method :-

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

The basic syntax of update() method is as follows:

```
>db.COLLECTION_NAME.update(SELECTION_CRITERIA,
                           UPDATED_DATA)
```

Example

Consider the mycol collection has the following data.

```
{"_id": ObjectId(5983548781331adf45ec5),
                    "title":"MongoDB Overview"}
{"_id": ObjectId(5983548781331adf45ec6),
                    "title":"NoSQL Overview"}
{"_id": ObjectId(5983548781331adf45ec7),
                    "title":"Tutorials Point Overview"}
```

Following example will set the new title 'New MongoDB Tutorial' of the documents whose title is 'MongoDB Overview'.

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

```
>db.mycol.find()

{"_id": ObjectId(5983548781331adf45ec5),
                    "title":"New MongoDB Tutorial"}
{"_id": ObjectId(5983548781331adf45ec6),
                    "title":"NoSQL Overview"}
{"_id": ObjectId(5983548781331adf45ec7),
                    "title":"Tutorials Point Overview"}
```

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})
```

MongoDB save() Method :

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

The basic syntax of MongoDB save() method is-

```
>db.COLLECTION_NAME.save({_id:ObjectId(), NEW_DATA})
```

Example

Following example will replace the document with the id '59835487813331adf45ec7'.

```
> db.mycol.save(
{
  "_id": ObjectId(59835487813331adf45ec7),
                  "title": "Tutorials Point New Topic",
  "by": "Tutorials Point"
})
```

4]. DELETE - The remove() Method :-

MongoDB's remove() method is used to remove a document from a collection.

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.

Basic syntax of remove() method is as follows:

```
>db.COLLECTION_NAME.remove(DELETION_CRITERIA)
```

Example :-

Consider the mycol collection has the following data :

```
{"_id":ObjectId(5983548781331adf45ec5),
    "title":"MongoDB Overview"}
{"_id":ObjectId(5983548781331adf45ec6),
    "title":"NoSQL Overview"}
{"_id":ObjectId(5983548781331adf45ec7),
    "title":"Tutorials Point Overview"}
```

Following example will remove all the documents whose title is 'MongoDB Overview'.

```
>db.mycol.remove({'title':'MongoDB Overview'})
>db.mycol.find()
```

```
{"_id":ObjectId(5983548781331adf45ec6),
    "title":"NoSQL Overview"}
{"_id":ObjectId(5983548781331adf45ec7),
    "title":"Tutorials Point Overview"}
```

LOGICAL OPERATORS :-

AND in MongoDB

Syntax :

In the find() method, if you pass multiple keys by separating them by ',' then MongoDB treats it as AND condition. Following is the basic syntax of AND -

```
>db.mycol.find({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({"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"
}
>
```

For the above given example, equivalent where clause will be 'where by = 'tutorials point' AND title='MongoDB Overview''. You can pass any number of key, value pairs in find clause -

ROLL NO. 31258

Date: 1/20 | Page No.

(11)

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()
```

Example will show all the tutorials written by 'tutorials point' or whose title is 'MongoDB Overview'.

```
>db.mycol.find({$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"
}
```

Using AND and OR Together Example :-

The following example will show the documents that have likes greater than 100 and whose title is either 'MongoDB Overview' or by is 'tutorials point'. Equivalent SQL where clause is

'where likes > 100 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"
}
```

Conclusion :-

Thus, We have studied MongoDB queries using CRUD operations.

OUTPUT :-

A.] CREATE :-

1.) Create Database assignment_no_9 and use assignment_no_9.

```
> use assignment_no_9

switched to db assignment_no_9
```

2.) Create Collection 'Employee'.

```
> db.createCollection("Employee")

{ "ok" : 1 }
```

B.] INSERT :-

3.) Insert One document in Employee collection. (use insertOne())

```
> db.Employee.insertOne({_id:1, name:{firstname:"Diane", lastname:"Murphy"},
email:"dmurphy@gmail.com", officeCode:1, jobTitle:"President", address:[{state:"Maharashtra"},
{city:"Pune"}]})

{ "acknowledged" : true, "insertedId" : 1 }
```

4.) Insert nine documents in Employee collection. (use insertMany())

```
> db.Employee.insertMany([

   {_id:2, name:{firstname:"Mary", lastname:"Patterson"}, email:"mpatterson@gmail.com",
officeCode:1, reportsTo:1, jobTitle:"VP Sales", address:[{state:"Goa"}, {city:"Panaji"}]},

   {_id:3, name:{firstname:"Jeff", lastname:"Firrelli"}, email:"jfirrelli@gmail.com",
officeCode:1, reportsTo:1, jobTitle:"VP Marketing", address:[{state:"Maharashtra"},
{city:"Ahmednagar"}]},

   {_id:4, name:{firstname:"William", lastname:"Patterson"}, email:"wpatterson@gmail.com",
officeCode:6, reportsTo:2, jobTitle:"Sales Manager", address:[{state:"Gujarat"},
{city:"Ahmadabad"}]},

   {_id:5, name:{firstname:"Gerard", lastname:"Bondur"}, email:"gbondur@gmail.com",
officeCode:4, reportsTo:2, jobTitle:"Sale Manager", address:[{state:"Haryana"},
{city:"Ambala"}]},

   {_id:6, name:{firstname:"Anthony", lastname:"Bow"}, email:"abow@gmail.com", officeCode:1,
reportsTo:2, jobTitle:"Sales Manager", address:[{state:"Kerala"}, {city:"Kochi"}]},

   {_id:7, name:{firstname:"Leslie", lastname:"Jennings"}, email:"ljennings@gmail.com",
officeCode:1, reportsTo:6, jobTitle:"Sales Rep", address:[{state:"Maharashtra"},
{city:"Mumbai"}]},

   {_id:8, name:{firstname:"Leslie", lastname:"Thompson"}, email:"lthompson@gmail.com",
officeCode:1, reportsTo:6, jobTitle:"Sales Rep", address:[{state:"Punjab"},
{city:"Amritsar"}]},

   {_id:9, name:{firstname:"Julie", lastname:"Firrelli"}, email:"jfirrelli@gmail.com",
officeCode:2, reportsTo:6, jobTitle:"Sales Rep", address:[{state:"Rajasthan"},
{city:"Jaipur"}]},
```

```
   {_id:10, name:{firstname:"Steve", lastname:"Patterson"}, email:"spatterson@gmail.com",
officeCode:2, reportsTo:6, jobTitle:"Sales Rep", address:[{state:"Uttar Pradesh"},
{city:"Agra"}]}

   ])

{
   "acknowledged" : true,
   "insertedIds" : [
      2,
      3,
      4,
      5,
      6,
      7,
      8,
      9,
      10
   ]
}
```

C.) READ :-

5.) Display all documents from 'Employee' collection. (use find())

```
> db.Employee.find().pretty()

{
   "_id" : 1,
   "name" : {
      "firstname" : "Diane",
      "lastname" : "Murphy"
   },
   "email" : "dmurphy@gmail.com",
   "officeCode" : 1,
   "jobTitle" : "President",
   "address" : [
      {
         "state" : "Maharashtra"
      },
      {
         "city" : "Pune"
      }
   ]
}
{
"_id" : 2,
"name" : {
   "firstname" : "Mary",
   "lastname" : "Patterson"
},
"email" : "mpatterson@gmail.com",
"officeCode" : 1,
"reportsTo" : 1,
"jobTitle" : "VP Sales",
"address" : [
   {
      "state" : "Goa"
   },
   {
      "city" : "Panaji"
```

```
      }
   ]
}
{
   "_id" : 3,
   "name" : {
      "firstname" : "Jeff",
      "lastname" : "Firrelli"
   },
   "email" : "jfirrelli@gmail.com",
   "officeCode" : 1,
   "reportsTo" : 1,
   "jobTitle" : "VP Marketing",
   "address" : [
      {
         "state" : "Maharashtra"
      },
      {
         "city" : "Ahmednagar"
      }
   ]
}
{
   "_id" : 4,
   "name" : {
      "firstname" : "William",
      "lastname" : "Patterson"
   },
   "email" : "wpatterson@gmail.com",
   "officeCode" : 6,
   "reportsTo" : 2,
   "jobTitle" : "Sales Manager",
   "address" : [
      {
         "state" : "Gujarat"
      },
      {
         "city" : "Ahmadabad"
      }
   ]
}
{
   "_id" : 5,
   "name" : {
      "firstname" : "Gerard",
      "lastname" : "Bondur"
   },
   "email" : "gbondur@gmail.com",
   "officeCode" : 4,
   "reportsTo" : 2,
   "jobTitle" : "Sale Manager",
   "address" : [
      {
         "state" : "Haryana"
      },
      {
         "city" : "Ambala"
      }
   ]
}
{
   "_id" : 6,
   "name" : {
      "firstname" : "Anthony",
```

```
      "lastname" : "Bow"
   },
   "email" : "abow@gmail.com",
   "officeCode" : 1,
   "reportsTo" : 2,
   "jobTitle" : "Sales Manager",
   "address" : [
      {
         "state" : "Kerala"
      },
      {
         "city" : "Kochi"
      }
   ]
}
{
   "_id" : 7,
   "name" : {
      "firstname" : "Leslie",
      "lastname" : "Jennings"
   },
   "email" : "ljennings@gmail.com",
   "officeCode" : 1,
   "reportsTo" : 6,
   "jobTitle" : "Sales Rep",
   "address" : [
      {
         "state" : "Maharashtra"
      },
      {
         "city" : "Mumbai"
      }
   ]
}
{
   "_id" : 8,
   "name" : {
      "firstname" : "Leslie",
      "lastname" : "Thompson"
   },
   "email" : "lthompson@gmail.com",
   "officeCode" : 1,
   "reportsTo" : 6,
   "jobTitle" : "Sales Rep",
   "address" : [
      {
         "state" : "Punjab"
      },
      {
         "city" : "Amritsar"
      }
   ]
}
{
   "_id" : 9,
   "name" : {
      "firstname" : "Julie",
      "lastname" : "Firrelli"
   },
   "email" : "jfirrelli@gmail.com",
   "officeCode" : 2,
   "reportsTo" : 6,
   "jobTitle" : "Sales Rep",
   "address" : [
```

```
        {
            "state" : "Rajasthan"
        },
        {
            "city" : "Jaipur"
        }
    ]
}
{
    "_id" : 10,
    "name" : {
        "firstname" : "Steve",
        "lastname" : "Patterson"
    },
    "email" : "spatterson@gmail.com",
    "officeCode" : 2,
    "reportsTo" : 6,
    "jobTitle" : "Sales Rep",
    "address" : [
        {
            "state" : "Uttar Pradesh"
        },
        {
            "city" : "Agra"
        }
    ]
}
```

D.] LOGICAL OPERATORS :-

6.) Find the documents where, officeCode is 6 and employees reports to VP Sales. (use and)

```
> db.Employee.find({$and:[{"officeCode" : 6}, {"reportsTo" : 2}]}).pretty()
{
    "_id" : 4,
    "name" : {
        "firstname" : "William",
        "lastname" : "Patterson"
    },
    "email" : "wpatterson@gmail.com",
    "officeCode" : 6,
    "reportsTo" : 2,
    "jobTitle" : "Sales Manager",
    "address" : [
        {
            "state" : "Gujarat"
        },
        {
            "city" : "Ahmadabad"
        }
    ]
}
```

7.) Find the documents with either firstname is 'Diane' or lastname is 'Jennings'. (use or)

```
> db.Employee.find({$or:[{"name.firstname":"Diane"}, {"name.lastname":"Jennings"}]}).pretty()

{
    "_id" : 1,
    "name" : {
        "firstname" : "Diane",
```

```
      "lastname" : "Murphy"
   },
   "email" : "dmurphy@gmail.com",
   "officeCode" : 1,
   "jobTitle" : "President",
   "address" : [
      {
         "state" : "Maharashtra"
      },
      {
         "city" : "Pune"
      }
   ]
}
{
"_id" : 7,
"name" : {
   "firstname" : "Leslie",
   "lastname" : "Jennings"
},
"email" : "ljennings@gmail.com",
"officeCode" : 1,
"reportsTo" : 6,
"jobTitle" : "Sales Rep",
"address" : [
   {
      "state" : "Maharashtra"
   },
   {
      "city" : "Mumbai"
   }
]
}
```


E.] UPDATE :-

8.) Find the document where firstname is 'Steve' and Update lastname to 'Thompson'. (use findOneAndUpdate())


```
> db.Employee.findOneAndUpdate({"name.firstname":"Steve"}, {$set:{"name.lastname":"Thompson"}})
{
   "_id" : 10,
   "name" : {
      "firstname" : "Steve",
      "lastname" : "Patterson"
   },
   "email" : "spatterson@gmail.com",
   "officeCode" : 2,
   "reportsTo" : 6,
   "jobTitle" : "Sales Rep",
   "address" : [
      {
         "state" : "Uttar Pradesh"
      },
      {
         "city" : "Agra"
      }
   ]
}

> db.Employee.find({"_id":10}).pretty()
```

```
{
   "_id" : 10,
   "name" : {
      "firstname" : "Steve",
      "lastname" : "Thompson"
   },
   "email" : "spatterson@gmail.com",
   "officeCode" : 2,
   "reportsTo" : 6,
   "jobTitle" : "Sales Rep",
   "address" : [
      {
         "state" : "Uttar Pradesh"
      },
      {
         "city" : "Agra"
      }
   ]
}
```


9.) Update the documents where officeCode is 2, change officeCode to 3. (use update() with {multi:true})

```
> db.Employee.update({"officeCode":2}, {$set:{"officeCode":3}}, {multi:true})

WriteResult({ "nMatched" : 2, "nUpserted" : 0, "nModified" : 2 })

> db.Employee.find({"officeCode":3}).pretty()
{
   "_id" : 9,
   "name" : {
      "firstname" : "Julie",
      "lastname" : "Firrelli"
   },
   "email" : "jfirrelli@gmail.com",
   "officeCode" : 3,
   "reportsTo" : 6,
   "jobTitle" : "Sales Rep",
   "address" : [
      {
         "state" : "Rajasthan"
      },
      {
         "city" : "Jaipur"
      }
   ]
}
{
"_id" : 10,
"name" : {
   "firstname" : "Steve",
   "lastname" : "Thompson"
},
"email" : "spatterson@gmail.com",
"officeCode" : 3,
"reportsTo" : 6,
"jobTitle" : "Sales Rep",
"address" : [
   {
      "state" : "Uttar Pradesh"
   },
   {
      "city" : "Agra"
```

```
        }
    ]
}
```

F.] save() Method :-

10.) Insert a document using save() method.

```
> db.Employee.save({"_id":11, name:{firstname:"George", lastname:"Vanauf"},
email:"gvanauf@gmail.com", officeCode:3, reportsTo:6, jobTitle:"Sales Rep",
address:[{state:"Assam"}, {city:"Dispur"}]})

WriteResult({ "nMatched" : 0, "nUpserted" : 1, "nModified" : 0, "_id" : 11 })
```

```
> db.Employee.find({"_id":11}).pretty()
{
    "_id" : 11,
    "name" : {
        "firstname" : "George",
        "lastname" : "Vanauf"
    },
    "email" : "gvanauf@gmail.com",
    "officeCode" : 3,
    "reportsTo" : 6,
    "jobTitle" : "Sales Rep",
    "address" : [
        {
            "state" : "Assam"
        },
        {
            "city" : "Dispur"
        }
    ]
}
```

11.) Update a document using save() method.

```
> db.Employee.save({"_id":11, name:{firstname:"Barry", lastname:"Jones"},
email:"bjones@gmail.com", officeCode:3, reportsTo:6, jobTitle:"Sales Rep",
address:[{state:"Assam"}, {city:"Dispur"}]})

WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

```
> db.Employee.find({"_id":11}).pretty()
{
    "_id" : 11,
    "name" : {
        "firstname" : "Barry",
        "lastname" : "Jones"
    },
    "email" : "bjones@gmail.com",
    "officeCode" : 3,
    "reportsTo" : 6,
    "jobTitle" : "Sales Rep",
    "address" : [
        {
            "state" : "Assam"
        },
```

```
        {
            "city" : "Dispur"
        }
    ]
}
```

12.) Use sort() and limit() to display the Data.

```
> db.Employee.find({}).sort({"name.firstname":1}).limit(3).pretty()
{
    "_id" : 6,
    "name" : {
        "firstname" : "Anthony",
        "lastname" : "Bow"
    },
    "email" : "abow@gmail.com",
    "officeCode" : 1,
    "reportsTo" : 2,
    "jobTitle" : "Sales Manager",
    "address" : [
        {
            "state" : "Kerala"
        },
        {
            "city" : "Kochi"
        }
    ]
}
{
    "_id" : 11,
    "name" : {
        "firstname" : "Barry",
        "lastname" : "Jones"
    },
    "email" : "bjones@gmail.com",
    "officeCode" : 3,
    "reportsTo" : 6,
    "jobTitle" : "Sales Rep",
    "address" : [
        {
            "state" : "Assam"
        },
        {
            "city" : "Dispur"
        }
    ]
}
{
    "_id" : 1,
    "name" : {
        "firstname" : "Diane",
        "lastname" : "Murphy"
    },
    "email" : "dmurphy@gmail.com",
    "officeCode" : 1,
    "jobTitle" : "President",
    "address" : [
        {
            "state" : "Maharashtra"
        },
        {
            "city" : "Pune"
```

```
        }
    ]
}
```

13.) Demonstrate unwind.

```
> db.Employee.aggregate([{$unwind:"$address"}]).pretty()
```

G.] DELETE :-

14.) Delete a document where id is 11. (use deleteOne())

```
> db.Employee.deleteOne({"_id":11})
{ "acknowledged" : true, "deletedCount" : 1 }
```

15.) Delete documents where Employee reports to President. (use deleteMany())

```
> db.Employee.deleteMany({"reportsTo":1})
{ "acknowledged" : true, "deletedCount" : 2 }
```

16.) Delete all documents. (use deleteMany())

```
> db.Employee.deleteMany({})
{ "acknowledged" : true, "deletedCount" : 8 }
```