MongoDB-CRUD-Session-I: Inserts, Load, Read

Prashanth B S[1]

[1]Department of Information Science & Engineering, Nitte Meenakshi Institute of Technology, Yelahanka - 560064, Bengaluru

April 16, 2022

1 Writing Scripts in MongoDB

To execute queries line by line might be tedious task in hand. Writing a script that executes a bunch of queries might be quintessential in certain situation. Though mongo does not support the scripts, it does provides us JS shell, where we can write JS scripts which contain mongoDB queries and execute them with a single command. To execute the ".js" file the MongoDB shell provides us *load*() function which takes in a valid *.js* file as an input and upon loading executes the queries inside them. Here is an example for the same.
It follows two steps, they are,

1. First we will define our JS file to write our script,

```
// Let us define query.js file as follows,
// File: test.js on the Desktop in Path: /home/username/Desktop/
var myvar = [ // storing the document under JS variable
    {
        _id: 1,
        "name": "Sumith",
        "salary": 20000,
        "designation" : "Software Developer",
        "SSN" : 1,
        "Email" : "sumith@gmail.com"
    },
    {
        _id: 2,
        "name": "Rajesh",
        "salary": 330000,
        "designation" : "Team Lead",
        "SSN" : 2,
        "Email" : "rajesh@gmail.com"
    }
] ;
db.test.insert(myvar) ; // inserting the document using insert command(mongo)
```

2. Second we create our datbase, collection and use the Mongo Load function to execute the JS script as follows,

```
> use testDB ;
> db.createCollection("test") ;
// To load full path of the file needs to be specified
```

```
>load("/home/username/Desktop/query.js")
true //output of the load function is boolean
```

2 CRUD-Inserts

MongoDB provides three variations for inserts. They are,

1. $insert(\{document\})$: inserts a document or documents into a collection and returns a $WriteResult$ object for single inserts and a $BulkWriteResult$ object for bulk inserts
2. $insertOne(\{document\})$: inserts a document into a collection and returns a document
3. $insertMany([\{document1\}, \{document2\}])$: inserts multiple documents into a collection and returns a document

   The difference between insert and insertMany lies in the way that the output and error handling is done. The insert command returns a document in both success and error cases. But the insertOne and insertMany commands throws exceptions[1].

Here are some examples for the above three methods, [2]

```
> use testDB;
>db.createCollection("test")
// insert method
db.test.insert([{_id: 1, "name": "Sumith"}, {_id: 2,"name": "Rajesh"}]);
// output
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
// ***************//
// for the same database, let us insert one more record using insertOne
>db.test.insertOne({_id:3, "name":"Sukshith"}) ;
{ "acknowledged" : true, "insertedId" : 3 }

// ***************//
// for the same database, let us insert two more record using insertMany
>db.test.insertMany([{_id: 4, "name": "Sagar"}, {_id: 5,"name": "Ramesh"}]);
{ "acknowledged" : true, "insertedIds" : [ 4, 5 ] }
```

3 CRUD-Selection/Read

In order to select or Read contents of a collection, Mongo provides method $find()$, this method is used for selecting documents based on certain criteria as well. By default $find()$ method displays all

[1] https://stackoverflow.com/questions/36792649/whats-the-difference-between\-insert-insertone-and-insertmany-method

[2] NOTE: Alternative to insert, $db.test.save(\{\})$ method can be used as well

the documents in the collection. There are two variations to the $find()$ method they are,

1. $db.col\_name.find()$: Displays all the documents in the collection
2. $db.col\_name.findOne()$: Displays the first document in the collection

Following are the example commands on the same database(testDB)

```
// find() method
> db.test.find()
{ "_id" : 1, "name" : "Sumith" }
{ "_id" : 2, "name" : "Rajesh" }
{ "_id" : 3, "name" : "Sukshith" }
{ "_id" : 4, "name" : "Sagar" }
{ "_id" : 5, "name" : "Ramesh" }
// findOne()
> db.test.findOne()
{ "_id" : 1, "name" : "Sumith" }
```

The real essence of selction commands comes into play when the $find()$ method is used in conjunction with the selection criteria to extract specific documents based on the user choice. The general syntax to write these selection criteria is as shown in figude 1,

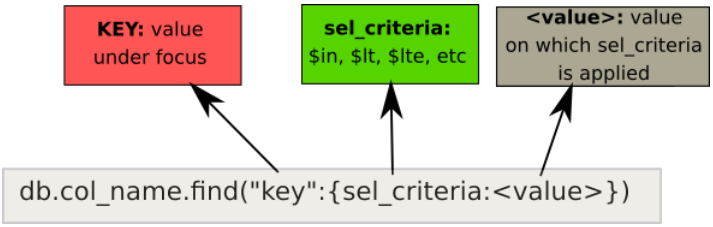

Figure 1: $find()$ with selection criteria syntax

The selection criteria can be specified using various inbuild keywords in mongoDB. Some of the commonly known selection criterias are ,

- $\$lt$: less than
- $\$lte$: less than equal to
- $\$gt$: greater than
- $\$gte$: greater than equal to
- $\$ne$: Not equals
- $\$in$: value in array
- $\$nin$: not in array
- $\$and$: cascading one or more selection criteria(True only when both criteria is true)
- $\$or$: cascading one or more selection criteria(True only when any one criteria is true)

Examples: For the above selection criteria examples, lets create a database called as "BookDB" with "Book" as collection and document with the feilds like book_name,author_namefirst_name, last_name, bid, tags (BSON array), Price as feilds.

```
// Creating the Database and collection and inserting a record
> use BookDB;
>db.createCollection("Book") ;
>db.Book.insert({"_id" : 1,"BookName" : "Lord of the Rings", "Author" : {"initials" :
    "J.R.R","name" : "Tolkien"}, "BID" : "AE001",
"tags" : ["Fiction","Action","Adventure"],"Price" : 100}) ;
// insert few more records this way

// To find the record with equals to criteria
> db.Book.find({"BookName":"Lord of the Rings"}); // equals
{ "_id" : 1, "BookName" : "Lord of the Rings", "Author" : { "initials" : "J.R.R", "name"
    : "Tolkien" }, "BID" : "AE001", "tags" : [ "Fiction", "Action", "Adventure" ],
    "Price" : 100 }

// To access a complex feild in the document say "Author" by "initials" = "J.R.R" we use
    dot operator
>db.Book.find({"Author.initials":"J.R.R"}) // equals
{ "_id" : 1, "BookName" : "Lord of the Rings", "Author" : { "initials" : "J.R.R", "name"
    : "Tolkien" }, "BID" : "AE001", "tags" : [ "Fiction", "Action", "Adventure" ],
    "Price" : 100 }
```

Some of the other examples are shown below, which involves the rest of the selection criterias, as follows,

```
// Finding out the books which are having Price> 180 , lets say 1 record is there
> db.Book.find({"Price":{$gt:180}}).pretty() // Exclusive of 180
{
        "_id" : 2,
        "BookName" : "Da-Vinci Code",
        "Author" : {
                "initials" : "Dan",
                "name" : "Brown"
        },
        "BID" : "AE002",
        "tags" : [
                "Fiction",
                "Action",
                "Adventure",
                "Scifi"
        ],
        "Price" : 200
}

// Finding out the books which are having Price< 150 (Exclusive, lets say 1 record is
    there
> db.Book.find({"Price":{$lt:150}}).pretty()
{
        "_id" : 1,
        "BookName" : "Lord of the Rings",
        "Author" : {
                "initials" : "J.R.R",
                "name" : "Tolkien"
        },
        "BID" : "AE001",
```

```
        "tags" : [
                "Fiction",
                "Action",
                "Adventure"
        ],
        "Price" : 100
}

// $in - listing out all the books whose tags matches with "Scifi"
> db.Book.find({"tags":{$in:["Scifi"]}}).pretty()
{ "_id" : 1, "BookName" : "Lord of the Rings", "Author" : { "initials" : "J.R.R", "name"
    : "Tolkien" }, "BID" : "AE001", "tags" : [ "Fiction", "Action", "Adventure","Scifi"
    ], "Price" : 100 }

// Similarly write and execute queries for $gte, $lte, $ne, $nin
```

Multiple selection criteria can be combined with the help of $*and* and $*or* keywords and they can also be combined to fetch a specific document as well. Some of the examples are as follows,

---

```
// Syntax: db.col_name.find({$and:[{criteria1},{criteria2}]})
// Find a book whose Price < 180 AND Author name = "Grisham"
>db.Book.find({$and:[{"Price":{$lt:180}},{"Author.name":"Grisham"}]})
{ "_id" : 3, "BookName" : "The Lawyer", "Author" : { "initials" : "John", "name" :
    "Grisham" }, "BID" : "AE003", "tags" : [ "Fiction", "Action", "Adventure" ], "Price"
    : 150 }

// Syntax: db.col_name.find({$or:[{criteria1},{criteria2}]})
// Find a book whose Price < 180 OR Author name = "Grisham"
>db.Book.find({$or:[{"Price":{$lt:180}},{"Author.name":"Grisham"}]})
{ "_id" : 1, "BookName" : "Lord of the Rings", "Author" : { "initials" : "J.R.R", "name"
    : "Tolkien" }, "BID" : "AE001", "tags" : [ "Fiction", "Action", "Adventure" ],
    "Price" : 100 }
{ "_id" : 3, "BookName" : "The Lawyer", "Author" : { "initials" : "John", "name" :
    "Grisham" }, "BID" : "AE003", "tags" : [ "Fiction", "Action", "Adventure" ], "Price"
    : 150 }

// $and and $or together - Nesting
// Find a book Whose Price>150 AND (Author name = "Brown" OR Author initials = "Dan" )
>
    db.Book.find({"Price":{$gt:150},$or:[{"Author.name":"Brown"},{"Author.initials":"Dan"}]})
{ "_id" : 2, "BookName" : "Da-Vinci Code", "Author" : { "initials" : "Dan", "name" :
    "Brown" }, "BID" : "AE002", "tags" : [ "Fiction", "Action", "Adventure", "Scifi" ],
    "Price" : 200 }
{ "_id" : 4, "BookName" : "Angels and Demons", "Author" : { "initials" : "Dan", "name" :
    "Brown" }, "BID" : "AE004", "tags" : [ "Fiction", "Action", "Adventure", "Scifi" ],
    "Price" : 180 }
```
