**Experiment No. 6**

**Title:** Perform crude operation of MongoDB.

**Batch:B1 RollNo.:1514033 ExperimentNo.:6**

### Aim: Perform crude operation of MongoDB.

### --------------------------------------------------------------------------------------------

**Resources needed:**Meteor

### 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.

### 1. The use Command

### In MongoDB use DATABASE\_NAME is used to create database. The command will create a new database if it doesn't exist, otherwise it will return the existing database.

### Syntax

use DATABASE\_NAME

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

>db

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

>show dbs

## 2. The dropDatabase() Method

MongoDB **db.dropDatabase()** command is used to drop a existing database.

### Syntax

db.dropDatabase()

This will delete the selected database.

## 3. The createCollection() Method

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

Syntax

db.createCollection(name, options)

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

|  |  |  |
| --- | --- | --- |
| **Parameter** | **Type** | **Description** |
| Name | String | Name of the collection to be created |
| Options | Document | (Optional) Specify options about memory size and indexing |

## 4. The drop() Method

MongoDB's **db.collection.drop()** is used to drop a collection from the database.

Syntax

db.COLLECTION\_NAME.drop()

## 5. The insert() Method

To insert data into MongoDB collection, you need to use MongoDB's **insert()** or **save()** method.

Syntax

>db.COLLECTION\_NAME.insert(document

## 6. The find() Method

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

Syntax

>db.COLLECTION\_NAME.find()

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

## 7. The pretty() Method

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

### Syntax

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

**8. findOne()**

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

**9. Update() Method**

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

Syntax

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

**10. Save() Method**

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

### Syntax

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

**11. The remove() Method**

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

### Syntax

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

## -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)

## -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()

### Results: (Screen shots of application development steps, program code and web browser displaying the specified message.

### Show dbs

meteor:PRIMARY> show dbs

local 0.063GB

### Use dbs

meteor:PRIMARY> use 1514033

switched to db 1514033

### Create collection “mycollection”

meteor:PRIMARY> db.createCollection("mycollection")

{ "ok" : 1 }

### Show collection

meteor:PRIMARY> show collections

Uddesh\_Kadu

mycollection

system.indexes

### Insert row

meteor:PRIMARY> db.Uddesh\_Kadu.insert({Name:'Uddesh Kadu',RollNo:33,Div:'B',gend

er:'M',Marks:85})

WriteResult({ "nInserted" : 1 })

### Find() with pretty()

meteor:PRIMARY> db.Uddesh\_Kadu.find().pretty()

{

"\_id" : ObjectId("5ab49a33f3a4916196985ee5"),

"Name" : "Uddesh Kadu",

"RollNo" : 33,

"Div" : "B",

"gender" : "M",

"Marks" : 85

}

{

"\_id" : ObjectId("5ab49a4ff3a4916196985ee6"),

"Name" : "Niket Kini",

"RollNo" : 25,

"Div" : "B",

"gender" : "M",

"Marks" : 75

}

{

"\_id" : ObjectId("5ab49a6cf3a4916196985ee7"),

"Name" : "Soham Hichkad",

"RollNo" : 21,

"Div" : "A",

"gender" : "M",

"Marks" : 90

}

{

"\_id" : ObjectId("5ab49a8df3a4916196985ee8"),

"Name" : "Prasad Gujar",

"RollNo" : 19,

"Div" : "B",

"gender" : "M",

"Marks" : 65

}

{

"\_id" : ObjectId("5ab49aabf3a4916196985ee9"),

"Name" : "Siddharth Sharma",

"RollNo" : 29,

"Div" : "A",

"gender" : "M",

"Marks" : 77

}

{

"\_id" : ObjectId("5ab49ad4f3a4916196985eea"),

"Name" : "Alekh Shah",

"RollNo" : 10,

"Div" : "B",

"gender" : "M",

"Marks" : 88

}

{

"\_id" : ObjectId("5ab49afdf3a4916196985eeb"),

"Name" : "Mansi Mota",

"RollNo" : 32,

"Div" : "A",

"gender" : "F",

"Marks" : 40

}

{

"\_id" : ObjectId("5ab49b18f3a4916196985eec"),

"Name" : "Hinal Panchal",

"RollNo" : 35,

"Div" : "A",

"gender" : "F",

"Marks" : 67

}

{

"\_id" : ObjectId("5ab49b2ef3a4916196985eed"),

"Name" : "Ashi Pandey",

"RollNo" : 36,

"Div" : "B",

"gender" : "F",

"Marks" : 72

}

### Greater than

meteor:PRIMARY> db.Uddesh\_Kadu.find({Marks:{$gt:75}}).pretty()

{

"\_id" : ObjectId("5ab49a33f3a4916196985ee5"),

"Name" : "Uddesh Kadu",

"RollNo" : 33,

"Div" : "B",

"gender" : "M",

"Marks" : 85

}

{

"\_id" : ObjectId("5ab49a6cf3a4916196985ee7"),

"Name" : "Soham Hichkad",

"RollNo" : 21,

"Div" : "A",

"gender" : "M",

"Marks" : 90

}

{

"\_id" : ObjectId("5ab49aabf3a4916196985ee9"),

"Name" : "Siddharth Sharma",

"RollNo" : 29,

"Div" : "A",

"gender" : "M",

"Marks" : 77

}

{

"\_id" : ObjectId("5ab49ad4f3a4916196985eea"),

"Name" : "Alekh Shah",

"RollNo" : 10,

"Div" : "B",

"gender" : "M",

"Marks" : 88

}

### And

meteor:PRIMARY> db.Uddesh\_Kadu.find({$and:[{Marks:{$gte:55}},{Div:'A'}]}).pretty

()

{

"\_id" : ObjectId("5ab49a6cf3a4916196985ee7"),

"Name" : "Soham Hichkad",

"RollNo" : 21,

"Div" : "A",

"gender" : "M",

"Marks" : 90

}

{

"\_id" : ObjectId("5ab49aabf3a4916196985ee9"),

"Name" : "Siddharth Sharma",

"RollNo" : 29,

"Div" : "A",

"gender" : "M",

"Marks" : 77

}

{

"\_id" : ObjectId("5ab49b18f3a4916196985eec"),

"Name" : "Hinal Panchal",

"RollNo" : 35,

"Div" : "A",

"gender" : "F",

"Marks" : 67

}

### Not

meteor:PRIMARY> db.Uddesh\_Kadu.find({gender:{$not:{$eq:'M'}}}).pretty()

{

"\_id" : ObjectId("5ab49afdf3a4916196985eeb"),

"Name" : "Mansi Mota",

"RollNo" : 32,

"Div" : "A",

"gender" : "F",

"Marks" : 40

}

{

"\_id" : ObjectId("5ab49b18f3a4916196985eec"),

"Name" : "Hinal Panchal",

"RollNo" : 35,

"Div" : "A",

"gender" : "F",

"Marks" : 67

}

{

"\_id" : ObjectId("5ab49b2ef3a4916196985eed"),

"Name" : "Ashi Pandey",

"RollNo" : 36,

"Div" : "B",

"gender" : "F",

"Marks" : 72

}

### Or

meteor:PRIMARY> db.Uddesh\_Kadu.find({$or:[{gender:{$ne:'M'}},{Marks:{$lte:65}}]}

).pretty()

{

"\_id" : ObjectId("5ab49a8df3a4916196985ee8"),

"Name" : "Prasad Gujar",

"RollNo" : 19,

"Div" : "B",

"gender" : "M",

"Marks" : 65

}

{

"\_id" : ObjectId("5ab49afdf3a4916196985eeb"),

"Name" : "Mansi Mota",

"RollNo" : 32,

"Div" : "A",

"gender" : "F",

"Marks" : 40

}

{

"\_id" : ObjectId("5ab49b18f3a4916196985eec"),

"Name" : "Hinal Panchal",

"RollNo" : 35,

"Div" : "A",

"gender" : "F",

"Marks" : 67

}

{

"\_id" : ObjectId("5ab49b2ef3a4916196985eed"),

"Name" : "Ashi Pandey",

"RollNo" : 36,

"Div" : "B",

"gender" : "F",

"Marks" : 72

}

### Descending in price

meteor:PRIMARY> db.Uddesh\_Kadu.find().pretty().sort({Marks:-1})

{

"\_id" : ObjectId("5ab49a8df3a4916196985ee8"),

"Name" : "Prasad Gujar",

"RollNo" : 19,

"Div" : "B",

"gender" : "M",

"Marks" : 99

}

{

"\_id" : ObjectId("5ab49a6cf3a4916196985ee7"),

"Name" : "Soham Hichkad",

"RollNo" : 21,

"Div" : "A",

"gender" : "M",

"Marks" : 90

}

{

"\_id" : ObjectId("5ab49ad4f3a4916196985eea"),

"Name" : "Alekh Shah",

"RollNo" : 10,

"Div" : "B",

"gender" : "M",

"Marks" : 88

}

{

"\_id" : ObjectId("5ab49a33f3a4916196985ee5"),

"Name" : "Uddesh Kadu",

"RollNo" : 33,

"Div" : "B",

"gender" : "M",

"Marks" : 85

}

{

"\_id" : ObjectId("5ab49a4ff3a4916196985ee6"),

"Name" : "Niket Kini",

"RollNo" : 25,

"Div" : "B",

"gender" : "M",

"Marks" : 75

}

{

"\_id" : ObjectId("5ab49b2ef3a4916196985eed"),

"Name" : "Ashi Pandey",

"RollNo" : 36,

"Div" : "B",

"gender" : "F",

"Marks" : 72

}

{

"\_id" : ObjectId("5ab49b18f3a4916196985eec"),

"Name" : "Hinal Panchal",

"RollNo" : 35,

"Div" : "A",

"gender" : "F",

"Marks" : 67

}

{

"\_id" : ObjectId("5ab49afdf3a4916196985eeb"),

"Name" : "Mansi Mota",

"RollNo" : 32,

"Div" : "A",

"gender" : "F",

"Marks" : 40

}

### Update query

meteor:PRIMARY> db.Uddesh\_Kadu.update({\_id:ObjectId("5ab49a8df3a4916196985ee8")}

,{$set:{Marks:99}})

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

meteor:PRIMARY> db.Uddesh\_Kadu.find({\_id:ObjectId("5ab49a8df3a4916196985ee8")}).

pretty()

{

"\_id" : ObjectId("5ab49a8df3a4916196985ee8"),

"Name" : "Prasad Gujar",

"RollNo" : 19,

"Div" : "B",

"gender" : "M",

"Marks" : 99

}

**Remove** meteor:PRIMARY> db.Uddesh\_Kadu.remove({Name:'Siddharth Sharma'},1)

WriteResult({ "nRemoved" : 1 })

meteor:PRIMARY> db.Uddesh\_Kadu.find().pretty()

{

"\_id" : ObjectId("5ab49a33f3a4916196985ee5"),

"Name" : "Uddesh Kadu",

"RollNo" : 33,

"Div" : "B",

"gender" : "M",

"Marks" : 85

}

{

"\_id" : ObjectId("5ab49a4ff3a4916196985ee6"),

"Name" : "Niket Kini",

"RollNo" : 25,

"Div" : "B",

"gender" : "M",

"Marks" : 75

}

{

"\_id" : ObjectId("5ab49a6cf3a4916196985ee7"),

"Name" : "Soham Hichkad",

"RollNo" : 21,

"Div" : "A",

"gender" : "M",

"Marks" : 90

}

{

"\_id" : ObjectId("5ab49a8df3a4916196985ee8"),

"Name" : "Prasad Gujar",

"RollNo" : 19,

"Div" : "B",

"gender" : "M",

"Marks" : 99

}

{

"\_id" : ObjectId("5ab49ad4f3a4916196985eea"),

"Name" : "Alekh Shah",

"RollNo" : 10,

"Div" : "B",

"gender" : "M",

"Marks" : 88

}

{

"\_id" : ObjectId("5ab49afdf3a4916196985eeb"),

"Name" : "Mansi Mota",

"RollNo" : 32,

"Div" : "A",

"gender" : "F",

"Marks" : 40

}

{

"\_id" : ObjectId("5ab49b18f3a4916196985eec"),

"Name" : "Hinal Panchal",

"RollNo" : 35,

"Div" : "A",

"gender" : "F",

"Marks" : 67

}

{

"\_id" : ObjectId("5ab49b2ef3a4916196985eed"),

"Name" : "Ashi Pandey",

"RollNo" : 36,

"Div" : "B",

"gender" : "F",

"Marks" : 72

}

Drop Collection

meteor:PRIMARY> show collections

Uddesh\_Kadu

mycollection

system.indexes

meteor:PRIMARY> db.mycollection.drop()

true

meteor:PRIMARY> show collections

Uddesh\_Kadu

system.indexes

### Drop databse

meteor:PRIMARY> db.dropDatabase()

{ "dropped" : "Uddesh\_Kadu", "ok" : 1 }

meteor:PRIMARY> show dbs

local 0.063GB

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Questions:**

1. **Explain features of MongoDB in detail.**

**Ans.**

**Full-Stack Solution:**

 - MEAN stack is just an instance of using a combination of technologies (MongoDB, Angular.js, Express.js and Node.js) and it cannot be called a framework. When using MEAN stack, developers are required to perform synchronization between Node.js and MongoDB as well as between Express.js and Angular.js.

**Development Ecosystem:**

 - Meteor.js is more than a JavaScript development framework. It is an open-source Isomorphic Development Ecosystem (IDevE). Meteor.js allows building real-time web applications from scratch. It contains all the necessary front-end and back-end components (such as frameworks, libraries, configuration tools, databases and more) that aid developers through the entire app development lifecycle, right from setup and development to deployment.

**Isomorphic JavaScript Code:**

 -Meteor allows using the same code on the front-end and the back-end as well as for mobile and web applications. It prevents developers from requiring to install and configure different libraries, module managers, APIs, drivers and more.

**Front-end Solution:**

 - Meteor offers Blaze.js, a front-end development framework. Though, it is not an advanced framework, it offers some good features. But, Meteor also integrates with popular modern front-end frameworks like Backbone.js in order to yield better results.

**Database Integration:**

 - One of the disadvantages of Meteor.js is that it supports only MongoDB as of now. So, you cannot use Meteor if you need to include NoSQL database support for your applications. But, the Meteor ecosystem offers MongoDB database as well as a front-end representation of MongoDB called Minimongo, which is written entirely in JavaScript. Meteor has a Mongo API which seamlessly integrates MongoDB on the back-end and Minimongo on the front-end. This result in faster page reloads and page updates thus mitigating latency.

**Live Reload:**

 -Another prominent feature of Meteor.js is its integrated live-browser reloading. Whenever there are any development changes to make on the front-end, it automatically reloads the live web page. Live reloading also allows refreshing only the required DOM elements on the web page without making an entire page to reload in spite of any dependent changes to data on the back-end or front-end.

**Custom Package Manager:**

 - One can use NPM modules (98,000+ modules) with Meteor but it has also got its own custom Package Manager which features the necessary NPM functionalities and some additional functionalities. Meteor’s official repository of Meteor packages named as atmosphere.js features more than 2,600 smart packages.

### Outcomes:

Understand the need, features, architecture and applications of frameworks

**Conclusion: (Conclusion to be based on the objectives and outcomesachieved)**.

Thus the CRUD operations were successfully performed in MONGO DB to study its functionality.

**Grade: AA / AB / BB / BC / CC / CD/DD**

**Signature of faculty in-charge withdate**

**References:**

1. By Isaac Strack; “Getting Started with Meteor.js JavaScript Framework”, 2nd Edition

;Packt Publishing, June 2015