Experiment No: 3

Aim: To execute MongoDB Commands.

About MongoDB:

MongoDB supports many datatypes. Some of them are −

- **String** − This is the most commonly used datatype to store the data. String in MongoDB must be UTF-8 valid.
- **Integer** − This type is used to store a numerical value. Integer can be 32 bit or 64 bit depending upon your server.
- **Boolean** − This type is used to store a boolean (true/ false) value.
- **Double** − This type is used to store floating point values.
- **Min/ Max keys** − This type is used to compare a value against the lowest and highest BSON elements.
- **Arrays** − This type is used to store arrays or list or multiple values into one key.
- **Timestamp** − ctimestamp. This can be handy for recording when a document has been modified or added.
- **Object** − This datatype is used for embedded documents.
- **Null** − This type is used to store a Null value.
- **Symbol** − This datatype is used identically to a string; however, it's generally reserved for languages that use a specific symbol type.
- **Date** − This datatype is used to store the current date or time in UNIX time format. You can specify your own date time by creating object of Date and passing day, month, year into it.
- **Object ID** − This datatype is used to store the document's ID.
- **Binary data** − This datatype is used to store binary data.
- **Code** − This datatype is used to store JavaScript code into the document.
- **Regular expression** − This datatype is used to store regular expression.

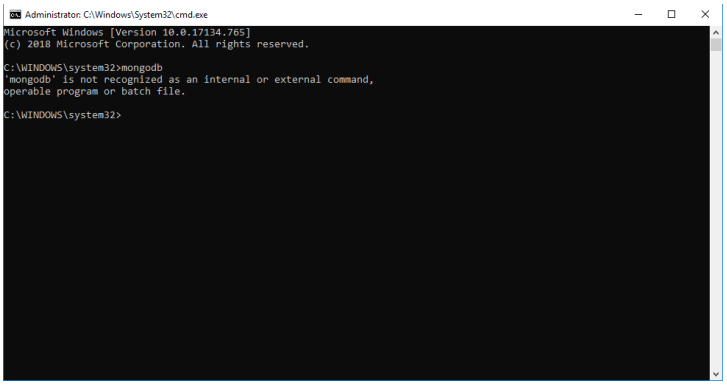
a) To install and configure MongoDB.

Steps:

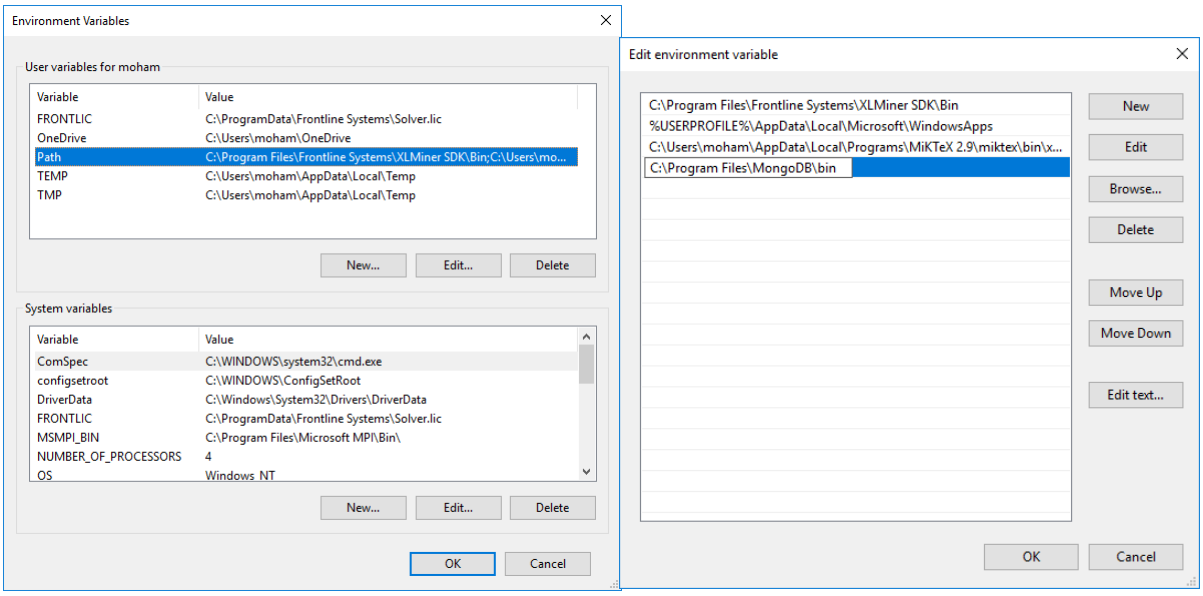
1. Install from https://www.mongodb.com/download-center/community.

2. Select OS and install it by following steps.

3. To configure MongoDB , add its path in environment variable advanced settings.



4. For example path C:\Program Files\MongoDB\bin; add to Path…Click on edit and add path.



5) Close Cmd and open again and type. Mongo to see following screen.

```
Administrator: C:\Windows\System32\cmd.exe - mongo
C:\WINDOWS\system32>mongo
MongoDB shell version v4.0.10
connecting to: mongodb://127.0.0.1:27017/?gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("15d2c02b-8a16-40c5-95d7-87c5de0a9197") }
MongoDB server version: 4.0.10
Welcome to the MongoDB shell.
For interactive help, type "help".
For more comprehensive documentation, see
        http://docs.mongodb.org/
Questions? Try the support group
        http://groups.google.com/group/mongodb-user
Server has startup warnings:
2019-05-31T08:37:52.698-0700 I CONTROL  [initandlisten]
2019-05-31T08:37:52.699-0700 I CONTROL  [initandlisten] ** WARNING: Access control is not enabled for the database.
2019-05-31T08:37:52.699-0700 I CONTROL  [initandlisten] **          Read and write access to data and configuration is u
nrestricted.
2019-05-31T08:37:52.699-0700 I CONTROL  [initandlisten]
---
Enable MongoDB's free cloud-based monitoring service, which will then receive and display
metrics about your deployment (disk utilization, CPU, operation statistics, etc).

The monitoring data will be available on a MongoDB website with a unique URL accessible to you
and anyone you share the URL with. MongoDB may use this information to make product
improvements and to suggest MongoDB products and deployment options to you.

To enable free monitoring, run the following command: db.enableFreeMonitoring()
To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---

>
```

6) Type Mongod to start of its server.

7) Type net start Mongodb.

```
Administrator: C:\Windows\System32\cmd.exe

Free Monitoring options:
  --enableFreeMonitoring arg            Enable Cloud Free Monitoring
                                        (on|runtime|off)
  --freeMonitoringTag arg               Cloud Free Monitoring Tags

WiredTiger options:
  --wiredTigerCacheSizeGB arg           maximum amount of memory to allocate
                                        for cache; defaults to 1/2 of physical
                                        RAM
  --wiredTigerJournalCompressor arg (=snappy)
                                        use a compressor for log records
                                        [none|snappy|zlib]
  --wiredTigerDirectoryForIndexes       Put indexes and data in different
                                        directories
  --wiredTigerCollectionBlockCompressor arg (=snappy)
                                        block compression algorithm for
                                        collection data [none|snappy|zlib]
  --wiredTigerIndexPrefixCompression arg (=1)
                                        use prefix compression on row-store
                                        leaf pages


C:\Program Files\MongoDB\bin>net start Mongodb
The requested service has already been started.

More help is available by typing NET HELPMSG 2182.


C:\Program Files\MongoDB\bin>
```

b) To execute NO SQL Commands.

Steps:

1. Type Mongo to start shell and type cls to clear things.

2. To show database:

Show dbs

3. To create new database:

Use project1

4. To check current database:

db

5. To create user and add roles to it:

```
> db.createUser(
...    {
...      user: "saiqa",
...      pwd: "1234",
...      roles: [ "readWrite", "dbAdmin" ]
...    }
... )
Successfully added user: { "user" : "saiqa", "roles" : [ "readWrite", "dbAdmin" ] }
```

6. To create and see collections (or tables in RDBMS)

db.createCollection('customers');

7. show collections

To insert documents in collections

db.customers.insert({first_name:"John",last_name:"Smith"})

8. To find collections

db.customers.find();

_id is object id.

9. To create more collections unstructured form:

```
> db.customers.insert([{first_name:"John",last_name:"Smith"},{first_name:"saiqa","last_name:"khan",gender:female)}

> db.customers.find();

{ "_id" : ObjectId("5cf171dd2fa371c9c11d116a"), "first_name" : "John", "last_name" : "Smith" }

{ "_id" : ObjectId("5cf173ad2fa371c9c11d116b"), "first_name" : "John", "last_name" : "Smith" }

{ "_id" : ObjectId("5cf173ad2fa371c9c11d116c"), "first_name" : "saiqa", "last_name" : "khan", "gender" : "female" }
```

10. To do formatting:

```
> db.customers.find().pretty();

{
    "_id" : ObjectId("5cf171dd2fa371c9c11d116a"),
    "first_name" : "John",
    "last_name" : "Smith"
}
{
    "_id" : ObjectId("5cf173ad2fa371c9c11d116b"),
    "first_name" : "John",
    "last_name" : "Smith"
}


{
    "_id" : ObjectId("5cf173ad2fa371c9c11d116c"),
    "first_name" : "saiqa",
    "last_name" : "khan",
    "gender" : "female"
```

11. Update Command

db.collection.update(query, update, options)

Modifies an existing document or documents in a collection. The method can modify specific fields of an existing document or documents or replace an existing document entirely, depending on the update parameter.

By default, the update() method updates a single document. Set the Multi Parameter to update all documents that match the query criteria.

The update() method has the following form:

```
db.collection.update(
  <query>,
  <update>,
  {
    upsert: <boolean>,
    multi: <boolean>,
    writeConcern: <document>,
    collation: <document>,
    arrayFilters: [ <filterdocument1>, ... ]
  }
)
```

The update() method takes the following parameters:

Parameter	Type	Description
query	document	The selection criteria for the update. The same query selectors as in the find() method are available. *Changed in version 3.0:* When you execute an update() with upsert: trueand the query matches no existing document, MongoDB will refuse to insert a new document if the query specifies conditions on the _id field using dot notation. For more information and an example, see upsert:true with a Dotted _id Query.
update	document	The modifications to apply. For details see Update Parameter.

Parameter	Type	Description
upsert	boolean	Optional. If set to true, creates a new document when no document matches the query criteria. The default value is false, which does *not* insert a new document when no match is found.

Save vs Update :

update modifies an existing document matched with your query params. If there is no such matching document, that's when upsert comes in picture.

- upsert : false : Nothing happens when no such document exist
- upsert : true : New doc gets created with contents equal to query params and update params

save : Doesn't allow any query-params. if _id exists and there is a matching doc with the same _id, it replaces it. When no _id specified/no matching document, it inserts the document as a new one.

Let us consider the two cases here for save :-

1) Having _id in doc.

2) Not having _id in doc.

```
                Save ()
               /     \
              /       \

          Having _id     Not Having _id

->In this case save will do    ->  It will do normal insertion
  upsert to insert.Now            in this case as insert() do.
  what that means, it means
  take the document and replace
  the complete document having same
  _id.
```

Let us consider the two cases here for insert:-

1) Having _id of doc in collection.

2) Not having _id of doc in collection.

```
                Insert()
               /        \
              /          \

 Doc Having _id in collection    Doc Not Having _id
->  E11000 duplicate key     ->Insert a new doc inside the collection.
    error index:
```

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})
```

12. To remove collections:

Db.customers.remove({first_name:”saiqa”});

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

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(
  {
    $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"

}
```

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.

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

Following 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 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"
}
>
```