Assignment No. 10

Title of Assignment: Basic operations **in MongoDB**

Problem Definition: Implement database with suitable example using MongoDB and implement all basic operations and administration commands using two tier architecture.

1.1 Pre-requisite: Basics of **MongoDB**

You should have a basic understanding of database, text editor and execution of programs etc. You must have understanding on basic concepts of Database (RDBMS).

1.2 Learning Objective:

To learn all type mongoDB commands and their uses

1.3 Relevant Theory / Literature Survey:

1.3.1 Introduction to MongoDB

- 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 its 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 documents may hold different types of data.
- Below given table shows the relationship of RDBMS terminology with MongoDB

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by mongodb itself)
Database Server and Client	
Mysqld/Oracle	mongod
mysql/sqlplus	mongo

Advantages of MongoDB over RDBMS

- Schema less : MongoDB is document database in which one collection holds different different documents.
- Number of fields, content and size of the document can be differ from one document to another.
- Structure of a single object is clear
- No complex joins
- Deep query-ability. MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL
- Tuning
- Ease of scale-out: MongoDB is easy to scale
- Conversion / mapping of application objects to database objects not needed
- Uses internal memory for storing the (windowed) working set, enabling faster access of data.

Why should use MongoDB

- Document Oriented Storage : Data is stored in the form of JSON style documents
- Index on any attribute
- Replication & High Availability
- Auto-Sharding
- Rich Queries
- Fast In-Place Updates
- Professional Support By MongoDB

Where should use MongoDB?

- Big Data
- Content Management and Delivery
- Mobile and Social Infrastructure
- User Data Management
- Data Hub

[I] Create Database

The use Command

- 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

- **Example:**

  If you want to create a database with name **<mydb>**, then **use DATABASE** statement would be as follows:

  >use mydb
  switched to db mydb

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

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

  Your created database (mydb) is not present in list. To display database you need to insert atleast one document into it.

  >db.movie.insert({"name":"tutorials point"})
  >show dbs
  local 0.78125GB
  mydb 0.23012GB
  test 0.23012GB

[II] Drop Database

The dropDatabase () Method

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

Syntax:

db.dropDatabase()

This will delete the selected database. If you have not selected any database, then it will delete default 'test' database.

Example:

First, check the list available databases by using the command **show dbs**

If you want to delete new database **<mydb>**, then **dropDatabase()** command would be as follows:

>use mydb
switched to db mydb

>db.dropDatabase()
>{ "dropped" : "mydb", "ok" : 1 }

Now check list of databases
>show dbs

[III] Create Collection

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 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

Options parameter is optional, so you need to specify only name of the collection.

Examples:

```
>db.createCollection("mycollection")
{ "ok" : 1 }
>
```
You can check the created collection by using the command **show collections**

In mongodb you don't need to create collection. MongoDB creates collection automatically, when you insert some document.

e.g >db.tutorialspoint.insert({"name" : "tutorialspoint"})

[IV] Drop Collection

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

Syntax:
db.COLLECTION_NAME.drop()

Example:

```
>db.mycollection.drop()
true
>
```
Again check the list of collections into database

e.g >show collections

drop() method will return true, if the selected collection is dropped successfully otherwise it will return false

[V] Insert Document

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)

- If the collection doesn't exist in the database, then MongoDB will create this collection and then insert document into it.
- In the inserted document if we don't specify the _id parameter, then MongoDB assigns an unique ObjectId for this document.
- _id is 12 bytes hexadecimal number unique for every document in a collection. 12 bytes are divided as follows:
- _id: ObjectId(4 bytes timestamp, 3 bytes machine id, 2 bytes process id, 3 bytes incrementer)
- To insert multiple documents in single query, you can pass an array of documents in insert() command.

- To insert the document you can use **db.post.save(document)** also.

- 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.

[VI] MongoDB - Query Document

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.

The pretty() Method

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

Syntax :

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

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

[VII] 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

[VIII] AND in MongoDB

In the **find()** method if you pass multiple keys by separating them by ',' then MongoDB treats it **AND** condition.

Syntax :

>db.mycol.find({key1:value1, key2:value2}).pretty()

[IX] OR in MongoDB

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

Syntax :

>db.mycol.find({$or: [{key1: value1}, {key2:value2}]}).pretty()

[X] Using AND and OR together

Example

Below given 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>10 AND (by = 'tutorials point' OR title= 'MongoDB Overview')'**

```
>db.mycol.find("likes": {$gt:10}, $or: [{"by": "tutorials point"}, {"title": "MongoDB Overview"}] }).pretty()
```

[XI] MongoDB Update Document

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

MongoDB Update() method

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

Syntax:

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

- 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'}})
```

- By default mongodb will update only single document, to update multiple you need to set a paramter 'multi' to true.

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

[XII] MongoDB Save() Method

The **save()** method replaces the existing document with the new document passed in save() method
Syntax
```
>db.COLLECTION_NAME.save({_id:ObjectId(),NEW_DATA})
```

[XIII] MongoDB Delete Document

The remove() Method

- MongoDB's **remove()** method is used to remove document from the collection.
- remove() method accepts two parameters. One is deletion criteria and second is justOne flag
  1. **deletion criteria :** (Optional) deletion criteria according to documents will be removed.
  2. **justOne :** (Optional) if set to true or 1, then remove only one document.

Syntax:

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

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

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

Remove only one

If there are multiple records and you want to delete only 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()

[XIV] MongoDB Projection

- In mongodb projection meaning is selecting only necessary data rather than selecting whole of the data of a document.
- If a document has 5 fields and you need to show only 3, then select only 3 fields from them.

The find() Method

- In MongoDB when you execute **find()** method, then it displays all fields of a document.
- To limit this you need to set list of fields with value 1 or 0.
- 1 is used to show the filed while 0 is used to hide the field.

Syntax:

>db.COLLECTION_NAME.find({},{KEY:1})

Following example will display the title of the document while quering the document.

>db.mycol.find({},{"title":1,_id:0})

Please note : _id field is always displayed while executing **find()** method, if you don't want this field, then you need to set it as 0.

[XV] MongoDB Limit Records

The Limit() Method

To limit the records in MongoDB, you need to use **limit()** method. **limit()** method accepts one number type argument, which is number of documents that you want to displayed.

Syntax:

>db.COLLECTION_NAME.find().limit(NUMBER)

Following example will display only 2 documents while quering the document.

>db.mycol.find({},{"title":1,_id:0}).limit(2)

If you don't specify number argument in **limit()** method then it will display all documents from the collection.

[XVI] MongoDB Skip() Method

Apart from limit() method there is one more method **skip()** which also accepts number type argument and used to skip number of documents.

Syntax:
>db.COLLECTION_NAME.find().limit(NUMBER).skip(NUMBER)

Example:

Following example will only display only second document.
>db.mycol.find({},{"title":1,_id:0}).limit(1).skip(1)

Please note default value in **skip()** method is 0

[XVII] MongoDB Sort Documents

The sort() Method

- To sort documents in MongoDB, you need to use **sort()** method.
- **sort()** method accepts a document containing list of fields along with their sorting order.
- To specify sorting order 1 and -1 are used. 1 is used for ascending order while -1 is used for descending order.

Syntax:

>db.COLLECTION_NAME.find().sort({KEY:1})

Example

Following example will display the documents sorted by title in descending order.

>db.mycol.find({},{"title":1,_id:0}).sort({"title":-1})

Please note if you don't specify the sorting preference, then **sort()** method will display documents in ascending order.

Conclusion

We have studied all the basic commands of mongoDB and also learnt how to make use of them.