Page No.
Date

Assignment - B1

* Date of Completion: 19-10-2020
* Date of Submission: 26-10-2020
* Title: Study of Open Source NoSQL Database: MongoDB
* Problem Statement: Implement database with suitable example using MongoDB and implement Study of Open Source NoSQL Database: MongoDB (Installation, Basic CRUD Operation, Execution)
* Learning Objectives:
  - Understand the concept of NoSQL Database
  - Understand concept of MongoDB with CRUD operation
  - Understand the basic installation and administrative commands of MongoDB.
* Learning Outcomes:
  - Perform installation and setup of MongoDB
  - Implementation of database in MongoDB
  - Perform basic CRUD operations and administration commands in MongoDB.
* Software & Hardware Requirements:
  MongoDB 4.4, Windows 10 (64 bit), Intel i5 processor, I/O Devices.

Teacher's Sign.: _____

Page No.
Date

Theory :-

MongoDB:

MongoDB is a cross platform, document oriented database that provides, high performance, high availability and easily scalability. it is a NoSQL data base works on concept of collection and document. A single MongoDB server typically has multiple documents databases.

Collection:

It is a group of MongoDB documents. It is equivalent of an RDBMS table. A collection exists within a single database. It donot 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 pair. Documents have dynamic schema ie, that documents in the same collection donot need to have the same as set of fields or structure. Every document is uniquely identified by a key '_id' which is a 12 byte hexadecimal number by default if not explicitly provided by user. These 12 bytes consists of first 4 bytes of current time stamp, next 3 bytes of machine id next 2 bytes of process id and 3 bytes of are simple incremental value.

Teacher's Sign.: _____

Page No.
Date

Advantages of MongoDB over RDBMS

i) Schema less
ii) Structure of a single object is well defined
iii) No complex joins
iv) Ease of scaling
v) Conversion of/a objects into relational models is not required
vi) Deep query ability

- CRUD Operations

i) Create:

db.collection.insert()

Syntax: db.collection.insert (<doc>)

or

db.collection.insert ([<doc1>, <doc2>, ...])

eg:

```
db.collection.insert ([
    { name: "Varun", Age: 20 },
    { name: "Tejas", }
]).
```

- insertOne()

syntax:

db.collection.insertOne (<document>)

- insertMany()

syntax: db.collection.insertMany ([<doc1>, <doc2>, ...])

Teacher's Sign.: _____

ii) Retrieve:

- find()

Syntax: db.collection.find(<query Document>)

Selects document in a collection or view and returns a cursor to document that match the query criteria

eg.

db.movie.find() // returns all documents in collection movie

db.movie.find({"_id":6}) // return document with _id=6.

- findone():

Returns one document that satisfies the specified criteria on document or view. If multiple documents satisfy, this method returns first document

Syntax: db.collection.findone(<querydocument>)

iii) Update:

update() modifies an existing document as or documents in collections

Syntax: db.collection.update(

<query>, <update>, {upsert:<boolean>, multi:<boolean>

})

Upsert:

when upsert is set to true.

- if document(s) match the query performs an update
- if document matches the query doesn't not exist update() inserts a single document

eg: db.movie.update({movies.rating:"good",

{"_id":10, name:"varun", movies.rating:"good"}

{upsert:true}

)

Teacher's Sign.:

- UpdateOne() : updates single document within the collection based on the filter.

Syntax:

```
db.collection.UpdateOne( <filter>, <update>,
                         { upsert: <true/false> }
                       )
```

- UpdateMany() : updates all documents that match the specified filter for a collection.

Syntax:

```
db.collection.UpdateMany(
  { <filter>, <update>, { upsert: <true/false> }
```

- Delete:
  - remove() : removes the document from a collection.

Syntax: db.collection.remove(<query>, <justone>)

eg:

```
db.movie.remove( { age: { $lt: 25 } } )
db.remove( { } ) // removes all documents
```

- Query operators in MongoDB:
  - Comparisons:

$eq: matches the values that are equal to a specified value
syntax { <field>: { $eq: <value> } }

$lt: matches value that are less than a specified value
syntax: { <field>: { $lt: <value> } }

$gt: matches value that are greater than a specified value
syntax: { <field>: { $gt: <value> } }

Teacher's Sign.: _____

$in : matches any of the values specified in an array
syntax: {<field>:{$in: [<value1>, <value2>, ...]}}

- logical :
  $and : joins the query expressions with logical AND
  Syntax: { $and : [{<exp1>}, {<exp2>}, ... {<exp n>}] }

  $or: joins the query expressions with logical OR
  syntax: { $or: [{<exp1>}, ... {<exp n>}] }

  $not: inverts the effect of query expression
  syntax: { $not: { < operator-expression > } }

- db.adminCommand ( ): Administration commands
- CreateUser:

  following example uses data db.adminCommand() method to create a user named "Alice" with the 'dbOwner' role on the 'adminidatabase'

```
eg: db.adminCommand ( { createUser: "Alice",
        pwd: "Alice123" passwordPrompt(),
        roles: [{role: "dbOwner", db: "admin"}].
})
```

- DropUser:
  db.dropUser ("username")

- create/switch Database
  use < database-name>

Teacher's Sign.: _____

- Drop database:
  use <database_name>
  db.dropDatabase();

- create collection():
  db.createcollection("collection_name")

- Drop collection():
  db.<collection_name>.drop()

- show commands // server level
  show dbs;
  show users;
  show roles;

  use <database_name>;
  show collections;

- Conclusion:
  We completed the study of MongoDB database along with installation of MongoDB database version 4.4.1 and implemented basic CRUD operations on a sample database and collection.

Teacher's Sign.: _____