Student Name:Siddhesh Shirke

Class: Msc. CS SEM I

Roll No: 30

ADT Research Work Topic :

Demonstration of noSQL database- MongoDB

Title: MongoDB

Theory:

- MongoDB is document database designed for ease and scaling
- available as community and enterprise edition.
- However the community edition itself is very powerful.

Key Features:

1. High Performance:

   MongoDB provides high performance data persistence. In particular,

   - Support for embedded data models reduces I/O activity on database system.
   - Indexes support faster queries and can include keys from embedded documents and arrays.

2. Rich Query Language:
   MongoDB supports a rich query language to support read and write operations (CRUD)

3. Horizontal Scalability:
   MongoDB provides horizontal scalability as part of its *core* functionality

Terms used for MongoDB vs terms used for SQL/ relational databases:

SQL Terms	MongoDB terms
Database	Database
tables	collections
rows	documents (BSON)
columns	fields

Steps:

Commands:

- show dbs : shows list of available databases
- use databasename : Create a new or switch databases
- db : View current Database
- db.dropDatabase() : Delete Database

- show collections : show collections
- db.createCollection('comments') : Create a collection named 'comments'
- db.comments.drop() : Drop a collection named 'comments'

- db.comments.find() : Show all Rows in a Collection
- db.comments.find().pretty() : Show all Rows in a Collection (Prettified)
- db.comments.findOne({name: 'Harry'}) : Find the first row matching the object

- Insert One Row:

```
db.comments.insert({
    'name': 'Harry',
    'lang': 'JavaScript',
    'member_since': 5
 })
```

- Insert many Rows:

```
db.comments.insertMany([{
   'name': 'Harry',
   'lang': 'JavaScript',
   'member_since': 5
   },
   {'name': 'Rohan',
   'lang': 'Python',
   'member_since': 3
   },
   {'name': 'Lovish',
   'lang': 'Java',
   'member_since': 4
}])
```

- `db.comments.find({lang:'Python'})` : Search in a MongoDb Database
- `db.comments.find().limit(2)` : Limit the number of rows in output
- `db.comments.find().count()`: Count the number of rows in the output
- Update a row;

```
db.comments.update({name: 'Shubham'},
{'name': 'Harry',
   'lang': 'JavaScript',
   'member_since': 51
}, {upsert: true})
```

- Mongodb Increment Operator:

```
db.comments.update({name: 'Rohan'},
{$inc:{
   member_since: 2
}})
```

- Mongodb Rename Operator:

```
db.comments.update({name: 'Rohan'},
{$rename:{
   member_since: 'member'
}})
```

- `db.comments.remove({name: 'Harry'})`: Delete Row

- Less than/Greater than/ Less than or Eq/Greater than or Eq:

```
db.comments.find({member_since: {$lt: 90}})
db.comments.find({member_since: {$lte: 90}})
db.comments.find({member_since: {$gt: 90}})
db.comments.find({member_since: {$gte: 90}})
```

Screenshot/ Output:

1.
```
> show dbs
< CWHBlog   65.5 kB
  Project   73.7 kB
  achme     98.3 kB
  admin     65.5 kB
  config    73.7 kB
  local     73.7 kB
```

2.
```
> use Project
< 'switched to db Project'
Project >
```

3.
```
> show collections
< student
```

4. Creating collection ‘Students’

```
> db.createCollection('Students')
< { ok: 1 }
```

5. Inserting data into collection:

```
> db.Students.insertOne({'name':'Siddhesh', "rollno":30,'course':'CS'})
< { acknowledged: true,
    insertedId: ObjectId("61911dedd3c2f0dcd8ed05fb") }
```

6. Retrieving all data from collection:

```
> db.Students.find()
< { _id: ObjectId("61911dedd3c2f0dcd8ed05fb"),
    name: 'Siddhesh',
    rollno: 30,
    course: 'CS' }
```

7. Inserting multiple documents (record) at a time:

```
> db.Students.insertMany([{'name':'Harry','rollno':1, 'course':'CS', 'mobno':'9930012834'},{'name':'Summer','rollno':2,'course':'CS'}])
< { acknowledged: true,
  insertedIds:
   { '0': ObjectId("61911f92d3c2f0dcd8ed05fc"),
     '1': ObjectId("61911f92d3c2f0dcd8ed05fd") } }
```

8. Retrieving data where name=Siddhesh:

```
> db.Students.find({'name':'Siddhesh'})
< { _id: ObjectId("61911dedd3c2f0dcd8ed05fb"),
    name: 'Siddhesh',
    rollno: 30,
    course: 'CS' }
```

9. Count number of documents (records) present in the database:

```
> db.Students.find().count()
< 3
```

10. Update course to IT where name= Siddhesh

```
> db.Students.updateOne({'name':'Siddhesh'},{$set: {'name':'Siddhesh','rollno':30,'course':'IT'}},{upsert:true})
< { acknowledged: true,
    insertedId: null,
    matchedCount: 1,
    modifiedCount: 1,
    upsertedCount: 0 }
> db.Students.find({'name':'Siddhesh'})
< { _id: ObjectId("61911dedd3c2f0dcd8ed05fb"),
    name: 'Siddhesh',
    rollno: 30,
    course: 'IT' }
```

11. Delete document where name=Summer

```
> db.Students.remove({'name':'Summer'})
< 'DeprecationWarning: Collection.remove() is deprecated. Use deleteOne, deleteMany, findOneAndDelete, or bulkW
< { acknowledged: true, deletedCount: 1 }
> db.Students.find()
< { _id: ObjectId("61911dedd3c2f0dcd8ed05fb"),
    name: 'Siddhesh',
    rollno: 30,
    course: 'IT' }
  { _id: ObjectId("61911f92d3c2f0dcd8ed05fc"),
    name: 'Harry',
    rollno: 1,
    course: 'CS',
    mobno: '9930012834' }
```

12. Delete all documents from Students collection:

```
> db.Students.remove({})
< { acknowledged: true, deletedCount: 2 }
> db.Students.find()
<
Project >
```

13. Drop Students Collection

```
> db.Students.drop()
< true
Project >
```

14. Drop Project Database:

```
> db.dropDatabase()
< { ok: 1, dropped: 'Project' }
Project >
```

Conclusion:

MongoDB are used to save unstructured data.

MongoDB is saves data in BSON (Binary format of JSON) format.

MongoDB does not support advanced analytics and joins like SQL databases support.