ASSIGNMENT 1

Aim

Create a sample database using MongoDB & implement the CRUD operations.

Objective

- To learn NoSQL database MongoDB
- To study & execute CRUD operations.

Problem Statement

To create a sample database using MongoDB & implement CRUD operations. Perform all of the basic commands :

Theory

- **What is MongoDB ?**

MongoDB is a document-oriented NoSQL database used for high volume data storage. Instead of using tables & rows as in the traditional relational databases, MongoDB makes use of collections & documents. Documents consist of key-value pairs which are basic unit of data in MongoDB, while collections contain sets of documents.

- **Important characteristics of MongoDB**

- Schema - less

MongoDB is a schema-less database which makes it more flexible than traditional database tables. The benefit is the lack of setup & reduced friction with OOP.

- BSON

BSON stands for Binary JSON which is a binary encoded serialization of JSON like documents that MongoDB uses when storing documents. It is more effecient in size & speed, enables internal indexing & adds support for data types not in JSON.

- Indexing

Indexes are created to improve the performance of searches. It allows the database engine to effeciently resolve queries which makes it one of the best features of MongoDB.

- Aggregation Framework

The aggregation framework enables users to obtain the kind of results for which the SQL GROUP BY clause is used.

- Sharding

Sharding is a method for distributing data across multiple machines. MongoDB uses sharding to support deployments w/ very large datasets & high throughput operations, while providing horizontal scalability.

- Commands

- use

Switch current database to <db>

Syntax : use <db>

- show dbs

Syntax : show dbs

Print a list of all databases on the servers.

show collections

Syntax : show collections

Print a list of all collections for current database.

createCollection

Syntax : db.createCollation (<name>,

```
{
    capped : <boolean>,
    autoIndexId : <boolean>,
    size : <number>,
    max : <number>,
    storageEngine : <document>,
    validator : <document>,
    validationLevel : <string>,
    indexOptionDefaults : <document>,
    viewOn : <string>,
    pipeline : <pipeline>,
    collation : <document>,
    writeConcern : <document>
}
)
```

creates a new collection or view.

- insert

Syntax : db.collection.insert (
<document or array of documents>,
{
writeConcern : <document>,
ordered : <boolean>
}
)

Insert a document or documents into a collection.

- insertMany

Syntax : db.collection.insertMany (
[<document 1>, <document 2>, ...],
{
writeConcern : <document>,
ordered : <boolean>
}
)

Insert multiple documents into a collection.

- insertOne

Syntax : db.collection.insertOne (
<document>,
{
writeConcern : <document>
}
)

Insert a document into a collection.

- deleteOne

Syntax : db.collection.deleteOne(
<filter>,
{
writeConcern : <document>,
collation : <document>
}
)

Removes a single document from a collection

- deleteMany

Syntax : db.collection.deleteMany(
<filter>,
{
writeConcern : <document>,
collation : <document>
}
)

Remove all documents that match the filter from a collection.

- dropDatabase

Syntax : db.dropDatabase(<writeConcern - optional>)

Removes current database, deleting the associated data files.

- remove

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

Removes documents from a collection.

- update

Syntax : db.collection.update (
<query>,
<update>,
{
upsert : <boolean>,
multi : <boolean>,
writeConcern : <document>,
collation : <document>,
arrayFilters : [<filterDoc1>, ...],
hint : <document|string>
}
)

Modifies an existing document(s) in a collection.

- updateOne
- Syntax : db.collection.updateOne (
  <filter>,
  <update>,
  {
  upsert : <boolean>,
  writeConcern : <document>,
  collation : <document>,
  arrayFilters : [<filterDoc1>, ...],
  hint : <document | string>
  }
)

Updates a single document within the collection based on the filter.

- updateMany

Syntax : db.collection.updateMany (
<filter>,
<update>,
{
upsert : <boolean>,
writeConcern : <document>,
collation : <document>,
arrayfilter : [<filterDoc1>, ...],
hint : <document | string>
}
)

Updates all documents that match the specified filter for a collection.

- save

Syntax : db.collection.save (
<document>,
{
writeConcern : <document>
}
)

Updates an existing document or inserts a new document, depending on it's <document> parameter.

- find

Syntax : db.collection.find (
<query>,
<projection>
)

Selects documents in a collection or view & returns a cursor to the selected documents.

- findOne

Syntax : db.collection.findOne (
<query>,
<projection>
)

Returns one document that satisfies the specified query criteria on the collection.

drop

Syntax : db.collection.drop (
{ writeConcern : <document> }
)

Removes a collection from the database.

Input

Sample Collection

Output

Execution of all commands explained.

Platform

Linux

Conclusion

Thus created sample DB using mongoDB & implemented (RUD) operations.

FAQs

Q. What are CRUD operations ? Enlist some of them.

A. CRUD stands for Create, Read, Update & Delete. It is the set of operations you can perform with a database. for e.g. :-

- Create : Insert new document into a collection.
- Read : Retrieve document from a collection
- Update : Modify existing document in a collection.
- Delete : Remove documents from a collection.

Q. Compare SQL & NoSQL.

A.

SQL	NoSQL
- Relational Database Management System.	Non-relational or Distributed Database System.
- Fixed or pre-defined schema.	Dynamic schema.
- Not suited for hierarchical data storage.	Best suited for hierarchical data storage.
- Best suited for complex queries	Not suited for complex queries.
- Vertically scalable.	Horizontally scalable.

Q. Compare & explain mongo, mongod & mongos.

A.

- Mongo
  - An interactive shell (cli)
  - Fully functional JavaScript environment for use with an existing MongoDB instance.

- mongos
  - MongoDB shard utility
  - Controller & query router for the sharded cluster.

- mongod
  - Primary daemon process for MongoDB instance.
  - Handles data requests, access, & performs background mgmt operations.