Experiment No.3
To install and configure MongoDB to execute NoSQL commands
Date of Performance:
Date of Submission:

AIM: To install and configure MongoDB/ Cassandra/ HBase/ Hypertable and to execute NoSQL commands.

THEORY:

MongoDB can be downloaded from https://www.mongodb.com/try/download/community2 Now open command prompt and run the following command

```
C:\>move mongodb-win64-* mongodb
    1 dir(s) moved.
```

MongoDB requires a data folder to store its files. The default location for the MongoDB data directory is c:\data\db. So create the folder using the Command Prompt. Execute the following command sequence.

```
C:\>md data
C:\md data\db
```

In case mongodb is stored in some other location, navigate to that folder.

In command prompt navigate to the bin directory present into the mongodb installation folder. Suppose the installation folder is D:\set up\mongodb

```
C:\Users\XYZ>d:
D:\>cd "set up"
D:\set up>cd mongodb
D:\set up\mongodb>cd bin
D:\set up\mongodb\bin>mongod.exe --dbpath "d:\set up\mongodb\data"
```

Now to run the mongodb, open another command prompt and issue the following command:

```
D:\set up\mongodb\bin>mongo.exe
MongoDB shell version: 2.4.6
connecting to: test
>db.test.save( { a: 1 } )
>db.test.find()
{ "_id" : ObjectId(5879b0f65a56a454), "a" : 1 }
>
```

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

The dropDatabase () Method

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

Syntax:

db.dropDatabase()

The createCollection() Method

MongoDB db.createCollection(name, options) is used to create collection.

Syntax:

db.createCollection(name, options)

Insert Document

To insert data into MongoDB collection, you need to use MongoDB's insert() or save()method

Syntax

>db.COLLECTION_NAME.insert(document)

Example:

```
>db.post.insert([
{
 title: 'MongoDB Overview',
 description: 'MongoDB is no sql database',
tags: ['mongodb', 'database', 'NoSQL'], likes:
100
}, {
 title: 'NoSQL Database',
 description: 'NoSQL database doesn't have tables',
tags: ['mongodb', 'database', 'NoSQL'], likes: 20,
comments: [
 { user:'user1',
 message: 'My first comment', dateCreated:
new Date(2022,11,10,2,35), like: 0
 }
 ]
}
])
```

Creating sample document:

Example

Suppose a client needs a database design for his blog website. Website has the following requirements.

- Every post has the unique title, description and url.
- Every post can have one or more tags.

 Every post has the name of its publisher and total number of likes.

 Every Post have comments given by users along with their name, message, data-time and likes.

 On each post there can be zero or more comments.

Document:

```
{
 _id: POST_ID  title: TITLE_OF_POST,
description: POST_DESCRIPTION,  by:
POST_BY,  url: URL_OF_POST,  tags:
[TAG1,  TAG2,  TAG3],  likes:
TOTAL_LIKES,  comments: [
 {
 user:'COMMENT_BY',
message:  TEXT,  dateCreated:
DATE_TIME,  like: LIKES
 },
 {
 user:'COMMENT_BY',
message:  TEXT,  dateCreated:
DATE_TIME,  like: LIKES

 }
 ]
}
```

Screenshot :

C:\Program Files\MongoDB\Server\5.0\bin\mongo.exe

```
        2023-10-13T22:10:50.835+05:30: Access control is not enabled for the database. Read and write access to data and configuration is u
---
---
        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()
---
> use db1
switched to db db1
> db.createCollection('Post')
{
        "ok" : 0,
        "errmsg" : "Collection already exists. NS:  db1.Post",
        "code" : 48,
        "codeName" : "NamespaceExists"
}
> db.Post.insert([
...     {
...         title: 'MongoDB Overview',description: 'MongoDB is no sql database', tags: ['mongodb','database','NoSQL'], likes:100
...     },
...
...     {
...        title:'NoSQL Database',
...         description:'NoSQL database doesnt have tables',
...         tags:['mongodb','database','NoSQL'],
...         likes:20,
...         comments:{
...             user:'user1',
...             message:'My first comment',
...                   dateCreated:new Date(2022,11,10,2,35),
...                   like:0}
...     }
... ])
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 2,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
>
```

Conclusion:

In this experiment we learned No-SQL databases. MongoDB, Cassandra, HBase, and Hypertable are different NoSQL databases, with each of them requiring specific installation setup and configuration. Executing NoSQL commands encompasses various data operations using database-specific languages or APIs. The selection of the appropriate NoSQL database should be driven by project needs and database features. To effectively use any NoSQL database, a deep understanding of its architecture and query language is crucial for successful implementation.