Laboratory Tutorial 11: NoSQL - Document DB and MongoDB

In this laboratory tutorial you will work with one of the document-based databases – MongoDB.

Preamble

In the previous lab tutorial, you gained experience in working with JSON. Specifically, you represented data in JSON format, read and wrote JSON files, converted a JSON string to Python object and vice versa, and validated JSON documents.

In this lab tutorial you will work with No SQL – Document DB, specifically:

- Install MongoDB
- Create databases, collections, and documents in MongoDB
- Perform queries from the collections in MongoDB

Exercise 11.1: Introduction to MongoDB

As taught in the lecture, one of the types of NoSQL is Document DB. Document DB uses a key/value pair, and consists of collections (like tables) and documents. One example of Document DB is MongoDB. MongoDB differs from relational databases in the sense that it is a schema-less database. The format of MongoDB documents are similar to JSON. MongoDB provides backend services to a number of websites, including eBay, New York Times, and Viacom.

Exercise 11.2: Installing MongoDB

Step 1

Click on the link: https://www.mongodb.com/try/download/community

Note: From the MongoDB Community Server, click download

Step 2

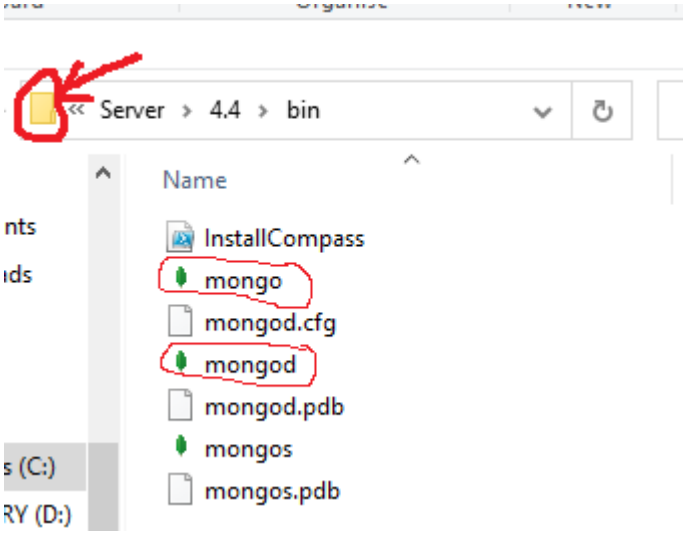
We will create a data folder where MongoDB will save all its data. Go to **C drive**, create a new folder called **data**. Within the folder data create another folder called **db.** We need to make sure this step is met otherwise we will not be able to run MongoDB.

Step 3

To run MongoDB we need to start MongoDB server first.

Go to C: --Program Files –MongoDB—Server—4.4—bin

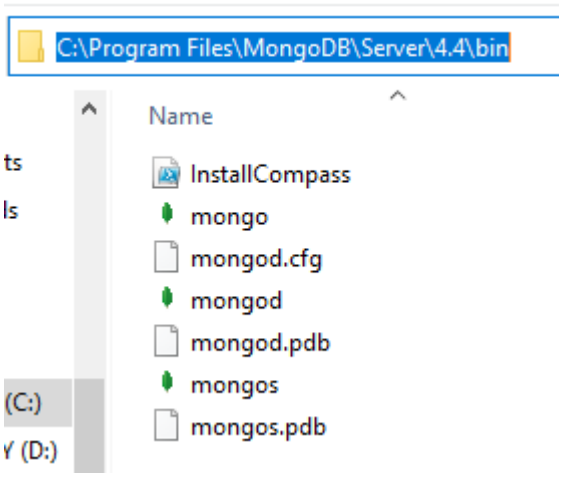
As shown in the screenshot below, we have the mongod (which is the server) and the mongo, both as executable files.



First we need to start the server so that we can run MongoDB.

Click on the small icon (denoted by the arrow) and you will be transferred with the below path**. Add \mongod, and press enter:**

C:\Program Files\MongoDB\Server\4.4\bin\mongod



The **mongod** command is used to run a server at a specific port number ‘27017’

MongoDB server is now on.

Step 4

We can now run our mongoDB.

Click on the small icon (denoted by the arrow in step 3) again and you will be transferred with the below path**. This time add \mongo, and press enter:**

C:\Program Files\MongoDB\Server\4.4\bin\mongo

The > sign means we are ready to start with MongoDB.

If you are using mac, then use the following link:

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/

Exercise 11.3: Creating databases, collections, and documents in MongoDB

A MongoDB can hold a number of databases. Each database can hold a set of collections (similar to tables that are used in relational DB). Each collection holds a set of documents. Documents are a set of key-value pairs.

From:

C:\Program Files\MongoDB\Server\4.4\bin\mongo

1. We are going to check if we have any databases. To do that we type the following command:
   >show dbs

   Press enter and you should see something like:

   ```
   admin    0.000GB
   config   0.000GB
   demo     0.000GB
   local    0.000GB
   ```

   The above are current databases. We need our own database called 'data2'

2. To create our own database, we use the following command.
   >use data2

   Press enter and you should see the following:

   ```
   switched to db data2
   ```

   *Our database 'data2' is now created, which we are currently working with. If we type **show dbs** we will still not see our database. This is because we have not created at least one collection.*

3. We need to create a collection:
   >db.createCollection("info")

   Syntax: db.createCollection(*collection_name*)

4. To see our collection(s):
   >show collections

5. Now, we will add one document in our collection:

   ```
   > db.info.insertOne({"_id":101,"name":"JS","age":23,"course":"Computer Science"})
   ```

Press enter. *You will receive the following acknowledgement.*

```
{ "acknowledged" : true, "insertedId" : 101 }
```

Note: name of the collection is **info**, and documents are everything within the curly braces (id of 101, age of 23, ….). **insertone**: insert **one** document

Syntax: db.*collection_name*.insertone()

6. We may need more documents in our collection. To do that we use **insertMany.**

Syntax: db.*collection_name*.insertMany()

Let us add two more documents:

>db.info.insertMany([{"_id":102, "name": "SG", "age":19, "course": "Software Engineering"},{"_id":103, "name": "AS", "age":21, "course": "Artificial Intelligence"}])

Note: we have used an array of 3 objects

7. Next, we want to check whether the entries have been added. We simply type **db.info.find()** and then press enter.

```
> db.info.find()
{ "_id" : 101, "name" : "JS", "age" : 23, "course" : "Computer Science" }
{ "_id" : 102, "name" : "SG", "age" : 19, "course" : "Software Engineering" }
{ "_id" : 103, "name" : "AS", "age" : 21, "course" : "Artificial Intelligence" }
```

Exercise 11.4: Performing queries in MongoDB

Use of find() method

We can use db.*collection_name*.find(query,projection), where query and projection are parameters.

query – Optional. Specifies selection filter using query operators (refer to the tables below). To return all documents in a collection, omit this parameter.

Table 1: Comparison Query operators

Name	Description
$eq	Matches values that are equal to a specified value
$gt	Matches values that are greater than a specified value
$gte	Matches values that are greater than or equal to a specified value
$in	Matches any of the values specified in an array
$nin	Matches none of the value specified in an array
$lt	Matches values that are less than a specified value
$lte	Matches values that are less than or equal to a specified value
$ne	Matches all values that are not equal to a specified value

Table 2: Logical Query operators

Name	Description
$and	Logical AND operation
$not	Logical NOT operation
$or	Logical OR operation

Syntax: { $and | $or | $not : [{ <expression1> }, { <expression2> } , ... , { <expressionN> }] }

8. Below is an example of the use of the parameter query to list details of all students who are taking computer science course:

```
> db.info.find({course:{$eq:"Computer Science"}})
```

Output:

```
{ "_id" : 101, "name" : "JS", "age" : 23, "course" : "Computer Science" }
```

Now, answer the following questions where you have to use this parameter

9. Write a query to list details of all students who are above 20 years old.

10. Write a query to list details of all students where the following courses are not involved – Artificial Intelligence and Computer Science

11. Write a query to select all documents in the info collection where an age exists and the age is not equal to 25

More on find() method

We can use db.*collection_name*.find(query,projection), where query and projection are parameters.

projection – optional. Specifies the fields to return in the documents that match the query filter. To return all fields in the matching documents, omit the parameter.

Let us go through one example of projection

12. Suppose we only want the **course** for JS. We can write the following command:

```
> db.info.find({"name": {$eq:"JS"}},{"course":1})
```

1 is true, 0 is false. We get the following result:

```
{ "_id" : 101, "course" : "Computer Science" }
```

But we do not want that, we only want the course to appear.

So, we need this command:

```
> db.info.find({"name": {$eq:"JS"}},{"course":1,"_id":0})
```

Result:

```
{ "course" : "Computer Science" }
```

Below is a table of three additional MongoDB Collection methods

Table3: MongoDB Collection methods

Methods	Description	Syntax
sort()	Sort the documents in ascending or descending order. Default value is ascending order.	db.*collection_name*.find().sort({*field_key*:1 or -1}) 1 is ascending order whereas -1 is descending order
limit()	Limits the of documents returned	db.*collection_name*.find().limit(*num_of_documents*)
count()	Counts the number of documents that match a find() query	db.*collection_name*.find(*query*).count()
	To count all documents of a collection without any query	db.*collection_name*.find().count() **or** db.*collection_name*.count()

Now, answer the following questions where you have to use the above methods:

13. Write a query to return number of students who are enrolled on Artificial Intelligence.

14. Write a query to display details of all students, with age in descending order.

15. Write a query to display the first two documents of the collection.

16. **Adding more than one database**

You can add as many databases as you want. Let us create another database called 'HCI':

>use HCI

Now, you have been switched to DB 'HCI'. If you want to work with DB 'data2' (our first database) you need the following command:

>use data2

You are now switched back to DB 'data2'

If you are not sure which DB you are working with (especially when you have a lot), then use the command:

>db

'data2' is displayed. This means you are currently working with DB 'data2'

17. **Drop a DB**

If you want to drop a database, you need to follow some steps. Let us drop DB 'HCI' which we just created:

>use HCI

switched to db HCI #You have now switched to DB 'HCI'

>db

HCI #You are working with DB 'HCI'

>db.dropDatabase()

{ "ok : 1 } #DB 'HCI' is now deleted

>show dbs

You will now see that DB 'HCI' has been deleted.

Summary

In this tutorial you have learned how to install MongoDB. Additionally, you have learned how to run the MongoDB server and MongoDB. You have gained knowledge of creating databases, adding collections and documents within a database. Finally, you have learned how to create queries.

Further Reading

https://beginnersbook.com/2017/09/mongodb-update-document/

https://www.analyticsvidhya.com/blog/2020/02/mongodb-in-python-tutorial-for-beginners-using-pymongo/