lab 2 Document Databases

Goals

The objectives of this work are:

- Understand the fundamentals of document-based databases.
- Install and use a free-to-use solution - mongodb.
- Develop solutions for various use cases

prior note

This module should preferably be developed in Linux. If you intend to use Windows, check the compatibility notes for the software you will be using.
Submit code/results/reports to elearning. Use a folder (1, 2, ..) for each exercise, compressed into a single file.
Good job!

2.1 MongoDB – Command-line installation and exploitation

MongoDB is a document-oriented database represented in a JSON structure (internally it uses BSON, a binary version of JSON). It is an open source project, licensed under the GNU AGPL (Affero General Public License).

a) Install MongoDB on your personal computer (https://www.mongodb.com/) and run the server (mongod). For example (may vary by OS):

```
$ mongod --dbpath <path to data directory>
```

b) Study the functioning of the system by testing the most used commands, through the command line (client programmongo):

```
$ mongo [--username <user> --password <pass> --host <host> --port <portN>]
```

You can also use an interactive management application such as, for example, studio3Qor theMongo Compass .

See the slides provided for the course and websites with documentation about MongoDB. Some examples:

- *MongoDB Docs,https://docs.mongodb.com*
- *https://www.tutorialspoint.com/mongodb/*

You should study concepts and features such as:

- Storage structure (DB, Collections, Documents)
- JSON and Javascript
- Write, Read, Edit, Delete (CRUD)
- types and arrays

- Indexes
- Aggregations and mapreduce

c) Produce a notebook,CBD_L201_<NMEC>.txt,with your personal notes on this phase, including all iterations with mongo, commands and a few lines of output. Example:

```
# CBD – Lab201 – <Name>

> show dbs
admin      0,000GB
cbd        0.001GB
config     0,000GB
place      0,000GB
sales      0,000GB

> . . .
```

2.2 MongoDB – Query construction

For this exercise you should use the file "restaurants.json". This contains information about 3772 restaurants, according to the following JSON structure:

```
{
      "address": {
           "building":"1007",
           "coord":[
                 - 73.856077,              // assume as longitude
                 40.848447                 // assume as latitude
           ],
           "street":"Morris Park Ave",
           "zipcode":"10462"
    },
    "locality":"Bronx",
    "gastronomy":"Bakery",
    "grills":[
             { "date": { "$date": 1393804800000 }, "grade": "A", "score": 2 }, { "date": { "$date":
             1378857600000 }, "grade": "A" , "score": 6 }, { "date": { "$date": 1358985600000 },
             "grade": "A", "score": 10 }, { "date": { "$date": 1322006400000 } , "grade": "A",
             "score": 9 }, { "date": { "$date": 1299715200000 }, "grade": "B", "score": 14 }

      ],
      "name":"Morris Park Bake Shop",
      "restaurant_id":"30075445"
}
```

You can put this data in your local mongo installation, using the following command:

```
$ mongoimport --db cbd --collection restaurants --drop --file <path/
>restaurants.json
```

Run the mongo client (command line) and check if the data has been uploaded to the server.

```
$ mongo

> use cbd
> show collections
restaurants
> db.restaurants.count() 3772
```

Using mongo in command line mode, or a client program, write expressions/ queries to get the expected results for the following questions. Write all answers in the fileCBD_L202_*<NMEC>*.txt, Where*<NMEC>*must be replaced by its mechanistic number. For each question, write the question, the command used, and the result. If you want to include comments use "//".

Example:

```
// NMEC: 12345

// 1. List all documents in the collection.
db.restaurants.find()
// 3772

// 2. Present the restaurant_id, name, location and gastronomy fields for all documents in the
collection
. . .
```

Queries:

1. List all documents in the collection.
2. Display the fields*restaurant_id*,*Name*,*locality*and*gastronomy*for all documents in the collection.
3. Present the fields*restaurant_id*,*Name*,*locality*and postal code (*ZIP code*), but delete the field*_id*of all documents in the collection.
4. Enter the total number of restaurants located in the Bronx.
5. List the top 15 restaurants located in the Bronx, sorted in ascending order of name.
6. List all restaurants that have at least one*score*higher than 85.
7. Find the restaurants that scored one or more scores (*score*) between [80 and 100].
8. Indicate restaurants with latitude lower than -95.7.
9. Indicate the restaurants that do not have*gastronomy*"American", had a (or more) score greater than 70 and are at a latitude less than -65.
10. List the*restaurant_id*, O*Name*, a*locality*and*gastronomy*of restaurants whose name starts with "Wil".
11. List the*Name*, a*locality*and the*gastronomy*of the restaurants that belong to the Bronx and whose*gastronomy*is of the "American" or "Chinese" type.
12. List the*restaurant_id*, O*Name*, a*locality*and the*gastronomy*restaurants located on "Staten Island", "Queens", or "Brooklyn".

13. List the*Name*, a*locality*, O*score*and*gastronomy*of the restaurants that always achieved scores lower than or equal to 3.

14. List the*Name*and the reviews of the restaurants that obtained a rating with a *grid*"a", a*score*10 on the date "2014-08-11T00: 00: 00Z" (ISODATE).

15. List the*restaurant_id*, O*Name*and the*score*of the restaurants in which the second evaluation was*grid*"A" and occurred on ISODATE "2014-08-11T00:00:00Z".

16. List the*restaurant_id*, O*Name*, O*address*(*address*) and geographic coordinates (*coordinate*) of restaurants where the 2nd element of the coordinate matrix has a value greater than 42 and less than or equal to 52.

17. List*name, cuisine and location*of all restaurants in ascending order of*gastronomy*and second, in descending order of*locality*.

18. List*Name*,*locality, grid*and*gastronomy*of all restaurants located in Brooklyn that do not include*gastronomy*"American" and obtained a rating (*grid*) "A". You must present them in descending order of*gastronomy*.

19. Count the total number of restaurants in each location.

20. List all restaurants whose average*score*is greater than 30.

21. Indicate the restaurants that have*gastronomy*"Portuguese", the sum of*score*is greater than 50 and are at a latitude less than -60.

22. Provide the name and score of the 3 restaurants with the highest average score.

23. Display the number of different cuisines on "Fifth Avenue" street

24. Count how many restaurants there are per street and order in descending order

25. .. 30. Describe 5 additional questions to the database (items 26 to 30), significantly different from the previous ones, and also present the research solution for each question.

2.3 MongoDB – Server-side functions

In this exercise we intend to use and develop javascript functions to run on the MongoDB server.

a) Copy the file*populatePhones.js*to a folder on your desktop. Start the mongo client and download the function "populatePhones". Review and test the operation of this function.

```
> load("<your desktop folder>/populatePhones.js") true


> populatePhones
function (country, start, stop) {

    var prefixes = [21, 22, 231, 232, 233, 234]; for (var i =
    start; i <= stop; i++) {

        var prefix = prefixes[(Math.random() * 6) << 0] var
        countryNumber =
            (prefix * Math.pow(10, 9 - prefix.toString().length)) + i; var num =
        (country * 1e9) + countryNumber;
        var fullNumber = "+" + country + "-" + countryNumber;
        db.phones.insert({
            _id: num,
            components: {
                country: country,
                prefix: prefix,
                number: i
            },
            display: fullNumber
        });
        print("Inserted number " + fullNumber);
    }
    print("Done!");
}

> populatePhones(351, 1, 5) Inserted
number +351-233000001 Inserted
number +351-231000002 Inserted
number +351-234000003 Inserted
number +351-234000004 Inserted
number +351-220000005 Done!
```

(Note : loads in the "current shell". Does not store on server)

b) After the tests of the previous paragraph, clean the collection (db.phones.drop()) and using this function create 200,000 numbers, for example:

```
> populatePhones(351, 1, 200000)
```

(Note: this operation is not immediate).

At the end, check the contents of the collection (using the functionsfind, count,...).

c) Build a function/expression that counts the number of telephones in each of the national codes (*prefix*).

d) Build, and test on the server, a JavaScript function that finds a type of pattern in the list (eg, capicuas, sequences, non-repeating digits, etc.).

2.4 MongoDB – Driver (Java)

For this exercise, you should use the collection*restaurants*, but now accessing it programmatically through one of the drivers available on the mongo website (https://docs.mongodb.com/ecosystem/drivers/).
You should use the Java driver creating a project*maven*, or using the following jar files:monogodb driver core ,mongodb driver ,bson .

a) Develop a simple program that allows testing insertion, editing and searching of records about the collection.

b) Create indexes: one for locality; another for gastronomy; and a text one for the name. Use surveys to test health and verify performance (*as there are few documents, the results may not improve*).

c) Select 5 questions/commands from exercise 2.2 and re-implement them in Java.

d) Build and test the following methods, presenting the result of their execution in the fileCBD_L204_<NMEC>.txt:

- public int countLocalities()

```
Number of different locations: 6
```

- Map<String, Integer> countRestByLocality()

```
Number of restaurants by location:
  - > Queens - 738
  -> Staten Island - 158
  -> Manhattan - 1883
  -> Brooklyn – 684
. . .
```

- List<String> getRestWithNameCloserTo(String name)

```
Name of restaurants containing 'Park' in the name:
  -> Morris Park Bake Shop
  - > New Park Pizzeria & Restaurant
  - > Parkside Restaurant
  - > New Parkway Restaurant
. . .
```

2.5 Free Themed Database

This exercise aims to create a collection that takes advantage of the MongoDB data model. Please note the following recommendations and requirements:

(a) The database can be created by adapting or importing a public dataset.

(b) The number of documents in the database and the complexity of the data structure of each document is a factor to be taken into account, in order to be able to create queries with some complexity and diversity of logical operations. For example, create a database with hundreds of documents and use “array of embedded documents”.

(c) Create 6 expressive queries of your domain of knowledge using the operator*find({…}, {…}).*

(d) Create 6 expressive queries of your knowledge domain using the operator *aggregate($group, $project, $unwind, $match, etc)*.

Note: in c) and d) can be done in mongo scripts or using the java API. It should not replicate queries available on public forums.
Describe the dataset structure and all the questions and answers in the file CBD_L205_*<NMEC>*.TXT.