NoSQL Databases
Lab 3: Document Databases - MongoDB

MongoDB is a free and open-source cross-platform document-oriented database. Classified as a NoSQL database, MongoDB avoids the traditional table-based relational database structure in favor of JSON-like documents with dynamic schemas, making the integration of data in certain types of applications easier and faster.

A - Installing and running the MongoDB server

MongoDB can be downloaded from the following webpage: https://www.mongodb.org/downloads
The first step is to install MongoDb. The installation details is gvien in the file linked to this lab.

B - Create and fill a database

Once MongoDb is installed, no database (and therefore no data) exists. That's what we're going to do now. There are two ways to do this:

1. Create the database and manually insert the records.
2. Create the database and import data using a JSON format data file.

Schematically, a MongoDb server can store several databases which themselves can contain several collections. To create a database, just use the command use < **DBname** >. Executing this command creates the base and selects it as the current base.
We will now see the two ways to populate a collection.

B.1 - Creation and manually insertion

The creation of a collection is somehow automatic when you insert a record in a collection that does not exist yet. The command to insert a record is:

db. <dbname> .insert (<Json record>)

Insert the following records into the sid1 database:

```
{
 "address": {
    "building": "1007",
    "coord": [ -73.856077, 40.848447 ],
    "street": "Morris Park Ave",
    "zipcode": "10462"
```

```
  },
  "borough": "Bronx",
  "cuisine": "Bakery",
  "grades": [
     { "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"
}
```

To verify that the insertion went well, run the db.sid1.find () command. This command returns all the contents of the sid1 database.

B.2 - Creation and automatically insertion

For this part, we will use a classic dataset. This one describes some restaurants as well as notes given by the customers. Here is an example from the dataset:

1. Download the data (data.json) from teams.
2. Run the command below:

mongoimport –db sid –collection restaurants –drop –file data.json

C - Querying a collection

In the previous section, we saw that the find function without arguments returns the set of records in a collection. We will now see how to add conditions to filter the results returned by the find function. The format of the parameter to specify to return a result that matches the specified conditions is as follows:

<field1>: <value1>, <field2>: <value2>, ...

If the field is not in a nested document, quotation marks (single or double) are not required. On the other hand, if field is in a nested object or in a table, it will have to go through the object notation (the point). At this point, the quotation marks are mandatory.

Operators MongoDb provides operators to specify conditions, such as comparison operators. These operators are usually of the form:

<field1>: <operator1>: <value1>

- L'opérateur > : gt
- L'opérateur < : lt

Combine the conditions It is possible to combine the operators with the classic AND and OR.

- The logical AND: It is done by separating the conditions by a comma.
- The logical OR: The logical OR is expressed with the or operator. For example, to return restaurants whose specialties are Italian OR whose zip code is 10075, the request must be entered:

```
db.restaurants.find(
   { $or: [ { "cuisine": "Italian" }, { "address.zipcode":
"10075" } ] }
)
```

Sort records In this section, we will see how to sort the result of a query. To this end, MongoDb introduces the command / function sort. We will study how it works. Just like the find command, the sort command accepts a list of options to define, among other things, which fields to sort the documents in and in which order (1 for ascending and -1 for descending).
Here is an example that finds all male, English or American actors, and who orders them according to their nationality, in reverse alphabetical order.

```
db.sid1.find({gender: 'm', $or: [{nationality: 'english'},{
   nationality: 'american'}]}).sort({nationality: -1});
```

To sort on two fields, you must proceed as for the conjunction of conditions in the find command.

Limit returned records If we want to limit the number of returned records, MongoDb introduces the limit command which is an example:

```
db.sid1.find({gender: 'm', \$or: [{nationality: 'english'},
    {nationality: 'american'}]}).limit(2);
```

If, in addition, we want to ignore the first two records and return only the third and fourth, then we will use the skip command, here is an example:

```
db.sid1.find({gender: 'm', \$or: [{nationality: 'english'},
    {nationality: 'american'}]}).limit(2).skip(2);
```

Update of records MongoDb obviously provides a command to update records. Here is the syntax:

db.<collection>.update(<conditions>,$set:<field>:<value>);

where <conditions> are conditions specified as for the find command.

Write a MongoDB query to:

1. Display all the documents in the collection restaurants.
2. Display the fields restaurant_id, name, borough and cuisine for all the documents in the collection restaurant.
3. Display the fields restaurant_id, name, borough and cuisine, but exclude the field_id for all the documents in the collection restaurant.
4. Display the fields restaurant_id, name, borough and zip code, but exclude the field_id for all the documents in the collection restaurant.
5. Display all the restaurant which is in the borough Bronx.
6. Display the first 5 restaurant which is in the borough Bronx.
7. Display the next 5 restaurants after skipping first 5 which are in the borough Bronx.
8. Find the restaurants who achieved a score more than 90.
9. Find the restaurants that achieved a score, more than 80 but less than 100.
10. Find the restaurants which locate in latitude value less than -95.754168.
11. Find the restaurants that do not prepare any cuisine of 'American' and their grade score more than 70 and latitude less than -65.754168.
12. Find the restaurants which do not prepare any cuisine of 'American' and achieved a score more than 70 and located in the longitude less than -65.754168. Note : Do this query without using $and operator.
13. Find the restaurants which do not prepare any cuisine of 'American ' and achieved a grade point 'A' not belongs to the borough Brooklyn. The document must be displayed according to the cuisine in descending order.
14. Find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Wil' as first three letters for its name.
15. Find the restaurant Id, name, borough and cuisine for those restaurants which contain 'ces' as last three letters for its name.

16. Find the restaurant Id, name, borough and cuisine for those restaurants which contain 'Reg' as three letters somewhere in its name.

17. Find the restaurants which belong to the borough Bronx and prepared either American or Chinese dish.

18. Find the restaurant Id, name, borough and cuisine for those restaurants which belong to the borough Staten Island or Queens or Bronxor Brooklyn.

19. Find the restaurant Id, name, borough and cuisine for those restaurants which are not belonging to the borough Staten Island or Queens or Bronxor Brooklyn.

20. Find the restaurant Id, name, borough and cuisine for those restaurants which achieved a score which is not more than 10.

21. Find the restaurant Id, name, borough and cuisine for those restaurants which prepared dish except 'American' and 'Chinees' or restaurant's name begins with letter 'Wil'.

22. Find the restaurant Id, name, and grades for those restaurants which achieved a grade of "A" and scored 11 on an ISODate "2014-08-11T00:00:00Z" among many of survey dates..

23. Find the restaurant Id, name and grades for those restaurants where the 2nd element of grades array contains a grade of "A" and score 9 on an ISODate "2014-08-11T00:00:00Z".

24. Find the restaurant Id, name, address and geographical location for those restaurants where 2nd element of coord array contains a value which is more than 42 and upto 52..

25. Arrange the name of the restaurants in ascending order along with all the columns.

26. Arrange the name of the restaurants in descending along with all the columns.

27. Arranged the name of the cuisine in ascending order and for that same cuisine borough should be in descending order.

28. Know whether all the addresses contains the street or not.

29. Select all documents in the restaurants collection where the coord field value is Double.

30. Select the restaurant Id, name and grades for those restaurants which returns 0 as a remainder after dividing the score by 7.

31. Find the restaurant name, borough, longitude and attitude and cuisine for those restaurants which contains 'mon' as three letters somewhere in its name.

32. Find the restaurant name, borough, longitude and latitude and cuisine for those restaurants which contain 'Mad' as first three letters of its name.