UNIVERSITAT POLITÈCNICA
DE CATALUNYA
BARCELONATECH

Big Data Management (BDMA & MDS Masters)
Session: Document Stores (Training)

The purpose of this document is that you get familiar with the environment and tools required for the lab session. This is an optional task, thus no delivery is expected and no grade will be assigned. This document is far from being a fully-fledged tutorial, we refer you to the following references for in-depth guides to the used tools:

- MongoDB - `https://docs.mongodb.com/manual/`

Virtual Machine

All tools you need are provided in the BDM virtual machine, accessible via *Virtech* or available to download. Please, refer to the provided manual for details on how to set up the virtual machine. Notice that as lab sessions are performed in couples, only one virtual machine will be required during the session.

MongoDB Setup

First, start the `mongod` server

```
sudo service mongod start
```

In this first exercise we will import a JSON sample dataset using the `mongoimport` command. Particularly, this dataset depicts restaurants and a sample document looks like the following:

```
{
  "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"
}
```

In order to populate the `test` database, you have to download the dataset from the following url:

```
https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json
```

Then, execute the `mongoimport` to insert the documents into the restaurants collection in the test database. If the collection already exists in the test database, the operation will drop the restaurants collection first.

```
mongoimport --db test --collection restaurants --drop --file primer-dataset.json
```

The `mongoimport` connects to a `mongod` instance running on localhost on port number 27017. To import data into a `mongod` instance running on a different host or port, specify the hostname or port by including the –host and the –port options in your `mongoimport` command.

Now, connect to the `mongod` instance using the `mongo` client.

MongoDB Queries

If you imported the data successfully, the `show dbs` command will show the newly created `test` database. After selecting it, `use test`, the restaurants collection should be available, `show collections`, and you should be able to explore it via `db.restaurants.findOne()`. Now, in this exercise we ask you to implement the following queries in the shell:

Q1: Restaurants whose borough is "Manhattan".

Q2: Number of restaurants in "Manhattan" with some grade scored greater than 10.

Q3: Average score by cuisine type

Check the MongoDB CRUD operations section of the manual (`https://docs.mongodb.com/manual/crud/` to get familiar with the syntax. To perform aggregations you have to use the Aggregation Framework. Read more about the Aggregation Framework in `https://docs.mongodb.org/manual/core/aggregation-introduction/`. If you get many results you can use the `findOne` method to improve readability.

If you want to drop the `test` database, you can issue the following command:

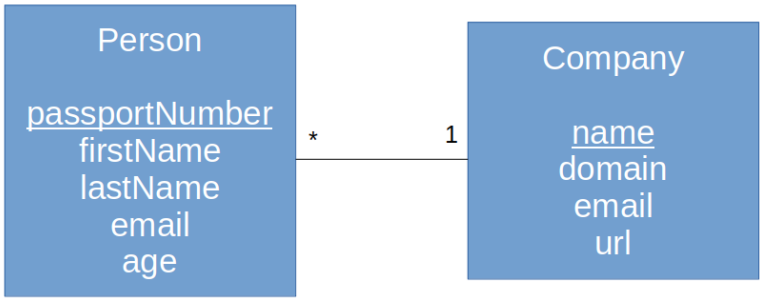
```
use test
db.dropDatabase()
```

MongoDB Java Driver

Here, we will explore the Java driver for MongoDB, which you will be required to use in the lab session. To this end, we refer you to the excellent tutorial *MongoDB Driver Quick Start*, available at `https://mongodb.github.io/mongo-java-driver/3.10/driver/getting-started/quick-start/`. Try to use such code to interact with your restaurants collection.

Modeling and the Java driver

Now we ask you to populate a specific data model and run queries over it using MongoDB's Java driver. Precisely, we will work with the following conceptual model depicted in UML.



Let us assume we are interested in obtaining the following information:

Q_1: For each person, his/her name and the domain of his/her company.

Q_2: For each company, the name and the number of employees.

The chosen design here consists on creating two collections, one for *People* and one for *Companies* and materialize the many-to-one relationship as an attribute in *People*. To populate the database we will use `JFairy` a random data generator for Java. After obtaining a new instance of Fairy using `Fairy.create()`, you can get the data of interest via the following methods:

- `Person p = fairy.person()`, provides a new instance of person. If the primary key is duplicate, this instance should be dismissed.
- `p.getCompany()`, provides the company where p works in.

In LearnSQL you will find a Java project with three classes. In this exercise, you are required to implement the following methods:

- In `populateDB.java`, implement the method `populate(N)`. This method must populate the two previously mentioned collections for N persons.
- In `Q1.java`, implement the method `execute()` to run query Q_1.
- In `Q2.java`, implement the method `execute()` to run query Q_2.

You can run the main class using the run configuration `"populate N"` to populate the database with N instances, and later `"Q1"` or `"Q2"` to run both queries.

Finally, try to think of alternative designs for the documents in the database. Take into account the trade-off between data replication and query performance.