CentraleSupélec

SACLAY CAMPUS, COMPUTER SCIENCE DEPARTMENT
2019-2020

Models and Systems for Big Data
Document Database
MongoDB & Javascript-based Query Language

Pr. Nacéra Seghouani

The purpose of this practical work is to use MongoDB Community as NoSQL document-oriented database [1] server. Studio 3T [2] is required as a client to connect to a MongoDB server (instance) and to submit queries.

1 HOW TO START WITH MONGODB, DATA IMPORT AND STUDIO3T

First, you need to install MongoDB Community. The instructions are available at:

- `https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/` for macOS, all programs are available in `/usr/local/.../mongodb/4.0.4_1/bin` (if you follow carefully the installation the repository is explicitly displayed)
- and `https://docs.mongodb.com/v3.2/tutorial/install-mongodb-on-windows/` for Windows.

Then, you install the client Studio 3T using the link `https://studio3t.com/download/`.

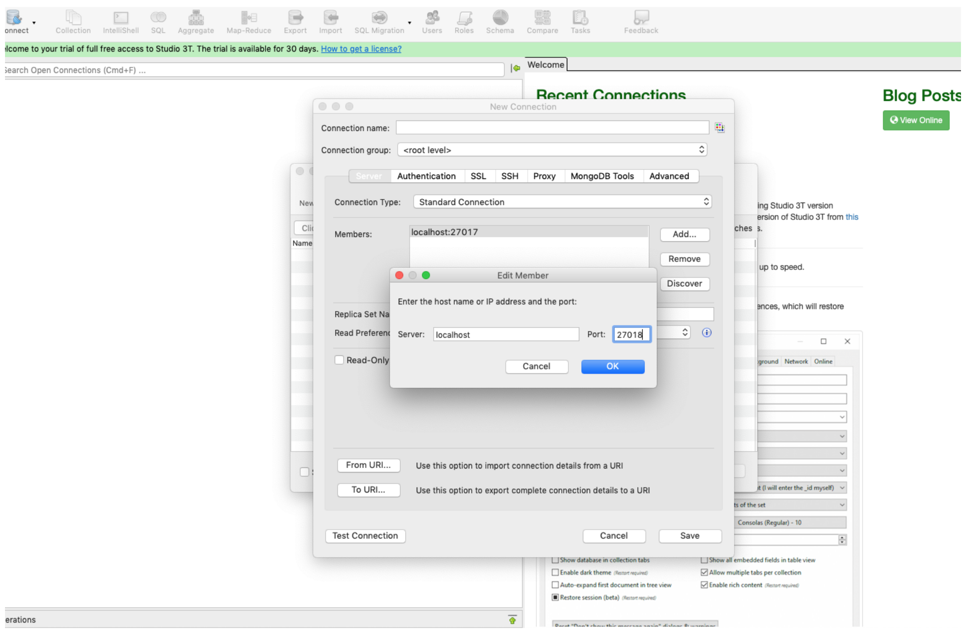
Here are the instructions to launch an instance of MongoDB server :

- Create a directory `"data"` in `"/PATH"` where `"PATH"` is the path you used.
- Open a terminal to run an instance of MongoDB server by typing the following command. **Keep this terminal open** otherwise the launched instance will be stopped.

  ```
  mongod --dbpath /PATH/data --port 27018
  ```

  The option `port` is used to specify the port the server listens to. The option `dbpath` is used to specify the directory where the data files will be stored. Each MongoDB instance manages its own data directory. For each MongoDB instance you must specify a new data directory.

[1] `https://www.mongodb.com`
[2] `https://studio3t.com/features/`



lcome to your trial of full free access to Studio 3T. The trial is available for 30 days. How to get a license?

Connection name:

Connection group: <root level>

Connection Type: Standard Connection

Members: localhost:27017

Enter the host name or IP address and the port:

Server: localhost Port: 27018

Use this option to import connection details from a URI

Use this option to export complete connection details to a URI

Figure 1: Creating connection to localhost MongoDB server at port number 27018

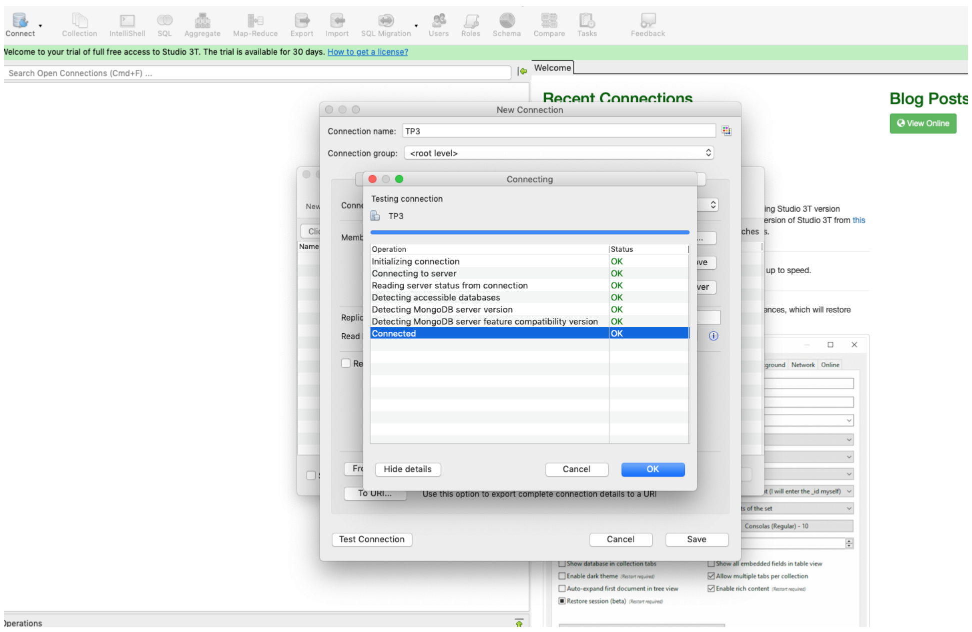
- Open another terminal to import the data `moviesEmbedded.json` (the json file is available on Edunao). using the following command.

  ```
  mongoimport --db movie --collection moviesEmbedded --drop  --jsonArray
  --file  /PATH/moviesEmbedded.json --port 27018
  ```

  Where `PATH` is the path to access to `moviesEmbedded.json` file you downloaded. The option `db` creates a database named `movie`, the option `collection` creates the collection named `moviesEmbedded` which will store the documents imported from file `moviesEmbedded.json`. The option `drop` removes the collection if it already exists.

- To query MongoDB database, we will use Studio 3T's through IntelliShell. Launch Studio 3T and create a new connection to MongoDB server by specifying its location as `localhost` and the port number 27018. Then open intellishell to edit/execute queries (see Figures 2, 1 and 3).

- Here are some useful queries: to get the list of the databases `show databases`, to access to the target database `use movie`, to show the collections located in the current database `show collections`, to count the number of documents in the target collection `db.moviesEmbedded.count()`. You can also shutdown the server (as admin) using: `use admin` and `db.shutdownServer()`



Welcome to your trial of full free access to Studio 3T. The trial is available for 30 days. How to get a license?

Connection name: TP3

Connection group: <root level>

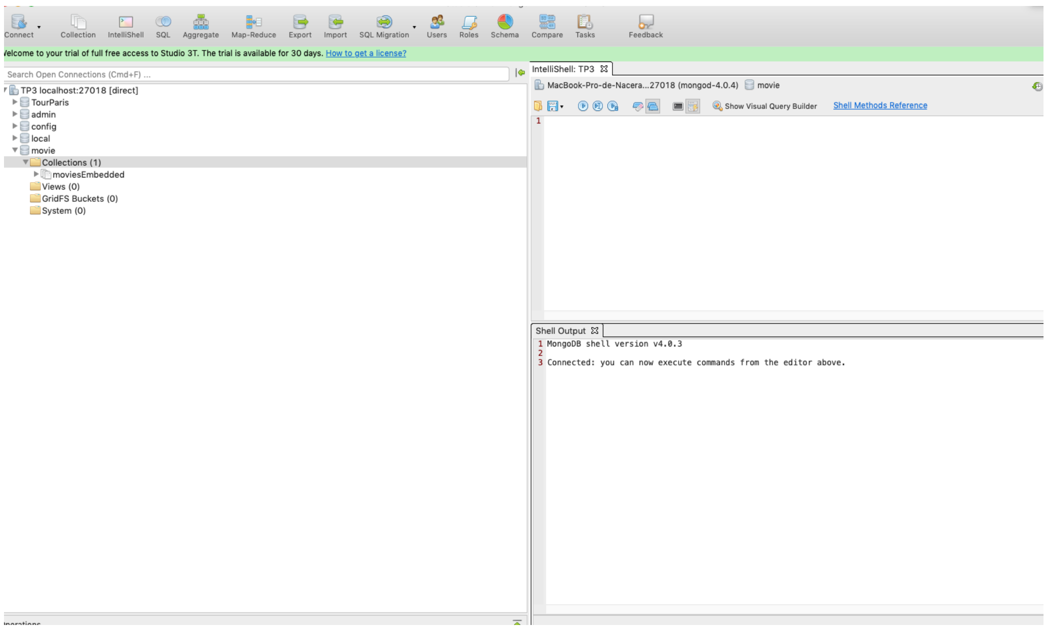
Testing connection

TP3

Operation	Status
Initializing connection	OK
Connecting to server	OK
Reading server status from connection	OK
Detecting accessible databases	OK
Detecting MongoDB server version	OK
Detecting MongoDB server feature compatibility version	OK
Connected	OK

To URI... Use this option to export complete connection details to a URI

Figure 2: Testing connection to localhost MongoDB server at port number 27018



Welcome to your trial of full free access to Studio 3T. The trial is available for 30 days. How to get a license?

```
MongoDB shell version v4.0.3

Connected: you can now execute commands from the editor above.
```

Figure 3: Studio 3T frames

2 NoSQL Movie Document Model

MongoDB is a document datastore. A MongoDB server can manage several databases, a *database* being a set of *collections*. A collection is a set of documents and each *document* has a unique identifier, referred to as `_id`. Unlike the relational databases, MongoDB allows a flexible schema, meaning that it is not necessary to define a schema before adding the data. Different data models can be represented. A document can contain another document (i.e., an *embedded* document) or reference another document by using its identifier.

The data describing the movies are represented in JSON (Javascipt Object), a lightweight data-interchange format, where a document is an ordered set of key-value pairs.

- Here is an extract of a collection of movies, where the documents describing the artists (directors and actors) are embedded in the documents.

```
[{
"_id": "movie:1",
"title": "Vertigo",
"year": 1958,
"genre": "drama",
"summary": "Scottie Ferguson, ancien inspecteur de police, ... ",
"country": "DE",
"director":  {
"_id": "artist:3",
"last_name": "Hitchcock",
"first_name": "Alfred",
"birth_date": "1899"
},
"actors": [{
"_id": "artist:15",
"first_name": "James",
"last_name": "Stewart",
"birth_date": "1908",
"role": "John Ferguson"
}, ...
{...
}]
}, ...]
```

- Here is an extract of a collection of movies, where a document describing a movie contains the references to artists (director and actors) described in external documents.

```
[{
"_id": "movie:1",
"title": "Vertigo",
"year": 1958,
"genre": "drama",
"summary": "Scottie Ferguson, ancien inspecteur de police, ... ",
"country": "DE",
"director":  {
"_id": "artist:3"
},
"actors": [{
```

```
"_id": "artist:15",
"role": "John Ferguson"
},
...
{
"_id": "artist:282",
"role": null
}]
}, ...]
```

Here is a document that describes an artist.

```
{
"_id": "artist:15",
"last_name": "Stewart",
"first_name": "James",
"birth_date": "1908"
}
```

What are the differences between the two representations of the data? Compare with the relational model (Practical Work 1).

- Differences between the two representations: possible redundancies and inconsistencies of data in embedded documents. In the second model movies and artists are two autonomous entities. The root node in the first model is movie. From query processing point of view, in the second model querying would require joins between two different collections, not suitable in a distributed environnement.
- Differences between relational model and document (json) model is normalisation.

3 Querying Movie Database

Write and execute queries to retrieve the following data:

Exercise 1 The movies titled "Gladiator".

```
db.moviesEmbedded.find({title:"Gladiator"})
```

Exercise 2 The distinct genre values of movies.

```
db.moviesEmbedded.distinct("genre")
```

Exercise 3 The movies of "crime" or "drama" genre.

```
db.moviesEmbedded.find({ $or:[ { genre:"crime" }, { genre:"drama" } ] })
db.moviesEmbedded.find({genre:{$in:["drama", '"crime"]}})
????
```

Exercise 4 The list of movies directed by "Hitchcock", display only title and year and sort them by year.

```
db.moviesEmbedded.find({"director.last_name":"Hitchcock" },
{title:1, year:1, _id:0}).sort({year:1})
```

Exercise 5 The list of movies where "Cotillard" played.

```
db.moviesEmbedded.find( { "actors.last_name": "Cotillard" } )
???
```

Exercise 6 The movies released between 1967 and 1995.

```
db.moviesEmbedded.find({year:{$gte:1967, $lte:1995}}, {title:1, year:1})
???
```

Exercise 7 The list of the movies released between 1967 and 1995, by displaying only title, year, director's last name sorted by year. The same query by adding actor's last name.

```
db.moviesEmbedded.find({year:{$gte:1967, $lte:1995}},
{title:1, year:1, _id:0, "director.last_name":1}).
sort({year:1, title:1})

db.moviesEmbedded.find({year:{$gte:1967, $lte:1995}},
{title:1, year:1, _id:0, "actors.last_name":1}).
sort({year:1, title:1})
```

Exercise 8 The number of movies by country, then by director, then by actor and then by (country, actor).

```
db.moviesEmbedded.aggregate({"$group":{_id:"$country", count:{$sum:1}}})

db.moviesEmbedded.aggregate({"$group":{_id:"$country", count:{$sum:1}}},
{$sort:{count:-1}})

db.moviesEmbedded.aggregate({"$group":{_id: {Country:"$country"}, count:{$sum:1}}},
{$sort:{count:-1}})

db.moviesEmbedded.aggregate({"$group":{_id:{country:"$country", year:"$year"},
count:{$sum:1}}})

db.moviesEmbedded.aggregate({"$group":{"_id": {country:"$country",
country:"$director.last_name"}, "count":{"$sum":1}}})

db.moviesEmbedded.aggregate({"$unwind":"$actors"},
{"$group" : {"_id": {"actor_lastName":"$actors.last_name"}, "count":{"$sum":1}}})
```

4 Working with TourPedia Database

Now, import the `moviesEmbedded.json` file available on Edunao using the following command:

```
mongoimport --db tourPedia --collection paris --drop
--file  /PATH/tourPedia_paris.json --port 27018
```

Exercise 1 Display using `find()` query the content of `paris` collection. Give an extract of a document or the tree description of a document.

Exercise 2 Give the name and the contact phone of location where the number phone is given([3])

Exercise 3 Give the name of locations whose name contains hotel([4])

Exercise 4 Give the names of locations providing a service 'chambres non-fumeurs'

Exercise 5 Give the names and services providing exactly 5 services, then at least 5 services

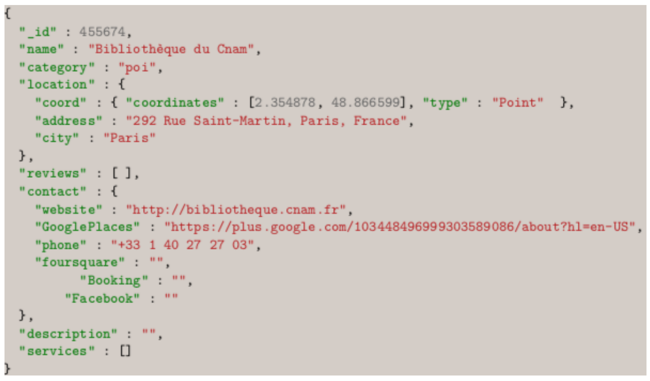
> ☛ `$size:value` is allowed only with an exact value. Use array indexes.

Exercise 6 Give the categories of locations rated at least 4

> ☛ `reviews.rating`.

Exercise 7 Give the number of locations with 'accommodation' category, providing 'blanchisserie' service, by city

Exercise 8 Give the sorted number of reviews by contact



```
{
  "_id" : 455674,
  "name" : "Bibliothèque du Cnam",
  "category" : "poi",
  "location" : {
    "coord" : { "coordinates" : [2.354878, 48.866599], "type" : "Point"  },
    "address" : "292 Rue Saint-Martin, Paris, France",
    "city" : "Paris"
  },
  "reviews" : [ ],
  "contact" : {
    "website" : "http://bibliotheque.cnam.fr",
    "GooglePlaces" : "https://plus.google.com/103448496999303589086/about?hl=en-US",
    "phone" : "+33 1 40 27 27 03",
    "foursquare" : "",
          "Booking" : "",
        "Facebook" : ""
  },
  "description" : "",
  "services" : []
}
```

[3] https://docs.mongodb.com/manual/reference/operator/query/exists/index.html

[4] https://docs.mongodb.com/manual/reference/operator/query/regex/