CentraleSupélec

Saclay Campus, Computer Science Department
2019-2020

Models and Systems for Big Data

MongoDB & Advanced Query Language

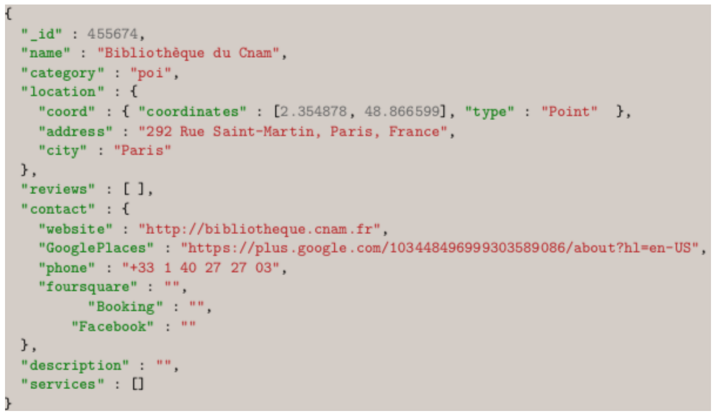
Pr. Nacéra Seghouani

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. We need to import data from `tourPedia_paris.json` file available on Edunao using the following command:

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

1 Querying Using Find and Aggregate Functions

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



```
{
  "_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" : []
}
```

Figure 1: A document example

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

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

```
db.paris.find({'contact.phone':{$exists:true}},
{name:1, 'contact.phone':1, _id:0})
```

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

```
db.paris.find({name:{ $regex: /hotel/i }},{name:1})
```

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

```
db.paris.find({services:'chambres non-fumeurs'},{name:1, services:1, _id:0})
```

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.

```
db.TourPediaParis.find({services:{$size:5}},{services:1, _id:0})
db.TourPediaParis.find({'services.0':{$exists:true},
'services.1':{$exists:true}, 'services.2':{$exists:true} },{services:1})
```

Exercise 6 Give the categories of locations rated at least 4

☛ `reviews.rating`.

```
db.paris.find({'reviews.rating':{$gte: 4}},{category:1})
```

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

```
db.paris.aggregate([
{$match:{'services' : 'blanchisserie'}},
{$group:{_id:'$location.city', total:{$sum:1}}}
]);
```

Exercise 8 Give the review sources of locations with at least one review from Facebook

```
db.paris.find({'reviews.source': 'Facebook'}, {'reviews.source':1})
```

Exercise 9 Give the distinct list of review sources

```
db.paris.aggregate([
{$unwind: "$reviews"}, {$unwind: "$reviews.source"},
{$group: {_id:"$reviews.source"}}
])
```

Exercise 10 Give the sorted number of reviews by source

```
db.tourPedia_paris.aggregate([
{$unwind: "$reviews"}, {$unwind: "$reviews.source"},
{$group: {_id:"$reviews.source", total:{$sum:1}}},
{$sort:{total:1}}
])
```

Exercise 11 Give the number of reviews by category and language

```
db.tourPedia_paris.aggregate([
{$unwind: "$reviews"}, {$unwind: "$reviews.language"},
{$group:
{_id:{category:'$category', language:'$reviews.language'},
total:{$sum:1}}},
{$sort:{total:-1}}
]);
```

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

2 Querying using Indexes

Exercise 1 `explain()` applied to `find()` show the query execution plan.

```
db.paris.find({"services" : "chambres non-fumeurs",
 "reviews.rating" : {$gte : 4}}).explain();
```

It is also possible to show the execution plan of `aggregate()` using `explain()`

```
 db.paris.aggregate([{$match:{"services" : "chambres non-fumeurs"}},
        {$group:{_id:"$type", total : { $sum : 1}}}], {explain:true});
```

`COLSCAN` means that all the column is scanned. Now, create an index on services attribute `services`.

```
db.paris.createIndex({"services":1});
```

What do you observe when you execute the previous queries ? Now, create an another index on services attribute `reviews.rating`. What do you observe when you execute the previous queries?

```
    "stage" : "COLSCAN",
    "stage" : "IXSCAN",
     "rejectedPlans" : []
```

Exercise 2 Use 2d-index MongoDB geospatial queries can interpret geometry on a flat surface or a sphere. We need to query the names and addresses of restaurants with a radius of 200 meters around :
- Eiffel Tower Paris France
- Pyramide du Louvre
- Boulevard Saint-Michel

The document structure about location coordinates is:

```
"location" : {
  "coord" : {
  "type" : "Point",
  "coordinates" : [1.53414, 42.50729 ]
  }
}
```

To achieve this kind of query we need to create an index([5], [6])

```
db.paris.createIndex( { "location.coord" : "2dsphere" } );
```

Use variables to store the coordinates of these locations and the operator `$near`

[5] https://docs.mongodb.com/manual/tutorial/query-a-2d-index/
[6] https://docs.mongodb.com/manual/geospatial-queries/

```
db.tourPedia_paris.find({
'location.coord': {$near:
{$geometry:{"type":"Point",
    "coordinates":[2.3516704899184,48.857770855496]},$maxDistance:200}
}}) ;
```

Exercise 3 Compute the average rating of restaurants located in this area

```
db.tourPedia_paris.aggregate([
{$geoNear:
{near:{"type":"Point",
    "coordinates":[2.3516704899184,48.857770855496]},"maxDistance":200,
    "distanceField" : "location.coord", "spherical":true}
},
{$match:  {"category":'restaurant'}},
{$unwind: "$reviews"}, {$unwind: "$reviews.rating"},
{$group: {_id:"$name", average:{$avg:"$reviews.rating"}}},
{$sort: {average:1}}
]) ;
```

Use `$geoNear` operator when you use `aggregate`.

3 QUYERING USING MAPREDUCE FUNCTIONS

Answer the following questions using `mapreduce()` :
Exercise 1 The number of reviews languages for reviews with rating > 4.

```
var mapFunction = function () {
  for(var i=0 ; i < this.reviews.length ; i++){
   if(this.reviews[i].rating > 4) emit(this.reviews[i].language, 1);
  }
 };
var reduceFunction = function (key, values) {
  return Array.sum(values);
};

var queryParam = {"query":{}, "out":"result_set"};
db.paris.mapReduce(mapFunction, reduceFunction, queryParam);
```

Exercise 2 The average rating of each location.
Exercise 3 The average rating of each location by category