MongoDB

NoSQL

A4- S8

ESILV
jihane.mali (at) devinci.fr

Chapter Contents

Chapter 1
Import data into MongoDB

To use MongoDB, the server has to be launched in a first command tool. The MongoDB console has to be launched in a second command tool. All the related instructions (installation, launcher, console, querying) are described in the guide available on the Moodle.

1.1 Practice Work's data set

1.1.1 Download and unzip `DBLP.zip` (available on DVL, just aside this practice work),
☒Do not take the one from ”datasets” which is a cleaned dataset.

1.1.2 Import the data file in the “TP” database and “dblp” collection (cf. Guide),
`mongoimport --db TP --collection dblp --jsonArray dblp.json`
☒Must be applied in a **shell** (not a mongo shell)

1.1.3 In the MongoDB console, connect to the `DBLP` database and count the number of stored documents.

This dataset is an automatic extraction from DBLP, a website which stores all the research publications in computer science. You have here a sample of 118k documents.

Note that some keys can be available or not (”booktitle”, ”cites”...), some others can change of data types(”pages”).

```
{
   "_id" : "books/daglib/0025185",
   "type" : "Book",
   "title" : "MongoDB - The Definitive Guide: Powerful and Scalable Data Storage.",
   "pages" : "I",
   "publisher" : "O'Reilly",
   "year" : 2010,
   "authors" : [ "Kristina Chodorow", "Michael Dirolf" ],
   "isbn" : [ "978-1-449-38156-1" ]
}
```

```
{
   "_id" : "conf/icod/BouzeghoubG83",
   "type" : "Article",
   "title" : "The Design of an Expert System for Database Design.",
   "booktitle" : "ICOD-2 Workshop on New Applications of Data Bases",
   "pages" : { "start" : 203, "end" : 223 },
   "year" : 1983,
   "url" : "db/conf/icod/icod83w.html#BouzeghoubG83",
   "authors" : [ "Mokrane Bouzeghoub", "Georges Gardarin" ],
   "cites" : ["journals/tods/AstrahanBCEGGKLMMPTWW76", "books/bc/AtzeniA93",
   "journals/tcs/AtzeniABM82", "journals/jcss/AbiteboulB86"]
}
```

You can find the following keys:

- Conference and scientific journal are of ”type” “Article”, with a ”title” and sometimes ”booktitle”,
- Book chapters are of ”type” ”Book” with a title and booktitle,
- PhD thesis are of ”type” : ”Phd”,
- Every publications are linked to a list of authors and sometimes publisher (also called editor),
- Some of them can embed a ”cites” key with a list of references

Chapter 2 Querying MongoDB collections

2.1 MongoDB *simple* queries

For each of the following sentence, give the corresponding query in the MongoDB API:

2.1.1 All the book titles (type 'Book'),
Correction :

```
db.dblp.find({"type" : "Book"});
```

2.1.2 All the publications since 2011,
Correction :

```
db.dblp.find({"year" : {"$gte" : 2011}});
```

2.1.3 All the books since 2014,
Correction :

```
db.dblp.find({"type" : "Book", "year" : {"$gte" : 2014}});
```

2.1.4 Titles of publications with an existing publisher,
Correction :

```
db.dblp.find({"publisher" : {$exists:1}}, {"title":1});
```

2.1.5 Titles of *Jeffrey D. Ullman*'s publications,
Correction :

```
db.dblp.find({"authors" : "Jeffrey D. Ullman"}, {"title" : 1});
```

2.1.6 Give those in which he is the first author,
Correction :

```
db.dblp.find({"authors.0" : "Jeffrey D. Ullman"}, {title:1});
```

2.1.7 And when he is the only author,
Correction :

```
db.dblp.find({"authors" : ["Jeffrey D. Ullman"]}, {title:1, _id:0});
```

2.1.8 List of all publications which title contains the word "database".
Correction : "i" for case insensitive

```
db.dblp.find({"title" : {$regex: /database/, $options: 'i'}});
db.dblp.find({"title" : /database/i});
```

2.2 Distinct queries

2.2.1 List of distinct publishers,
Correction :

```
db.dblp.distinct("publisher");
```

2.2.2 List of distinct authors.
Correction :

```
db.dblp.distinct("authors");
```

2.3 Aggregates

2.3.1 *Complex* queries

2.3.1 All *Jeffrey D. Ullman*'s publications, projected the title and sorted by the starting page,
Correction :

```
db.dblp.aggregate([{"$match":{"authors" : "Jeffrey D. Ullman"}},
      { "$sort" : { "pages.start" : 1 } },
      {"$project" : {"title" : 1, "pages" : 1}}]);
```

2.3.2 Count how much publications he published per year,
Correction :

```
db.dblp.aggregate([{"$match":{"authors" : "Jeffrey D. Ullman"}},
      {"$group":{"_id":"$year", "total" : { "$sum" : 1}}}]);
```

2.3.3 Count how much publications he published,
Correction :

```
db.dblp.aggregate([{"$match":{"authors" : "Jeffrey D. Ullman"}},
      {"$group":{"_id":null, "total" : { "$sum" : 1}}}]);
```

2.3.2 *Hard* queries

2.3.4 For each author, count the number of its publications. Sort the result in descending order,
Correction :

```
db.dblp.aggregate([
    { "$unwind" : "$authors" },
    { "$group" : { _id : "$authors", "number" : { "$sum" : 1 } }},
    { "$sort" : {"number" : -1}}
] );
```

2.3.5 For each publisher (if exists) and year, give the number of publications
Correction :

```
var matchpublisher = {$match:{"publisher" : {$exists : 1}}};
var groupYearPub = {$group:{
   "_id":{year : "$year", "publisher" : "$publisher"},
   "nb" : {$sum : 1}
 }};
db.dblp.aggregate([
 matchpublisher,
 groupYearPub ]);
```

2.3.6 For each publisher (if exists), give the average rate of publications per year. Sort the result decreasingly.

Correction :

```
var groupAvgPub = {
 $group:{"_id":"$_id.publisher",
 "average" : { $avg : "$nb"}
}};
db.dblp.aggregate([
 matchpublisher,
 groupYearPub,
 groupAvgPub,
 {$sort:{average:-1}}
]);
```

Chapter 3 MongoDB Updates

Store documents can be updated in the database. You have to use the MongoDB API to query the required documents.

update function: 1st param is for mapping, 2nd param is for setting ($set, $unset).
delete function: 1st param is for mapping.

3.1 Examples

Give a sentence corresponding to the following update queries:

-
  ```
  db.dblp.update ( { "authors" : "Jeffrey D. Ullman" },
        {$set : { "label" : "Ullman" } } );
  ```

  Correction : For each publications in which Jeffrey D. Ullman is one of the authors, add (or update) the label "Ullman".

  By default, only the first document is updated (for scalability purposes). To update all documents, add a new document at the end containing {"multi":"true"}

  ```
  db.dblp.updateMany ( { "authors" : "Jeffrey D. Ullman" },
        {$set : { "label" : "Ullman" } });
  ```

-
  ```
  db.dblp.deleteMany(
        { "title" : "Parallel, Distributed and Multiagent Production Systems" } )
  ```
  Correction : Delete all publications which title is "Parallel, Distributed and Multiagent Production Systems"

3.2 Update queries

For each of the following sentence, query the updating query. Verify the targeted information before the update.

3.2.1 Update all books containing "database" by adding the label attribute "database",
Correction :

```
db.dblp.updateMany({ "title" : /database/i, "type":"Book" },
     {$set : { "label" : "database" } });
```

3.2.2 Update all publications with a "*publisher*" containing "ACM" by removing the "pages" attribute,
Correction :

```
db.dblp.updateMany( { "publisher" : /ACM/i },
     {$unset : { "journal.volume" : "" } } );
```

3.2.3 Remove all publications without any authors,
Correction :

```
db.dblp.deleteMany( { "authors" : {$size: 0} });

db.dblp.deleteMany( { "authors.0" : {$exists: 0} });
```

3.2.4 Update all publications to add a "pp" field which corresponding to the number of pages of the publication.
Correction : What we want to process is a JSON document, but it does not work properly.

```
db.dblp.updateMany( {"pages.start" : {$exists : true}, "pages.end" : {$exists : true}} ,
     {$set : { "pp" : {$substract : ["$pages.end", "$pages.start"] } } } );
```

For this, we need a ”forEach” function.

```
db.dblp.find(
   {"pages.start": { $exists: true },"pages.end": { $exists: true}}).forEach(
      function (pub) {
         pub.pp = pub.pages.end - pub.pages.start + 1;
         db.dblp.updateMany(
            { _id: pub._id },
            { $set: { pp: pub.pp }});
      }
   );
```

Chapter 4

BONUS: Indexing in MongoDB

Indexing attributs enables fast querying on this attribute.

4.1 Indexing single attributes

4.1.1 For the "Jeffrey D. Ullman" query, generates the execution plan by adding "`.explain()`",
Correction :

```
db.dblp.find({"authors" : "Jeffrey D. Ullman"}).explain();
```

4.1.2 Add an index on authors: "`db.dblp.createIndex( { "authors" :1 } );`",

4.1.3 Repeat the query with the explain plan. You will obtain a new one which uses the created index (more efficient).

4.2 2Dsphere indexes (complex/hard queries)

The 2DSphere index can index 2 dimensions like latitudes and longitudes, the concept is explained here: `http://docs.mongodb.org/manual/applications/geospatial-indexes/`.

4.2.1 Dataset

- Download the `cities.json.zip` file (section MongoDB on DVO), unzip it.
- To import this file in a `cities` collection, use the `mongoimport` command. The coordinates must have this schema: `"location" : {"type": "Point", "coordinates" : [XXXXX, YYYYY]}`
  Correction : mongoimport –db TP –collection cities cities.json
- Index the coordinates with: `db.cities.ensureIndex( { location : "2dsphere" } );`
- The documentation for querying in 2D is available here:
  `http://docs.mongodb.org/manual/tutorial/query-a-2d-index/`

4.2.2 2D queries

4.2.1 Give the coordinates of *Paris*, *Lyon* and *Bordeaux*,
Correction :

```
db.cities.find({"name":{$in:["Paris","Lyon","Bordeaux"]}},
         {"name":1,"location.coordinates":1,"_id":0});
>>>>>
{ "name" : "Paris", "location" : { "coordinates" : [ 2.3488, 48.85341 ] } }
{ "name" : "Lyon", "location" : { "coordinates" : [ 4.84671, 45.74846 ] } }
{ "name" : "Bordeaux", "location" : { "coordinates" : [ -0.5805, 44.84044 ] } }

var paris = [ 2.3488, 48.85341 ];
var lyon = [ 4.84671, 45.74846 ];
var bordeaux = [ -0.5805, 44.84044 ];
```

4.2.2 Give the number of cities near *Paris* (less than 100 km). Use the $near operator,
Correction :

```
var dist = 100000;
db.cities.find({location:{$near : {$geometry :
   { type : "Point", coordinates : paris}, $maxDistance : dist}}}).count();

>>> 229
```

4.2.3 Sum the amount of population is this area,
Correction :

```
db.cities.aggregate([
     {"$geoNear": {
       "near":{"type":"Point", "coordinates" : paris},"maxDistance":dist,
       "distanceField" : "outputDistance", "spherical":true}},
     {"$group":{"_id":null, "total" : { "$sum" : "$population"}}}]);
```

4.2.4 Find the number of cities within the triangle *Paris-Lyon-Bordeaux*. Use the `$geoWithin` operator,
Correction :

```
var triangle = [[paris, lyon, bordeaux, paris]];
db.cities.find({location:{$geoWithin : {$geometry :
  { type : "Polygon", coordinates :
  triangle},}}}).count();
>>> 75
```

4.2.5 Find the name of cities which have more than 100 000 inhabitants in this zone.
Correction :

```
db.cities.find({location:{$geoWithin : {$geometry :
  { type : "Polygon", coordinates :
  triangle},}},
  "population":{$gte:100000}},{name:1, population:1, _id:0});
>>>>
{ "name" : "Bordeaux" }
{ "name" : "Orléans" }
{ "name" : "Paris" }
{ "name" : "Lyon" }
{ "name" : "Clermont-Ferrand" }
{ "name" : "Limoges" }
```

Chapter 5
BONUS: Map/Reduce - really hard "queries"

5.1 Map

For each of the following sentence, produce a map and a reduce function. Execute it and show the content of the output collection ('`result_set`').

5.1.1 Apply this Map/Reduce function for practice. Give the sentence corresponding to this query.

```
var mapFunction = function () {
 if(this.type == "Book")
   emit(this.title, 1);
};
var reduceFunction = function (key, values) {
 return Array.sum(values);
};
var queryParam = {query:{}, out:"result_set"};
db.dblp.mapReduce(mapFunction, reduceFunction, queryParam);
db.result_set.find();
```

Correction : For each "Book" document, give the number of occurrences of titles.

5.1.2 Gives the number of publications with more than 3 authors,
Correction :

```
var mapFunction = function () {if(this.authors.length >3)emit(null, 1);};
var reduceFunction = function (key, values) {return Array.sum(values);};
```

5.1.3 For each book containing chapters (attribute booktitle enabled) published by *Springer*, count the number chapters and show only those containing at least two ones[1],

Correction :

```
var mapFunction = function () {if(this.publisher=="Springer" && this.booktitle)
 emit(this.booktitle, 1);};
var reduceFunction = function (key, values) {return Array.sum(values);};
var queryParam = {query:{}, out:"result_set"};
db.dblp.mapReduce(mapFunction, reduceFunction, queryParam);
db.result_set.find({value:{$gte:2}});
```

5.1.4 For publications published by *Springer*, gives the number of publications per year,
Correction :

```
var mapFunction = function () {if(this.publisher == "Springer")emit(this.year, 1);};
var reduceFunction = function (key, values) {return Array.sum(values);};
```

5.1.5 For each key[2] "publisher & year" (for those with a publisher), gives the number of publications,

Correction :

[1]Warning! On MongoDB, the reduce function is applied only if at least two *'emits'* are produced by the map function. To apply the required resultset, you need to query the collection *'result_set'* with the appropriate query
[2]Warning! the key of an emit must be a single value or a document

```
var mapFunction = function () {if(this.publisher)
                emit({publisher:this.publisher, year:this.year}, 1);};
var reduceFunction = function (key, values) {return Array.sum(values);};
```

5.1.6 For *Jeffrey D. Ullman*'s publications, gives the number of publications per year,
For this query, you must use the option "query" in the "queryParam" to filter the authors.
Correction :

```
var mapFunction = function () {emit(this.year, 1);};
var reduceFunction = function (key, values) {return Array.sum(values);};
var queryParam = {query:{authors:"Jeffrey D. Ullman"}, out:"result_set"};

Without queryParam filter:
var mapFunction = function () {if(Array.contains(this.authors, "Jeffrey D. Ullman"))
                emit(this.year, 1);};
var reduceFunction = function (key, values) {return Array.sum(values);};
var queryParam = {query:{}, out:"result_set"};
```

5.1.7 For *Jeffrey D. Ullman*'s publications, gives the average number of pages (only for those where the information is provided),
Correction :

```
var mapFunction = function () {emit(null, this.pages.end - this.pages.start);};
var reduceFunction = function (key, values) {return Array.avg(values);};
var queryParam = {query:{authors:"Jeffrey D. Ullman"}, out:"result_set"};
```

5.1.8 For each author, gives a list of all his publications' titles[3],
Correction :

```
var mapFunction = function () {
   for(var i=0;i<this.authors.length;i++)
      emit(this.authors[i], this.title);};
var reduceFunction = function (key, values) {return {titles : values};};
```

5.1.9 For each author, give for each year the number of publications,
Correction :

```
var mapFunction = function () {
   for(var i=0;i<this.authors.length;i++)
      emit({author : this.authors[i], year : this.year}, 1);};
var reduceFunction = function (key, values) {return Array.sum(values);};
```

5.2 Reduce

5.2.1 For the publisher "Springer", gives for each year the distinct number of authors,
Correction :

[3]Warning! An array is not a proper ouput value for map and reduce

```
var mapFunction = function () {
   if(this.authors)
      emit(this.year, {nb : this.authors.length, authors_d : this.authors});};
var reduceFunction = function (key, values) {
    var distinct = 0;
    var v;
    var authors = new Array();
    for(var i=0;i<values.length;i++){
        v = values[i].authors_d;
        for(var j=0;j<v.length;j++){
            if(!Array.contains(authors, v[j]) && v[j])
                {authors[authors.length] = v[j];}
        }
    }
return {nb : authors.length, authors_d : authors};};
```

5.2.2 For each publisher, gives the average number of pages per publication,
Correction :

```
var mapFunction = function () {
    if(this.pages && this.pages.end)
        emit(this.publisher, this.pages.end - this.pages.start);};
var reduceFunction = function (key, values) {return Array.avg(values);};
var queryParam = {query:{}, out:"result_set"};
```

5.2.3 For each author, gives the first and last publishing year and the total number of publications,
Correction :

```
var mapFunction = function () {
    for(var i=0;i<this.authors.length;i++)
        emit(this.authors[i], {min : this.year, max : this.year, number : 1});};
var reduceFunction = function (key, values) {
    var v_min = 1000000;var v_max = 0;var v_number = 0;
    for(var i=0;i<values.length;i++){
        if(values[i].min < v_min)v_min = values[i].min;
        if(values[i].max > v_max)v_max = values[i].max;
        v_number++;
    }
return {min:v_min, max:v_max, number:v_number};};
```

5.2.4 For each book title, return the number of distinct authors[4],
Correction :

```
var mapFunction = function () {if(this.type == "Book")
  for (var i=0;i<this.authors.length;i++)
    emit(this.booktitle, this.authors[i]);};
var reduceFunction = function (key, values) {
  var authors = values[0];
  for(var i=1;i<values.length;i++)
    if(!Array.contains(authors, values[i]))
      authors[authors.length] = values[i];
  return authors.length;};
var queryParam = {query:{}, out:"result_set"};
```

[4]Several documents can have the same booktitle