DIT032 – Data Management

Assignment 3 – SOLUTIONS

University of Gothenburg | Deadline: **Tue 06.03.2018** 23:55

General Remarks

- There are in total 100 points to be reached for this assignment, and you need at least 70 points to pass the assignment. **Note that you need to pass all assignments individually.** It is not possible to "carry over" some points from this assignment to the next ones.
- Assignment solutions are to be produced in teams of two students. Discussions with other colleagues (e.g., through the forum in GUL) are encouraged, but don't share your assignment solutions, or significant parts, with your colleagues (neither through the forum nor through other means). If we find that multiple groups submitted the same assignments, we reserve the right to fail **all** involved groups.
- Remember to also hand in self- and peer assessment forms. You can use the forms available separately on GUL. Please contact us if you have any questions regarding this part. Assessment forms can either be scanned and attached to the submission (sign first!), handed in during the lecture or supervision, or brought in person to the office hour.
- We offer two teaching assistant supervision hours (Wednesday and Friday, 13:15 – 15:00), where you can talk to teaching assistants, get feedback on your assignment work, and ask questions.
- **No** deadline extensions are given. Start early and, after finishing your assignment, upload your submission as a **single ZIP file** in GUL. The submission system closes automatically, and we do not accept late submissions. If you can't make it in time, you will need to submit during the first re-submission date in May.
- Shortly after the deadline, we grade the assignment and send you feedback. If you fail the assignment, you have the possibility to submit an improved version of the assignment for re-grading in May or October (see the website for details). We only re-grade failed submissions. If you have detailed questions it is better to ask them in the forum or during the supervision.

1 Setup MongoDB (10 Points)

As a first step, we again need to install some more software. Points are again awarded for successfully running the sanity check query at the end.

INST1 Download and install the MongoDB 3.6 community server (earlier versions before 3.6 won't work!). Follow the detailed installation instructions for your OS here:
https://github.com/joe4dev/dit032-setup

INST2 Import the Garden and Plant JSON documents (available from GUL).
Find some MongoDB Getting Started resources here:
https://github.com/joe4dev/dit032-setup/blob/master/MongoDB.md
To import, run the the following commands in your shell:

```
# Mac:
mongoimport -d plants -c plants plants.json
mongoimport -d plants -c gardens gardens.json
# Windows:
mongoimport /d plants /c plants plants.json
mongoimport /d plants /c gardens gardens.json
```

This creates a new database "plants" with two collections ("plants" and "gardens"), and imports the JSON documents from the file into them.

INST3 Validate that the new database has been created by running `show dbs` again in the mongo shell. Switch to the "plants" database with

```
use plants
```

and report the results of the following query with a screenshot:

```
db.plants.count()
db.gardens.count()
```

(counts how many documents there are in the "plants" and "gardens" collections).

Solution

10 and 3

2 Querying MongoDB and Updating Data (30 points)

Write the following queries against the plants database with the *plants* and *garden* collections you imported. Use the query-by-example API (i.e., `db.<collection>.find()` and its variants[1]) in the mongo shell. You will need to refer to the documentation of MongoDB to assemble the queries (the few example statements covered in the lecture will not be sufficient to solve these tasks).

MGDB1 Find all plants.

Solution

```
db.plants.find()
```

MGDB2 Find all gardens.

Solution

```
db.gardens.find()
```

MGDB3 Find and return all plant _ids.

Solution

```
db.plants.find({}, {_id: true})
```

MGDB4 Find all roses (i.e., all plants of type rose).

Solution

```
db.plants.find({"plant.Type":"Rose"})
```

Note that `db.plants.find({plant:{"Type":"Rose"}})` won't work, because that only matches exact sub-documents.

MGDB5 Find all English names of plants. Return only the English name, and no other attributes (e.g., no "_id" field). The list does not have to be unique. This query should return a JSON document with the following structure for every plant: `{ "plant" : { "English_name" : "Maple" } }`

Solution

```
db.plants.find({}, {"plant.English_name" : true, "_id" : false} )
```

MGDB6 Using the `db.<collection>.update()` operation, change the name of the garden originally called "Orchid Garden" to "Gothenburg Orchid Garden".

Solution

```
db.gardens.update({"garden.name" : "Orchid Garden" }, {$set : {"garden.name"
: "Gothenburg Orchid Garden" }})
```

MGDB7 Find all gardens with exactly two roses in them.

Solution

```
db.gardens.find({"garden.plants.roses": { $size : 2 }})
```

MGDB8 Find all gardens with more than a single tree in them.

Solution

```
db.gardens.find({"garden.plants.trees.1": {$exists: true}})
```

[1] https://docs.mongodb.com/manual/reference/method/db.collection.find/

MGDB9 Find all orchids whose petals are more than 0.25 cm^2 in area. You need to use the `$expr` operator[2] to solve this query.

Solution

```
db.plants.find( { $expr:  { $and :  [{$gt :  [{$multiply:["$plant.Petal_size.Width",
"$plant.Petal_size.Height"]}, 0.25]}, {$eq :  ["$plant.Type", "Orchid"] }]}})
```

MGDB10 Find the name of the garden(s) where plants with the English name "Maple" grow. You need to use the aggregation pipeline[3] to solve this query.

Solution

```
db.gardens.aggregate([{$unwind:  "$garden.plants.trees"}, {$lookup :  { from:
"plants", localField:  "garden.plants.trees", foreignField:  "plant.Plant_code",
as:  "p" }},{$match :  {"p.plant.English_name" :  "Maple" }}, { $project :  {
"garden.name" :true, "_id":false} } ])
```

2.1 Submission Information

Create a simple text file called `queries.js` containing all queries. Use comments in the style `// MGDB<number>` to mark each query.

```
// MGDB11
db.gardens.findOne()

// MGDB12
db.gardens.distinct("garden")
...
```

2.2 Tips

- Both, the official documentation[4] and TutorialsPoint[5] have a lot of easy-to-follow examples for interacting with MongoDB and the mongo shell. Use these to get started.
- If you want to check out what the data in the database looks like, you can just open the files "plants.json" and "gardens.json" (which you imported in the first step) using any plain text editor or IDE.
- If you have a hard time with the commandline client of MongoDB you can also check out Robo 3T[6], a GUI for MongoDB.
- For this part of the assignment you will need to use the operator reference list[7].

[2] https://docs.mongodb.com/manual/reference/operator/query/expr/index.html
[3] https://docs.mongodb.com/manual/reference/operator/aggregation/
[4] https://docs.mongodb.com/manual/
[5] https://www.tutorialspoint.com/mongodb/index.htm
[6] https://robomongo.org
[7] https://docs.mongodb.com/manual/reference/operator/

3 Reflective Questions (60 points)

Provide answers to the following questions *in your own words*. Reflect over the entire course. Submit a PDF document with your answers to the questions. Answer concisely, but cover what's important - one sentence will be too little, but more than three paragraphs (and maybe a figure) will be too much. You do not need to cite sources, but plagiarism (i.e., copying another group or copying some other existing text) will lead to an immediate fail of the assignment (independently of how many points you reach otherwise).

RQ1 Early in lecture, we discussed the difference between data, information, and knowledge. Summarize and relate these concepts. Use as the Mondial database used in Assignment 1 and 2 to explain what would be data, information, and knowedge.

RQ2 In the course we learned about two types of databases – relational ones (SQL) and non-relational ones (NoSQL, e.g., MongoDB). What are the advantages and disadvantages of both models? Which kinds of applications would you say are both models suitable for?

RQ3 What is the difference between structured, unstructured, and semi-structured data? Think back to the different data representation formats we have seen in the course (e.g., the relational model, JSON, and to a lesser extent XML or CSV) and classify them.

RQ4 Briefly explain how the Map/Reduce model works. How is it different from executing a "normal" query in SQL or MongoDB? What is the advantage of Map/Reduce?