Cours Advanced Database – MSE HES-SO

Laboratory 2 – Diving deeper with Neo4j

This laboratory exercise aims at showing databases from a performance and operation point of view. It is again based on Neo4j, and will build on the knowledge acquired during the first laboratory.

For this laboratory, it is allowed to create teams of two persons. But you're allowed to work on your own as well.

Essentially, this laboratory consists of loading in neo4j a relatively big graph: the dblp article data. DBLP was an original effort led by University Trier in Germany to create a map of scientific contributions, and this way before scholar.google.com emerged as the leading tool to track papers and citations.

The DBLP dataset is openly available here: https://www.aminer.org/citation. We will consider the latest dataset (v13) which is available for download here: https://originalstatic.aminer.cn/misc/dblp.v13.7z

Mind that the compressed file is 2.5 GB big, and once uncompressed, the resulting JSON file is nearly 18GB big. This is a big file to deal with. Your goal is to exploit this data to create in Neo4j a graph with more than 5 millions of nodes.

But this is not all. We also ask you to achieve this goal

1. without scaling-up too much the machine loading the data…
2. In a decent time…

Preliminary instructions:

We expect you to deliver a git repository containing the following elements:

- A shell script called "build.sh" in charge of building a docker image (locally) containing your code, ready to load the data into neo4j.
- A docker-compose.yaml descriptor, describing how to launch two containers, one offering neo4j, and the other one running your loading application.

DO NOT COMMIT the big JSON file into the GIT repository. **Anyone doing this will automatically loose the bonus associated with this lab.**

To save you time, an example repository with some of the boilerplate is provided:

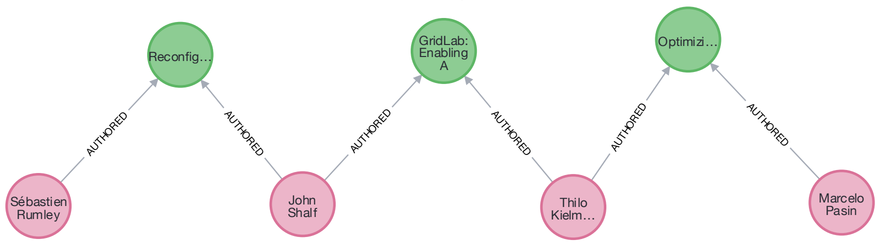
https://github.com/sebastienrumley/mse-advDaBa-2022.git

You can either fork it, or download the content and start a new repo from scratch. Make sure you grand access to the repo to the teachers.

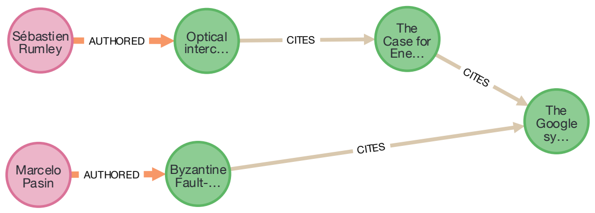
The example java code starts the Neo4j driver, and reads the 100 first lines of an example JSON file.

Graph structure

At the end we would like something like that:



We want to be able to find thru which co-authors one can relate an author to another.

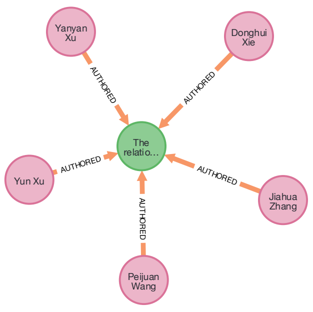


We also want to be able to navigate thru citations. Marcelo and Sébastien have apparently never cited the same paper, but by relaxing the constraint to citations of citations, one can find a "root citation".

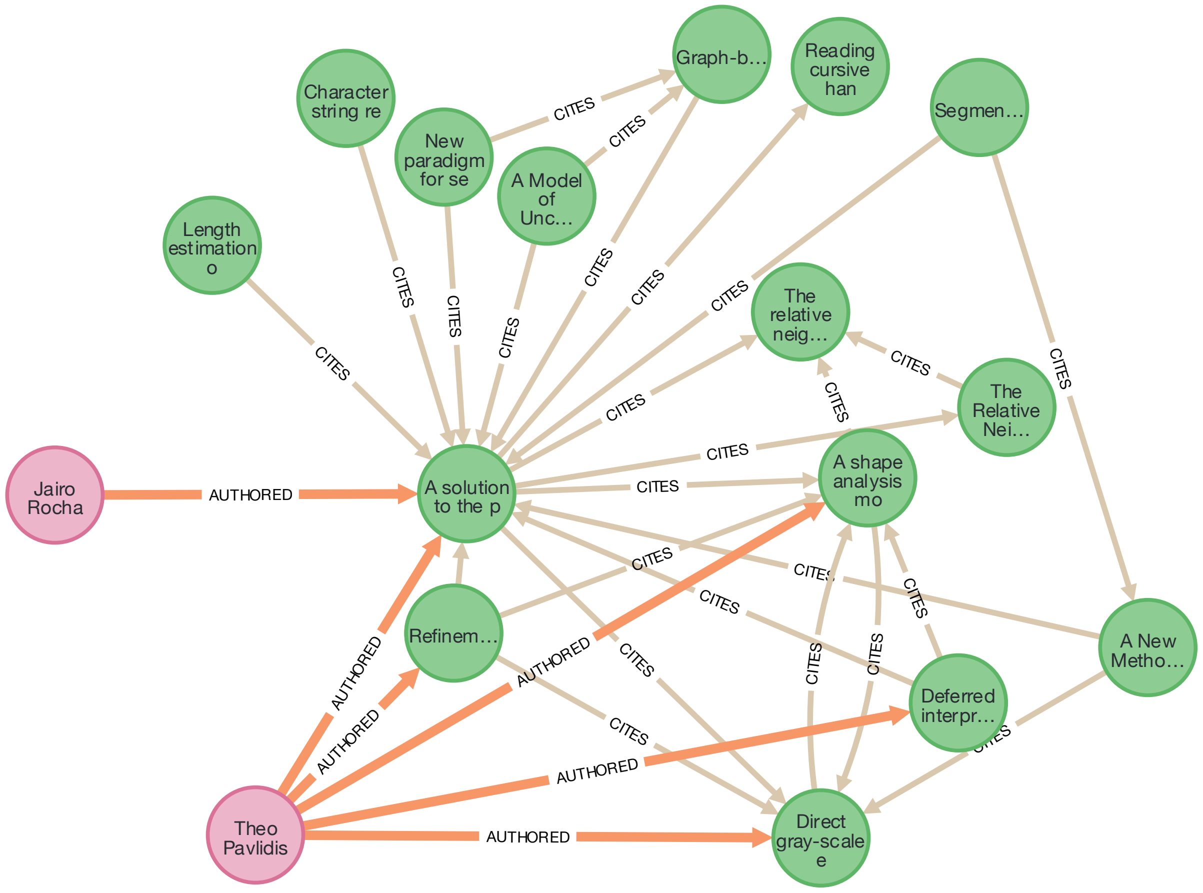
So we will build the graph as follows:

- ARTICLE nodes
- AUTHOR nodes
- CITES relationships
- AUTHORED relationships
- Each node (AUTHOR and ARTICLE) with have a key "_id" that corresponds to a key in the JSON file. For instance the _id of the first article is *"_id" : "53e99784b7602d9701f3e3f5"*.
- ARTICLE nodes will also have a title property. For the first instance of article in the JSON, the title is *"title" : "3GIO."*
- AUTHOR nodes will have a name property. For the second instance of article in the JSON (the first one has no authors), the first author is *"name" : "Peijuan Wang"*
- AUTHOR node will be connected to the ARTICLE nodes with the AUTHORED relationship.
- ARTICLE nodes will be related to other ARTICLE nodes they cite with the CITE relationship.

Below the graph resulting from the second article in the file. This article has no references and is not cited.



Below the neighbors of the third article in the JSON. It has two authors, a few references (5) and has collected 8 citations.



Docker limitations

As you can see in the docker-compose file, we want the RAM of the neo4j container to be restricted. In the default configuration, the limit is 3g. You can raise it to 4g but not more.

If you want to test without docker-compose, here is more or less the configuration:

```
docker run  -it --name advDBlab2  -p7474:7474 -p7687:7687
            -v $PWD/neo4j_mount/data:/data
            -v $PWD/neo4j_mount/conf:/conf
           -v $PWD/neo4j_mount/logs:/logs
            --memory="3g"  --env NEO4J_AUTH=neo4j/test  neo4j:4.4.15-community
```

Final instructions

You are totally free to choose any programming language to load the graph into the DB. The only requirement is that it must be possible to pack it as a docker image, as done in the example. Later on, it should be possible to load the graph using the “docker-compose up” command.
Mind the MAX_NODES=10000 variable in the docker-compose descriptor. It is highly recommended to adopt a mechanism to set a limit to the number of nodes that should be

loaded. In this way, one can test your code with a subset of the dblp graph. In the branch/tag you submit as your solution, make sure you set the parameter to a value your code supports. **We will not test your solution with higher values.**

Next to the “functional requirements” described above, we also ask you to write a very short report (it can be a README.md):

- Describing your approach for loading the graph
- Listing the parameter values you picked
- Reporting at least one result of a performance test.

A performance test result is a triplet {“number_of_articles”=XX, “memoryMB”=”3000”, “seconds”=”YY”}

The repositories will be tested and the codes assessed during the winter break, so the deadline is December 25th, midnight.

Good luck!