Lab-2: AuraDB and Neo4j GDS

Madhulika Mohanty[1]

December 19, 2024

Submission deadline: Wednesday, January 8, 2025 at 23:55:00 (Paris time).

1 Lab organization

In today's lab and its assignment, we will set up an AuraDB instance (a cloud version of Neo4j) and write a Python client to query this DB. The dataset to be used for this part will be small so as to fit within the trial version of the software. In the second part of the lab, we will use the ICIJ dataset (previously loaded for Lab-1) and explore the Graph Data Science features provided by Neo4j.

1.1 AuraDB

AuraDB is a cloud based service for Neo4j. It differs from the desktop version in the fact that the management of the server is undertaken by the cloud provider. It is supported by many platforms like Google Cloud, Microsoft Azure and AWS. Visit `https://neo4j.com/product/auradb/` and create a Free account on the cloud using Google Cloud.

You can find many resources on using AuraDB here: `https://neo4j.com/docs/aura/classic/auradb/` and with Python here: `https://neo4j.com/docs/python-manual/current/`. We will create a database instance and subsequently load data into it. We will be using Python programming to build an application using this DB.

Assignment and questions

We will work with a tiny StarWars dataset. Please download the `Lab 2 datasets` zip file from Moodle. The archive contains ten CSV files[2]. The files having nodes contain different types of nodes in the graph, each line in a file contains a node ID with the properties of this node. Similarly, each file containing relationships has one relationship on each line with its source and target node ids. The archive also has a JSON file providing the graph model to be used during import. You will see that it essentially contains the Schema of the dataset. On this dataset, using the tutorials above, you need to solve the following tasks.

Task 0: setup

1. Load this dataset into the newly created instance using the Import feature of AuraDB by providing the model with data (zip) file and running the import.
2. Launch the Explore tool of the UI and explore the dataset. What does the dataset contain?
3. Launch the Query tool of the UI. Write a Cypher query to find the number of nodes in the graph.
4. Write a Python program to be run locally that connects to this remote database server. Verify the connection. Using this connection, write the above Cypher query and fetch the number of nodes using Python.

Task 1: building an application

Develop a menu-driven Python application to take user inputs to perform the following:

1. Given the name of a character, return their home world.
2. Given a speed, return all the Starships that can fly at that speed.

[1]madhulika.mohanty@inria.fr
[2]CSV stands for Comma Separated Values.

3. Given a species, return all the planets hosting them.

4. Given two characters, find the shortest path between them.

5. For each planet, retrieve its statistics: the number of occupants, the number of species and the number of films involving them. Finally, display the most populous planet with all its habitants.

NOTE: Do not forget to stop your instance once your assignment is over. This may not cost money as we are using a free tier but unless stopped explicitly, the server will keep running and waste resources.

1.2 Neo4j Graph Data Science (GDS)

Along with the ability to query a graph dataset using Cypher, Neo4j also provides many graph data science operations over the graph. This feature needs to be enabled explicitly for each database on the server (local or remote). To be able to use a large dataset and stay within the free tier, we will use this feature on our local systems. You can find detailed documentation and tutorials here: `https://neo4j.com/docs/graph-data-science/current/` and here: `https://neo4j.com/docs/graph-data-science-client/current/`.

We will work with the ICIJ dataset introduced in Lab-1.

1. Launch the locally installed Neo4j Desktop version (used for Lab-1). (If you no longer remember the password or have dropped the database, reload it using the instructions from Lab-1.)

2. Select the ICIJ project previously used for Lab-1 and enable the *Graph Data Science* library from the list of plugins available for the database (`https://neo4j.com/docs/graph-data-science/current/installation/neo4j-desktop/`).

Assignment and questions

Task 0: setup

1. As opposed to Cypher queries, GDS library requires loading a projected graph in-memory in to a Graph object. There are many ways to create this object (`https://neo4j.com/docs/graph-data-science/current/management-ops/graph-creation/`). We will be using a Cypher query on a Neo4j Browser (cypher-shell) (`https://neo4j.com/docs/graph-data-science/current/management-ops/graph-creation/graph-project-cypher-projection/`). Write such a query to project all the nodes but only the "officer_of", "intermediary_of", "registered_address" and "connected_to" relationships.

2. Following instructions from `https://neo4j.com/docs/graph-data-science-client/current/getting-started/`, write a Python program to be run locally that connects to this database server for graph data science operations. Print the library version.

Task 1: computing centrality measures in the graph

Each algorithm in the GDS library can be called directly from Cypher using Neo4j cypher-shell, or from a client code using a Neo4j Driver in the language of choice. We will be using Python for running our algorithm. Further, multiple execution modes are supported depending on whether (and where) the results are to be persisted (`https://neo4j.com/docs/graph-data-science/current/common-usage/running-algos/`). We will explore two modes.

Write a Python program that performs the following (in exactly this order):

1. Load the projected graph into a Python object and print its number of nodes and number of relationships.

2. Compute (but not store) the pagerank for all the nodes using the default parameters.

3. Did the algorithm converge? If not, retry using more iterations until convergence and report the number of iterations. Finally, save the computed values in the projected graph. Verify that the node properties have indeed been modified by printing the list of node properties in the projected graph. What other properties do you see and why?

4. Write the pagerank values to the database. Using a Cypher query (within Python), retrieve the top-20 highest pagerank valued nodes and display them.

2 Submission Guidelines

Please read the assignment description carefully. If you have any questions about the lab or on the course material, do not hesitate to ask them during the lab session, via Moodle or over email. Late submission penalties apply as for Lab-1. The timestamp on Moodle will be counted as the submission time so submit well before in advance.

You should turn in a zip file named "LASTNAME_FIRSTNAME.zip" for your solutions to the tasks. It should on unzipping give the following elements:

1. A PDF report: include here the setup information with screenshots, explain **the complete list of steps or queries** you used, as well as **all the results/output** (you may provide an excerpt if it is too big).

2. A folder titled *AuraDB*: include here your code with required commands to execute it for performing each task of the assignment on AuraDB.

3. A folder titled *GDS*: include here your code with required commands to execute it for performing each task of the assignment on Neo4j GDS.