M222: Analysis of Social Networks
University of Athens
Deprt. of Informatics & Telecommunications

Programming Project II
Today's Date: April 19^{th}, 2021
Due Date: May 24^{th}, 2021

PREAMBLE

In this project, you will design, implement and demonstrate a REST API that uses a Neo4j database and allows for efficient retrieval of information related to computer science publications.

PROBLEM DESCRIPTION:

The challenge to be addressed in the context of this project is to expose information about computer science publications through a REST API. You will download and review raw computer science bibliography data[1] to come up with an appropriate schema for the Labeled Property Graph model. You will then process the data to create a Graph Database than can answer related queries.

The raw data are in XML format. You will focus on the following entities:

- **article**: An article published in a journal. Mandatory fields: author, title, year, journal.
- **inproceedings**: Article that appeared in the proceedings of a conference. Mandatory fields: author, title, booktitle, year.
- **incollection**: Part of a book having its own title. Mandatory fields: author, title, booktitle, publisher, year.

Using this information you will create a network schema according to the Labeled Property Graph Model. Take extra care while modeling the data as nodes, relationships and properties. Make sure that all mandatory information is used. Please also note that some optional fields must be used as they are necessary in the context of certain queries. You can use all optional information if you wish.

IMPLEMENTATION ASPECTS:

You will use the `Neo4j` Graph Database in this project, to model and persist the data provided and build a REST API that queries this database to expose the data. You can find a simple example in Python (`hello-world-py2neo.py`) using the Neo4j database here:
`https://github.com/panagiotisl/neo4j-py2neo-vagrant-example`.

You can use the Vagrant file provided in this example to get a Neo4j database up and running or set-up your own database. The Vagrantfile pre-installs libraries py2neo and Flask which can help you with this project. However, you are free to use any programming language or library.

Moreover, you have the option of limiting the data you will use in this project by following one of the approaches below:

[1] `https://dblp.uni-trier.de/xml/`

1. Use a limited number of journals, conferences and books. In any case, you should use at least two of each category, and no less than 20 in total.
2. Use data concerning a particular time period, that spans a minimum of 3 years.

Your database should be able to answer the following queries that will be exposed through your API:

1. Find the titles (title, year) of publications that a particular author has published.
2. Find the co-authors of an author (name, number of co-authorships) for a particular year.
3. Find the top-K authors (name, count) with regard to most conference/journal publications. (2 methods)
4. Find the top-K authors (name, count) with regard to most co-authors in a single work.
5. Find the top-K authors (name, count) with regard to most co-authors in a particular year.
6. Find the top-K authors (name, count) with regard to most active years.
7. Find the top-K authors (name, count) with regard to most co-authors that have not published together.
8. Find the top-K authors (name, count) with regard to largest average number of journal publications per year (consider only *active* years).
9. Find the top-K authors (name, count) that a given author has not worked with, with regard to most co-authorships with authors that the given author has worked with.
10. Find the authors (name, count) that have published more than three works in a given single year.
11. Find the number of pages that a particular author has published in a given year.
12. Find the top-K authors (name, count) with regard to articles published in a particular journal as a first/last author in a given year. (2 methods)
13. Find the three authors that have appeared as co-authors for the most times in a particular journal.
14. Find pairs of authors that have appeared in different parts of the same book and have never co-authored a work.
15. Find the authors that have published work for K consecutive years.
16. Find the top-K authors with regard to average number of co-authors in their publications.
17. Find the authors of consecutively published papers with more than a given amount of years between them.
18. Find the author (name, count) with the most parts in a single book of collective works.

In case some of the above queries do not have any results using the data of your database you are free to create *bogus* information to test your queries.

You should use indices and constraints wherever you believe is appropriate to do so.

HINTS:

- It might be a good idea to process bibliographic entries one at a time.
- DBLP uses a DTD file that specifies elements and attributes used in its bibliographic entries. You will face difficulties parsing the text as XML if you do not use this file.
- Certain elements, such as the pages of a publication, might not always be in the desired format. We are looking for a best effort approach that works most of the time. You are not supposed to fix individual entries that cannot be parsed in a straightforward manner.

You do not have the option of forming a team in this project. However, discussions on most aspects of the project in the classroom's forum are encouraged.

COOPERATION:

You do not have the option of forming a team in this project. However, discussions on most aspects of the project in the classroom's forum are encouraged.

REPORTING:

The final *typed* project report (brief report) must consist of:

1. Code for populating the database using the provided files. In case you limit the data used, this should include your filtering approach.
2. Your Cypher queries.
3. Code for your API.
4. Curl commands with examples for all queries of your API (these can be easily exported from API clients such as Insomnia[2])
5. Results for each of your queries.

Finally, you will have to demonstrate your work.

[2] `https://insomnia.rest/download/`