Barcelona School of Economics

# 23D020: Big Data Management for Data Science
# Lab 1: Graph Databases

*This is a hands-on lab on Property Graphs. We will be using what is, most probably, the most popular property graph database:* `Neo4j`*. We will practice how to create and manage graphs, as well as querying/processing them.*
*In the training document, we provide instructions for setting up the environment and here we list the exercises to be solved. One team member, in the name of the team, must upload the solution. Please use your* `[TeamID]` *as the name of the encapsulating folder of your deliverable. Please check the assignment deadline and be sure to meet it. It is a strict deadline!*

## Assignment

## A Modeling, Loading, Evolving

This part is about modeling graph data for a given domain. The main goal is to develop your skills on modeling and instantiating graph data.

### A.1 Modeling

We want to create a graph to model research articles. In this domain, authors write research articles that can be published as scientific papers (papers for short) in the proceedings of a conference/workshop (a conference is a well-established forum while a workshop is typically associated to new trends still being explored), or in a journal. A proceeding is a published record which includes all the papers presented in the conference/workshop. A conference/workshop is organized in terms of editions. Each edition of a conference/workshop is held in a given city (venue) at a specific period of time of a given year. Oppositely, journals do not hold joint meeting events and, like a magazine, a journal publishes accepted papers in terms of volumes. There can be various volumes of a journal per year.
A paper can be written by many authors, however only one of them acts as corresponding author. A paper can be cited by another paper (meaning their content is related). A paper relates to one or more topics through the concept of keywords (e.g., property graph, graph processing, data quality, etc.). Keywords are used by readers to quickly identify the main topics discussed in the paper. Furthermore, any scientific communication must

contain an abstract (i.e., a summary of its content).
Finally, we also want to include in the graph the concept of review. When a paper is submitted to a conference or a journal, the conference chair or the journal editor assigns a set of reviewers (typically three) to each paper. Reviewers are scientists and therefore they are relevant authors (i.e., published many papers in relevant conferences or journals). Obviously, the author of a certain paper cannot be reviewer of her own paper.

*Note: The attributes of the concepts are not explicitly given, thus use common sense and the descriptions of the next sections to define attributes for each concept, e.g., a paper contains attributes like Year, Pages, DOI, etc. Be sure to clearly specify the attributes you considered in the lab document.*

**Tasks**

Perform and explain your solution to the following tasks in the lab document:

1. Create a visual conceptual representation of the graph you would create (in terms of nodes and edges).

2. Justify your design decisions. Besides maintenance or reusability issues, also consider the performance of queries in Part B when creating your solution.

*Note: The solution to A.1 must be compiled in the lab document under section A.1.*

## A.2 Instantiation/Loading

1. Define the Cypher expressions to create and instantiate the solution created for the previous section.

*Note: The solution for A.2 is a program/script that creates the graph by inserting sample data. Besides that, explain in the lab document, under section A.2, how did you generate this data and any other assumption you want to make explicit about your solution.*

You should not create the data manually (one by one). It is recommended that you use real data, e.g., load the graph from DBLP[1] (`https://dblp.uni-trier.de/`), or from Semantic Scholar (`https://www.semanticscholar.org/`). Unfortunately, most data out there is not natively produced as graph data. Therefore, this is a mandatory step in most projects: transform JSON or CSV data to graph data. To facilitate such tasks, most graph databases incorporate built-in bulk load functionalities.

[1] `https://dblp.org/faq/How+can+I+download+the+whole+dblp+dataset.html`

For example, the DBLP data is available in XML format. In Neo4j, the bulk load is implemented via the LOAD CSV command: `https://neo4j.com/docs/cypher-manual/current/clauses/load-csv/`. You can use the following tool to convert DBLP data from XML to CSV: `https://github.com/ThomHurks/dblp-to-csv`, which allows you to generate a CSV file that is Neo4j-compatible providing some options, e.g.:

```
python XMLToCSV.py --annotate --neo4j
data/dblp.xml data/dblp.dtd dblp.csv --relations
author:authored_by journal:published_in
```

Note that these files do not contain information for all the enumerated concepts in A.1, thus you can fill in the missing part with synthetic data. In any case, if obtaining real data turns out to be difficult, you can generate synthetic data. Make sure you generate enough data as to get results for the queries in Section B and C.

### A.3 Evolving the graph

One key aspect of graph databases is their flexibility to absorb changes in the data coming into the system. The goal of this part is to experience such characteristics.

In the model and instances you created you were asked to identify the reviewers of a certain paper. Now, we want to change such modeling to store the review sent by each reviewer. A review, apart from its content (i.e., a textual description) has a suggested decision. A paper is accepted for publication if a majority of reviewers supported acceptance. Typically, the number of reviewers is 3 but every conference or journal may have a different policy on the number of reviewers per paper. Furthermore, we also want to extend the model to store the affiliation of the author. That is an author is affiliated to an organization which can be a university or company.

#### Tasks

1. In the lab document, propose a modification of the graph model created in A.1 and highlight the changes introduced. Justify your decision.

2. Create a program with the necessary Cypher queries to transform your Neo4j graph (including data and metadata) into an updated graph aligned with the evolved model proposed in the previous item.

3. Add the required additional Cypher queries for instantiating the new concepts of your graph in the developed application.

*Note: The solution for 1 must be included in the lab document under section A.3, while 2 and 3 in applications created.*

# B Querying

In this part we will query the generated graph data. Write the Cypher queries for the following:

1. Find the top 3 most cited papers of each conference.
2. For each conference find its community: i.e., those authors that have published papers on that conference in, at least, 4 different editions.
3. Find the impact factors of the journals in your graph (see `https://en.wikipedia.org/wiki/Impact_factor`, for the definition of the impact factor).
4. Find the h-indexes of the authors in your graph (see `https://en.wikipedia.org/wiki/H-index`, for a definition of the h-index metric).

*Note: The queries must be included in the lab document under section B and in your program.*

# C Graph algorithms

You can process graph data with traditional graph metrics / algorithms. To do so, you must use Neo4J's Graph Data Science library `https://neo4j.com/docs/graph-data-science/current/`.

The Graph Data Science library is delivered as a plugin, so you need to install it. Follow the instructions at: `https://neo4j.com/docs/graph-data-science/current/installation/`. You have two options, either through Neo4J's Desktop or by carefully following the steps mentioned in the link above.

Before going on, verify the plugin has properly been setup in your setup. Beyond regular queries, graphs allow to exploit graph theory and use well-known graph algorithms. The main goal of this part is to develop your skills on using graph algorithms to query graph data. Graph algorithms are used to compute metrics for graphs, nodes, or relationships and they are parametrized depending on the domain. For instance, the Page Rank algorithm applied to a 'web page search' domain with parameters like `('Page','LINKS')`, would find the importance/relevance of the pages taking into account the number and the relevance of links to the page. Similarly, when applying any algorithm you should contextualize it to our domain (i.e., they should make sense in the research article publication domain).

Carefully read the algorithms, understand them and understand how to parametrize and call them. The algorithms you can chek for the lab are the following. You can also look into any other algorithm not available here.

1. Centrality Algorithms
    - PageRank
    - Betweenness
    - Closeness
2. Community Detection Algorithms
    - Triangle counting
    - Louvain
    - (Strongly) Connected components
3. Similarity Algorithms
    - Node similarity
4. Path Finding Algorithms
    - Dijkstras shortest path (from one node to another)

**Tasks**

1. Choose one algorithm and write the Cypher script that triggers it correctly (not only syntactically but also semantically).
2. Give a rationale of the result obtained: i.e., what data are you obtaining? Provide an interpretation from the domain point of view.

*Note: The solution for this section must be included in the lab document under section C and in your program.*

## Deliverables

1. Python program or Cypher scripts to implement the tasks defined in Sections A, B, and C. The scripts must be included in a single zip file.
    - The code must include comments to facilitate the understanding. At the header of each file, include an overall comment explaining what are the steps implemented in the pipeline, and refer to these steps when explaining the code in the subsequent comments.
    - The execution of the program should be facilitated. For instance, the code should not include absolute paths or fixed user credentials (e.g., they should be requested by the user or stored in configuration files).

2. A PDF file (max two A4 pages, 2.5cm margins, font size 12, inline space 1.15) to answer the questions/decisions/assumptions in Sections A, B, and C.

   **Important**: Please use your `[Team ID]` as the name of the encapsulating folder of your deliverable..

## Assessment Criteria

1. Conciseness of explanations
2. Understandability
3. Coherence
4. Soundness