# Lab 4 Graph-oriented Databases

## Goals

The objectives of this work are:

Understand the fundamentals of graph representation-oriented databases.

Install and use an open source solution.

Develop solutions for various use cases.

**previous note**

This module requires the use of Open Source Neo4j Desktop software available for MAC, Linux and Windows platforms.
Submit code/results/reports in elearning.
Good job!

### 4.1 Neo4j – Installation and exploitation

Neo4j is a graph database developed by Neo4j, Inc. the version *Community Edition* CE) is made available under license *GNU General Public License (GPL) v3*

a) Install the CE version of *Neo4j Desktop* on your personal computer. Follow the instructions available on the website https://neo4j.com/download/

b) Once installed, run the application *Neo4j Desktop*, create a database (eg *tests41*) and open the *url* available (http://localhost:7474/browser/

c) Study the functioning of the system, exploring the examples, namely the collection *Movie*

Consult the slides provided for the course and websites with documentation on Neo4j:

*https://neo4j.com/developer/get-started/*

*https://neo4j.com/docs/developer-manual/current/cypher/functions/*

*https://neo4j.com/graph-databases-book/*

You should study concepts and features such as:

Creation of projects and database

Creating nodes and relationships

Write, Read, Edit, Delete (CRUD)

Search through language *Cypher*

*Note: It is not necessary to deliver any results in this section. However, it is recommended that you take the time to understand the main concepts before moving on to the next paragraphs.*

## 4.2 Exploration of the movie graph

in the application *Neo4j Desktop* create the project *cbd* and inside the database *films* start up *films* and import the movie database that is already available in neo4j using the command:

```
$ :play movie-graph
```

*Note: after this process, do not insert more vertices/edges in this database. If you intend to do insertion tests, use another DB.*

Run and analyze the results of the following expressions, in *Cypher*

```
MATCH (n) RETURN n
```

```
MATCH (n)
WITH COUNT(n) AS numVertices
MATCH (a)-[e]->(b)
RETURN numVertices, COUNT(e) AS numEdges
```

Analyze and execute the following expression, which displays the types of nodes that exist in the database:

```
match(n)
return labels(n) as labels, keys(n) as keys, count(*) as total order by total
desc;
```

In this expression:

match() select any node

match(n) selects any node and assigns it to the variable n

- the function labels(n) returns a list of type (*label*) of each node n
- the function keys(n) returns the list of properties for each node n
- the function count(n) is an aggregation function, in this case it counts each group *labels keys* O GROUP BY is implied in *Cypher*

In the same way, we can extract information about existing relationships:

```
match (m)-[r]->(n)
return labels(m), type(r), labels(n), count(*) as total order by total
desc;
```

We can also obtain the properties of each relation:

```
match()-[r]->()
return type(r) as type, keys(r) as keys, count(*) as total order by type;
```

Build now expressions *Cypher* to answer the following questions:

*Note : Write all the answers in the file CBD_L42_<NMEC>.TXT, Where NMEC> must be replaced by its mechanographic number. For each question, N, always write a previous line with the content #N. If you want to include comments use "//". Example:*

```
// NMEC: 12345
# 1
match...
```

1. Find all the actors who directed a movie they also starred in and display the actor's name and the movie title.

2. For each film made after 2005, list the names of all actors who played in that film.

3. Find pairs of nodes with more than one relationship to each other.

4. Find all pairs of people who have reviewed the same movie. Submit your names and title of each movie.

5. Find all pairs of actors who have acted in multiple movies together.

6. Determine the average age of the cast of the movie "Apollo 13" in the year of the movie's release.

7. Find the 10 movies with the oldest cast at the time of the movie's release. Present the film and average age rounded to 2 decimal places, in descending order.

8. Present the subgraph ACTED_IN of the film with the youngest cast, at the time of the film's release.

9. What is the shortest path (using any kind of relationship) between John Cusack and Demi Moore?

10. What is the shortest path dimension (using any kind of relationship) between Keanu Reeves and Tom Cruise?

11. What are the dimensions of the shortest path between people named Jim and people named Kevin?

12. Which people have a distance of 2 to Jim Cash (the distance between two people is the length of the shortest path between them)?

13. What is the furthest distance from a person to Kevin Bacon?

14. What is the greatest distance between two people?

15. What is the distribution of distances in pairs (ie, for distance 1, 2, 3, ..., how many pairs of people are that distance apart)?

16. Indicate the 10 people with the smallest average distance in which the path between them are ACTED_IN relations.

17. Introduce actors who had at least one role for characters prefixed with 'Dr.'. The result should be a list with the pair “Actor, List of roles”.

18. .. 20. Create 3 new additional questions to the database (items 18 to 20) and also present the research solution for each question. These should not be similar to the questions in paragraphs 1 to 17. The pertinence and degree of difficulty of the questions will be valued. Examples drawn from public sources, literature or documentation will not be considered.

## 4.3 Analysis of a network of programmers and projects

In this exercise, we intend to analyze a network made up of projects and their members, which are part of a software repository.
*Note : Write all the answers in the file CBD_L43_<NMEC>.TXT*

a) Based on the content of the file *git_selection.csv*, model the entities and relationships needed to load your content into net4j.

b) Create the database *github* Copy the file to the folder *import* of this database. See the link *http://neo4j.com/docs/operations-manual/current/configuration/file-locations/* to understand where the database is stored on your OS. in the interface *http* from Neo4j, load the contents of the file using the LOAD CSV command (pay attention to the syntax of this command, the structure of the CSV file and the modeling you did previously).

c) Build expressions *Cypher* to provide the following information:

1. List each user's information.
2. List each user's name.
3. List the information of each project, in which at least one user has participated.
4. List the users and total number of projects in which each one collaborates.
5. List the users and the total number of projects in which each collaborates in descending order of the total.
6. List projects and total members in each project.
7. List projects and total members with role "Committer" on each project.
8. List all users who participated in the same projects as the user "atm" (id). Show attributes: "atm" name, username 2, project name.
9. List all users who participated with the role "Committer" in projects in which the user "atm" (id1) participated with the role "PMC".

### 4.4 Free Theme - Driver (Java)

a) Install a Neo4j driver for Java (or another language) and create a small program to connect to the Neo4j server, to a new database (eg *lab44*

b) Identify a *dataset* of your choice, different from the previous ones with a minimum size of 500 nodes and at least three different types of relationships. You can search and download it from the internet or build one yourself, in a format that is suitable for loading via the driver. must not use *dataset* public tutorials or Neo4j documentation.

c) Create a program to insert the *dataset* in the Neo4j database. Programmatically build a minimum of 10 queries of your choice and display in the file *CBD_L44c_output.txt* each search and its result.