Group Project - Neo4j

of Systems and Methods for Big and Unstructured Data Course (SMBUD)
held by
Brambilla Marco
Tocchetti Andrea

Group 78

Pisante Giuseppe
10696936

Raffaelli Martina
10709893

Academic year 2024/2025

POLITECNICO
MILANO 1863

Contents

1 Introduction

The project aims to design and implement a database system to support the management of data related to the film industry. The database will include entities such as Person (actor, director, writer), Title (film, TV series), Episode, ratings, and genre. The goal is to create a comprehensive system that can store and query information about films, TV series, and the people involved in their production. The project will develop with Neo4j, to exploit the relations between the entities and find significant insights from the data, such as collaborations between directors and actors, trend of genres over time, most working actors and the most successful films.

2 Assumptions

The project is based on the following assumptions:

- Each person has a unique ID and a name, surname, and date of birth. Some could also have a death date, if passed away.
- Each person can be associated with multiple roles (actor, director, writer, archive footage, music department, producer)
- Each title has a unique identifier and a title type (film, TV series, shortfilm)
- Each title can have multiple episodes
- Each title can have multiple genres
- Each user can rate a title if and only if it has watched the film
- Each title has a unique rating, average rating from the users, and the number of votes
- Each person can be associated with multiple titles
- Each title can have multiple people associated with it

3 ER diagram

POLITECNICO
MILANO 1863

Figure 1: E-R Diagram

3.1 Entities

Starting from the considerations previously exposed regarding the implementation hypotheses, we have drawn an ER diagram (**Figure 1**) which includes 5 different entities and 7 many-to-many relationships described below in the logical model:

- **Person**(<u>nconst</u>, PrimaryName, BirthYear, DeathYear, PrimaryProfession, KnownForTitles)
- **Title**(<u>tconst</u>, PrimaryTitle, OriginalTitle, TitleType, StartYear, EndYear, RuntimeMinutes, Genres)
- **Episode**(<u>tconst</u>, ParentTconst, SeasonNumber, EpisodeNumber)
- **Ratings**(<u>tconst</u>, AverageRating, NumVotes)

- **Genre**(Name)

The **Person** entity describes every possible individual with their own personal data, including their primary profession and titles they are known for. The **Title** entity represents films, TV series or short films with their respective attributes. The **Episode** entity is used to detail episodes of TV series, linked to their parent series. The **Ratings** entity captures the average rating and number of votes for each title. Finally, the **Genre** entity categorizes the titles into different genres.

3.2 Relationships

- **HasRole**(nconst, tconst, Category)
- **HasRating**(nconst, tconst, Rating)
- **HasGenre**(tconst, Genre)
- **HasEpisode**(tconst, Episode)
- **HasTitle**(nconst, tconst)
- **HasPerson**(tconst, nconst)
- **HasTitleGenre**(tconst, Genre)

4 Dataset description

One of the most critical parts of working with Big Data is managing large amounts of data collected in large datasets. To test and simulate the use of the database for contact tracing activities, some sample datasets were generated, saved in .csv format and imported into Neo4j through the command: `LOAD CSV FROM "file: ///file.csv" AS ....`

Each dataset is divided into various fields that trace the structure of the tables expressed in the ER model, each one was generated randomly through Python scripts (you can find these ones into the folder `"db"`) and to experiment and perform at best the possible tests on the queries and commands that can be executed thanks to Neo4j the number of entries foreseen for each dataset is in the order of magnitude of the hundreds, as a whole, starting exclusively from the data loaded from the datasets, the database provides:

Homes	101	Residence Relationship	300
People	300	Meetings	726
Public Places	20	Public Places Visits	600
Vaccine Types	4	Vaccinations	526
Test Types	2	Tests	450
TOTAL NODES	**427**	**TOTAL RELATIONSHIPS**	**2602**

5 Queries and Commands

The correct functioning of the information system involves the implementation of some essential commands and queries for the database in order to properly support the app and to ensure the right execution of searches among the data available for statistical or practical purposes.

First of all you need to load the .csv files with the query you can find following the path `"documents/importDB.txt"`

5.1 Queries

5.1.1 Find how many people went without greenpass in a public place

This query allows us to find all the people that went in a public place without the GreenPass and the datetime related.

The output is given by:

- The name and the surname of the person

- The entrance time in the public place
- The name of the public place

POLITECNICO

MILANO 1863

5.1.2 Find who lives with an infected individual

This query allows us to detect a "family contact" with an infected person and obtain the people that need to be quarantined.

The output is given by:

- The house identifier
- The name and the surname of the person
- The name and the surname of the infected cohabitant

Figure 2: Visual representation of the result with NeoDash

5.1.3 Find public place contact with infect people

This query allows us to find the people who went in the same public place (e.g. restaurant, cinema,...) at the same time with infected people starting from 15 days before their positive test.

POLITECNICO
MILANO 1863

The output is given by:

- The name and the surname of the person
- The name and the surname of the infected cohabitant
- The datetime of the entrance of the health person
- The datetime of the entrance of the ill person
- The public place

5.1.4 Find who got vaccinated in a temporal range

This query allows us to find all the vaccinated people in the temporal range of October 2021.

POLITECNICO
MILANO 1863

The output is given by:

- The name and the surname of the person
- The type of the vaccine
- The datetime of the vaccine

POLITECNICO
MILANO 1863

5.1.5 Statistical analysis of the vaccination campaign

This query allows us to compute an analysis on the number of the vaccinated people (and the percentage) grouped by age ranges.

The output is given by:

- Total people
- Total vaccinated people (with %)
- Vaccinated people in age ranges (with %)

5.2 Commands

5.2.1 Federico moves in an other house (CREATE)

In this command we simulate a person moving in an another house.

5.2.2 Delete all the public place records older than 1 year (DELETE)

In this command we simulate the need of removing old records that are not useful anymore.

5.2.3 Name change of public place (UPDATE)

In this command we simulate the need, of a public place manager, of changing name of the activity.

6 UI description & User Guide

6.1 UI description

The design of the information system ended with the development of *ContagionShield*, an application connected to the database with a simple GUI that is very intuitive and easy to use. The UI was created with the Python programming language by means of the libraries: *PySimpleGUI* for the creation of the graphical interface, *Py2neo* to work with Neo4j through the syntax offered by Python and *pandas* to manage the analysis and manipulation of data, you can find these ones into the folder `"contagionshield"`.

Figure 3: Query Interface

As you can see from **Figure 3**, the main screen of the application is divided into two sections: one on the left that allows you to create customizable queries by choosing date, place, time interval and whether to display the vaccinated, non-vaccinated or with negative test, in **Figure 4** there is an example query with the results obtained; the other side section on the right (**Figure 5**) presents some predefined queries, some of them follow the ones specified and described in Chapter 5 of the report.

Figure 4: Example Query with Results

Figure 5: Predefined Queries

Finally, as shown in **Figure 6**, it is possible to execute some of the commands already presented in Chapter 5 by reaching the appropriate section of the application by means of the `"Commands"` button in the lower left corner of the main screen. The executable commands allow you to create new meetings between several people by indicating their social security numbers, the date and time, create new visits to public places by specifying who went to a given place and when and in the end you can also delete all the visits to public places recorded in the last year.

Figure 6: Command Interface

6.2 User Guide

In order to run *ContagionShield* it is necessary to verify some requirements and to perform some actions:

- At first you have to check if you have the right Python version installed by using the command: `python --version`, if not you could download it from the official website: https://www.python.org/
- Then make sure your local database is at `localhost:7687` and that it has `"smbud"` as password
- Install the required packages (in the ”contagionshield” folder): `pip install -r requirements.txt`
- Finally make it run by navigating to the folder where you have been saved the materials, then into `"contagionshield"` folder and run: `python contagionshield.py`

- From there you can execute queries about the collected data

7 References & Sources

[1] Course Slides

[2] https://pysimplegui.readthedocs.io/en/latest/call

[3] https://py2neo.org/

[4] https://neo4j.com/docs/cypher-manual/current/

[5] https://neo4j.com/developer/python/

[6] http://iniball.altervista.org/Software/ProgER

[7] https://neo4j.com/developer/cypher/

[8] https://pandas.pydata.org/docs/