Advanced Databases (DAT410) Assignment 4

Luca Modica
Hugo Manuel Alves Henriques e Silva

May 17, 2024

1 Introduction

With a similar approach as in the previous assignment, this report aims to illustrate how data from a relational database are prepared, loaded and represented in a graph database. The graph DBMS used in this work is **Neo4j**, which represents data and their relationships with nodes; compared to GraphDB, more suitable for semantic web applications and knowledge graphs, Neo4j is meant to handle graph data more efficiently thanks to a different architecture and slightly different graph model, making it popular for graph analysis of complex data.

After describing how the relational data (in the form of CSV files) are prepared, we will illustrate several results of querying the transformed data to answer specifically asked questions. This will be achieved by using **Cypher**, a query language meant for working with graph data.

2 Data preparation

The first task is to prepare the CSV files to be loaded into Neo4j. By approaching this part of the assignment, we observe a notable similarity in how GraphDB and Neo4j represent data through a graph model; thus, by using the ontology model created in previous assignments, the first step was to convert the CSV files in RDF triples, conforming to the OWL/RDFS produced. Next, to use the converted data in Neo4j, we employed a plugin called Neosemantics: It is a specialized tool to facilitate RDF triples mapping into a Neo4j graph data model, allowing us to convert object properties in relationships between nodes, as well as RDF data properties in node properties.

The main benefit of our approach is to gain a flexible standardized data representation through the intermediary RDF triples conversion step, for a more robust initial data modeling.

3 Cypher queries

Once the RDF triples are loaded into Neo4j through plugin *Neosemantics (n10s)*, the next task consists of querying the data to answer the given questions in the assignment. Down below, for each question, we will report the related Cypher queries and the results obtained by running them.

- **Question 1: find the name, director and department of all programmes.**

  CYPHER query:

```
match (programme:Programme)
match (programme)-[:programmeDirectedBy]->(teacher:SeniorTeacher)
match (programme)-[:programmeBelongsTo]->(department:Department)
return programme.programmeName as programmeName,
       teacher.teacherId as teacherId,
       teacher.teacherName as teacherName,
       department.departmentName as departmentName
```

  Result:

programmeName	teacherId	teacherName	departmentName

”””P-01”””	”””19620522-0023”””	”””Teacher23”””	”””D1”””
”””P-42”””	”””19620831-0024”””	”””Teacher24”””	”””D5”””
”””P-41”””	”””19580218-0007”””	”””Teacher7”””	”””D5”””
”””P-34”””	”””19610918-0027”””	”””Teacher27”””	”””D4”””
”””P-32”””	”””19570826-0012”””	”””Teacher12”””	”””D4”””
”””P-33”””	”””19570828-0008”””	”””Teacher8”””	”””D4”””
”””P-31”””	”””19650303-0019”””	”””Teacher19”””	”””D4”””
”””P-21”””	”””19570615-0011”””	”””Teacher11”””	”””D3”””
”””P-12”””	”””19610623-0005”””	”””Teacher5”””	”””D2”””
”””P-13”””	”””19690408-0009”””	”””Teacher9”””	”””D2”””
”””P-11”””	”””19620424-0026”””	”””Teacher26”””	”””D2”””
”””P-14”””	”””19560812-0016”””	”””Teacher16”””	”””D2”””
”””P-73”””	”””19600814-0002”””	”””Teacher2”””	”””D8”””
”””P-71”””	”””19610620-0006”””	”””Teacher6”””	”””D8”””
”””P-72”””	”””19660630-0020”””	”””Teacher20”””	”””D8”””
”””P-74”””	”””19601021-0018”””	”””Teacher18”””	”””D8”””
”””P-61”””	”””19680712-0028”””	”””Teacher28”””	”””D7”””
”””P-52”””	”””19611219-0014”””	”””Teacher14”””	”””D6”””
”””P-53”””	”””19600905-0003”””	”””Teacher3”””	”””D6”””
”””P-51”””	”””19580515-0017”””	”””Teacher17”””	”””D6”””
”””P-54”””	”””19630126-0001”””	”””Teacher1”””	”””D6”””

- **Question 2: find the names of all students who worked as teaching assistants in courses given by the D3-2 division in study period 2 in academic year 2023/2024.**

  CYPHER query:

```
match (s:Student)
match (s)-[:workAsTA]->(ta:TeachingAssistant)
match (th:TeacherHours)-[:teacherHoursIn]->(ta)

match (th)-[:courseHoursIn]->(cInstance:CourseInstance)
where cInstance.courseInstanceAcademicYear = "2023-2024"
    and cInstance.studyPeriod = 2

match (cInstance)-[:courseInstanceOf]->(c:Course)-[:courseBelongsTo]->(d:
   Division {divisionName: "D3-2"})

return s.studentName as studentName
```

  Result:

studentName
”””TA60”””
”””TA138”””
”””TA36”””
”””TA38”””
”””TA74”””

- **Question 3: find all teachers who are assigned more than 120 hours in course 1015 in study period 1 in academic year 2018/2019.**

  CYPHER query:

```
match (th:TeacherHours)-[:teacherHoursIn]->(t:Teacher)
match (th)-[:courseHoursIn]->(cInstance:CourseInstance)-[:courseInstanceOf
    ]->(c:Course {courseCode: 1015})
where cInstance.studyPeriod = 1
    and cInstance.courseInstanceAcademicYear = "2018-2019"
    and th.assignedHours > 120
return t.teacherId as teacherId,
    t.teacherName as teachername,
    th.assignedHours as assignedHours
```

Result:

teacherId	teacherName	assignedHours
”””19660630-0020”””	”””Teacher20”””	240.0
”””19750102-0059”””	”””TA59”””	140.0
”””19650303-0019”””	”””Teacher19”””	240.0
”””19790702-0038”””	”””TA38”””	140.0
”””19580218-0007”””	”””Teacher7”””	280.0

- **Question 4: find all students registered for course instance I-910 that were not registered for course instance I-911.**

  CYPHER query:

```
match (registration:Registration)-[:registrationContainsInstance]->(
    courseInstance:CourseInstance {instanceId: "I-910"}),
       (registration)-[:studentRegistered]->(student:Student)
optional match (student)-[:studentRegistered]->(registration2:Registration)
    -[:registrationContainsInstance]->(courseInstance2:CourseInstance {
    instanceId: "I-911"})
with student, courseInstance2
where courseInstance2 is null
return student.studentId as studentId, student.studentName AS studentName
```

Result:

studentId	studentName
”””19921201-0094”””	”””TA94”””

- **Question 5: find all programmes along with the total number of owned courses. List the results in descending order of number of owned courses.**

  CYPHER query:

```
MATCH (course:Course)
MATCH (course)-[:courseOwnedBy]->(programme:Programme)
WITH  programme, COUNT(course) AS numberOfCourses
RETURN programme.programmeCode AS programmeCode,
       programme.programmeName AS programmeName,
       numberOfCourses
ORDER BY numberOfCourses  DESC
```

Result:

programmeCode	programmeName	numberOfCourses
10061	”””P-61”””	45
10021	”””P-21”””	33
10001	”””P-01”””	32
10041	”””P-41”””	21
10042	”””P-42”””	20
10032	”””P-32”””	14
10071	”””P-71”””	14
10052	”””P-52”””	13
10033	”””P-33”””	12
10012	”””P-12”””	12
10054	”””P-54”””	11
10051	”””P-51”””	11
10011	”””P-11”””	10
10072	”””P-72”””	10
10074	”””P-74”””	9
10013	”””P-13”””	8
10014	”””P-14”””	8
10073	”””P-73”””	6
10034	”””P-34”””	6
10053	”””P-53”””	3
10031	”””P-31”””	2

- **Question 6a: find the number of: senior teachers**

  CYPHER query:

```
match (seniorTeacher:SeniorTeacher)
with COUNT(seniorTeacher) as numberOfSeniorTeachers
return numberOfSeniorTeachers
```

  Result:

numberOfTeacher
30

- **Question 6b: find the number of: all people**

  CYPHER query:

```
call {
  match (teacher:Teacher)
  return teacher.teacherId as id
  union
  match (student:Student)
  return student.studentId as id
}
return count(distinct id) as numberOfPeople
```

  Result:

numberOfPeople
440