**Graph Theory Assignment**

**Ríona Greally – G00325504**

**Introduction:**

As Part of my Graph Theory Module we were required to create a Timetable Database using Neo4j. I created group C’s semester one timetable, for Software Development, Year 3 in GMIT using the query language Cypher.

**Neo4j:**

Neo4j is one of the World’s leading Graph Databases. Neo4j is an open-source NoSQL graph database implemented in Java and Scala. Neo4j is a highly scalable graph database. This Graph database focuses on the relationships between data as well as the data itself.

“No broken links” is the main rule behind graph databases. This states that you cannot delete a node without also deleting the relationships associated with these nodes. This is because a relationship always has a start and an end node.

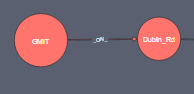
Cypher is the query language used in Neo4j for expressive and efficient querying and updating of the graph store. Cypher is a declarative language, meaning what you want to retrieve can be seen in the query expression. Unlike Java and other scripting languages, Cypher focuses on the clarity of expressing what to retrieve from a graph and not how to retrieve it.

**Database Structure:**

Start node: GMIT

GMIT is the start node as it is the College in which the student would be attending and would be where the student would begin when seeking their timetable.

GMIT has a -[:\_ON\_]-> relationship with Dublin\_Rd



Dublin\_Rd is of type Campus. This would be the next factor the student would choose. There would be other campuses like, ‘CCAM’, ‘Mayo’ etc on a larger scale.

Dublin\_Rd has a -[:\_HAS\_]-> relationship with Software\_Development.

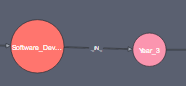


Software\_Development is of type Course, on Dublin Road campus of GMIT.

Properties: name: ‘ Software\_Development’ , department: ‘Science’

The course Software Development is in the Department of Science on Dublin Road campus of GMIT.

Software\_Development has a –[:\_HAS-]-> relationship with Year\_3



Year\_3 represents the year of study of the course Software Development the student is in.

Year\_3 has a -[:\_HAS\_]-> relationship with Group\_C.



Group\_C is a class group of students. On a larger scale timetable, Year\_3 would also have other class groups like Group\_A and Group\_B etc.

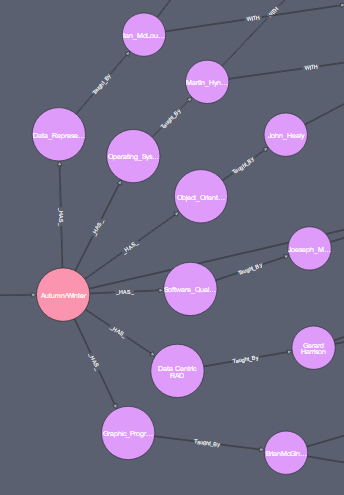
Group\_C has a -[:Semester\_1]-> relationship with Autumn/Winter.



Autumn/Winter is Semester 1 of the course year. A possible sister node of Autumn/Winter would be Spring/Summer, being semester 2. The semesters split the modules being studied during the course year, into the period of the course in which they are studied.

Autumn/Winter has a -[:\_HAS\_]-> relationship with the corresponding Module nodes. The modules represent the modules studied by the student in Group\_C for Semester\_1. Data\_Representation\_&\_Querying and Operating\_Systems are example of these module nodes.

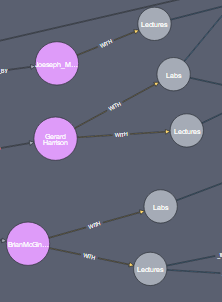
The Module nodes have a -[:Taught\_By]-> relationship with the corresponding lecturer that teaches that module.



I then have each lecturer nodes pointing to two other nodes in most cases, depending on the modules. Majority of the modules in Semester One are taught with lecture classes as well as labs, while others are taught with just lectures. I have split them into labs and lectures so the student could access a specific type easier.

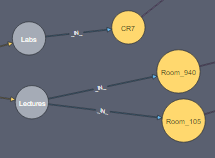
(lecturer) – [:WITH] -> (labs)

(lecturer) – [:WITH] -> (lectures)



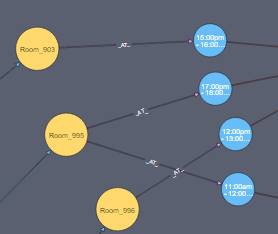
Labs or Lectures are then followed by with a relationship with the rooms they are in.

This is a -[:\_IN\_]-> type relationship.



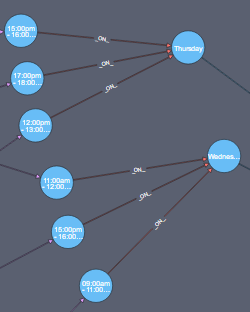
The rooms are then connected to the times in which the class starts and ends in that room.

This is a -[:\_AT\_]-> relationship between the rooms and time nodes.



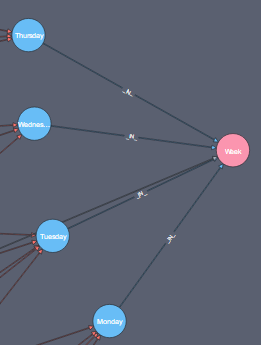
The TIME nodes are then connected to the DAY nodes. This relationship represents the day on which the modules on at that TIME.

This is a -[:\_ON\_]-> relationship.



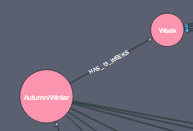
The Day nodes are connected to Week. There are only four Day nodes as Group\_C have no classes on a Friday for Semester\_1 of Year\_3.

This is a -[:\_IN\_]-> relationship between the Day and Week nodes.



Finally, I then have a single relationship between Autumn/Winter and Week nodes. This relationship states that there are 13 weeks if college term in Semester\_1.

It’s a -[:HAS\_13\_WEEKS]-> relationship.



**Querying:**

We create queries in Cypher in order to retrieve specific information we want from the database.

Match (n) return n

This is used to retrieve all nodes and relationships in the database.

Query: match (col:College) -[:\_ON\_]-> (cam:Campus) -[:\_HAS\_]->(c:Course)

return col.title AS COLLEGE, cam.name AS CollegeCampus, collect(c.name) as COURSE

Order By COLLEGE



Match (c:Course) -[:\_IN\_]-(y:Year) -[:\_HAS\_]- (g:Group) -[:Semester\_1]- (sem:Semester)

return c.title AS COURSE, y.title AS YEAR , g.title AS ClassGroup, sem.title AS SEMESTER



**List Of Modules that have property: semester : ‘1’**

*MATCH (mod:Module)*

*WHERE mod.semester STARTS WITH '1'*

*RETURN mod.title AS Semester\_1*

*ORDER BY Semester\_1 ASC LIMIT 10;*



**List All Modules and the Lecture that teaches that module**

MATCH (module:Module)-[:Taught\_By]->(lec:Lecturer)

RETURN module.title AS MODULE, collect(lec.name) AS LECTURER

ORDER BY MODULE ASC LIMIT 10;



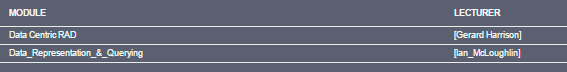
**Module Name and Lecture of that Module where Module title starts with ‘D’**

*MATCH (module:Module)-[:Taught\_By]->(lec:Lecturer)*

*WHERE module.title STARTS WITH "D"*

*RETURN module.title AS MODULE, collect(lec.name) AS LECTURER*

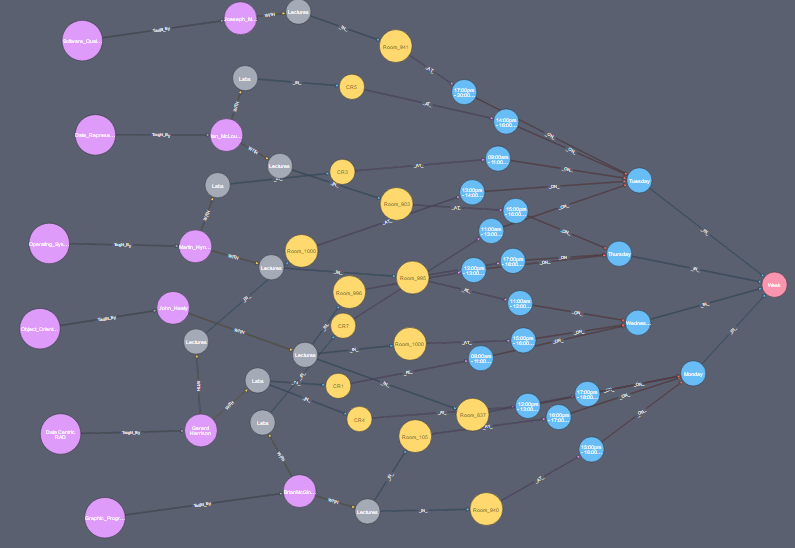
*ORDER BY MODULE ASC*



match (n)

where n.semester STARTS WITH '1'

return n



Returns the modules, lecturers, labs, lectures, rooms, time and days and their associated relationships.

**Commands to create some nodes with properties and the relationships between those nodes:**

CREATE (col:College {title: 'GMIT', name: 'GMIT', campus:'Dublin\_Rd'})

CREATE (cam:Campus {title: 'Dublin\_Rd', name: 'Dublin\_Rd', college:'GMIT'})

CREATE (c:Course {title: 'Software\_Development', name: 'Software\_Development', department:'Science'})

CREATE (year:Year {title: 'Year\_3', name: 'Year\_3', course:'Software\_Development', department:'Science'})

match (col:College {title: 'GMIT', name: 'GMIT', campus:'Dublin\_Rd'}),

(cam:Campus {title: 'Dublin\_Rd', name: 'Dublin\_Rd', college:'GMIT'})

create (col)-[:\_ON\_]->(cam)

match (c:Course {title: 'Software\_Development', name: 'Software\_Development', department:'Science'})

,(cam:Campus {title: 'Dublin\_Rd', name: 'Dublin\_Rd', college:'GMIT'})

create (cam)-[:\_HAS\_]->(c)

match (c:Course {title: 'Software\_Development', name: 'Software\_Development', department:'Science'}),

(year:Year {title: 'Year\_3', name: 'Year\_3', course:'Software\_Development', department:'Science'})

(c)-[:\_IN\_]->(year)

**Deleting a relationship between two nodes where relationship ID is 21**

MATCH ()-[r]-()

WHERE ID(r) = 21

DELETE r;

**Set a new property to an existing node**

MATCH (n:Course)

SET n.degree : 'Level\_7'

RETURN n

**Conclusion:**

Overall, I created a Timetable database for Semester 1 of a student in Group C of Year 3 studying Software Development in GMIT Dublin Road. In my opinion I think this database has a good structure and includes all the necessary nodes and relationships that a timetable should have.

**References:**

<https://neo4j.com/developer/graph-database/>

<https://neo4j.com/docs/developer-manual/current/cypher/clauses/>

<https://neo4j.com/docs/developer-manual/3.1/cypher/execution-plans/combining-operators/>