**Introduction**

This is a Project for the subject of graph theory at GMIT. The requirements for this project was to build a graph database for a timetabling system and implement it using the NEO4J framework.

The database keeps records about rooms, subjects, lecturer, student groups and its afore mentioned interactions.

**DB Schema**

The database schema consists of 4 nodes, said nodes will be used to keep information about .The choice of 4 nodes was made to keep the design more concise, since most of the tabling information will be kept on the relationships between nodes. Node’s names and description are detailed below.

The lecturer node is used to keep names of the lecturer of the college, it has the label “Lecturer” and a property called name, said property will keep the lecturer’s name.

Cypher syntax: (:Lecturer{name:”Lecturer’s name”})

The subject node is used to keep data of all subjects. it has the label “Subject” and two properties, one called name, which is the name of the subject. The other one is subject code, which is unique code for every subject.

Cypher syntax: (:Subject{name:”Subject’s name”, subject\_code:”Subject’s code” })

The room node is used to keep data of all rooms of the college, it has the label “Rooms” and two properties. One called number, which keep the room’s number, and the other one is called name. The name property keeps the room’s name, if the room has no special name, such as Gmit’s Blue Theater, the property must be filled with the room’s number.

Cypher syntax: (:Room {number: ‘room’s number’, name: room’s name})

The group node is used to keep data of all subjects. it has the label “Group” and two properties One called groupID, said property will keep the group id, and the other which is called classID, the classID property will keep id of the class.

Cypher syntax: (:Group { groupID:” Group ID”, classID:”Class ID” })

If group id is empty, means that all groups should attend

Example: 3rd software development group A

classID = KSOFG73

groupID = A

**Relationships Description**

The Lecturer node will have a relationship with the Subject node, the relation will have a label of type Teaches.

The Group node will have a relationship with the Subject node, the relation will have a label of type Attends.

The Group node will have a relationship with the Room node, the relation will have a label of type Uses and 3 properties. One is called time, which is time of the day, other called day, which day the room is going to be used and third one which is called groupid, which references the groups id. If more than one group should attend, said property will have and array with the codes of all referenced groups.

If all groups must attend, the group id property will be empty, and only the classID will be used. otherwise groupID will have the id of the only group that has to attend to said subject.

TimeB(Time Begin) property saves the time that said lecture will start.

TimeF(Time Finish) property saves the time that said lecture will end.

**How to obtain the data**

Part of the data used to populate the database are publicly available at gmit’ timetables web site.

To obtain the public available data yourself, follow the steps below:

1. Click [here](http://timetable.gmit.ie/)
2. Select “Academic Year 2016/17”
3. Click on programmes and then select the Department of Computer Science and Applied Physics
4. On “Programme Search String (optional):” enter KSOFG73 and click on Search
5. Select the program and click on view timetable

The Lecturer names are private information and can’t be obtained without being an valid user of [gmit’s learnoline](https://learnonline.gmit.ie/) website.

**Sample Queries**

Get all labs that only group B needs to attend to and its time:

Match (n:Subject)-[r:Uses {groupID:"B"}]-(m) return m,r

Get all lectures which all groups must attend to and its time:

Match (n:Subject)-[r:Uses {groupID:""}]-(m) return n,r

Get the whole timetable for 3rd Year Software Development

Match (n)-[r:Uses { classID :”KSOFG73”}]-(m) return n,r

Get all rooms and when they are used by the subject “Graph Theory”

Match (:Subject {name:"Graph Theory"})-[r:Uses]-(m) return r,m

Get all lectures

match (n:Lecturer) return n;

Get all lectures that a lecturer teaches

Match (:Lecturer {name:"Martin Hynes"})-[]-(m:Subject) return m

Get all lectures’ times and rooms of a lecturer

Match (:Lecturer {name:"Martin Hynes"})-[]-(m:Subject)-[r:Uses]-() return r

**Conclusion**

This project gave me the chance to work with a framework that I had to previous knowledge about, afore mentioned framework has an architecture that differs greatly from those used by others NoSQl DBs used at the market. Which imposed a big challenge in how to design the tabling system in a functional way and with great usability.

Thus I come from this project with acquired knowledge in a great framework and different design ideas which will certainly be helpful in the future.