**Graph database based timetabling system**

Overview of project:

This is a semester 2 project for Graph Theory module of year 3 Software development course. Aim of this project is to develop a prototype timetabling system for GMIT like the one currently used. As this is a graph theory module the Neo4J database will be used to store the timetable data. Neo4j stores data in nodes connected by relationships so the biggest immediate problem is deciding on what data should go into nodes and what sort of relationships should there be between its units.

Data to be stored in Database:

Looking at the existing database model that is used here in GMIT. I have decided that main data to stored can be separated into these basic units:

1. Programme name;
2. Subject/module name;
3. Lab group ID;
4. Room name/number;
5. Timeslot for lecture to take place at;

Appropriate Neo4J nodes have been created for a small-scale prototype of the timetable. Every node in the timetable has a single property “Name” which shows unique nodes among nodes of same type. I am using a single programme type node named Software Dev. as a root node for the prototype graph.

Cypher Query to create programme node:

**Create (p:Programme{Name:"Software Dev"});**

Cypher query to create a subject/module node:

**Create (s:Subject{Name:"Graph Theory"});**

Cypher query to create LabGroup node:

**Create (l:LabGroup{Name:”Group X”});**

Cypher query to create Room node:

**Create (r:Room{Name:”Room X”});**

Cypher query to create Timeslot node:

**CREATE (t:Timeslot{Name:”Slot X”});**

For the needs of prototype, I have created 1 Programme node (Software Dev), 2 Subject nodes (Graph Theory and Database Management), 3 LabGroup nodes (Group X, Group Y and Group Z), 3 Room nodes (Room X, Room Y, Room Z) and 4 Timeslot nodes (Slot W, Slot X, Slot Y and Slot Z).

This makes it a total of 13 nodes in small scale prototype database.

Database structure:

For the design of the database and how the layout of it should look like I decided to try and follow the logical flow of hierarchy.

1. Programme contains many modules in it;
2. Subject is being taught to 1 or more groups in the programme year;
3. Student group studies the given module in the assigned room;
4. Room is being given to group of students at certain timeslot;

From the above list 4 Relationships can be obtained:

1. Contains;
2. Taught to;
3. In;
4. At.

As such, these are the names for Relationships between the nodes in prototype database.

Cypher queries used to create above relationships:

1. **Match (p:Programme{Name:"Software Dev"}), (s:Subject{Name:"Database Mgmt"}), (s2:Subject{Name:"Graph Theory"}) create (p)-[r:Contains]->(s), (p)-[r:Contains]->(s2) return p, s, s2;**
2. Following query is for single relationship only. Other relationships of this type were created by changing LabGroup name accordingly. **Match (s:Subject{Name:"Graph Theory"}), (l:LabGroup{Name:"Group X"}) create (s)-[r:Tought\_To]->(l);**

For Database Management to LabGroup relationships I created all relationships in single query:

**Match (s:Subject{Name:"Database Mgmt"}),(la:LabGroup{Name:"Group X"}),(lb:LabGroup{Name:"Group Y"}), (lc:LabGroup{Name:"Group Z"}) create (s)-[:Tought\_To]->(la), (s)-[:Tought\_To]->(lb), (s)-[:Tought\_To]->(lc);**

1. Single example of creating relationship between LabGroup X and all the rooms this group studies in: **Match (l:LabGroup{Name:"Group X"}), (r:Room) create (l)-[rel:In]->(r);**  Other group to room relationships were created in same way.
2. Single example of relationship creation between multiple timeslots that room can be assigned to and a single room: **Match (r:Room{Name:"Room X"}), (t:Timeslot) create (r)-[rel:At]->(t);**

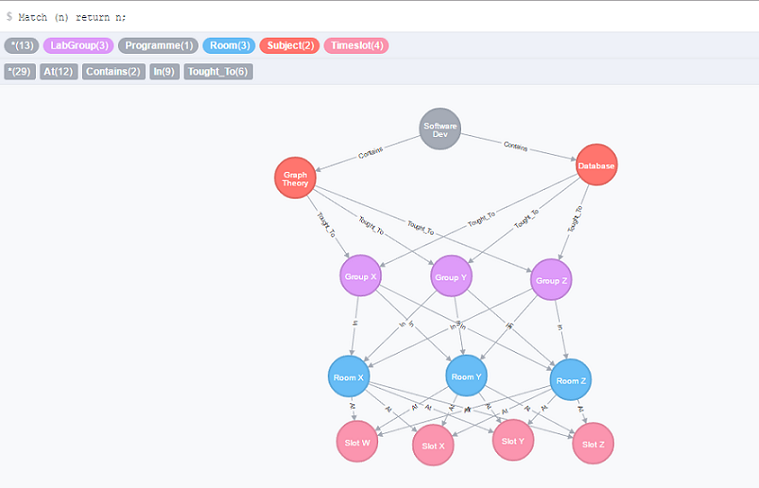
Also, part of database design and relationships coming out of my approach to it there are the several rules for timetable:

1. Same room can only be assigned to single timeslot at a time;
2. Same student group can be in only one room per timeslot;
3. Same student group can be in same room at different timeslots (Example of multi hour lectures/labs or different subjects being taught in same room to same group of students in succession).

Outcome of my approach:

Once implemented, the database has main logical relationships in place for easy tracking of what subject is being taught to which group in what room at the given timeslot. My target was to make the timetable easy to navigate and understand.

Final prototype of the database looks like this:



Observations and probable development beyond prototype stage:

* Restrictions are needed for the relationships. Once room has been assigned to timeslot for a specific group, there should be no way for other group to book same room on same timeslot.
* Programme unique lab group ID’s should be introduced once timetable moves beyond single programme. This is due to fact that different programmes have different number of groups (due to difference in student numbers studying given course).
* Some subjects are taught in multiple programmes at the same time. This could mean introducing unique ID for subjects as well.
* For added logic once programmes from different schools (like science, engineering or arts) are added the supervising departments can be added as nodes above programme nodes. Possible relationship between department and programme nodes can be “Supervises” as departments usually supervise programmes that are being taught in their respective schools in college.