1. **Database Comparisons**

Neo4j is a graphical database which means it uses graph structures for queries using nodes and edges to represent and store data.

A relational database such as SQL stores highly structured data in tables with columns that are predetermined to be of a specific type and with rows of the same type of information.

* 1. **Reference To other data**

With a relational database such as SQL reference data in other tables and rows are made by referring the primary keys of their data using a foreign key column. For many to many relationships a joining table would have to be used to hold the foreign keys.

The neo4j graph database has nodes which hold the different properties. These properties are stored as key value pairs. To provide connection between the different nodes relationships are used which are direct connections between two different nodes.

Using a relational database instead of neo4j would thus require us to use foreign key columns instead of the simple relationships used. This would result in more complex and more resource expensive system.

* 1. **Queries**

Neo4j uses Cypher which is a declarative graph query language to handle the database queries. Cypher is based on basic concepts of SQL. The syntax is more concise than that of SQL queries.

* 1. **Creating new users**

Creating a new user in Cypher:

CREATE (u:User {username:”UserOne”, password:”12345”})

RETURN u

Creating a new user in SQL:

INSERT INTO User (username, password)

VALUES (“userOne”, “12345”)

From the above syntaxes it can be seen that both methods would result in using generally simple and concise code.

* 1. **Relationship Checks**

A cypher query to return topics liked by a user looks like the following

MATCH (u : User) – [:LIKES] -> (n : Topic)

WHERE u.username = {user}

RETURN n

In SQL querying for topics liked by a user would require a much more complex syntax as is shown in the example query below:

SELECT topic FROM user

LEFT JOIN User\_Topic

ON User.Id = User\_Topic.UserId

LEFT JOIN Topic

ON Topic.Id = User\_Topic.TopicId

Both queries would return all topics that the user likes. Thus having used a relational database would have resulted in having longer query statements.

* 1. **Editing a table**

In SQL editing a row of data in a table would correspond to editing a node in neo4j. Using the relational database the syntax would be as below:

UPDATE User

SET bio = “Insert a bio update”

WHERE username = “UserOne”

The syntax is quite similar to the one for editing a node in neo4j.

* 1. **Conclusion**

The above examples comparing the relational database (SQL) to the graphical database used (neo4j) show that for our application where relationships are important a relational database would have been much difficult to implement due to the use of foreign key columns. Some of the syntax would be similar between the two databases.