Introduction

- Any user that queries any type of database expects at least one record to be returned.
- If the results are not what is expected, altering the query is the next logical step, maybe changing the way an item is spelled or choosing a different department to query.
- This approach is similar to trial and error. These alterations to the query do not help the user to understand a database better.

RDBMS vs GraphDB

- Chapman et al. proposed the first ever approach to solve why queries in RDBMS.
- This approach had a linear flow since RDBMS is executed in a linear fashion
- Same approach cannot be applied to Graph DBs
- Vasilyeva et al. proposed a differential graph approach to solve why queries in graph databases.

Goal

The aim of this capstone is analyzing and implementing at least one state-of-the-art algorithm on ‘Why? Queries’ to provide insight to users about unexpected answers. Specifically Why Empty? Queries.

Solution Design Flow

1. For each node in the query graph, first find the mappings in the data graph
2. For each of those mappings run a DFS search with that node as the source vertex.
3. The DFS will run on the query graph and the data graph simultaneously, trying to find mappings which satisfy the query requirements.
4. We will return a graph when there are no more edges to left to traverse in the query graph.
5. We will replace our final result graph if our current graph is bigger.

This is repeated for all the nodes in the query graph.

Algorithm

- Chapman et al. proposed the first ever approach to solve why queries in RDBMS.
- This approach had a linear flow since RDBMS is executed in a linear fashion
- Same approach cannot be applied to Graph DBs
- Vasilyeva et al. proposed a differential graph approach to solve why queries in graph databases.

There are 4 main problems to the base algorithm.
- Problems 1 and 3 arise because the authors have made an assumption that nodes are unique. This is not the case with data sets like protein genome synthesis datasets.
- Problem 2 and 4 arises because the algorithm only looks forward.
- Problems 2 and 4 can be fixed by using a Restart strategy.

Conclusion

- Neo4j is not an ideal database for running Why Queries.
- Neo4j stores information in a compressed manner to lower the memory usage.
- For us, since we need to retrieve a lot of information, we need to make specific structures to perform operations, which increases the execution times as reflected in our results.
- Many more optimizations are possible to reduce the execution times.

References