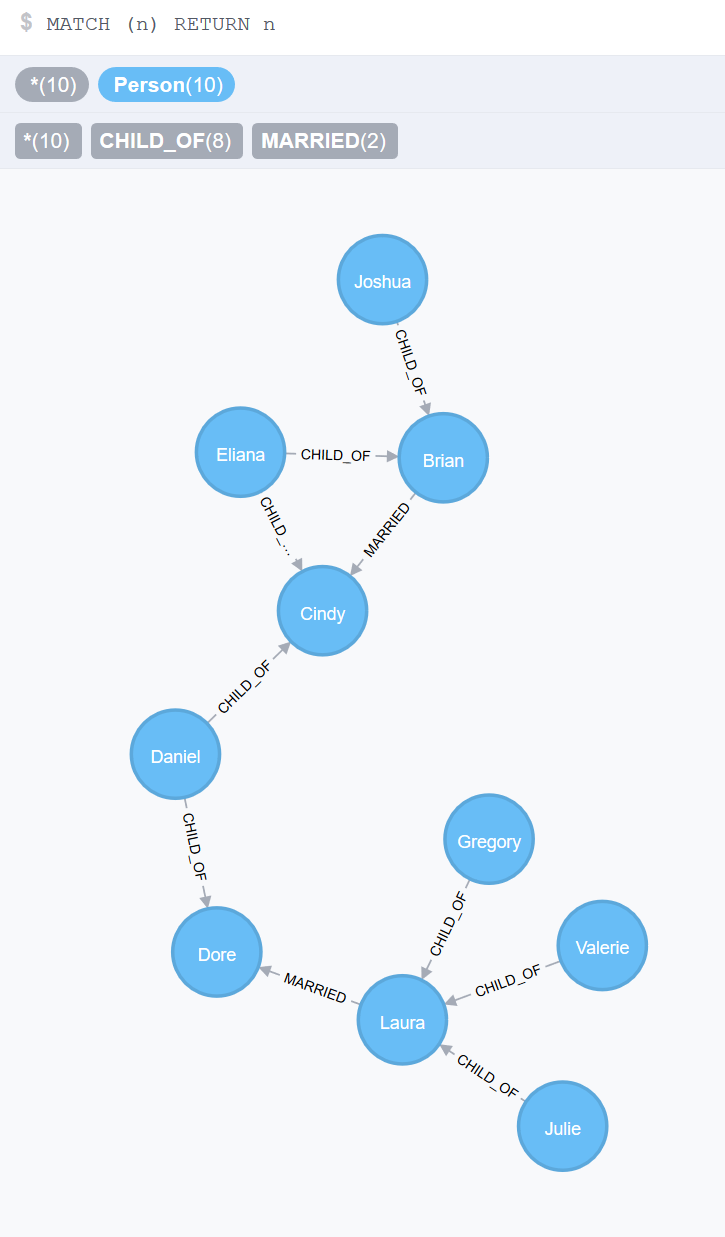
# Neo4j assignment – Daniel Teichman

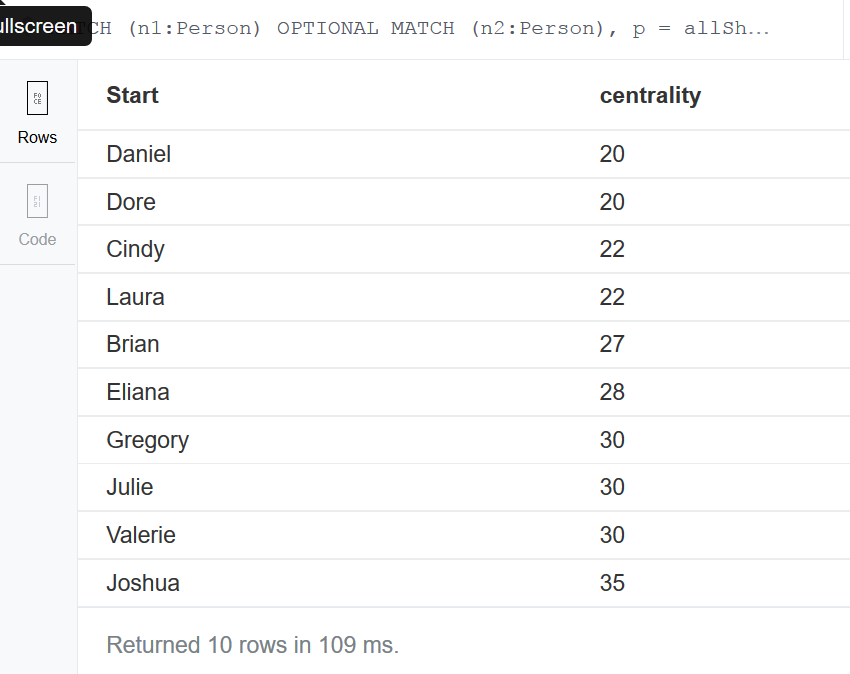
**2.1) Create people**  
CREATE (Daniel:Person {name: "Daniel"})  
CREATE (Dore:Person {name: "Dore"})  
CREATE (Laura:Person {name: "Laura"})  
CREATE (Julie:Person {name: "Julie"})  
CREATE (Valerie:Person {name: "Valerie"})  
CREATE (Gregory:Person {name: "Gregory"})  
CREATE (Cindy:Person {name: "Cindy"})  
CREATE (Brian:Person {name: "Brian"})  
CREATE (Joshua:Person {name: "Joshua"})  
CREATE (Eliana:Person {name: "Eliana"})

**2.2) Create Relationships**  
CREATE  
(Daniel)-[:CHILD\_OF]->(Dore),  
(Daniel)-[:CHILD\_OF]->(Cindy),  
(Eliana)-[:CHILD\_OF]->(Cindy),  
(Eliana)-[:CHILD\_OF]->(Brian),  
(Joshua)-[:CHILD\_OF]->(Brian),  
(Julie)-[:CHILD\_OF]->(Laura),  
(Valerie)-[:CHILD\_OF]->(Laura),  
(Gregory)-[:CHILD\_OF]->(Laura),  
(Laura)-[:MARRIED]->(Dore),  
(Brian)-[:MARRIED]->(Cindy)

**2.3) View graph of family**  
MATCH (n) RETURN n



**3.x) Get centrality score for all nodes (lowest score is most central)**  
MATCH (n1:Person)  
OPTIONAL MATCH (n2:Person), p = shortestPath((n1)-[\*]-(n2))  
RETURN n1.name as Start, sum(length(p)) as centrality  
ORDER BY centrality, n1.name



**x) Clean up graph**  
MATCH (a) OPTIONAL MATCH (a)-[r1]-() DELETE a,r1

**=======Discussion Question=======**

It is not possible to have a constant time traversal.

Imagine, if you will, a graph of n nodes arranged in a straight line where every intermediate node has an in-degree of 1 and an out-degree of 1; and the first node in the graph has an in-degree of 0 and an out-degree of 1; and the last node in the graph has an in-degree of 1 and an out-degree of 0.

Example: O->O or O->O->O or O->O->O->O etc.

Traversing such a graph completely from the first node requires n-1 hops.  Therefore, it is trivial to determine that the best-case, average-case, and worst-case complexity for visiting each node is n and therefore not constant time.

Such is the definition of traversal in graph theory:

[http://www.merriam-webster.com/dictionary/traverse (Links to an external site.)](http://www.merriam-webster.com/dictionary/traverse)

It \*is\* however, possible to use hashmaps to index the path(s) between any two nodes in the graph.  Because looking up a value in a hashmap with few or no collisions can be considered to have a complexity of O(1), it could be argued that the state of a graph traversal is accessible with O(1) complexity.

The downside of such a technique would be that the indexes would need to be updated for every node every time a relationship is added, changed, or removed.

This could be a useful technique in cases where the overhead of maintaining such an index is not too costly, when updates are so infrequent that updating the index is less costly than the saved on all of the "traversals."