LEGIS-GRAPH DATABASE

Legis-graph use case with Neo4j sandbox

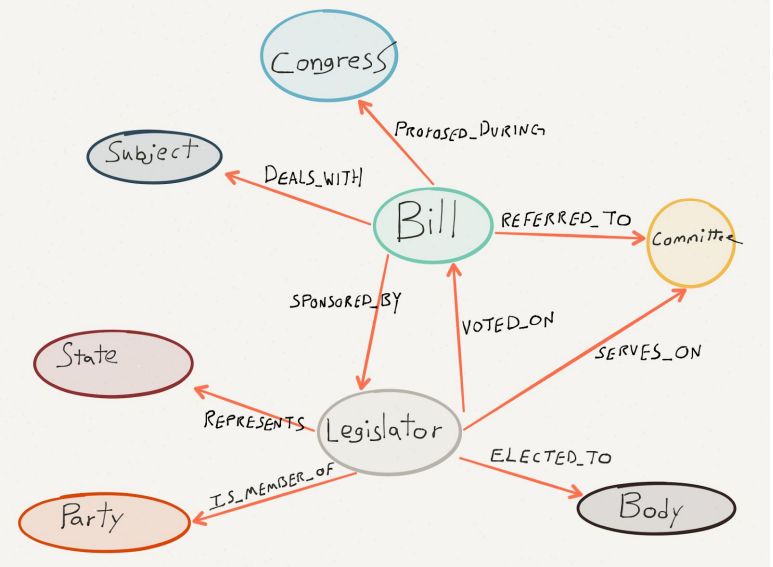
In this paper we will take a look at the Legis-graph sandbox, by Neo4j. Understanding how the database is structured can be achieved running some simple queries or using the panel to explore the graph and view the relationships.

After that we can start digging into the exercises proposed by Neo4j itself. Those contain a set of different interrogations that helped getting to know Cypher, discovering some interesting features such as the *apoc* functions.

EXPLORING THE DATABASE

Once created and opened the sandbox through Neo4j browser, we can take a look at the data. It contains a graph model of the US congress, including Bills, Legislators and Committee.

These are the relationships between all the nodes:



The Explore section shows some basic and other more complex queries, demonstrating how legislators, bills and committees are connected. We can also have a panoramic of the topics related to any legislator.

We could run this query if we wanted to know each *Legislator* that represents New York, elected to which *Body* (*House* or *Senate*) and member of which *Party* (*Democrat* or *Republican*)

```
MATCH (s:State)<-[:REPRESENTS]-(l:Legislator)
WHERE s.code = "NY"
MATCH (p:Party)<-[:IS_MEMBER_OF]-(l)-[:ELECTED_TO]->(b:Body)
RETURN l.lastName, p.name, b.type
```

We can also use Cypher syntax to study and analyze the dataset. The following two queries show the structure of one *Committee* node,

```
MATCH (n:Committee) RETURN n LIMIT 1
```

and the existing types between all the committees.

```
MATCH (n:Committee) RETURN distinct(n.type)
```

EXERCISES

After a few very basic queries (not reported in this paper), the first one states:

1. Find the bills referred to the House Committee on Agriculture that mention livestock in their title

```
MATCH (b:Bill)-[:REFERRED_TO]->(c:Committee)
WHERE b.officialTitle CONTAINS 'livestock'
    AND c.name = 'House Committee on Agriculture'
RETURN b as Bill, c as Committee
```

2. Who are the legislators who represent NY?

```
MATCH (l:Legislator)-[:REPRESENTS]->(s:State)
WHERE s.code = 'NY'
RETURN l
```

3. What political parties do they represent?

Here we add the relationship from legislators to parties

```
MATCH
(p:Party)<-[:IS_MEMBER_OF]-(l:Legislator)-[:REPRESENTS]->(s:State)
WHERE s.code = 'NY'
RETURN l,p
```

4. How many NY Democrats are serving in the House?

A third relationship is required to get the information about the body type

```
MATCH
(p:Party)<-[:IS_MEMBER_OF]-(l:Legislator)-[:REPRESENTS]->(s:State)
WHERE p.name = 'Democrat' AND s.code = 'NY'
MATCH (l)-[:ELECTED_TO]->(b:Body {type: 'House'})
RETURN COUNT(l)
```

5. For the legislators representing NY, what are the Committees on which they serve?

Version 1: see the graph containing all NY legislators connected to their committees

```
MATCH (l:Legislator)-[:REPRESENTS]->(s:State)
WHERE s.code = 'NY'
MATCH (l)-[:SERVES_ON]->(c:Committee)
RETURN l,c
```

Version 2: return a list of all the different committees

```
MATCH (l:Legislator)-[:REPRESENTS]->(s:State)
WHERE s.code = 'NY'
MATCH (l)-[:SERVES_ON]->(c:Committee)
RETURN DISTINCT(c.name)
```

6. What are the subjects of the bills referred to these committees?

The queries are getting more complex, therefore more tuning is required for an optimal result.

Version 1: no usage of *DISTINCT* clause

```
MATCH (l:Legislator)-[:REPRESENTS]->(s:State)
WHERE s.code = 'NY'
MATCH (l)-[:SERVES_ON]->(c:Committee)
WITH c
MATCH (c)<-[:REFERRED_TO]-(:Bill)-[:DEALS_WITH]->(sub:Subject)
RETURN sub.title
```

Started streaming 362244 records after 1 ms and completed after 5731 ms, displaying first 1000 rows.

Version 2: Using *DISTINCT* in the return section is ~ 10 times faster. That means the subjects are very redundant (we could have guessed it).

```
MATCH (l:Legislator)-[:REPRESENTS]->(s:State)
WHERE s.code = 'NY'
MATCH (l)-[:SERVES_ON]->(c:Committee)
WITH c
MATCH (c)<-[:REFERRED_TO]-(:Bill)-[:DEALS_WITH]->(sub:Subject)
RETURN DISTINCT sub.title
```

Started streaming 872 records after 1 ms and completed after 643 ms.

Version 3: Here we use *DISTINCT* also in the *WITH* section, in this way we pass less committees to the next part of the query, speeding it up by ~ 5 times.

```
MATCH (l:Legislator)-[:REPRESENTS]->(s:State)
WHERE s.code = 'NY'
MATCH (l)-[:SERVES_ON]->(c:Committee)
WITH DISTINCT c
MATCH (c)<-[:REFERRED_TO]-(:Bill)-[:DEALS_WITH]->(sub:Subject)
RETURN DISTINCT sub.title
```

Started streaming 872 records after 1 ms and completed after 116 ms.

The three versions have been executed a few times to check the expected difference of execution time, and the results were consistent.

7. Who sponsors the most bills?

Return the legislator who sponsored the most bills and how many of them
Version 1:

```
MATCH (b:Bill)-[:SPONSORED_BY]->(l:Legislator)
RETURN l, COUNT(b) AS bills
ORDER BY bills DESC LIMIT 1
```

Version 2: The function *apoc.agg.maxItems()* stores the maximum value, along with the items that match that value.

```
MATCH (b:Bill)-[:SPONSORED_BY]->(l:Legislator)
WITH l, COUNT(b) AS bills
WITH apoc.agg.maxItems(l,bills) AS max
RETURN max.items AS Legislator, max.value as Bills
```

8. Who sponsors the most bills for the subject "News media and reporting"?

Version 1:

```
MATCH
(s:Subject)<-[:DEALS_WITH]-(b:Bill)-[:SPONSORED_BY]->(l:Legislator
)
WHERE s.title = "News media and reporting"
RETURN l, COUNT(b) AS bills
ORDER BY bills DESC LIMIT 1
```

Version 2: With this interrogation we have a better understanding of the apoc function. Here the max number of bills dealing with the subject is 6. But in fact there are two legislators who sponsored 6 bills, and in this way we can see both of them.
UNWIND is only used for better presentation.

```
MATCH
(s:Subject)<-[:DEALS_WITH]-(b:Bill)-[:SPONSORED_BY]->(l:Legislator
)
WHERE s.title = 'News media and reporting'
WITH l, COUNT(b) AS bills
WITH apoc.agg.maxItems(l,bills) AS max
UNWIND max.items AS Legislator
RETURN Legislator, max.value as Bills
```

9. Which legislators frequently sponsor bills together?

Show the ten couples who sponsored the most bills together. The *WHERE* clause using *ID* properties is essential to avoid the same couple to be returned twice.
EX.
Grijalva - Norton - 453
Norton - Grijalva - 453

```
MATCH
(l1:Legislator)<-[:SPONSORED_BY]-(b:Bill)-[:SPONSORED_BY]->(l2:Leg
islator)
WHERE ID(l1) < ID(l2)
RETURN l1, l2, COUNT(*) AS bills
ORDER BY bills DESC LIMIT 10
```

10. Choose a specific legislator and find the subjects of the bills this legislator sponsors. What are the most common subjects?

Legislator chosen is Maria Cantwell

```
MATCH
(l:Legislator)<-[:SPONSORED_BY]-()-[:DEALS_WITH]->(s:Subject)
WHERE l.lastName = "Cantwell"
WITH s, COUNT(s) AS count
RETURN s, count ORDER BY count DESC LIMIT 25
```

11. **(Bonus)** Only include bills where this legislator was the main sponsor.

```
MATCH (l:Legislator)<-[:SPONSORED_BY {cosponsor :
false}]-(b:Bill)-[:DEALS_WITH]->(s:Subject)
WHERE l.lastName = "Cantwell"
WITH s, COUNT(s) AS count
RETURN s, count ORDER BY count DESC LIMIT 25
```