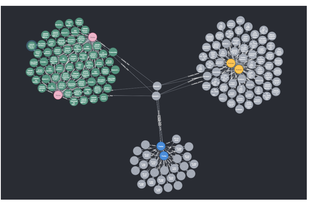
In [ ]:
```
#Vamos a crear las relaciones del neo4j, usando la libreria GraphDatabase para la ejecucion de las consultas y las relaciones
#que tiene cada nodo
```

Jonathan Atancuri

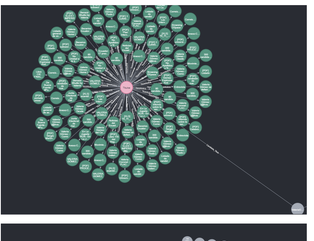
Generar un modelo que permita obtener y almacenar los datos en los grafos.

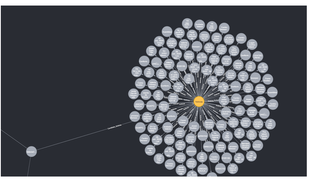
En este vamos a almacenar los datos en la base de Graphos neo4j, mediante la tecnica del webscraping Vamos a extraer lo que serian los datos de la asamblesita, el cual se crearia las respectivas realciones. En esta caso escogi la Asamblesat Valeria Gomez para el ejemplo

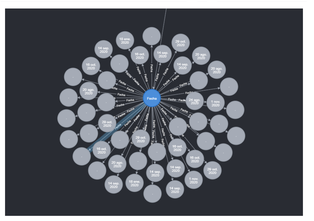
Vincular los datos con el candidato seleccionado.



En este caso tenemos la realacion que se puede observar , y que pertencen al nodo ValeriaGomez en este caso las noticias , titulo y la fecha



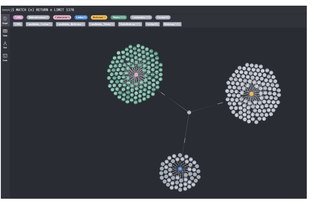




Se debe tener al menos 1000 nodos generados, en caso de tener estos datos seleccionar al partido (UNES) (UNES AZUAY) (Candidato Presidencial)

Como podemos observar tenemos los 1000 nodos. En este caso se eligio la asamblesita Valeria Gomez para analizar sus datos

Observamos nodos que estan relacionados con la asamblesita Valeria Gomez , sus nodos suesesores son la noticia, titulo,fecha los cuales es informacion que le pertenece a la asamblesita



Conclusiones

Con esto determinamos que los grafos nos ayuda a determinar los datos que existen , mediante la tecnica del webscraping pudimos obtener esta informacion y poder concretar que la herramienta de python + neo4j es muy util para poder ver datos del mundo real y es muy facil detectar los datos que este relacionados con cada nodo

Recomendaciones

Contar con python para poder realizar el proceso de webscraping

Tener neo4j para la ejecucion de las graficas

Tener conocimiento en consultas Cypher el cual es el lenguaje que usa neo4j

Tener conocimiento en Python

In [ ]:
```
#Se usa la libreria GraphDatabase para hacer las realciones y el grafico de los nodos en la base orientada a grafos
```

In [ ]:
```
from neo4j import GraphDatabase

class Neo4jService(object):

    def __init__(self, uri, user, password):
        self._driver = GraphDatabase.driver(uri, auth=(user, password))

    def close(self):
        self._driver.close()

    def crear_cabecera(self, tx, nombre):
        tx.run("CREATE (:Cabecera {nombre: $nombre})", nombre=nombre)

    def crear_candidata(self, tx, nombre):
        tx.run("CREATE (:ValeriaGomez {nombre: $nombre})", nombre=nombre)

    def crear_noticias(self, tx, nombre):
        tx.run("CREATE (:Noticias {nombre: $nombre})", nombre=nombre)

    def crear_links(self, tx, nombre):
        tx.run("CREATE (:Links {nombre: $nombre})", nombre=nombre)

    def crear_contenido(self, tx, nombre):
        tx.run("CREATE (:Contenido {nombre: $nombre})", nombre=nombre)

    def crear_titulo(self, tx, nombre):
        tx.run("CREATE (:Titulo {nombre: $nombre})", nombre=nombre)

    def crear_fecha(self, tx, nombre):
        tx.run("CREATE (:Fecha {nombre: $nombre})", nombre=nombre)

    def crear_relacion_noticias(self, tx, nombre_noticias, nombre_noticias):
        tx.run("MATCH (a:Noticias {nombre: $nombre_noticias}) "
               "MATCH (b:Contenido {nombre: $nombre_noticias}) "
               "MERGE (a)-[:Noticias]->(b)",
               nombre_noticias=nombre_noticias, nombre_noticia=nombre_noticias)

    def crear_relacion_cabeceras(self, tx, nombre_cabecera, nombre_titulo):
        tx.run("MATCH (a:Cabecera {nombre: $nombre_cabecera}) "
               "MATCH (b:Titulo {nombre: $nombre_titulo}) "
               "MERGE (a)-[:TituloNoticia]->(b)",
               nombre_cabecera=nombre_cabecera, nombre_titulo=nombre_titulo)

    def crear_relacion_fecha(self, tx, nombre_links, nombre_fecha):
        tx.run("MATCH (a:Links {nombre: $nombre_links}) "
               "MATCH (b:Fecha {nombre: $nombre_fecha}) "
               "MERGE (a)-[:Fecha]->(b)",
               nombre_links=nombre_links, nombre_fecha=nombre_fecha)

    #relacion de candidata  cabecera links noticias
    def crear_relacion_candidata_titulo(self, tx, nombre_candidata, nombre_titulo):
        tx.run("MATCH (a:ValeriaGomez {nombre: $nombre_candidata}) "
               "MATCH (b:Cabecera {nombre: $nombre_titulo}) "
               "MERGE (a)-[:Candidata_Titulo]->(b)",
               nombre_candidata=nombre_candidata, nombre_titulo=nombre_titulo)

    def crear_relacion_candidata_fecha(self, tx, nombre_candidata, nombre_fecha):
        tx.run("MATCH (a:ValeriaGomez {nombre: $nombre_candidata}) "
               "MATCH (b:Fecha {nombre: $nombre_fecha}) "
               "MERGE (a)-[:Candidata_Fecha]->(b)",
               nombre_candidata=nombre_candidata, nombre_fecha=nombre_fecha)

    def crear_relacion_candidata_contenido(self, tx, nombre_candidata, nombre_noticias):
        tx.run("MATCH (a:ValeriaGomez {nombre: $nombre_candidata}) "
               "MATCH (b:Noticias {nombre: $nombre_noticias}) "
               "MERGE (a)-[:Candidata_Noticias]->(b)",
               nombre_candidata=nombre_candidata, nombre_noticias=nombre_noticias)

print("Creacion de Nodos")
```

Codigo Fuente

In [ ]:

```python
#Vamos a realizar el webscraping con la libreria BeatifulSoup
import requests
from bs4 import BeautifulSoup
neo4j = Neo4jService('bolt://localhost:7687', 'neo4j', 'root')
with neo4j._driver.session() as session:

    session.write_transaction(neo4j.crear_candidata , "ValeriaGomez")
    session.write_transaction(neo4j.crear_cabecera , "Titulos")
    session.write_transaction(neo4j.crear_links , "Fecha")
    session.write_transaction(neo4j.crear_noticias , "Noticias")
linea = "C:/Users/jhonn/OneDrive/Documentos/Asambleista.txt"
salida =  "C:/Users/jhonn/OneDrive/Documentos/salida.txt"
#convierte un  archivo de texto en una lista
with open(linea,"r") as archivo, open(salida, "w") as salida:

    for linea in archivo:
        print(linea)

        i=0
        a=0
        e=0
        m=0

        #Solicita la URL
        page = requests.get(linea)

        #ve si la URL se ha codificado correctamente imprimir (r.url)
        r_linea = page.text

        #analiza la página de inicio descargada para obtener un objeto beautifulsoup
        new_xml = BeautifulSoup(r_linea, features = "xml").prettify()

        #Escribe una nueva lista en el directorio
        salida.write(new_xml)

        soup = BeautifulSoup(page.content,'html.parser')
        #print(soup.prettify())


        #result =soup.find_all("a href")
        #titulos
        titulos =soup.find_all("div",{"class":"BNeawe vvjwJb AP7Wnd"})
        #Contenido
        contenidohtml =soup.find_all("div",{"class":"BNeawe s3v9rd AP7Wnd"})
        #Fecha
        fecha =soup.find_all("span",{"class":"r0bn4c rQMQod"})

        print("resultado de la busqueda links ")
        a=0

        tituloNeo=list()
        for i in titulos:
            tituloNeo.append(i.text)

        descripcion=list()
        for i in contenidohtml:
            descripcion.append(i.text)
            #print(link)

#descripcion filtrada
        descripcionAux=list()
        for elemento in descripcion:
            if((a%2)==0):
                descripcionAux.append(descripcion[a])
            a=a+1


        fechaNeo=list()
        for i in fecha:
            fechaNeo.append(i.text)



        print("Titulo")
        for elemento in tituloNeo:
            session.write_transaction(neo4j.crear_titulo , elemento)
            print(elemento)

        #Descripcion print("descripcion")
        for elemento1 in descripcionAux:
            session.write_transaction(neo4j.crear_contenido , elemento1)
            print(elemento1)


        print("fecha")
        for elemento2 in fechaNeo:
            session.write_transaction(neo4j.crear_fecha , elemento2)
            print(elemento2)



#relaciones
        #relacion de descripcion
        for total in descripcionAux:
            session.write_transaction(neo4j.crear_relacion_noticia, "Noticias", total)

        #relacion de Fecha
        for date in fechaNeo:
            session.write_transaction(neo4j.crear_relacion_fecha, "Fecha", date)

        #relacion de Titulo
        for titul in tituloNeo:
            session.write_transaction(neo4j.crear_relacion_cabeceras, "Titulos", titul)


session.write_transaction(neo4j.crear_relacion_candidata_titulo, "ValeriaGomez", "Titulos")
session.write_transaction(neo4j.crear_relacion_candidata_contenido, "ValeriaGomez", "Noticias")
session.write_transaction(neo4j.crear_relacion_candidata_fecha, "ValeriaGomez", "Fecha")
print("La Ejecucion termino")
```

In [ ]: