Sistema de Recomendación y Busqueda con Neo4j y Java

Nicolas Añazco, Jordan Murillo
Sistemas, Universidad Politécnica Salesiana, UPS
Cuenca, Ecuador
jmurillov1@est.ups.edu.ec
nanazco@est.ups.edu.ec

The present work tries to demonstrate the uses that can be given to the DB Neo4j based on graphs and the usefulness of the algorithms that its libraries have. Furthermore, it aims to find new ways to solve problems by applying new forms of technology.

***Index Terms*—Neo4j, databases, algorithms, graphs**

I. Introducción

En la actualidad es muy común que salgan nuevas plataformas para el almacenamiento de datos, ese es el caso de Neo4j una nueva base de datos que se separa de sql para trabajar, ademas presentar una nueva forma de representar los datos mediante grafos de nodos que presentan muchas ventajas tanto en visualizado como para realizar calculos, esta base tiene unas librerias donde implementa algunos algoritmos muy utíles que pueden ser usados para hacer cálculos y retornar datos como predicciones, rutas, entre otros. Cabe agregar que la base tiene los drivers para funcionar con los lenguajes de programación más comunes como Java, Python, entre otros.

II. Desarrollo

A. Metodología

Para el desarrollo de la investigación se hizo uso de la base de datos Neo4j conjuntamente con la documentación que esta provee, en relación a los aspectos principales para almacenamiento de datos y uso de los algoritmos que esta posee. Ademas de contar con acceso a una maquina con Neo4j instalado y configurado. Para hacer uso de esta herramienta eligimos previamente de todos los algoritmos que posee, uno de cada tipo en total son cinco que seran usando para dar solución a un problema planteado. Para la informacion a guardar en la base se elgio a una de las principales ciudades del Ecuador como lo es Guayaquil, de la cual sacamos informacion refrente a: parques, iglesias, hospitales, bomberos, policía, escuelas, centros médicos, lugares turísticos, centros de estimulación temprana y centros educativos para el desarrollo de niños.

Luego de obtener estos datos los colocamos en un archivo de excel para su posterior guardado en la base de datos. Para la toma de datos se hara uso de Google Maps, con la cual calcularemos los datos necesarios para guardar. Además al hacer esto tendremos una vista previa de como mas o menos quedaría en la base.

Fig. 1. Tomado de Datos con Google Maps

1) Descripción detallada del problema.

El problema que planteamos fue que a veces se quiere buscar datos sobre algo, en nuestro caso un lugar, entonces para eso ello a veces se busca datos de ese lugar de diferentes tipos de fuentes, pero no siempre encontramos la informacion de forma rapida y bien sintetizada es por ello que hemos creado un prototipo de un progrma que segun los datos pasados me devuelve datos muy importantes y especificos de lo que buscamos, consiste en un sistema de recomendación y de búsqueda que mediante la base de datos Neo4j. Este problema es común porque hay personas que quieren viajar pero no pueden decirdirce a veces, por eso que mejor que decidir de una forma rapida al seleccionar unos datos muy simples.

2) Propuesta de Solución

Para la solución del problema como se dijo anteriormente diseñamos el prototipo de un programa que nos ayuda a recomendar y buscar lugares que se envías como entradas de texto.

La solución consiste en un programa desarrollado en Java que hace uso de la base de datos Neo4j, tanto para el almacenamiento de datos y uso de algoritmos presentes en esta que sirven para determinar distintos datos que sirven para buscar o recomendar algo en específico.

Además con los resultados obtenidos se pueden hacer gráficas estadísticas que ayudan a interpretar mejor los datos y de esta manera dar un toque mas especial al programa.

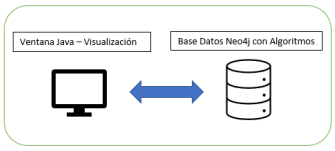

Fig. 2. Esquema del Programa

El esquema de la aplicacion es algo sencillo, pero la importancia radica en el uso de los algoritmos que provee Neo4j para dar solución al problema, que mas adelante explicaremos adecuadamente.

A continuación se muestra un prototipo de la parte gráfica de el sistema.

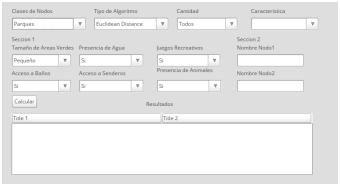

Fig. 3. Sistema de Recomendación

Fig. 4. Sistema de Búsqueda

Ahora vamos a presentar la arquitectura del programa desarrollado, para ello podemos fijarnos en la siguiente imagen.

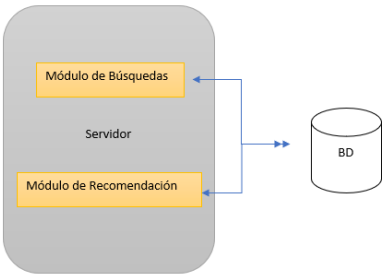

Fig. 5. Arquitectura

3) Descripción de la Solución

Para diseñar la solución, primero estudiamos y entendimos los algoritmos a ser usados, los cuales explicaremos a continuación.

A. Algoritmos de Centralidad

Para es tipo de algoritmos elegimos al Centralidad de Grados, el cual nos ayuda a encontrar el numero de nodos a los que estamos conectados y de los que nos conectamos, es decir, la centralidad de grados mide el número de relaciones entrantes y salientes de un nodo. El algoritmo DC puede ayudarnos a encontrar nodos populares en un gráfico.

```
CALL gds.alpha.degree.stream({
  nodeProjection: 'User',
  relationshipProjection: {
    FOLLOWS: {
      type: 'FOLLOWS',
      projection: 'REVERSE'
    }
  }
})
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score AS followers
ORDER BY followers DESC
```

Fig. 6. Codigo Sencillo DC

B. Algoritmos de Detección de Comunidad.

Para es tipo de algoritmos elegimos al de Optimización de Modularidad, el cual nos ayuda a identificar el Id de Comunidad de los nodos en base a calculos sobre la densidad de los nodos.El algoritmo de optimización de modularidad intenta detectar comunidades en el gráfico en función de su modularidad. La modularidad es una medida de la estructura de un gráfico, que mide la densidad de las conexiones dentro de un módulo o comunidad.

```
CALL gds.beta.modularityOptimization.stream('myGraph', { relationshipWeightProperty: 'weight' })
YIELD nodeId, communityId
RETURN gds.util.asNode(nodeId).name AS name, communityId
ORDER BY name
```

Fig. 7. Codigo Sencillo MO

C. Algoritmos de Similitud

Para es tipo de algoritmos elegimos al de Distancia Euclideana, esta mide mediante una formula parecida a la de Pitagoras la distancia entre nodos, es decir, la distancia euclidiana mide la distancia en línea recta entre dos puntos en el espacio n-dimensional.

```
MATCH (p1:Person {name: 'Zhen'})-[likes1:LIKES]->(cuisine)
MATCH (p2:Person {name: 'Praveena'})-[likes2:LIKES]->(cuisine)
RETURN p1.name AS from,
       p2.name AS to,
       gds.alpha.similarity.euclideanDistance(collect(likes1.score), collect(likes2.score)) AS similarity
```

Fig. 8. Codigo Sencillo ED

D. Algoritmos de Ruta/Camino

Para es tipo de algoritmos elegimos al de Depth First Search, este busca la ruta de un lugar a otro en base a la profundidad del grafo, es decir, el algoritmo Depth First Search es un recorrido de gráfico que comienza en un nodo dado y explora lo más posible a lo largo de cada rama antes de retroceder. Un algoritmo relacionado es el algoritmo Breath First Search.

```
MATCH (a:Node{tag:'a'})
WITH id(a) AS startNode
CALL gds.alpha.dfs.stream('myGraph', {startNode: startNode})
YIELD path
UNWIND [ n in nodes(path) | n.tag ] AS tags
RETURN tags
ORDER BY tags
```

Fig. 9. Codigo Sencillo DFS

E. Algoritmos de Predicción de Enlaces

Para es tipo de algoritmos elegimos al de Misma Comunidad este compara a través de un característica si dos nodos en este caso pertenecen a la misma comunidad, es decir, la misma comunidad es una forma de determinar si dos nodos pertenecen a la misma comunidad.

```
MATCH (p1:Person {name: 'Michael'})
MATCH (p2:Person {name: 'Zhen'})
RETURN gds.alpha.linkprediction.sameCommunity(p1, p2) AS score
```

Fig. 10. Codigo Sencillo SC

Bueno esas son las descripciones de los algoritmos que elegimos, los pasos en Java para el desarrollo son muy simples ya que debes crear la conexion de Neo4j y Java para su trabajo.

Para el uso en Java solo necesitabamos instalar la libreria requerida de Neo4j y comenzar a trabajar. primero se crea una conexion con la base y luego se le pasa los datos a para establecer la comunicación, para llamar a ejecutar una accion se pasa un parametro de tipo texto con el query.

```
private final Driver driver;

public Metodos() {
    driver = GraphDatabase.driver("bolt://localhost:7687", AuthTokens.basic("neo4j", "5658"));
}

@Override
public void close() throws Exception {
    driver.close();
}

public List<Nodo> similitudEuclideanaL(int[] a) {
    try (Session session = driver.session()) {
        List<Nodo> vec = new ArrayList<>();
        String greeting = session.writeTransaction((Transaction tx) -> {
            Result result = tx.run("MATCH (n:LugarTuristico) "
                    + "UNWIND [n.tam,n.tiendas,n.averdes,n.aextremas,n.gal] AS v "
                    + "WITH n, collect(CASE v WHEN 'Grande'  THEN 3 "
                    + "WHEN 'Mediano' THEN 2 "
                    + "WHEN 'Pequeño'THEN 1 "
                    + "WHEN 'si'THEN 1 "
                    + "WHEN 'no'THEN 0 "
                    + " END) as vector "
                    + "RETURN n.nombre AS Lugar, "
                    + "gds.alpha.similarity.euclideanDistance($a,vector) as Sim ORDER BY Sim"
                    + " ASC", parameters("a", a));
            for (Record record : result.list()) {
                String nomi = record.get("Lugar").asString();
                Double costo = record.get("Sim").asDouble();
```

Fig. 11. Ejemplo de Neo4j en Java

Los pasos para estos algoritmos son muy sencillos, y se aprenden fácilmente. Cabe destacar que hay algunos errores en la documentación oficial pero solo se debe analizar bien la información y se entenderá correctamente.

Para el desarrollo creamos un proyecto en Netbeans para trabajar en Maven y obtener las librerías de forma adecuada y rápida, y porque tambien ayuda mucho a la hora de crear interfaces sencillas y de buena estetica.

Luego fuimos dandole forma a la propuesta de solución con la ayuda de Netbeans, creamos un proyecto con paquetes para subdividir en estos los diferentes archivos generados.

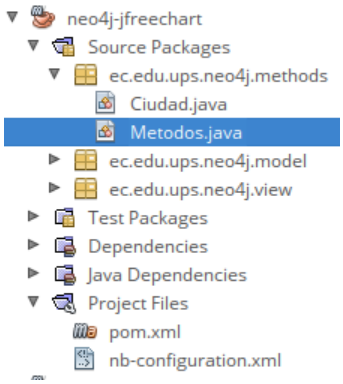

Fig. 12. Vista de Paquetes Java

Para retornar y guardar los datos que me devuelve el Neo4j en Java creamos una clase llamada Nodo. Esta guarda de forma adecuada los datos que vienen de Neo4j.

La clase Metodos tiene la conexion y los metodos necesarios con los algoritmos para hacer posible el proceso de llamado desde java a Neo4j.

```
public List<Nodo> mismaComunidadLT(String lugar, String carac) {
    try (Session session = driver.session()) {
        List<Nodo> vec = new ArrayList<>();
        String greeting = session.writeTransaction((Transaction tx) -> {
            Result result = tx.run("MATCH (p1:LugarTuristico {nombre:$lugar}) "
                    + "MATCH (p2:LugarTuristico) WHERE p1<>p2 "
                    + "RETURN p2.nombre as Nombre, gds.alpha.linkprediction.sam
            for (Record record : result.list()) {
                String nomi = record.get("Nombre").asString();
                Double costo = record.get("score").asDouble();
                Nodo ns = new Nodo(nomi, costo);
                vec.add(ns);
            }
            return null;
        });
        return vec;
    }
}
```

Fig. 13. Ejemplo Metodo desde Java Misma Comunidad

Ya solo queda las clases para la vista por lo que diseñamos tres clases una principal que es nuestra portada y dos internas una para el Sistema de Recomendación y otra para el Sistema de Búsqueda.

En la Principal solo se llama a cada ventana a hacerle visible, como pudimos ver anteriormente al inicio un prototipo del programa, al final no cambio mucho.

Además nosotros utilizamos todos los tipos de nodos para búsqueda y para recomendación solo los parques y lugares turísticos.

4) Resultado de análisis obtenidos

En base a lo obtenido podemos determinar que el uso de Neo4j simplifica bastante los procesos de desarrollo de algoritmos desde cero, además permite ampliar nuestra forma de como guardar datos. Tambien el desarrollo de programas o aplicaciones, ya que se enfoca en una nueva forma de guardar datos.

Ahora veremos al programa en ejecución:

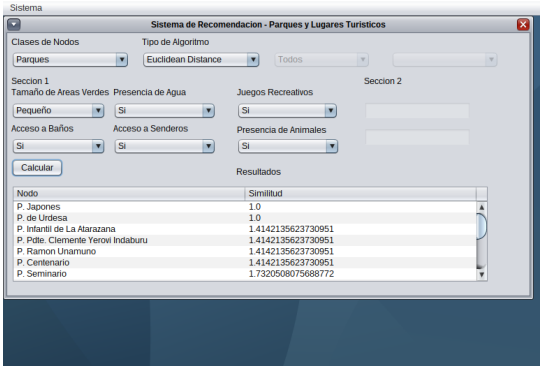

Fig. 14. Ejemplo Sistema de Recomendación - Resultados

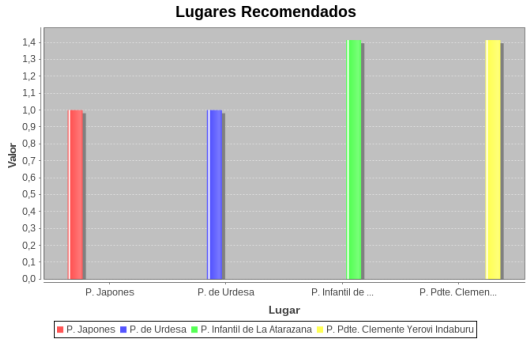

Fig. 15. Ejemplo Sistema de Recomendación - Gráfica

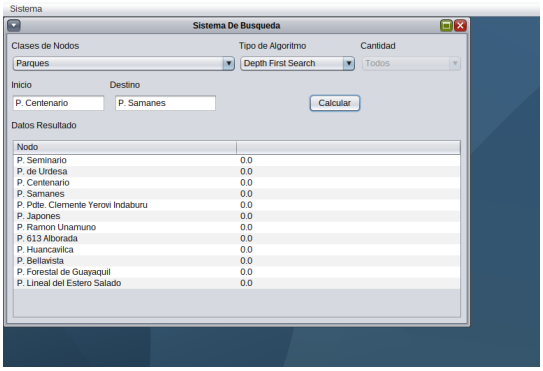

Fig. 16. Ejemplo Sistema de Búsqueda - Resultados

III. Opinion y Recomendaciones

Para nosotros Neoj4 es una forma de trabajar con los datos mas divertidad y se le puede dar varios usos nuevos que podrian ayudar en futuras investigaciones o trabajos.

Recomendamos probar Neoj4 porque presenta caracteristicas nuevas que las bases de datos tradicionales no tienen. Además, tiene una version libre y es muy fácil de instalar.

References

[1] BD Neoj4. *The Neo4j Graph Data Science Library Manual v1.2-preview* 2020. Disponible en: https://neo4j.com/docs/graph-data-science/1.2-preview/

[2] BD Neo4j *Using Neo4j from Java* 2020. Disponible en: https://neo4j.com/developer/java/