Ingeniería en Sistemas Computacionales

Minería de Grafos

Cálculo de distancia geoespacial

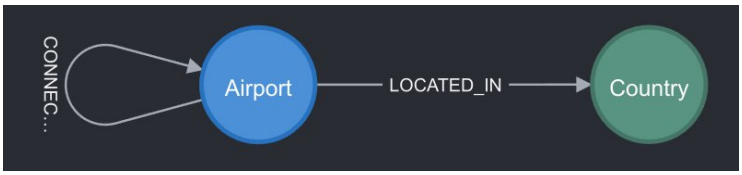
Marco Ricardo Cordero Hernández

Tlaquepaque, Jal., 16 de noviembre de 2023

En el contexto del análisis de datos dentro de colecciones relacionadas entre ellas mismas, usualmente se desprecia el potencial de un par de atributos contenidos dentro de una entidad, o más específicamente, un nodo para las bases de datos orientadas a grafos: los componentes geográficos. El posicionamiento de un objeto dentro de un sistema de coordenadas geográficas usualmente se representa como el par de valores *latitud* y *longitud*, es decir, un conjunto de líneas imaginarias dadas en medidas angulares desde el centro de la tierra.

En el contexto del curso actual, la herramienta *Neo4j* para manejo de bases de grafos, en conjunto de su extensión *NeoDash* para creación y visualización de tableros, pueden ser utilizados para crear, manipular, y ver distintos puntos de interés a nivel mundial, así como el cálculo de la distancia entre dos ubicaciones en el plano geoespacial (**Nota**: los términos geoespacial y geodésico se usarán como conceptos intercambiables a lo largo del documento).

Para demostrar el potencial del conjunto de características encontradas en la herramienta mencionada, un set de datos con contenido relacionado a aeropuertos previamente visto (práctica 2) nuevamente será utilizado. Como paso inicial, la estructura del grafo puede ser revisada:



Como se puede apreciar, únicamente existen dos tipos de nodos: aeropuerto y país; ambos pueden contener latitud y longitud como atributos. Para evitar ambigüedad, la siguiente consulta puede emitirse para obtener sus propiedades individuales.

```
CALL db.schema.nodeTypeProperties()
YIELD nodeType AS Nodo, propertyName AS Propiedad, propertyTypes AS Tipo;
```

Nodo	Propiedad	Tipo
":`Airport`"	"name"	["String"]
":`Airport`"	"id"	["Long"]
":`Airport`"	"latitude"	["Double"]
":`Airport`"	"longitude"	["Double"]

En este caso, el nodo aeropuerto cuenta con estas propiedades almacenadas como *double*, es decir, tipo flotante, ideal para el propósito actual. **Nota**: los resultados han sido cortados debido a la poca utilidad que su visualización otorga; los nodos de tipo país solo cuentan con su nombre como atributo.

Una vez que se tiene lo necesario para demostrar las capacidades espaciales de Neo4j, se proponen los siguientes ejercicios:

1. Crea la distancia geodésica desde el aeropuerto de Guadalajara hacia el aeropuerto de Puerto Vallarta

   Para atender este punto, primero es necesario saber *cómo* crear la distancia geodésica, e incluso antes de ello, saber *qué es* la distancia geodésica. Este término hace referencia a la línea que representa el camino más corto entre dos puntos en una superficie curva, como lo es la Tierra, y es una forma de mostrar la distancia en un elipsoide mientras que la misma medición se proyecta sobre una superficie plana.

   Una vez que el contexto ha sido proporcionado, se puede encontrar fácilmente que la función integrada dentro de Neo4j `point.distance()` funciona a la perfección para el propósito establecido. Esta función recibe como argumentos dos datos de tipo `POINT` con formato básico `point({latitude: 0.0, longitude: 0.0})`; para el caso, un par de argumentos adicionales *pueden* ser pasados también, de forma que se tiene el formato:

```
point({
    latitude: 0.0,
    longitude: 0.0,
    crs: 'WGS-84',  // Opcional
    srid: 4326      // Opcional
})
```

   Con este formato, se puede calcular la distancia geodésica con relativa facilidad.

   Ahora, para obtener lo que se pide en el ejercicio actual, el siguiente código puede utilizarse:

```
MATCH (a1:Airport {name: 'Guadalajara'}),
      (a2:Airport {name: 'Puerto Vallarta'})
WITH point({longitude: a1.longitude, latitude: a1.latitude}) as pt1,
     point({longitude: a2.longitude, latitude: a2.latitude}) as pt2
RETURN point.distance(pt1, pt2) AS Distancia;
// RETURN point.distance(pt1, pt2) / 1000 AS Distancia;
```

Distancia
203226.93411296894

Distancia
203.22693411296893

   Neo4j reconoce el sistema de los datos utilizados y retorna un valor utilizando *metros* como unidades, por lo cual se puede dividir el resultado original para convertirlo a kilómetros o la unidad que se desee dependiendo el caso.

   Utilizando Google Maps y sus herramientas de medición, es posible corroborar este dato:

Para este ejercicio, el resultado obtenido es bastante aproximado, teniendo tan solo un porcentaje de error del 0.215%. **Nota**: Los resultados de Google Maps también son aproximados, y no deberían tomarse como medidas exactas.

2. Crea la distancia geodésica desde el aeropuerto de Guadalajara hacia otro aeropuerto de tu elección

A manera personal, se ha de comentar que dos puntos de interés prevalecen para este segundo ejercicio: Tepic en Nayarit, y Tuxtla Gutiérrez en Chiapas. Ambos lugares tienen cierto significado especial, por lo que resultaría apropiado su uso en el presente, sin embargo, la elección de la primera opción supondría cierta similitud con el ejercicio anterior, por lo cual, la segunda resulta idónea. El código ya visto puede ser mínimamente modificado para obtener el resultado deseado:

```
MATCH (a1:Airport {name: 'Guadalajara'}),
      (a2:Airport {name: 'Tuxtla Gutierrez'})
WITH point({longitude: a1.longitude, latitude: a1.latitude}) as pt1,
     point({longitude: a2.longitude, latitude: a2.latitude}) as pt2
RETURN point.distance(pt1, pt2) AS Distancia;
// RETURN point.distance(pt1, pt2) / 1000 AS Distancia;
```

Distancia
1171443.8387038342

Distancia
1171.4438387038342

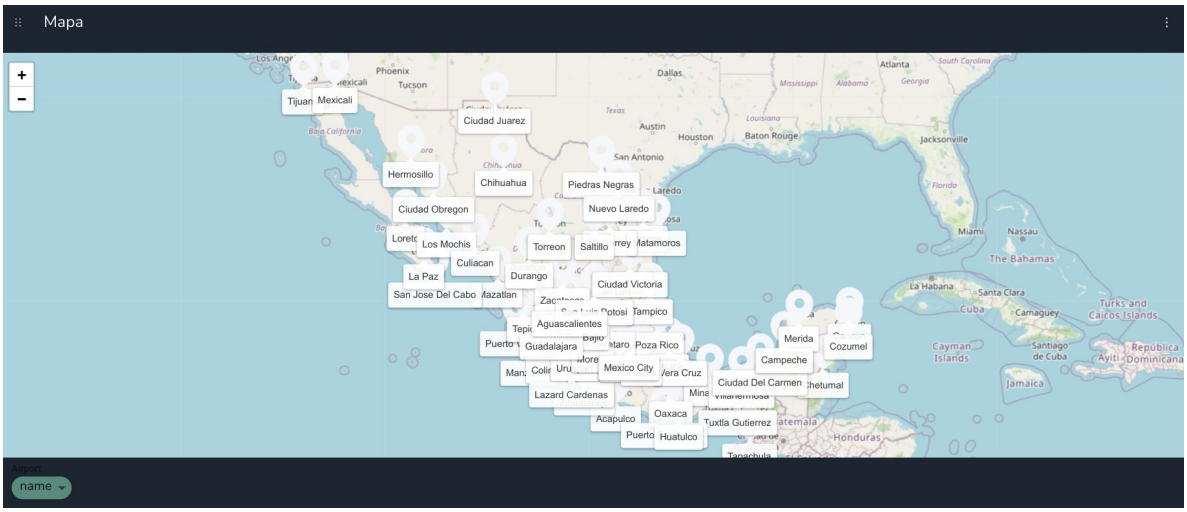
Nuevamente, Google Maps puede usarse como herramienta de comprobación.

En esta ocasión, el error es del 0.115%

3. Muestra el resultado en un mapa utilizando NeoDash

Finalmente, se utilizará un widget dentro de un tablero básico de NeoDash. Con la siguiente consulta es posible obtener todos los aeropuertos de México:

```
MATCH (a:Airport)-[:LOCATED_IN]->(c:Country)
WHERE c.name = 'Mexico'
RETURN a;
```



NeoDash permite la visualización de relaciones entre nodos, o en este caso, la conexión entre aeropuertos. El siguiente código incluye las distancias dentro de las relaciones:

```
MATCH (a1:Airport {name: 'Guadalajara'}),
      (a2:Airport {name: 'Puerto Vallarta'}) // y name: 'Tuxtla Gutierrez'
MERGE (a1)-[r:CONNECTS_WITH]->(a2)
ON MATCH
    SET
        r.distance = point.distance(
            point({longitude: a1.longitude, latitude: a1.latitude}),
            point({longitude: a2.longitude, latitude: a2.latitude})
        ) / 1000
ON CREATE
    SET
        r.distance = point.distance(
            point({longitude: a1.longitude, latitude: a1.latitude}),
            point({longitude: a2.longitude, latitude: a2.latitude})
        ) / 1000;
```

Luego, el siguiente código se usa para mostrar el mapa actualizado: