BDA01.- Tarea

En este documento tendréis que elaborar un documento, a modo de tutorial, que incluya toda la información (código, capturas de pantalla, resultados). En el aula virtual subid el documento en formato PDF.

Introducción

Yelp ayuda a las personas a encontrar negocios locales basándose en reseñas, preferencias y recomendaciones. A finales de 2018 se habían escrito más de 180 millones de reseñas en la plataforma. Desde 2013, Yelp ha organizado el reto Yelp Dataset, un concurso que anima a la gente a explorar e investigar el conjunto de datos abiertos de Yelp. A partir de la ronda 12 (realizada en 2018) del desafío, el conjunto de datos abiertos contenía:

- Más de 7 millones de reseñas más consejos
- Más de 1,5 millones de usuarios y 280.000 fotos
- Más de 188.000 negocios con 1,4 millones de atributos
- 10 áreas metropolitanas

Desde su lanzamiento, el conjunto de datos se ha hecho popular, con cientos de artículos académicos escritos utilizando este material. El conjunto de datos de Yelp representa datos reales muy bien estructurados y altamente interconectados. Es un gran escaparate para los algoritmos de grafos que también puedes descargar y explorar.

Caso práctico

Utilizaremos un caso práctico en el que trabajamos para una empresa de información de viajes. El objetivo es encontrar buenas recomendaciones de lugares para alojarse y cosas que hacer en ciudades importantes como Las Vegas.

Importación de datos

Los datos se encuentran en https://www.yelp.com/dataset

1. Escribir el comando de importación, recomendable usar LOAD_CSV.

```
./bin/neo4j-admin database import full --nodes
User="import/yelp_academicdataset_user.csv" --nodes
Business="import/yelp_academic_dataset_business.csv" --nodes
Review="import/yelp_academic_dataset_review.csv"
--ignore-empty-strings --delimiter="," --high-parallel-io=on
--additional-config=conf/neo4j.conf --overwrite-destination
--ignore-extra-columns


CREATE TEXT INDEX IX_User

FOR (n:User)

ON (n.user_id)


CREATE TEXT INDEX IX_Business

FOR (n:business)

ON (n.business_id)


call apoc.periodic.iterate("MATCH (r:review) MATCH (u:User
{u.user_id:r.user_id}) return u,r", " MERGE
(r)-[:REVIEWED]->(u)", { batchSize:20000, parallel: false })


call apoc.periodic.iterate("MATCH (r:review) MATCH
(b:business  {r.business_id:b.business_id}) return r", "
MERGE (r)-[:REVIEWS]->(b)", { batchSize:20000, parallel:
false })
```

2. Realizar un grafo del modelo.

```
CALL db.schema.visualization()
```

3. Describir los nodos y sus relaciones.

   Están los nodos Business, los Review y los User.

   User se relaciona con Review a través de REVIEWED.

   Review se relaciona con Business a través de REVIEWS.

Aplicación de viajes

En esta aplicación estamos interesados en encontrar los hoteles más importantes y los usuarios más importantes, existen una infinidad de maneras de obtenerlo pero en este caso nos basta con obtener la siguiente información:

1. Encuentra los 10 hoteles con mayor número de reviews.

```
MATCH (r:review)-[r1:REVIEWED]-(b:business)
RETURN b.name, count(r) AS rev
ORDER BY rev desc
LIMIT 10
```

2. Encuentra los 10 usuarios con un número mayor de reviews realizadas.

```
MATCH (usuario:User)
RETURN usuario
ORDER BY usuario.review_count DESC
LIMIT 10
```

Consultoría empresa de viajes

En este caso se ha detectado que los usuarios más influyentes son los que han ido al hotel Bellagio Hotel

1. Encuentra los 50 usuarios con mayor número de reviews que han hecho una review del hotel Bellagio Hotel.

```
MATCH (u:User)-[:REVIEWED]->(:Review)->[:REVIEWS]-(:Business{
name:"Bellagio Hotel"})
WITH u
ORDER BY u.review_count DESC
```

```
LIMIT 50

MATCH (u)-[:REVIEWED]->(Review)->[:REVIEWS]-(b:Business)

RETURN b.name AS Hotel
```

2. Buscar todos los hoteles que estos usuarios han hecho una review.

```
call {
  MATCH (u)-[]-(r:review)-[]-(b:business { name: "Bellagio Hotel" })
  RETURN u ORDER BY u.review_count desc
  LIMIT 50
}
WITH u
MATCH (u)-[]-(r:review)-[]-(b:business)
RETURN DISTINCT b.name
```

3. Obtén el hotel con mayor número de reviews de los usuarios obtenidos en el apartado 1 de esta sección

```
CALL {
  MATCH (u)-[]-(r:review)-[]-(b:business)
  WHERE b.name CONTAINS "Bellagio Hotel"
  RETURN u.user_id AS resultado ORDER BY u.review_count DESC
LIMIT 50 }
MATCH (uu)-[]-(rr:review)-[]-(bb:business)
WHERE uu.user_id IN resultado AND bb.categories CONTAINS "Hotel"
RETURN bb.name, bb.review_count ORDER BY bb.review_count DESC
LIMIT 1
```