

Proceso para migrar una base transaccional a una de grafos específicamente de postgresql a neo4j

Existen diferentes formas de migrar una base de datos transaccional a una de grafos, en esta ocasión se presenta dos maneras para hacerlo.

Primera opción:

- Realizar una consulta SQL seleccionando toda la información a migrar y luego exportar la misma a un archivo CSV.
- EN la base de datos receptora (Neo4J) se debe preparar una consulta tipo CYPHER con el uso de la herramienta **load csv** para importar todos los datos del csv creado anteriormente.

```
USING PERIODIC COMMIT
LOAD CSV WITH HEADERS FROM 'file:/translations.csv' AS row
MERGE (from:Phrase {
  externalId: row.id_from
})
ON CREATE SET
  from.uuid = randomUUID(),
  from.text = row.text_from,
  from.lang = row.lang_from
MERGE (to:Phrase {
  externalId: row.id_to
})
ON CREATE SET
  to.uuid = randomUUID(),
  to.text = row.text_to,
  to.lang = row.lang_to
MERGE (from)-[r:TRANSLATES {
  code: row.lang_from + '-' + row.lang_to
}]->(to);
```

Ejemplo de una consulta realizada en Cypher.

- Correr la consulta del primer paso
- Exportar el archivo csv creado a la carpeta de Neo4J

Segunda forma.

- Tener descargado el archivo APOC Kit de Neo4j en la ruta \$Neo4j_Home/plugins.
- Para poder migrar los datos de la base de datos PostgreSQL primero debemos tener descargado el archivo JDBC .jar y mantenerlo en la ruta anteriormente mencionada, una vez realizado esto, debemos reiniciar el servidor Neo4j.
- Una vez reiniciado el servidor, cargamos el driver con la ayuda de APOC que descargamos al inicio.

```
CALL apoc.load.driver('org.postgresql.Driver');
```

- Paso seguido creamos la llamada para obtener los datos de PostgreSQL para Neo4J

```
with 'jdbc:postgresql://localhost:5432/testdb?user=postgres&password=postgres' as url
CALL apoc.load.jdbc(url,'employee_details') YIELD row
RETURN count(*);
```

```
apoc.jdbc.postgresql_url.url=jdbc:postgresql://localhost:5432/testdb?user=
```

- Hecho esto ya podemos obtener los datos directamente.

```
CALL apoc.load.jdbc('postgresql_url','employee_details') YIELD row  
RETURN count(*);
```

- Realizado esto procedemos a definir el esquema y las llaves.

```
/**  
 * Here we define schema and key. In first column we define those  
 column_name  
 * which can be null and In the second we those column name which we  
 want unique.  
 */  
  
CALL apoc.schema.assert( {Detail:['name','age','address','salary']},  
 {Detail:['id']});
```

- Ahora, procedemos a cargar los datos en el Neo4j y creamos los nodos con ayuda del esquema definido anteriormente.

```
CALL  
apoc.load.jdbc('jdbc:postgresql://localhost:5432/testdb?user=postgres&pass  
word=postgres','employee_details') yield row  
CREATE (t:Detail {id:toString(row.id), name:row.name,  
age:toString(row.age), address:row.address, salary:toString(row.salary)})
```

- Una vez realizado el último paso ya tenemos importada nuestra base.

Referencias:

<https://eagertoit.com/2019/05/23/data-migration-postgres-neo4j/>

<https://blog.knoldus.com/neo4j-with-scala-migrate-data-from-other-database-to-neo4j/>