Universidad Central de Venezuela

Facultad de Ciencias

Escuela de Computación

Bases de Datos NoSQL

Proyecto - Fase II

Preparador:

Robinson Roa

Estudiantes:

Christian Padrón

C.I.V-26.996.952

Ismael Guerrero

C.I. V-28.073.233

Índice

Introducción

El trabajo se dividió en 2 partes, que vienen siendo la migración y las consultas, siendo Ismael el escritor principal del código de migración, mientras Christian se encargó principalmente de las consultas, sin embargo, también hubo tareas repartidas, como las pruebas de migraciones y consultas.

Explicación

Para la migración de datos de MongoDB a Neo4J se creó un fichero llamado migrarBD.js, el cual contiene una función llamada MigrateData() esta función se encarga de migrar todos los documentos presentes en la base de datos de mongoDb usada para la fase anterior del proyecto a una nueva base de datos Neo4J con las características especificadas en el enunciado, la cual es ejecutada al correr el archivo:

Al principio del código se preparan las conecciones:

```
require('dotenv').config();
const MongoDBClient = require('./servicios/MongoDBClient');

const mongoose = require('mongoose');

const neo4j = require('neo4j-driver');
```

Una vez con esta parte lista se procede con la propia función la cual consta de establecer las conecciones:

```
const mongoClient = new MongoDBClient(MONGO_URI, MONGO_DB_NAME);

        await mongoClient.connect();


        const driver = neo4j.driver('neo4j://localhost:7687',
neo4j.auth.basic('neo4j', '12345678'));

        const session = await driver.session({ database: 'neo4j' });

        console.log("Neo4j Conectado!");
```

Posteriormente se inicia el ciclo de migraciones, donde se crean las entidades que contendrá los Juegos, empresas, géneros, tags y plataformas, siguiendo una estructura:

Se solicitan los datos, en este caso generos, y se registran en Neo4J uno a uno.

```
const genres = await mongoClient.getGeneros();

        console.log(`Hemos leido ${genres.length} Generos.`);



        for (const genre of genres) {

            await session.run(

                `CREATE (g:Genero { name: $name })`,

                { name: genre.name }

            );

        }

        console.log('Generos migrados!.');
```

De igual manera se hace lo mismo con tags, y básicamente así con todas las 5 entidades principales.

```
const tags = await mongoClient.getTags();

        console.log(`Hemos leido ${tags.length} tags.`)



        for(const tag of tags){//Se itera sobre los valores y no los objetos al
solo ser {clave:valor}

            // console.log(tag)
```

```
            await session.run(
              `CREATE (t:tags { name: $name })`,
              { name: tag.name }
            );
        }
        console.log('Tags migrados!.');
```

El siguiente bloque crea nodos en una base de datos Neo4j para representar cada empresa y finalmente confirma que la migración de las empresas ha sido exitosa.

```
        const empresas = await mongoClient.getEmpresas();
        console.log(`Hemos leido ${empresas.length} empresas.`);
         for (const empresa of empresas) {
           await session.run(
             `CREATE (e:Empresa {
               id: $id, name: $name, slug: $slug, games_count: $games_count
             })`,
             {
               id: empresa.id, name: empresa.name, slug: empresa.slug,
               games_count: empresa.games_count
             }
           );
         }
        console.log('Empresas Migradas!.');
```

El siguiente bloque de codigo comienza leyendo las plataformas desde MongoDB, luego imprime en la consola la cantidad de plataformas leídas. A continuación, para cada plataforma leída, ejecuta una consulta Cypher para crear un nodo en Neo4j representando esa plataforma, asignando las propiedades como id, name, slug y games_count a partir de los datos obtenidos de la base de datos MongoDB. Una vez que se han creado todos los nodos de plataforma, el proceso de migración se completa, indicando en la consola que se han leído y migrado las plataformas con éxito.

```
const plataformas = await mongoClient.getPlataformas();
        console.log(`Hemos leido ${plataformas.length} plataformas.`)
```

```
        for (const plataforma of plataformas) {
          await session.run(
            `CREATE (p:Plataforma {
              id: $id, name: $name, slug: $slug, games_count: $games_count
            })`,
            {
              id: plataforma.id, name: plataforma.name, slug: plataforma.slug,
              games_count: plataforma.games_count
            }
          );
        }
```

y por último la migración de los videojuegos:

El proceso comienza obteniendo los videojuegos de la base de datos MongoDB y luego muestra en la consola la cantidad de videojuegos leídos. Posteriormente, se limita la cantidad de videojuegos a procesar a solo los primeros 10 elementos de la lista original. Luego, para cada uno de estos videojuegos seleccionados, se ejecuta una consulta Cypher para crear un nodo en Neo4j que representa el videojuego, asignando las propiedades como id, slug, name, rating, ratings_count, entre otros, a partir de los datos obtenidos de la base de datos MongoDB.

```
const videojuegos = await mongoClient.getVideojuegos();
       console.log(`Hemos leido ${videojuegos.length} Videojuegos.`)

       let v = videojuegos.slice(0,10)//AQUI RESTRINJO LA CANTIDAD DE VIDEOJUEGOS

        for (const juego of v) {
            // console.log(juego)
           // Crear nodo de Videojuego
           await session.run(
             `CREATE (v:Videojuego {
               id: $id,
               slug: $slug,
               name: $name,
               rating: $rating,
               rating_top: $rating_top,
               ratings_count: $ratings_count,
               reviews_text_count: $reviews_text_count,
               added: $added,
```

```
        metacritic: $metacritic,
        playtime: $playtime,
        suggestions_count: $suggestions_count,
        reviews_count: $reviews_count
      })`,
      {
        id: juego.id,
        slug: juego.slug,
        name: juego.name,
        rating: juego.rating,
        rating_top: juego.rating_top,
        ratings_count: juego.ratings_count,
        reviews_text_count: juego.reviews_text_count,
        added: juego.added,
        metacritic: juego.metacritic,
        playtime: juego.playtime,
        suggestions_count: juego.suggestions_count,
        reviews_count: juego.reviews_count
      }
    );
```

Posteriormente se crean las relaciones:

```
    for (const plataformaId of juego.platforms) {
      await session.run(
        `MATCH (v:Videojuego {id: $vid}), (p:Plataforma {id: $pid})
         CREATE (v)-[:DISPONIBLE_EN]->(p)`,
        { vid: juego.id, pid: plataformaId }
      );
    }
```

Se migra la relación entre un videojuego y la empresa que lo desarrolló. Se busca el videojuego y la empresa por sus respectivos IDs y se crea una relación de tipo DESARROLLADO_POR entre ellos en la base de datos Neo4j.

```
                    // Migrar relaciones con Plataformas
                    for (const plataformaId of juego.platforms) {
                     await session.run(
                       `MATCH (v:Videojuego {id: $vid}), (p:Plataforma {id: $pid})
                        CREATE (v)-[:DISPONIBLE_EN]->(p)`,
                       { vid: juego.id, pid: plataformaId }
                     );
                    }
```

En el segundo bloque, se están migrando las relaciones entre un videojuego y las plataformas en las que está disponible. Para cada plataforma asociada al videojuego, se crea una relación de tipo DISPONIBLE_EN entre el videojuego y la plataforma correspondiente en Neo4j.

```
                    //Migrar la relacion con los nuevos nodos genero
                    for (const genre of juego.genres){
                        await session.run(
                                `MATCH (v:Videojuego {id: $vid}), (g:Genero {name:
$name})
                            CREATE (v)-[:PERTENECE_A]->(g)`,
                           { vid: juego.id, name: genre.name }
                        );
                    }
```

En el tercer bloque, se realiza la migración de las relaciones entre un videojuego y los diferentes géneros a los que pertenece. Se busca el videojuego y el género por su nombre y se crea una relación de tipo PERTENECE_A entre ellos en la base de datos Neo4j.

```
                    //Migrar la relacion con los nuevos nodos tags
                    for (const tag of juego.tags){
                        await session.run(
                                 `MATCH (v:Videojuego {id: $vid}), (t:tags {name:
$name})
```

```
                    CREATE (v)-[:ETIQUETADO_COMO]->(t)`,
                  { vid: juego.id, name: tag.name }
                );
              }
```

Por último, en el cuarto bloque, se está migrando la relación entre un videojuego y las etiquetas asociadas a él. Para cada etiqueta del videojuego, se crea una relación de tipo ETIQUETADO_COMO entre el videojuego y la etiqueta correspondiente en Neo4j. Una vez completada la migración, se muestra un mensaje en la consola confirmando que las relaciones Videojuego-Etiqueta han sido migradas con éxito. Cada bloque se encarga de establecer y migrar relaciones específicas entre los nodos de Videojuego y los nodos relacionados en la base de datos Neo4j.

Consultas

Estas consultas están resueltas sobre NodeJS en el fichero **ConsultasFase_2.js** anexado a esta entrega.

Consulta 1

```
MATCH (j:Videojuego)-[:PERTENECE_A]->(g:Genero)

    WHERE g.name IN ['Action', 'Adventure']

    RETURN j;
```

- Buscamos un nodo que representa un videojuego y lo etiquetamos como "j".
- Establecemos una relación entre el videojuego (j) y un género (g). La flecha indica que la relación va desde el videojuego hacia el género, es decir, el videojuego pertenece a un género.
- Filtramos los resultados para que solo se consideren los géneros cuyo nombre sea "Action" o "Adventure".
- Devuelve el nodo de videojuego (j) que cumple con las condiciones anteriores.

Consulta 2

```
MATCH (j:Videojuego)-[:DESARROLLADO_POR]->(e:Empresa {name: 'NombreEmpresa'})

RETURN j

LIMIT n
```

- MATCH (j:Juego)-[:DESARROLLADO_POR]->(e:Empresa {nombre: 'NombreEmpresa'}): Encuentra juegos que están relacionados con una empresa desarrolladora específica. Asegúrate de reemplazar 'NombreEmpresa' con el nombre de la empresa a la que deseas hacer referencia.
- RETURN j: Devuelve los juegos que están asociados a la empresa especificada.
- LIMIT n: Limita los resultados a una cantidad específica de juegos, donde n es el número de juegos que deseas recuperar.

Consulta 3

```
MATCH (j:Videojuego)-[:DISPONIBLE_EN]->(p:Plataforma)
WITH j, COUNT(DISTINCT p) AS numPlataformas
WHERE numPlataformas > 4
RETURN COUNT(j) AS totalJuegos
```

- Se hace una coincidencia de los nodos de juego (Juego) que están relacionados con nodos de plataforma (Plataforma) a través de la relación DISPONIBLE_EN.
- Se cuenta el número de plataformas distintas a las que cada juego está relacionado.
- Se filtran los juegos que están relacionados con más de 4 plataformas.
- Finalmente, se devuelve el recuento de juegos que cumplen con esta condición.

Consulta 4

Se realiza un aggregate sobre Videojuego, donde se compara la cantidad de plataformas con el valor requerido, para después asociarlo con las plataformas de la colección plataformas y retornar los campos requeridos.

```
MATCH (j:Videojuego)-[:DISPONIBLE_EN]->(p:Plataforma)
WITH j, COUNT(DISTINCT p) AS numPlataformas
WHERE numPlataformas > 4
RETURN j
```

- Se busca la coincidencia de juegos que están relacionados con las plataformas Playstation 3, Xbox 360 y Nintendo Wii.
- Se agrupan las plataformas relacionadas con cada juego en una lista.
- Se filtran los juegos que tienen más de 2 plataformas asociadas, lo que significa que están disponibles en al menos 3 de las plataformas especificadas.
- Se retorna la lista de juegos que cumplen con este criterio.

Consulta 5

```
MATCH (v:Videojuego)
WHERE v.rating > 2
RETURN v
```

- Se hace una coincidencia de todos los nodos de Videojuego (Videojuego) en la base de datos.
- Se filtran los juegos según tengan una calificación (rating) mayor a 2 utilizando la cláusula WHERE.
- Se devuelven los juegos que cumplen con esta condición.

Consulta 6

```
MATCH (v:Videojuego)-[:ETIQUETADO_COMO]->(t:tags)
WHERE t.name IN ['Multiplayer', 'Singleplayer']
MATCH (v)-[:PERTENECE_A]->(g:Genero)
WHERE g.name IN ['Action', 'Adventure']
RETURN v
```

- Se hace una coincidencia de los nodos de Videojuego (Videojuego) que están relacionados con nodos de Etiqueta (Tag) y se filtran por las etiquetas "Multiplayer", "Singleplayer", "Action" y "Adventure".
- Se realiza otra coincidencia para encontrar los géneros asociados a cada juego y se filtran por los géneros "Action" y "Adventure".

- Finalmente, se devuelven los juegos que cumplen con estas condiciones.

Consulta 7

```
MATCH (v:Videojuego)
WHERE
  size((v)-[:ETIQUETADO_COMO]->(t:Tag) WHERE t.name IN ${tags}) = size(${tags})
  AND
  size((v)-[:PERTENECE_A]->(g:Genero) WHERE g.name IN ${generos}) =
size(${generos})
RETURN v;
```

- Se calcula el promedio de calificaciones para los géneros "RPG" y "FPS".
- Luego, se buscan los juegos que pertenecen a estos géneros y tienen una calificación mayor que el promedio calculado.