Bases de Datos Orientada a Grafos Neo4j

1ra Laura Camila Scarpetta Rodríguez
Universidad Distrital Francisco José de Caldas
Maestría en Ciencias de la Información y la Comunicación
Bogotá D.C., Colombia
lcscarpettar@correo.udistrital.edu.co

2do José Manuel Vargas Montero
Universidad Distrital Francisco José de Caldas
Maestría en Ciencias de la Información y la Comunicación
Bogotá D.C., Colombia
jomvargasm@correo.udistrital.edu.co

***Resumen*—En este documento se verá una introducción al concepto de bases de datos orientada a grafos. Luego se hablará sobre la base de datos Neo4j. Se explicará su proceso de instalación para diferentes sistemas operativos y sus configuraciones iniciales. Posterior a ello se mostrará un manejo básico con este gestor de base de datos.**

***Index Terms*—Neo4j, BDOG, base de datos, relaciones, consultas**

I. Introducción

Las bases de datos orientadas a grafos ofrecen una gran oportunidad en la actualidad en la que tenemos una creciente cantidad de información.
Este modelo ofrece tantas ventajas, como lo pueden resumir las famosas cuatro V: volumen, velocidad, veracidad y variedad.
Este modelo de bases de datos se puede implementar por medio de diferentes gestores de bases de datos. Uno de los que toma gran importancia es Neo4j, por ser este de los pocos de OpenSource existentes.
Neo4j inició en el año 2007. Opera sobre Java. Es un sistema multiplataforma, desarrollado por la empresa Neo Technology.
Como ejemplo de su gran importancia y rápido crecimiento se debe decir que Neo4j tiene como clientes a las empresas Hewlett-Packard, eBay y Cisco.
En la Figura 1 se observa el logo del software Neo4j.

Figura 1. Logo de Neo4j.

II. Bases de datos orientadas a grafos

Las bases de datos orientadas a grafos (DBOG) cambian el concepto completamente de las bases de datos relacionales (SQL) y hasta de las bases de datos no relaciones (NoSQL). La información, los objetos, de la base de datos se representan por nodos. Las relaciones entre los objetos se relacionan en los grafos con las aristas. Un ejemplo de un diagrama de grafos para una base de datos se muestra en la Figura 2.

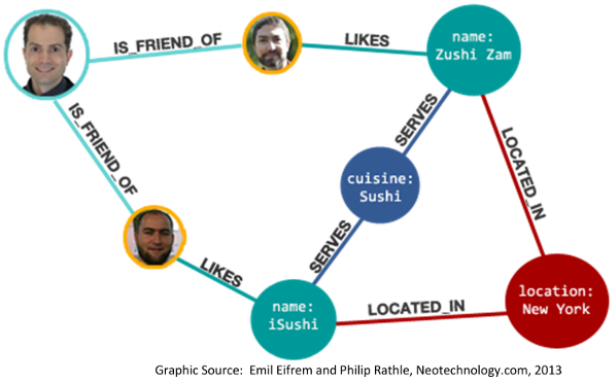

Figura 2. Ejemplo de una DBOG.

Como se observa, el diagrama es bastante intuitivo de lo que nosotros entendemos por una relación.
Neo4j usa grafos de propiedad. Son grafos con peso, con etiquetas y donde podemos asignar propiedades tanto a nodos como relaciones (por ejemplo, cuestiones como nombre, edad, país de residencia,nacimiento).

III. Instalación

El proceso de instalación del programa Neo4j es diferente para cada sistema operativo. En el presente documento se explicará el proceso para la instalación en Windows y en una distribución de Linux (Ubuntu).
Se recomienda visitar la página de Neo4j para mayor información:

- Requerimientos: https://neo4j.com/docs/operations-manual/current/installation/requirements/
- Instalación para distribuciones Linux de Debian (Ubuntu): https://neo4j.com/docs/operations-manual/current/installation/linux/debian/

- Instalación para sistemas Window: https://neo4j.com/docs/operations-manual/current/installation/windows/

III-A. Instalación en Windows

Para instalar Neo4j en un computador (o servidor) con Windows se debe descargar el instalador de la página: https://neo4j.com/download/
Una vez se haya descargado, se debe dar doble clic para ejecutarlo y se deben seguir las instrucciones.
Después de aceptar el acuerdo de licencia se mostrará la siguiente ventana, como se observa en la Figura 3.

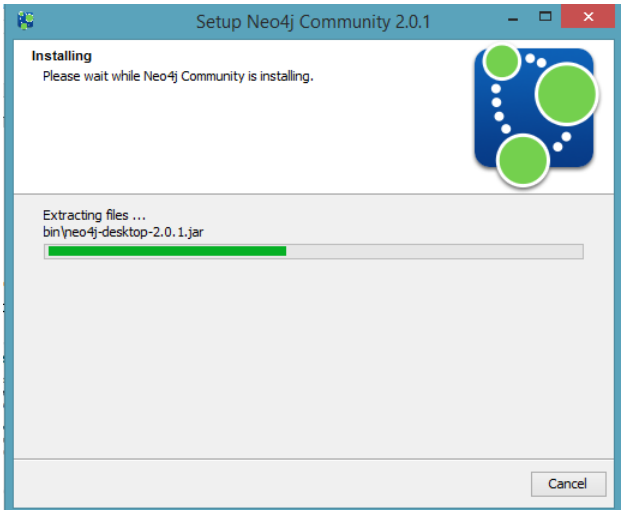

Figura 3. Instalación de Neo4j en Windows.

Una vez termine el proceso de instalación, se debe ejecutar Neo4j. Se abrirá la ventana que se muestra en la Figura 4.

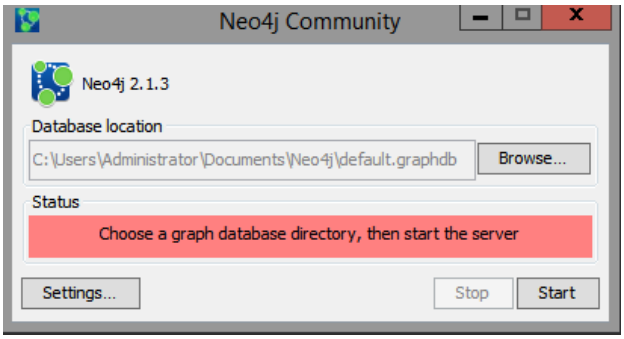

Figura 4. Ejecución de Neo4j en Windows.

Si se desea realizar configuraciones como la cantidad de memoria RAM asignad a los nodos y las aristas, la configuración del acceso desde el explorador o desde un sitio remoto, el puerto de la conexión a la base de datos, habilitar auto indexación, entre otras opciones, se debe dar clic en Settings... Se desplegará una ventana como la que se muestra en la Figura 5.

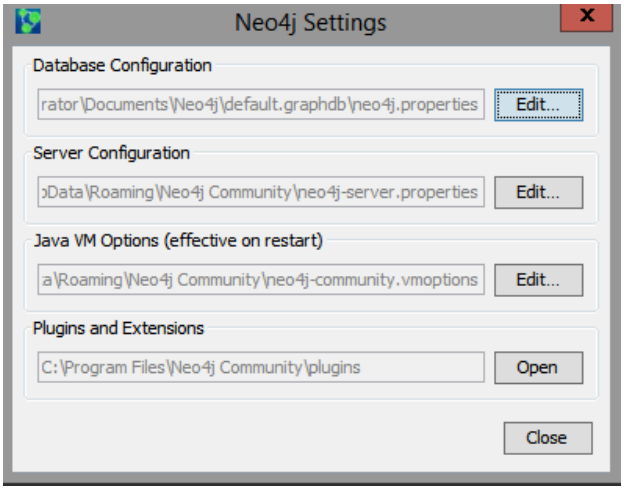

Figura 5. Configuraciones de Neo4j en Windows.

Finalmente, hechas las configuraciones deseadas, se debe dar clic en Start para hacer que el servidor arranque. La ventana de Neo4j debe quedar como en la Figura 6.

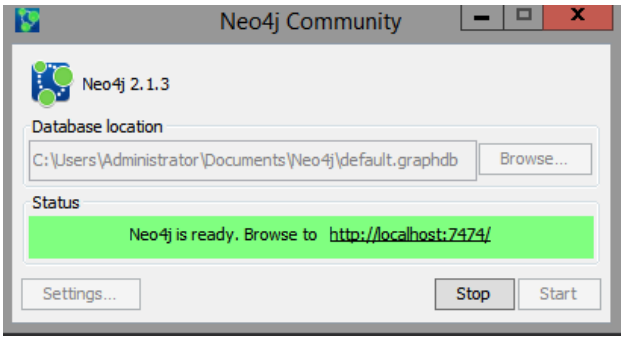

Figura 6. Arranque de Neo4j en Windows.

III-B. Instalación en Linux (Debian/Ubuntu)

Para las distribuciones de Linux el proceso es diferente. Es recomendable ir a la página https://neo4j.com/docs/operations-manual/current/installation/linux/debian/ para mayor información.
Para instalarlo se debe cumplir con el requisito de tener instalado Java 8 runtime. Para ello se debe ejecutar las siguientes sentencias, para tener acceso al repositorio e Java 8, como se muestran en las Figuras 7 y 8

```
pruebas@pruebas-VirtualBox:~$ sudo add-apt-repository ppa:webupd8team/java
```

Figura 7. Agregar repositorio de Java Runtime.

```
pruebas@pruebas-VirtualBox:~$ sudo apt-get update
```

Figura 8. Actualizar repositorios de Linux.

Se usa la siguiente sentencia para poder instalar como tal el runtime de Java, como se observa en la Figura 9

```
pruebas@pruebas-VirtualBox:~$ sudo apt-get install oracle-java8-installer
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  gsfonts-x11 java-common oracle-java8-set-default
Suggested packages:
  binfmt-support visualvm ttf-baekmuk | ttf-unfonts | ttf-unfonts-core
  ttf-kochi-gothic | ttf-sazanami-gothic ttf-kochi-mincho
  | ttf-sazanami-mincho ttf-arphic-uming
The following NEW packages will be installed:
  gsfonts-x11 java-common oracle-java8-installer oracle-java8-set-default
0 upgraded, 4 newly installed, 0 to remove and 3 not upgraded.
Need to get 54.8 kB of archives.
After this operation, 272 kB of additional disk space will be used.
Do you want to continue? [Y/n]
```

Figura 9. Instalación del Runtime de Java en Linux.

Una vez hecho esto, se debe agregar el repositorio de Neo4j. Para ello se ejecuta la siguiente sentencia desde la terminal, como se muestra en la Figura 10.

```
pruebas@pruebas-VirtualBox:~$ wget -O - https://debian.neo4j.org/neotechnology.g
pg.key | sudo apt-key add -
--2017-10-27 15:39:18--  https://debian.neo4j.org/neotechnology.gpg.key
Resolving debian.neo4j.org (debian.neo4j.org)... 52.0.233.188
Connecting to debian.neo4j.org (debian.neo4j.org)|52.0.233.188|:443... connected
.
HTTP request sent, awaiting response... 200 OK
Length: 4791 (4.7K) [application/octet-stream]
Saving to: 'STDOUT'

-                   100%[===================>]   4.68K  --.-KB/s    in 0s

2017-10-27 15:39:37 (434 MB/s) - written to stdout [4791/4791]

OK
pruebas@pruebas-VirtualBox:~$ echo 'deb http://debian.neo4j.org/repo stable/' |
sudo tee -a /etc/apt/sources.list.d/neo4j.list
deb http://debian.neo4j.org/repo stable/
pruebas@pruebas-VirtualBox:~$ sudo apt-get update
```

Figura 10. Repositorios de Neo4j en Linux.

Finalmente, se instala Neo4j, haciendo lo que se muestra en la Figura 11.

```
pruebas@pruebas-VirtualBox:~$ sudo apt-get install neo4j
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  cypher-shell daemon
The following NEW packages will be installed:
  cypher-shell daemon neo4j
0 upgraded, 3 newly installed, 0 to remove and 3 not upgraded.
Need to get 87.7 MB of archives.
After this operation, 102 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
```

Figura 11. Repositorios de Neo4j en Linux.

Para iniciar el servidor y arrancar en gestor de base de datos, se debe ejecutar el comando que se muestra en al Figura 12.

```
pruebas@pruebas-VirtualBox:~$ sudo service neo4j start
```

Figura 12. Inicio del servidor de Neo4j en Linux.

III-C. Prueba de la instalación

Sin importar qué distribución o qué sistema operativo se haya utilizado, se puede probar la correcta instalación utilizando el explorador Web. Se dirige a la dirección $localhost:7474/browser/$.
En esta dirección se tendrá todo el acceso y gestión de la base de datos: es un aplicativo corriendo sobre un explorador Web. Esto se muestra en la Figura 13.

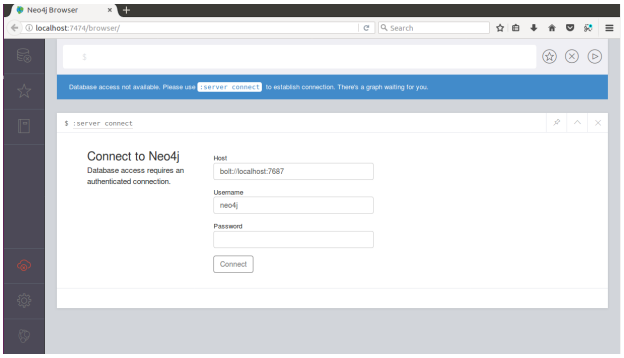

Figura 13. Neo4j corriendo desde el explorador Web.

La contraseña al iniciar Neo4j por primera vez por defecto es $neo4j$. Luego de ingresarla se pedirá se cambia la contraseña, como se muestra en la Figura 14.

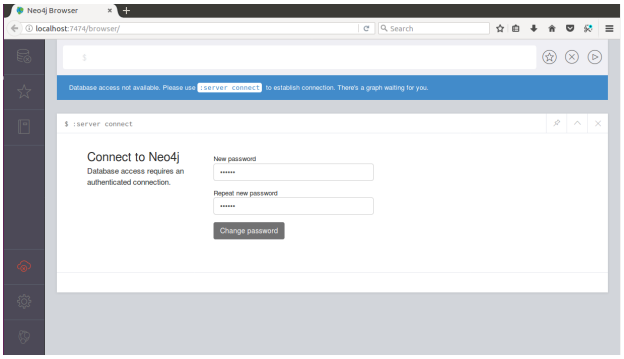

Figura 14. Cambio de contraseña en Neo4j por primera vez.

Una vez hecho esto, se estará en la página de inicio de Neo4j. Se verá la barra de ingreso de comandos (consultas), un espacio con las noticias y principales mensajes, un menú de accesos en el costado izquierdo y la conexión al servidor en sí. Todo esto se ilustra en la Figura 15.

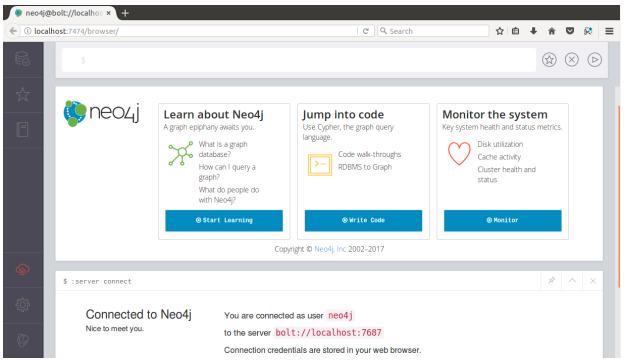

Figura 15. Página de inicio de Neo4j.

IV. Creación de nodos.

Una vez en el aplicativo de explorador se pueden crear nodos usando la sentencia `CREATE (n)`. Al ejecutar la

sentencia se verá un mensaje que dice que se creó un nodo en un tiempo determinado. Si se desea observar el nodo, se debe crear de esta manera: `CREATE (n) RETURN n`. Esto se muestra en la Figura 16.

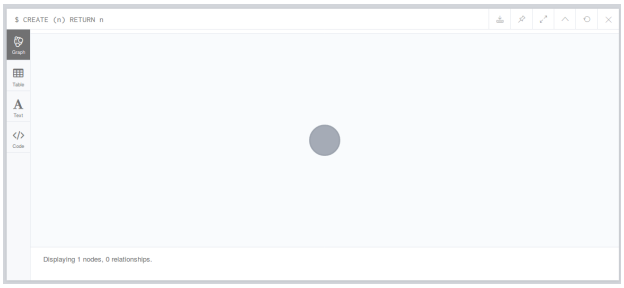

Figura 16. Creación y visualización de un nodo.

Así mismo es posible crear varios nodos al tiempo y visualizarlos con el comando `CREATE (n),(m) RETURN n,m`.
Un concepto importante en los nodos de Neo4j, son los label. Esta es una característica que define a qué grupo, clase o categoría pertenece el nodo. Por ejemplo, podemos crear nodos que pertenezcan y hagan referencia a personas.
Para agregar un label se hace de la siguiente manera: `CREATE (n:Persona) RETURN n`. Se debe aclarar aquí que la letra `n` es únicamente una referencia al nodo dentro de la sentencia específica. Bien se podría llamarlo o hacer referencia de otra manera y no cambiaría nada en el resultado. Por ejemplo `CREATE (yo:Persona) RETURN yo`. El resultado se muestra en la Figura 17.

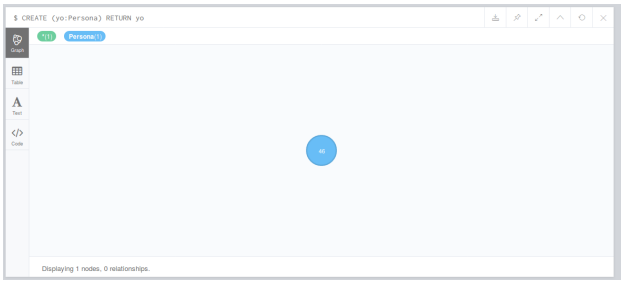

Figura 17. Nodo con label.

El número que se encuentra dentro del nodo (el círculo) es el id. Este es un identificador único de cada nodo. Si se posiciona el cursor sobre el nodo se observará que dice id.
Dependiendo de la aplicación Neo4j permite tener más de un label para un nodo. Este sería el caso, por ejemplo, para nodos que pertenezcan a dos o más categorías. Un ejemplo sería una persona que tenga una nacionalidad. Esto se realiza agregando labels con : de manera consecutiva. Se puede usar la siguiente sentencia como ejemplo: `CREATE (persona:Persona:Colombia) RETURN persona`. Esto se observa en la Figura 18.

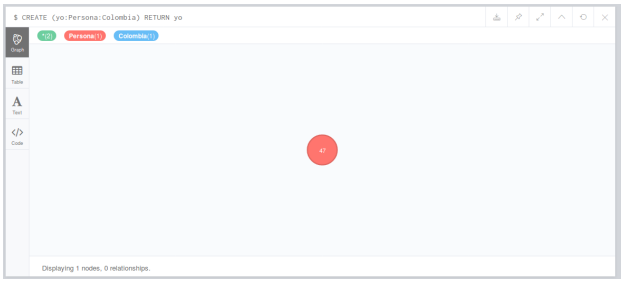

Figura 18. Nodo con dos label.

Hasta este punto se han creado nodos que pertenecen a una o más categorías. Ahora hace falta añadir información a cada nodo. Esto se realiza por medio de las propiedades. Para agregar información se escribe dentro de { } al momento de crear el nodo (se puede agregar o editar información luego de crearlos). La información se agrega escribiendo el nombre del campo seguido de : y finalmente el valor. El valor puede ser numérico (entero o de coma flotante), un string, valor booleano o un arregla del mismo tipo de datos.
Un ejemplo de la creación de un nodo con propiedades es usando la siguiente sentencia: `CREATE (persona:Persona { name: "Mi nombre"} ) RETURN persona`. Así mismo se pueden agregar más propiedades. Esto se muestra en la Figura 19.

Figura 19. Nodo con propiedades.

V. Consultas básicas.

Para hacer consultas en Neo4j se utiliza la palabra clave `MATCH`. Esta palabra va seguido de la información del nodo o nodos que se desean consultar. Por ejemplo, si se busca por algún label específico, por el valor de alguna propiedad o hasta por una relación (se verá en la siguiente sección). Luego, algo importante, es utilizar la sentencia `RETURN`, con el fin de ver los nodos encontrados. Un ejemplo clásico que retorna todos los nodos existentes es la siguiente sentencia `MATCH (n) RETURN n`. El resultado se observa en la Figura 20.

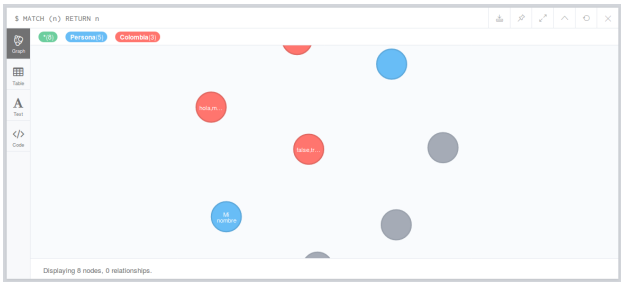

Figura 20. Consulta de todos los nodos.

Como se observa, si se tuviera una base de datos extensa, se tendría problemas para visualizar los nodos. Es por esto que es recomendable limitar el número de resultados. Para hacer esto se utiliza la sentencia `LIMIT` seguida del número de nodos máximos que se desea aparezcan. Por ejemplo, se usaría la sentencia `MATCH (n) RETURN n LIMIT 4`. El resultado se observa en la Figura 21.

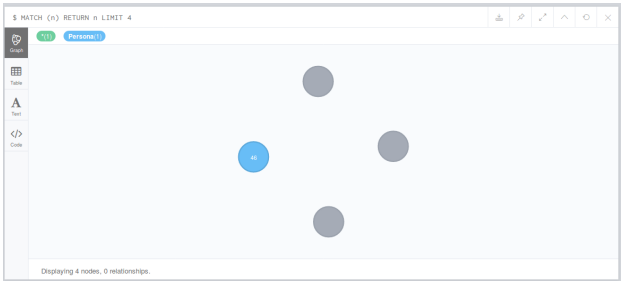

Figura 21. Consulta de los nodos con límite de 4.

Si se quiere buscar los nodos que pertenezcan a la misma categoría, grupo, clase o label, se debe especificar el label. Para ello se hace lo mismo que cuando se crea un nodo de una categoría. Es decir, se utiliza la sentencia `(n:Label)` en vez de `(n)`. Por ejemplo, al buscar todos los nodos que pertenezcan a personas se utilizaría la sentencia `MATCH (n:Persona) RETURN n LIMIT 4`. El resultado se observa en la Figura 22.

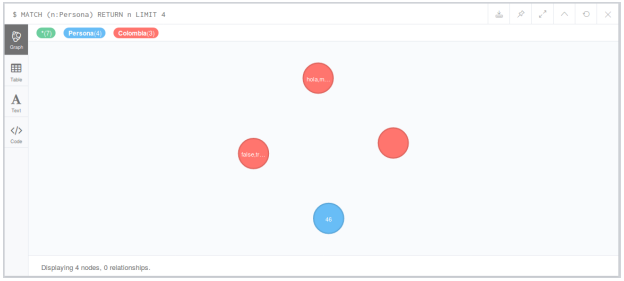

Figura 22. Consulta de los nodos que pertenezcan a Persona.

Otra manera de consultar nodos y filtrarlos es por sus propiedades. Se podría preguntar por los nodos que tengan cierto nombre. La sentencia es similar a la manera que se creó un nodo con propiedades, usando { } e ingresando las propiedades dentro. Por ejemplo, se podría buscar los nodos que tengan el nombre "Mi nombre". Para ello se utiliza al sentencia `MATCH (n { name: "Mi nombre"}) RETURN n LIMIT 4`. Se insiste que la letra `n` es el nombre que se le asigna a los nodos encontrados en esta sentencia. De manera que bien se podría utilizar otra letra o cualquier palabra, sin cambiar el resultado. También es posible hacer la consulta buscando por los nodos que tengan el nombre "Mi nombrez que además perezcan a Personas. Esto se muestra en la siguiente sentencia `MATCH (yo:Persona { name: "Mi nombre"}) RETURN yo LIMIT 4`. El resultado se observa en la Figura 23

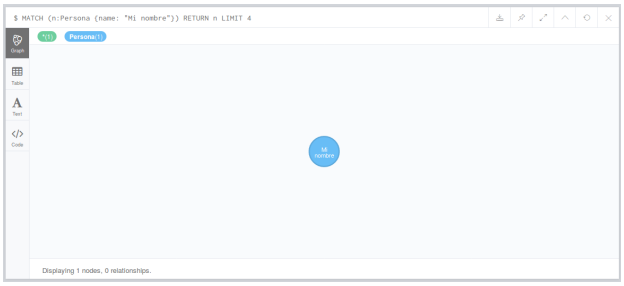

Figura 23. Consulta de los nodos que pertenezcan a Persona y se llaman "Mi nombre".

Para finalizar con las consultas básicas se va a utilizar un último criterio de búsqueda: consultas por el id del nodo. Para conocer todas las propiedades de un nodo, se debe posicionar el cursor sobre el nodo deseado y aparecerán las propiedades en la parte inferior. Junto a las propiedades se observa el id del nodo dentro de < >. Esto se observa en la Figura 24.

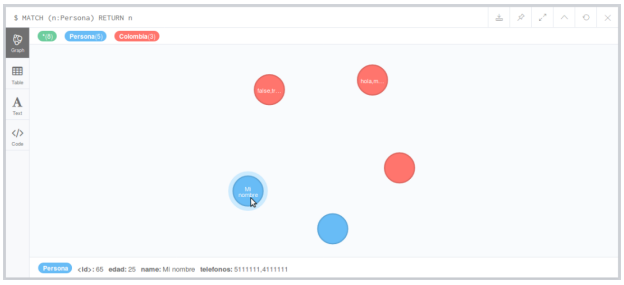

Figura 24. Visualización de las propiedades del nodo.

Para consultar por el id del nodo se utiliza simplemente la condición `WHERE id(n)=65`. El número 65 es un ejemplo para este caso. La consulta completa quedaría así `MATCH (n) WHERE id(n)=65 RETURN n`. El resultado se observa en la Figura 25.

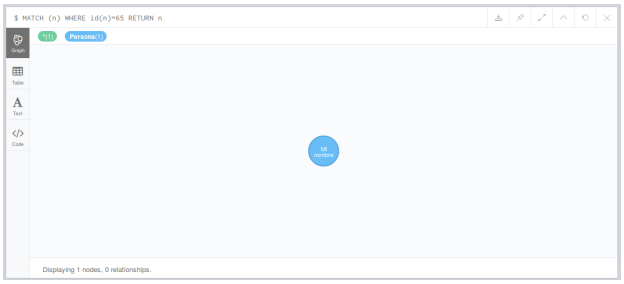

Figura 25. Consulta del nodo por el id.

VI. Cambiar o agregar propiedades de los nodos.

Para cambiar o añadir alguna propiedad a un nodo ya creado, se debe primero realizar una consulta que direccione hacia dicho nodo. Por ejemplo, de los nodos creados el nodo con id 43 no pertenece a ninguna categoría, ningún label. Primero se debe asegurar es el nodo deseado, así que se ejecuta la consulta `MATCH (n) WHERE id(n)=43 RETURN n`. El resultado de la consulta se observa en la Figura 26.

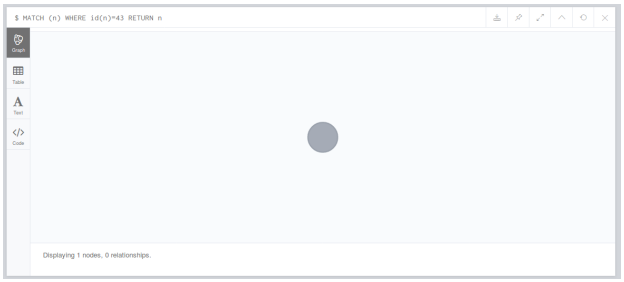

Figura 26. Consulta del nodo por el id.

Ahora, se utiliza la palabra reservada `SET` para cambiar alguna propiedad o algún Label. En este caso se quiere cambiar el label, así que se utiliza la consulta seguida de la sentencia `SET n:Persona`. La sentencia quedaría así: `MATCH (n) WHERE id(n)=43 SET n:Persona RETURN n`. El resultado se observa en la Figura 27.

Figura 27. Cambio del label de un nodo.

Si el nodo ya tenía un label, al hacer esta sentencia simplemente se agregaría un nuevo label. Para agregar dos o más labels en una sola sentencia se utiliza la misma lógica, solo que se agregan los labels deseados separados por : Esto se muestra en esta sentencia `MATCH (n) WHERE id(n)=43 SET n:Persona:Colombia RETURN n`

Para agregar una propiedad o para cambiar el valor de una propiedad ya existente se utiliza la sentencia `SET` seguida del nombre que se desea asignar al nodo (normalmente `n`) luego un punto ., seguido del nombre de la propiedad, luego el signo igual y finalmente el valor que deseamos asignar. Un ejemplo sería `SET n.name= "Otro nombre"`. En este caso, si la propiedad ya existe y tiene valor en dicho nodo, Neo4j cambiará el valor por el que se le está ingresando. En caso que no exista, se creará la propiedad y se asignará el valor.

Un ejemplo de agregar una propiedad sería el siguiente `MATCH (n) WHERE id(n)=64 SET n.name= "Otro nombre"RETURN n`. El resultado se muestra en la Figura 28.

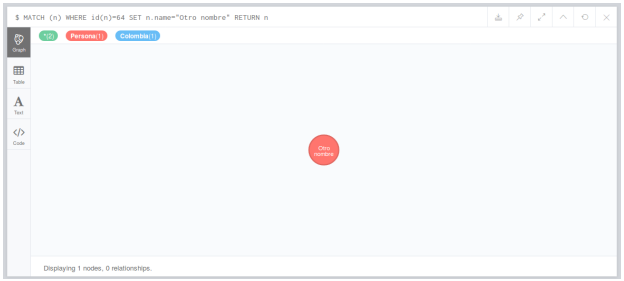

Figura 28. Agregar la propiedad nombre de un nodo.

Otra manera de agregar o cambiar propiedades es utilizando una sentencia similar a la usada cuando se crearon las propiedades con los { }. Aquí se debe tener cuidado. La sentencia `SET` quedaría así `SET n += { name: "Otro nombre"}`. En este caso la sentencia buscaría si existe alguna propiedad `name`. Si es así cambia su valor a `.°tro nombre"`, sino la crea y asigna el valor. Si no se escribe el signo + junto al signo = se dejará como única propiedad la que se está asignando, es decir, se borrarán todas las demás propiedades.

Un ejemplo de agregar o editar una propiedad con este tipo de sentencias es el siguiente: `MATCH (n) WHERE id(n)=64 SET n += { name: "Segundo nombre"} RETURN n`. El resutlado se muestra en la Figura 29.

Figura 29. Cambiar la propiedad nombre de un nodo.

Para eliminar una propiedad se realiza similar a establecer su valor. Solo que se asigna un valor nulo o `null`. Esta

es una manera sencilla de hacerlo. Un ejemplo de esto se muestra en la sentencia `MATCH (n) WHERE id(n)=64 SET n.name= null RETURN n`. El resultado se muestra en la Figura 30.

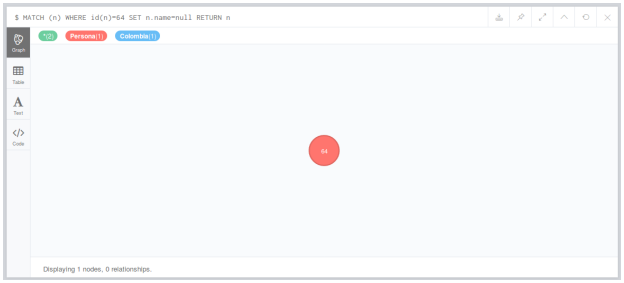

Figura 30. Borrar la propiedad nombre de un nodo.

Otra manera se borrar una propiedad es utilizando la palabra clave `REMOVE`. Se usa de manera similar a la palabra `SET`, pero no se asigna valor a la propiedad. Por ejemplo se usaría simplemente `REMOVE n.name`. Una sentencia de ejemplo sería `MATCH (n) WHERE id(n)=64 REMOVE n.name RETURN n`. El resultado se muestra en la Figura 31.

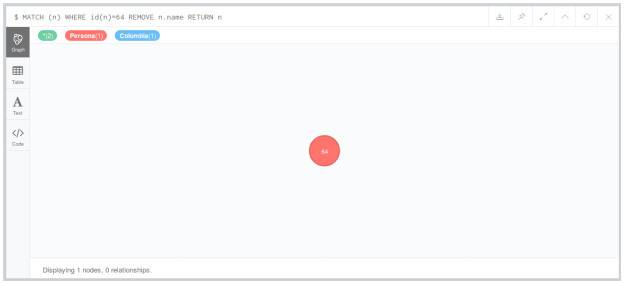

Figura 31. Borrado de la propiedad nombre de un nodo.

Finalmente con la misma sentencia `REMOVE` se puede borrar labels se un nodo. Se hace de la misma manera que se asignaron con la sentencia `SET`. Un ejemplo sería el siguiente `REMOVE n:Colombia`. La sentencia quedaría así: `MATCH (n) WHERE id(n)=43 REMOVE n:Colombia RETURN n`. El resultado se observa en la Figura 32.

Figura 32. Borrado del label de un nodo.

VII. Creación de relaciones.

Antes de crear relaciones, necesitamos poder hacer una consulta a lo dos nodos que queremos relacionar. Para ello se realiza de la misma manera usando comas para relacionar varios nodos.
Un ejemplo sería `MATCH (n) , (m) WHERE id(n)=64 AND id(m)=65 RETURN n , m`. El resultado se observa en la Figura 33.

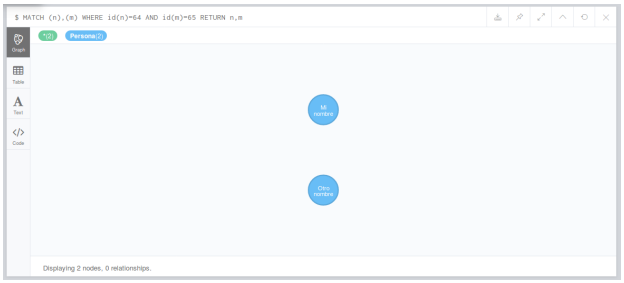

Figura 33. Consulta de dos nodos simultáneos.

Para crear la relación se utiliza la palabra clave `CREATE` y de la siguiente manera: `CREATE (n) -[r:Amigos]-> (m) RETURN n , m`. En este caso el nombre de la relación es r y el tipo de la relación (similar a label) es Amigos. El ejemplo se muestra en la Figura 34.

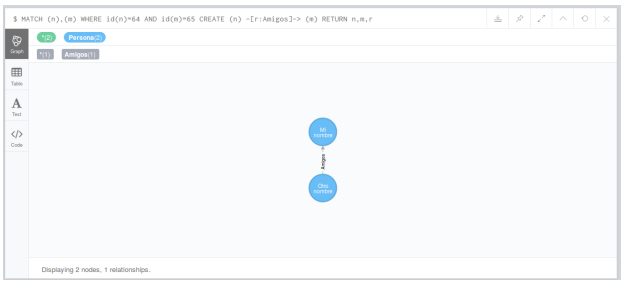

Figura 34. Crear una relación.

Para consultar la relación y los nodos, y visualizarlos se realiza de esta manera: `MATCH (n) -[r]-> (m) WHERE id(n)=64 AND id(m)=65 RETURN n , m, , r`. El resultado se observa en la Figura 35.

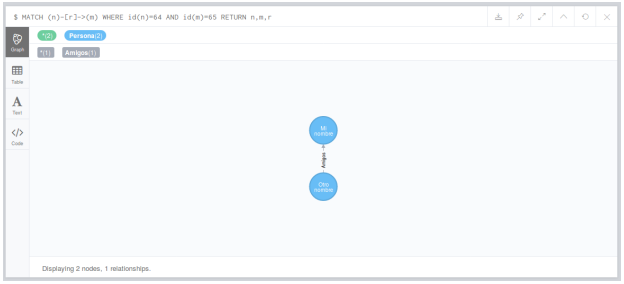

Figura 35. Consulta de la relación creada.

A las relaciones se pueden establecer propiedades. Se realiza de varias maneras, ya sea al momento de crearlas o después de tenerlas ya creadas. Al crearlas se pueden agregar las propiedades de manera similar que como se hace con los nodos, utilizando { } y escribiendo dentro la propiedad y su valor separados por :. Un ejemplo sería la sentencia `MATCH (n) , (m) WHERE id(n)=64 AND id(m)=65 CREATE (n) -[r:Amigos {year: 2010 }]->(m) RETURN n , m, r.`

Para establecer la propiedad después de haber creado la relación, se realiza la consulta y se utiliza la palabra reservada `SET`. Se hace de la misma manera que con los nodos, utilizando el nombre de la relación, escribiendo un punto, seguido del nombre de la propiedad, después un signo = y finalmente el valor de la propiedad. Una sentencia de ejemplo sería `MATCH (n) -[r]->(m) WHERE id(n)=64 AND id(m)=65 SET r.name = "La relacion "RETURN n , m, r.`
Utilizando la misma lógica, se podría eliminar propiedades de las relaciones. Utilizando la palabra clave `REMOVE` en vez de `SET`. Y obviamente sin asignar ningún valor. La sentencia quedaría de esta manera: `MATCH (n) -[r]->(m) WHERE id(n)=64 AND id(m)=65 REMOVE r.name RETURN n , m , r.`
Finalmente, para eliminar una relación se realiza la misma consulta y se utiliza el comando `REMOVE r`. En este caso no se retorna la relación. La sentencia quedaría así: `MATCH (n) -[r]->(m) WHERE id(n)=64 AND id(m)=65 DELETE r RETURN n , m.`

VIII. Constrains.

Los contrains son una parte fundamental en cualquier base de datos. En el caso de Neo4j no es la excepción. Para crearla se utiliza primero la palabra `CREATE` seguida de `CONSTRAIN ON`. A continuación se puede especificar si el constrain va dirigido a un nodo a una relación. Si va dirigido a un nodo se debe escribir lo siguiente `( n:Label )`. En este caso la letra `n` es el nombre del nodo con el que serán referenciados, mientras que `Label` es el label o categoría de los nodos que se desean aplicar el constrain. Después se escribe la palabra `ASSERT`. Finalmente se escribe una condición lógica a alguna propiedad del nodo `n`, similar como se hace con otras bases de datos.
Un ejemplo de un constrain para un nodo sería la sentencia `CREATE CONSTRAIN ON (n:Persona) ASSERT n.name IS UNIQUE.`
Otro ejemplo es el siguiente `CREATE CONSTRAIN ON (n:Persona) ASSERT exists(n.telefono).`
Para aplicar constrains a una relación se debe escribir primero `CREATE CONSTRAIN ON`, seguido de la representación de la relación, es decir, `()-[r:Label]-()`. El resto es lo mismo, es decir, la palabra `ASSERT` seguido de la condición del constrain que se desea aplicar.
Un ejemplo de un constrain a una relación es la siguiente sentencia `CREATE CONSTRAIN ON ()-[r:Label]-() ASSERT exists(r.year).`
Para eliminar un constrain, sin importar de que tipo sea, se realiza la misma sentencia que se ejecutó para crearlo, pero en vez de utilizar la palabra `CREATE` se utiliza la palabra `DROP`. Un ejemplo sería texttt`DROP CONSTRAIN ON (n:Persona) ASSERT n.name IS UNIQUE.`

Agradecimientos

Se da agradecimientos a la empresa Neo Technology por el desarrollo de la base de datos Neo4j de una manera intuitiva y fácil de manejar. Además se agradece a los desarrolladores del presente documento.

Referencias

[1] G. Eason, B. Noble, and I. N. Sneddon, “On certain integrals of Lipschitz-Hankel type involving products of Bessel functions,” Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955.
[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.
[3] I. S. Jacobs and C. P. Bean, “Fine particles, thin films and exchange anisotropy,” in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.
[4] K. Elissa, “Title of paper if known,” unpublished.
[5] R. Nicole, “Title of paper with only first word capitalized,” J. Name Stand. Abbrev., in press.
[6] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, “Electron spectroscopy studies on magneto-optical media and plastic substrate interface,” IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].
[7] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.