Bases de Datos en grafo: Neo4j

Master Executive BigData y Business Analytics

****

**Asignatura**: Bases de Datos No Convencionales

**Módulo**: Tecnologías de Big Data/Gestión de Datos

**Coordinador**: Elena García Barriocanal, [elenagarcia@campusciff.net](mailto:elenagarcia@campusciff.net)

**Profesores:** Jordi Conesa [jordiconesa@campusciff.net](mailto:jordiconesa@campusciff.net)

Elena García Barriocanal, [elenagarcia@campusciff.net](mailto:elenagarcia@campusciff.net)

Contenido

[Instalación de Neo4j 2](#_Toc412381078)

[Arrancar el servicio de Neo4j 2](#_Toc412381079)

[Acceder a la interficie web de Neo4j 2](#_Toc412381080)

[Primeros pasos: entendiendo mejor los nodos 3](#_Toc412381081)

[Consultas simples 3](#_Toc412381082)

[Eliminando datos de la base de datos 4](#_Toc412381083)

[Carga de la base de datos de ejemplo “Movie Graph” 4](#_Toc412381084)

[Consultando la base de datos 5](#_Toc412381085)

[Creación de indices 6](#_Toc412381086)

[Consultas avanzadas 6](#_Toc412381087)

[Materialización de relaciones 8](#_Toc412381088)

# Instalación de Neo4j

Ir a la página web <http://neo4j.com/download/> y pulsar en el botón “Download Community Edition”. En teoría, el navegador os llevará a la página de descarga de Neo4j para el sistema operativo que tengáis.

# Arrancar el servicio de Neo4j

En caso de tener un sistema operativo Windows, se deberá hacer click sobre el icono de Neo4j en el menú de inicio. El sistema pedirá que se ubique la base de datos con la que trabajar. Si no le indicamos nada, trabajará con una base de datos vacía.

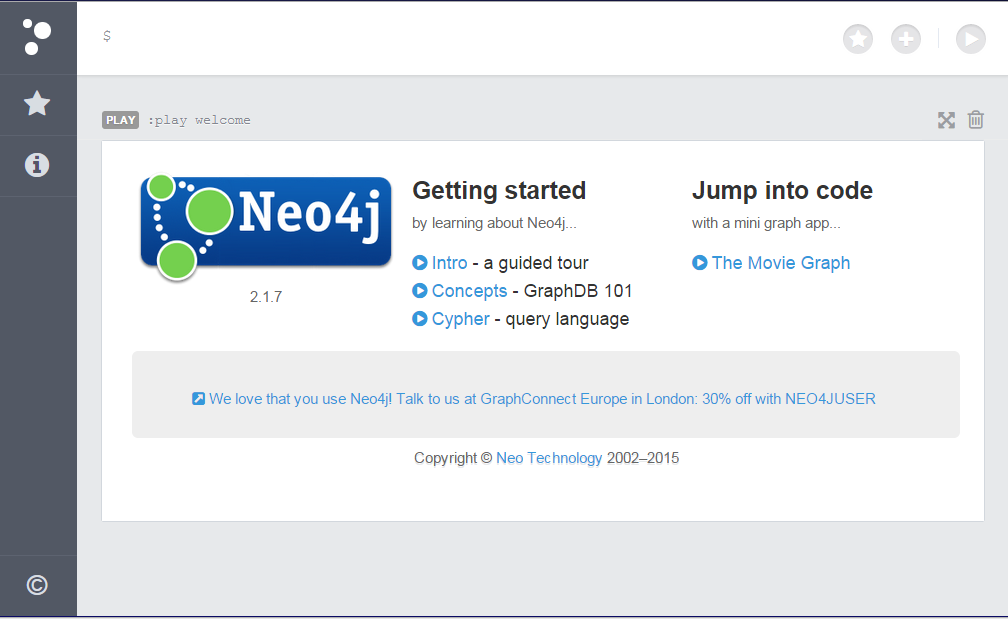
En caso de tener un sistema operativo distinto de Windows, se deberá abrir un terminal. Una vez en el terminal, se deberá ir al directorio donde se ha desinstalado el programa y ejecutar la siguiente sentencia:

bin/neo4j start

# Acceder a la interficie web de Neo4j

Se deberá abrir la siguiente página web (http://localhost:7474/browser/) en el navegador.

Si aparece algo parecido a lo siguiente es que todo va bien:



# Primeros pasos: entendiendo mejor los nodos

Primero crearemos un nodo:

CREATE (p)

¿Qué es lo que se ha creado? Lo consultamos mediante:

MATCH (n) RETURN (n)

¿Qué nos ha devuelto?

Ahora crearemos un nodo con información asociada. Para no tener que consultarlo después, podemos añadir un RETURN al final de la cláusula

CREATE (p {name:"Josep Pi", edat: "20", CP: "08012"}) RETURN p

Podemos consultar sus atributos e información haciendo click sobre el nodo.

Pero el nodo no tiene tipo, ¿Cómo crearíamos un nodo con tipo?

CREATE (p:Person {name:"Jordi Conesa", edat: "25", CP: "17300"}) RETURN p

¿Y si quisiéramos relacionar un par de nodos?

CREATE (p:Person {name:"Pedro"}),(q:Person {name:"Pablo"}), p-[r:FRIEND {from:1996}]->q RETURN p,q

**Ejercicio: Cread un nodo con vuestra información, otro con información de la asignatura y cread una relación que indique que estáis matriculados en esta asignatura.**

**Solución oculta**

CREATE (p:Person

{name:"Jordi Conesa", edat: "25", CP: "17300"}),

(s:Subject {name:"Bases de Datos NoSQL"}),p-[:ENROLLED\_IN]->s

RETURN p, s;

# Consultas simples

Supongamos que queremos obtener todos los nodos de tipo persona:

MATCH (p:Person)RETURN p

Ahora supongamos que queremos obtener el nodo que hemos creado con nuestra información. ¿Qual de las siguientes operaciones es la correcta?

MATCH (p:Person {name:"Jordi Conesa"}) RETURN p

MATCH (p:Person) WHERE p.name="Jordi Conesa" RETURN p

MATCH (p) WHERE p.name="Jordi Conesa" RETURN p

Ejecutad la siguiente sentencia dos veces:

CREATE (p:Person {name:"Jose"}) RETURN p

¿Què diferencia hay entre estas dos sentencias?

MATCH (p) RETURN p.name

MATCH (p) RETURN distinct p.name

# Eliminando datos de la base de datos

Supón que queremos eliminar los nodos que acabamos de crear para dejar la base de datos vacía. ¿Qué sentencias ejecutarías?

Para consultar todos los nodos de la base de datos podemos hacer:

MATCH p RETURN p

Ahora nos disponemos a borrarlos:

MATCH p DELETE p

La siguiente sentencia no funciona. ¿Por qué?

**Solución oculta**

Porqué hay nodos que tienen relaciones asignadas y no pueden borrarse nodos que están relacionados. Antes de borrar los nodos debemos borrar sus relaciones.

Para eliminar los elementos podemos ejecutar la siguiente sentencia:

MATCH p-[r]-q DELETE p,r,q

Ahora solo quedan los nodos sin relaciones, por tanto podemos eliminarlos con la sentencia anterior:

MATCH p DELETE p

¿Por qué es necesaria esta segunda consulta?

**Solución oculta**

Porque los nodos no relacionados no responden al patrón de la primera consulta y por tanto no se han podido eliminar entonces. Para hacer la eliminación en una sola consulta podemos utilizar la cláusula OPTIONAL MATCH de Neo4j.

Realizad la eliminación anterior en una sola consulta

**Solución oculta**

MATCH (p)

OPTIONAL MATCH (p)-[r]-(q)

DELETE p,r,q

RETURN COUNT(\*)

# Carga de la base de datos de ejemplo “Movie Graph”

Podemos cargar la base de datos a través de los botones de la izquierda de la interface o mediante la siguiente sentencia:

:play movie graph

Una vez cargada la base de datos podemos realizar la siguiente sentencia para obtener un fragmento de la misma:

MATCH (a) RETURN a LIMIT 100

A partir de ahora vamos a trabajar con esta base de datos.

# Consultando la base de datos

Una vez cargada la base de datos podemos preguntar por su pseudo-esquema, mediante el script “What is related, and how”.

Podemos empezar obteniendo información sobre el actor “Keanu Reeves”.

**Solución oculta**

MATCH (a:Person) where a.name="Keanu Reeves" RETURN (a)

Ahora podemos consultar todos los nodos con los que Keanu está relacionado:

**Solución oculta**

MATCH (a:Person {name: "Keanu Reeves"})--(b) RETURN a,b

Hasta ahora hemos consultado información de los nodos, pero ¿Cómo podríamos preguntar por los tipos de relación en los que participa Keanu?

**Solución oculta**

MATCH (a:Person {name: "Keanu Reeves"})—[r]-()   
RETURN DISTINCT type(r)

Podemos también pedir todos aquellos nodos relacionados por una relación de tipo “ACTED\_IN” con un nodo de tipo “Movie”.

MATCH (a)—[:ACTED\_IN]-(:Movie)   
RETURN a

Teniendo en cuenta que el tipo de relación “ACTED\_IN” indica que un autor ha participado en una película, obtened para cada actor el nombre de las películas en que ha trabajado.

**Solución oculta**

MATCH (a)—[:ACTED\_IN]->(m:Movie)

RETURN a.name, m.title;

**Ejercicio: Hacer distintas consultas sobre la base de datos para familiarizaros con su estructura y contenido.**

Realizar una consulta que devuelva, para cada película, sus autores y sus directores.

**Solución oculta**

MATCH (a)-[:ACTED\_IN]->(m)<-[:DIRECTED]-(d)

RETURN m.title AS movie, a.name AS actor, d.name AS director;

Los datos son los correctos pero el orden no es conveniente. Reordenar los datos para que se muestren por película:

**Solución oculta**

MATCH (a)-[:ACTED\_IN]->(m)<-[:DIRECTED]-(d)

RETURN m.title AS movie, a.name AS actor, d.name AS director

ORDER BY movie;

Una vez llegados aquí, es el momento de jugar con los agregados. Realizar una consulta para obtener el número de actores de cada película. Usa los resultados de la consulta para identificar las 5 películas con más actores.

**Solución oculta**

MATCH (m)<-[:ACTED\_IN]-(a)

RETURN m.title AS movie, count(a) AS Actors\_of\_the\_Movie

ORDER BY Actors\_of\_the\_Movie DESC;

Hay otras operaciones de agregación utiles, como por ejemplo la operación “collect”. Ejecutar las dos sentencias siguientes, analizad sus resultados y descubrir que hace la operación “collect”.

MATCH (a)-[:ACTED\_IN]->(m)<-[:DIRECTED]-(d) RETURN a.name, d.name, m.title;

MATCH (a)-[:ACTED\_IN]->(m)<-[:DIRECTED]-(d) RETURN a.name, d.name, collect(m.title);

Otras operaciones de agregación son el min, el max, el avg y el sum. Utiliza algunas de estas operaciones para obtener el número medio de actores por película.

**Solución oculta**

MATCH (m)<-[:ACTED\_IN]-(a)

WITH m AS Movie, count(a) AS Num\_Actors

RETURN avg(Num\_Actors)

Realizad una consulta que devuelva aquellos actores que han trabajado con “Cristina Ricci”.

**Solución oculta**

MATCH (actor)-[:ACTED\_IN]->(movie),

(Christina {name: "Christina Ricci"})-[:ACTED\_IN]->(movie)

RETURN DISTINCT actor.name;

Ahora modificad la consulta anterior para que devuelva también el número de películas en las que cada actor ha trabajado con Cristina Ricci.

**Solución oculta**

MATCH (actor)-[:ACTED\_IN]->(movie),

(Christina {name: "Christina Ricci"})-[:ACTED\_IN]->(movie)

RETURN DISTINCT actor.name, count(movie);

# Creación de indices

Ahora vamos a ver cómo crear índices para aumentar la velocidad de las consultas en Chyper. Primero crearemos un índice sobre el atributo “name” del tipo “person”:

CREATE INDEX ON :Person(name)

Cread un índice que permita incrementar la velocidad al consultar películas por su título.

CREATE INDEX ON :Movies(title)

# Consultas avanzadas

Vamos a consultar las películas anteriores al 2000 donde apareció Kevin Bacon.

**Solución oculta**

MATCH (actor:Person {name:"Kevin Bacon"})-[r:ACTED\_IN]->(movie)

WHERE movie.released < 2000

RETURN DISTINCT movie.title, movie.released;

Queremos obtener todas las películas que obtuvieron una puntuación mayor a un 70 sobre 100 en alguna revisión de un crítico.

**Solución oculta**

MATCH (reviewer)-[r:REVIEWED]->(movie)

WHERE r.rating > 70

RETURN reviewer.name, movie.title, r.rating;

Obtener el valor medio que le han dado los revisores a todas las películas revisadas.

**Solución oculta**

MATCH (reviewer)-[r:REVIEWED]->(movie)

RETURN movie.title, avg(r.rating)

Repetir la consulta anterior (películas con una puntuación mayor a 70) teniendo en cuenta la puntuación média de las películas.

**Solución oculta**

MATCH (reviewer)-[r:REVIEWED]->(movie)

WITH movie.title AS Title, avg(r.rating) AS Rating

WHERE Rating > 70

RETURN Title, Rating

Obtener todas las películas que contengan en su título la cadena de caracteres “Matrix”.

**Solución oculta**

MATCH (movie)

WHERE movie.title=~".\*Matrix.\*"

RETURN DISTINCT movie.title;

Identificar los actors que han actuado en alguna película bajo el nombre de “Morpheus”.

**Solución oculta**

MATCH (actor)-[r:ACTED\_IN]->(movie)

WHERE r.roles = "Morpheus"

RETURN actor.name, movie.title

¡¡No funciona!!! Será por el tipo de datos de r.roles…

**Solución oculta**

MATCH (actor)-[r:ACTED\_IN]->(movie)

WHERE "Morpheus" IN r.roles

RETURN actor.name, movie.title

Identificad los 5 actores que han realizado más películas.

**Solución oculta**

MATCH (a:Person)-[:ACTED\_IN]->()

RETURN a.name, count(\*) AS count

ORDER BY count DESC LIMIT 5;

# Materialización de relaciones

Suponed que debemos hacer algunos análisis al respecto de quien ha trabajado con quien. Para agilizar los análisis, queremos materializar la relación “WORKED\_WITH”, que indica que dos actores han trabajado juntos. La relación debe contener un atributo que indique las veces que han trabajado juntos:

**Solución oculta**

MATCH (a:Person)-[]->(m:Movie)<-[]-(b:Person)

WITH a AS PersonA, b AS PersonB, count(\*) AS Number\_of\_times

CREATE UNIQUE (PersonA)-

[:HAS\_WORKED\_WITH {times:Number\_of\_times}]-(PersonB)

Ahora podemos consultar fácilmente las personas que han trabajado juntas en más de 5 películas.

**Solución oculta**

MATCH (p1:Person)-[r:HAS\_WORKED\_WITH]-(p2:Person)

WHERE r.times > 5

RETURN p1.name, p2.name, r.times

ORDER BY r.times DESC

Fijaos que cada pareja aparece duplicada, podemos evitarlo añadiendo la direccionalidad a la relación.

**Solución oculta**

MATCH (p1:Person)-[r:HAS\_WORKED\_WITH]->(p2:Person)

WHERE r.times > 5

RETURN p1.name, p2.name, r.times

ORDER BY r.times DESC

Fijaos también que nos aparece una actriz relacionada con si misma “Nancy Meyers”. Eso es porque no hemos contemplado esa posibilidad al crear las relaciones. Podriamos recrear-las correctamente:

MATCH (p1:Person)-[r:HAS\_WORKED\_WITH]-(p2:Person)

DELETE r

MATCH (a:Person)-[]->(m:Movie)<-[]-(b:Person)

WHERE a.name<>b.name

WITH a AS PersonA, b AS PersonB, count(\*) AS Number\_of\_times

CREATE UNIQUE (PersonA)-

[:HAS\_WORKED\_WITH {times:Number\_of\_times}]-(PersonB)

MATCH (p1:Person)-[r:HAS\_WORKED\_WITH]->(p2:Person)

WHERE r.times > 5

RETURN p1.name, p2.name, r.times

ORDER BY r.times DESC

Por último utilizaremos la relación HAS\_WORKED\_WITH para aprender a utilizar la navegación múltiple por relaciones. Suponed que queremos saber que conocidos tiene Kevin Bacon. Entenderemos por conocidos aquellos que han trabajado con él.

**Solución oculta**

MATCH (kevin:Person {name:"Kevin Bacon"})-

[:HAS\_WORKED\_WITH]-(coworker)

RETURN DISTINCT coworker.name;

Pero suponed que no solo queremos obtener la gente con la que ha trabajado (conocidos) sino también los conocidos con los que ha trabajado, lo que se denomina “Friend of a Friend”.

**Solución oculta**

MATCH (kevin:Person {name:"Kevin Bacon"})-

[:HAS\_WORKED\_WITH\*2]-(fof)

RETURN DISTINCT fof.name;

Y para rizar el rizo podríamos preguntar sólo por los de segundo nivel, es decir los que han trabajado con gente que ha trabajado con él.

**Solución oculta**

MATCH (kevin:Person {name:"Kevin Bacon"})-

[:HAS\_WORKED\_WITH\*2]-(fof)

WHERE NOT (kevin)-[:HAS\_WORKED\_WITH]-(fof) AND kevin<>fof

RETURN DISTINCT fof.name;