**Sistemas de Información de Gestión y**

**Bussiness Intelligence**

**11.11.2019**

He estado investigando los diferentes tipos de software que podemos aplicar en esta asignatura. En primer lugar, he visitado, tal y como hicimos en clase, DB-Engines (<https://db-engines.com/en/>) para observar qué tipos de bases de datos son las qué mas se utilizan y por qué motivos. Aquí es cuando observé que aparece Neo4j como el número 1 en base de datos de grafos. Como el profesor nos lo mencionó está herramienta en clase decidí buscar información en Google acerca de Neo4j y cómo funciona.

Neo4j es una base de datos de grafos optimizada para relacionar de forma fácil y rápida la información. Esta herramienta permite ejecutar sentencias con 100 millones de nodos en un tiempo corto comparado con otro tipo de base de datos. El lenguaje de esta herramienta es Cypher.

Para entender mejor el funcionamiento de Neo4j he visualizado el siguiente video:

(<https://www.youtube.com/watch?v=k3h_y9w_7l4>**)**

Gracias al vídeo o comprendido como realizar sentencias en Neo4j de forma sencilla. También he entendido que a las relacionas se les trata como ciudadanos de primera clase de forma que tienen la misma relevancia que los propios nodos. Un ejemplo claro sería el siguiente:

C**REATE (m:Player{name:"Lionel Messi"}),**

**(b:Team{name:"Barcelona"})**

**WITH m, b**

**CREATE (m)-[p:PlaysFor]->(b)**

**SET p.since=date("2001-02-01")**

**RETURN m, p, b**

De esta forma tendríamos un jugador llamado Messi que juega en el Barcelona desde el 1-2-2001.

Ahora bien, una vez que he aprendido las bases de Neo4j me descargué el programa para poder realizar mis propios grafos. Una vez todo instalado y configurado utilicé el segundo ejemplo donde se muestra un conjunto de personas y sus amigos:

// Create nodes

CREATE (rob:Person{name:'Roberto'}), (isidro:Person{name:'Isidro'}),

(tony:Person{name:'Antonio'}), (nora:Person{name:'Nora'}),

(lily:Person{name:'Lilian'}), (freddy:Person{name:'Alfredo'}),

(lucas:Person{name:'Lucas'}), (mau:Person{name:'Mauricio'}),

(alb:Person{name:'Albina'}), (reg:Person{name:'Regina'}),

(j:Person{name:'Joaquín'}), (julian:Person{name:'Julián'})

// Create relationships

CREATE

(rob)-[:FriendsWith]->(isidro), (rob)-[:FriendsWith]->(tony), (rob)-[:FriendsWith]->(reg),

(rob)-[:FriendsWith]->(mau), (rob)-[:FriendsWith]->(julian),

(tony)-[:FriendsWith]->(reg), (tony)-[:FriendsWith]->(j),

(alb)-[:FriendsWith]->(reg), (lily)-[:FriendsWith]->(isidro), (lily)-[:FriendsWith]->(j),

(mau)-[:FriendsWith]->(lucas), (lucas)-[:FriendsWith]->(nora), (freddy)-[:FriendsWith]->(nora);

// Query the relationships

MATCH friendships=()-[:FriendsWith]-()

RETURN friendships

Como bien se puede observar para realizar la búsqueda de todos los amigos de todas las personas es simple. Si solo quisiéramos saber los amigos de amigos solo tendríamos que especificar en primer lugar que persona tomamos como referencia para buscar los amigos de sus amigos. Posteriormente añadimos un \*2 dentro de los corchetes de FriendsWith de forma que no cogería todos los amigos sino los amigos que están en el segundo círculo de la persona de referencia.

Para poder conseguir el mejor resultado en la asignatura me dispongo a ver todos los videos del canal Tensor4Dummies de forma que pue3de utilizar tensorflow de forma fácil y también me dispongo a estudiar más en profundidad como programar en Python.