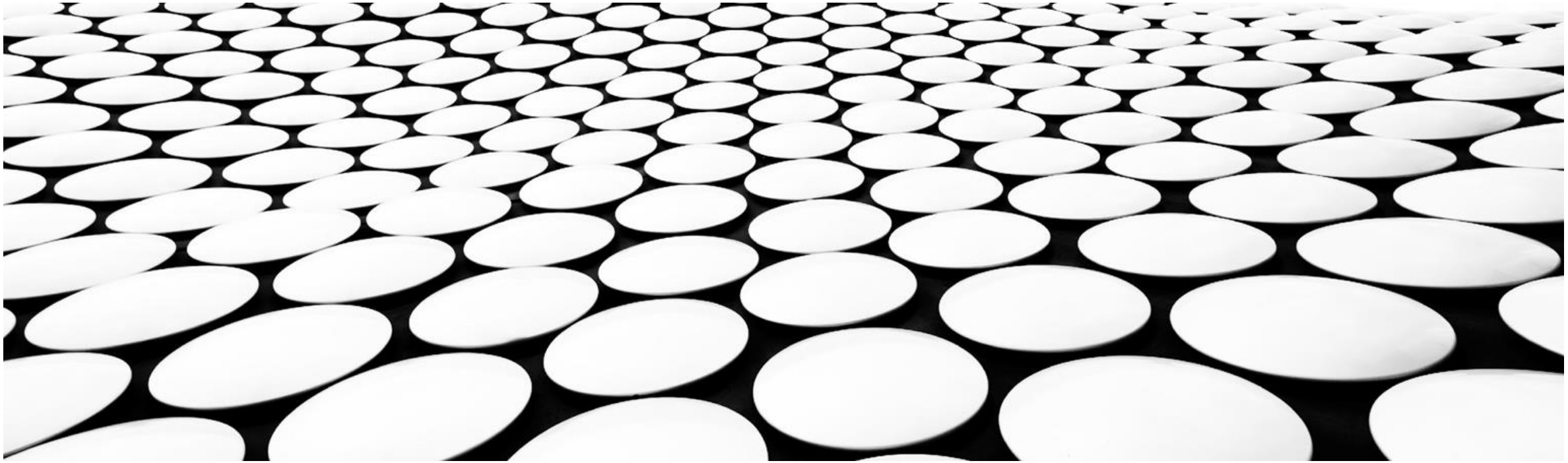
NEO4J

BASE DE DATOS ORIENTADA A GRAFOS



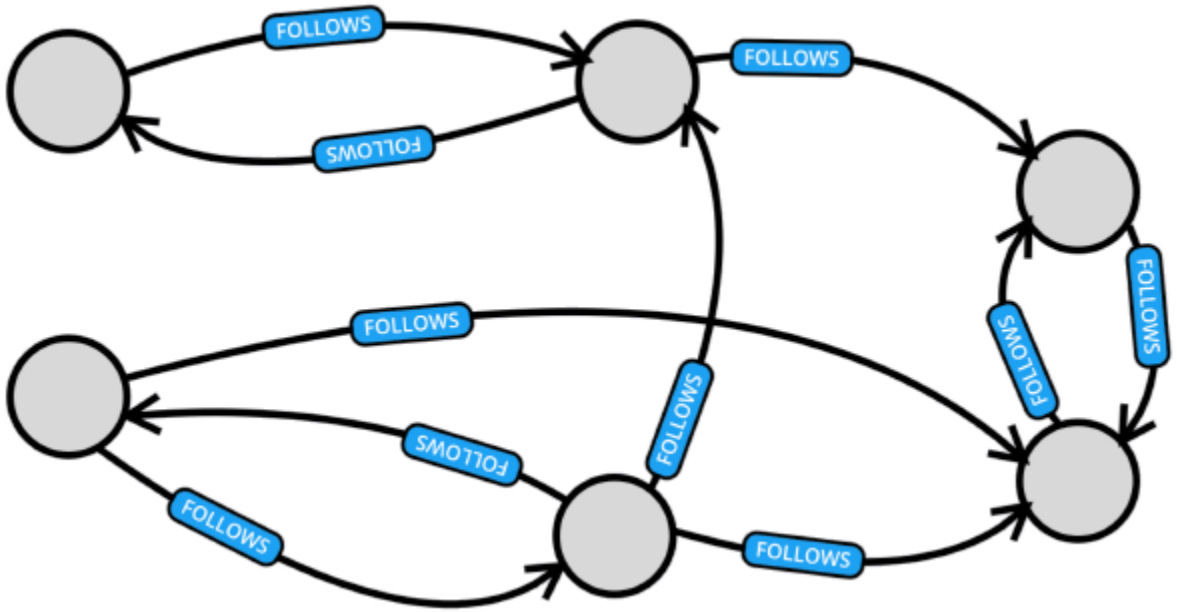
NEO4J - INTRODUCCIÓN

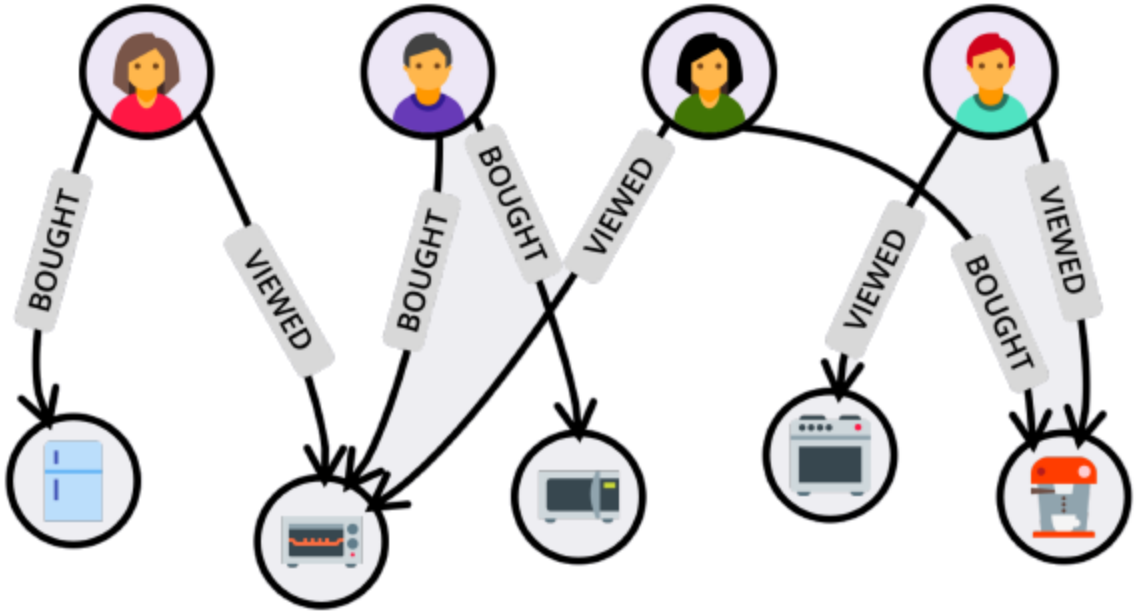
- Las bases de datos orientadas a grafos (**BDOG**) ayudan a encontrar relaciones y dar sentido al puzzle completo.
- Una de las más popular es **Neo4j**. Implementado en Java.
- Su primera versión fue lanzada en febrero de 2010 y en estos momentos está bajo dos tipos de licencia:
  - Licencia comercial.
  - Affero General Public License (**AGPL**).
- Desarrollada en ***Neo Technology***.

NEO4J - CARACTERÍSTICAS

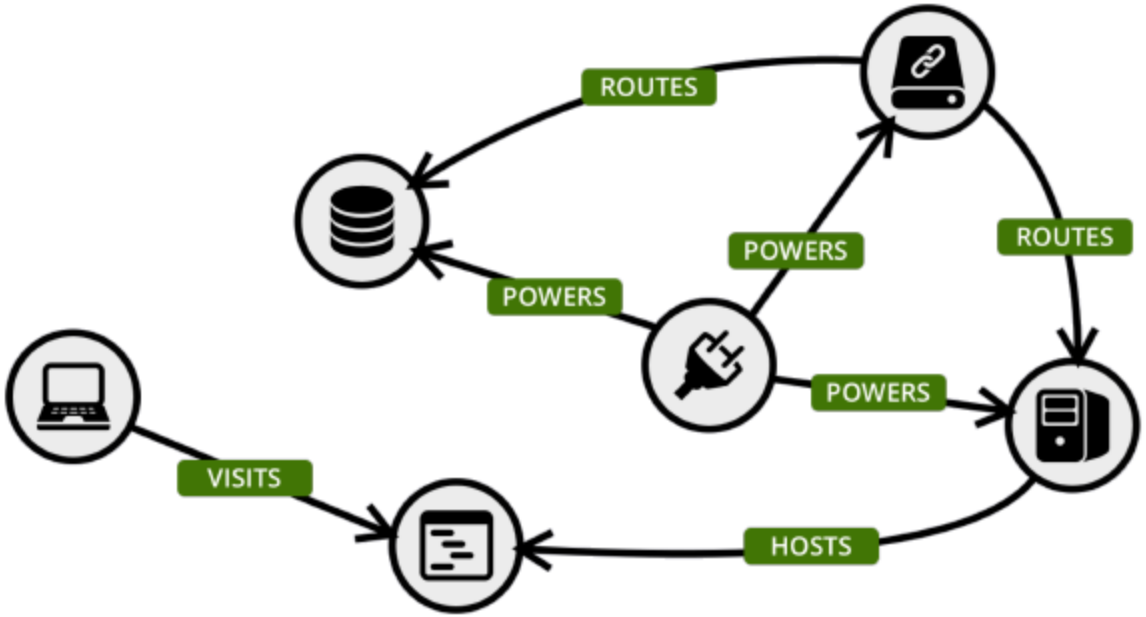
- Neo4j usa grafos para representar datos y las relaciones entre ellos. Un grafo se define como cualquier representación gráfica formada por vértices y aristas.
- Grafos no dirigidos, Grafos dirigidos, Grafos con peso, Grafos con etiquetas.
- **Grafos de propiedad**: es un grafo con peso, con etiquetas y donde podemos asignar propiedades tanto a nodos como relaciones (por ejemplo, cuestiones como nombre, edad, país de residencia, nacimiento).
- **Neo4j** utiliza grafos de propiedad para extraer valor añadido de los datos con gran rendimiento y de una forma ágil, flexible y escalable.

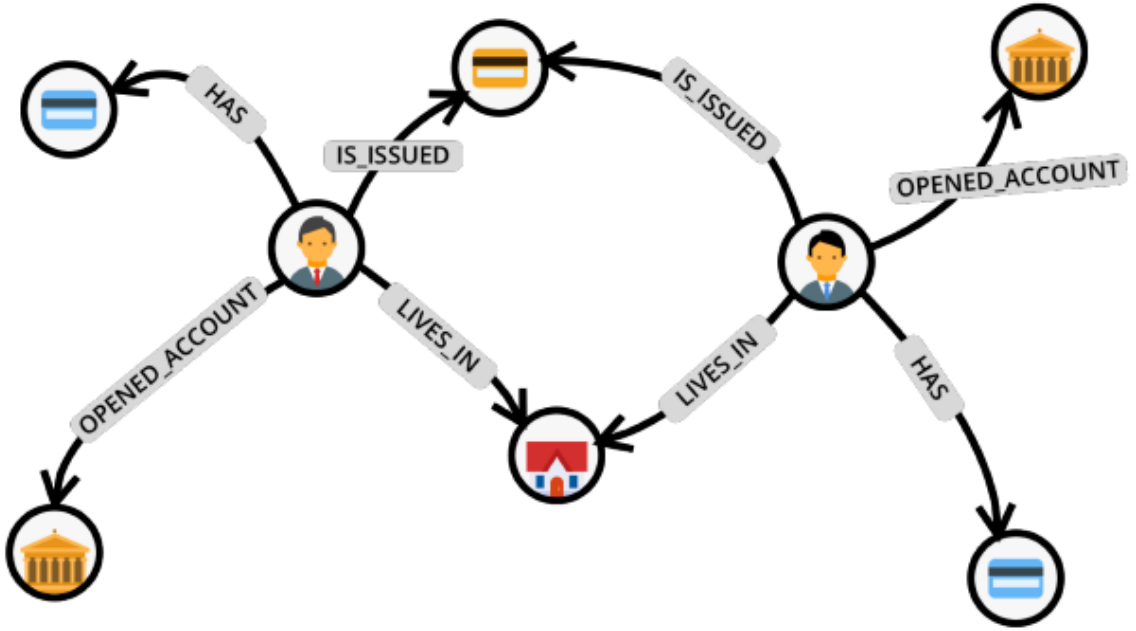
NEO4J – APLICACIONES





NEO4J – APLICACIONES





NEO4J – SIN ESQUEMAS (SCHEMALESS)

- **Neo4j** es una base de datos *schemaless*.
- Esto no quiere decir que la base de datos no utilice un esquema.
- El esquema es flexible y no es necesario que esté prefijado ni definido antes de la introducción de nuevos datos.
- Permite que las modificaciones (o alteraciones) del esquema se puedan realizar fácilmente.
- Los nodos de un mismo tipo pueden tener propiedades distintas.

NEO4J – SIN ESQUEMAS (SCHEMALESS)

- Aún así, el esquema tiene bastante protagonismo.
- Las etiquetas permiten indicar el tipo de los nodos y las relaciones.
- Los diseñadores deberán tener en cuenta, antes de crear la base de datos, qué tipos de nodos y de relaciones se van a tener que representar. Además de sus principales propiedades y las etiquetas correspondientes.
- Por otro lado, en **Neo4j**, el esquema puede utilizarse para indicar algunas restricciones de integridad.

NEO4J – RESTRICCIONES DE INTEGRIDAD

- **Unicidad**: permite indicar que el valor de una propiedad debe ser único para todos los nodos del mismo tipo. Por ejemplo, se podría indicar que no puede haber dos libros con el mismo ISBN (en otras palabras, no existen dos nodos de tipo *:Book* con el mismo valor para la propiedad isbn).
- **Existencia**: permite indicar que una propiedad debe existir para todos los nodos de un tipo. Por ejemplo, se podría indicar que todo libro debe tener un ISBN y un título (todos los nodos de tipo *:Book* deben tener las propiedades isbn y título).
- **Clave**: permite indicar que una propiedad es clave para todos los nodos de un determinado tipo, es decir, que todos sus nodos deben tener definida la propiedad y que el valor de la propiedad es único.

NEO4J - CYPHER

- **Cypher** es un lenguaje declarativo, inspirado en SQL, que permite manipular datos en Neo4j.
- Guían de referencia: https://neo4j.com/docs/cypher-refcard/current/

NEO4J - INSTALACIÓN

- Requiere JAVA.
- LINUX:
  - https://neo4j.com/docs/operations-manual/current/installation/linux/
- Windows:
  - https://neo4j.com/docs/operations-manual/current/installation/windows/
- Docker:
  - https://neo4j.com/docs/operations-manual/current/docker/introduction/

NEO4J – CYPHER

- **Cypher** es el lenguaje propio de **Neo4j**.
- Los nodos se representan con círculos y las relaciones con flechas. Su representación consiste en poner nodos entre paréntesis y relaciones como flechas etiquetadas por corchetes.
  - `(nodo)-[:RELACIÓN]->(nodo)`
- Las propiedades de los nodos o las relaciones se indican con una estructura parecida a un diccionario:
  - `(nodo {name:'Oscar', surname:'Garcia'})`
- Las etiquetas se indican después de definir la variable:
  - `(nodo:Person {name:'Oscar', surname:'Garcia'})`

NEO4J – CREACIÓN DE NODOS CON PROPIEDADES

- `CREATE (pa:Persona {nombre:'Paco', nacio:1964})`
- `CREATE (ju:Persona {nombre:'Juan', nacio:1967})`
- `CREATE (an:Persona {nombre:'Andrés', nacio:1961})`
- `CREATE (hu:Persona {nombre:'Hugo', nacio:1960})`
- `CREATE (na:Persona {nombre:'Natalia', nacio:1967})`

NEO4J - RELACIONES

- **MATCH** (pa {nombre:'Paco'}), (ju {nombre:'Juan'})

  CREATE (pa)-[:AMIGO_DE {role:['Amigo de Trabajo']}]->(ju)

- **MATCH** (pa {nombre :'Paco'}), (an {nombre :'Andrés'})

  CREATE (pa)-[:AMIGO_DE {role:['Amigo de Trabajo']}]->(an)

- **MATCH** (ju {nombre :'Juan'}), (hu{ nombre:'Hugo'})

  CREATE (ju)-[:AMIGO_DE {role:['Amigo de la infancia']}]->(hu)

NEO4J - RELACIONES

- **MATCH** (pa {nombre:'Paco'}), (tel {denominacion:'Telefonica'})
  CREATE (pa)-[:TRABAJA_EN {cargo:['Director de Marketing']}]->(tel)

- **MATCH** (an {nombre:'Andres'}), (tel {denominacion:'Telefonica'})
  CREATE (an)-[: TRABAJA_EN{cargo:['Director de Marketing']}]->(tel)

- **MATCH** (na {nombre:'Natalia'}), (rep {denominacion:'Repsol'})
  CREATE (na)-[: TRABAJA_EN{cargo:['Director de Marketing']}]->(rep)

- **MATCH** (na {nombre:'Natalia'}), (hu {nombre:'Hugo'})
  CREATE (na)-[:ES_FAMILIAR_DE {tipo:['Prima']}]->(hu)

NEO4J – ALGUNAS CONSULTAS

- `MATCH (n) RETURN n`
- `MATCH (n:Persona) RETURN n`
- `MATCH(a:Persona) WHERE a.nombre="Paco" RETURN a;`
- `MATCH(a:Compañia) WHERE a.oficinaCentral="Madrid" RETURN a;`
- `MATCH amigos=(a:Persona{nombre:'Paco'})-[:AMIGO_DE]-(amigo) RETURN amigos`
- `MATCH amigos=(a:Persona{nombre:'Paco'})-[:AMIGO_DE]-(amigo) RETURN amigo.nombre`
- `MATCH ({nombre:'Paco'})-[:AMIGO_DE]->(amigos) RETURN amigos`

GRACIAS

MARRERO LUCIANO