Se puede utilizar todas las conexiones, sin repetirlas ?
Euler path theorem: No tiene solucion, si el SISTEMA tiene mas de 2 nodos con cant. de conectores impar. No se puede recorrer todo sin pasar por el mismo "puente"

Se puede pensar a los elementos como:
Nodos —> Los Sustantivos
Relaciones —> Los Verbos

Que sea nativa, significa que no necesita calcular indices y realizar distintos joins, para acceder a la informacion consulta. Neo4j almacena directamente las relaciones, teniendo un desempeño mucho mas rapido, en especial en consultas que necesitan realizar varios joins para ser realizadas. Las bases de datos NO NATIVAS, ensamblan las relaciones a medida que se realizan las consultas, a diferencia de las nativas, las cuales ya las tienen cargadas.

Base de datos de grafos "nativa", en cuanto a la forma en que guarda y la forma en que procesa

Nombre completo: "Base de datos de grafos con propiedades de etiquetas"

- Es una base de datos, no es una visualización de datos.
- Paradigma ACID (no BASIC). Es un base de datos transaccional, es decir, que cumple con las características ACID.
- Así como las bases de datos relacionales guardan los datos en forma de tabla, Neo4j los guarda en forma de grafos, como un network, una red.

grafos: Nodos conectados por relaciones

"Grafo Conexo", todas las partes estan vinculadas mutuamente

Introducción – conceptos básicos

Cypher query sintaxis (o SQL for graphs)

Lenguaje declarativo: qué es lo que queremos, no cómo accedemos a los datos

() Nodes Nodo(clave:valor)

[] Relationships

- Cada nodo puede tener propiedades
- Los nodos pueden ser etiquetados con una o más etiquetas
- Las relaciones también pueden tener nombre y propiedades

()-[]->() Pattern

alias de un nodo **(var:)-[]->()**

La gran ventaja de esta base de datos, es que te permite especificar muchas cosas sobre las relaciones (los verbos)

(var:Process)-[]->() Process representa la etiqueta del nodo

(var:Process:Step)-[]->() un nodo se puede etiquetar con muchos nombres

(var:Process{name:'Offer'})-[]->() etiqueta con una propiedad

()-[:HIRED]->() la etiqueta de la relación; la relación se puede taggear con un nombre

()-[h:HIRED]->() etiqueta asignada al alias "h"

()-[h:HIRED{type:'ON DEMAND'}]->() la relación tiene una propiedad (type) y su valor es "ON DEMAND"

Comandos - queries

MATCH y **RETURN** palabras reservadas MUY UTILIZADAS

Comandos útiles

- Ctrl+space menú contextual
- Shift + enter salto de línea

```
//Eliminar todos los objetos de una base de datos
MATCH (allObjects) DETACH DELETE allObjects

//Crear un nodo
CREATE (helloWorld) RETURN helloWorld
//El motor crea un id pero es a fines internos, no usarlo.

//Para recuperar lo creado:
MATCH (allDBNodes) RETURN allDBNodes

//Otra forma de crear un nodo
CREATE (nodeVar{name: 'ciaoMondo'}) RETURN nodeVar
```

```
// obtengo el nodo creado
MATCH (nodeVar{name: 'ciaoMondo'}) RETURN nodeVar

//Usando la cláusula Where
MATCH (nodeVar)
WHERE  nodeVar.name = 'ciaoMondo'
RETURN nodeVar

//Usando como clave de búsqueda el prefijo de un string
MATCH (nodeVar)
WHERE  nodeVar.name STARTS WITH 'ciaoMondo'
RETURN nodeVar

//Que contenga, en alguna parte del nombre,  el string dado
MATCH (nodeVar)
WHERE  nodeVar.name CONTAINS 'ciaoMondo'
RETURN nodeVar

//puedo crear nodos de a uno o de forma masiva
CREATE (city{name:'Buenos Aires'}) RETURN city

CREATE
(co {name:'Corrientes'})
,(es {name:'Escobar'})
,(ol {name:'Olivos'})
RETURN co, es, ol
```

```
//Parámetros: crearlos y usarlos en la misma sesión; formato json.
//para llamarlo voy a usar “nodes”; son tres elementos con un atributo en común (name)
:params {nodes: [{name:'BA'},{name:'RE'},{name:'BO'}]}

//Comando UNWIND: permite transformar cualquier lista en listas individuales; permite iterar sobre
        colecciones
UNWIND $nodes AS nodeIt
CREATE (city{name:nodeIt.name})
RETURN city

//establecer un atributo adicional en un nodo ya creado
MATCH (city{name: 'BA'})
Set city.temp= 16
RETURN city
//La primera vez lo crea, la posterior lo actualiza
```

//RELACIONES – parámetros

```
//sin especificar ni atributos ni nombres
CREATE (buenosaires)-[rel:COUNTRY]->(argentina)
RETURN buenosaires, argentina

// código para crear los nodos de ciudades con tres elementos
:params // vamos a crear los nodos de ciudades con estos tres elementos
cities: [
            {
      name:'Buenos Aires'
      ,isCapital: true
      ,knownFor:'La Reina del Plata
            }
```

```
        ,{
        name:'Resistencia'
        ,isCapital: true
        ,knownFor: ' Ciudad de las Esculturas'
        }
        ,{
        name:'Corrientes'
        , isCapital: true
        , knownFor: 'Capital del carnaval'
        }
        ,{
        name:'La Plata'
        , isCapital: false
        , knownFor: 'La ciudad de las diagonales'
        }
        ,{
        name:'Bogotá'
        , isCapital: true
        , knownFor: 'La Atenas de Sudamérica'
        }
        ,{
        name:'Escobar'
        , isCapital: false
        , knownFor: 'Capital Nacional de la flor'
        }
]
```

```
//idem para países
:params
countries: [
                {
                name:'Argentina'
                } ,
                {
                name:'Colombia'
                }
        ]
```

Tengo dos parámetros: voy a crear nodos con la ayuda de los parámetros creados

```
UNWIND $countries AS countryIterator
        CREATE (countryVar:Country)
        SET countryVar=countryIterator
RETURN countryVar

//De la misma forma creo las ciudades mutatis mutandi
UNWIND $cities AS citiesIterator
        CREATE (cityVar:City)
        SET cityVar=citiesIterator
RETURN cityVar

//Creando relaciones
MATCH (city {name:'Buenos Aires'}) ,  (country {name:'Argentina'})
        CREATE (city) -[relVar :COUNTRY]-> (country)
RETURN city, country
```