



Bases de Datos Orientadas a Grafos

Ingeniería de Datos II
1º Cuatrimestre 2024

Ing. Damián Arnaudo - Facultad de Ingeniería y Ciencias Exactas

Taxonomía de BD NoSQL

Bases de Datos Documentales

Bases de Datos Orientadas a Grafos

Bases de Datos de Clave/Valor

Bases de Datos Orientadas a Objetos

Bases de Datos Tabulares

Bases de Datos Multidimensionales o de Arrays

Bases de Datos Orientadas a Grafos

- ▶ Basada en estructura de grafos (nodos y aristas) para representar y almacenar la información.
- ▶ Permiten un fácil almacenamiento de las entidades y las relaciones entre estas.
- ▶ Las entidades se representan a través de nodos con sus propiedades.
- ▶ Las aristas (arcos) son las relaciones entre los nodos y también pueden tener propiedades.

Bases de Datos Orientadas a Grafos

- ▶ Podemos hacer una analogía entre un Nodo y un objeto instanciado en una aplicación.
- ▶ Los arcos, aristas o relaciones tienen dirección.
- ▶ Los nodos están vinculados por relaciones que nos permiten encontrar patrones (trabaja_en, casado_con, etc.).
- ▶ La organización del grafo nos permite que los datos sean almacenados una vez y luego interpretados de distintas maneras basándonos en sus relaciones

Bases de Datos Orientadas a Grafos

Cuando deberíamos usarlas: Redes sociales o laborales



Datos Interconectados

Relaciones entre entidades de diferentes dominios (por ejemplo, redes y operaciones de IT, identidad y gestión de accesos). En una sola base de datos estas relaciones son más valiosas

Bases de Datos Orientadas a Grafos

Cuando deberíamos usarlas: Motores de Recomendaciones

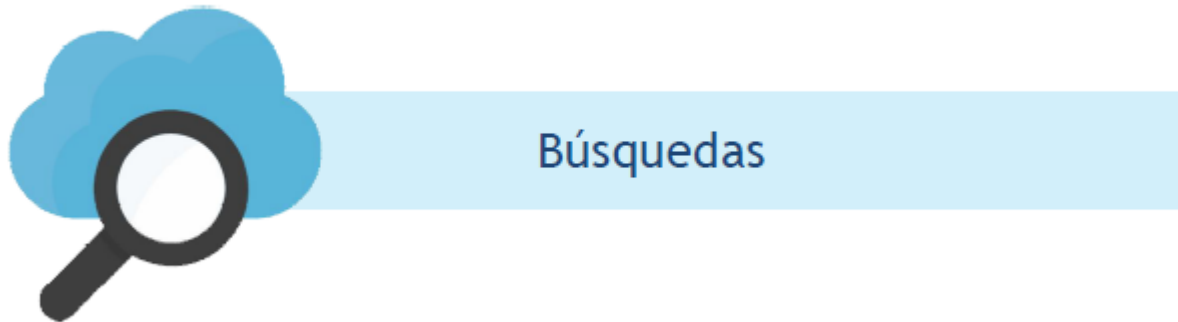


Motores de Recomendaciones

Como los nodos y las relaciones se crean en el sistema, pueden ser utilizados para hacer recomendaciones como “sus amigos también han comprado este producto” o “al facturar este artículo, estos otros artículos suelen ser facturados” en tiempo real

Bases de Datos Orientadas a Grafos

Cuando deberíamos usarlas: Búsquedas recursivas



- ▶ Sistemas con búsquedas recursivas con n niveles.
- ▶ Búsqueda de patrones para detectar el fraude en las transacciones en tiempo real
- ▶ Gestión de datos: Datos que naturalmente se modelan como grafos (líneas de organización y producción, etc.)

Cuando NO deberíamos usarlas:

Sistemas que requieren de actualizaciones masivas sobre todas las entidades o un conjunto de entidades para un atributo o conjunto de atributos específicos

Sistemas que requieren una alta distribución de datos debido a su gran tamaño

Pero si hablamos
de relaciones por
que no un RDBMS

Las BDs relacionales no
guardan las relaciones entre
los datos, solo tablas con id.



Esto implica

Gran número de
Joins

Joins recursivos para
representaciones de
árboles o jerarquías.

Esquema cambiante
de las relaciones
(frecuentes
reorganizaciones).

Bases de Datos Orientadas a Grafos





Neo4j

- ▶ Neo4j es un sistema de base de datos orientado a grafos de código abierto
- ▶ Está desarrollado en Java
- ▶ Utiliza como modelo de datos, los grafos basados en propiedades
- ▶ Es una de las pocas bases de datos orientadas a grafos que cuentan con un lenguaje de consulta propio: **Cypher**

Neo4j

- ▶ El DBMS puede administrar un servidor independiente o un grupo de servidores en un Clúster (una instancia de Neo4j es un proceso de Java que ejecuta el código del servidor Neo4j).
- ▶ El contexto de ejecución es un entorno de ejecución para una solicitud (una consulta, una transacción o una función o procedimiento interno).
- ▶ Base de datos es la partición administrativa de un DBMS (una estructura física de archivos organizados dentro de un directorio o carpeta, que tiene el mismo nombre que la base de datos).

Quienes usan Neo4j



Neo4j



- ▶ Solo hay un gráfico dentro de cada base de datos, pero se puede referenciar a un gráfico específico usando el nombre de la base de datos.
- ▶ Es posible hacer referencia a múltiples gráficos dentro de la misma transacción y consulta Cypher.

Neo4j

- ▶ La edición de Neo4j utilizada determina el número de posibles bases de datos:
 - ▶ Community: una sola base de datos de usuario
 - ▶ Enterprise: múltiples bases de datos de usuarios.
- ▶ Todas incluyen una base de datos de sistema, que contiene metadatos y configuración de seguridad.
- ▶ Los comandos administrativos están restringidos a usuarios con privilegios administrativos específicos.

Neo4j

El Clustering de Neo4j ofrece tres características principales:

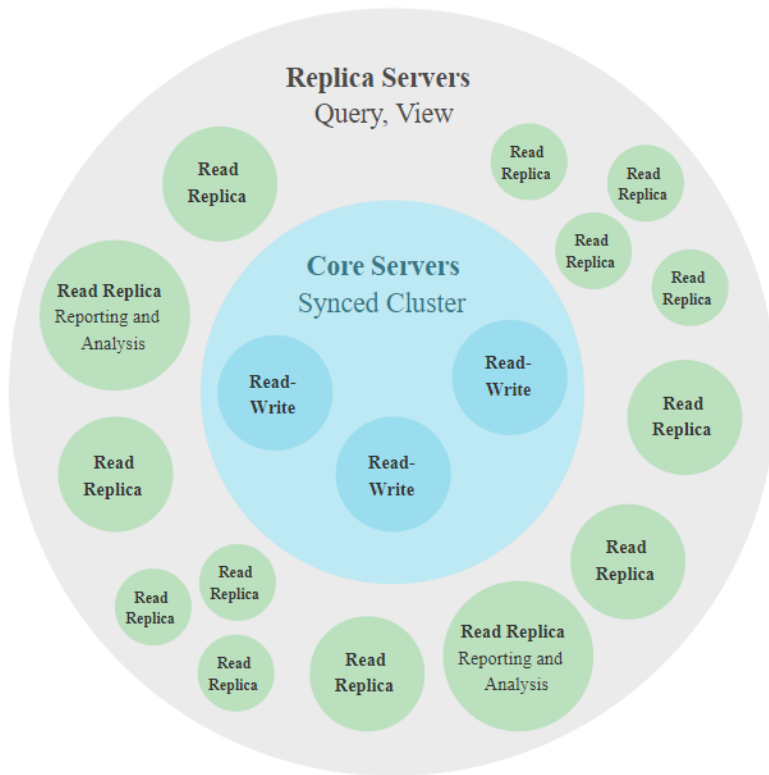
- 1) **Seguridad:** los servidores centrales proporcionan una plataforma tolerante a fallos para el procesamiento de transacciones que permanecerá disponible mientras la mayoría de los servidores centrales estén funcionando.
- 2) **Escala:** las réplicas de lectura proporcionan una plataforma enormemente escalable para consultas de gráficos que permite ejecutar cargas de trabajo de gráficos muy grandes en una topología ampliamente distribuida.
- 3) **Consistencia causal:** cuando se invoca, se garantiza que una aplicación cliente leerá al menos sus propias escrituras.

Neo4j

Esto permite:

- ▶ Un sistema del usuario final completamente functional.
- ▶ Leer y escribir en la base de datos aun en caso de múltiples fallas de hardware y red.
- ▶ Simplifica las interacciones con la base de datos.

Neo4j



- ▶ El clúster posee dos roles diferentes para los servidores: núcleos y réplicas de lectura.
- ▶ Fundamentales en cualquier implementación de producción asumiendo roles sobre la tolerancia a fallas y la escalabilidad del clúster en general.

Neo4j - Clustering

- ▶ Servidores centrales: Su principal responsabilidad es proteger los datos mediante replicación de todas las transacciones (protocolo Raft). En la práctica, esto significa que una vez que la mayoría de los servidores centrales de un clúster ($N / 2 + 1$) han aceptado la transacción, es seguro reconocer el compromiso con la aplicación del usuario final.
- ▶ Esto tiene un impacto en la latencia de escritura (las escrituras serán reconocidas por la mayoría más rápida, pero a medida que aumenta el número de servidores centrales en el clúster, también lo hace el tamaño de la mayoría necesaria para reconocer una escritura).

Neo4j - Clustering

- ▶ Debemos tener relativamente pocas máquinas en un clúster Core Server esto se calcula con la fórmula

$$M = 2F + 1$$

- ▶ donde M es el número de servidores centrales necesarios para tolerar fallas y F el número de cores fallidos. (Ej. para tolerar dos servers fallidos, necesitamos un clúster de cinco Cores).
- ▶ También es posible crear un clúster que consta de solo dos núcleos. Sin embargo, ese clúster no será tolerante a errores. Si uno de los dos servidores falla, el servidor restante pasará a ser de solo lectura.

Neo4j - Clustering

- ▶ Las réplicas de lectura tienen responsabilidad de escalar las cargas de trabajo de los gráficos.
- ▶ Actúan como cachés para los datos de gráficos que los servidores centrales protegen. Son capaces de ejecutar consultas y procedimientos arbitrarios (solo lectura).
- ▶ Las réplicas de lectura se trabajan de forma asincrónica desde los servidores centrales mediante el envío del registro de transacciones.

Neo4j - Clustering

- ▶ Muchas réplicas de lectura pueden recibir datos de una cantidad relativamente pequeña de servidores centrales, lo que permite escalar una gran carga de trabajo.
- ▶ Las réplicas de lectura deben ser muchas (relativamente) y tratarse como desechables.
- ▶ Tampoco afecta su pérdida a las capacidades de tolerancia a fallas del clúster, solo su porción de proceso.

Neo4j

Consistencia Eventual

- ▶ Es importante pensar en cómo las aplicaciones usarán la base de datos para realizar su trabajo (operaciones de lectura y escritura sobre un grafo) teniendo en cuenta las operaciones realizadas con anterioridad.
- ▶ La coherencia causal es un modelo que se utiliza en la informática distribuida. Garantiza que todas las instancias del sistema vean las operaciones relacionadas en el mismo orden.

Neo4j

Consistencia Eventual

- ▶ Se garantiza que las aplicaciones cliente leerán sus propias escrituras, independientemente de la instancia con la que se comuniquen (permite asumir a los clientes que son tratados por un único servidor).
- ▶ La coherencia causal hace posible escribir en servidores centrales (donde los datos están seguros) y leer esas escrituras desde una réplica de lectura (donde las operaciones de gráficos se escalan horizontalmente).
- ▶ Por ejemplo, la escritura que creó una cuenta de usuario estará presente cuando ese mismo usuario intente iniciar sesión posteriormente.

Neo4j Tareas Administrativas

BackUp: on lines off line - total - incremental

Restore: instance - cluster

Actualizaciones.

Autorización y Autenticación: Nativa - LDAP - KERBEROS.

Seguridad: SSL y Certificados

Neo4j Monitoreo

- ▶ Métricas.
- ▶ Log.
- ▶ Consultas y Transacciones.
- ▶ Conexiones y Accesos.
- ▶ Estado del Cluster.
- ▶ Base de Datos.





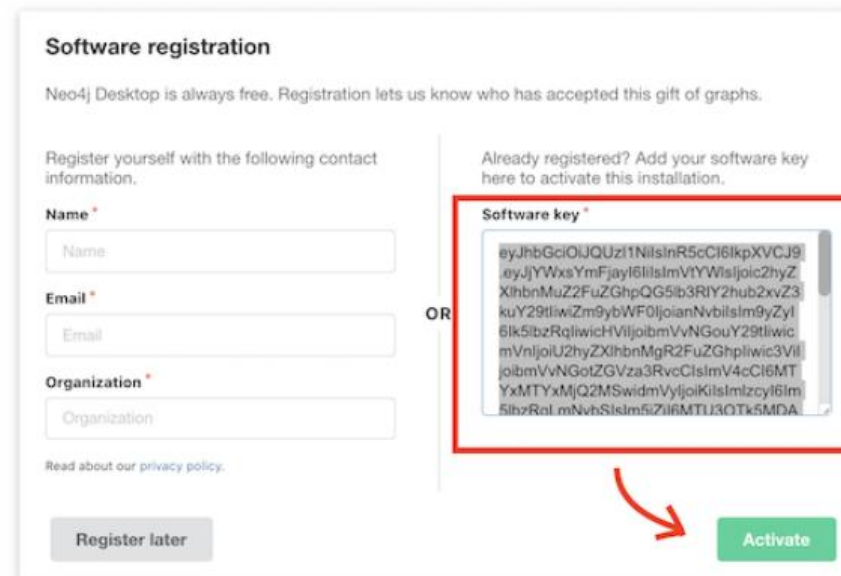
Instalación y configuración

Neo4j

Instalación y configuración

Descargar Neo4j Desktop de <https://neo4j.com/deployment-center/>

Seguir la guía de instalación, según el sistema operativo, de <https://neo4j.com/docs/operations-manual/current/installation/>



The image shows a 'Software registration' form for Neo4j Desktop. It has two main sections: 'Register yourself with the following contact information' and 'Already registered? Add your software key here to activate this installation.' The first section has input fields for 'Name', 'Email', and 'Organization', with a 'Register later' button at the bottom. The second section has a 'Software key' input field, which is highlighted with a red box. A red arrow points from the 'Activate' button to the 'Software key' field. The text 'OR' is placed between the two sections.

Software registration

Neo4j Desktop is always free. Registration lets us know who has accepted this gift of graphs.

Register yourself with the following contact information.

Name *

Name

Email *

Email

Organization *

Organization

[Read about our privacy policy.](#)

[Register later](#)

OR

Already registered? Add your software key here to activate this installation.

Software key *

```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...
```

[Activate](#)



Cypher

Neo4j

Cypher

- ▶ Toda la documentación de Neo4j está disponible en <https://neo4j.com/docs/>
- ▶ La guía de Cypher es <https://neo4j.com/docs/cypher-manual/current/introduction/>
- ▶ Hoja de Trucos de Cypher https://neo4j.com/docs/cypher-cheat-sheet/5/neo4j-community#_write_only_query_structure

Cypher

- ▶ Su sintaxis utiliza un estilo **ascii-art** para hacer los comandos muy intuitivos.
- ▶ Mientras los nodos se representan con círculos y las relaciones con flechas, su representación en ascii-art consiste en poner nodos entre paréntesis y relaciones 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'})`

- ▶ Y las etiquetas se indican después de definir la variable:

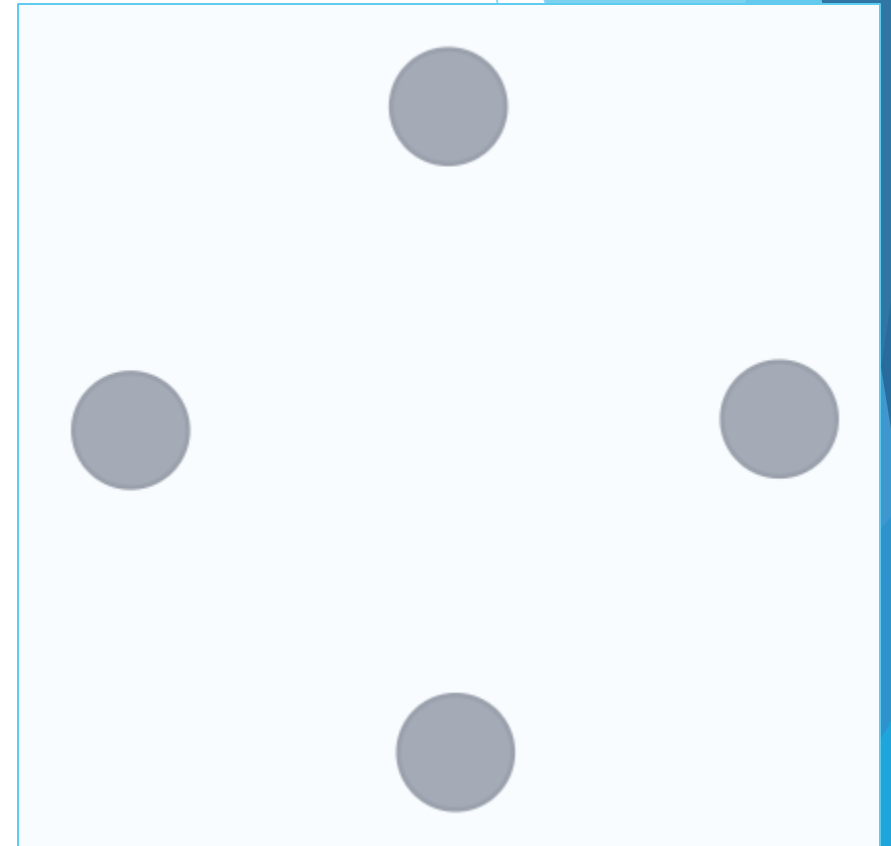
`(nodo:Person {name:'Oscar', surname:'Garcia'})`

Cypher

1. Empezamos por crear un nodo:
`CREATE (os {name:'Óscar', surname:'García'})`
2. Veamos si se ha creado:
`MATCH (n) RETURN n`
3. Creemos un par de nodos más y visualicémoslos:
va -> name: Vanessa, surname: Gómez
co -> name: Coffee, type: Drink
na -> name: Napolitana, type: Dessert
4. ¿Como será la carga múltiple?

CREATE

```
(va {name:'Vanessa', surname:'Gómez'}),  
(co {name:'Coffee', type:'Drink'}),  
(na {name:'Napolitana', type:'Dessert'})
```



Cypher

- ▶ Para actualizar un nodo utilizamos la clausula **SET**:

```
MATCH (p:Person {name: 'Óscar'})  
SET p.age = 35  
RETURN p
```

- ▶ RETURN p: devuelve el nodo actualizado
- ▶ Se puede personalizar esta consulta para actualizar cualquier otra propiedad del nodo.
- ▶ Para actualizar múltiples propiedades, se debe agregar más cláusulas SET.
- ▶ Y si necesitas actualizar varios nodos al mismo tiempo, puedes ajustar la parte MATCH de la consulta para seleccionar los nodos deseados.

Cypher

1. Vamos a añadir algunas etiquetas (ejecutar cada línea por separado):

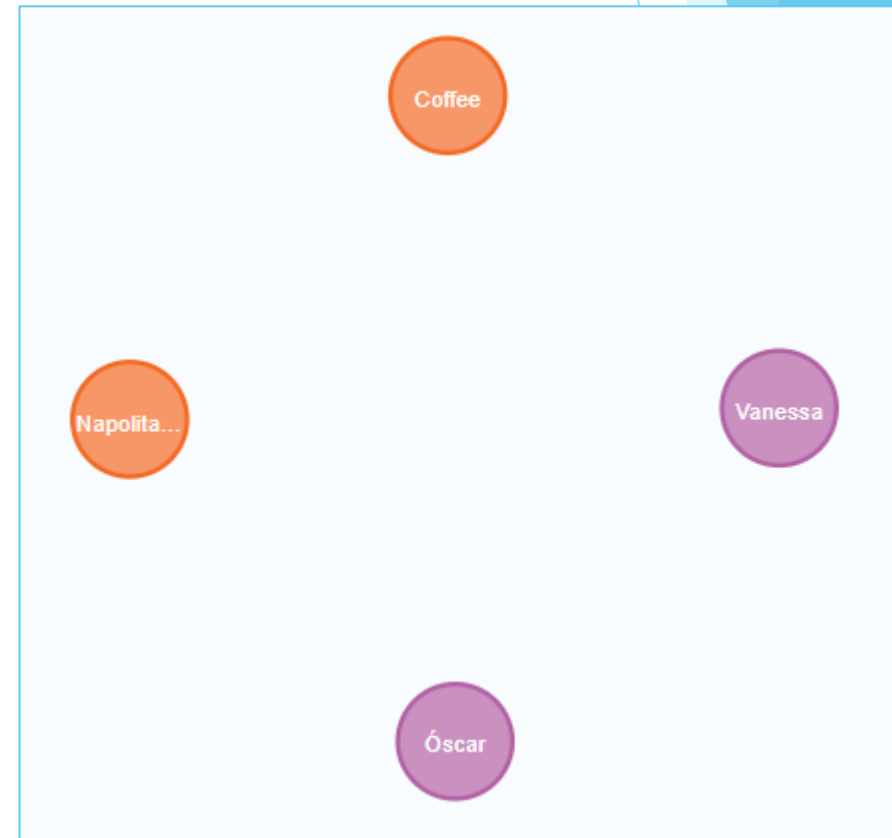
```
MATCH (os {name:'Óscar'}) SET os:Person
```

```
MATCH (va {name:'Vanessa'}) SET va:Person
```

```
MATCH (co {name:'Coffee'}) SET co:Food
```

```
MATCH (na {name:'Napolitana'}) SET na:Food
```

2. Visualicemos otra vez para ver el cambio.



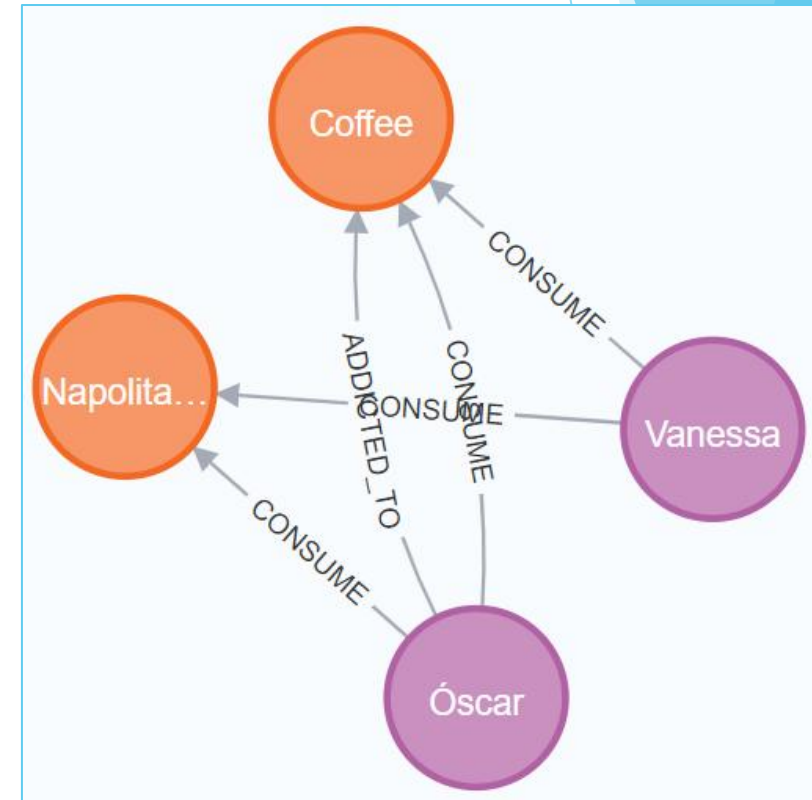
Cypher

- ▶ Ahora vamos a establecer una relación entre dos nodos

```
MATCH (o {name:'Óscar'}), (c {name:'Coffee'})  
CREATE (o)-[:ADDICTED_TO]->(c)
```

- ▶ Podemos crear relaciones entre nodos de forma simultánea:

```
MATCH (p:Person), (f:Food)  
CREATE (p)-[:CONSUME]->(f)
```



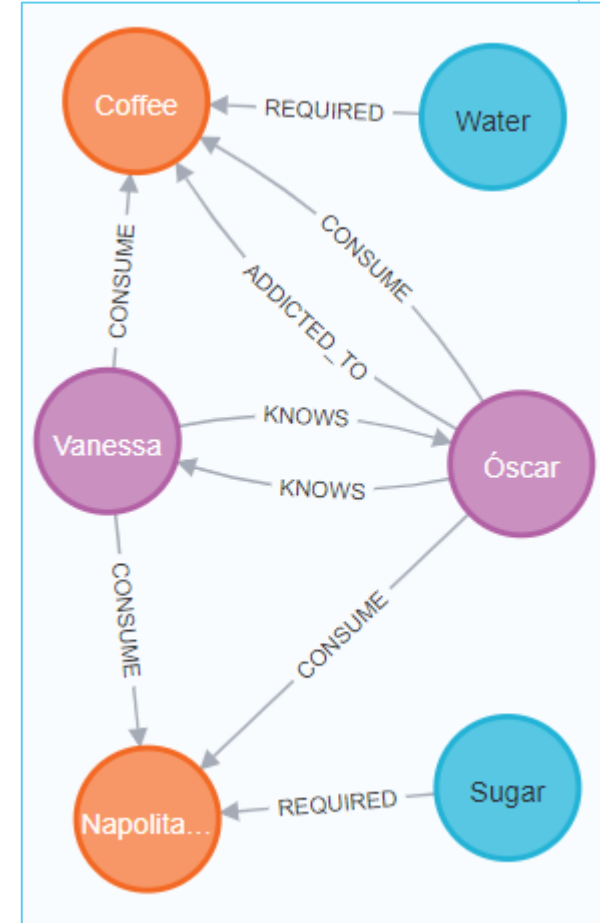
Cypher

- Creemos una relación bidireccional:

```
MATCH (a:Person), (b:Person)
WHERE a.name = 'Óscar' AND b.name = 'Vanessa'
CREATE (a)-[:KNOWS]->(b), (b)-[:KNOWS]->(a)
```

- También podemos crear un par de nodos y relacionarlos en un sólo bloque:

```
CREATE
    (sug:Ingredient {name:'Sugar', type:'Liquid'}),
    (wat:Ingredient {name:'Water', type:'Liquid'})
WITH sug, wat
MATCH (na {name:'Napolitana'}), (co {name:'Coffee'})
CREATE
    (sug)-[:REQUIRED]->(na), (wat)-[:REQUIRED]->(co)
```



Cypher

Algunas Alternativas de visualización

- ▶ Todos los nodos con la misma etiqueta:
`MATCH (n:Person) RETURN n`
- ▶ Un par de atributos en forma tabular:
`MATCH (n:Person) RETURN n.name AS Names, n.surname AS Surnames`
- ▶ Query relacional:
`MATCH (o {name:'Óscar'})-[:CONSUME]->(f) RETURN f`
- ▶ La misma query pero solo un atributo en formato tabular:
`MATCH (o {name:'Óscar'})-[:CONSUME]->(f) RETURN f.name AS Food`
- ▶ Un par de queries más. ¿Qué hacen?:
`MATCH (o {name:'Óscar'})-[r]-(f) RETURN type(r)`
`MATCH (n:Person) WHERE n.name = 'Vanessa' RETURN n.surname AS Surname`

Cypher

- ▶ Para eliminar podemos usar las cláusulas **DELETE** o **DETACH DELETE**
- ▶ Eliminar un nodo:

```
MATCH (p:Person {name: 'Alice'})  
DELETE p
```
- ▶ Eliminar una relación:

```
MATCH (:Person {name: 'Alice'})-[r:KNOWS]-()  
DELETE r
```
- ▶ Eliminar un nodo y todas sus relaciones, y relaciones entrantes y salientes:

```
MATCH (p:Person {name: 'Alice'})  
DETACH DELETE p
```
- ▶ Eliminar completamente nuestro grafo:

```
MATCH (n)  
DETACH DELETE n
```