

Neo4j: BDD Orientada a Grafos

Escribimos en el lenguaje Cypher Query Language, que esta insiprado en SQL.

Neo4J lleva un modelo de grafo **etiqueta-propiedad**

Nodos

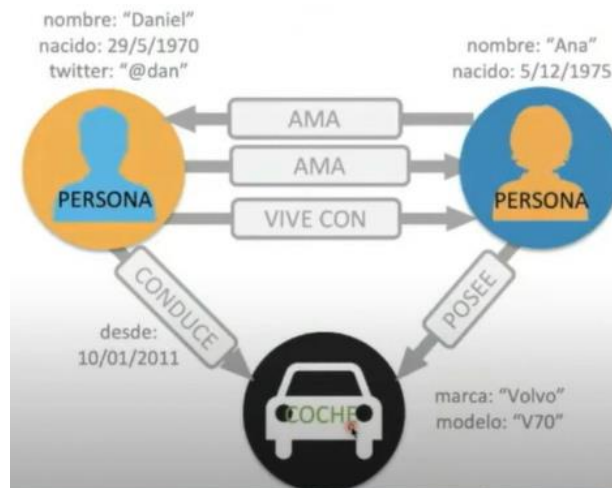
- Pueden tener etiquetas para su clasificacion
- Las etiquetas tienen índices nativos

Relaciones

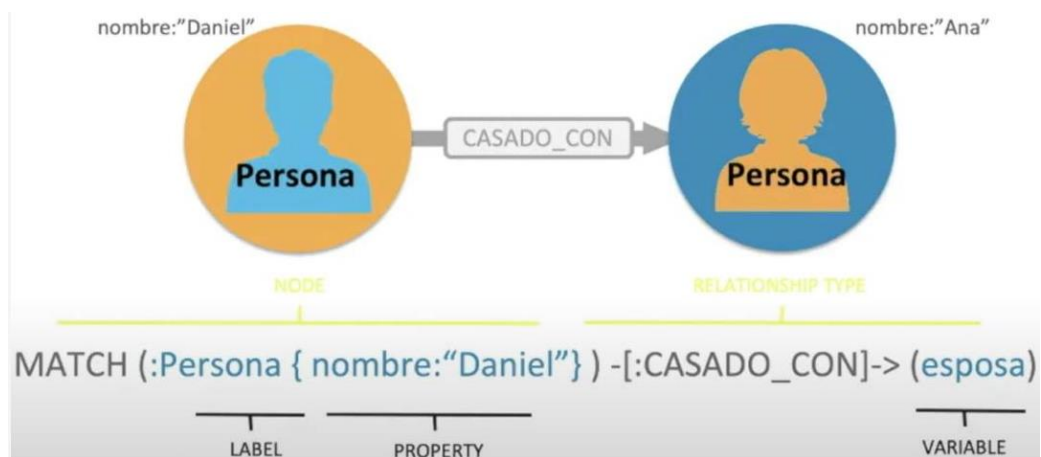
- Relacionan nodos por **tipo y dirección**: todo es direccionado en Neo4j

Propiedades

- Atributos de nodoso y relaciones
- Almacenados como **pares clave-valor**
- Pueden tener índices e índices compuestos

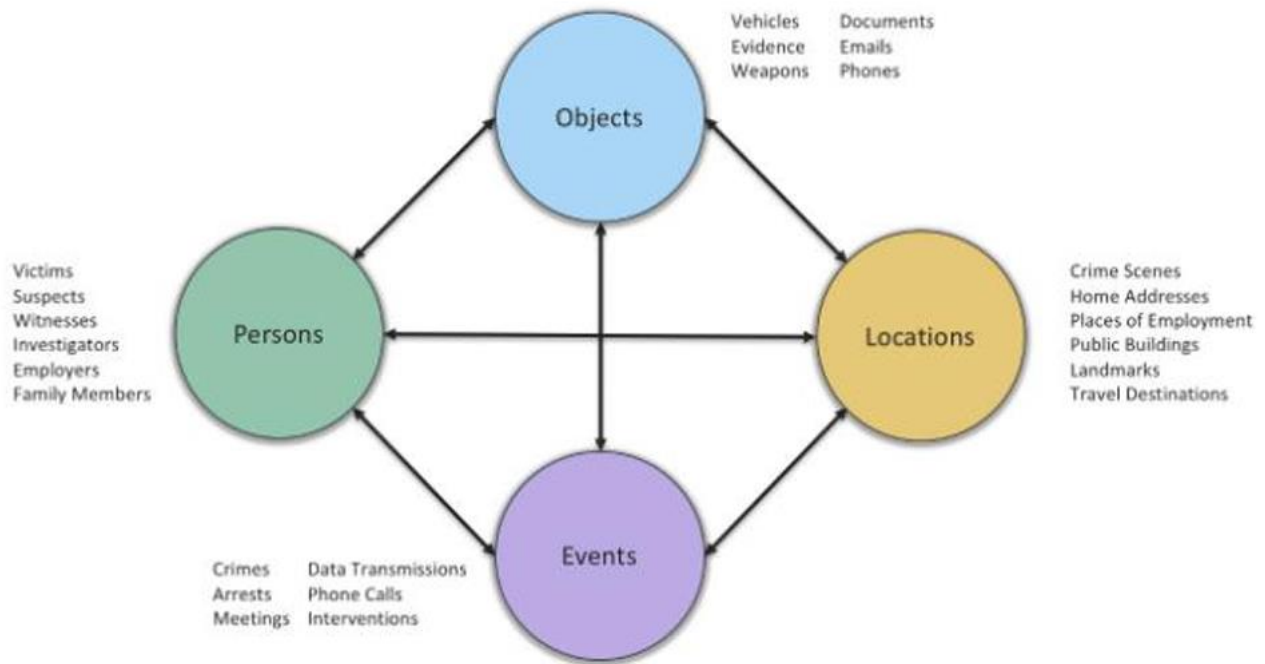


Estructura de consulta en Cypher

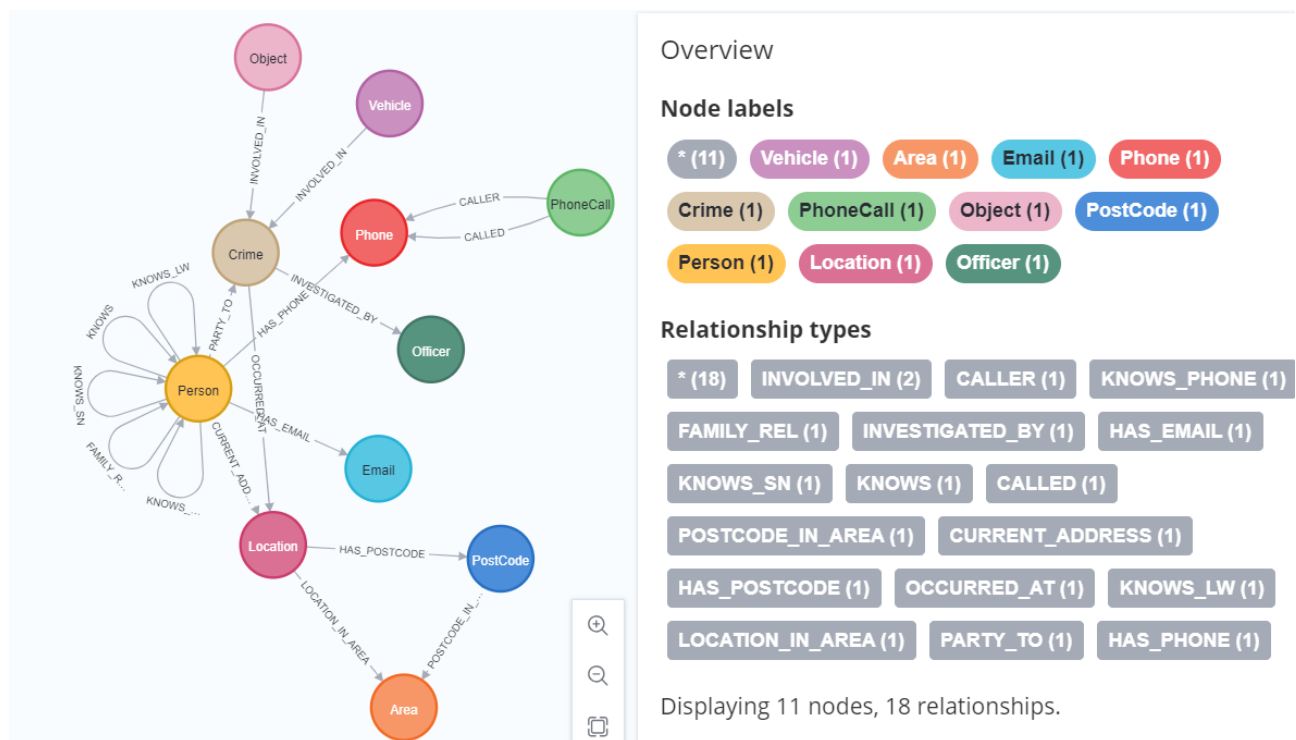


Ejemplo que vamos a usar: BDD de crímenes de Manchester

Modelo POLE: Person, Object, Location, Event



Y la base tiene las siguientes relaciones entre los nodos:



Para conocer como esta armada la base podemos usar los siguientes métodos:

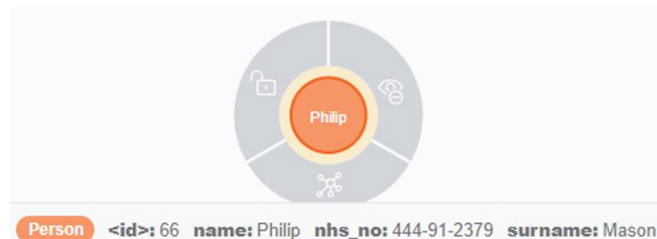
- **CALL db.schema.visualization()** arma el grafo de arriba con todas las relaciones
- **CALL db.labels** devuelve el nombre de los nodos
- **CALL db.relationshipTypes** devuelve las flechas que conectan los nodos
- **CALL db.propertyKeys**

Elementos de la base

NODOS

- Tienen valores para distintas propiedades (id interno)
- Pueden tener varias etiquetas:

Ejemplo



ARCOS (las flechas)

- Tienen un único tipo
- Pueden tener propiedades
- Son **direccionadas**, y con una sola direccion
- Vinculan dos nodos, o un único nodo si se vincula consigo mismo

Armado de queries

MATCH *Patrón*
[WHERE Filtros]
RETURN [DISTINCT] *Respuestas*
[ORDER BY Expresiones]
[SKIP cant. LIMIT cant.]

Patrón

- Estructura del subgrafo a revisar



Ejemplo: ce:Person{name: "Gregory"}

Filtros (opcional)

- Condiciones que debe cumplir el subgrafo para ser devuelto

Los filtros en WHERE no se escriben de la misma forma que los filtros de Patron pero **son equivalentes**:

Filtros en WHERE

```
MATCH (ce:Person)
WHERE ce.name = "Gregory"
RETURN ce
```

Filtros en patrón

```
MATCH (ce:Person {name : "Gregory" })
RETURN ce
```

Respuestas

- Se puede devolver nodos y arcos completos, o el valor de algunas propiedades
- Se separan con comas
- El DISTINCT es opcional

Orden y paginado (opcional)

- Similar a SQL

Manejo de Strings

Los valores son case sensitive. Además de los operadores clásicos, está el de expresiones regulares =~. *Ejemplo: a.name =~ "A.*"*.

También cuenta con STARTS WITH, ENDS WITH y CONTAINS

Manejo de labels

Esto es distinto a otras propiedades del nodo.

Ejemplo: devuelva el nombre y etiquetas de personas que no tienen "e" en sus nombres

```
MATCH (n:Person)
WHERE NOT n.name CONTAINS "e "
RETURN n.name, labels(n)
LIMIT 20
```

Se devuelven con la
función labels

Consultas Taller

1. Muestre en orden alfabético los nombres de las primeras 10 personas apellidadas "Smith"

```
MATCH (n:Person)
WHERE n.surname = "Smith"
RETURN n.name, n.surname
ORDER BY n.name ASC
LIMIT 10
```

2. Muestre la marca y modelos de los vehículos del año 2013

```
MATCH
(v:Vehicle)
WHERE v.year = '2013' RETURN v.make, v.model
```


3. Muestre el nombre, apellido y rango de los oficiales cuyos apellidos comiencen con 'Mc', ordenados por rango (rank)

```
MATCH (o:Officer)
WHERE o.surname STARTS WITH "Mc"
RETURN o.name, o.surname, o.rank
ORDER BY o.rank ASC
```

Para vincular un nodo con otro usamos la notación **--**. Indicación de arco:

En ciertos casos podemos querer indicar cual es la dirección y la relación en particular que queremos usar. Para hacer eso **debe ir entre corchetes**

```
MATCH
(n:Person)-[d:KNOWS_LW]->(m:Person)
WHERE m.surname = "Gordon" AND
m.name="Craig"
RETURN n, m
```



Además, podemos colocar **filtros para el arco**

```
MATCH
(a:Person)-[r:FAMILY_REL{rel_type:"SIBLING"}]-
(b:Person)
RETURN a,b
limit 20
```

Filtros de arco -Extra-

- Permiten definir dirección del arco

-[]- -[]-> <-[]-

- Permiten definir cantidades de arcos

-[*]- -[*cant]-
 -[*min..]- -[*..max]-
 -[*min..max]-

Si tengo pocos grados de separación puedo decir Nodo – [REL] -> (Nodo) -> [REL]. Por ejemplo

```
(o:Person)-[e:KNOWS]->(n:Person)-[d:KNOWS]->(m:Person)
```

Mas ejercicios...

4. Muestre el grafo de las locations en el área M30. Cuantos nodos hay?

```
MATCH (l:Location)--(a:Area)
WHERE a.areaCode = 'M30'
RETURN l
```

Si hacemos un RETURN COUNT(l) podemos ver que hay 211 nodos

5. Muestre el grafo de todos que conocen a alguien que conoce a Gordon Craig.

Es decir que busco aquellos que no conocen directamente a Gordon Craig, sino que tienen una persona de por medio. Es decir, están a 1 grado de separación de Gordon.

Puedo usar el filtro de grado de separación, o la relación anidada

```
MATCH (o:Person)-[e:KNOWS]-(n:Person)-[d:KNOWS]-(m:Person)
WHERE m.surname = "Gordon" AND m.name="Craig"
RETURN o,n,m
```

```
MATCH (p:Person)-[*2]-(gc:Person)
WHERE gc.surname = "Gordon" AND gc.name = "Craig"
RETURN p, gc
```

No es necesario que pongamos las flechas porque si se conocen, es mutuo, la flecha puede ir para los dos lados

6. Muestre las personas que están a distancia 3 de Gordon Craig.

Se refiere a los grados de distancia. Se pueden delimitar con el filtro de cantidad de arcos.

```
MATCH (p:Person)-[*3]-(gc:Person)
WHERE gc.surname = "Gordon" AND gc.name = "Craig"
RETURN p, gc
```


Alias de Subgrafos

Se les puede poner nombres a las consultas que armamos para después usarla como un subset de los datos enteros

```
MATCH s=((n:Person)-[HAS_PHONE]->(p:Phone)<-[]-(o:PhoneCall))
WHERE n.surname="Gordon" and n.name="Craig"
RETURN s
```

• Funciones sobre subgrafos:

- Length(s)
- Relationships(s)
- Nodes(s)

Subgrafos en filtros

Personas que conocen a Craig Gordon y no a Bonnie Robinson:

```
MATCH (n:Person{surname:"Gordon", name:"Craig"})-[]-(p:Person)
WHERE NOT (p)-[]-(:Person{surname:"Robinson", name:"Bonnie"})
RETURN DISTINCT p.name, p.surname
```

Tenemos un subgrafo Gordon que tiene una relación con “p” persona, y pedimos que esa p no este relacionada con Bonnie Robinson

Varios subgrafos

Se pueden poner varios subgrafos en el MATCH

Llamadas de Gordon a Moore

```
MATCH (n:Person{surname:"Gordon", name:"Craig"})--(p:Phone)--(o:PhoneCall),(m:Person{surname:"Moore", name:"Judith"})--(p1:Phone)--(o:PhoneCall)
RETURN o
```

Los subgrafos del match se separan con una coma.

Caminos mas cortos

Es común querer el camino mas corto entre dos nodos

```
MATCH s=shortestPath( (a)-[*]-(b) )
RETURN s
```

También se puede usar allShortestPaths. Si se especifica dirección de arco, todos la deben cumplir.

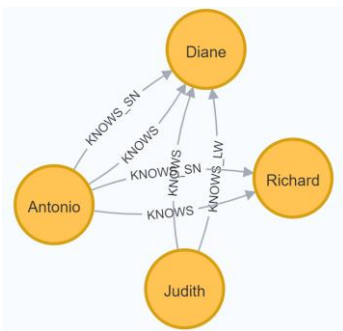
Mas ejercicios...

7. Muestre las personas conocidas de Roger Brooks que no participaron en ningún crimen.

```
MATCH (n:Person{surname:"Brooks", name:"Roger"})-[]-(p:Person)
WHERE NOT (p)-[]-(c:Crime)
RETURN n, p
```

8. Muestre el camino más corto de Judith Moore a Richard Green.

```
MATCH s=shortestPath((n:Person{surname:"Moore",
name:"Judith"})-[*]-(p:Person{surname:"Green", name:
"Richard"}))
RETURN s
```



9. Encuentre los oficiales que investigaron los crímenes cometidos en 165 Laurel Street.

Address: "165 Laurel Street"

```
MATCH (o:Officer)-[i:INVESTIGATED_BY]-(c:Crime)
WHERE (c)-[]-(l:Location{address:"165 Laurel Street"})
RETURN o
```

Funciones de Agregacion

Al incluir funciones de agregación en el RETURN, funciona similar a SQL, pero **no hace falta el groupby**

Ejemplo: devuelve el año del auto mas nuevo

```
MATCH (n:Vehicle) RETURN max(n.year)
```

Si se devuelven propiedades con función de agregación, se agrupa por ellas:

Ejemplo 1: Devolver cuantos vehículos marca Ford hay en la base

```
MATCH (a:Vehicle)
WHERE a.make CONTAINS "Ford"
RETURN a.make, COUNT(*)
```

Ejemplo 2: Para las 10 marcas de autos con más unidades, mostrar el promedio de año y cantidad.

```
MATCH (a:Vehicle)
RETURN a.make, AVG(toInteger(a.year)), COUNT(a)
ORDER BY COUNT(a) DESC
LIMIT 10
```

WITH

Es un reemplazo del RETURN. Permite manipular los resultados antes de pasar a la siguiente parte de la consulta.

Se puede usar para filtrar agregación.

Ejemplo: Hacer la misma consulta anterior, pero mostrando solo las marcas que tienen mas de 45 vehículos en la base

```
MATCH (a:Vehicle)
WITH a.make as marca, AVG(toInteger(a.year)) as prom_year
, COUNT(a) AS cant
WHERE cant > 45
RETURN marca, prom_year, cant
```

WITH – Múltiples MATCH

Con el resultado de un WITH, puedo anidar otro match

```
MATCH (a:Vehicle)
WITH MAX(a.year) AS maxyear
MATCH (v:Vehicle) WHERE v.year = maxyear
RETURN v.make, v.model, v.year
```

Mas consultas...

10. Obtenga el modelo, marca y año del auto más viejo de la base.

```
MATCH(a:Vehicle) RETURN a.make, a.model, min(a.year) ORDER BY
min(a.year) ASC LIMIT 1
```

a.make	a.model	min(a.year)
"Chrysler"	"Imperial"	"1926"

11. ¿A qué distancia se encuentra el auto más viejo de Roger Brooks?

```
MATCH (v:Vehicle) WITH MIN(v.year) AS oldest
MATCH (v:Vehicle), (p:Person{name: 'Roger', surname: 'Brooks'})
WHERE v.year = oldest
MATCH s=shortestPath((v)-[*]-(p))
RETURN length(s)
```

Rta: 8

12. Devuelva el nombre y apellido de personas que conozcan más de 10 personas.

Puedo contabilizar relaciones, poniendo el arco en el match como se muestra a continuación

```
MATCH (p:Person) -[k:KNOWS]- (p1:Person)
WITH COUNT(*) AS cantidad_conocidos, p.name AS nombre, p.surname AS
surname
MATCH (p:Person)
WHERE cantidad_conocidos ≥ 10 and p.name = nombre and p.surname =
surname
RETURN p.name, p.surname, cantidad_conocidos
```

13. ¿Cuántas personas hay en la base?

```
MATCH (p:Person) RETURN COUNT(*)
```

COUNT(*)
369

¿Cuántos tiene teléfono?

¿Cuántos tienen mail?

Puedo contabilizar todo junto uniendo MATCHs con WITH

```
MATCH (p:Person) WITH COUNT(*) as cant_personas
MATCH (p:Person)-[HAS_EMAIL]-(e:Email) WITH COUNT(*) as
cant_email, cant_personas
MATCH (p:Person)-[HAS_PHONE]-(ph:Phone) WITH COUNT(*) as
cant_phone, cant_email, cant_personas
RETURN cant_personas, cant_email, cant_phone
```

Si anidamos WITHs, tenemos que escribir la var que quiero poner en el return cada vez que termina el with. Por eso en el AS de los WITHs siempre voy escribiendo el cant_personas, y así sucesivamente.