Insertar elementos en un grafo en Neo4J

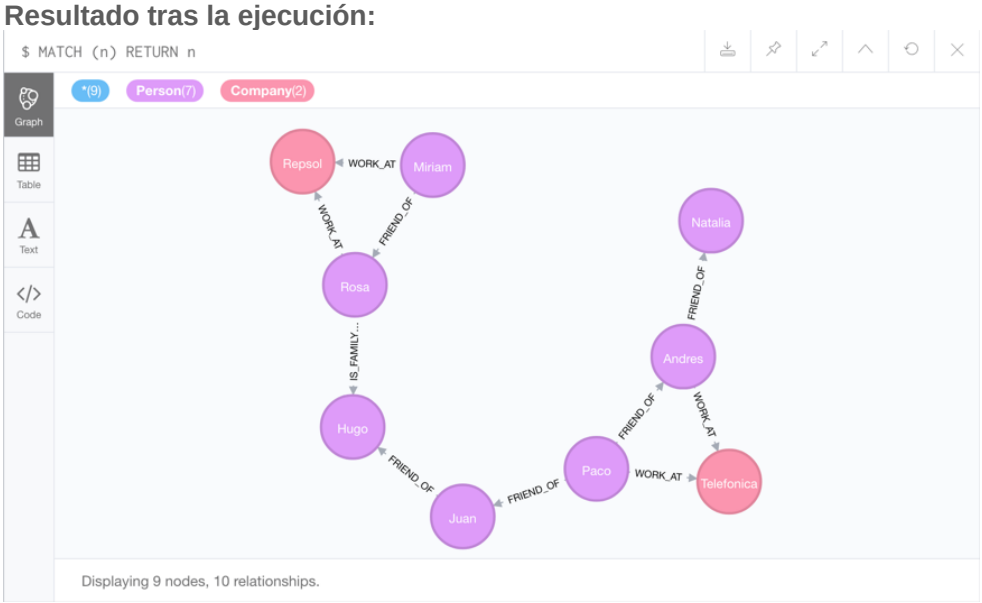
Crear nodos con propiedades

```
CREATE (Paco:Persona {nombre:'Paco', nacimiento:1973})
CREATE (Pedro:Persona {nombre:'Pedro', nacimiento:1980})
CREATE (Luis:Persona {nombre:'Luis', nacimiento:1990})
CREATE (Andres:Persona {nombre:'Andres', nacimiento:1985})
CREATE (Juan:Persona {nombre:'Juan', nacimiento:2000})
CREATE (Ana:Persona {nombre:'Ana', nacimiento:1979})
CREATE (Lola:Persona {nombre:'Lola', nacimiento:1999})
CREATE (Laura:Persona {nombre:'Laura', nacimiento:2005})
CREATE (Santiago:Persona {nombre:'Santiago', nacimiento:2002})
CREATE (Leire:Persona {nombre:'Leire', nacimiento:2006})
CREATE (Alba:Persona {nombre:'Alba', nacimiento:1985})

CREATE
(LOL:Juego {nombre:'LOL', sede:'Los Angeles', categoria:'duelos'}),
(WOW:Juego {nombre:'WOW', sede:'Berlin', categoria:'aventura'})
```

Crear relaciones entre nodos

```
CREATE
    (Paco)-[:AMIGO_DE {role:['Amigo de colegio']}]->(Juan),
    (Paco)-[:AMIGO_DE {role:['Amigo de colegio']}]->(Andres),
    (Juan)-[:AMIGO_DE {role:['Amigo de la infancia']}]->(Luis),
    (Andres)-[:AMIGO_DE  {role:['Amigo de la infancia']}]->(Leire),
    (Alba)-[:AMIGO_DE  {role:['Amigo de Trabajo']}]->(Ana)
CREATE
    (Paco)-[:JUEGA_A {criatura:['Orco']}]->(LOL),
    (Andres)-[:JUEGA_A {criatura:[ 'Enano']}]->(LOL),
    (Ana)-[:JUEGA_A {criatura:['Elfo']}]->(WOW),
    (Alba)-[:JUEGA_A {criatura:['Guerrero']}]->(WOW)
CREATE
    (Laura)-[:ES_FAMILIA_DE {position:['Prima']}]->(Pedro)
```

Resultado tras la ejecución:

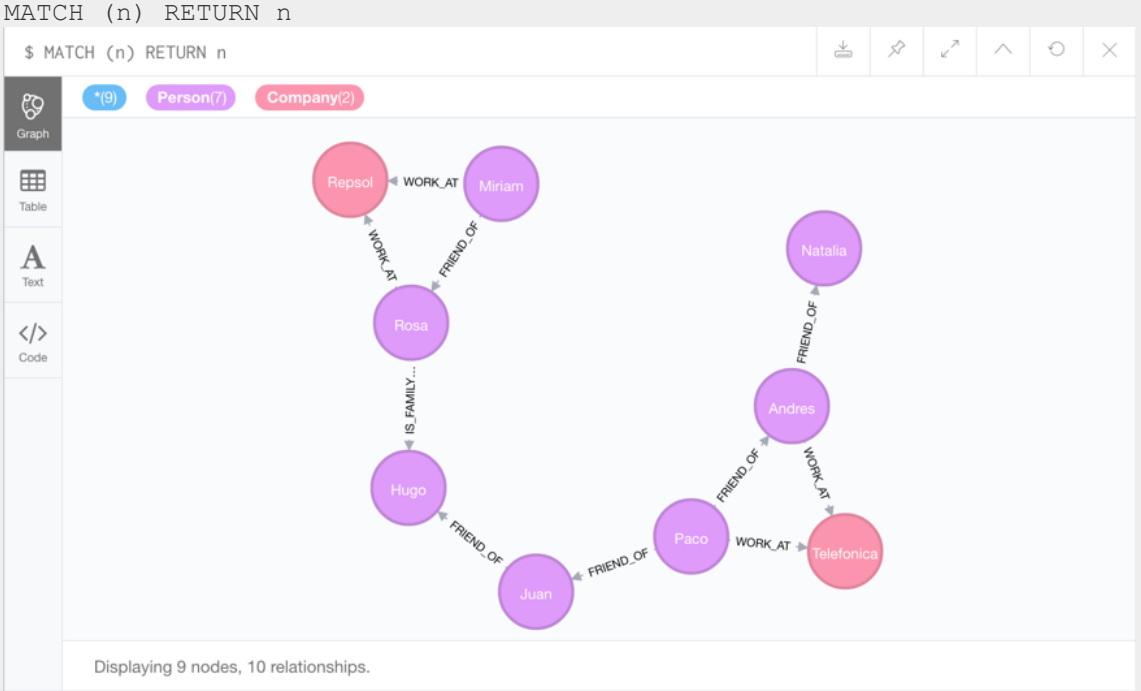


Ejemplo de gráfico en Neo4j

Buscar en un grafo Neo4j (Ejemplo)

Mostar todo el grafo

```
MATCH (n) RETURN n
```



Ejemplo 1: Buscar por propiedad de nodo

Buscar a Paco

```
MATCH (nombre {name: "Paco"}) RETURN nombre
```

Ejemplo 2: Buscar por nodo y relación

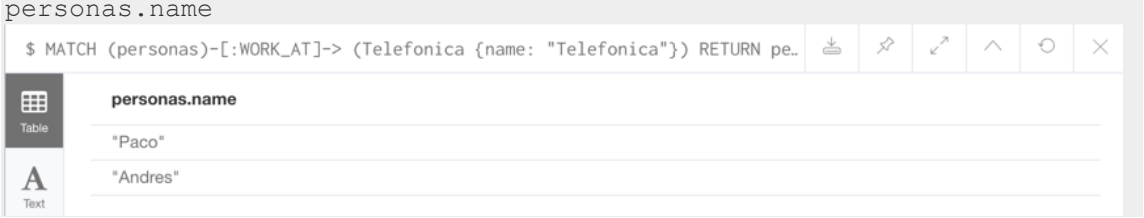
Buscar amigos de Paco

```
MATCH (Paco {name: "Paco"})-[:FRIEND_OF]->(amigos) RETURN amigos
```

Ejemplo 3: Buscar por nodo y relación

Todas las personas que trabajan en Telefonica

```
MATCH (personas)-[:WORK_AT]-> (Telefonica {name: "Telefonica"}) RETURN
personas.name
```



Ejemplo 4: Listar buscando por dos relaciones encadenadas

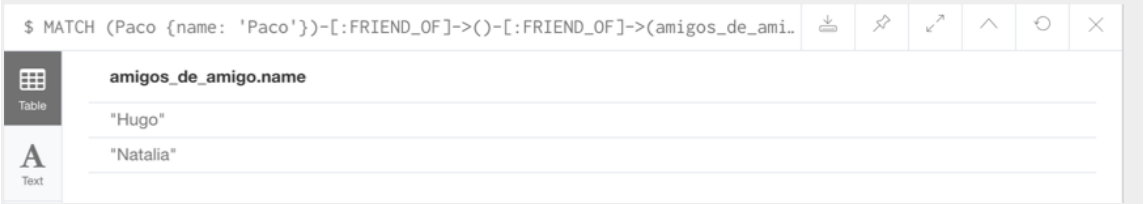
Listado de amigos de los amigos de Paco

```
MATCH (Paco {name: 'Paco'})-[:FRIEND_OF]->()-[:FRIEND_OF]-
>(amigos_de_amigo) RETURN amigos_de_amigo.name
```

Ejemplo 5: Listado buscando por una relación

Listado de personas y que trabajan en compañías

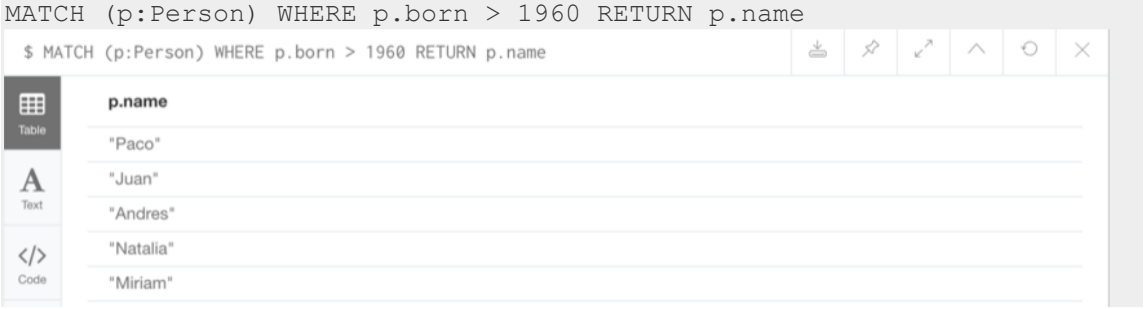
```
MATCH (persona)-[:WORK_AT]->(compania) RETURN persona.name,
compania.name
```



Ejemplo 6: Buscar con restricciones

Listado de personas nacidas después de los 60

```
MATCH (p:Person) WHERE p.born > 1960 RETURN p.name
```



Ejemplo 7: Contar elementos de dos relaciones concatenadas

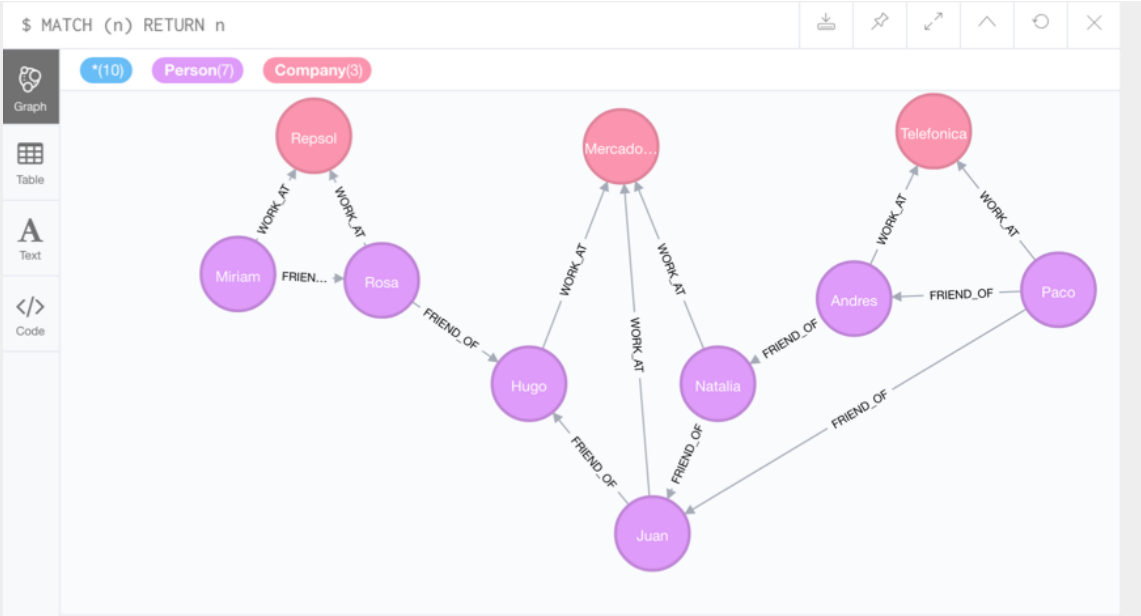
Buscar las personas con más amigos teniendo en cuenta amigos de amigos

```
MATCH (p1:Person)-[:FRIEND_OF]->(p2:Person)-[:FRIEND_OF]->(p3:Person)
RETURN p1.name AS Persona, COUNT(p2) + COUNT(p3) AS Amigos
```

Base de datos utilizada

Se define primeramente la base de datos a utilizar en el ejemplo.

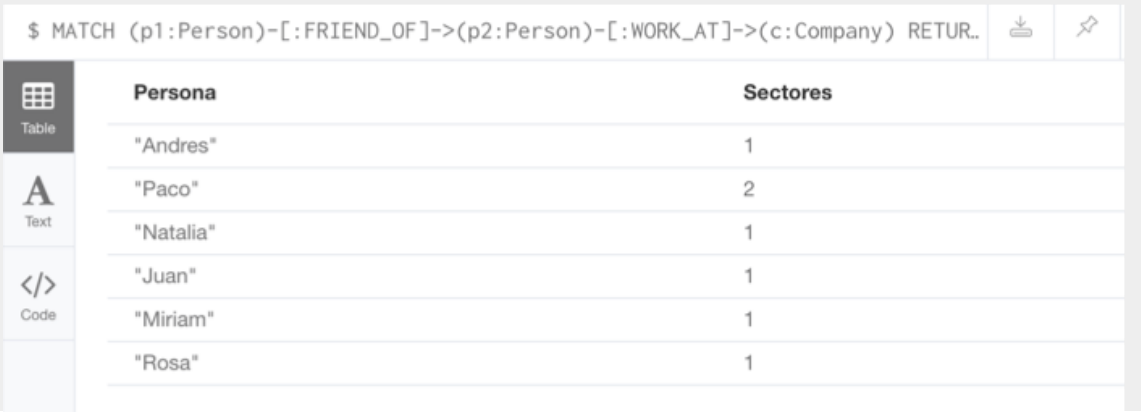
```
CREATE (Paco:Person {name:'Paco', born:1964}),
  (Juan:Person {name:'Juan', born:1967}),
  (Andres:Person {name:'Andres', born:1961}),
  (Hugo:Person {name:'Hugo', born:1960}),
  (Natalia:Person {name:'Natalia', born:1967}),
  (Miriam:Person {name:'Miriam', born:1965}),
  (Rosa:Person {name:'Rosa', born:1952})

CREATE
 (Telefonica:Company {name:'Telefonica', central_office:'Madrid',
sector:'telecomunicaciones'}),
 (Repsol:Company {name:'Repsol', central_office:'Madrid',
sector:'energia'}),
 (Mercadona:Company {name:'Mercadona', central_office:'Valencia',
sector:'alimentacion'})

CREATE
  (Paco)-[:FRIEND_OF {role:['Amigo de Trabajo']}]->(Juan),
  (Paco)-[:FRIEND_OF {role:['Amigo de Trabajo']}]->(Andres),
  (Juan)-[:FRIEND_OF {role:['Amigo de la infancia']}]->(Hugo),
  (Andres)-[:FRIEND_OF {role:['Amigo de la infancia']}]->(Natalia),
  (Miriam)-[:FRIEND_OF {role:['Amigo de Trabajo']}]->(Rosa),
  (Natalia)-[:FRIEND_OF {role:['Amigo de gimnasio']}]->(Juan),
  (Rosa)-[:FRIEND_OF {role:['Amigo de Trabajo']}]->(Hugo)

CREATE
  (Paco)-[:WORK_AT {position:['Director de Marketing']}]-
>(Telefonica),
  (Andres)-[:WORK_AT {position:['Director de Marketing']}]-
>(Telefonica),
  (Miriam)-[:WORK_AT {position:['Director de Marketing']}]-
>(Repsol),
  (Rosa)-[:WORK_AT {position:['Director de Marketing']}]->(Repsol),
  (Hugo)-[:WORK_AT {position:['Director de Marketing']}]-
>(Mercadona),
  (Juan)-[:WORK_AT {position:['Director de Marketing']}]-
>(Mercadona),
  (Natalia)-[:WORK_AT {position:['Director de Marketing']}]-
>(Mercadona)
```



Ejemplo 1: Contar elementos derivados de dos relaciones

Contar para cada persona el número de amigos que tiene trabajando en los diferentes sectores

```
MATCH (p1:Person)-[:FRIEND_OF]->(p2:Person)-[:WORK_AT]->(c:Company)
RETURN p1.name AS Persona , COUNT(c.sector) AS Sectores
```

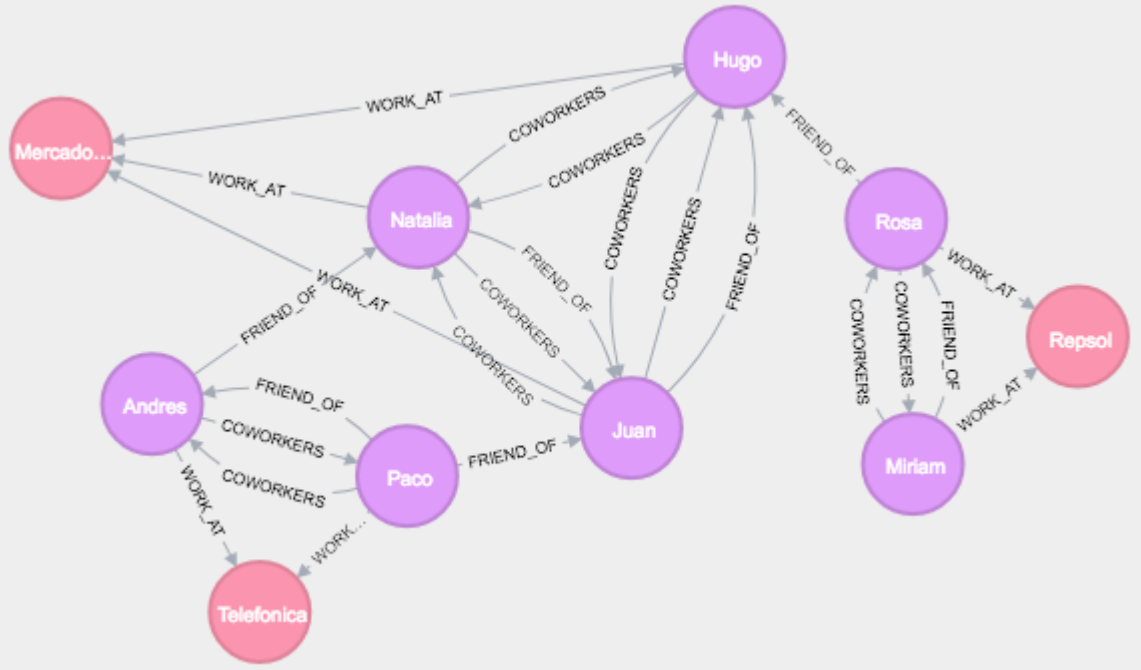


Persona	Sectores
"Andres"	1
"Paco"	2
"Natalia"	1
"Juan"	1
"Miriam"	1
"Rosa"	1

Ejemplo 2: Generar relaciones a partir de consultas

Crear la relación de compañeros de trabajo «coworkers» para las personas que trabajen en la misma compañía.

```
MATCH
  (p1:Person)-[r1:WORK_AT]->(c:Company),
  (p2:Person)-[r2:WORK_AT]->(c:Company)
CREATE (p1)-[r3:COWORKERS]->(p2)
RETURN p1,p2,c,r1,r2,r3
```



Ejemplo 3: Agrupaciones

Para cada persona agrupar por sectores en que trabajan sus amigos

```
MATCH (p1:Person)-[:FRIEND_OF]->(p2:Person)-[:WORK_AT]->(c:Company)
RETURN p1.name, COLLECT(c.sector)
```

```
$ MATCH (p1:Person)-[:FRIEND_OF]->(p2:Person)-[:WORK_AT]->(c:Company) RETUR…
```

Table | Text | Code

p1.name	p2.name	c.sector
"Paco"	"Andres"	"telecomunicaciones"
"Miriam"	"Rosa"	"energia"
"Andres"	"Natalia"	"alimentacion"
"Natalia"	"Juan"	"alimentacion"
"Paco"	"Juan"	"alimentacion"
"Rosa"	"Hugo"	"alimentacion"
"Juan"	"Hugo"	"alimentacion"

Modificar elementos en un grafo en Neo4J

Modificar propiedades a un nodo

```
MERGE (p:Person {name: 'Paco'}) SET p.age = 34, p.coat = 'Yellow'
RETURN p
```

Modificar propiedades a una relación

```
MERGE (Paco)-[r:FRIEND_OF]->(Juan) SET r.ages = 34 RETURN r
```

Borrar elementos del grafo Neo4j

Borrar nodos

MATCH (p:Person {name: «Paco»}), (c:Company {name: «Telefonica»}) DELETE p, c

Nota: Para poder borrar los nodos se tienen que borrar las relaciones entre ellos

Borrar relaciones entre nodos

```
MATCH (Miriam)-[:FRIEND_OF]->(Rosa) DELETE r
```

Borrar todo el grafo

```
MATCH (n) OPTIONAL MATCH (n)-[r]-() DELETE n,r
```