

Nombre: Walter Bau

Crear nodos con propiedades

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

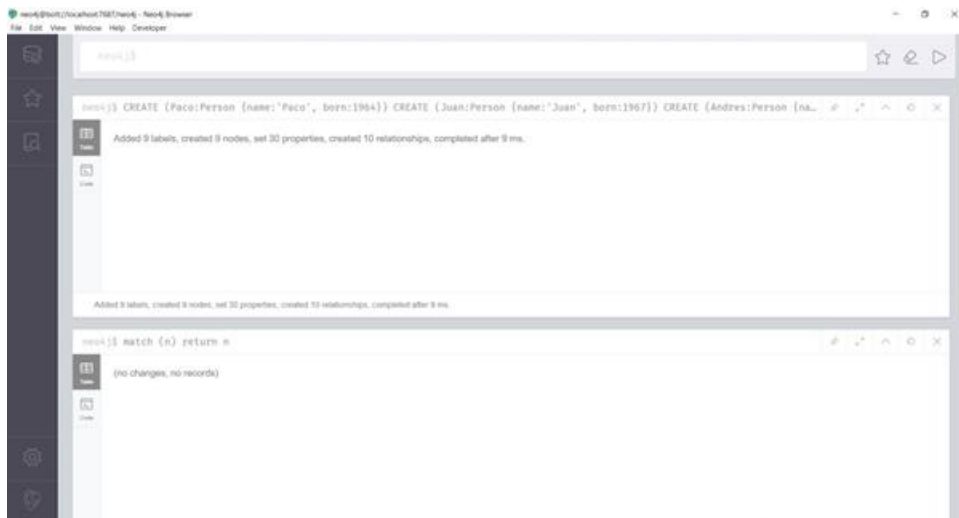
CREATE (Telefonica:Company {name:'Telefonica', central_office:'Madrid',
sector:'telecomunicaciones'}),
      (Repsol:Company {name:'Repsol', central_office:'Madrid',
sector:'energia'})
```

Crear relaciones entre nodos

```
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)

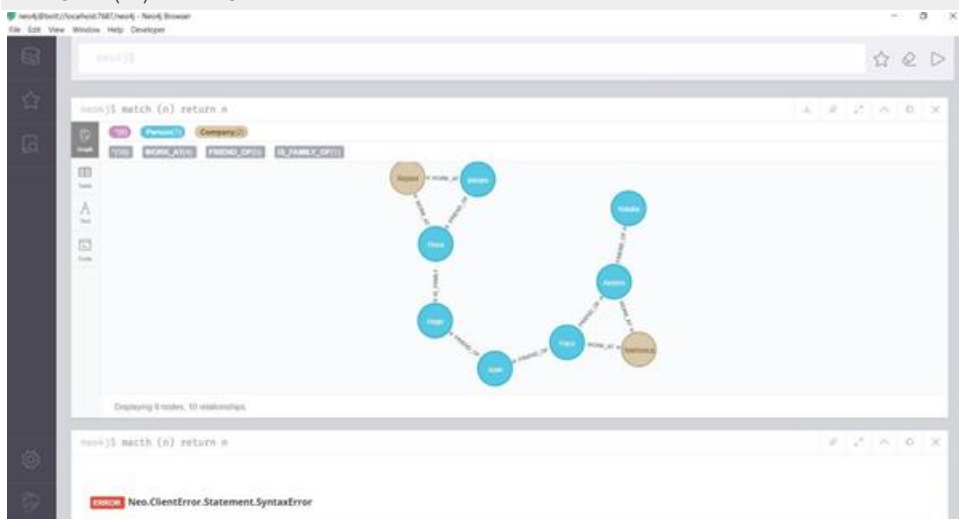
CREATE
  (Paco)-[:WORK_AT {position:['Director de Marketing']}]>
  (Telefonica),
  (Andres)-[:WORK_AT {position:['Director de Marketing']}]>
  (Telefonica),
  (Miriam)-[:WORK_AT {position:['Director de Marketing']}]>(Repsol),
  (Rosa)-[:WORK_AT {position:['Director de Marketing']}]>(Repsol)

CREATE
  (Rosa)-[:IS_FAMILY_OF {position:['Prima']}]>(Hugo)
```



Mostrar todo el grafo

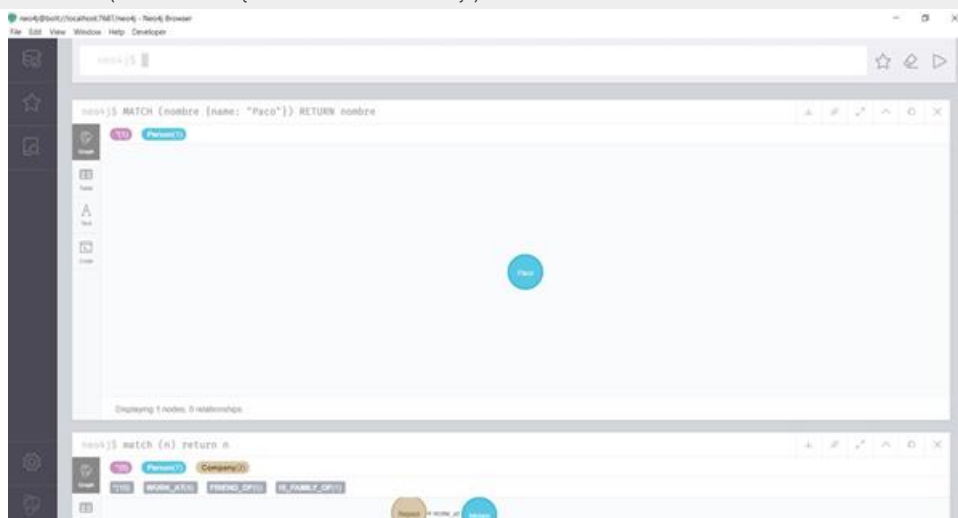
MATCH (n) RETURN n



Ejemplo 1: Buscar por propiedad de nodo

Buscar a Paco

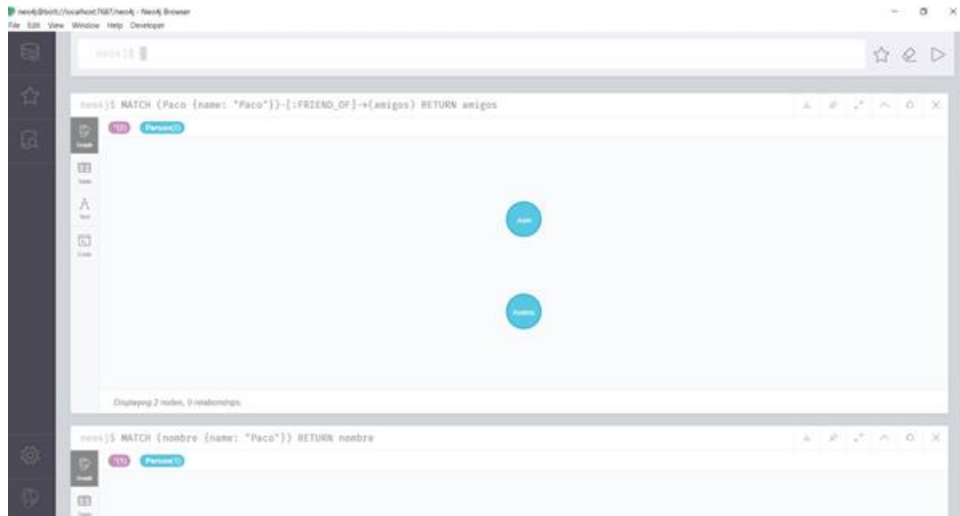
MATCH (nombre {name: "Paco"}) RETURN nombre



Ejemplo 2: Buscar por nodo y relación

Buscar amigos de Paco

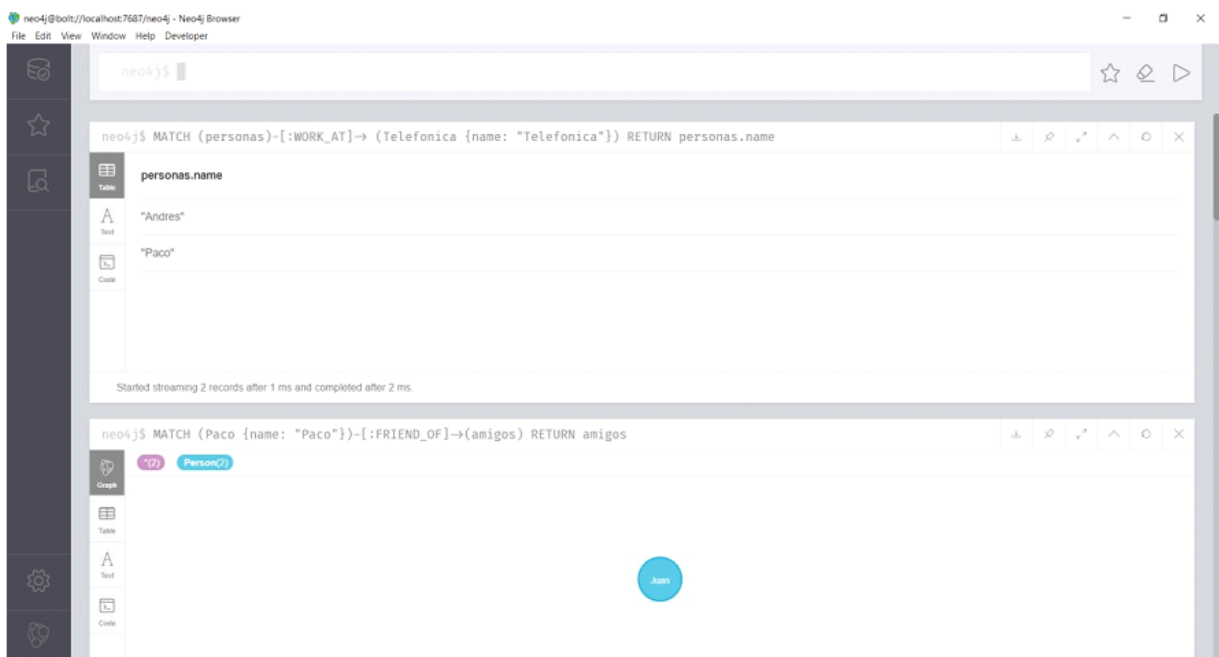
```
MATCH (Paco {name: "Paco"})-[:FRIEND_OF]->(amigos) RETURN amigos
```



Ejemplo 3: Buscar por nodo y relación

Todas las personas que trabajan en Telefonica

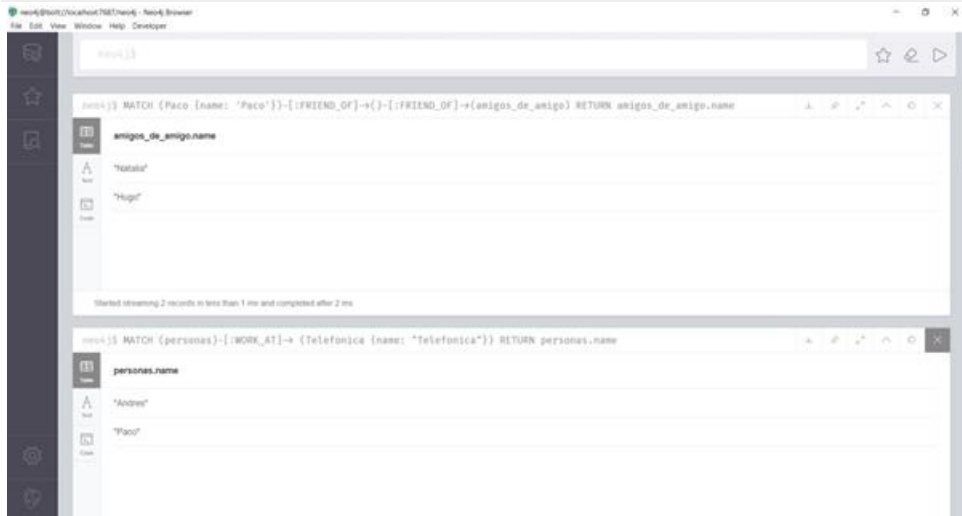
```
MATCH (personas)-[:WORK_AT]-> (Telefonica {name: "Telefonica"}) RETURN personas.name
```



Ejemplo 4: Listar buscando por dos relaciones encadenadas

Listado de amigos de los amigos de Paco

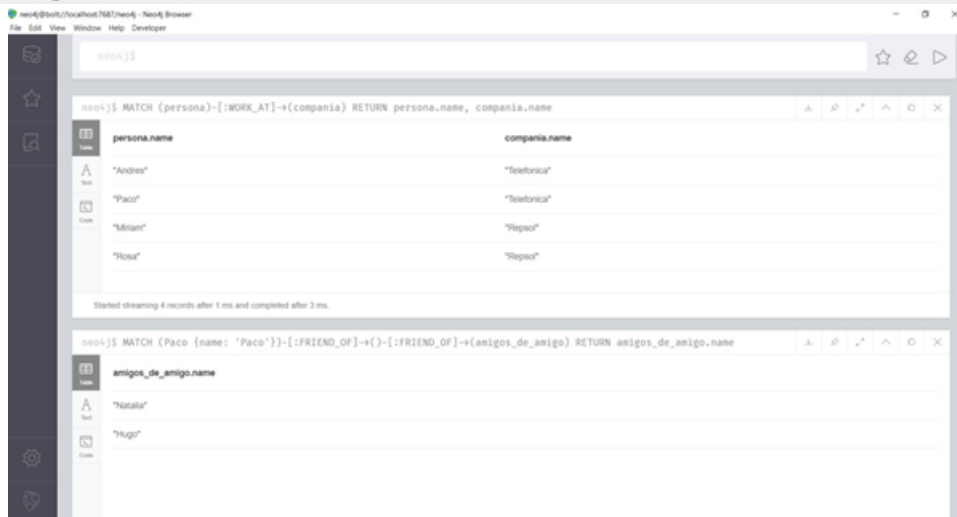
```
MATCH (Paco {name: 'Paco'})-[:FRIEND_OF]->()-[:FRIEND_OF]->(amigos_de_amigo) RETURN amigos_de_amigo.name
```



Ejemplo 5: Listado buscando por una relación

Listado de personas y que trabajan en compañías

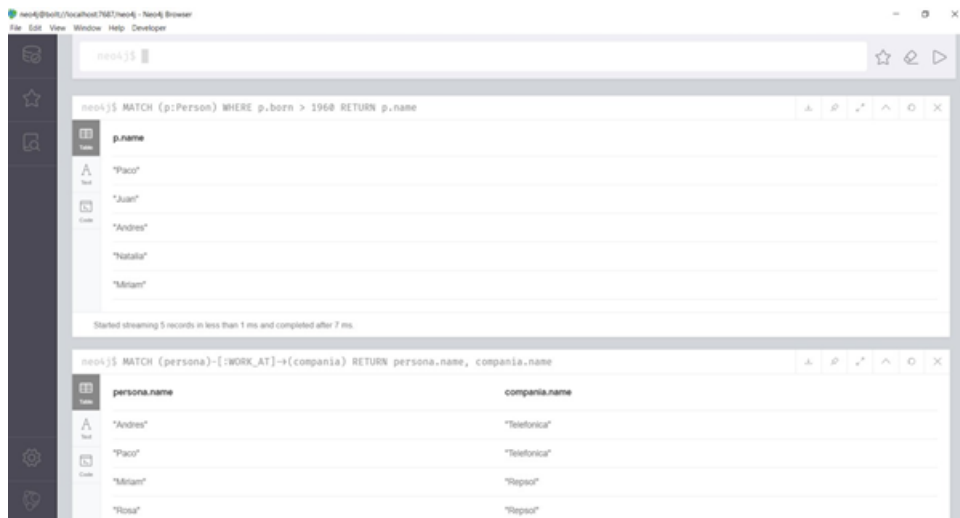
```
MATCH (persona)-[:WORK_AT]->(compania) RETURN persona.name, compania.name
```



Ejemplo 6: Buscar con restricciones

Listado de personas nacidas después de los 60

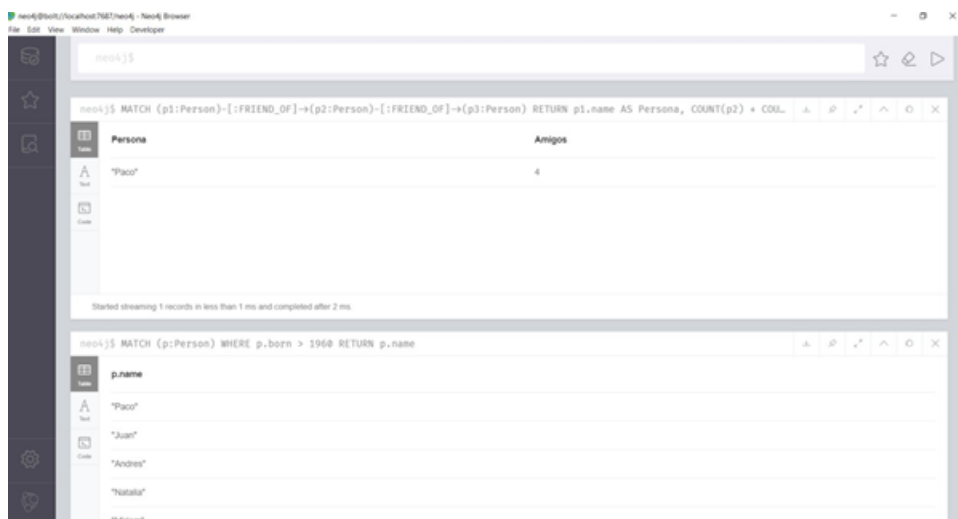
```
MATCH (p:Person) WHERE p.born > 1960 RETURN p.name
```



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})

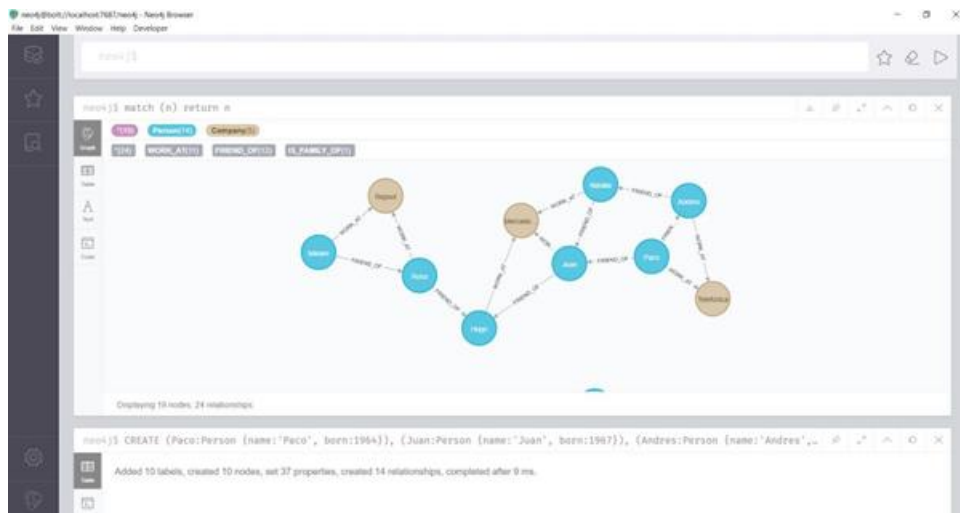
CREATE
  (Telefonica:Company {name:'Telefonica', central_office:'Madrid',
sector:'telecomunicaciones'}),
  (Repsol:Company {name:'Repsol', central_office:'Madrid',
sector:'energia'}),
  (Mercadona:Company {name:'Mercadona', central_office:'Valencia',
sector:'alimentacion'})

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
```

Neo4j Browser interface showing the results of the query. The query bar contains: `neo4j MATCH (p1:Person)-[:FRIEND_OF]->(p2:Person)-[:WORK_AT]->(c:Company) RETURN p1.name AS Persona , COUNT(c.sector) AS Sectores`. The results are displayed in a table:

Persona	Sectores
"Paco"	3
"Miriam"	2
"Andres"	1
"Natalia"	1
"Juan"	1
"Blanca"	1

Below the table, the status bar indicates: "Started streaming 6 records in less than 1 ms and completed after 2 ms".

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

```

The Neo4j Browser interface displays a graph visualization of the database. The graph shows nodes for people (blue circles) and companies (orange circles). Relationships include WORK_AT, FRIEND_OF, and COWORKERS. The table view below the graph shows the results of a query.

Query 1:

```
neo4j$ MATCH (p1:Person)-[r1:WORK_AT]->(c:Company), (p2:Person)-[r2:WORK_AT]->(c:Company) CREATE (p1)-[r3:COWORKERS]->(p2)
```

Query 2:

```
neo4j$ MATCH (p1:Person)-[:FRIEND_OF]->(p2:Person)-[:WORK_AT]->(c:Company) RETURN p1.name AS Persona , COUNT(c.sector) AS Sectores
```

Persona	Sectores
"Paco"	3

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)

```

The Neo4j Browser interface displays the results of a query in a table view. The table has two columns: p1.name and COLLECT(c.sector). The results show the names of people and the list of sectors their friends work in.

Query:

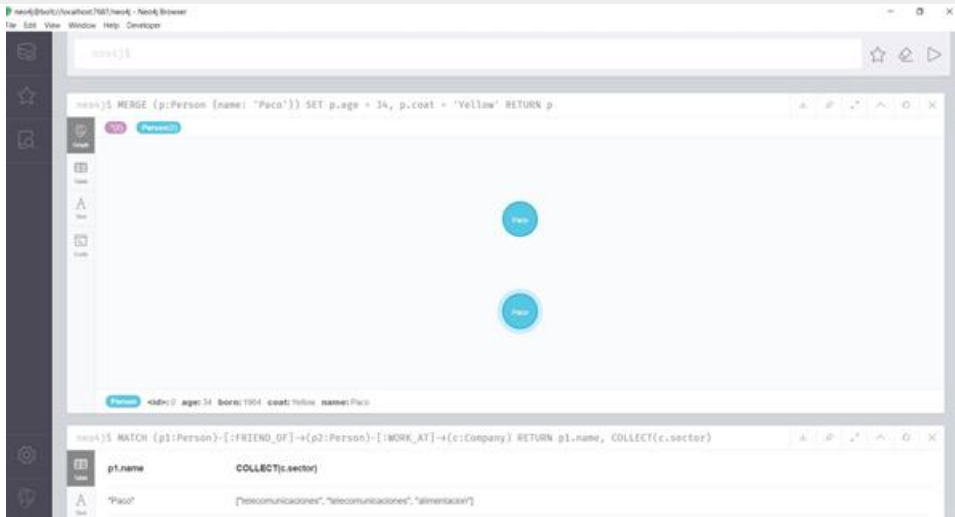
```
neo4j$ MATCH (p1:Person)-[:FRIEND_OF]->(p2:Person)-[:WORK_AT]->(c:Company) RETURN p1.name, COLLECT(c.sector)
```

p1.name	COLLECT(c.sector)
"Paco"	["telecomunicaciones", "telecomunicaciones", "alimentacion"]
"Miriam"	["energia", "energia"]
"Andres"	["alimentacion"]
"Natalia"	["alimentacion"]
"Juan"	["alimentacion"]
"Rosa"	["alimentacion"]

Started streaming 6 records in less than 1 ms and completed after 3 ms.

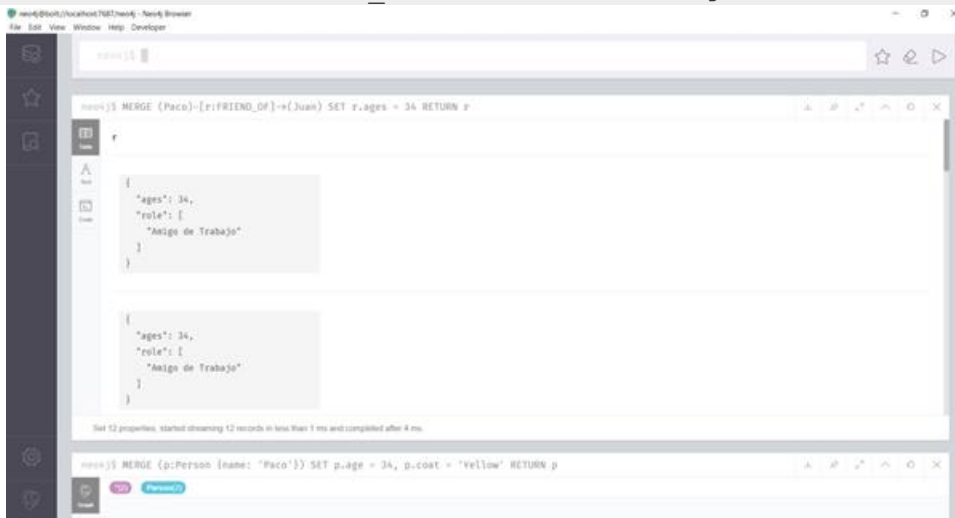
Modificar propiedades a un nodo

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



Modificar propiedades a una relación

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



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
```

neo4j@localhost:~/neo4j - Neo4j Browser

File Edit View Window Help Developer


neo4j\$


neo4j\$ match (n) return n


no changes, no records


Completed after 2 ms.

\$:play start

**neo4j**

Learn about Neo4j
A graph epiphany awaits you.
 What is a graph database?
How can I query a graph?
What do people do with Neo4j?

Jump into code
Use Cypher, the graph query language.
 Code walk-throughs
RDBMS to Graph

System information
Key system health and status metrics.
 Store sizes
ID allocation
Page cache
Transaction count
Cluster status

[Watch a video](#)

[Watch a video](#)

[Watch a video](#)