

Parte 2: Neo4j Introduction 2022

Análisis, Diseño y Procesamiento de Datos Aplicados a las Ciencias y a
las Tecnologías(ADP)

Máster Universitario en Inteligencia Computacional e Internet de las Cosas

Universidad de Córdoba, EPSC
2022/2023



Autor:

Antonio Gómez Giménez (i72gogia@uco.es)

Índice:

| | |
|--|----|
| 1. Query to display the schema of your database. | 3 |
| 2. Query to retrieve all nodes from the database. | 4 |
| 3. All Person nodes. | 5 |
| 4. All Movies nodes. | 6 |
| 5. Retrieve all Movie nodes that have a released property value of 2003. | 7 |
| 6. Retrieve all Movies released in 2006, returning their titles. | 8 |
| 7. Retrieve all Movie nodes from the database and return the title, released, and tagline values. | 9 |
| 8. Modify the query you just ran so that the headings for the columns of the table returned are more descriptive. | 10 |
| 9. Retrieve all people who wrote the movie Speed Racer. | 11 |
| 10. Explica brevemente lo que hace la siguiente instrucción: | 12 |
| 11. Explica brevemente lo que hace la siguiente instrucción: | 13 |
| 12. Explica brevemente lo que hace la siguiente instrucción: | 14 |
| 13. Devuelve las personas que hayan actuado y dirigido a la vez una película. | 15 |
| 14. Use MERGE to update (ON MATCH) the existing Production node for Forrest Gump to add the company property with a value of Paramount Pictures. | 16 |
| 15. Añadir la relación [:DIRECTED] entre Robert Zemeckis y el film "Forrest Gump": | 17 |
| 16. Borrar el nodo de la producción "Forrest Gump": | 18 |
| 17. Finding who directed Cloud Atlas movie : | 19 |
| 18. Finding all people who have co-acted with Tom Hanks in any movie: | 20 |
| 19. Finding all people related to the movie Cloud Atlas in any way: | 21 |
| 20. Finding Movies and Actors that are 3 hops away from Kevin Bacon : | 22 |
| PYTHON PY2NEO DRIVER EXERCISES | 23 |
| 1. Analiza los siguientes scripts python: | 23 |
| 2. Realiza los siguientes scripts en Python: | 24 |

1. Query to display the schema of your database.

En este ejercicio se pide cargar la base de datos de Movie Graph. Para ello primero es necesario eliminar los nodos y relaciones anteriormente creados con el siguiente comando (en el caso de que existan):

```
match(n) detach delete(n)
```

Posteriormente usamos el comando que activa el tutorial donde se encuentra la base de datos, siendo el comando:

```
:play movie graph
```

En la segunda pestaña que nos proporciona, ejecutamos el comando que inicia la base de datos, siendo el siguiente:

The screenshot shows a dark-themed interface for a Cypher query editor. At the top, a command prompt shows the command `$:play movie graph` with a blue play button and a star icon to its right. Below this, the interface is split into two main sections. On the left, under the heading "The Movie Graph", there is a "Create" section. It contains a text block explaining that a code block on the right contains a Cypher query statement composed of multiple CREATE clauses, which will create the movie graph. Below this text are instructions: "Click on the code block", "Notice it gets copied to the editor above", "Click the editor's play button to execute", and "Wait for the query to finish". A warning message at the bottom of this section states: "WARNING: This adds data to the current database". On the right side of the interface, a large code block contains the following Cypher query:

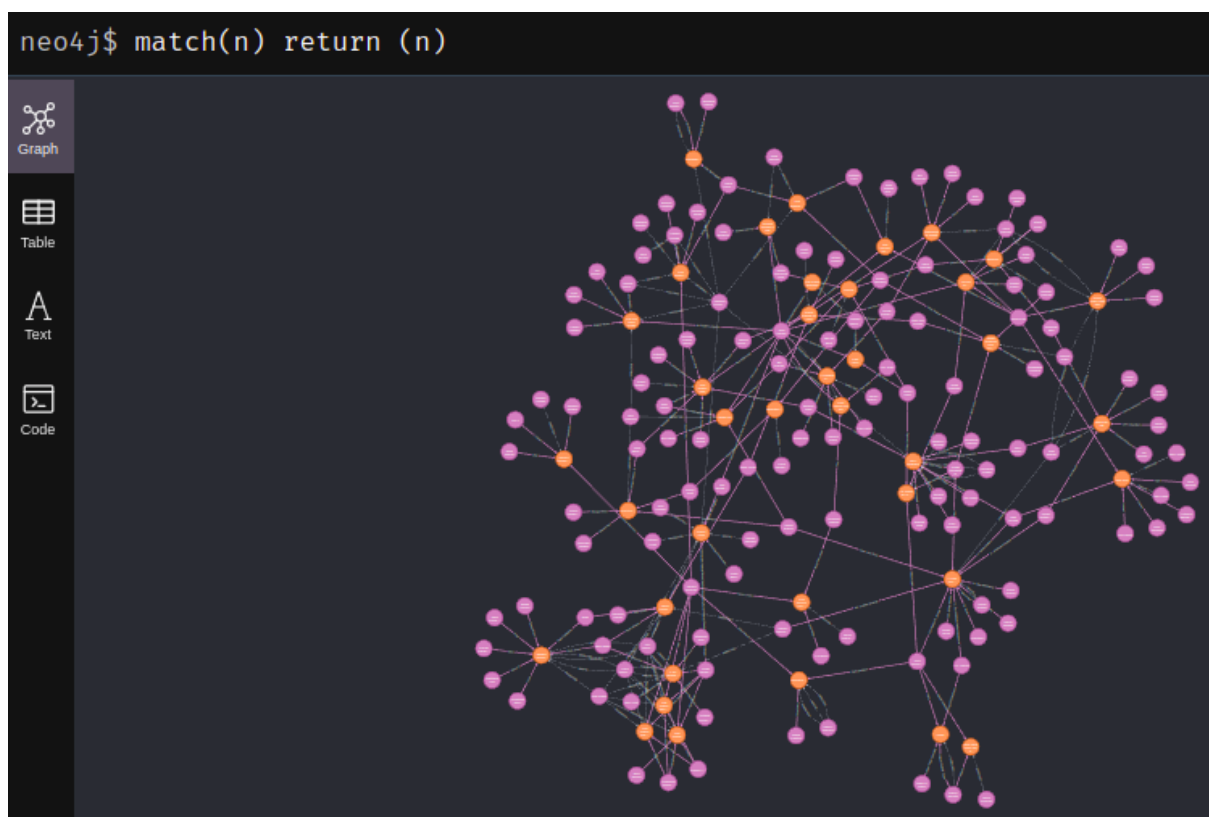
```
CREATE (TheMatrix:Movie {title:'The Matrix', released:1999, tagline:'Welcome to the Real World'})
CREATE (Keanu:Person {name:'Keanu Reeves', born:1964})
CREATE (Carrie:Person {name:'Carrie-Anne Moss', born:1967})
CREATE (Laurence:Person {name:'Laurence Fishburne', born:1961})
CREATE (Hugo:Person {name:'Hugo Weaving', born:1960})
CREATE (LillyW:Person {name:'Lilly Wachowski', born:1967})
CREATE (LanaW:Person {name:'Lana Wachowski', born:1965})
CREATE (JoelS:Person {name:'Joel Silver', born:1952})
CREATE
(Keanu)-[:ACTED IN {roles:['Neo']}]→(TheMatrix),
```

2. Query to retrieve all nodes from the database.

En este apartado se nos comenta mostrar todos los nodos de la base de datos movie, para ello se utiliza el siguiente comando:

```
match(n) return (n)
```

El resultado obtenido se puede apreciar en la siguiente captura:



3. All Person nodes.

En este apartado se nos comenta mostrar todos los nodos de la base de datos movie que son personas, para ello se utiliza el siguiente comando:

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

El resultado obtenido se puede apreciar en la siguiente captura:

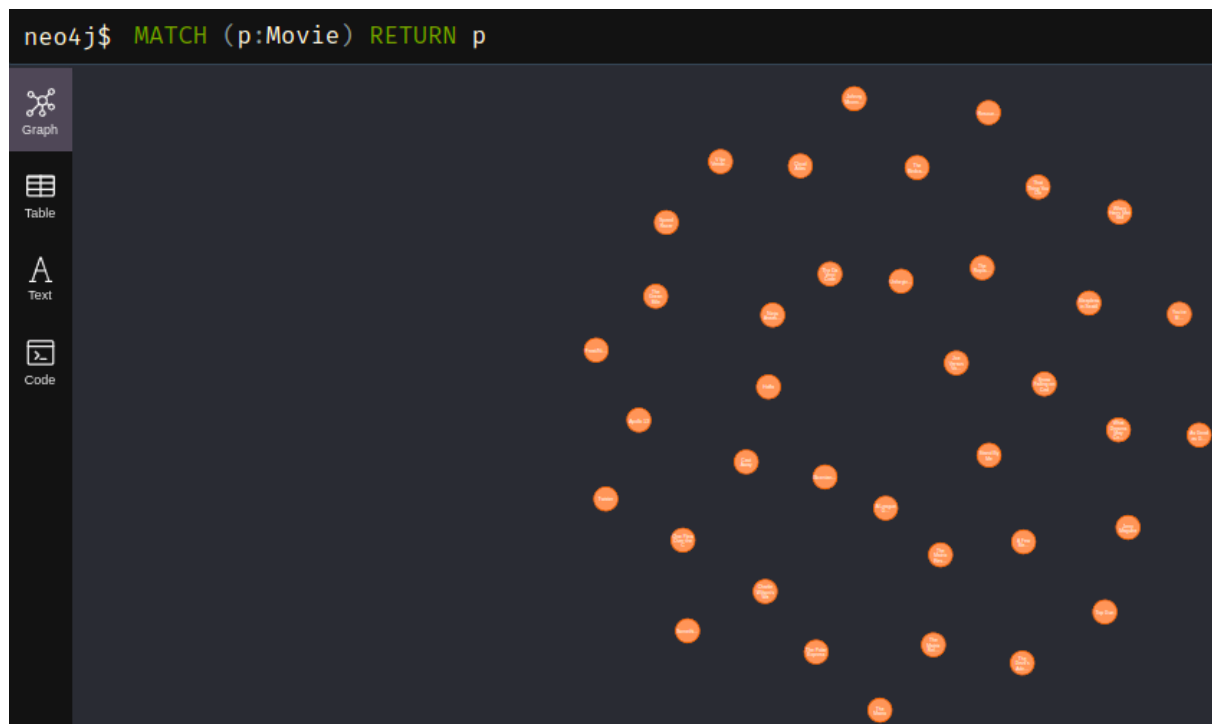


4. All Movies nodes.

En este apartado se nos comenta mostrar todos los nodos de la base de datos movie que son películas, para ello se utiliza el siguiente comando:

MATCH (p:Movie) RETURN p

El resultado obtenido se puede apreciar en la siguiente captura:

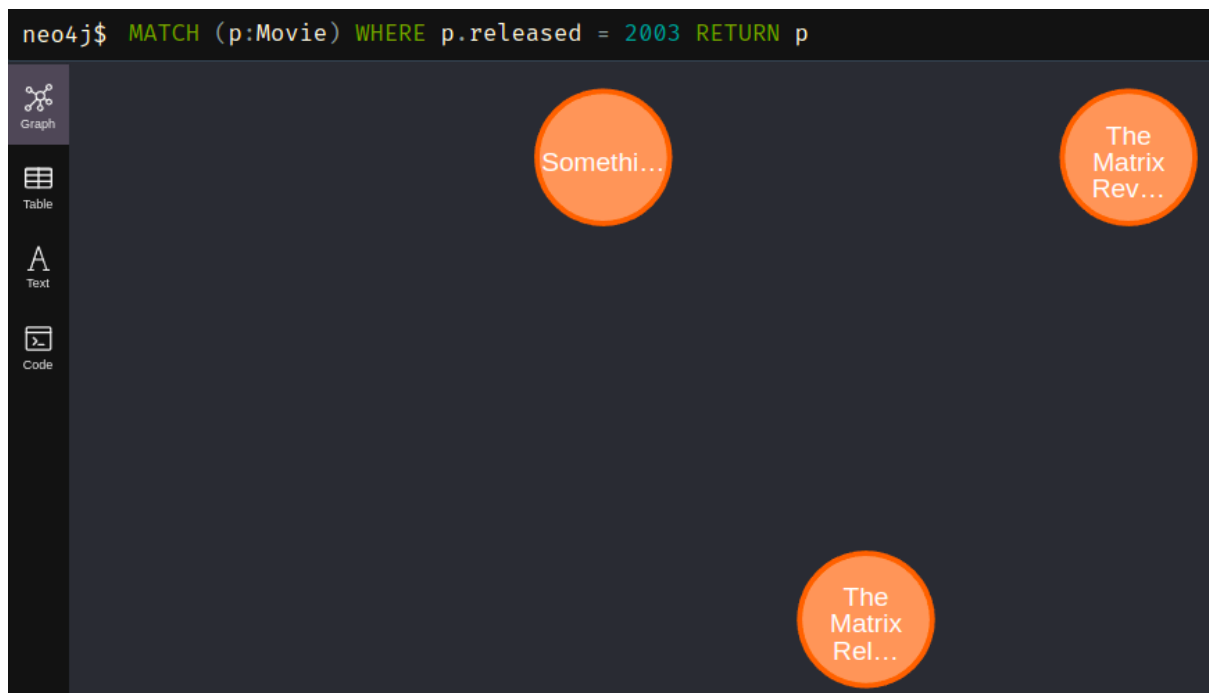


5. Retrieve all Movie nodes that have a released property value of 2003.

En este apartado se pide mostrar todos los nodos de la base de datos movie que son películas que cumplen la propiedad de ser lanzadas en el 2003, para ello se utiliza el siguiente comando:

```
MATCH (p:Movie) WHERE p.released = 2003 RETURN p
```

El resultado obtenido se puede apreciar en la siguiente captura:

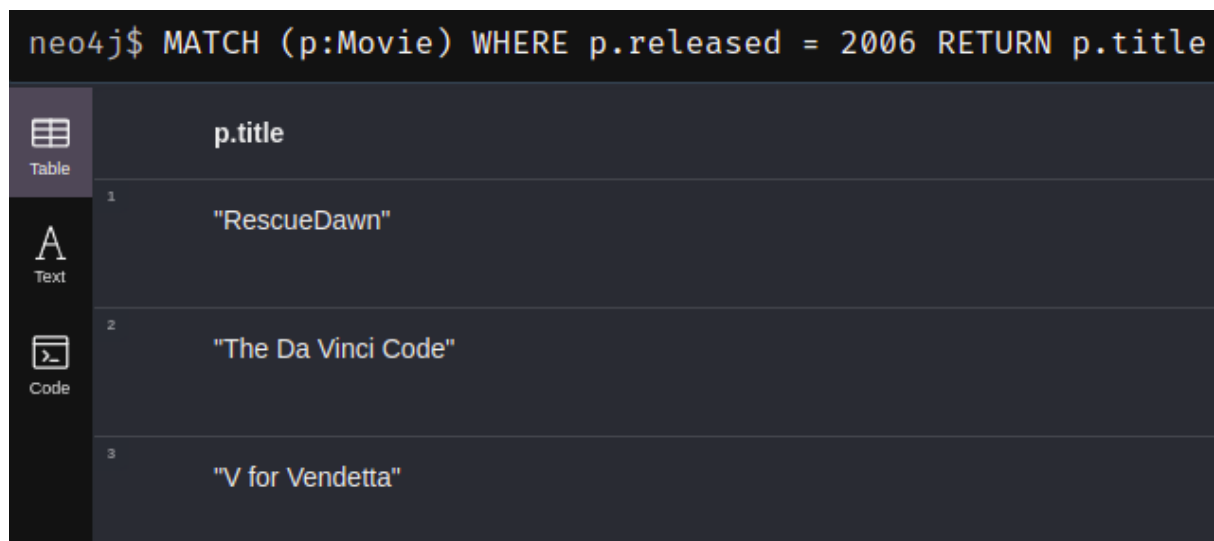


6. Retrieve all Movies released in 2006, returning their titles.

En este apartado se pide mostrar todos los nodos de la base de datos movie que son películas que cumplen la propiedad de ser lanzadas en el 2006 devolviendo únicamente sus títulos, para ello se utiliza el siguiente comando:

```
MATCH (p:Movie) WHERE p.released = 2006 RETURN p.title
```

El resultado obtenido se puede apreciar en la siguiente captura:



```
neo4j$ MATCH (p:Movie) WHERE p.released = 2006 RETURN p.title
```

| | p.title |
|---|---------------------|
| 1 | "RescueDawn" |
| 2 | "The Da Vinci Code" |
| 3 | "V for Vendetta" |

7. Retrieve all Movie nodes from the database and return the title, released, and tagline values.

En este apartado se pide mostrar todos los nodos de la base de datos movie que son películas devolviendo sus títulos, fecha de lanzamiento y su tagline, para ello se utiliza el siguiente comando:

```
MATCH (p:Movie) RETURN p.title, p.released, p.tagline
```

El resultado obtenido se puede apreciar en la siguiente captura:

```
neo4j$ MATCH (p:Movie) RETURN p.title, p.released, p.tagline
```

| | p.title | p.released | p.tagline |
|---|--------------------------|------------|--|
| 1 | "The Matrix" | 1999 | "Welcome to the Real World" |
| 2 | "The Matrix Reloaded" | 2003 | "Free your mind" |
| 3 | "The Matrix Revolutions" | 2003 | "Everything that has a beginning has an end" |
| 4 | "The Devil's Advocate" | 1997 | "Evil has its winning ways" |
| 5 | "A Few Good Men" | 1992 | "In the heart of the nation's capital, in a courthouse of the U.S. government" |
| 6 | "Top Gun" | 1986 | "I feel the need, the need for speed." |

8. Modify the query you just ran so that the headings for the columns of the table returned are more descriptive.

En este apartado se pide para la query anterior, añadirle un título más descriptivo cada columna

MATCH (p:Movie) RETURN p.title as Titulo, p.released as FechaSalida, p.tagline as FrasePelicula

El resultado obtenido se puede apreciar en la siguiente captura:

```
neo4j$ MATCH (p:Movie) RETURN p.title as Titulo, p.released as FechaSalida, p.tagline as FrasePelicula
```

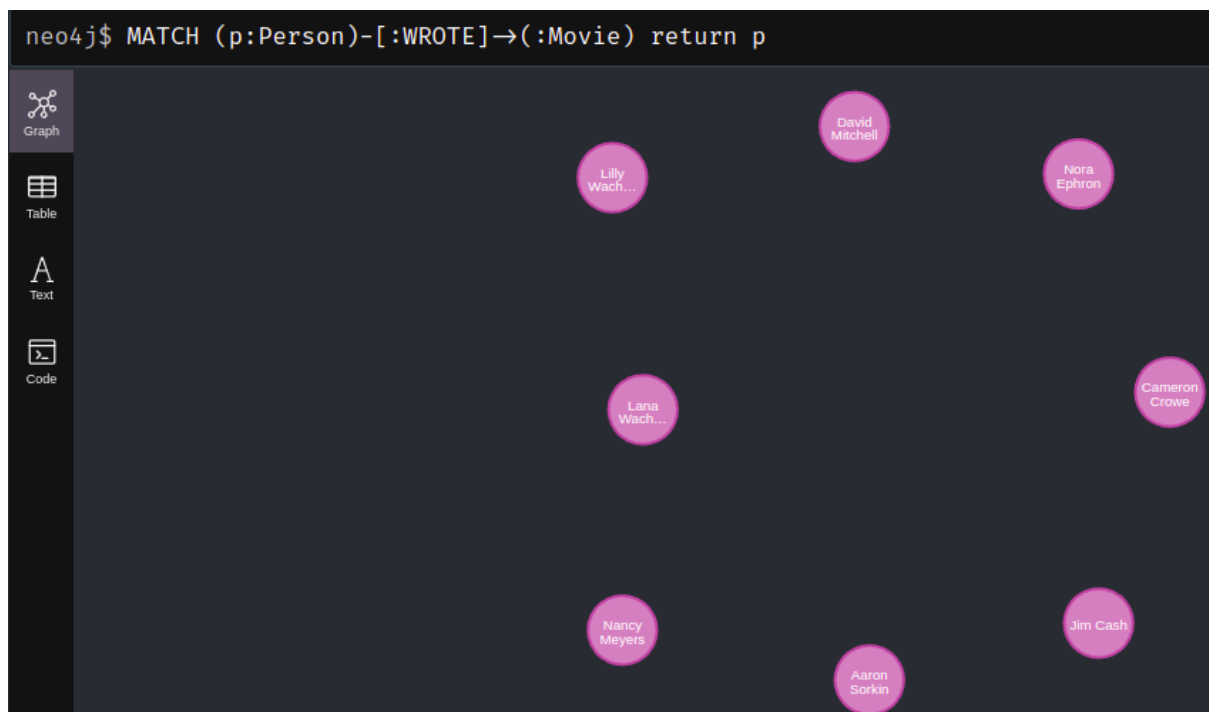
| | Titulo | FechaSalida | FrasePelicula |
|---|--------------------------|-------------|---|
| 1 | "The Matrix" | 1999 | "Welcome to the Real World" |
| 2 | "The Matrix Reloaded" | 2003 | "Free your mind" |
| 3 | "The Matrix Revolutions" | 2003 | "Everything that has a beginning has an end" |
| 4 | "The Devil's Advocate" | 1997 | "Evil has its winning ways" |
| 5 | "A Few Good Men" | 1992 | "In the heart of the nation's capital, in a courthouse of the U.S. government, one man will stop at |
| 6 | "Top Gun" | 1986 | "I feel the need, the need for speed." |

9. Retrieve all people who wrote the movie Speed Racer.

En este apartado se pide mostrar todos los nodos de la base de datos movie que son personas que han escrito la película Speed Racer, para ello se utiliza el siguiente comando:

```
MATCH (p:Person)-[:WROTE]->(m:Movie) return p
```

El resultado obtenido se puede apreciar en la siguiente captura:



10. Explica brevemente lo que hace la siguiente instrucción:

MATCH (m:Movie)-[rel]-(:Person {name: 'Tom Hanks'}) RETURN m.title, type(rel)

La instrucción anterior devuelve el título de la película y relación existente con el nodo persona cuyo nombre es Tom Hanks.

El resultado obtenido se puede apreciar en la siguiente captura:

```
neo4j$ MATCH (m:Movie)-[rel]-(:Person {name: 'Tom Hanks'}) RETURN m.title, type(rel)
```

| | m.title | type(rel) |
|---|--------------------------|------------|
| 1 | "You've Got Mail" | "ACTED_IN" |
| 2 | "The Polar Express" | "ACTED_IN" |
| 3 | "A League of Their Own" | "ACTED_IN" |
| 4 | "The Da Vinci Code" | "ACTED_IN" |
| 5 | "Apollo 13" | "ACTED_IN" |
| 6 | "Joe Versus the Volcano" | "ACTED_IN" |

11. Explica brevemente lo que hace la siguiente instrucción:

MATCH (m:Movie)-[rel:ACTED_IN]-(:Person {name: 'Tom Hanks'}) RETURN m.title, rel.roles

La instrucción anterior devuelve el título de la película y los nombres de los personajes de la relación entre películas donde actúa Tom Hanks.

El título de la película y el nombre del personaje que interpreta Tom Hanks(resumen de lo anterior).

El resultado obtenido se puede apreciar en la siguiente captura:

```
neo4j$ MATCH (m:Movie)-[rel:ACTED_IN]-(:Person {name: 'Tom Hanks'}) RETURN m.title, rel.roles
```

| | m.title | rel.roles |
|---|--------------------------|---|
| 1 | "You've Got Mail" | ["Joe Fox"] |
| 2 | "The Polar Express" | ["Hero Boy", "Father", "Conductor", "Hobo", "Scrooge", "Santa Claus"] |
| 3 | "A League of Their Own" | ["Jimmy Dugan"] |
| 4 | "The Da Vinci Code" | ["Dr. Robert Langdon"] |
| 5 | "Apollo 13" | ["Jim Lovell"] |
| 6 | "Joe Versus the Volcano" | ["Joe Banks"] |

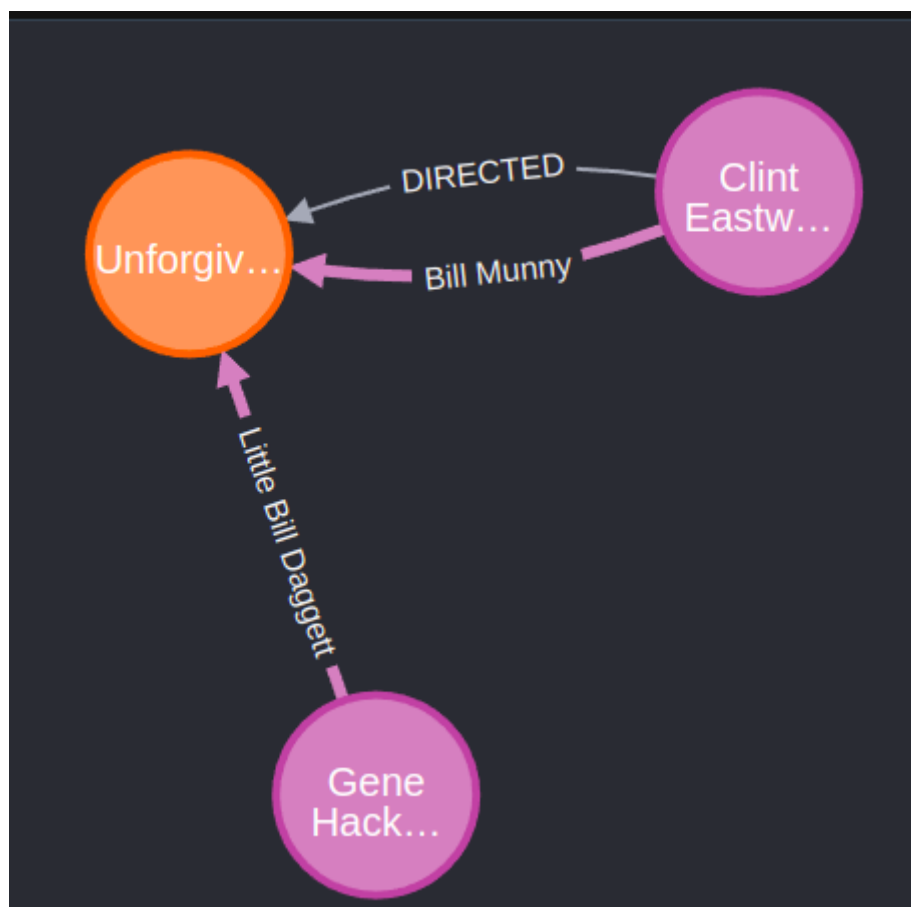
12. Explica brevemente lo que hace la siguiente instrucción:

```
MATCH (gene:Person)-[:ACTED_IN]->(m:Movie)←[:ACTED_IN]-(other:Person)
WHERE gene.name= 'Gene Hackman'
AND exists( (other)-[:DIRECTED]->(m) )
RETURN gene, other, m
```

La instrucción anterior realiza un match entre una persona que actúa en una película, y en dicha película actúa otra persona. Donde el nombre de la primera persona es Gene Hackman y existe una persona que dirige dicha película. Finalmente se devuelve la película, el nodo cuyo nombre es Gene Hackman y la persona que dirige la película.

En resumen devuelve el nodo persona del que es director de la película, el nodo de la película donde actúa Gene Hackman y el nodo personal de Gene Hackman.

El resultado obtenido se puede apreciar en la siguiente captura:

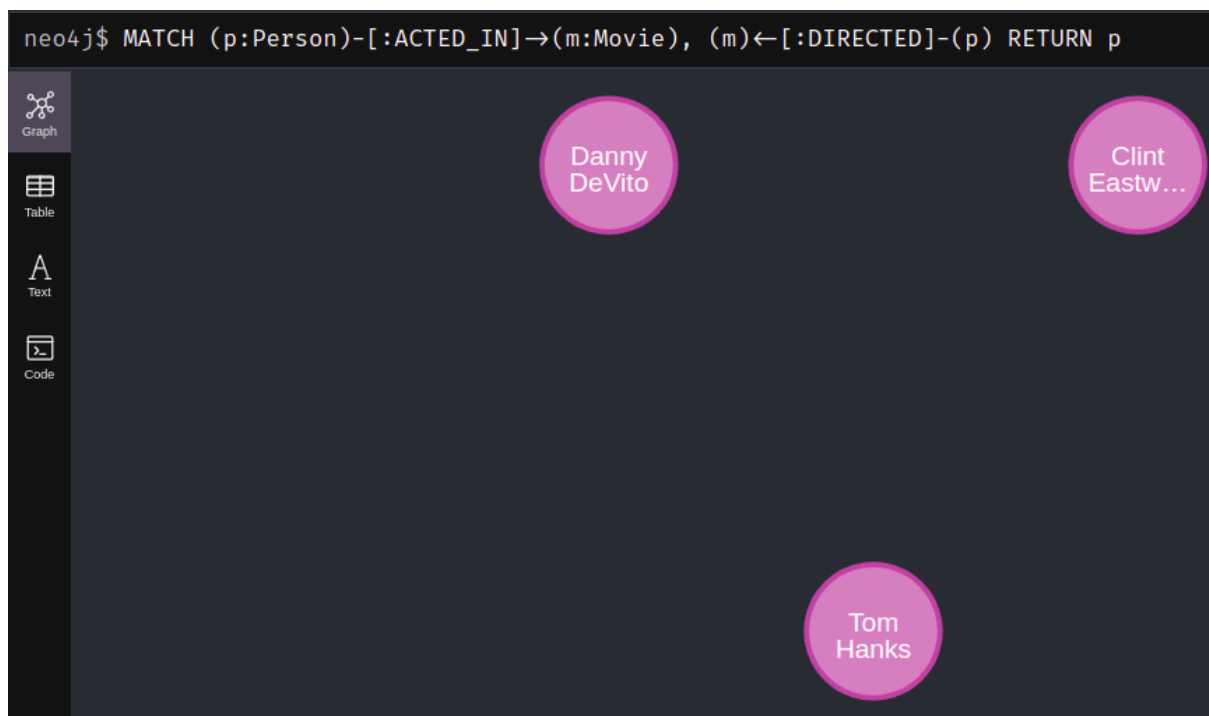


13. Devuelve las personas que hayan actuado y dirigido a la vez una película.

En este apartado crear una query donde se devuelvan las personas que hayan actuado y dirigido a la vez una película, para ello se creó el siguiente comando:

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie),  
(m)-[:DIRECTED]-(p)  
RETURN p
```

El resultado obtenido se puede apreciar en la siguiente captura:



14. Use MERGE to update (ON MATCH) the existing Production node for Forrest Gump to add the company property with a value of Paramount Pictures.

En este apartado se pide usar merge en una query para en el nodo existente Forrest Gump añadirle la propiedad Paramount Pictures:

```
MERGE (m:Movie {title: "Forrest Gump"})
ON MATCH SET m.company = "Paramount Pictures"
ON CREATE SET m.company = "Paramount Pictures"
Return m
```

El resultado obtenido se puede apreciar en la siguiente captura (como no existía se creó el nodo película con dicho nombre):



```
neo4j$ MERGE (m:Movie {title: "Forrest Gump"}) ON MATCH
```

The interface shows a sidebar with icons for Graph, Table, Text, and Code. The main area displays the result of the query, which is a JSON object representing a movie node.

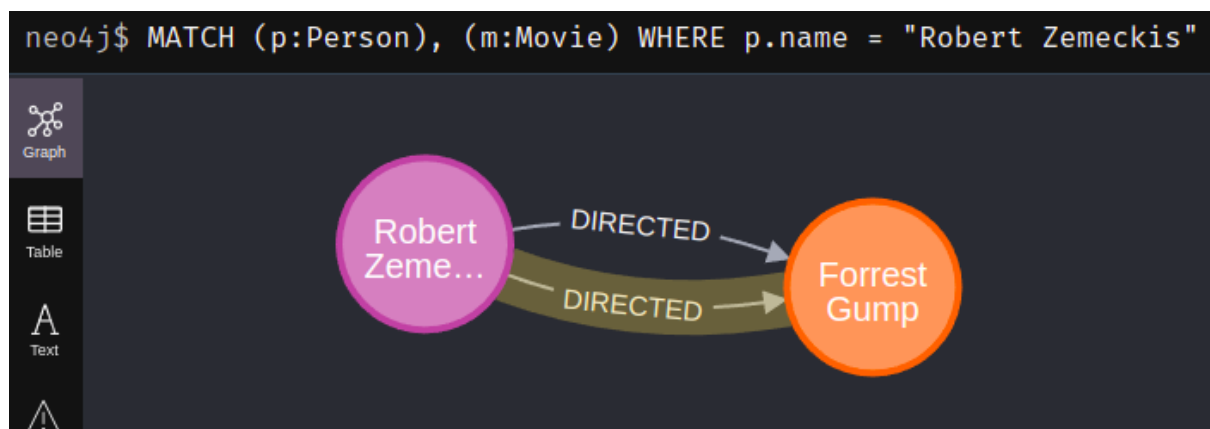
```
1
{
  "identity": 342,
  "labels": [
    "Movie"
  ],
  "properties": {
    "company": "Paramount Pictures",
    "title": "Forrest Gump"
  }
}
```

15. Añadir la relación [:DIRECTED] entre Robert Zemeckis y el film "Forrest Gump":

El query realizado es el siguiente

```
MATCH (p:Person), (m:Movie)
WHERE p.name = "Robert Zemeckis" and m.title = "Forrest Gump"
CREATE (p)-[w:DIRECTED]->(m)
RETURN p,m
```

El resultado obtenido se puede apreciar en la siguiente captura (en la imagen se pueden ver dos relaciones Directed porque se lanzo dos veces la query):

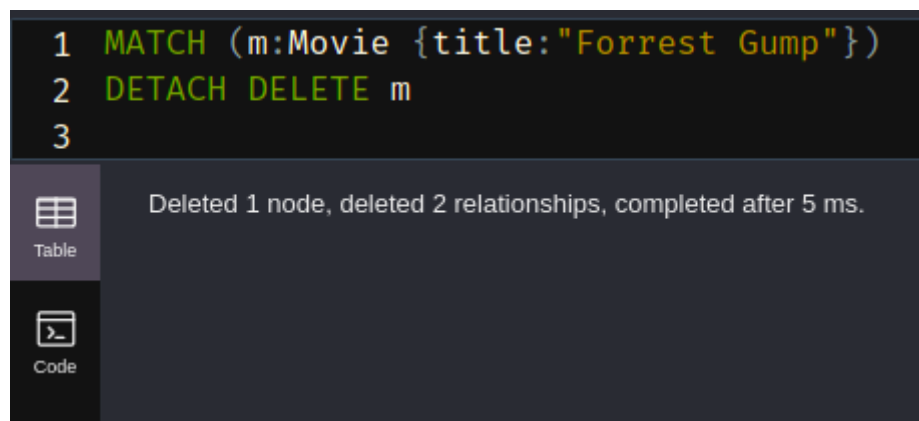


16. Borrar el nodo de la producción "Forrest Gump":

El query realizado es el siguiente:

```
MATCH (m:Movie {title:"Forrest Gump"})  
DETACH DELETE m
```

El resultado obtenido se puede apreciar en la siguiente captura:



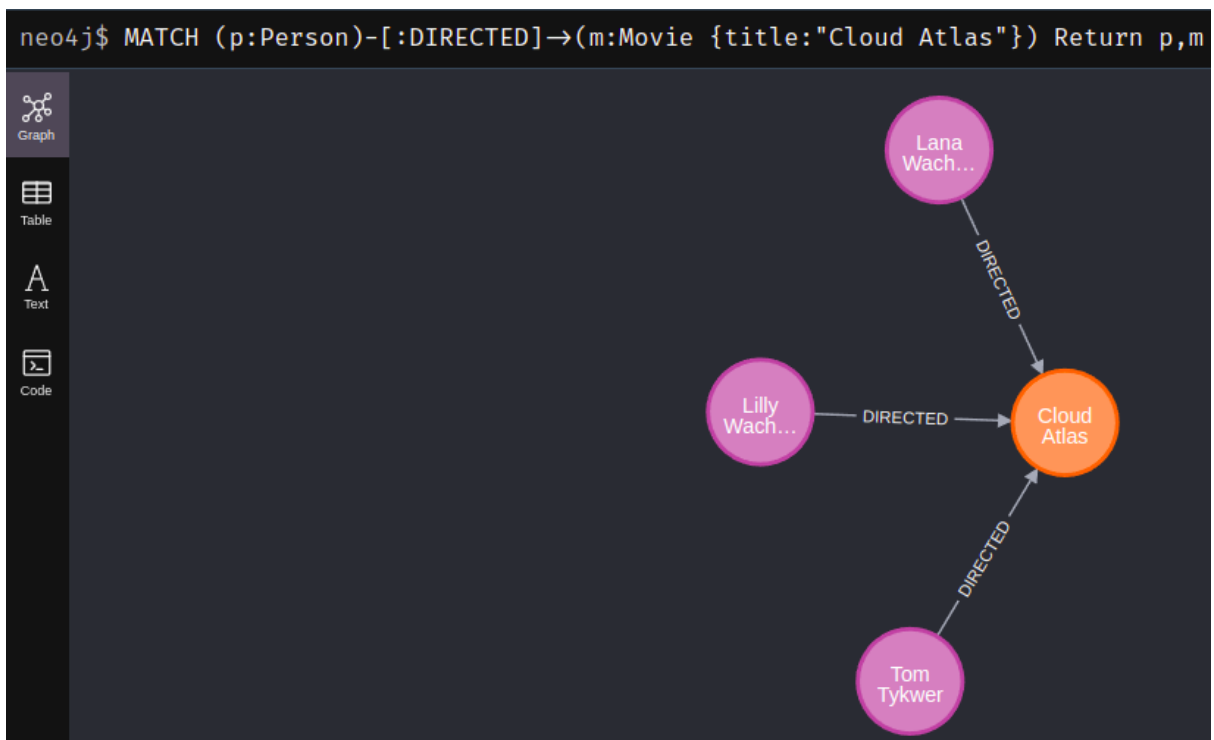
The screenshot shows a Cypher query execution interface. The query is displayed in a dark-themed editor with line numbers 1, 2, and 3. The query text is: `1 MATCH (m:Movie {title:"Forrest Gump"})`, `2 DETACH DELETE m`, and `3`. Below the query editor, there is a sidebar with two icons: a table icon labeled 'Table' and a code icon labeled 'Code'. The 'Table' icon is selected. The main area of the interface displays the result of the query: 'Deleted 1 node, deleted 2 relationships, completed after 5 ms.'

17. Finding who directed Cloud Atlas movie :

En este apartado se pide encontrar quien dirigió la película Atlas, para ello se creó el siguiente comando:

```
MATCH (p:Person)-[:DIRECTED]->(m:Movie {title:"Cloud Atlas"})  
Return p,m
```

El resultado obtenido se puede apreciar en la siguiente captura:



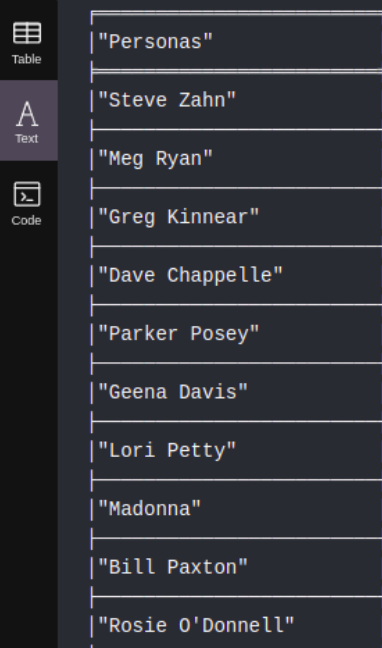
18. Finding all people who have co-acted with Tom Hanks in any movie:

En este apartado se pide encontrar todas las personas que han co-actuado con Tom Hanks, para ello se creó el siguiente comando:

```
MATCH (:Person{name: "Tom  
Hanks"})-[:ACTED_IN]->(:Movie)<-[:ACTED_IN]-(p:Person)  
Return p.name as Personas
```

El resultado obtenido se puede apreciar en la siguiente captura:

```
1 MATCH (:Person{name: "Tom Hanks"})-[:ACTED_IN]->(:Movie)<-[:ACTED_IN]-(p:Person)
2 Return p.name as Personas
3
```



The screenshot shows a database interface with a sidebar on the left containing icons for 'Table', 'Text', and 'Code'. The 'Table' view is selected, displaying a table with one column titled 'Personas'. The table contains the following names: Steve Zahn, Meg Ryan, Greg Kinnear, Dave Chappelle, Parker Posey, Geena Davis, Lori Petty, Madonna, Bill Paxton, and Rosie O'Donnell.

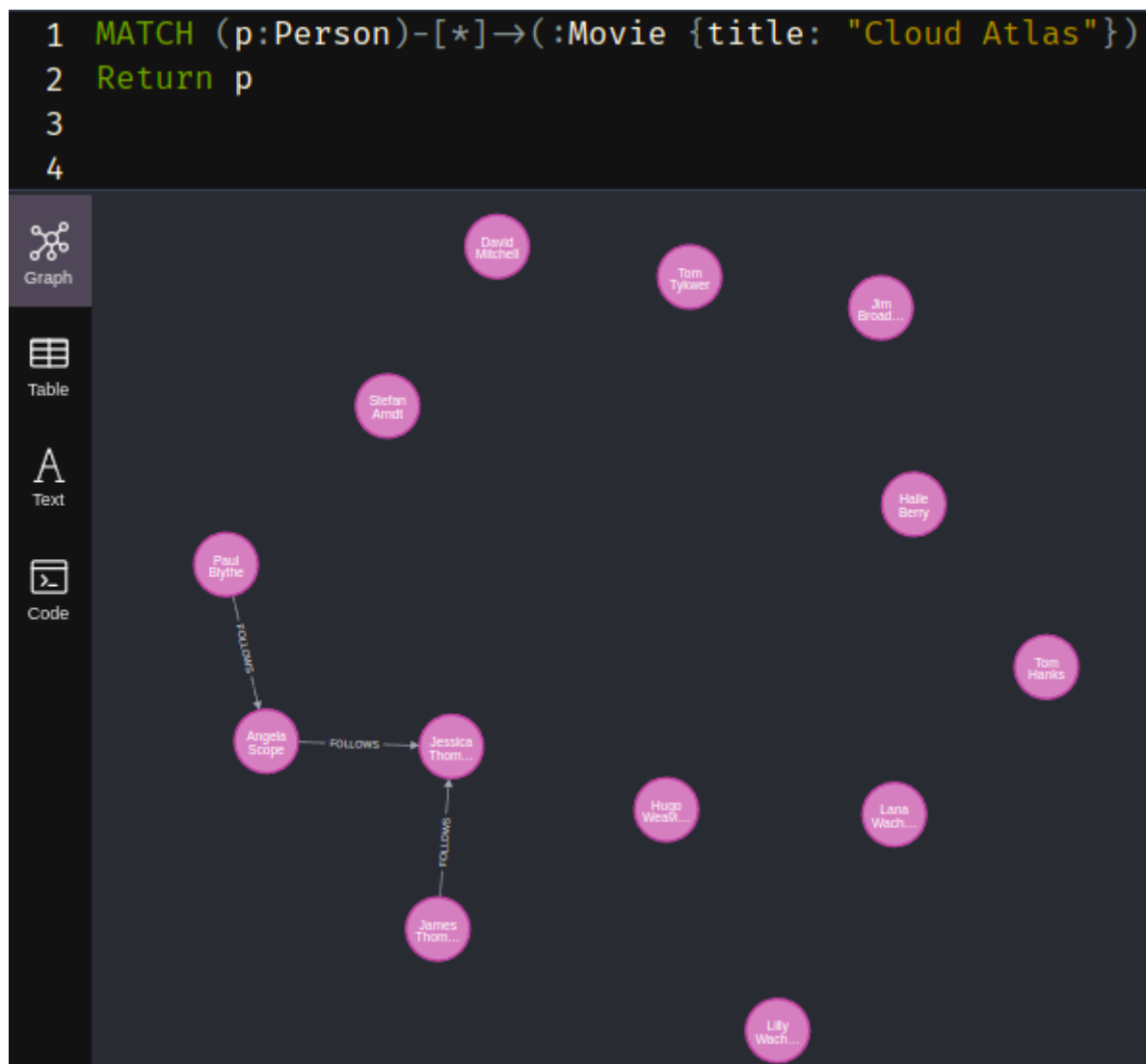
| Personas |
|-------------------|
| "Steve Zahn" |
| "Meg Ryan" |
| "Greg Kinnear" |
| "Dave Chappelle" |
| "Parker Posey" |
| "Geena Davis" |
| "Lori Petty" |
| "Madonna" |
| "Bill Paxton" |
| "Rosie O'Donnell" |

19. Finding all people related to the movie Cloud Atlas in any way:

En este apartado se pide encontrar todas las personas que tienen cualquier tipo de relación con la película Cloud Atlas, para ello se creó el siguiente comando:

```
MATCH (p:Person)-[*]->(Movie {title: "Cloud Atlas"})
Return p
```

El resultado obtenido se puede apreciar en la siguiente captura:

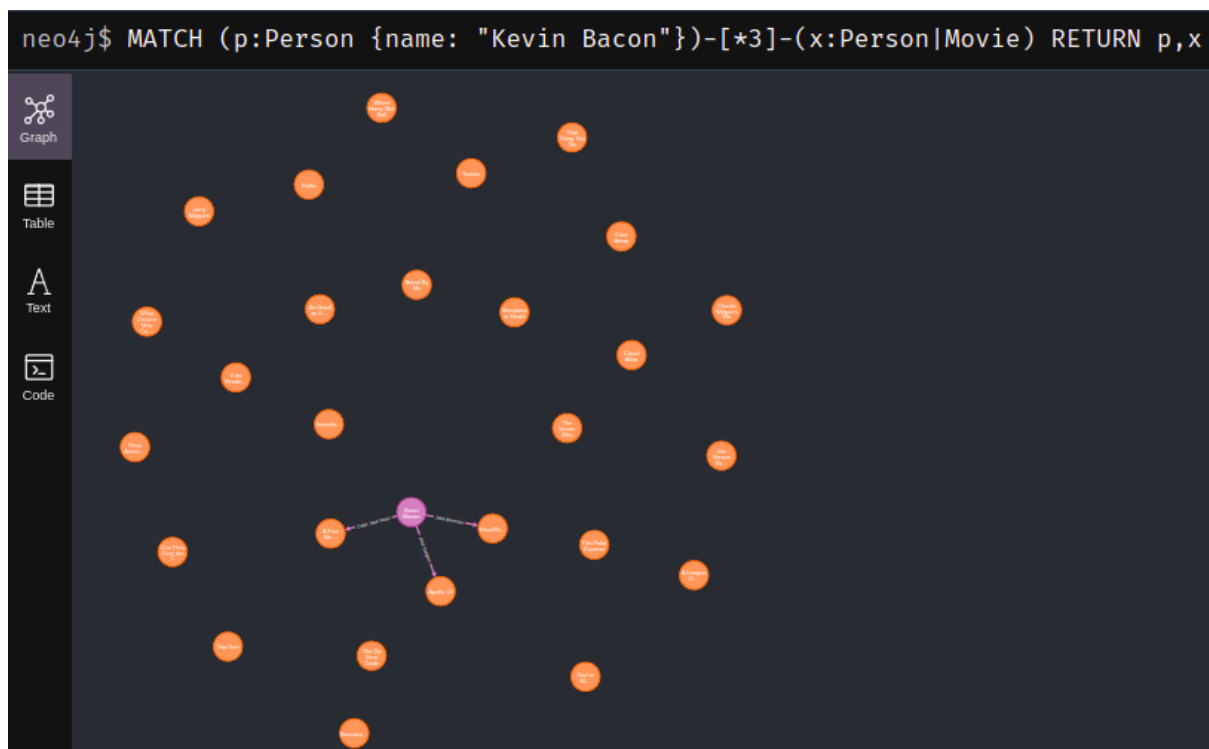


20. Finding Movies and Actors that are 3 hops away from Kevin Bacon :

En este apartado se pide encontrar las películas y los actores que se encuentran a tres saltos de Kevin Bacon, para ello se creó el siguiente comando:

```
MATCH (p:Person {name: "Kevin Bacon"})-[*3]-(x:Person|Movie)
RETURN p,x
```

El resultado obtenido se puede apreciar en la siguiente captura:



PYTHON PY2NEO DRIVER EXERCISES

1. Analiza los siguientes scripts python:

En este apartado se analizar los scripts de python colgados en moodle siendo los siguientes:

- A. **movies1.py**. Este script trata sobre la creación e inserción de los nodos y relaciones de la base de datos de películas pero sin insertar todos los datos, es decir, se han insertado solo algunos.
- B. **movies_full_dataset.py**. Este script realiza la inserción completa de datos de la base de datos movie.
- C. **query1.py**. Este script realiza una petición sobre la base de datos movie. Primero devuelve el nodo cuyo nombre es Keanu Reeves. Posteriormente devuelve el nodo película cuyo nombre es The Matrix, y finalmente, devuelve de todas las personas las 10 primeras.

2. Realiza los siguientes scripts en Python:

A. Script 1:

```
1  from py2neo import Graph
2  from py2neo import NodeMatcher, RelationshipMatcher
3
4  my_graph = Graph(password='neo4j-ADP1')
5
6  nodes = NodeMatcher(my_graph)
7
8  nm = nodes.match("Person")
9
10 file=open("personas.txt", "w")
11 c=1
12
13 for i in nm:
14     file.write(str(c)+"- "+i["name"]+", "+str(i["born"])+"\n")
15     c+=1
16
17 file.close()
```

B. Script 2:

```
1  from py2neo import Graph
2  from py2neo import NodeMatcher, RelationshipMatcher
3
4  my_graph = Graph(password='neo4j-ADP1')
5
6  nodes = NodeMatcher(my_graph)
7
8  nm = nodes.match("Person").where("__born > 1960").all()
9
10 file=open("personas_1960.txt", "w")
11 c=1
12
13 for i in nm:
14     file.write(str(c)+"- "+i["name"]+", "+str(i["born"])+"\n")
15     c+=1
16
17 file.close()
```

C. Script 3:

```
1  from py2neo import Graph
2  from py2neo import NodeMatcher, RelationshipMatcher
3
4  my_graph = Graph(password='neo4j-ADP1')
5
6
7  relations=RelationshipMatcher(my_graph)
8  rm=relations.match(None, r_type="ACTED_IN")
9
10 file=open("actores.txt", "w")
11 c=1
12
13 s=set(rm)
14
15 for i in s:
16     file.write(str(c)+"- "+i.start_node["name"]+"\n")
17     c+=1
18
19 file.close()
```

D. Script 4:

```
1  from py2neo import Graph
2  from py2neo import NodeMatcher, RelationshipMatcher
3  from py2neo import Node
4
5  my_graph = Graph(password='neo4j-ADP1')
6
7  relations=RelationshipMatcher(my_graph)
8  rm=relations.match(None, r_type="ACTED_IN").where("a.born > 1960").all()
9
10 file=open("actores_1960.txt", "w")
11 c=1
12
13 s=set(rm)
14
15 for i in s:
16     file.write(str(c)+"- "+i.start_node["name"]+": "+str(i.start_node["born"])+"\n")
17     c+=1
18
19 file.close()
```