Cypher Query Language

What is Cypher?

- Declarative query language
- Relatively simple
- Very powerful

Cypher is ASCII art

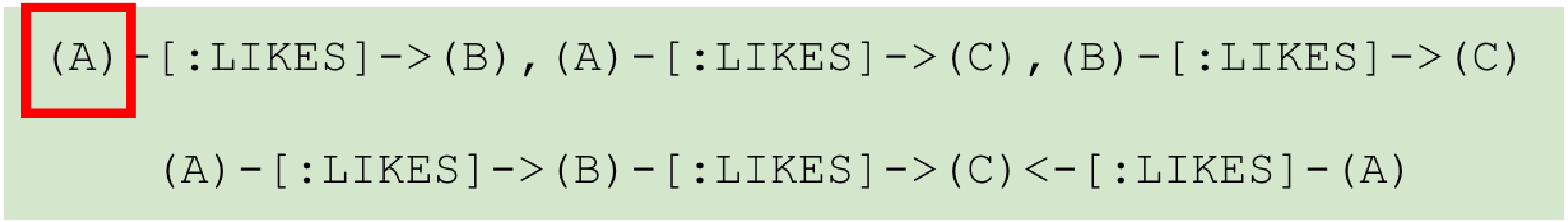
- Optimized for being read by humans

```
(A)-[:LIKES]->(B),(A)-[:LIKES]->(C),(B)-[:LIKES]->(C)

    (A)-[:LIKES]->(B)-[:LIKES]->(C)<-[:LIKES]-(A)
```

Nodes

- () - Circle on a whiteboard



```
(A)-[:LIKES]->(B),(A)-[:LIKES]->(C),(B)-[:LIKES]->(C)

    (A)-[:LIKES]->(B)-[:LIKES]->(C)<-[:LIKES]-(A)
```

Node Labels

- Used to group nodes
- Examples of Nodes
  - () // anonymous node not be referenced later in the query
  - (p) // variable p, a reference to a node used later
  - (:Person) // anonymous node of type Person
  - (p:Person) // p, a reference to a node of type Person
  - (p:Actor:Director) // p, a reference to a node of types Actor and Director

Examining the data model

- CALL db.schema.visualization
  - Gives information about the
    - Nodes
    - Labels
    - Relationships

Create New Database

- :USE SYSTEM
- CREATE DATABASE movieGraph
- SHOW DATABASES
- :USE movieGraph
- CALL db.schema.visualization()
- MATCH (node)-[rel]-(other) RETURN node, rel, other
- :play movies
- MATCH (node)-[rel]->(other) RETURN node, rel, other

Using MATCH to retrieve nodes

- Example queries to the Movie database:
  - MATCH (p:Person)   // returns all Person nodes in the graph
  - RETURN p

Properties

- Retrieve Person nodes that have a born property value of 1970.
  - MATCH (p:Person {born: 1970})
  - RETURN p

- Specify two property values for the query.
  - MATCH (m:Movie {released: 2003, tagline: 'Free your mind'})
  - RETURN m

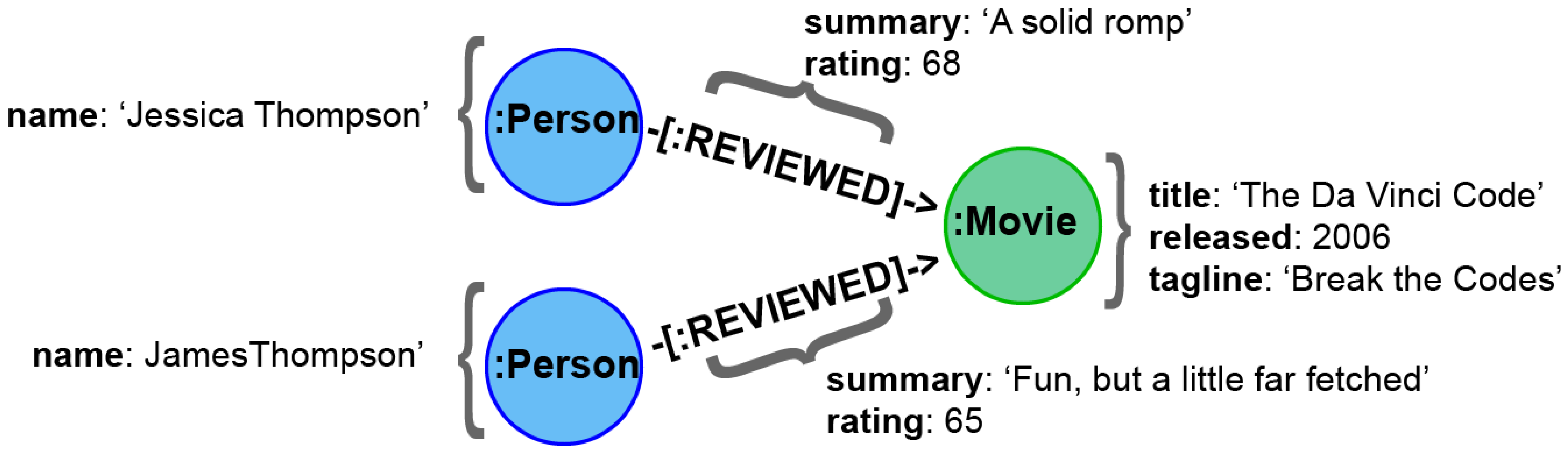
- Returning Property values
  - MATCH (p:Person {born: 1965})
  - RETURN p.name AS name, p.born AS `birth year`

Using a relationship in query

- The actors that acted in the movie - "The Matrix"
  - MATCH (p:Person)-[rel:ACTED_IN]->(m:Movie {title: 'The Matrix'})
  - RETURN p, rel, m

- The movies that "Tom Hanks" acted in and directed:
  - MATCH (p:Person {name: 'Tom Hanks'})-[:ACTED_IN|:DIRECTED]->(m:Movie)
  - RETURN p.name, m.title
- Retrieving the relationship types
  - MATCH (p:Person)-[rel]->(:Movie {title:'The Matrix'})
  - RETURN p.name, type(rel)

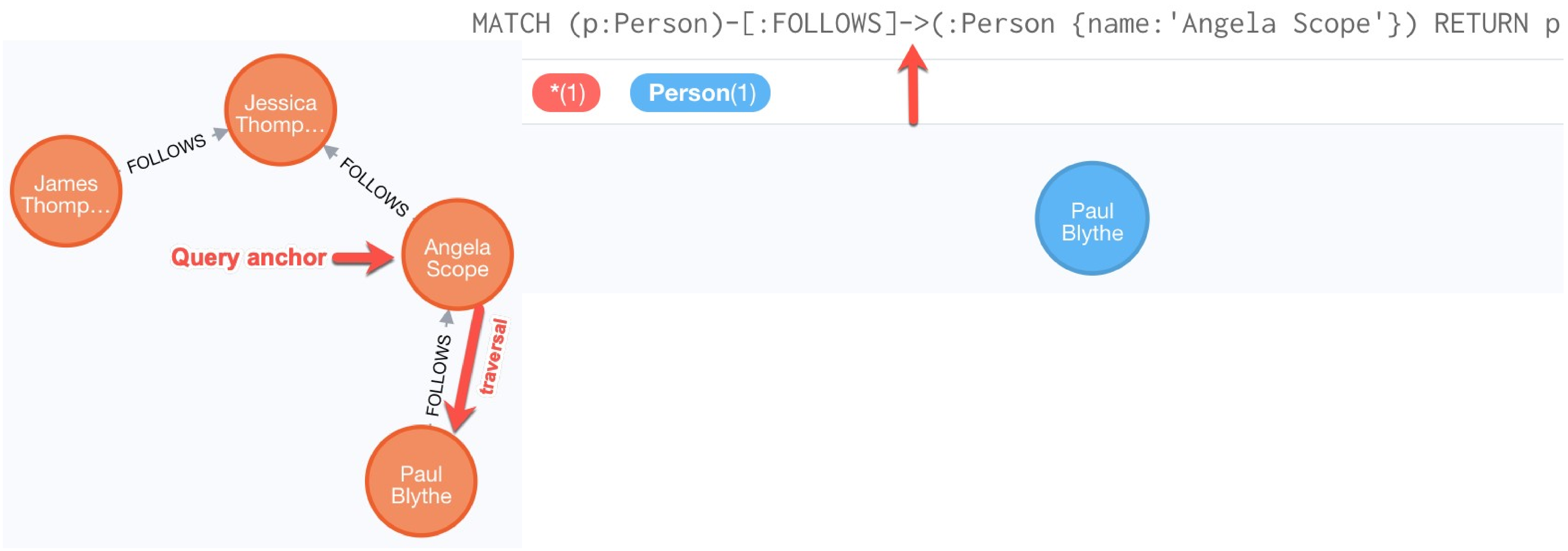
Retrieving relationships properties



- Can also specify property values for a relationship
- Returns the name of the person who gave the movie a rating of 65
  - MATCH (p:Person)-[:REVIEWED {rating: 65}]->(:Movie {title: 'The Da Vinci Code'})
  - RETURN p.name

Using patterns for queries

- We can perform a query that returns all Person nodes who follow **Angela Scope**:

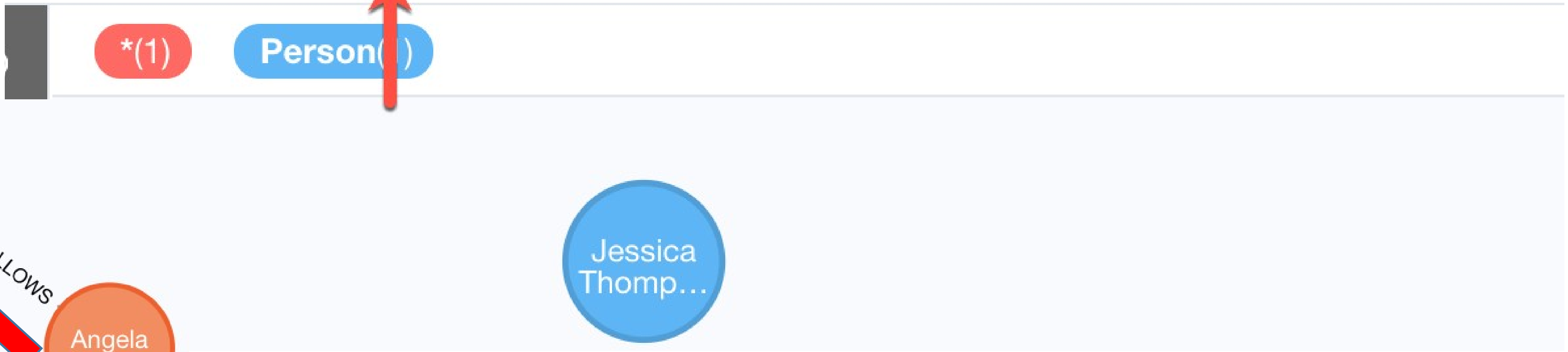


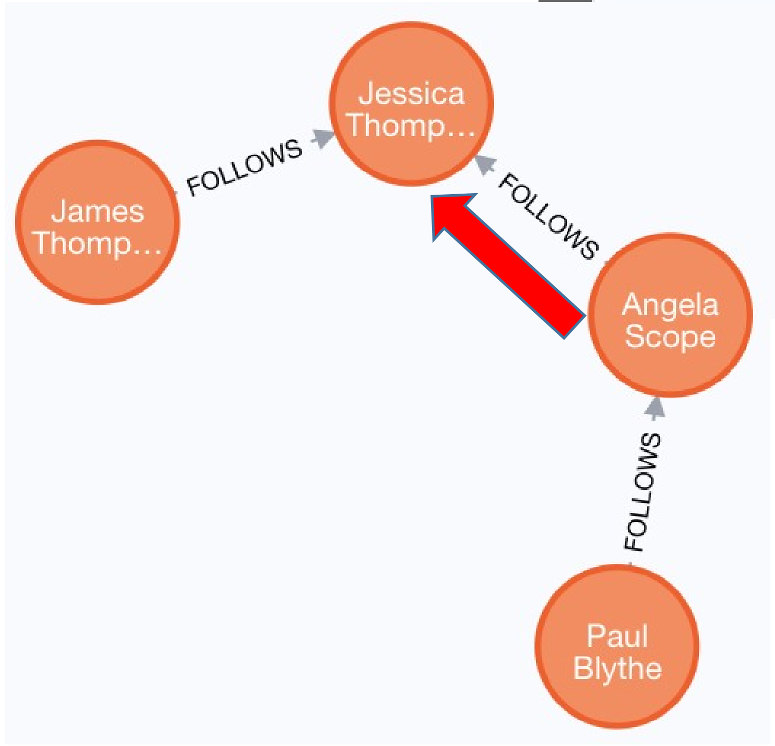
```
MATCH (p:Person)-[:FOLLOWS]->(:Person {name:'Angela Scope'}) RETURN p
```

Using patterns for queries

- If we reverse the direction in the pattern, the query returns different results:

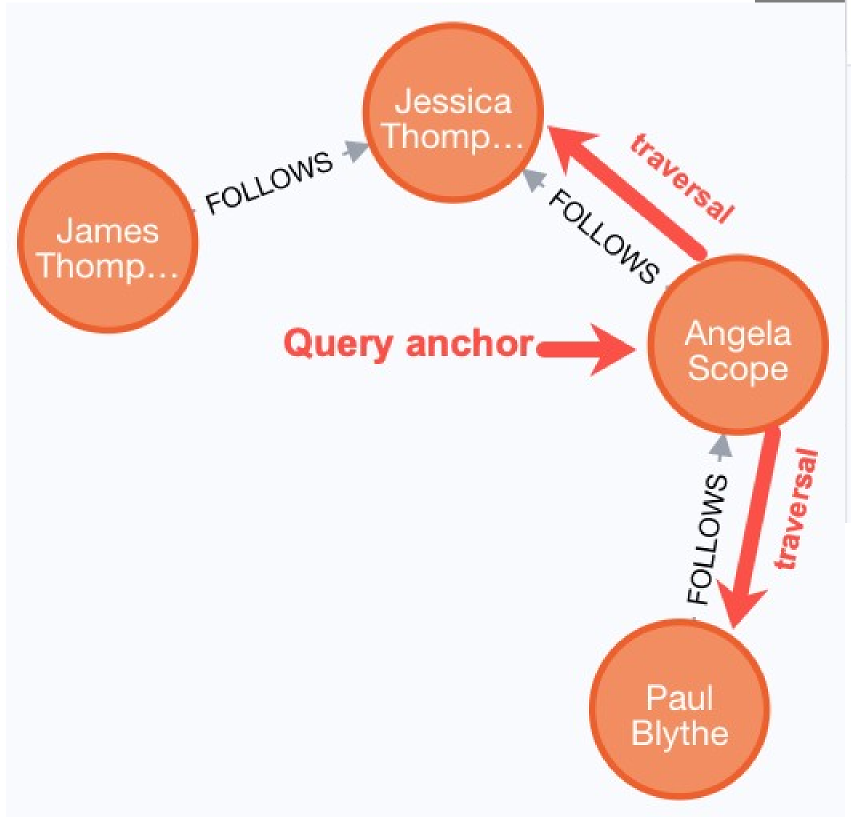
```
MATCH (p:Person)<-[:FOLLOWS]-(:Person {name:'Angela Scope'}) RETURN p
```

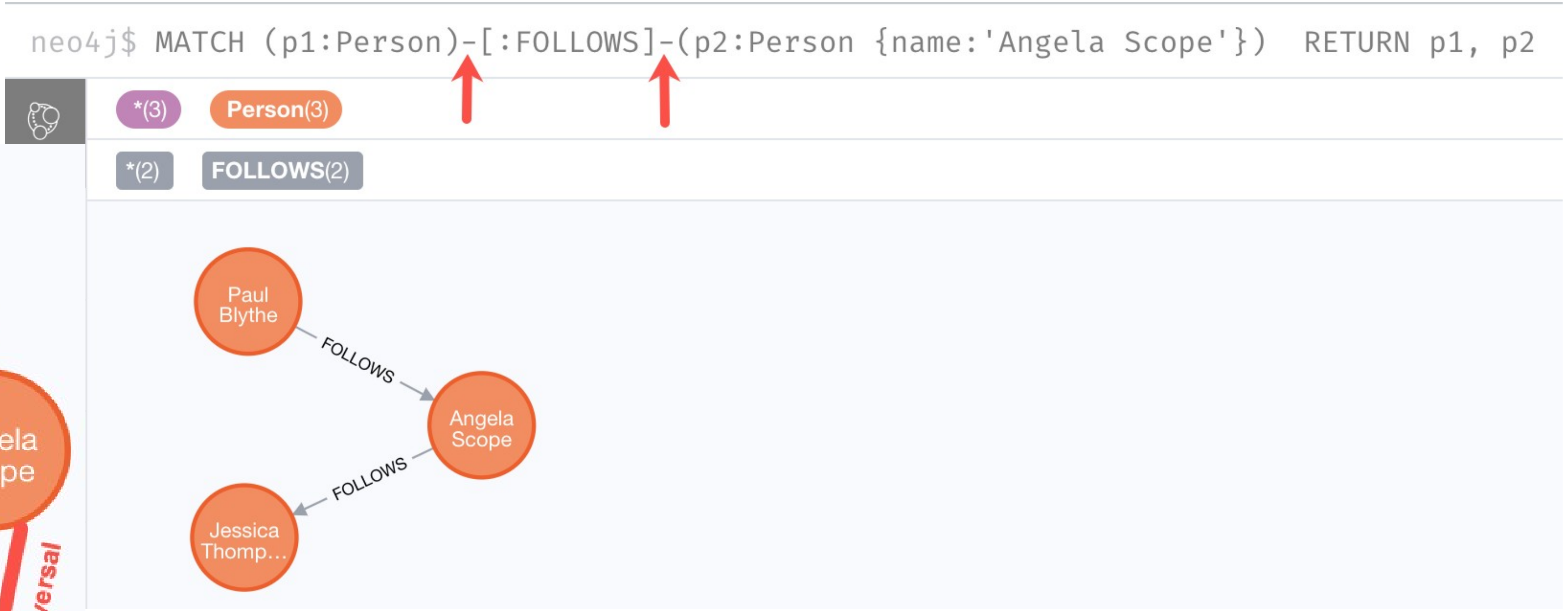




Querying by any direction

- We can also find out what Person nodes are connected by the FOLLOWED relationship in either direction

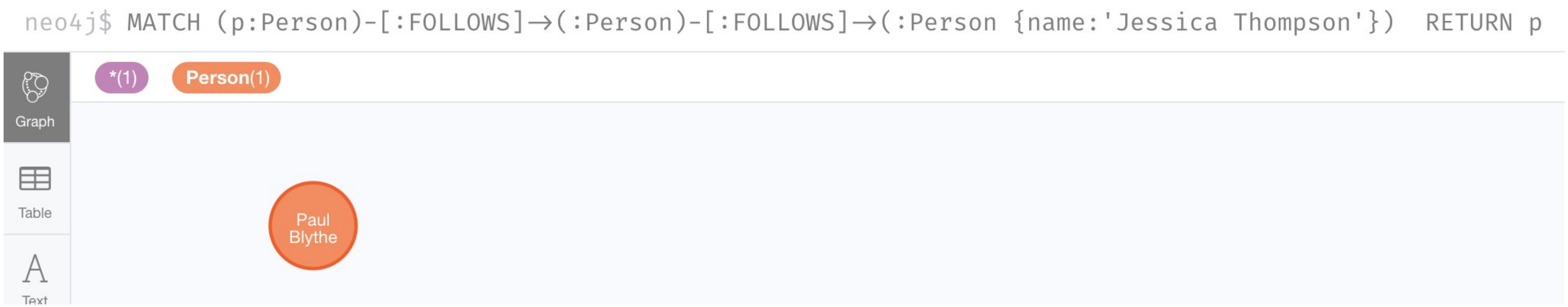




```
neo4j$ MATCH (p1:Person)-[:FOLLOWS]-(p2:Person {name:'Angela Scope'}) RETURN p1, p2
```

Traversing relationships

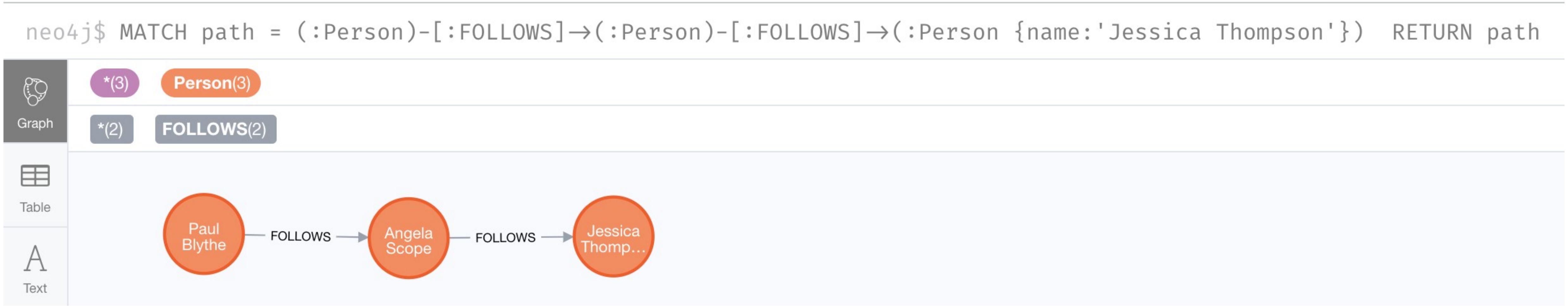
- Return all followers of the followers of Jessica Thompson.



```
neo4j$ MATCH (p:Person)-[:FOLLOWS]→(:Person)-[:FOLLOWS]→(:Person {name:'Jessica Thompson'})  RETURN p
```

Traversing relationships

- To return each person along the path:



```
neo4j$ MATCH path = (:Person)-[:FOLLOWS]→(:Person)-[:FOLLOWS]→(:Person {name:'Jessica Thompson'})  RETURN path
```

Filtering queries using WHERE

- MATCH (p:Person)-[:ACTED_IN]->(m:Movie {released: 2008})
- RETURN p, m

- OR

- MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
- WHERE m.released = 2008
- RETURN p, m

Specify complex conditions

- MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
- WHERE m.released = 2008 OR m.released = 2009
- RETURN p, m

- MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
- WHERE m.released >= 2003 AND m.released <= 2004
- RETURN p.name, m.title, m.released

Ordering results

- MATCH (p:Person)-[:DIRECTED | :ACTED_IN]->(m:Movie)
- WHERE p.name = 'Tom Hanks'
- RETURN m.released, collect(DISTINCT m.title) AS movies ORDER BY m.released DESC

Limiting number of results

- MATCH (m:Movie)
- RETURN m.title as title, m.released as year ORDER BY m.released DESC LIMIT 10

- MATCH (m:Movie)
- RETURN m.title as title, m.released as year ORDER BY m.released DESC SKIP 10 LIMIT 10

Creating nodes

- CREATE (m:Movie:Action {title: 'Batman Begins'})
- RETURN m.title

- CREATE
- (:Person {name: 'Michael Caine', born: 1933}),
- (:Person {name: 'Liam Neeson', born: 1952}),
- (:Person {name: 'Katie Holmes', born: 1978}),
- (:Person {name: 'Benjamin Melniker', born: 1913})

Adding labels to a node

- MATCH (m:Movie)
- WHERE m.title = 'Batman Begins'
- SET m:Action
- RETURN labels(m)

Removing labels from a node

- MATCH (m:Movie:Action)
- WHERE m.title = 'Batman Begins'
- REMOVE m:Action
- RETURN labels(m)

Adding properties to a node

- MATCH (m:Movie)
- WHERE m.title = 'Batman Begins'
- SET m.released = 2005, m.lengthInMinutes = 140
- RETURN m

Adding properties to a node

- MATCH (m:Movie)
- WHERE m.title = 'Batman Begins'
- SET m = {title: 'Batman Begins',
-           released: 2005,
-           lengthInMinutes: 140,
-           videoFormat: 'DVD',
-           grossMillions: 206.5}
- RETURN m

Removing properties from node

- MATCH (m:Movie)
- WHERE m.title = 'Batman Begins'
- SET m.grossMillions = null
- REMOVE m.videoFormat
- RETURN m

Creating relationships

- MATCH (a:Person), (m:Movie)
- WHERE a.name = 'Michael Caine' AND m.title = 'Batman Begins'
- CREATE (a)-[:ACTED_IN]->(m)
- RETURN a, m

- MATCH (a:Person), (m:Movie), (p:Person)
- WHERE a.name = 'Liam Neeson' AND
-     m.title = 'Batman Begins' AND
-     p.name = 'Benjamin Melniker'
- CREATE (a)-[:ACTED_IN]->(m)<-[:PRODUCED]-(p)
- RETURN a, m, p

Adding properties to relationships

- MATCH (a:Person), (m:Movie)
- WHERE a.name = 'Christian Bale' AND m.title = 'Batman Begins'
- CREATE (a)-[rel:ACTED_IN]->(m)
- SET rel.roles = ['Bruce Wayne','Batman']
- RETURN a, m

Removing properties from relationship

- MATCH (a:Person)-[rel:ACTED_IN]->(m:Movie)
- WHERE a.name = 'Christian Bale' AND m.title = 'Batman Begins'
- REMOVE rel.roles
- RETURN a, rel, m

Deleting relationships

- MATCH (a:Person)-[rel:ACTED_IN]->(m:Movie)
- WHERE a.name = 'Christian Bale' AND m.title = 'Batman Begins'
- DELETE rel
- RETURN a, m

Deleting nodes & relationships

- MATCH (p:Person)
- WHERE p.name = 'Liam Neeson'
- DETACH DELETE p

- #When you specify DETACH DELETE for a node, the relationships to and from the node are deleted, then the node is deleted.

THANK YOU