Getting More Out of Queries

neo4j

Overview

At the end of this module, you should be able to write Cypher statements to:

- Filter queries using the `WHERE` clause
- Control query processing
- Control what results are returned
- Work with Cypher lists and dates

Filtering queries using WHERE

Previously you retrieved nodes as follows:

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

A more flexible syntax for the same query is:

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

Testing more than equality:

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

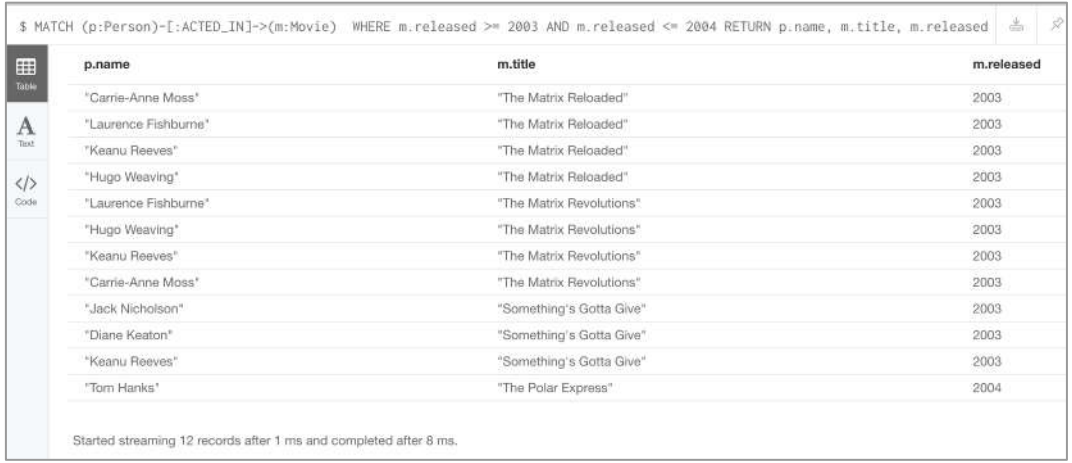
Specifying ranges in WHERE clauses

This query to find all people who acted in movies released between 2003 and 2004:

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

Is the same as:

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



p.name	m.title	m.released
"Carrie-Anne Moss"	"The Matrix Reloaded"	2003
"Laurence Fishburne"	"The Matrix Reloaded"	2003
"Keanu Reeves"	"The Matrix Reloaded"	2003
"Hugo Weaving"	"The Matrix Reloaded"	2003
"Laurence Fishburne"	"The Matrix Revolutions"	2003
"Hugo Weaving"	"The Matrix Revolutions"	2003
"Keanu Reeves"	"The Matrix Revolutions"	2003
"Carrie-Anne Moss"	"The Matrix Revolutions"	2003
"Jack Nicholson"	"Something's Gotta Give"	2003
"Diane Keaton"	"Something's Gotta Give"	2003
"Keanu Reeves"	"Something's Gotta Give"	2003
"Tom Hanks"	"The Polar Express"	2004

Started streaming 12 records after 1 ms and completed after 8 ms.

Testing labels

These queries:

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

```
MATCH (p:Person)-[:ACTED_IN]->(:Movie {title: 'The Matrix'})
RETURN p.name
```

Can be rewritten as:

```
MATCH (p)
WHERE p:Person
RETURN p.name
```

```
MATCH (p)-[:ACTED_IN]->(m)
WHERE p:Person AND m:Movie AND m.title='The Matrix'
RETURN p.name
```

Testing the existence of a property

Find all movies that *Jack Nicholson* acted in that have a tagline, returning the title and tagline of the movie:

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
WHERE p.name='Jack Nicholson' AND exists(m.tagline)
RETURN m.title, m.tagline
```

```
$ MATCH (p:Person)-[:ACTED_IN]->(m:Movie) WHERE p.name='Jack Nicholson' AND exists(m.tagline) RETURN m.title, m.tagline
```

Table | Text | Code

m.title	m.tagline
"A Few Good Men"	"In the heart of the nation's capital, in a courthouse of the U.S. government, one man will stop at nothing to keep his honor, and one will stop at nothing to find the truth."
"As Good as It Gets"	"A comedy from the heart that goes for the throat."
"Hoffa"	"He didn't want law. He wanted justice."
"One Flew Over the Cuckoo's Nest"	"If he's crazy, what does that make you?"

Testing strings

Find all actors whose name begins with *Michael*:

```
MATCH (p:Person)-[:ACTED_IN]->()
WHERE p.name STARTS WITH 'Michael'
RETURN p.name
```


```
MATCH (p:Person)-[:ACTED_IN]->()
WHERE toLower(p.name) STARTS WITH 'michael'
RETURN p.name
```

Testing with regular expressions

Find people whose name starts with *Tom*:

```
MATCH (p:Person)
WHERE p.name =~'Tom.*'
RETURN p.name
```

```
$ MATCH (p:Person) WHERE p.name =~'Tom.*' RETURN p.name
```

Table | Text | Code

p.name
"Tom Cruise"
"Tom Skerritt"
"Tom Hanks"
"Tom Tykwer"

Testing with patterns - 1

Find all people who wrote movies returning their names and the title of the movie they wrote:

```
MATCH (p:Person)-[:WROTE]->(m:Movie)
RETURN p.name, m.title
```

```
$ MATCH (p:Person)-[:WROTE]->(m:Movie) RETURN p.name, m.title
```

Table | Text | Code

p.name	m.title
"Aaron Sorkin"	"A Few Good Men"
"Jim Cash"	"Top Gun"
"Cameron Crowe"	"Jerry Maguire"
"Nora Ephron"	"When Harry Met Sally"
"David Mitchell"	"Cloud Atlas"
"Lilly Wachowski"	"V for Vendetta"
"Lana Wachowski"	"V for Vendetta"
"Lana Wachowski"	"Speed Racer"
"Lilly Wachowski"	"Speed Racer"
"Nancy Meyers"	"Something's Gotta Give"

Started streaming 10 records in less than 1 ms and completed after 1 ms.

Testing with patterns - 2

Find the people who wrote movies, but did not direct them, returning their names and the title of the movie:

```
MATCH (p:Person)-[:WROTE]->(m:Movie)
WHERE NOT exists( (p)-[:DIRECTED]->(m) )
RETURN p.name, m.title
```

$ MATCH (p:Person)-[:WROTE]->(m:Movie) WHERE NOT exists((…

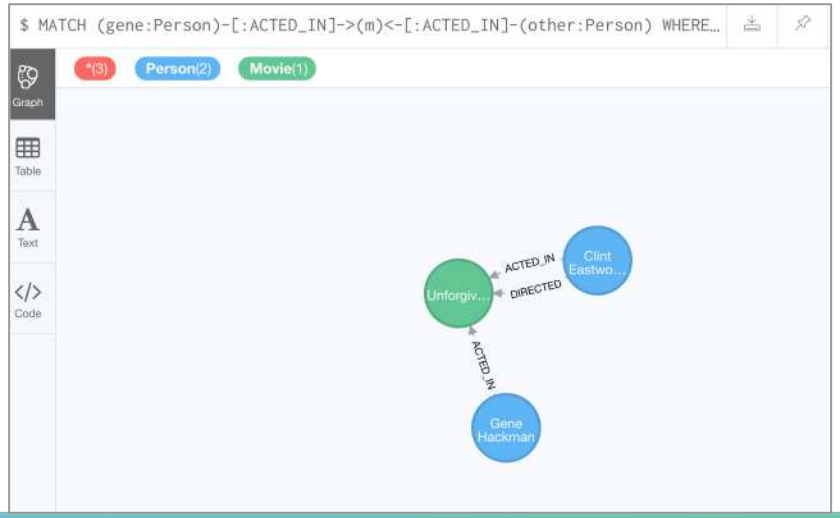
Table | Text | Code

p.name	m.title
"Aaron Sorkin"	"A Few Good Men"
"Jim Cash"	"Top Gun"
"Nora Ephron"	"When Harry Met Sally"
"David Mitchell"	"Cloud Atlas"
"Lana Wachowski"	"V for Vendetta"
"Lilly Wachowski"	"V for Vendetta"

Testing with patterns - 3

Find *Gene Hackman* and the movies that he acted in with another person who also directed the movie, returning the nodes found:

```
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
```



Testing with list values - 1

Find all people born in 1965 and 1970:

```
MATCH (p:Person)
WHERE p.born IN [1965, 1970]
RETURN p.name as name, p.born as yearBorn
```

```
$ MATCH (p:Person) WHERE p.born IN [1965, 1970] RETURN p.name as name, p.born as yearBorn
```

Table | Text | Code

name	yearBorn
"Lana Wachowski"	1965
"Jay Mohr"	1970
"River Phoenix"	1970
"Ethan Hawke"	1970
"Brooke Langton"	1970
"Tom Tykwer"	1965
"John C. Reilly"	1965

Started streaming 7 records after 1 ms and completed after 2 ms.

Testing with list values - 2

Find the actor who played *Neo* in the movie, *The Matrix*:

```
MATCH (p:Person)-[r:ACTED_IN]->(m:Movie)
WHERE  'Neo' IN r.roles AND m.title='The Matrix'
RETURN p.name
```

```
$ MATCH (p:Person)-[r:ACTED_IN]->(m:Movie) WHERE "Neo" IN r.roles and m.title="The Matrix" RETURN p.name
```

Table

p.name
"Keanu Reeves"

Exercise 4: Filtering queries using the WHERE clause

In Neo4j Browser:

:play intro-exercises

Then follow instructions for Exercise 4.

Controlling query processing

- Multiple MATCH clauses
- Varying length paths
- Collecting results into lists
- Counting results

Specifying multiple MATCH patterns

This query to find people who either acted or directed a movie released in *2000* is specified with two MATCH patterns:

```
MATCH (a:Person)-[:ACTED_IN]->(m:Movie),
      (m:Movie)<-[:DIRECTED]-(d:Person)
WHERE m.released = 2000
RETURN a.name, m.title, d.name
```

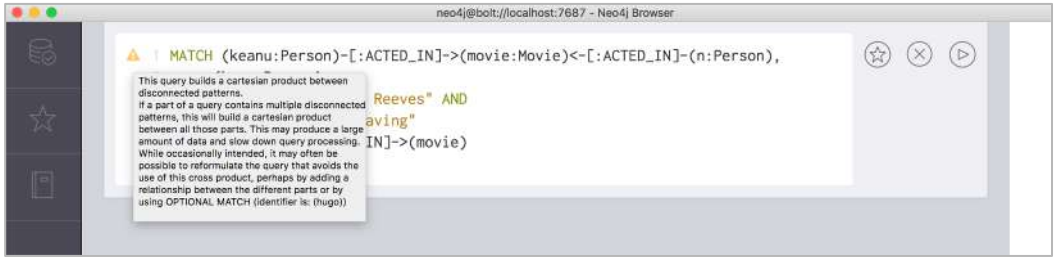
A best practice is to use a single MATCH pattern if possible:

```
MATCH (a:Person)-[:ACTED_IN]->(m:Movie)<-[:DIRECTED]-(d:Person)
WHERE m.released = 2000
RETURN a.name, m.title, d.name
```

Example 1: Using two MATCH patterns

Find the actors who acted in the same movies as *Keanu Reeves*, but not when *Hugo Weaving* acted in the same movie:

```
MATCH (keanu:Person)-[:ACTED_IN]->(movie:Movie)<-[:ACTED_IN]-(n:Person), (hugo:Person)
WHERE keanu.name='Keanu Reeves' AND hugo.name='Hugo Weaving' AND
      NOT (hugo)-[:ACTED_IN]->(movie)
RETURN n.name
```



Example 2: Using two MATCH patterns

Retrieve the movies that *Meg Ryan* acted in and their respective directors, as well as the other actors that acted in these movies:

```
MATCH (meg:Person)-[:ACTED_IN]->(m:Movie)<-[:DIRECTED]-(d:Person),
      (other:Person)-[:ACTED_IN]->(m)
WHERE meg.name = 'Meg Ryan'
RETURN m.title as movie, d.name AS director , other.name AS `co-actors`
```

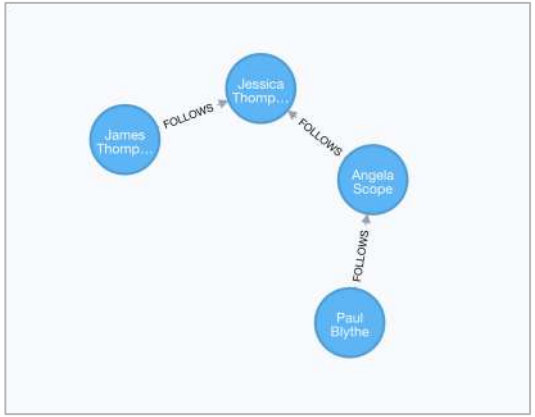
$ MATCH (meg:Person)-[:ACTED_IN]->(m:Movie)<-[:DIRECTED]-(d:Person), (other:Person)-[:ACTED_IN]->(m...

Table | Text | Code

movie	director	co-actors
"Joe Versus the Volcano"	"John Patrick Stanley"	"Tom Hanks"
"Joe Versus the Volcano"	"John Patrick Stanley"	"Nathan Lane"
"When Harry Met Sally"	"Rob Reiner"	"Bruno Kirby"
"When Harry Met Sally"	"Rob Reiner"	"Carrie Fisher"
"When Harry Met Sally"	"Rob Reiner"	"Billy Crystal"
"Sleepless in Seattle"	"Nora Ephron"	"Rosie O'Donnell"
"Sleepless in Seattle"	"Nora Ephron"	"Tom Hanks"
"Sleepless in Seattle"	"Nora Ephron"	"Bill Pullman"
"Sleepless in Seattle"	"Nora Ephron"	"Victor Garber"
"Sleepless in Seattle"	"Nora Ephron"	"Rita Wilson"
"You've Got Mail"	"Nora Ephron"	"Dave Chappelle"
"You've Got Mail"	"Nora Ephron"	"Steve Zahn"
"You've Got Mail"	"Nora Ephron"	"Greg Kinnear"
"You've Got Mail"	"Nora Ephron"	"Parker Posey"
"You've Got Mail"	"Nora Ephron"	"Tom Hanks"
"Top Gun"	"Tony Scott"	"Tom Skerritt"

Started streaming 20 records in less than 1 ms and completed after 2 ms.

Specifying varying length paths



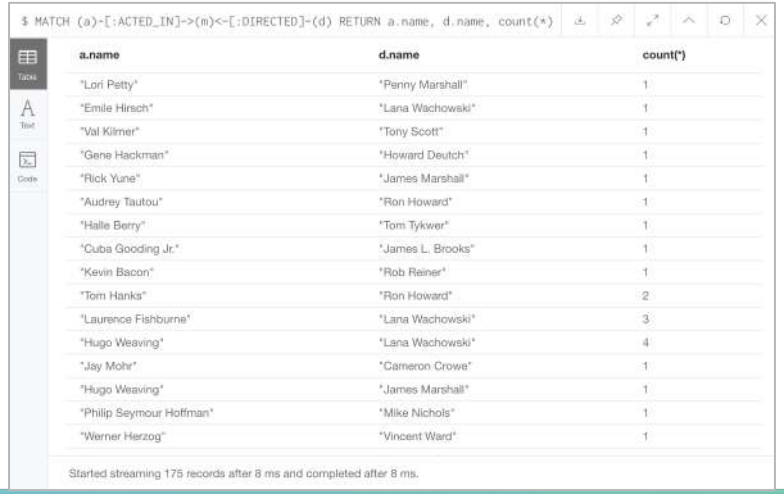
Find all people who are exactly two hops away from *Paul Blythe*:

```
MATCH (follower:Person)-[:FOLLOWS*2]->(p:Person)
WHERE follower.name = 'Paul Blythe'
RETURN p
```

Aggregation in Cypher

- Different from SQL - no need to specify a grouping key.
- As soon as you use an aggregation function, all non-aggregated result columns automatically become grouping keys.
- Implicit grouping based upon fields in the RETURN clause.

```
// implicitly groups by a.name and d.name
MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d)
RETURN a.name, d.name, count(*)
```



$ MATCH (a)-[:ACTED_IN]->(m)<-[:DIRECTED]-(d) RETURN a.name, d.name, count(*)

a.name	d.name	count(*)
"Lori Petty"	"Penny Marshall"	1
"Emile Hirsch"	"Lana Wachowski"	1
"Val Kilmer"	"Tony Scott"	1
"Gene Hackman"	"Howard Deutch"	1
"Rick Yune"	"James Marshall"	1
"Audrey Tautou"	"Ron Howard"	1
"Halle Berry"	"Tom Tykwer"	1
"Cuba Gooding Jr."	"James L. Brooks"	1
"Kevin Bacon"	"Rob Reiner"	1
"Tom Hanks"	"Ron Howard"	2
"Laurence Fishburne"	"Lana Wachowski"	3
"Hugo Weaving"	"Lana Wachowski"	4
"Jay Mohr"	"Cameron Crowe"	1
"Hugo Weaving"	"James Marshall"	1
"Philip Seymour Hoffman"	"Mike Nichols"	1
"Werner Herzog"	"Vincent Ward"	1

Started streaming 175 records after 8 ms and completed after 8 ms.

Collecting results

Find the movies that Tom Cruise acted in and return them as a list:

```
MATCH (p:Person)-[:ACTED_IN]->(m:Movie)
WHERE p.name ='Tom Cruise'
RETURN collect(m.title) AS `movies for Tom Cruise`
```

```
$ MATCH (p:Person)-[:ACTED_IN]->(m:Movie) WHERE p.name ='Tom Cruise' RETURN collect(m.title) AS `mo…
```

Table

movies for Tom Cruise
["Jerry Maguire", "Top Gun", "A Few Good Men"]

Text

Counting results

Find all of the actors and directors who worked on a movie, return the count of the number paths found between actors and directors and collect the movies as a list:

```
MATCH (actor:Person)-[:ACTED_IN]->(m:Movie)<-[:DIRECTED]-(director:Person)
RETURN actor.name, director.name, count(m) AS collaborations,
       collect(m.title) AS movies
```

$ MATCH (actor:Person)-[:ACTED_IN]->(m:Movie)<-[:DIRECTED]-(director:Person) RETURN actor.name, dir…

Table | Text | Code

actor.name	director.name	collaborations	movies
"Lori Petty"	"Penny Marshall"	1	["A League of Their Own"]
"Emile Hirsch"	"Lana Wachowski"	1	["Speed Racer"]
"Val Kilmer"	"Tony Scott"	1	["Top Gun"]
"Gene Hackman"	"Howard Deutch"	1	["The Replacements"]
"Rick Yune"	"James Marshall"	1	["Ninja Assassin"]
"Audrey Tautou"	"Ron Howard"	1	["The Da Vinci Code"]
"Halle Berry"	"Tom Tykwer"	1	["Cloud Atlas"]
"Cuba Gooding Jr."	"James L. Brooks"	1	["As Good as It Gets"]
"Kevin Bacon"	"Rob Reiner"	1	["A Few Good Men"]
"Tom Hanks"	"Ron Howard"	2	["The Da Vinci Code", "Apollo 13"]
"Laurence Fishburne"	"Lana Wachowski"	3	["The Matrix", "The Matrix Reloaded", "The Matrix Revolutions"]
"Hugo Weaving"	"Lana Wachowski"	4	["The Matrix", "The Matrix Reloaded", "The Matrix Revolutions", "Cloud Atlas"]
"Jay Mohr"	"Cameron Crowe"	1	["Jerry Maguire"]
"Hugo Weaving"	"James Marshall"	1	["V for Vendetta"]
"Philip Seymour Hoffman"	"Mike Nichols"	1	["Charlie Wilson's War"]
"Werner Herzog"	"Vincent Ward"	1	["What Dreams May Come"]

Started streaming 175 records after 14 ms and completed after 14 ms.

Exercise 5: Controlling query processing

In Neo4j Browser:

:play intro-exercises

Then follow instructions for Exercise 5.