

Resources available

# **Topics Covered**

Cypher queries & clauses

Lists / IN

WITH, UNWIND, UNION

HTTP RESTful API

Language Drivers

0:06 /1:01

Continue >







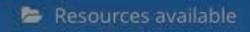


## **Section Overview**

- Lecture 1 Section intro
- Lecture 2 Other Cypher Queries & Clauses
- Lecture 3 Neo4j HTTP REST API
- Lecture 4 Language Drivers

# Neo4j

More on Cypher & Neo4j



### **Node Patterns**

(n:Person) Node with Person label

(n:Person:Member) Node with Person & Member labels

(n:Person {name:'Jeff'}) Node with properties

Continue >

# Relationship Patterns

- (a)-->(b) Relationship from a to b
- (a)--(b) Relationship in any direction between a and b
- (a:Person)-->(b) Node labeled Person with relationship to b
- (a)-[:KNOWS]->(b) Relationship of type "KNOWS" from a to b
- (a)-[:KNOWS|:LOVES]->(b) Relationship of type 'KNOWS' OR 'LOVES from a to b'
- (a)-[r]->(b) Bind the relationship to variable r
- (a)-[:KNOWS]->(b {property:value}) Relationship of type 'KNOWS' with a declared property

udemy

# Variable Length Patterns

- (a)-[\*1..5]->(b) Variable length path of between 1 and 5 relationships from a to b
- (a)-[\*]->(b) Variable length of any number or relationships from a to b
- size((a)-->()-->() ) Count paths that match pattern

# Lists & the IN Operator

- If Cypher knows something exists in a list, the result will be true
- Any list that contains a NULL and doesn't have a matching

element will return NULL

MATCH (n)
WHERE id(n) IN [0,3,5]
RETURN n

Expression	Result
2 IN [1, 2, 3]	TRUE
2 IN [1, NULL, 3]	NULL
2 IN [1, 2, NULL]	TRUE
2 IN [1]	FALSE
2 IN []	FALSE
NULL IN [1,2,3]	NULL
NULL IN [1,NULL,3]	NULL
NULL IN []	FALSE

### OPTIONAL MATCH

- Works like MATCH except if no matches are found, OPTIONAL
   MATCH will use NULLs for missing parts of the pattern
- Similar to Outer join in SQL

```
MATCH (a:Movie { title: 'Some Movie' })
OPTIONAL MATCH (a)-->(x)
RETURN x
```

## Aliases

You can rename columns and create aliases by using "AS"

MATCH (a {name: 'Bob'})
Return a.birth\_date AS Birthday

# Order By

You can use ORDER BY to sort by property

```
MATCH (n)
RETURN n
ORDER BY n.last_name DESC
```

MATCH (n)

RETURN n

ORDER BY n.last\_name, n.first\_name DESC

## Limit Results

You can limit results with LIMIT

MATCH (n)
RETURN n
ORDER BY n.name
LIMIT 5

## WITH

 Allows queries to be chained together, piping the results from one to be used as starting points or criteria in the next

```
MATCH (david { name: "David" })--(otherPerson)-->()
WITH otherPerson, count(*) AS foaf
WHERE foaf > 1
RETURN otherPerson
```

## UNWIND

Expands a list into a sequence of rows

UNWIND [1,2,3] AS x RETURN x

## UNION

Used to combine the result of multiple queries

MATCH (n:Actor)

RETURN n.name AS name

UNION ALL MATCH (n:Movie)

RETURN n.title AS name

# String Match Negation

You can use NOT to exclude all matches on a given string

MATCH (n)
WHERE NOT n.name ENDS WITH 'r'
RETURN n

# Operators

- Mathematical +, -, \*, /, %, ^
- Comparison =, <>, <, >, <=, >=, IS NULL, IS NOT NULL
- Special STARTS WITH, ENDS WITH, CONTAINS
- · Boolean AND, OR, XOR, NOT

# Aggregation

- Aggregate or group data while traversing patterns
- Happens in the RETURN clause
- Most common aggregation functions are available
- · count, sum, avg, min, max
- NULL values are skipped during aggregation

### Count

Count(\*) - Number of matching rows

Count(variable) - Number of non-NULL values

Count(DISTINCT variable) - Removes duplicates

MATCH (:Person)

RETURN count(\*) AS people

## SUM & AVG

- Sum(a.property) Find the sum of the total from any given rows
- Avg(a.property) get the average from the column given

```
MATCH (e:Employee) // Finds total and avg salary RETURN SUM(e.sal), AVG(e.sal)
```

