Querying Data with Cypher

Roland Guijt
MVP | CONSULTANT | TRAINER | AUTHOR

@rolandguijt rolandguijt.com

Module Overview

Data modeling

What is Cypher?

MATCH RETURN

Other language elements

Advanced syntax

Data Modelling

Consider where to put data based on your query needs

Often normalizing data to other nodes gives more flexibility

The performance impact on extra relationships is much lower than with relational databases

What Is Cypher?

Graph query language

Declarative

Easy on the brain

Pattern matching

Clauses

Cypher Queries Are About Pattern Matching

Think of a whiteboard friendly data pattern

Translate into ASCII art

Surround by clauses

Example


```
() -[:PLAYED]->()
```

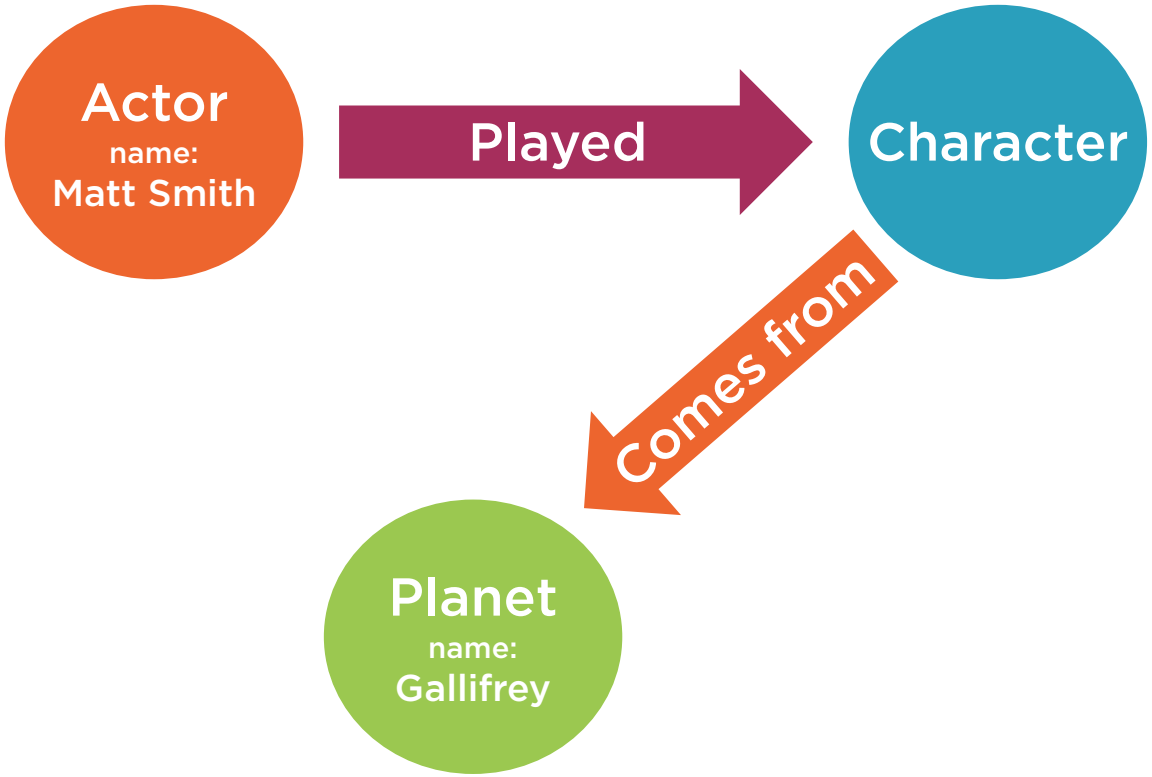
Example

(:Actor) –[:PLAYED]->(:Character)

Example

(:Actor {name: 'Matt Smith'}) –[:PLAYED]->(:Character)

Example



(:Actor{name:'Matt Smith'}) –[:PLAYED]->
(:Character)-[:COMES_FROM]->(:Planet{name:'Gallifrey'})

The MATCH and RETURN Keywords


```
(:Actor{name:'Matt Smith'}) -[:PLAYED]->(:Character)
(:Actor{name:'Matt Smith'}) -[:PLAYED]->(c:Character)
MATCH
(:Actor{name:'Matt Smith'}) -[:PLAYED]->(c:Character)
RETURN c
```

Query Example

Return the name properties of all nodes with the label Actor and put them side by side with the name properties of all nodes that are on the other end of the regenerated_to relation

```
MATCH (actors:Actor)-[:REGENERATED_TO]-> (others)

RETURN actors.name, others.name
```

Query Example

Collect all nodes with the Character label which have the enemy_of relation with the Doctor. Check if they have a comes_from relation with nodes with a Planet label. Return the name of the planets along with the number of occurrences

```
MATCH (:Character{name:'Doctor'})<-[:ENEMY_OF]-
(:Character)-[:COMES_FROM]->(p:Planet)

RETURN p.name as Planet, count(p) AS Count
```

Query Example

Give me all the episodes the character Amy Pond and the Actor Matt Smith were in. List the enemies of the Doctor that were in that episode beside it.

```
MATCH (:Actor{name:'Matt Smith'}) -[:APPEARED_IN]->
(ep:Episode) <-[:APPEARED_IN]- (:Character{name:'Amy
Pond'}),

(ep) <-[:APPEARED_IN]-(enemies:Character) <-
[:ENEMY_OF]-(:Character{name:'Doctor'})

RETURN ep AS Episode, collect(enemies.name) AS
Enemies;
```

WHERE: Filters Result Set

```
MATCH
(:Actor{name:'Matt Smith'}) -[:PLAYED]->(c:Character)
RETURN c

MATCH
(a:Actor) -[:PLAYED]->(c:Character)
WHERE a.name = 'Matt Smith'
RETURN c
```

ORDER BY: Orders Result Set

```
MATCH
(a:Actor) –[:PLAYED]->(c:Character)
WHERE a.name = ‘Matt Smith’
RETURN c
ORDER BY c.name
```

ORDER BY: Orders Result Set

```
MATCH
(a:Actor) –[:PLAYED]->(c:Character)
WHERE a.name = ‘Matt Smith’
RETURN c
ORDER BY c.name, c.salary
```

ORDER BY: Orders Result Set

```
MATCH
(a:Actor) –[:PLAYED]->(c:Character)
WHERE a.name = ‘Matt Smith’
RETURN c
ORDER BY c.name DESC
```

Skip and Limit

```
MATCH
(:Actor{name:'Matt Smith'}) -[:PLAYED]->(c:Character)
RETURN c
LIMIT 10
SKIP 5
```

UNION: Glues Result Sets

```
MATCH (a:Actor)
RETURN a.name
UNION
MATCH (c:Character)
RETURN c.name
```

UNION: Glues Result Sets

```
MATCH (a:Actor)
RETURN a.name
UNION ALL
MATCH (c:Character)
RETURN c.name
```

WITH: Manipulate Result Set

```
MATCH
(a:Actor)
WITH a.name AS name, count(a) AS count
ORDER BY name
WHERE count > 10
RETURN name
```

Predicates

Return true or false for a given input

Input can be properties or patterns

Mostly used in WHERE clause

ALL, ANY, NONE, SINGLE, EXISTS

Predicates

```
MATCH
(a:Actor)
WHERE EXISTS ((a)-[:PLAYED]->())
RETURN a.name
```

Scalar Functions

Return a single value

LENGTH, TYPE, ID, COALESCE, HEAD, LAST, TIMESTAMP, TOINT, TOFLOAT, TOSTRING

Scalar Functions

```
MATCH
p = (:Actor)-[:PLAYED]->(:Character)
RETURN LENGTH(p)
```

Collection Functions

Return collections of 'things'

NODES, RELATIONSHIPS, LABELS

EXTRACT, FILTER, TAIL

RANGE, REDUCE

Collection Functions

```
MATCH
p = (:Actor)-[:PLAYED]->(:Character)
RETURN NODES(p)
```

Mathematical Functions

ABS
ACOS
ASIN
ATAN
COS
COT
DEGREES
EXP
FLOOR
ROUND
SQRT
Etc.

String Functions

STR
REPLACE
SUBSTRING
LEFT
RIGHT
LTRIM
RTRIM
TRIM
LOWER
UPPER
SPLIT

Advanced Syntax: Directionless Relationships

```
MATCH
(:Episode)-[:PREVIOUS]-(e:Episode)
RETURN e
```

Advanced Syntax: No Relationship Defined

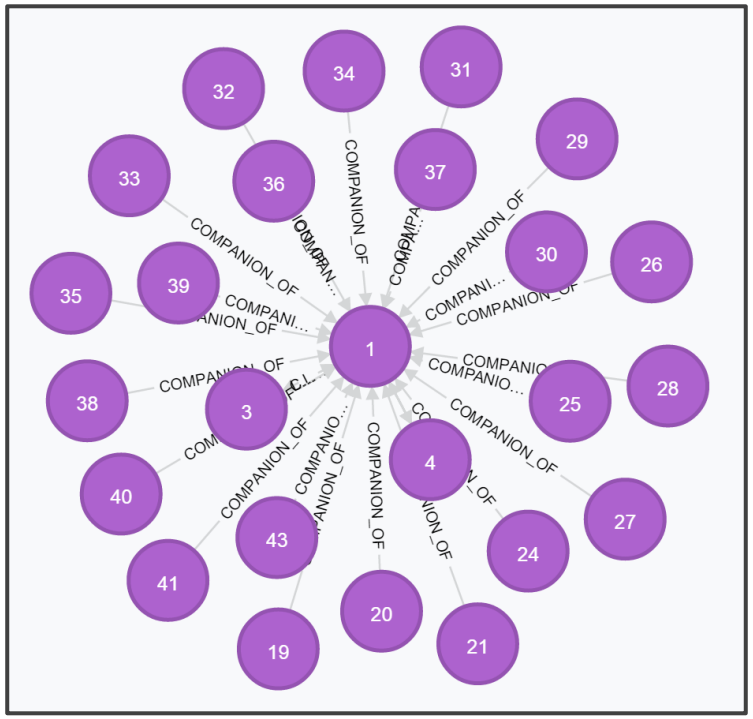
```
MATCH
(:Episode)-->(e:Episode)
RETURN e
```

Advanced Syntax: No Relationship Name

```
MATCH
(:Actor)-[]->()-[]->(p:Planet)
RETURN p
```

Advanced Syntax: Number of Hops

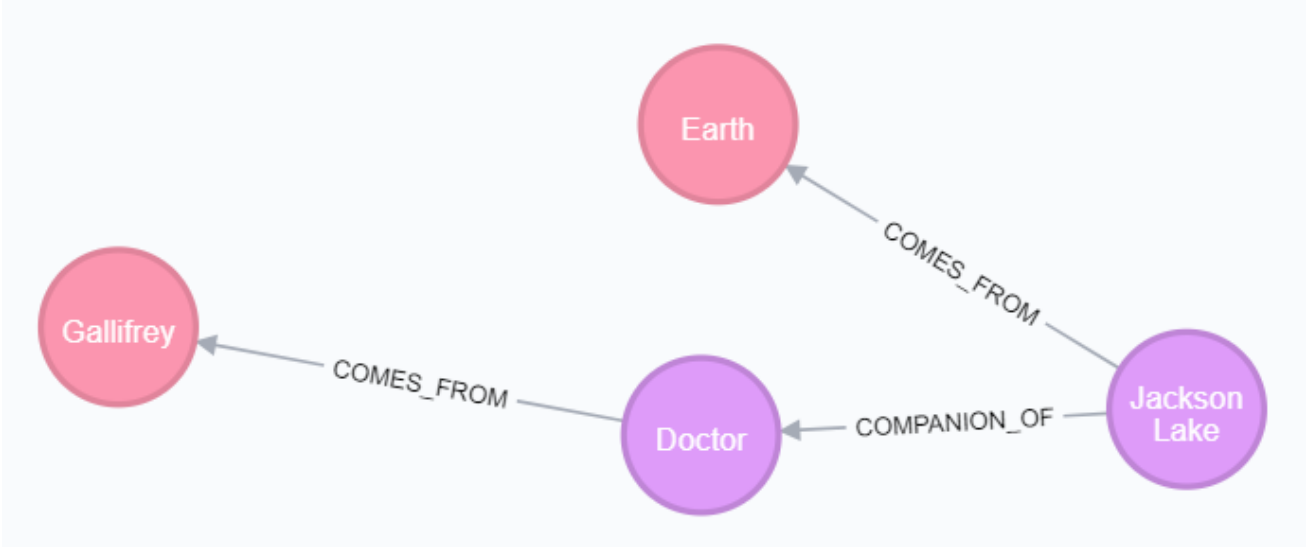
```
MATCH
(:Actor)-[*2]->(p:Planet)
RETURN p
```




```
MATCH
(c:Character)-
[:COMPANION_OF*1..2
]-(:Character)
RETURN c
```

Advanced Syntax: Shortest Path

```
MATCH (earth:Planet { name:"Earth" }),
(gallifrey:Planet { name:"Gallifrey" }),
p = SHORTESTPATH((earth)-[*..15]-(gallifrey))
RETURN p
```



Advanced Syntax: Optional MATCH

```
MATCH (a:Character)
OPTIONAL MATCH
(a)-[r:COMES_FROM]->()
RETURN r
```

Summary

Cypher is a powerful query language

Uses patterns to query data

Main keywords: MATCH and RETURN

Other keywords like WHERE

Built-in functions

Advanced syntax