

CMPINF 2110

Week 13

More Graph examples with Neo4j and Cypher

Dates and DateTime Neo4j

- Creating date and DateTime:
 - <https://neo4j.com/docs/cypher-manual/current/syntax/temporal/>
- Extracting DateTime components:
 - <https://neo4j.com/docs/cypher-manual/current/syntax/temporal/#cypher-temporal-accessing-components-temporal-instants>

Simple example to create a date and return it

The image displays two screenshots of the Neo4j Cypher Shell interface, demonstrating a simple query to create and return a date.

Top Screenshot:

- Query:** `neo4j$ RETURN date('2013-11-14') AS theday;`
- Result Table:** The table has a header `theday`. It contains one record with the value `"2013-11-14"`.
- Performance:** Started streaming 1 records after 6 ms and completed after 7 ms.

Bottom Screenshot:

- Query:** `neo4j$ RETURN date('2013-11-14')`
- Result Table:** The table has a header `date('2013-11-14')`. It contains one record with the value `"2013-11-14"`.
- Performance:** Started streaming 1 records after 2 ms and completed after 4 ms.

Multiple ways to create a date, but once created can extract datetime components

The image displays two screenshots of a SQL query editor, likely from a data science or analytics platform, demonstrating different methods to create a date and then extract its components.

Top Screenshot:

```
1 WITH date('2013-11-14') AS theday
2 RETURN theday.year, theday.month, theday.day, theday.dayOfWeek;
```

	theday.year	theday.month	theday.day	theday.dayOfWeek
1	2013	11	14	4

Started streaming 1 records after 9 ms and completed after 10 ms.

Bottom Screenshot:

```
1 WITH date({year: 2013, month: 11, day: 14}) AS theday
2 RETURN theday.year, theday.month, theday.day, theday.dayOfWeek;
```

	theday.year	theday.month	theday.day	theday.dayOfWeek
1	2013	11	14	4

Started streaming 1 records after 11 ms and completed after 15 ms.

PAW Patrol example – create Ryder Node as Label Character

The screenshot displays a Neo4j Cypher query editor interface. At the top, a code editor contains the following query:

```
1 CREATE (c:Character {name:'Ryder'})
2 RETURN *
```

Below the code editor, a toolbar includes icons for execution (play), favorites (star), download, share, zoom in, zoom out, and close. The main workspace shows a graph visualization with a single grey circular node labeled "Ryder". Above the graph, a status bar indicates *** (1)** and **Character(1)**. On the left side, a sidebar contains icons for different views: Graph (selected), Table, Text, and Code. At the bottom of the interface, a status message reads "Displaying 1 nodes, 0 relationships."

Create all Pups as Character Label nodes

```
1 //create all of the pups
2 CREATE (:Character {name:'Skye'}),
3       (:Character {name:'Zuma'}),
4       (:Character {name:'Chase'}),
5       (:Character {name:'Marshall'}),
6       (:Character {name:'Rubble'}),
7       (:Character {name:'Rocky'});
```

Added 6 labels, created 6 nodes, set 6 properties, completed after 8 ms.

Table

Code

Added 6 labels, created 6 nodes, set 6 properties, completed after 8 ms.

Create the PAW Patrol team label

```
1 //create the PAW Patrol team node
2 CREATE (:Team {teamName:'PAW Patrol'});
```

Added 1 label, created 1 node, set 1 property, completed after 11 ms.

Table

Code

Added 1 label, created 1 node, set 1 property, completed after 11 ms.

Create the relationship between the pups and the team, every pup as at least 1 role on the team

```
1 //create relationship between pups and the team
2 MATCH (pup:Character {name:'Skye'})
3 MATCH (team:Team {teamName:'PAW Patrol'})
4 CREATE (pup)-[r:HAS_ROLE {type:'air'}]→(team);
```



Table

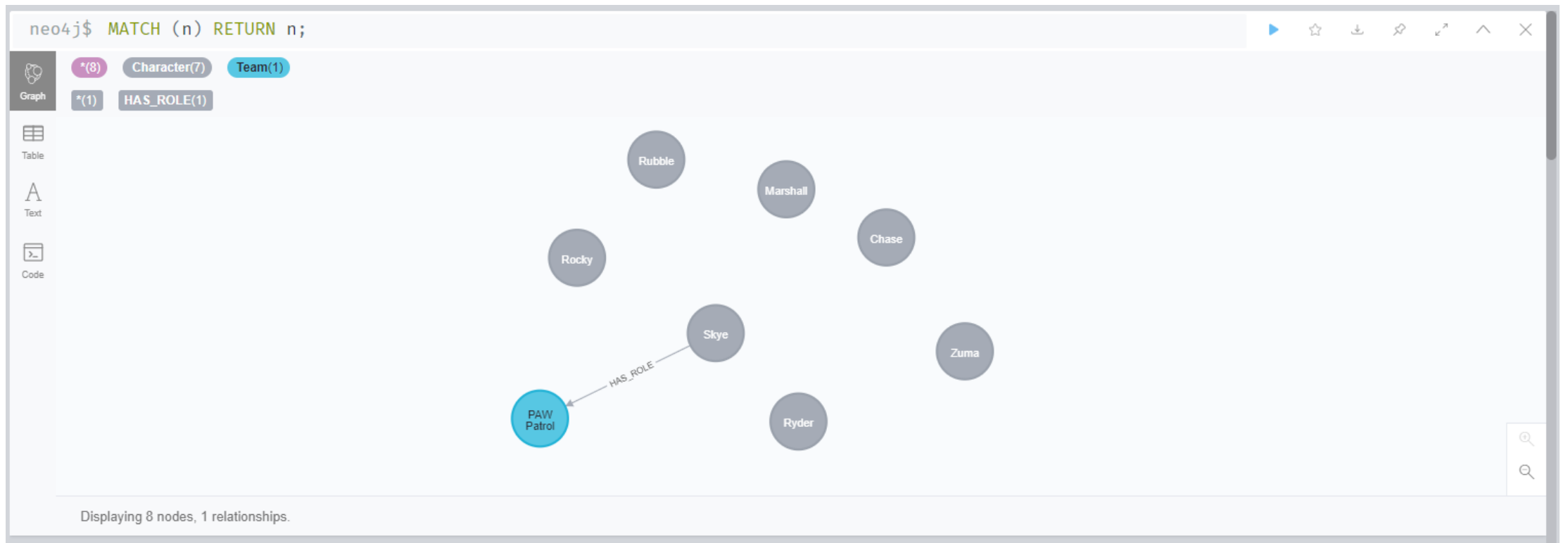


Code

Set 1 property, created 1 relationship, completed after 15 ms.

Set 1 property, created 1 relationship, completed after 15 ms.

Show the relationship



Some pups have multiple roles! This allows us to have multiple relationships between nodes

The screenshot displays two separate Cypher queries in a console interface, each with a 'Table' and 'Code' view toggle on the left. The first query creates a relationship of type 'repairs' between a pup named Rocky and a team named PAW Patrol. The second query creates a relationship of type 'recycling' between the same pup and team. Both queries are successful, as indicated by the status messages below each code block.

```
1 //create relationship between pups and the team
2 MATCH (pup:Character {name:'Rocky'})
3 MATCH (team:Team {teamName:'PAW Patrol'})
4 CREATE (pup)-[r:HAS_ROLE {type:'repairs'}]→(team);
```

Set 1 property, created 1 relationship, completed after 8 ms.

```
1 //create relationship between pups and the team
2 MATCH (pup:Character {name:'Rocky'})
3 MATCH (team:Team {teamName:'PAW Patrol'})
4 CREATE (pup)-[r:HAS_ROLE {type:'recycling'}]→(team);
```

Set 1 property, created 1 relationship, completed after 7 ms.

Create the roles for Marshall



The screenshot displays two instances of the Neo4j Cypher query editor. Each instance features a code editor on the left with a line number margin, a toolbar on the right with icons for execution, saving, and undo/redo, and a results pane on the right. The results pane includes a 'Table' view icon and a 'Code' view icon. The status bar at the bottom of each editor indicates the execution result.

Top Query:

```
1 //create relationship between pups and the team
2 MATCH (pup:Character {name:'Marshall'})
3 MATCH (team:Team {teamName:'PAW Patrol'})
4 CREATE (pup)-[r:HAS_ROLE {type:'paramedic'}]-(team);
```

Set 1 property, created 1 relationship, completed after 7 ms.

Bottom Query:

```
1 //create relationship between pups and the team
2 MATCH (pup:Character {name:'Marshall'})
3 MATCH (team:Team {teamName:'PAW Patrol'})
4 CREATE (pup)-[r:HAS_ROLE {type:'firefighter'}]-(team);
```

Set 1 property, created 1 relationship, completed after 8 ms.

Create the roles for Chase

The image shows a screenshot of a Cypher query editor interface with two panels. Each panel contains a code editor, a table view, and a status bar.

Top Panel:

```
1 //create relationship between pups and the team
2 MATCH (pup:Character {name:'Chase'})
3 MATCH (team:Team {teamName:'PAW Patrol'})
4 CREATE (pup)-[r:HAS_ROLE {type:'spy'}]→(team);
```

Table view: Set 1 property, created 1 relationship, completed after 7 ms.

Bottom Panel:

```
1 //create relationship between pups and the team
2 MATCH (pup:Character {name:'Chase'})
3 MATCH (team:Team {teamName:'PAW Patrol'})
4 CREATE (pup)-[r:HAS_ROLE {type:'police'}]→(team);
```

Table view: Set 1 property, created 1 relationship, completed after 7 ms.

Create the role for Zuma

```
1 //create relationship between pups and the team
2 MATCH (pup:Character {name:'Zuma'})
3 MATCH (team:Team {teamName:'PAW Patrol'})
4 CREATE (pup)-[r:HAS_ROLE {type:'aquatic'}]→(team);
```



Table



Code

Set 1 property, created 1 relationship, completed after 7 ms.

Set 1 property, created 1 relationship, completed after 7 ms.

Create the role for Rubble

```
1 //create relationship between pups and the team
2 MATCH (pup:Character {name:'Rubble'})
3 MATCH (team:Team {teamName:'PAW Patrol'})
4 CREATE (pup)-[r:HAS_ROLE {type:'construction'}]→(team);
```

Table

Code

Set 1 property, created 1 relationship, completed after 7 ms.

Set 1 property, created 1 relationship, completed after 7 ms.

Create the role for Ryder

```
1 //create relationship between Ryder and the team
2 MATCH (character:Character {name:'Ryder'})
3 MATCH (team:Team {teamName:'PAW Patrol'})
4 CREATE (character)-[r:HAS_ROLE {type:'manager'}]-(team);
```



Table

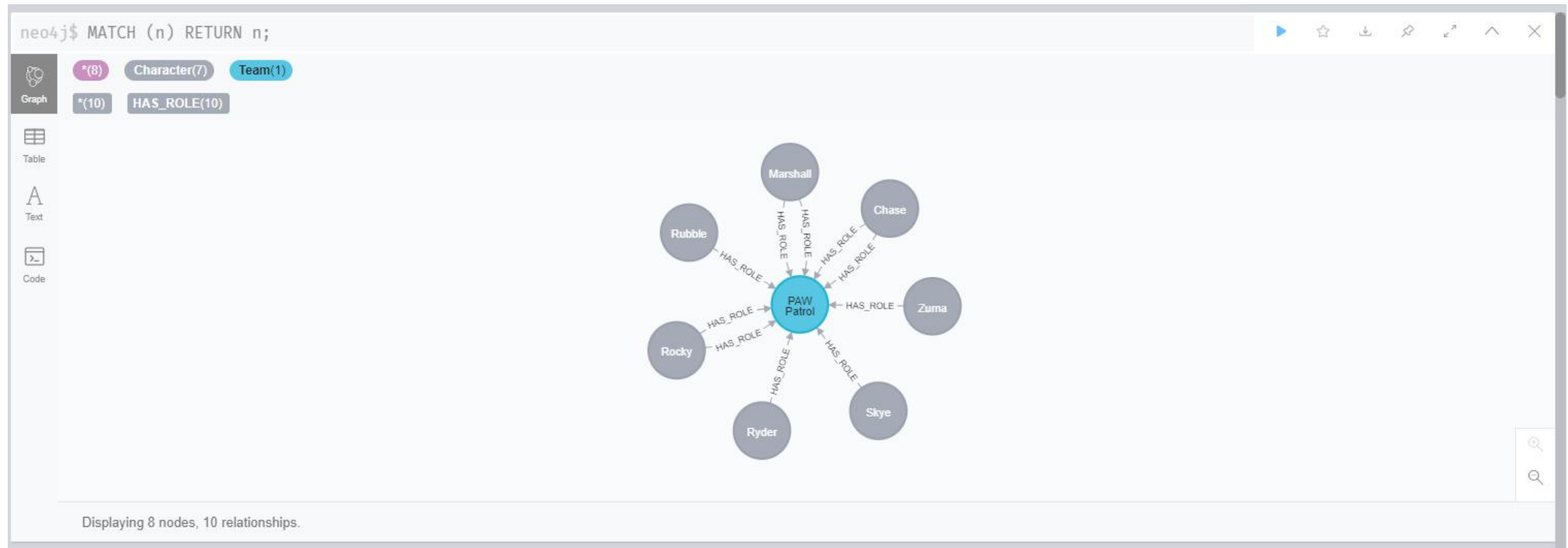


Code

Set 1 property, created 1 relationship, completed after 7 ms.

Set 1 property, created 1 relationship, completed after 7 ms.

The PAW patrol!



Create episode 1

```
1 //add in episode node
2 CREATE (e:Episode {episodeID: 1, airDate: date('2013-11-14')});
```

Added 1 label, created 1 node, set 2 properties, completed after 20 ms.

Table

Code

Added 1 label, created 1 node, set 2 properties, completed after 20 ms.

Create season 1

```
1 //create season
2 CREATE (s:Season {seasonID:1});
```

Added 1 label, created 1 node, set 1 property, completed after 10 ms.

Table

Code

Added 1 label, created 1 node, set 1 property, completed after 10 ms.

Create segments a and b of episode 1

```
1 //create segments a and b of episode 1
2 CREATE (s:Segment {segmentID: 'a', title:'Pups make a splash'}),
3      (s2:Segment {segmentID: 'b', title:'Pups Fall Festival'});
```



Table



Code

Added 2 labels, created 2 nodes, set 4 properties, completed after 11 ms.

Added 2 labels, created 2 nodes, set 4 properties, completed after 11 ms.

Relate segment to episode

```
1 //assign the segment to an episode
2 MATCH (e:Episode {episodeID: 1})
3 MATCH (s:Segment {segmentID: 'a'})
4 CREATE (s)-[r:SEGMENT_OF]→(e);
```



Table



Code

Created 1 relationship, completed after 12 ms.

Created 1 relationship, completed after 12 ms.

Relate segment to episode

```
1 //assign the segment to an episode
2 WITH [1] AS episode_use
3 MATCH (e:Episode)
4 MATCH (s:Segment {segmentID: 'b'})
5 WHERE e.episodeID IN episode_use
6 CREATE (s)-[r:SEGMENT_OF]→(e);
```



Created 1 relationship, completed after 14 ms.



Created 1 relationship, completed after 14 ms.

Without using the segment title we need a PATH to identify a specific segment!

```
1 //relate pup helping in an episode
2 MATCH (e:Episode {episodeID: 1})--(s:Segment {segmentID: 'a'})
3 MATCH (p:Character {name:'Skye'})
4 CREATE (p)-[r:RESPONDS]→(s);
```

Created 1 relationship, completed after 18 ms.

Table

Code

Created 1 relationship, completed after 18 ms.

Create a relationship for Marshal and segment a in episode 3

```
1 //create relationship between pup responding in a segment of an episode
2 MATCH (season:Season {seasonID: 1})--(episode:Episode {episodeID: 3})--(segment:Segment {segmentID: 'a'})
3 MATCH (pup:Character {name:'Marshall'})
4 CREATE (pup)-[r:RESPONDS {how:'backup'}]→(segment)
```



Table



Code

Set 1 property, created 1 relationship, completed after 7 ms.

Set 1 property, created 1 relationship, completed after 7 ms.

Network graph

