WEEK 2

Introduction to Graph database(NOSQL)

Name : Suhan B Revankar
SRN : PES2UG19CS412
Section : G SECTION

Problem Statement	Use NEO4j and create a sample graph database and perform the following operations on it 1. Create node with varying fields. 2. Add properties to node 3. Add relationships between the nodes . 4. Update an attribute value of the node 5. Retrieve and delete nodes, relationship

1. Create node and relationships

The CREATE clause is used to create nodes and relationships.

1.Create a single node :
Syntax: create (n) // create a single node without label

2. Create multiple nodes
Syntax: create(n),(m)

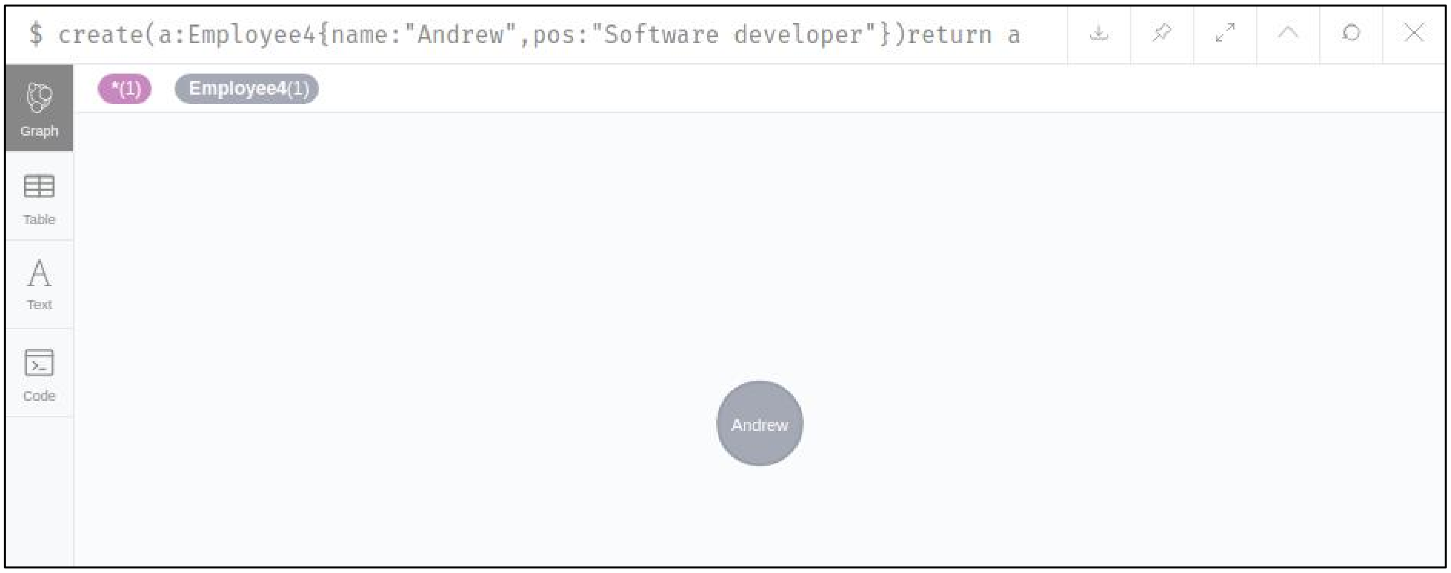
3. Create a node with a label Syntax:
create(n:lable name) Ex: CREATE
(n:Person)

Use match(n) return(n) to view the nodes

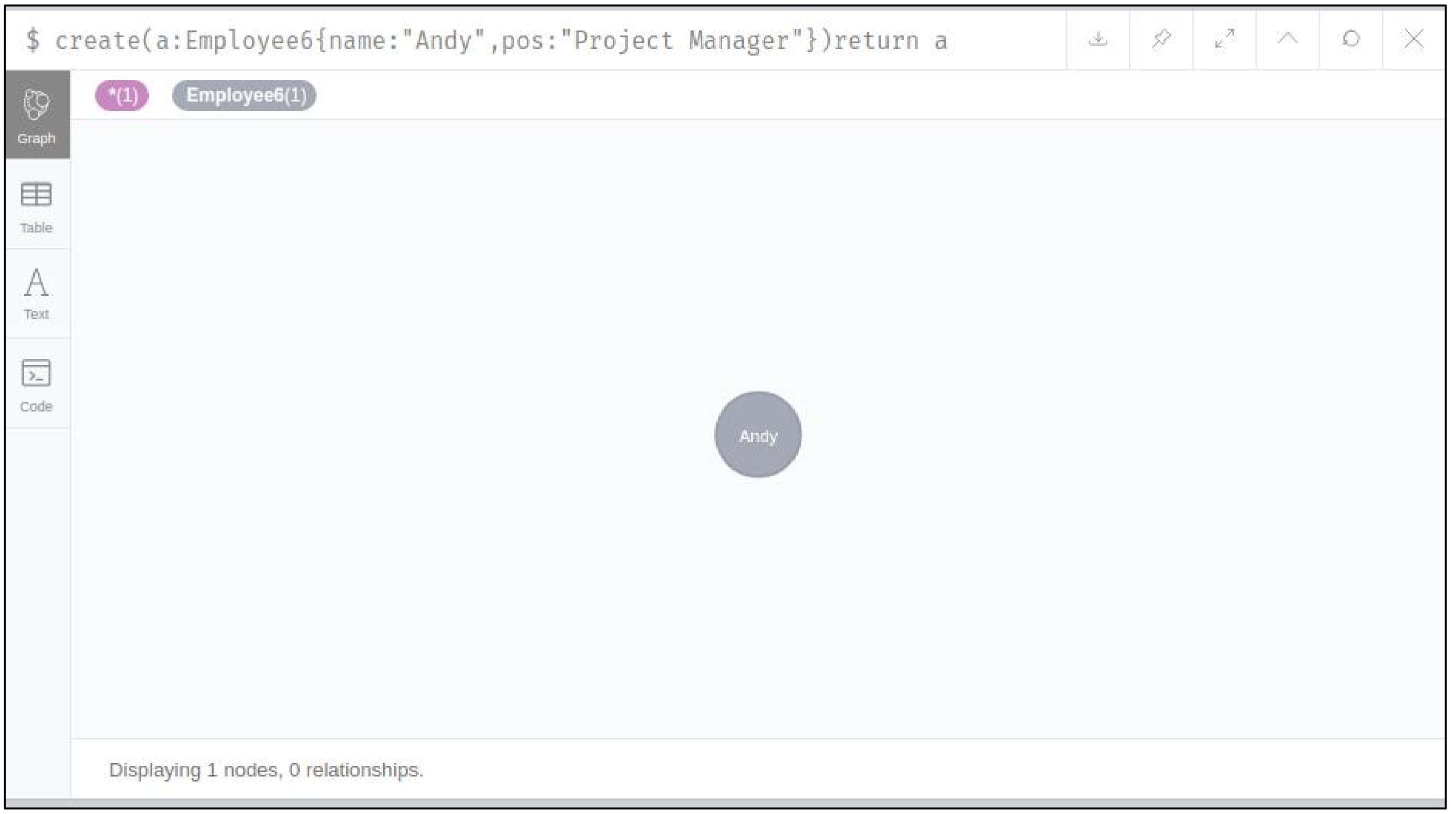
4. Create a node with multiple labels
To add labels when creating a node, use the syntax below. In this case, we add two labels. Ex:
CREATE (n:Person:Swedish)

5. Create node and add labels and properties
When creating a new node with labels, you can add properties at the same time Syntax:
Create(n:lablename {properties and values});
Ex: CREATE (n:Person {name: 'Andy', title: 'Developer'})

6. create nodes with parameters as properties
Syntax: 1. Define the property with parameter name. below example props is the parameter name
Ex:
```
 "props" : {
   "name" : "Andy",
   "position" : "Developer"
 }
}
```
2 add the parameter using the create clause
```
CREATE (n:Person $props)
   RETURN n
```

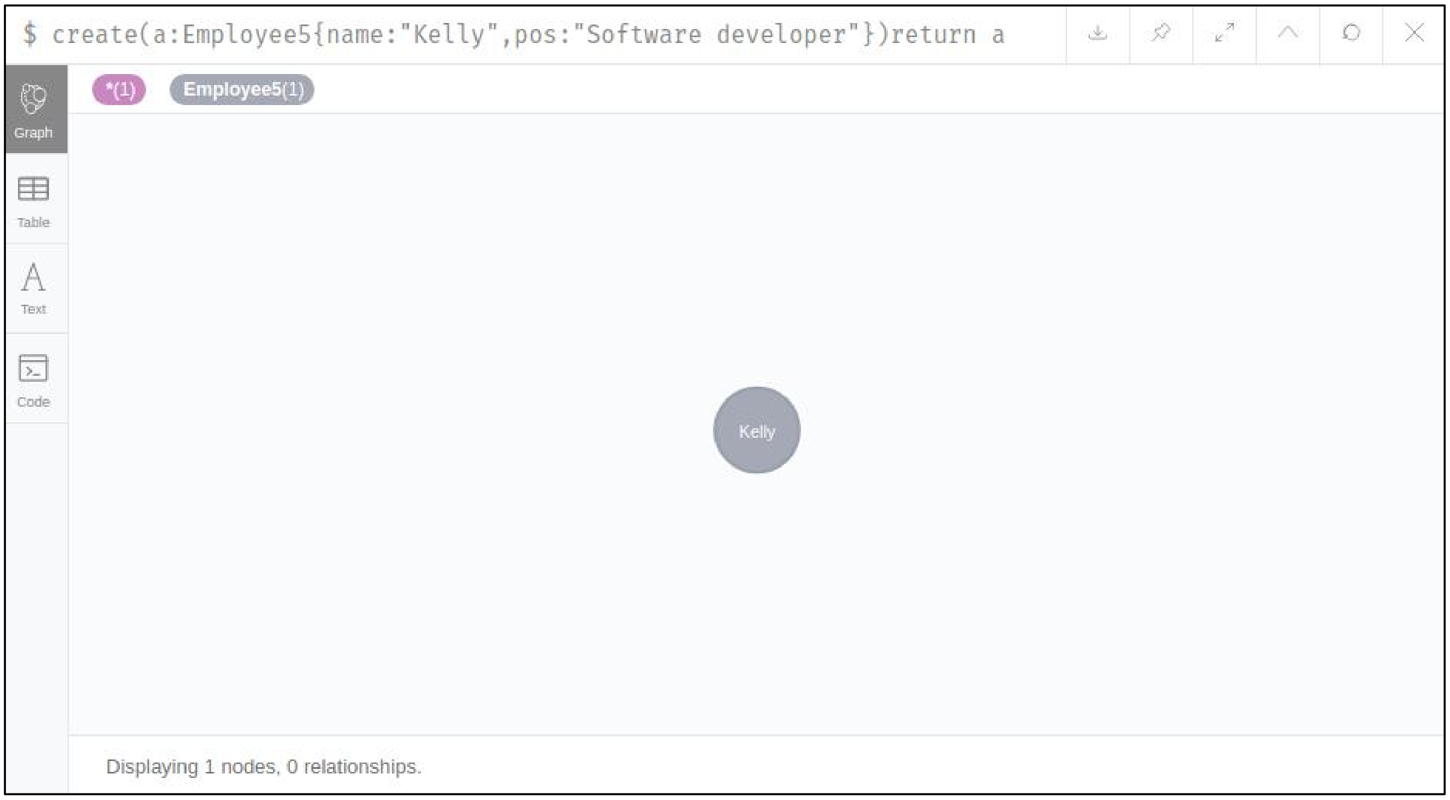


```
$ create(a:Employee4{name:"Andrew",pos:"Software developer"})return a
```



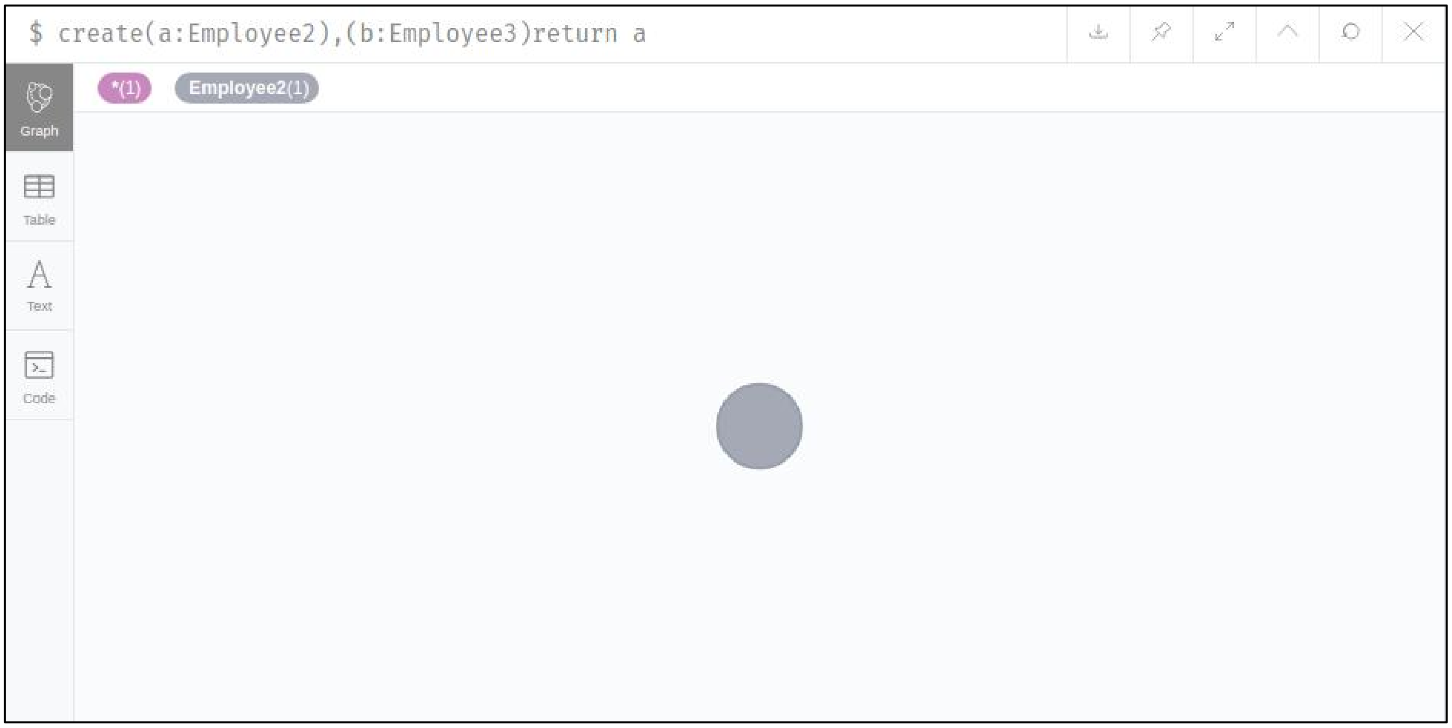
```
$ create(a:Employee6{name:"Andy",pos:"Project Manager"})return a
```

```
Displaying 1 nodes, 0 relationships.
```

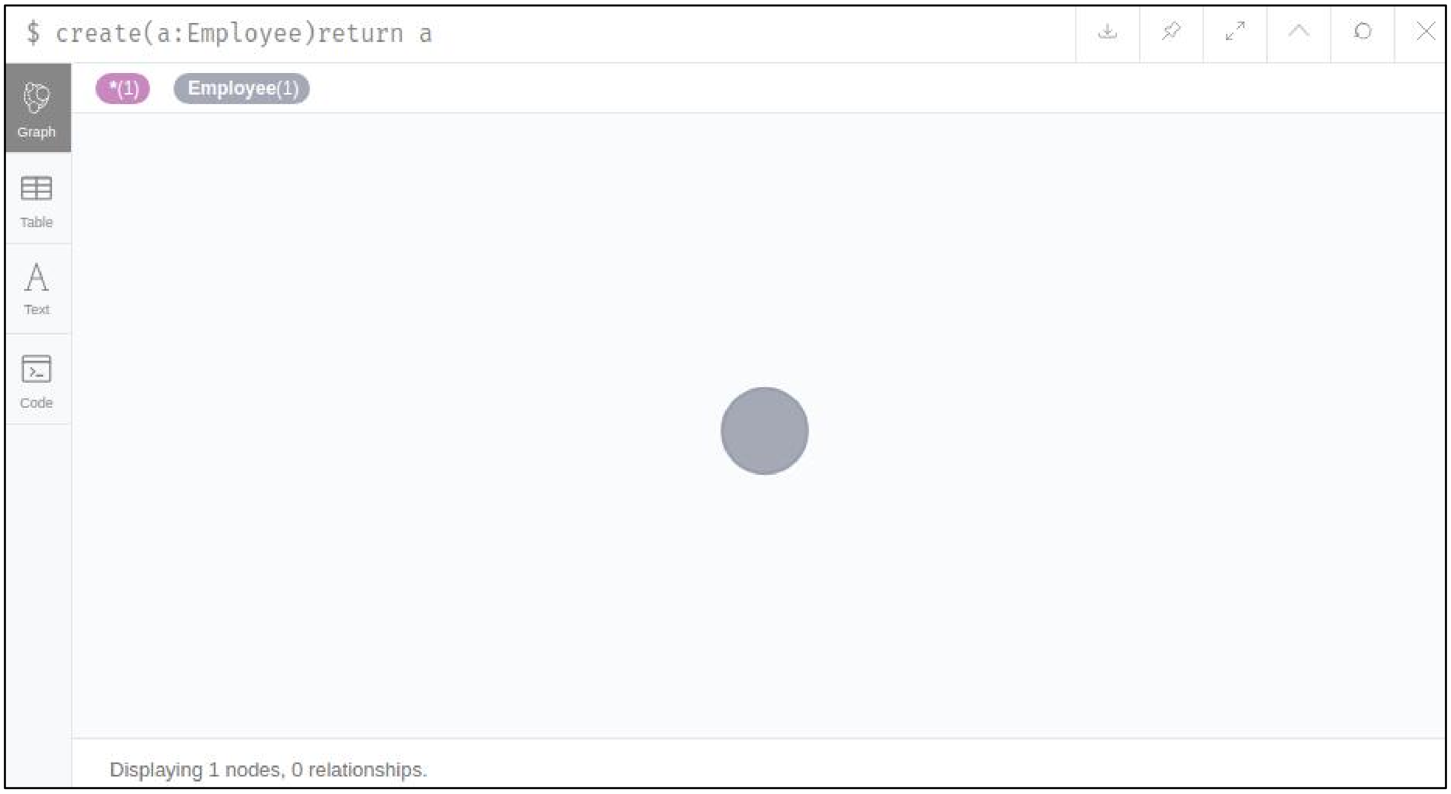


```
$ create(a:Employee5{name:"Kelly",pos:"Software developer"})return a
```

Displaying 1 nodes, 0 relationships.



```
$ create(a:Employee2),(b:Employee3)return a
```



```
$ create(a:Employee)return a
```

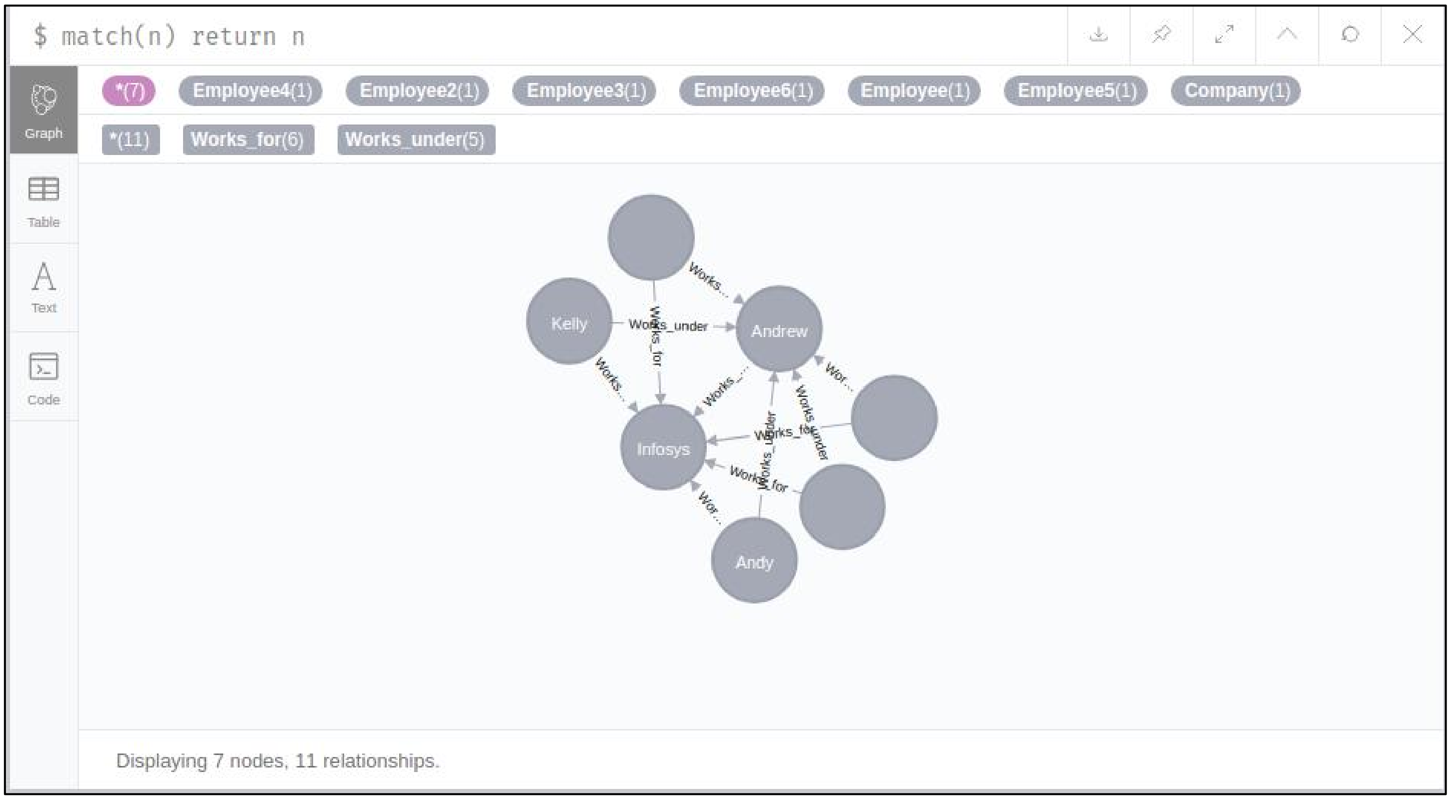
Displaying 1 nodes, 0 relationships.

2. Create Relationships between the nodes

```
Syntax:
Match
(node1) ,(node2)
Where condition
Create (node1) [relation type ] ->(node2)

Ex:
Match(u:university),(p:Person)
Where p.name=’x’ and
u.name=’pes’ Create(p)-
[stu:studiedAT] -> (u)
```

Creates the relationship Studiedat between person x and pes university Use match(n) return(n) to see the result

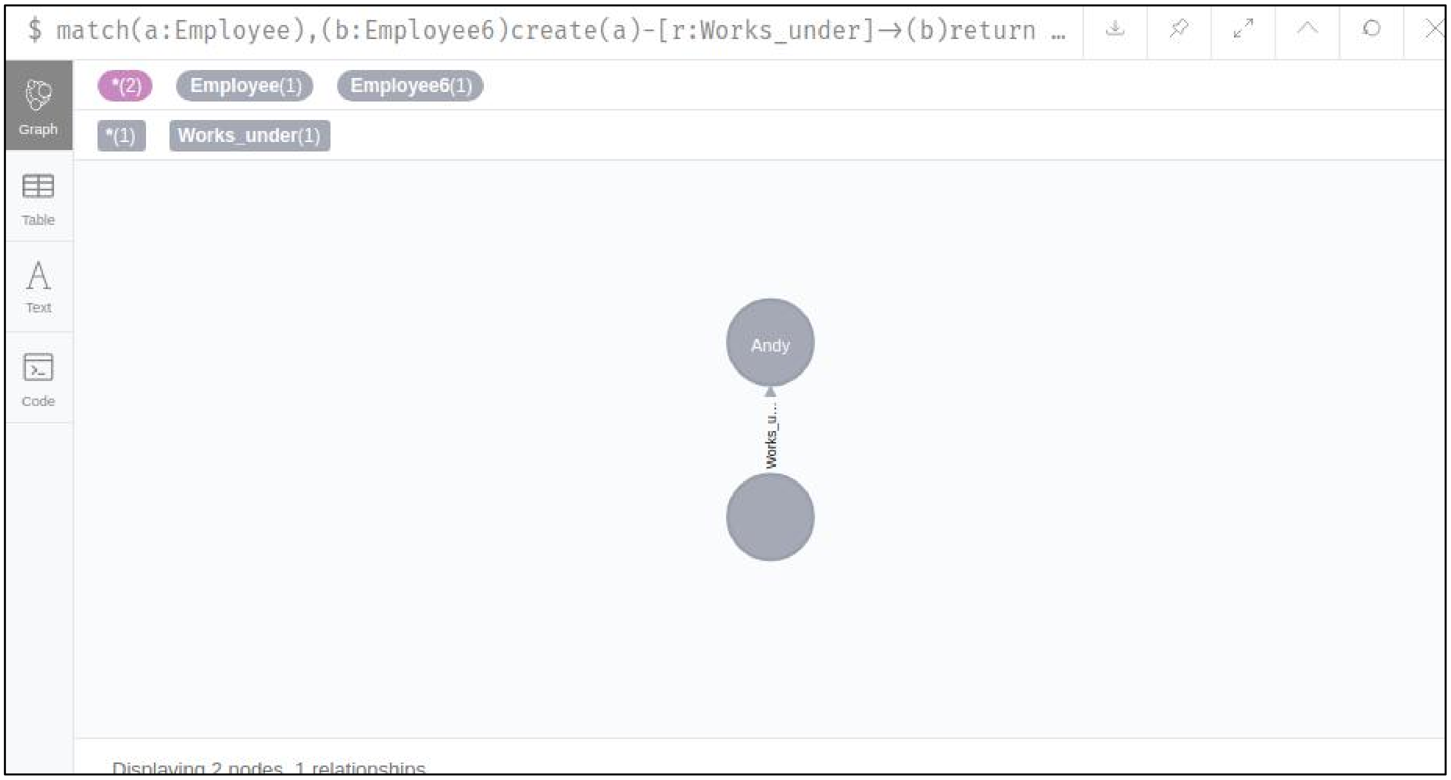


```
$ match(n) return n
```

*(7) Employee4(1) Employee2(1) Employee3(1) Employee6(1) Employee(1) Employee5(1) Company(1)

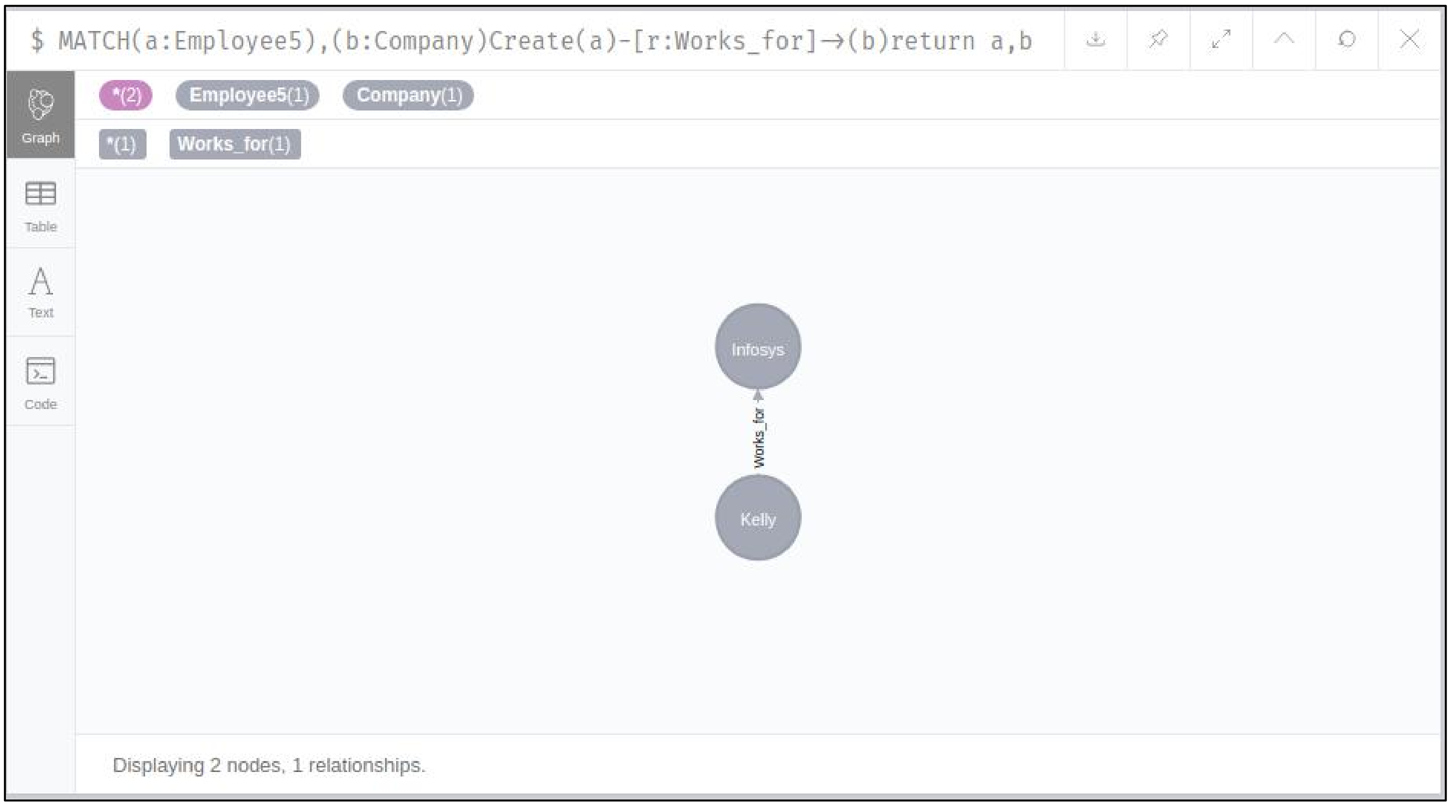
*(11) Works_for(6) Works_under(5)

Displaying 7 nodes, 11 relationships.



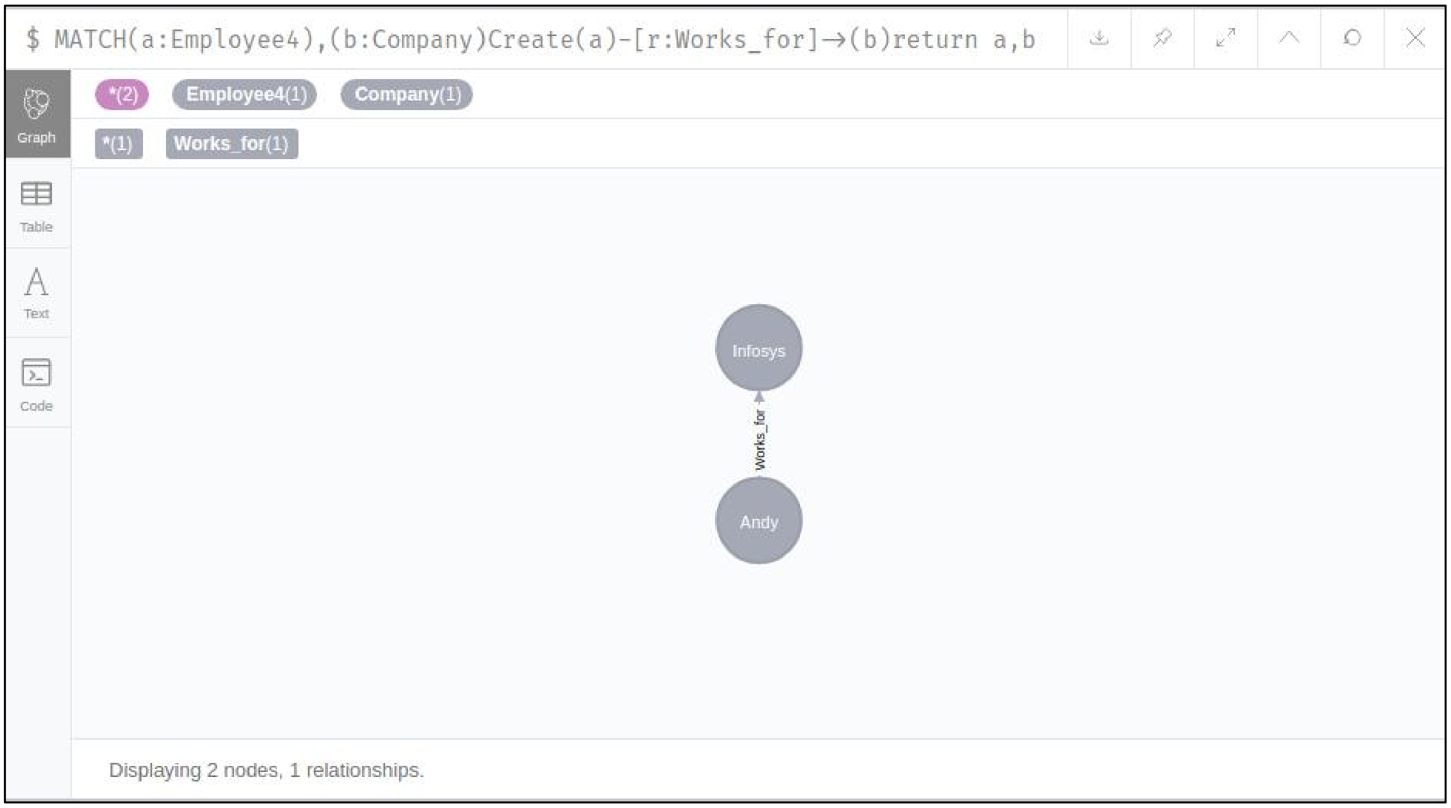
```
$ match(a:Employee),(b:Employee6)create(a)-[r:Works_under]->(b)return …
```

Displaying 2 nodes, 1 relationships



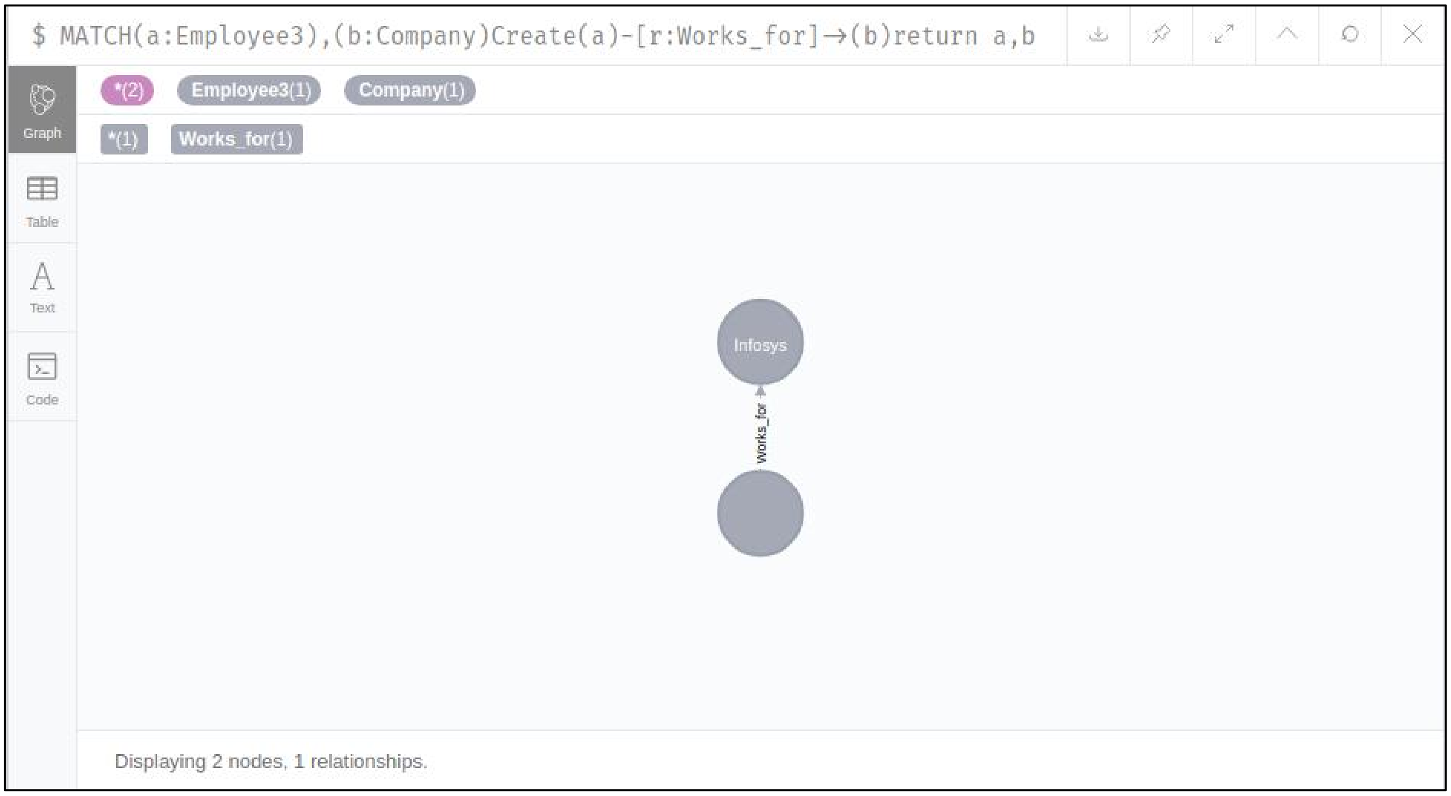
```
$ MATCH(a:Employee5),(b:Company)Create(a)-[r:Works_for]->(b)return a,b
```

Displaying 2 nodes, 1 relationships.



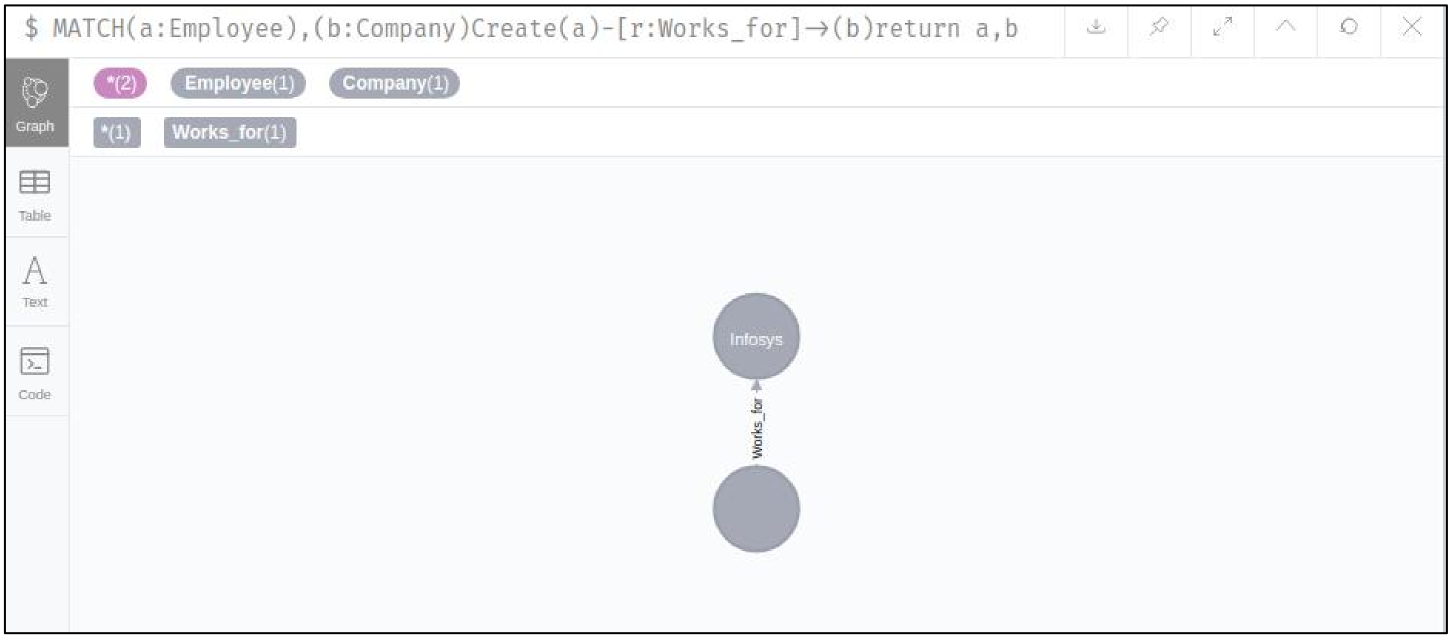
```
$ MATCH(a:Employee4),(b:Company)Create(a)-[r:Works_for]→(b)return a,b
```

Displaying 2 nodes, 1 relationships.



```
$ MATCH(a:Employee3),(b:Company)Create(a)-[r:Works_for]→(b)return a,b
```

Displaying 2 nodes, 1 relationships.



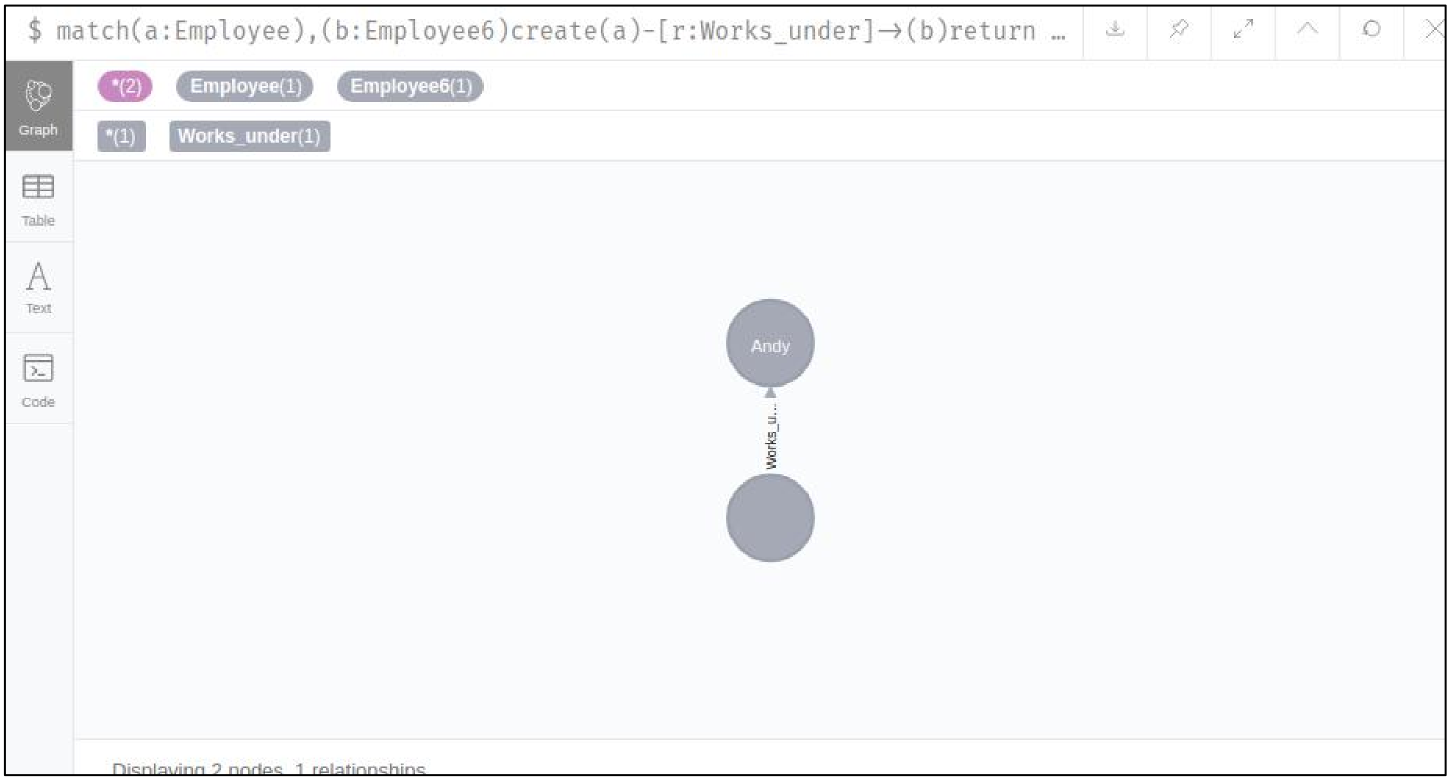
```
$ MATCH(a:Employee),(b:Company)Create(a)-[r:Works_for]→(b)return a,b
```

3. Read nodes and attributes (Node finding)

1. Get all nodes
By just specifying a pattern with a single node and no labels, all nodes in the graph will be returned**. Match(n) return(n)**

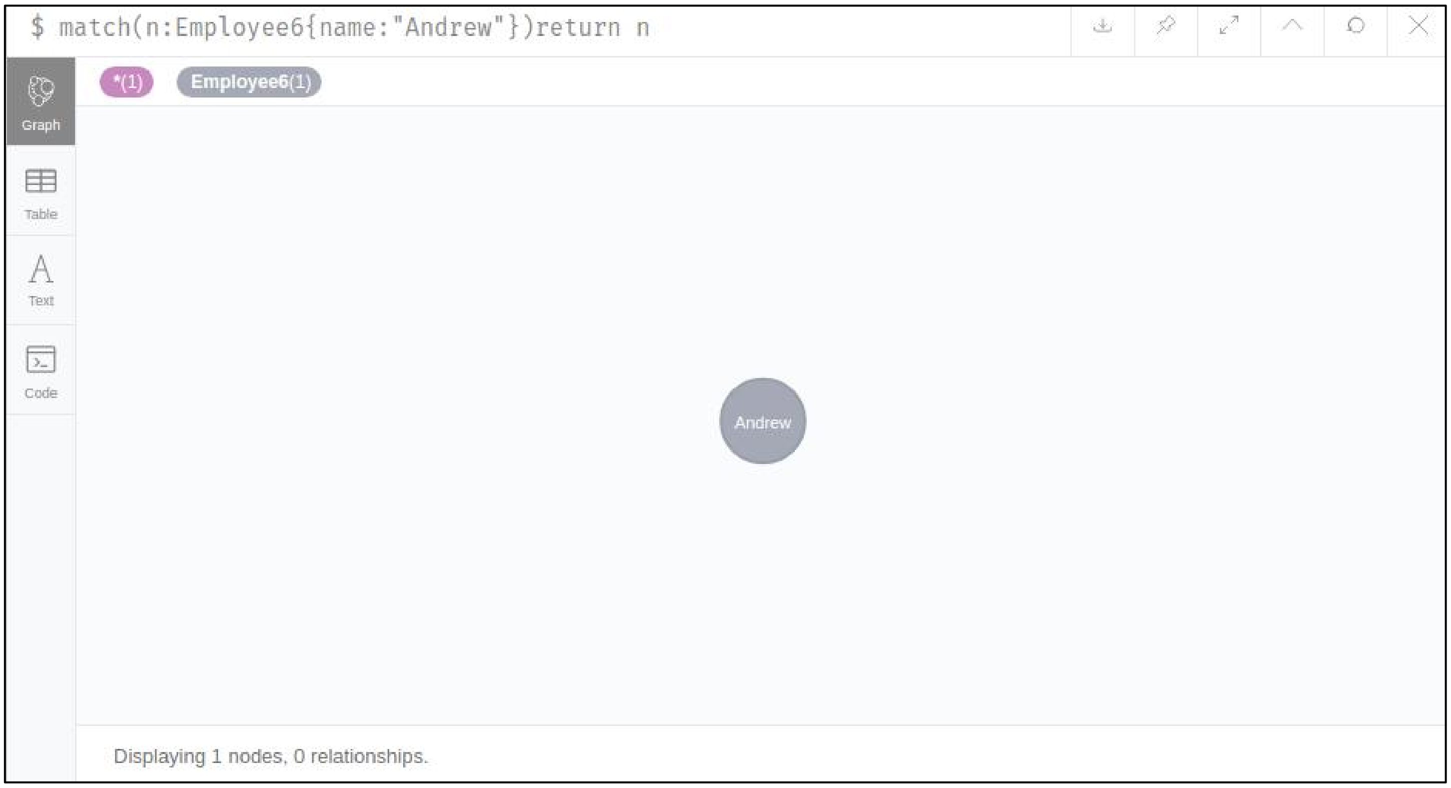
2. Get all nodes with a label
Getting all nodes with a label on them is done with a single node pattern where the node has a label on it.
MATCH
(movie:Movie)
RETURN movie.title
Returns all the movies in the database.

3. Related nodes
The symbol -- means related to, without regard to type or direction of the relationship.
MATCH (director {name: 'Oliver Stone'})--(movie)
RETURN movie.title
Returns all the movies directed by 'Oliver Stone'.



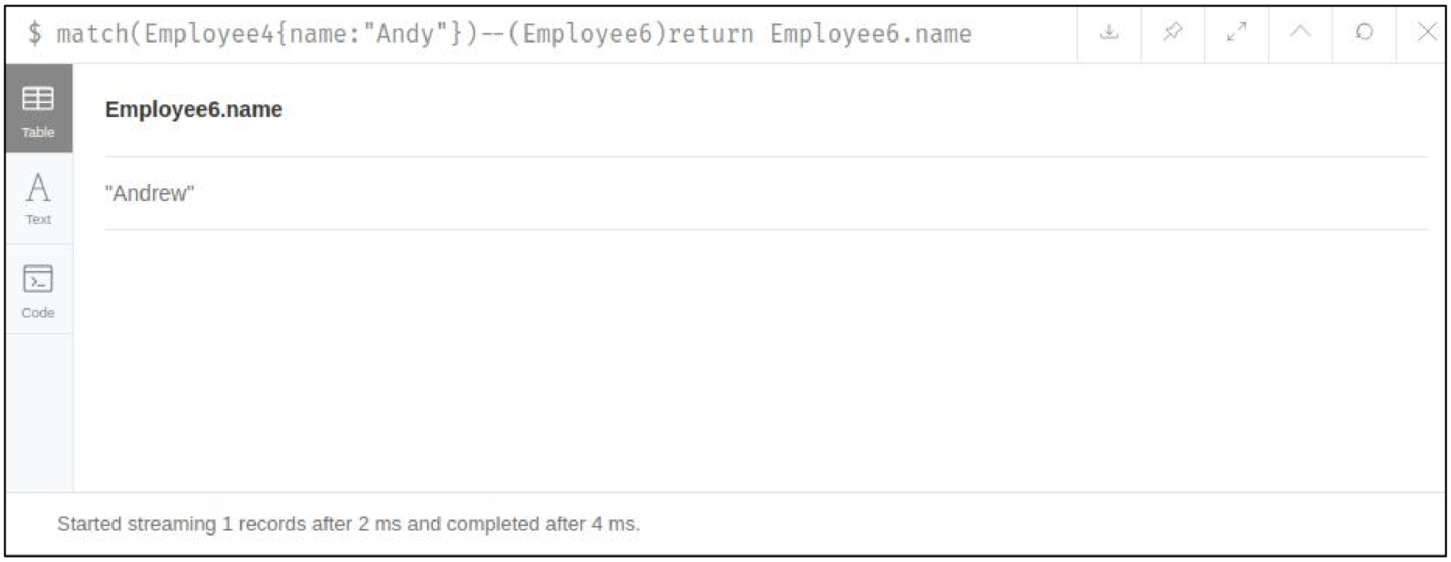
```
$ match(a:Employee),(b:Employee6)create(a)-[r:Works_under]→(b)return …
```

Displaying 2 nodes, 1 relationships



```
$ match(n:Employee6{name:"Andrew"})return n
```

Displaying 1 nodes, 0 relationships.



```
$ match(Employee4{name:"Andy"})--(Employee6)return Employee6.name
```

Employee6.name
"Andrew"

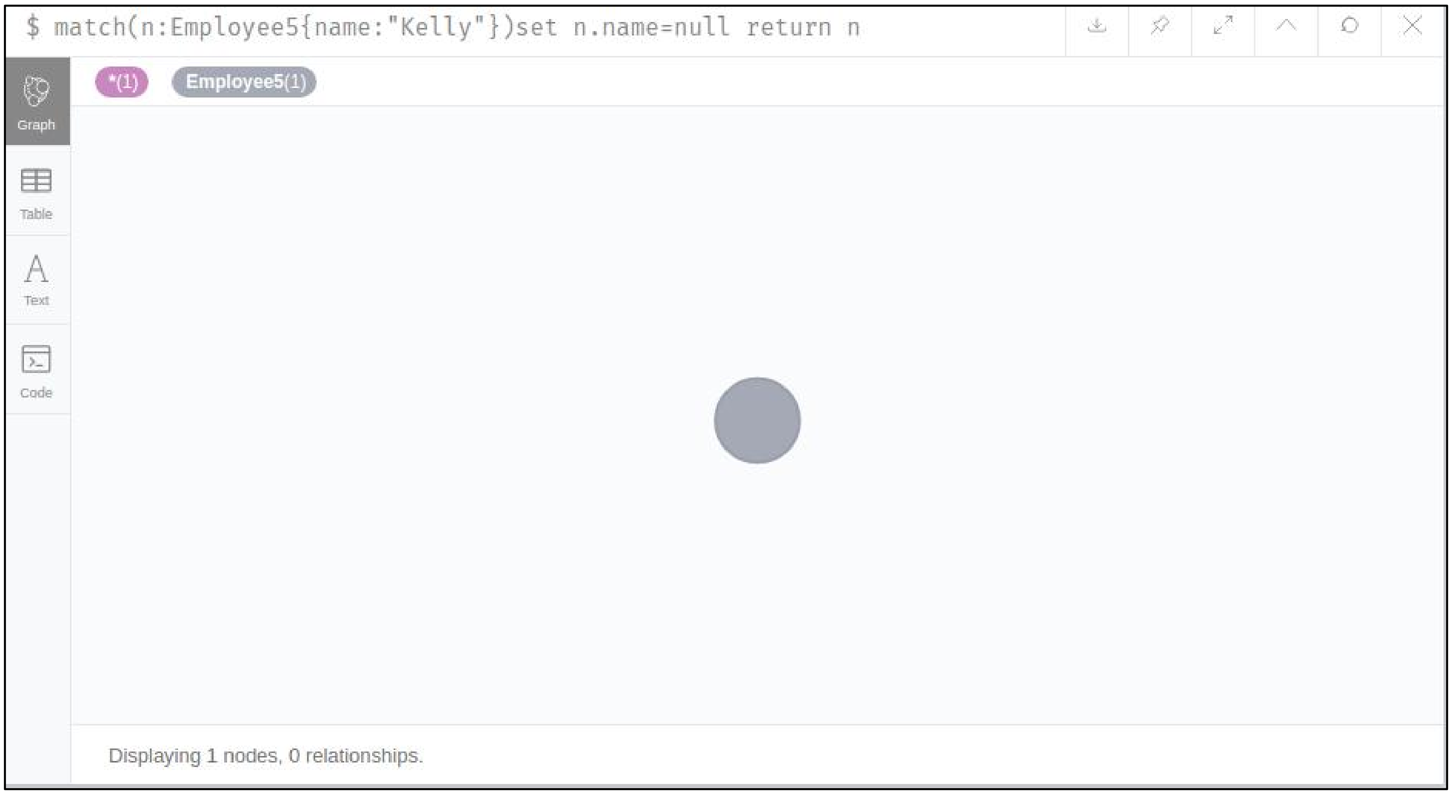
Started streaming 1 records after 2 ms and completed after 4 ms.

4. Update or set a value

Syntax:

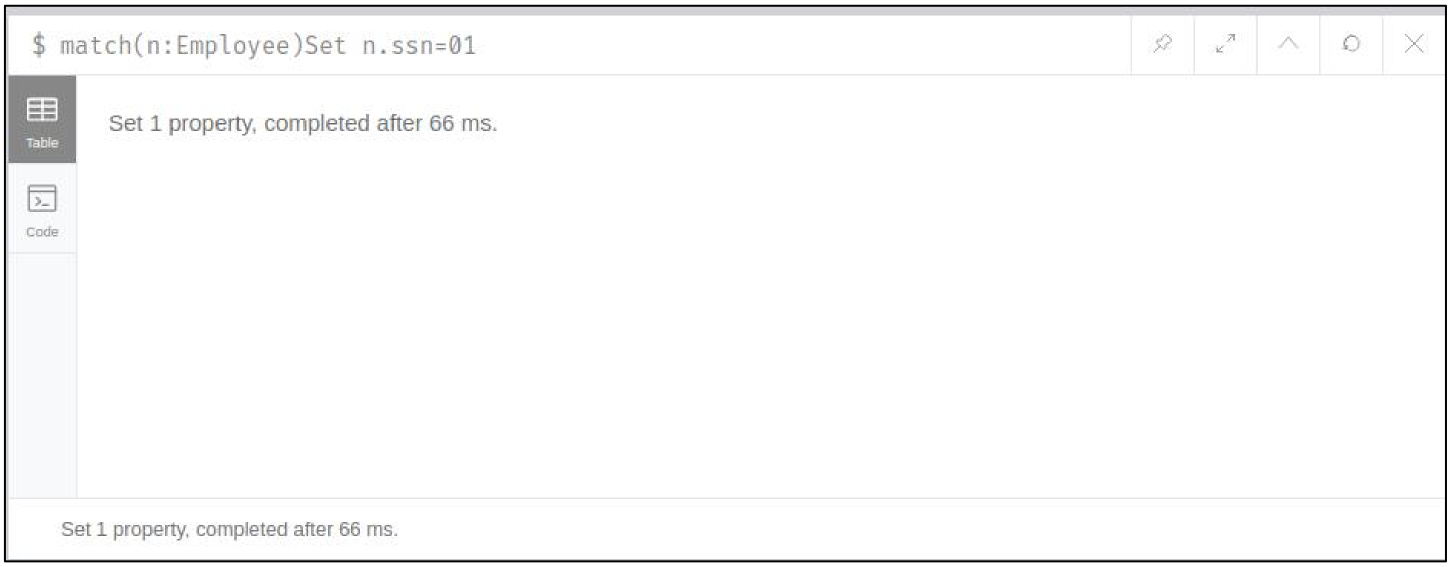
MATCH (n:Node)

Set n. propertyvalue = ‘newvalue’



```
$ match(n:Employee5{name:"Kelly"})set n.name=null return n
```

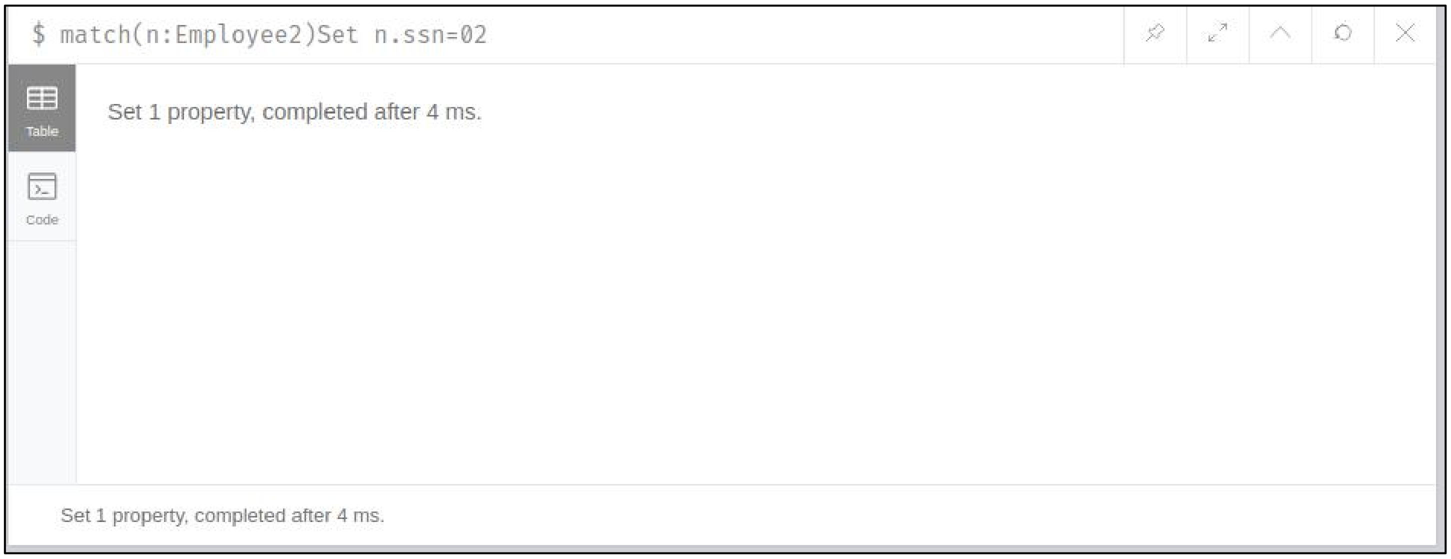
Displaying 1 nodes, 0 relationships.



```
$ match(n:Employee)Set n.ssn=01
```

Set 1 property, completed after 66 ms.

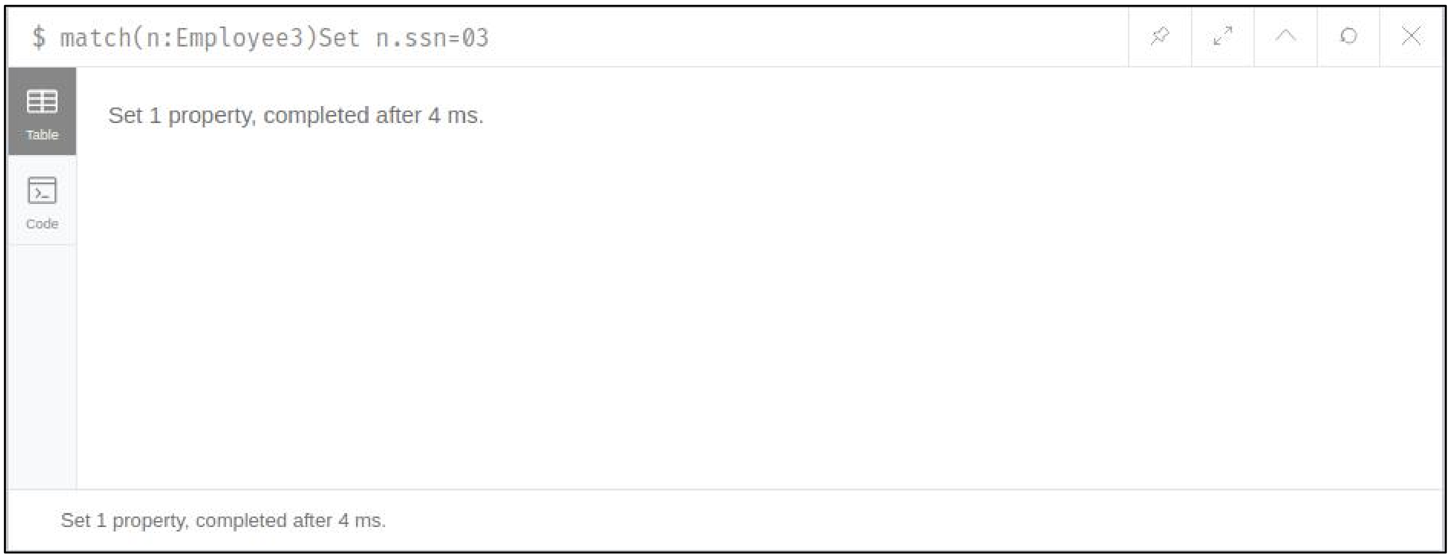
Set 1 property, completed after 66 ms.



```
$ match(n:Employee2)Set n.ssn=02
```

Set 1 property, completed after 4 ms.

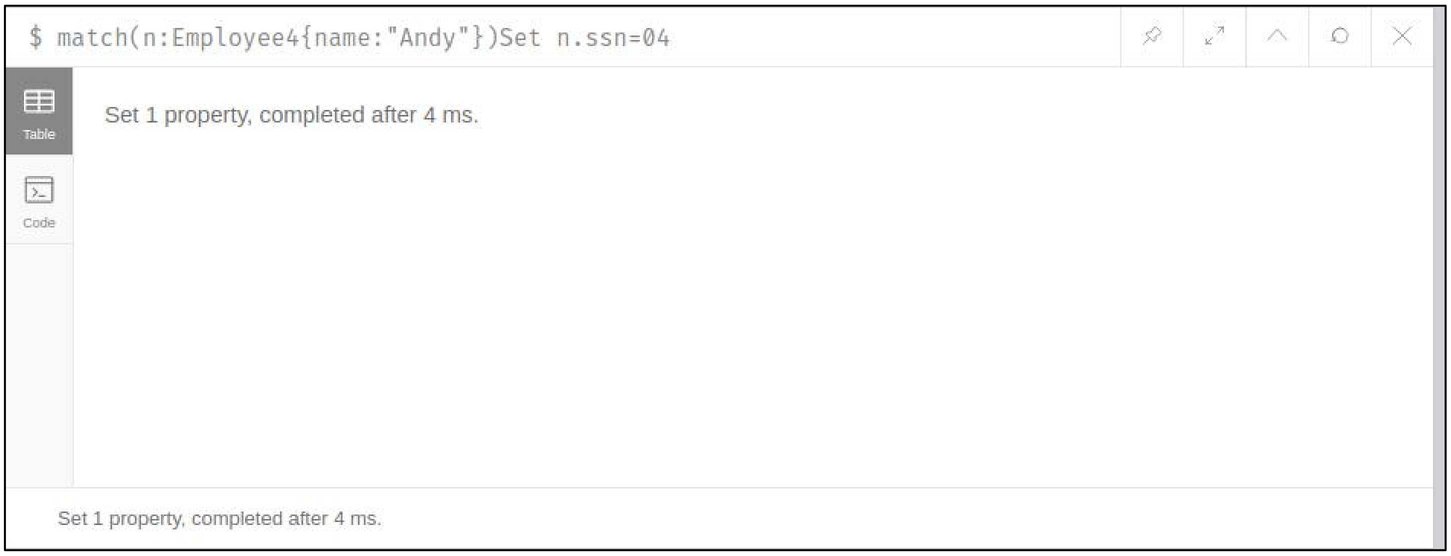
Set 1 property, completed after 4 ms.



```
$ match(n:Employee3)Set n.ssn=03
```

Set 1 property, completed after 4 ms.

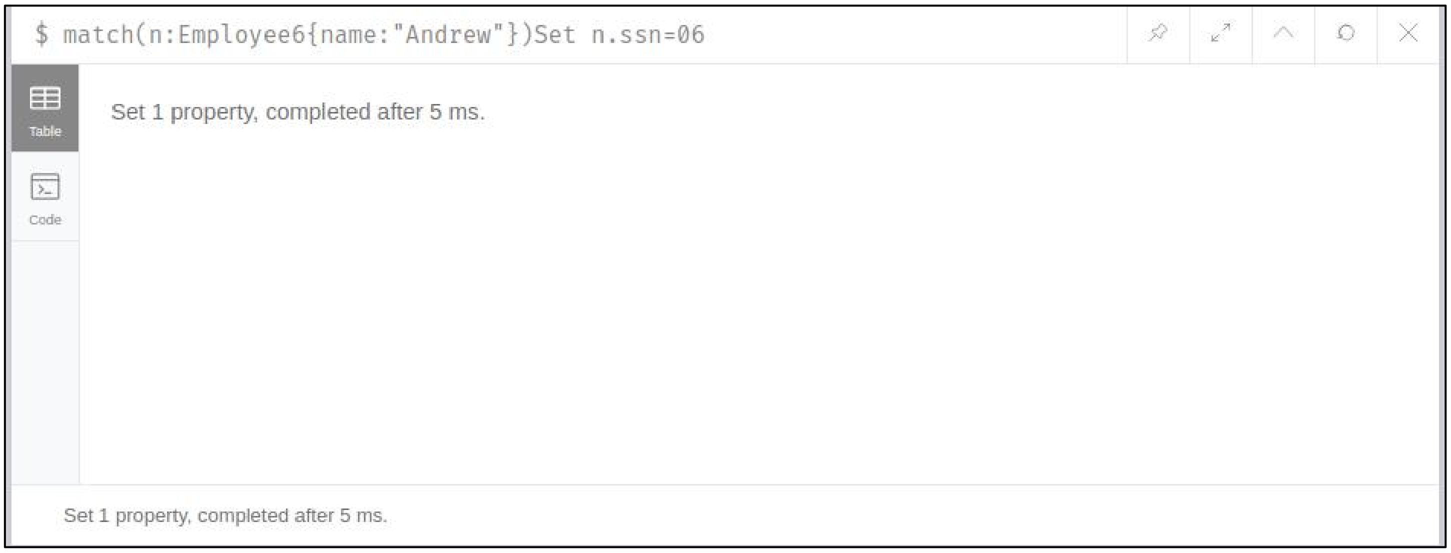
Set 1 property, completed after 4 ms.



```
$ match(n:Employee4{name:"Andy"})Set n.ssn=04
```

Set 1 property, completed after 4 ms.

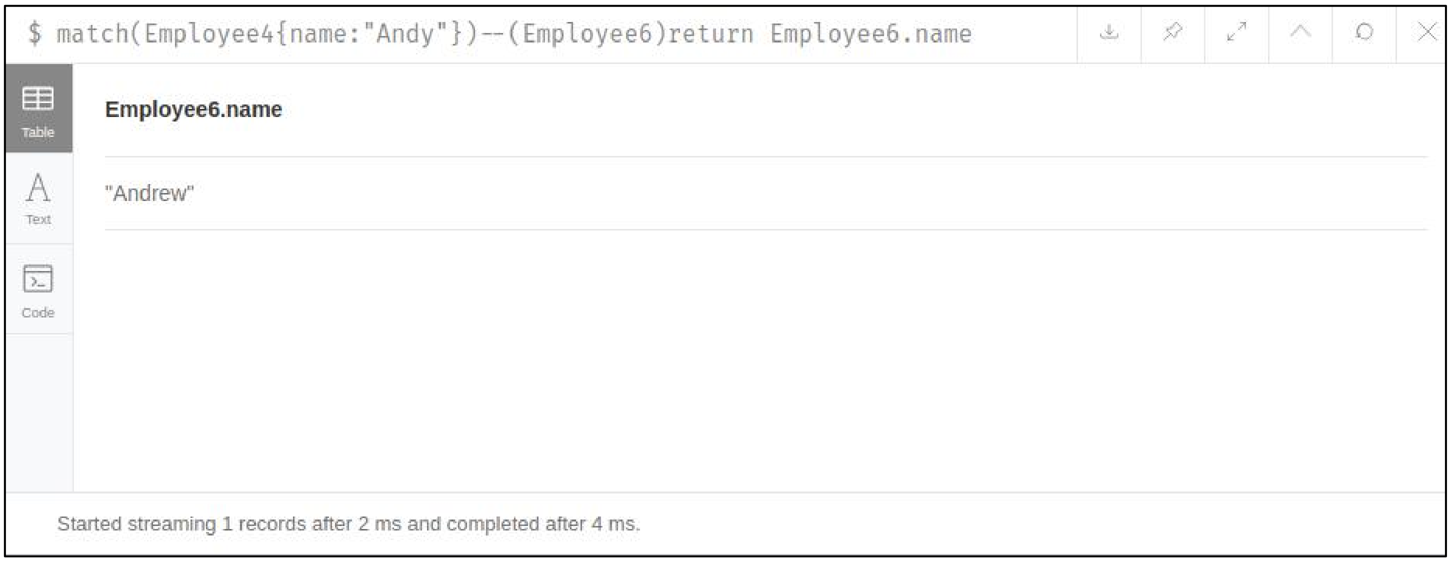
Set 1 property, completed after 4 ms.



```
$ match(n:Employee6{name:"Andrew"})Set n.ssn=06
```

```
Set 1 property, completed after 5 ms.
```

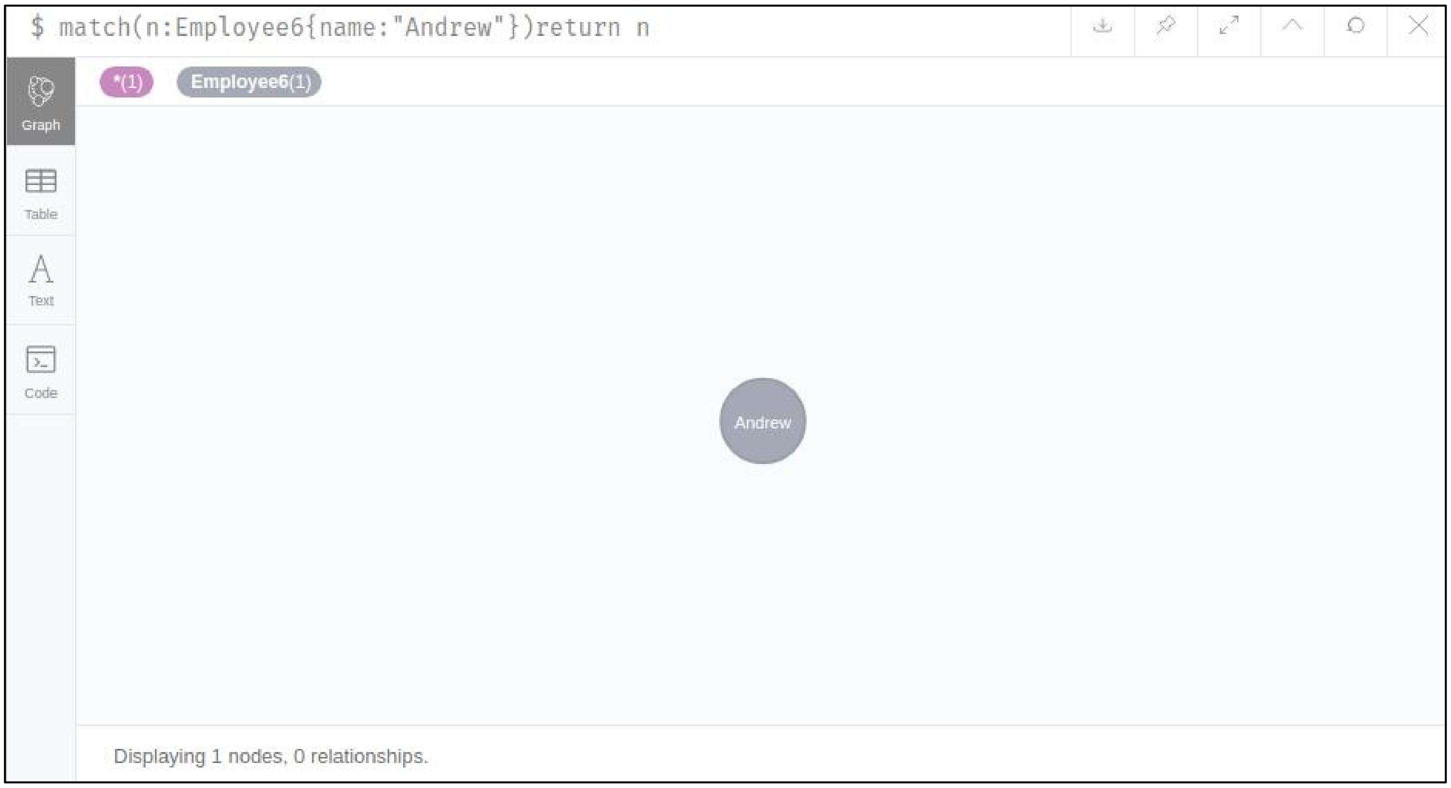
```
Set 1 property, completed after 5 ms.
```



```
$ match(Employee4{name:"Andy"})--(Employee6)return Employee6.name
```

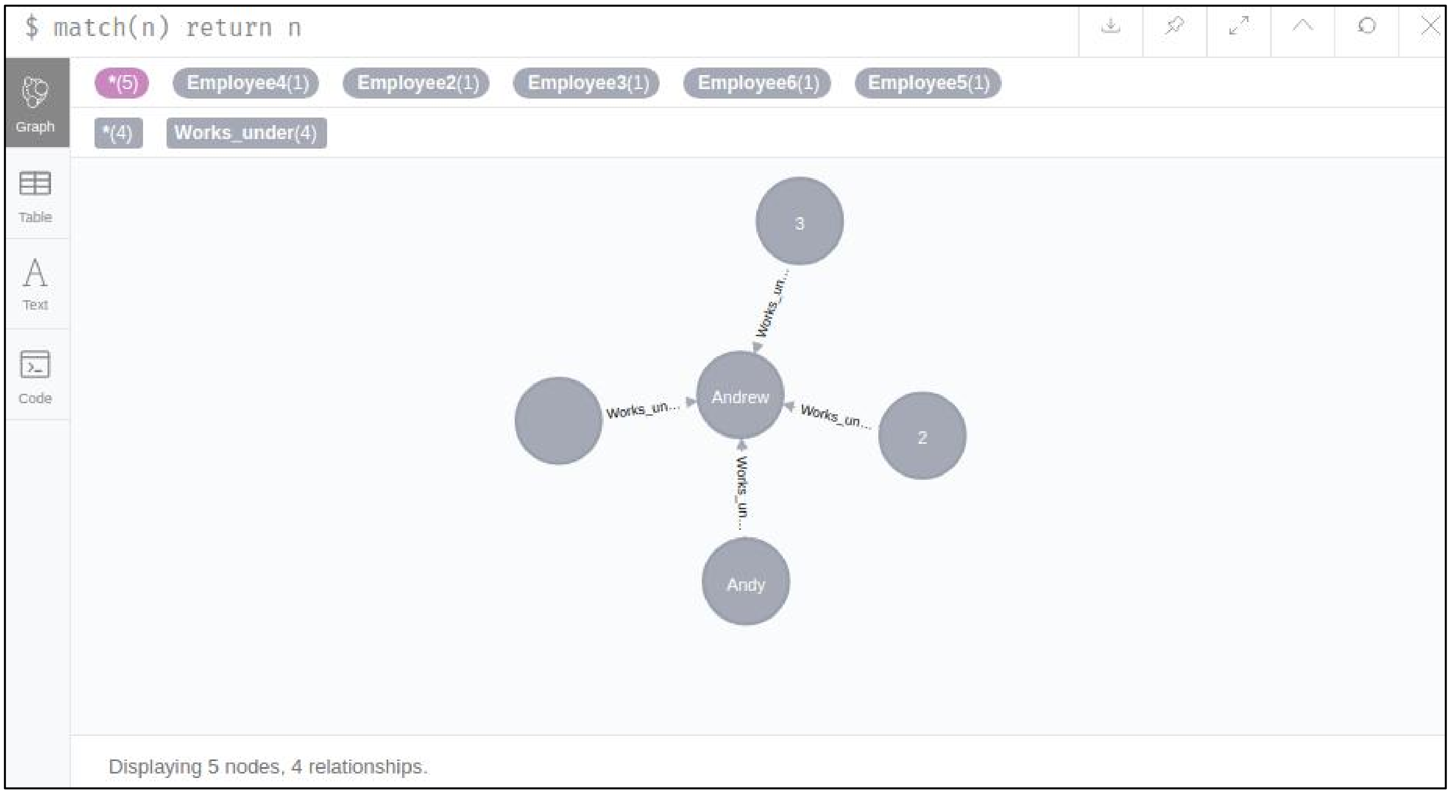
Employee6.name
"Andrew"

```
Started streaming 1 records after 2 ms and completed after 4 ms.
```



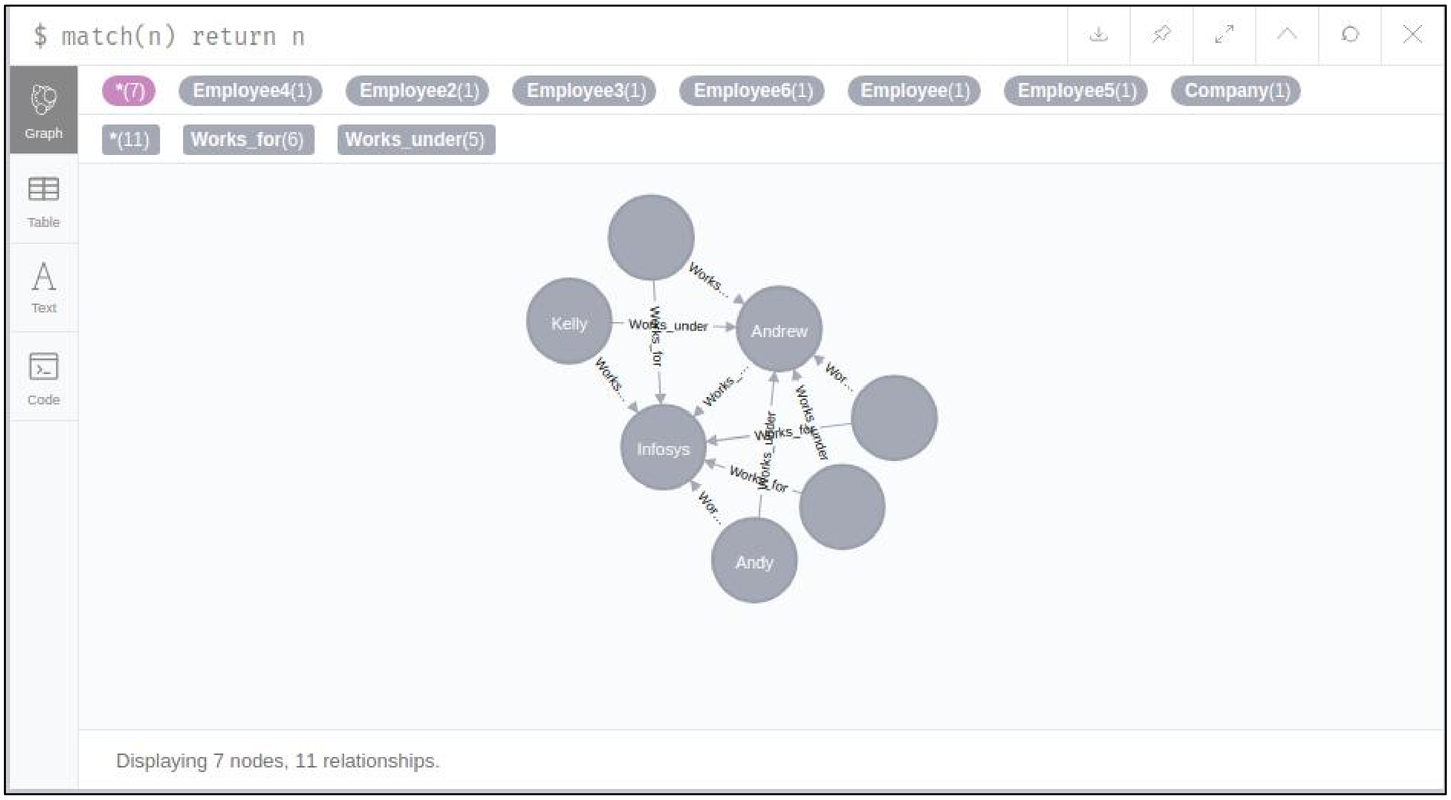
```
$ match(n:Employee6{name:"Andrew"})return n
```

Displaying 1 nodes, 0 relationships.



```
$ match(n) return n
```

Displaying 5 nodes, 4 relationships.



```
$ match(n) return n
```

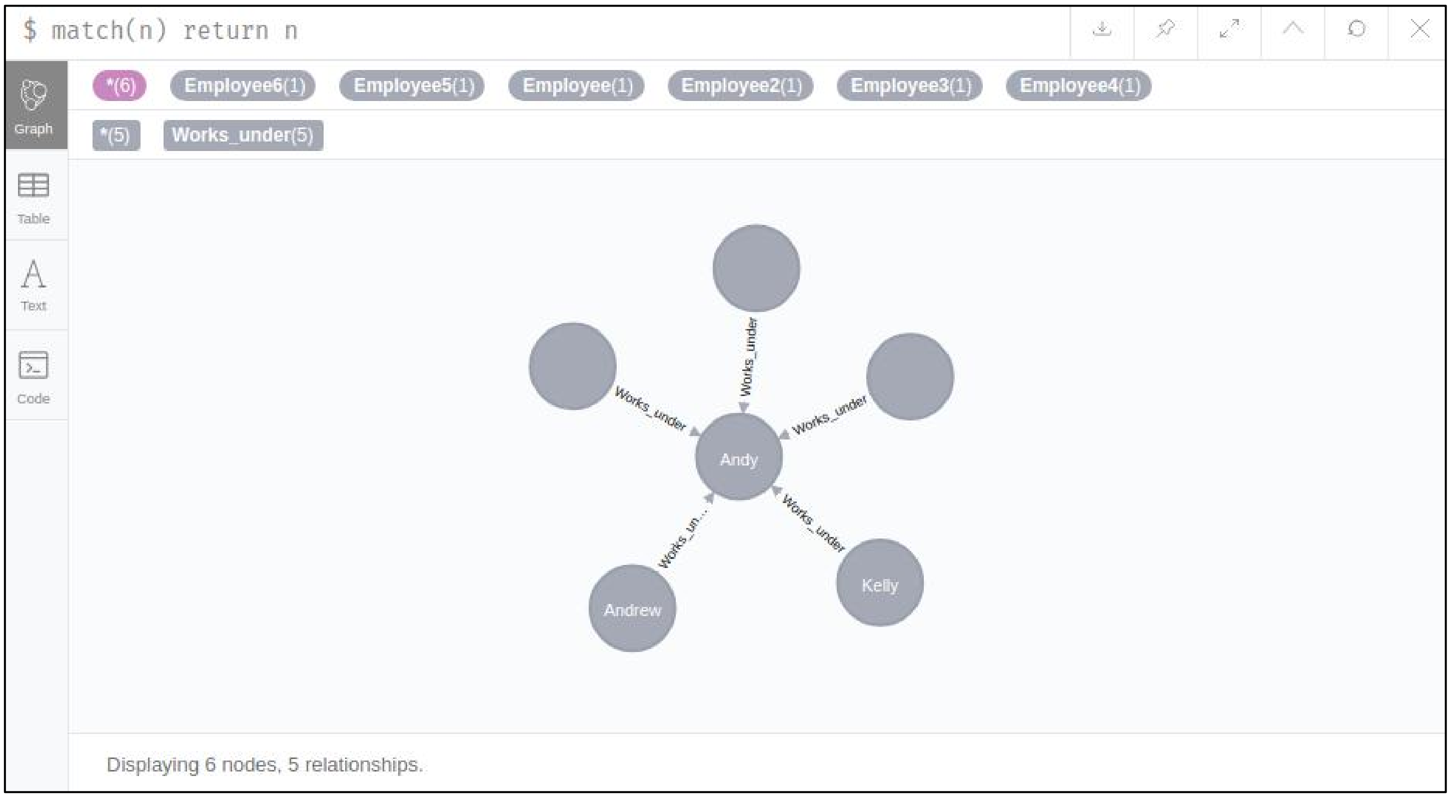
*(7) Employee4(1) Employee2(1) Employee3(1) Employee6(1) Employee(1) Employee5(1) Company(1)

*(11) Works_for(6) Works_under(5)

Displaying 7 nodes, 11 relationships.

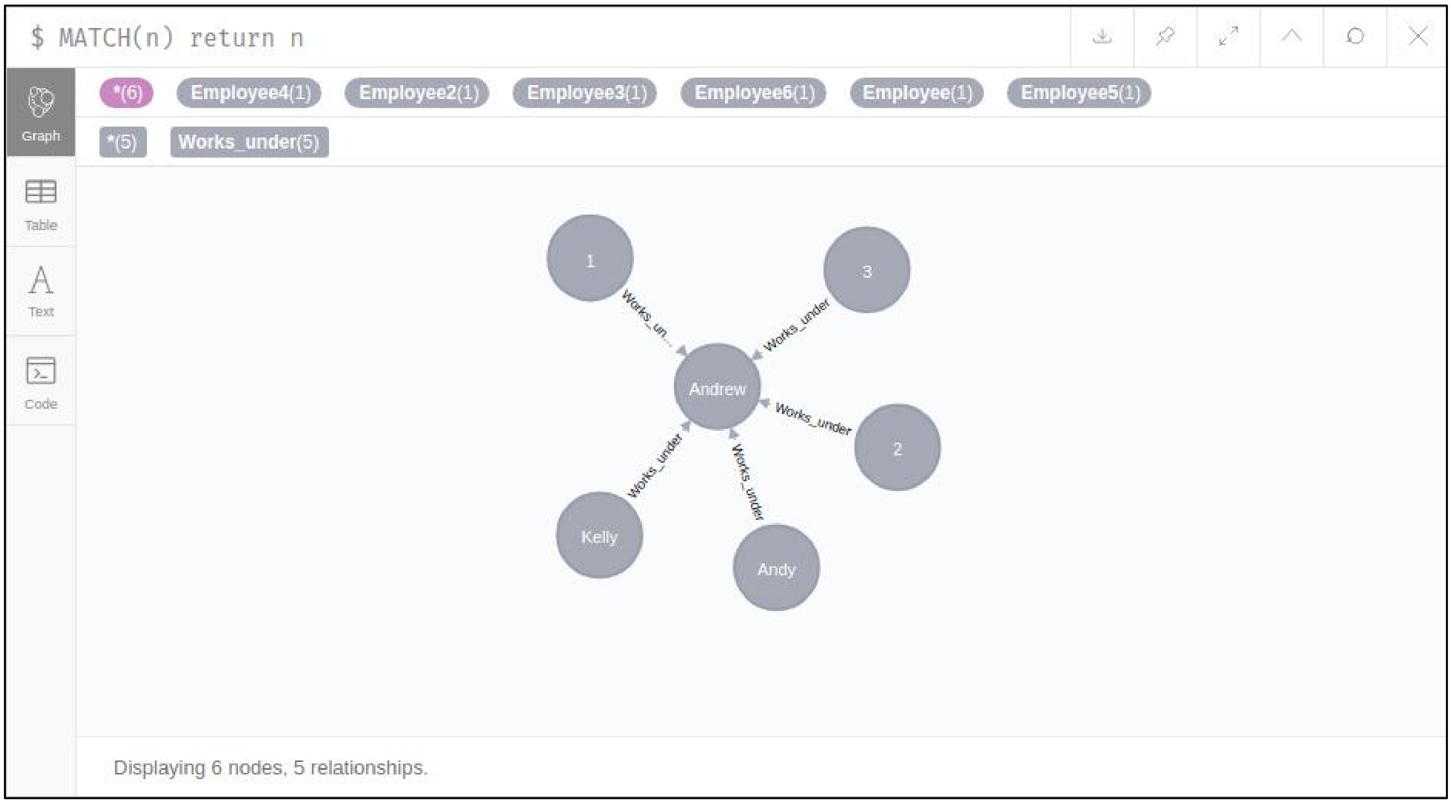
5. Delete operation

1. Delete all node To delete a node, use the DELETE clause. Match(n) delete (n) // delete all the nodes If the relationship exist we need to delete the relationship first before we delete the node **Delete the relationship** Match(n) detach (n)
2. Delete single node Syntax: match(filter) delete (n) Ex: MATCH (n:Person {name: 'UNKNOWN'}) DELETE n



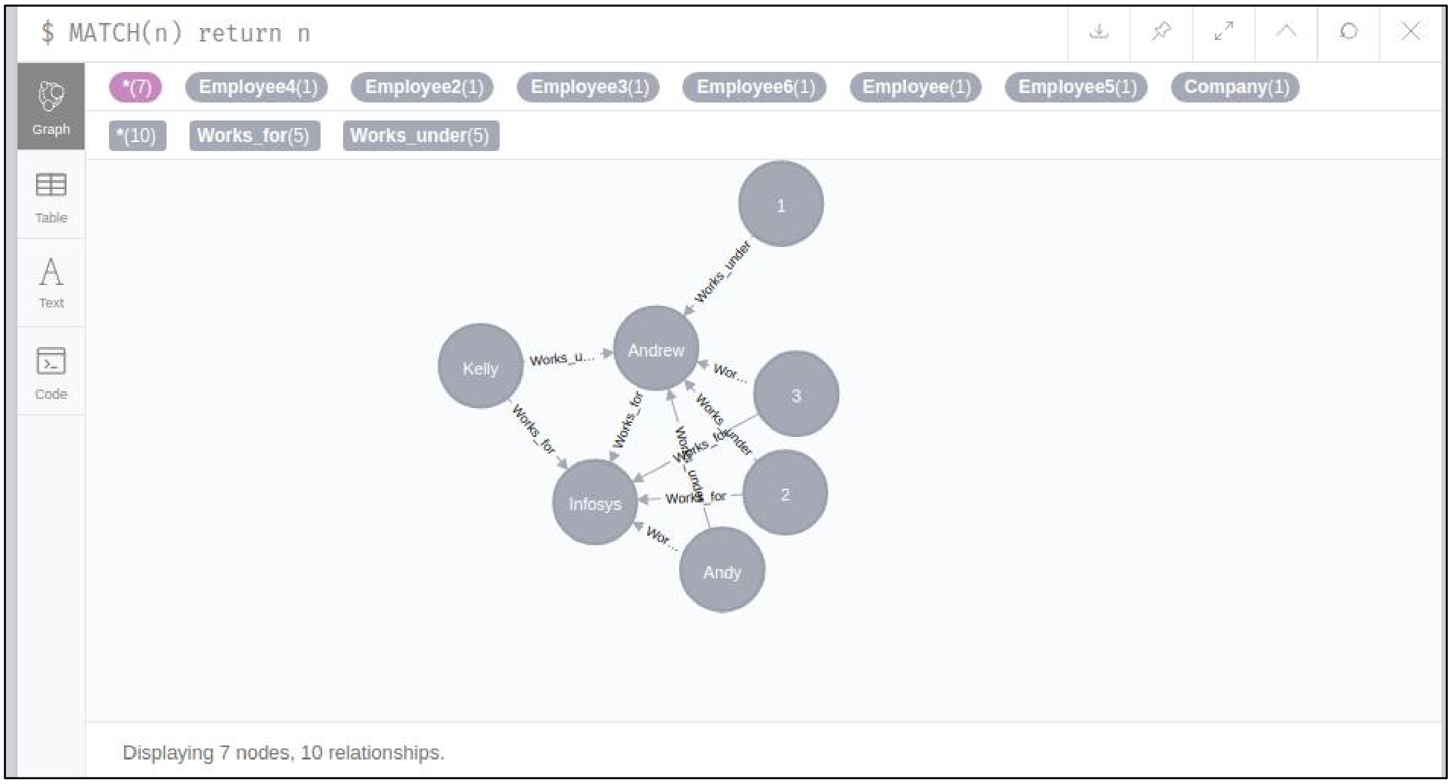
```
$ match(n) return n
```

Displaying 6 nodes, 5 relationships.



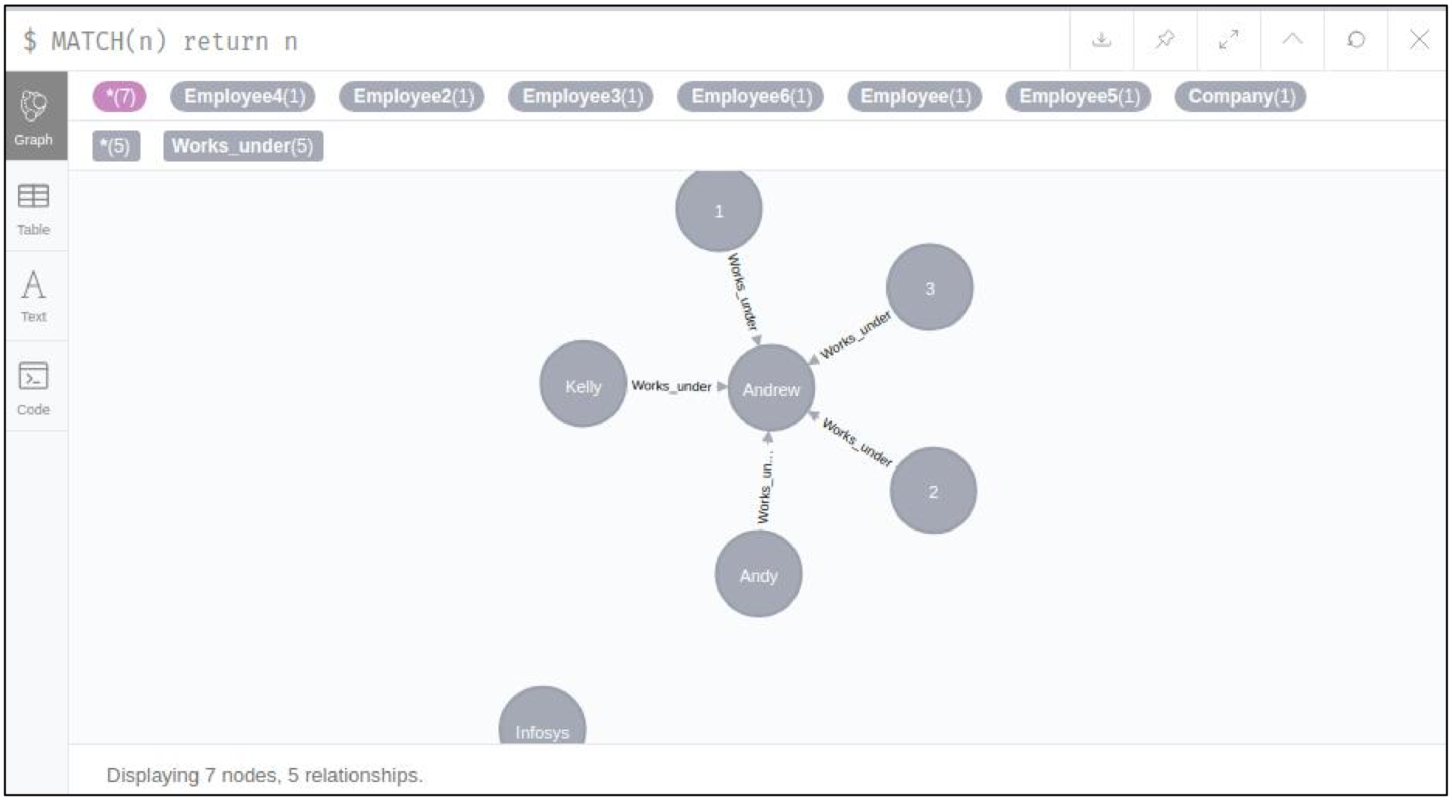
```
$ MATCH(n) return n
```

Displaying 6 nodes, 5 relationships.

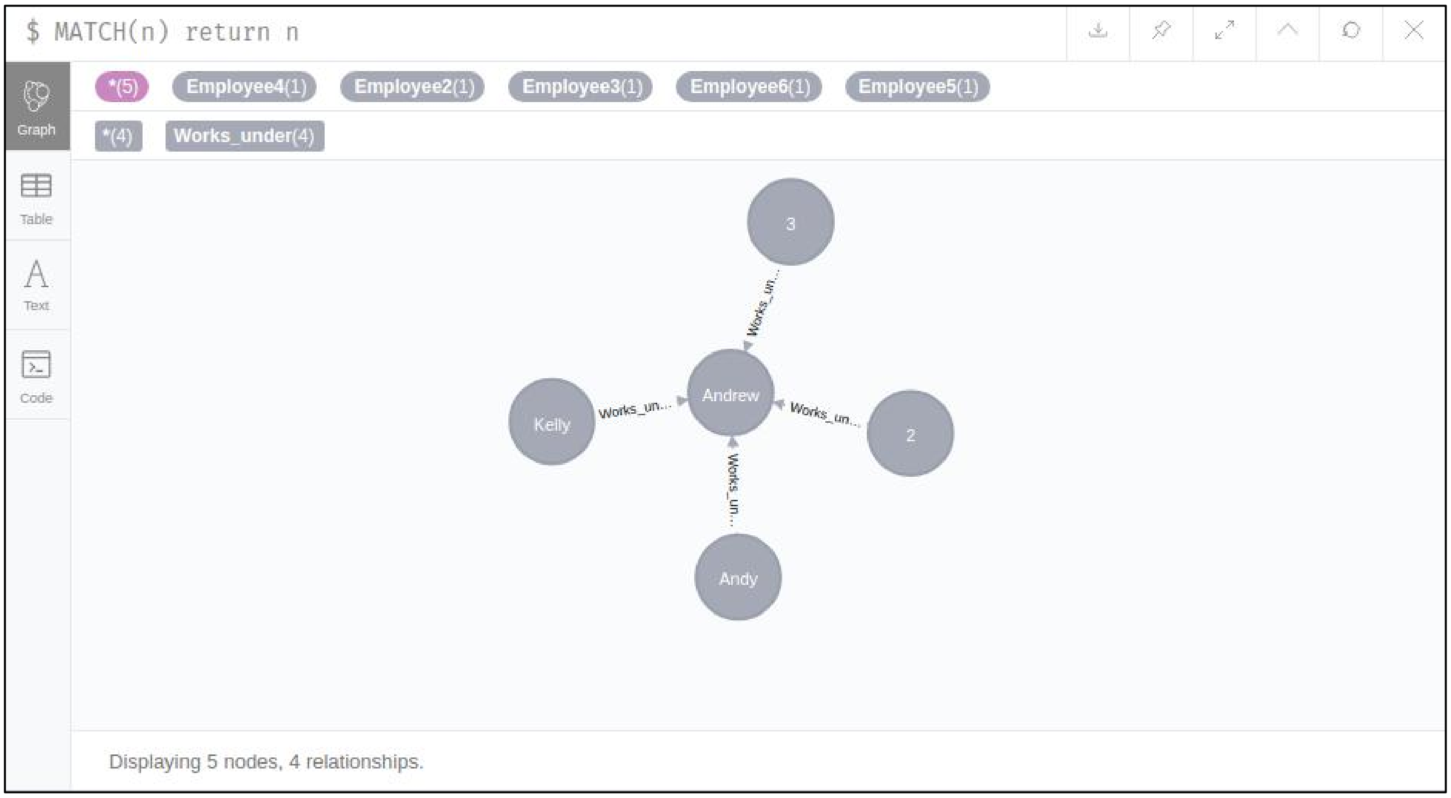


```
$ MATCH(n) return n
```

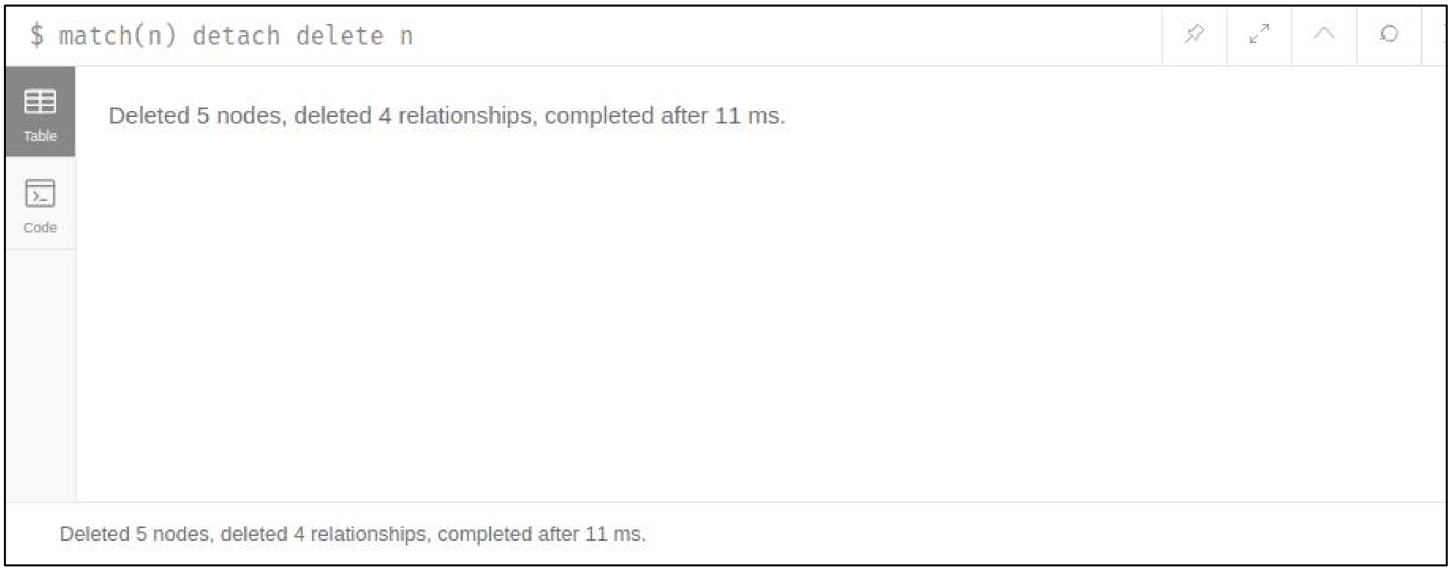
Displaying 7 nodes, 10 relationships.


```
$ MATCH(n) return n
```
*(7) Employee4(1) Employee2(1) Employee3(1) Employee6(1) Employee(1) Employee5(1) Company(1)
*(5) Works_under(5)
Graph
Table
Text
Code
1
3
Kelly
Andrew
2
Andy
Infosys
Works_under
Works_un...

Displaying 7 nodes, 5 relationships.


```
$ MATCH(n) return n
```
*(5) Employee4(1) Employee2(1) Employee3(1) Employee6(1) Employee5(1)
*(4) Works_under(4)
Graph
Table
Text
Code
3
Kelly
Andrew
2
Andy
Works_un...

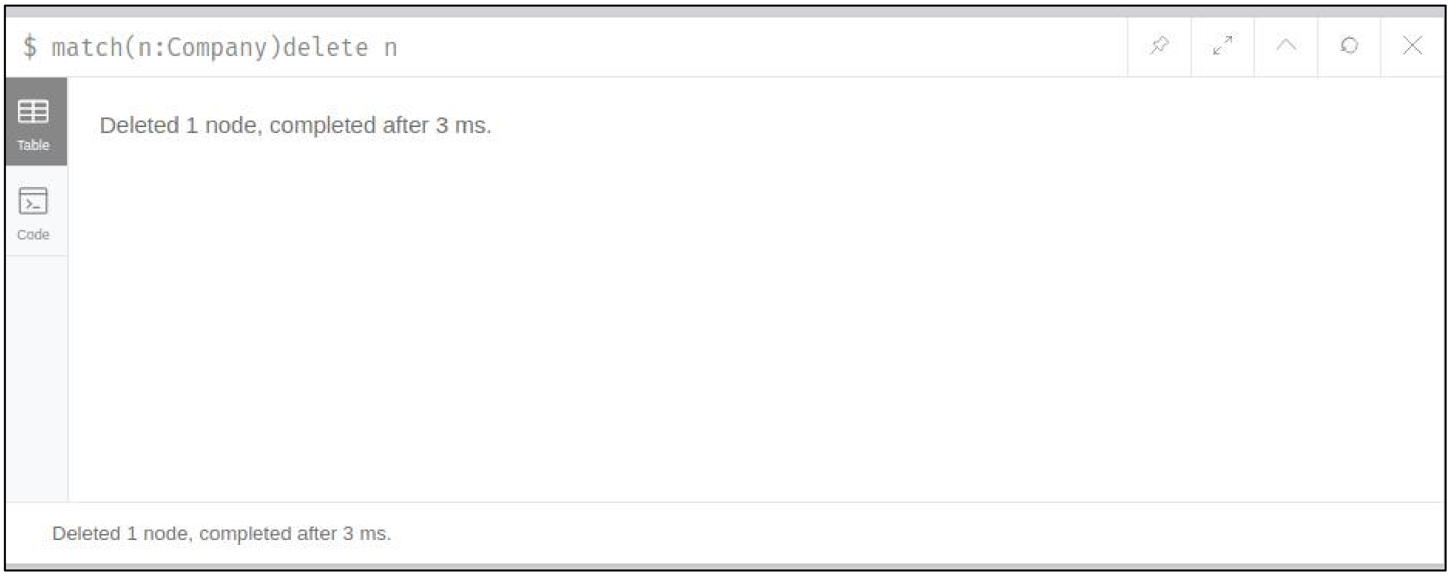
Displaying 5 nodes, 4 relationships.



```
$ match(n) detach delete n
```

Deleted 5 nodes, deleted 4 relationships, completed after 11 ms.

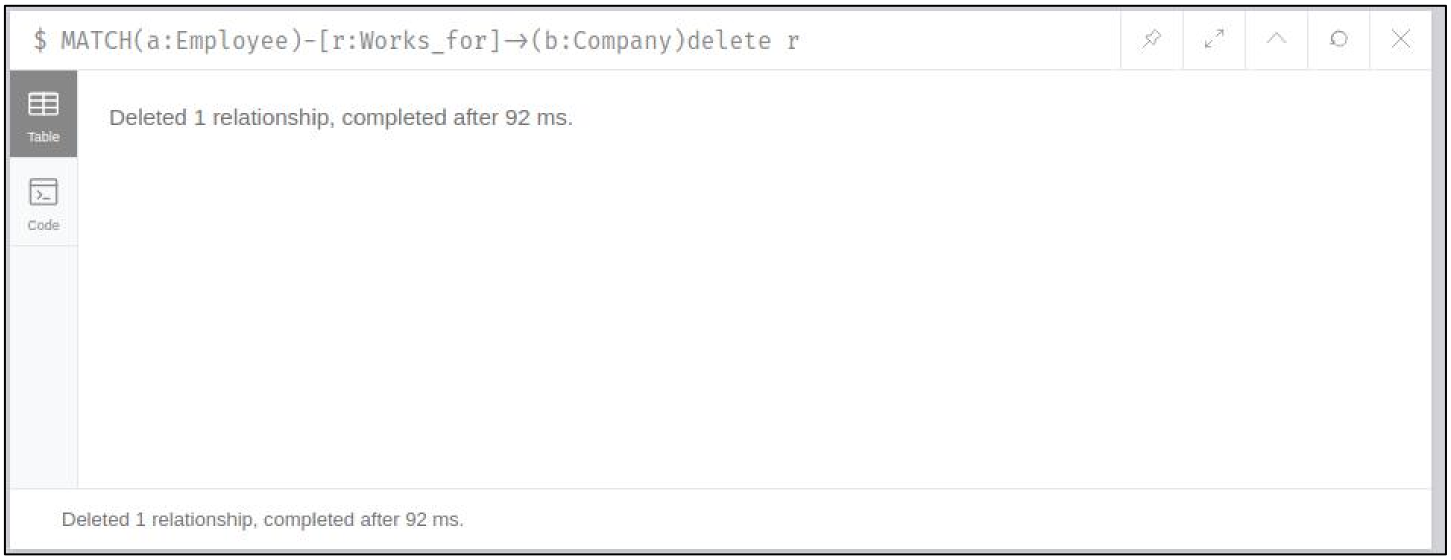
Deleted 5 nodes, deleted 4 relationships, completed after 11 ms.



```
$ match(n:Company)delete n
```

Deleted 1 node, completed after 3 ms.

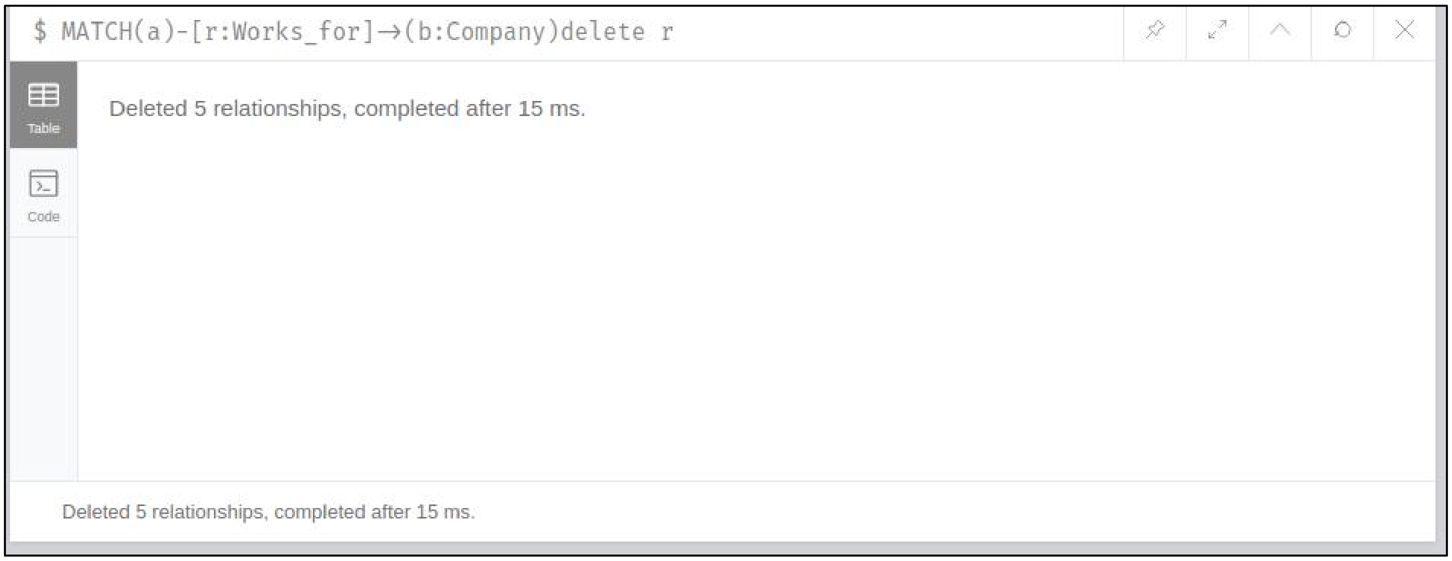
Deleted 1 node, completed after 3 ms.



```
$ MATCH(a:Employee)-[r:Works_for]→(b:Company)delete r
```

Deleted 1 relationship, completed after 92 ms.

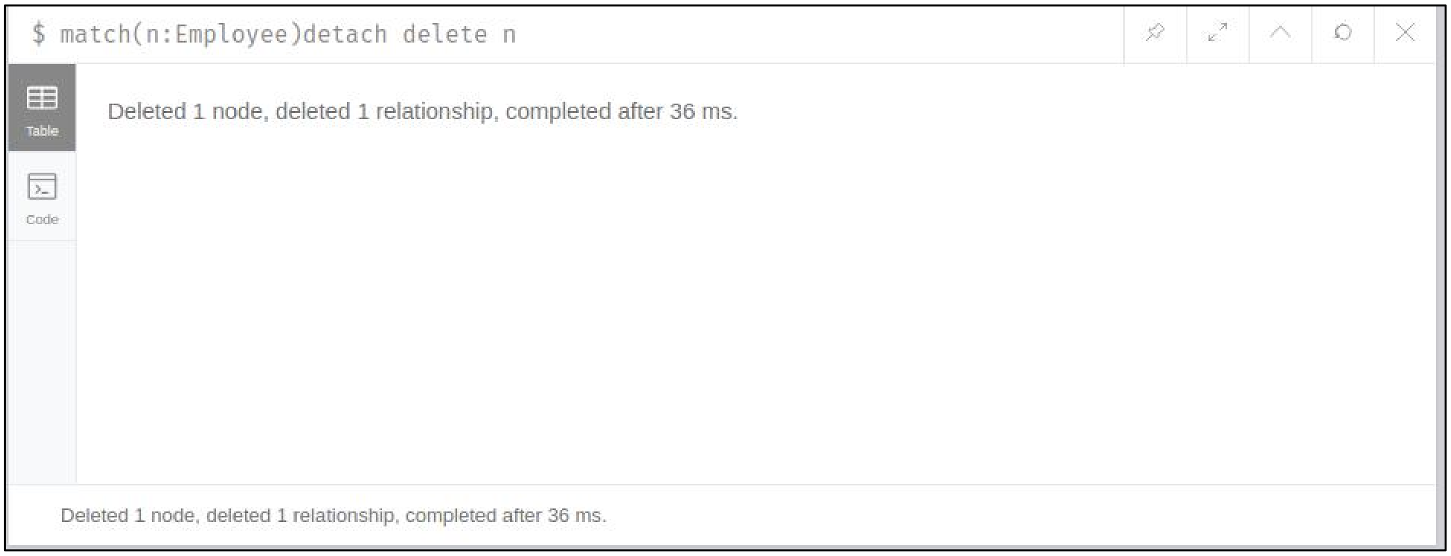
Deleted 1 relationship, completed after 92 ms.



```
$ MATCH(a)-[r:Works_for]→(b:Company)delete r
```

Deleted 5 relationships, completed after 15 ms.

Deleted 5 relationships, completed after 15 ms.



```
$ match(n:Employee)detach delete n
```

Deleted 1 node, deleted 1 relationship, completed after 36 ms.

Deleted 1 node, deleted 1 relationship, completed after 36 ms.

*****