Cypher Query Language

Dr James Xue

James.xue@northampton.ac.uk

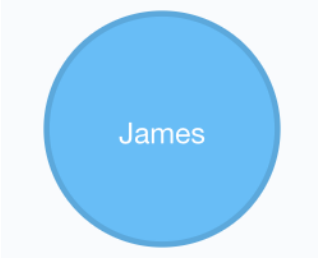
MERGE

- The Cypher MERGE operation is a MATCH or CREATE of the entire pattern. This means that if any element of the pattern does NOT exist, Neo4j will attempt to create the entire pattern.
- After a MERGE operation you are guaranteed to have a useable reference to all identifiers established during the Cypher MERGE operation because they were either found or created.

MERGE Nodes

- Assume the graph database is empty and we run the following command:

```
MERGE (l:Lecturer{name:"James"})
```



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

- If we run the command again, it doesn't create another :Lecturer node with the same name.

MERGE with ON CREATE SET, ON MATCH SET

- Utilizing the ON CREATE SET and ON MATCH SET is the right approach for setting the properties on the relationship based on whether it was created or set.

MERGE With ON MATCH SET

- From our previous example, the :Lecturer node exists, if we run the following command, if will not create a duplicated node; instead it adds a property called foundtime and assignment the system time to it.

```
MERGE (l:Lecturer{name:"James"})
ON CREATE SET l.createtime = timestamp()
ON MATCH SET l.foundtime = timestamp()
RETURN l.name, l.createtime, l.foundtime
```

Output:

l.name	l.createtime	l.foundtime
"James"	null	1509458238800

MERGE With ON CREATE SET

- Now, if we clear the database and run the following codes again:

```
MERGE (l:Lecturer{name:"James"})
ON CREATE SET l.createtime = timestamp()
ON MATCH SET l.foundtime = timestamp()
RETURN l.name, l.createtime, l.foundtime
```

Output:

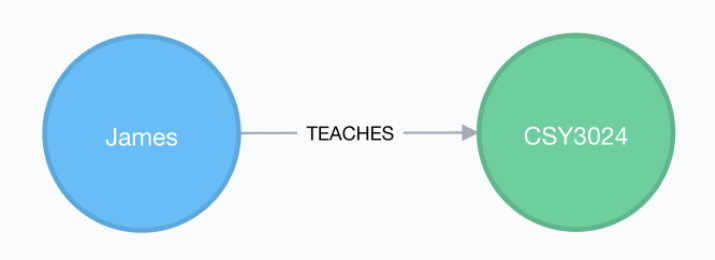
l.name	l.createtime	l.foundtime
"James"	1510313565818	null

MERGE Relationships

- When merging relationships, there would be different scenarios:
  - when two nodes and the relationship exist:
  - when two nodes exist, but not the relationship
  - when only one node exists, no relationship
  - when some node exists, and there is a unique constraint on some property of the node
  - two nodes and a relationship between exists, but has different direction with the one in the MERGE command
  - Merge a undirectional relationship

MERGE Relationships

- Case 1: when two nodes and the relationship exist, use MERGE only:



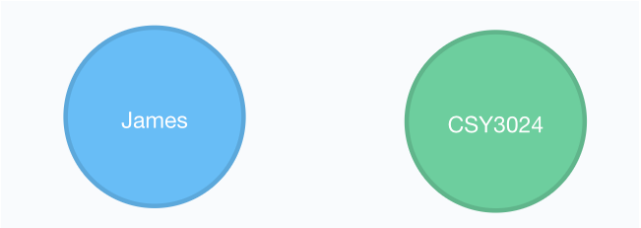

```
MERGE (l:Lecturer{name:"James"})-[:TEACHES]->(m:Module{code:"CSY3024"})
```

Output:

(no changes, no records)

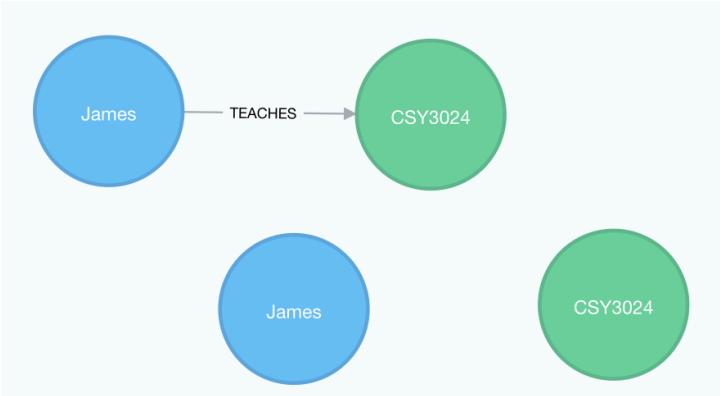
MERGE Relationships …

- Case 2: when two nodes exist, but not the relationship, use MERGE only:




```
MERGE (l:Lecturer{name:"James"})-[:TEACHES]->(m:Module{code:"CSY3024"})
```

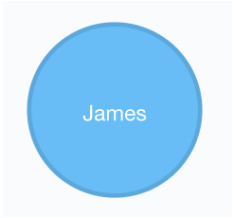
Output:



Added 2 labels, created 2 nodes, set 2 properties, created 1 relationship, completed after 1 ms.

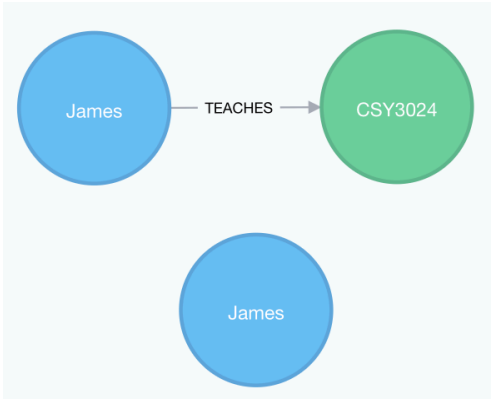
MERGE Relationships …

- Case 3: when only one node exists, no relationship; use MERGE only:




```
MERGE (l:Lecturer{name:"James"})-[:TEACHES]->(m:Module{code:"CSY3024"})
```

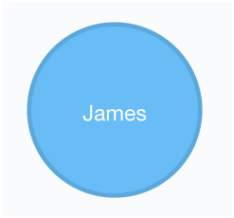
Output:



Added 2 labels, created 2 nodes, set 2 properties, created 1 relationship, completed after 1 ms.

MERGE Relationships …

- Case 4: when some node exists, and there is a unique constraint on some property of the node; use MERGE only:




```
CREATE CONSTRAINT ON (l:Lecturer) ASSERT l.name IS UNIQUE
```

These two commands need to be run separately.

```
MERGE (l:Lecturer{name:"James"})-[:TEACHES]->(m:Module{code:"CSY3024"})
```

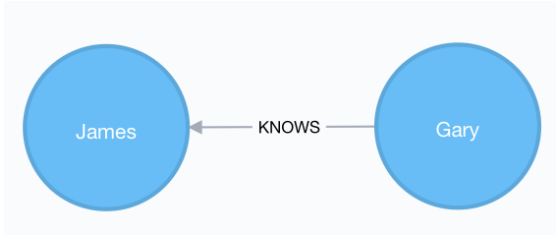
Output:

ERROR **Neo.ClientError.Schema.ConstraintValidationFailed**

```
Node(53) already exists with label `Lecturer` and property `name` = 'James'
```

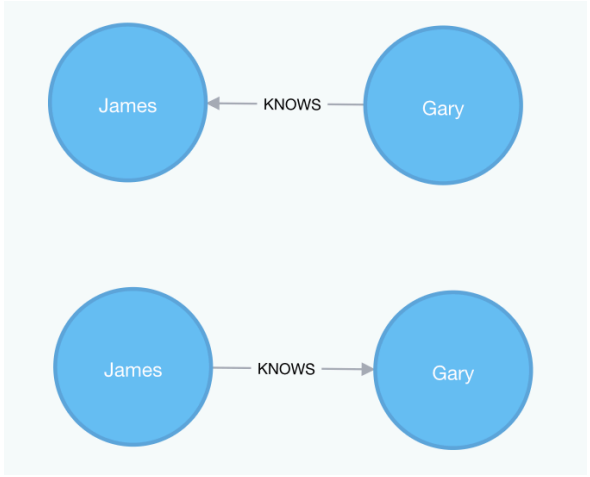
MERGE Relationships …

- Case 5: two nodes and a relationship between exists, but has different direction with the one in the MERGE command; use MERGE only:



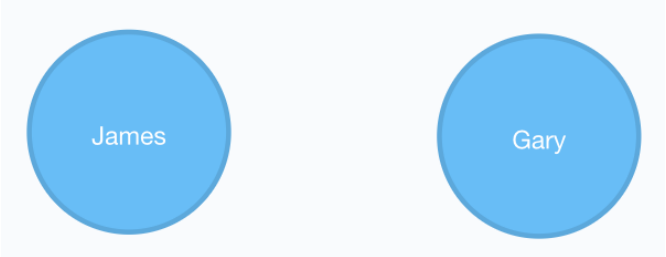

```
MERGE (:Lecturer{name:"James"})-[:KNOWS]->(:Lecturer{name:"Gary"})
```

Output:



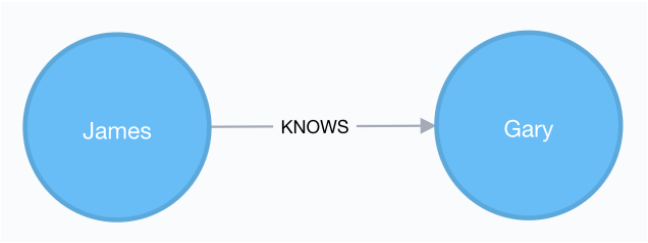
MERGE Relationships …

- Case 6: two nodes exist without relationship; merge a un-directional relationship with MATCH and MERGE:




```
MATCH (j:Lecturer{name:"James"}), (g:Lecturer{name:"Gary"})
MERGE (j)-[:KNOWS]-(g)
```

Output:

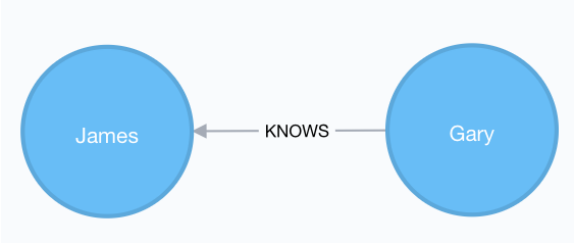


Created 1 relationship, completed after 4 ms.

As 'James' and 'Gary' do not know each other, this MERGE query will create a :KNOWS relationship between them. The direction of the created relationship is arbitrary. (Note: most of the time from left to right)

MERGE Relationships …

- Case 7: two nodes exist with relationship; MERGE a un-directional relationship with MATCH and MERGE:




```
MATCH (j:Lecturer{name:"James"}), (g:Lecturer{name:"Gary"})
MERGE (j)-[:KNOWS]-(g)
```

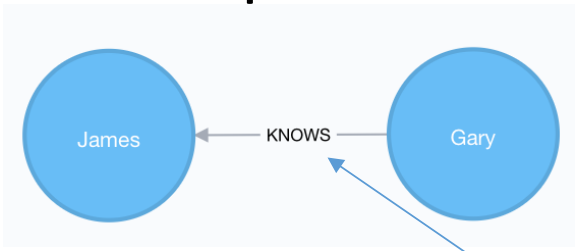
Output:

(no changes, no records)

Note: it doesn't matter which direction the relationship has in the graph,
As long as there is match, nothing will be created.

MERGE Relationships …

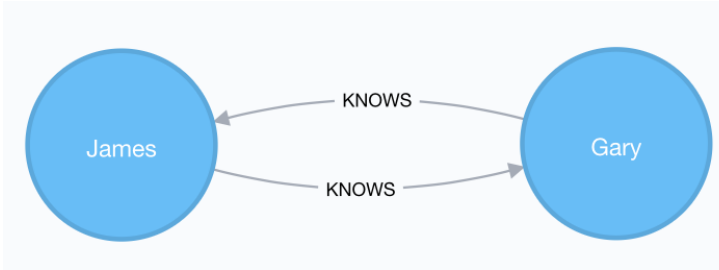
- Case 8: two nodes exist with relationship, which has a different direction with the relationship to be merged; use MATCH and MERGE:




```
MATCH (j:Lecturer{name:"James"}), (g:Lecturer{name:"Gary"})
MERGE (j)-[:KNOWS]->(g)
```

There is already a relationship between Two nodes with different direction with the one to be merged.

Output:



What's the difference between case 5 and 8?

Exercises

- Start Neo4j database
- Create some nodes and relationships based on the diagram on the previous slides (or a similar diagram of your choice).
- Practice the MERGE clause for all the eight different cases listed on the previous slides.
- Log your learning activities and reflection in your diary