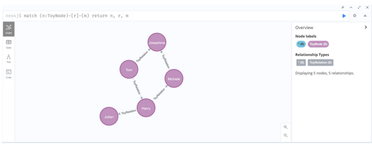
Hands-On: Modifying a Graph With Neo4j

We've already learned a little about the Neo4j interface, and we've learned how to create a relatively simple graph with it. In this activity, we're going to learn how to add another node and an edge to the graph.

For this activity, we are using the graph that we created in the previous activity:



Before starting, make sure to have your Neo4j container running.

Step 1. Add another node and edge CORRECTLY to the graph. Let's say that Julian has a fiancée and her name is Joyce and she works as a store clerk.

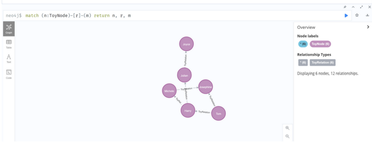
The code to produce that node would be the following:

```
match (n:ToyNode {name:'Julian'})
merge (n)-[:ToyRelation {relationship: 'fiancee'}]->(m:ToyNode {name:'Joyce', job:'store cl
```

This process involves two steps or two separate commands. First command requires you to find the node that you want to add the new node to, so we use the match command and we specify the *ToyNode* named Julian. Once that command is issued then we'll use the merge command and define the relation between Julian and the new node, and it's going to be fiancée. Finally, the new node will also be a *ToyNode*, and the name is *Joyce*, and her job is store clerk.

Run the command on Neo4j and see how the graph changes by running the following command:

```
match (n:ToyNode)-[r]-(m) return n, r, m
```



As you can observe, we successfully included a new node.

Step 2. Add a node incorrectly to an existing graph. To demonstrate more properly the incorrect addition of a new node, we are going to erase our current graph and recreate the original one.

First, run the following command to delete the graph:

```
match (n)-[r]-() delete n, r
```

In the command, we first match all nodes and relationships, to then delete them.

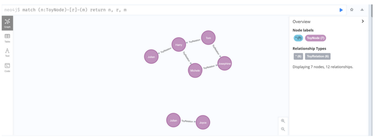
One of the nice things about Neo4j is that it keeps a history of all the commands we run. So we can easily find the command that created our original graph and run it again

Once you can verify the creation of the original graph by rerunning the command to view it.

Now, let's say that for some reason you assume that the *create* command is the command to use for this case, and you end up defining your code as following:

```
create (n:ToyNode {name:'Julian'})-[:ToyRelation {relationship: 'fiancee'}]->(m:ToyNode {na
```

Execute it, and you will get the following graph:



Instead of matching the node Julian and then merging the new node, what the code did was to create a new relationship separate from the original network.

Step 3. Correcting our mistake. We can run the following command to correct our mistake:

```
match (n:ToyNode {name:'Joyce'})-[r]-(m) delete n, r, m
```

This command matched the *ToyNode* with the name Joyce, and deletes all relationships and nodes associated with it. After running it, you will see that 2 nodes and 1 relationship are deleted.

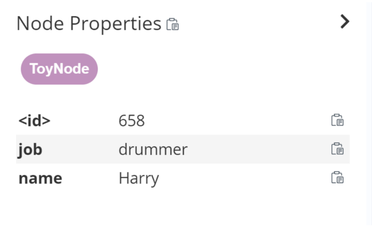
Step 4. Modify a Node's Information. If you remember, when we first created our network, Harry didn't have a job. Let's go ahead and add a job to Harry.

First, we will use the following command:

```
match (n:ToyNode) where n.name = 'Harry' set n.job = 'drummer'
```

The code uses the match command to locate the node in which the name is equal to Harry. Then we use the set command and specify that job will be equal to *drummer*.

Run the command and then display Harry's node.



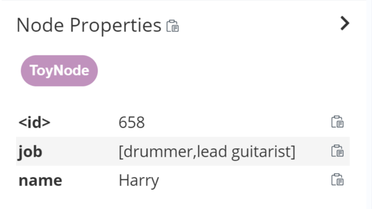
We successfully added a job for Harry.

Now, let's say that Harry is also a lead guitarist. We can include his additional job through the following command:

```
match (n:ToyNode) where n.name = 'Harry' set n.job = n.job + ['lead guitarist']
```

We are again matching the node where the property *name* is equal to harry, and now we are setting the value for the property to be equal to the current value plus a new value.

Run the code in Neo4j, and display the graph again. Select the node of Harry and you will see that now Harry has 2 jobs.



Go to next item ✓ Completed

Like Dislike Report an issue