IST769 Lab I

# Graph Model: Neo4J

### In this lab, we will explore the Graph model with the Neo4j database. We will learn how to query data with the Cypher query language and access data from Apache Spark with Cypher.

### Learning Outcomes

At the end of this lab you should be able to:

* Query a graph model using Neo4j’s Cypher query language.
* Import and export data from Neo4j into spark dataframes.
* Build a Graph from a table of data.

### Pre-Requisites

Before you begin:

* Open a terminal window in the lab environment
* Set the current working directory to **advanced-databases**
* Start the following services required by the lab Spark and Redis:   
  **jupyter neo4j**

### Tools Used In this Lab

The following tools will be used in this lab:

1. To access Jupyter Lab from your Windows host:  
   [http://localhost:8888](http://localhost:8888/)   
   The password is **SU2orange!**
2. To access The Neo4j Admin UI:  
   [http://localhost:7474](http://localhost:7474/)

# Lab Problem Set

**QUESTIONS:   
  
For each question, include a copy of the code required to complete the question along with a screenshot of the code and a screenshot of the output.**

1. Using the **:play northwind-graph** command, build the Northwind Product Catalog in Neo4j.   
   Diagram, table

   Description automatically generated  
   Make sure to load the Product, Categories and Suppliers nodes along with the PART\_OF and SUPPLIES relationships by running the sample code provided. This should just be a matter of following the commands in the first 3 steps of the Northwind graph.   
     
   As proof you've completed this correctly, write a Cipher query to display all Products, Suppliers and Categories using both relationships. Your screenshot should include your Cipher code plus the graph output. If you did it correctly there should be 8 categoires, 77 products, 29 supplier notes (and lines connecting nodes).

MATCH(c:Category) <- [:PART\_OF]-(p:Product)<-[:SUPPLIES]-(s:Supplier)

return c,p,s

1. Select a Supplier Contact Name of your choosing. Write a Cipher query to display the supplier name, the supplier’s company name, the names of the products they supply and the unit price of those products for only products that are not discontinued. Display this information in a table.

MATCH(s:Supplier{contactName:"Regina Murphy"})-[:SUPPLIES]->(p:Product)

WHERE p.discontinued = false

return s.contactName, s.companyName, p.productName, p.unitPrice, p.discontinued

1. Select two products of your choosing. Write a Cypher query to display a graph of the supplier and category for those products. Based on the graph displayed as output, are the two products you selected in the same category?

MATCH (s:Supplier)-[:SUPPLIES]->(p:Product)-[:PART\_OF]->(c:Category)

WHERE p.productName = "Grandma's Boysenberry Spread" or p.productName = "Northwoods Cranberry Sauce"

return s,p,c

1. You just sold 30 units of `laughing lumberjack lager` update the node to reflect the proper stock and display the output.

MATCH(p:Product)

WHERE p.productName = 'Laughing Lumberjack Lager'

SET p.unitsInStock = p.unitsInStock - 30

return p.productName, p.unitsInStock

1. Load a Spark dataframe of USA country suppliers and their products for products that are not discontined. Include supplier company and contact names, country and phone. From products include product name, discontinued, unit price and units in stock.

cipher\_query = '''

MATCH (s:Supplier)-[:SUPPLIES]->(p:Product)

WHERE s.country = "USA" and p.discontinued = false

return s.companyName, s.contactName, s.country, p.productName, p.discontinued, p.unitPrice, p.unitsInStock

'''

df = spark.read.format("org.neo4j.spark.DataSource") \

.option("url", bolt\_url) \

.option("query", cipher\_query) \

.load()

1. Load the **/datasets/fudgemart/fudgemart-employees.json** into a spark dataframe. Make sure to create a column **employee\_name** which combines the first and last names together.

from pyspark.sql.functions import col, concat, lit

q6\_emp\_df = spark.read.option("multiline",True) \

.json("file:///home/jovyan/datasets/fudgemart/fudgemart-employees.json") \

.withColumn("employee\_name", concat(col("employee\_firstname"), lit(" "), col("employee\_lastname")))

q6\_emp\_df.toPandas()

1. In Spark, load the employees into Neo4j under the label node Employee include employee\_name, employee\_department, employee\_id and employee\_jobtitle as node attributes. Make sure employee\_name is the first attribute as this will be the Node’s visible label.   
     
   Provide evidence the nodes were created in Neo4i UI with a Cypher query.

q7 = '''

MERGE (e:Employee { name: coalesce(event.employee\_name, 'Unknown'), dept: event.employee\_department,

jobtitle: event.employee\_jobtitle, id: event.employee\_id})

'''

employees = q6\_emp\_df.select("employee\_id", "employee\_name", "employee\_jobtitle", "employee\_department")

employees.write.format("org.neo4j.spark.DataSource").mode("Overwrite") \

.option("url", bolt\_url) \

.option("query", q7) \

.save()

MATCH (e:Employee) return e;

1. In Spark add a SUPERVISES relationship to the nodes. Basically you must match two nodes one where the id is the employee\_id and the other where the id is the employee\_supervisor\_id and then merge a relationship.  
     
   Provide evidence the relationships were created in Neo4i UI with a Cypher query to show the employees of a supervisor of your choosing.

q8\_cipher\_emp = '''

MATCH (e:Employee)

MATCH (s:Employee)

WHERE e.id = event.employee\_id AND s.id = event.employee\_supervisor\_id

MERGE (s)-[:SUPERVISES]->(e)

'''

sup\_empl = q6\_emp\_df.select("employee\_supervisor\_id", "employee\_id").orderBy("employee\_supervisor\_id")

sup\_empl.write.format("org.neo4j.spark.DataSource").mode("Overwrite") \

.option("url", bolt\_url) \

.option("query", q8\_cipher\_emp) \

.save()

MATCH (s:Employee {name: "Mary Mi"})-[:SUPERVISES]->(e:Employee) return s,e;

1. In Neo4J build an organizational chart by starting at the supervisor who is the “CEO” and the employees recursively 4 levels deep. To learn how to query recursively, check out:  
   <https://stackoverflow.com/questions/31079881/simple-recursive-cypher-query>   
   Display the graph.

MATCH (s:Employee {jobtitle: "CEO"})-[:SUPERVISES \*1..3]->(e:Employee) return s,e;

**IMPORTANT:** When you are finished with the lab, execute:

PS:> docker-compose stop

To turn off all running services, then shut down your Azure Lab instance.