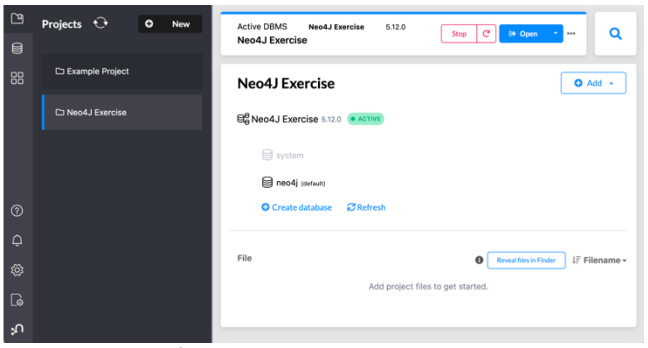
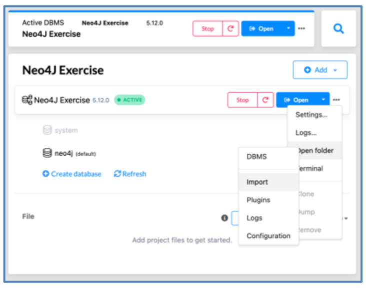
Data Science Infrastructures – Exercise 05 (DSI E05 ST 24)

23/05/2024 / Philipp Wieder / philipp.wieder@gwdg.de
Software needed: Neo4J Desktop (Link in text)
Files: orders.csv (downloadable from Stud.IP)

General Prerequisites



To fulfil this assignment, you need the Neo4J Desktop software, which you get here: https://neo4j.com/deployment-center/?ref=subscription#community. Install the Neo4J Desktop by following the instructions and have a quick look at the "Getting Started" information https://neo4j.com/docs/desktop-manual/current/. The first thing you have to do is to create a new project and a new local database. You find the respective menu items in the left menu bar of the Neo4J Desktop. Name both the project and the Database "Neo4J Exercise" (the database version is not important for us here, just choose the newest). Now start the database (Neo4J Desktop can only have one running database at a time). It takes a while. Once the database is operational, select it via the DBMS menu item and you will get basic information and control options. To work with the data for this exercise, you need to put the `csv` file into the Import folder. The screenshot to the left shows the respective menu. It opens the Import folder. Just copy the file for the exercise there.



Assignment I – View CSV Data in Neo4J

Points: 3

Once the database is up and running, you can open it with the Neo4J Browser. Just click on the "Open" Button assigned to the database (there are more options available through the drop-down menu, but you do not need them yet). The browser offers you among other things a command line for Cypher queries which we will use for our assignments.

Assignment

First you need to load some `csv` data into Neo4J. This is done with the Cypher statement `LOAD CSV FROM`. You find a detailed documentation how to do that here https://neo4j.com/developer/desktop-csv-import/.

The Cypher command follows this template:

```
LOAD CSV FROM 'file:///<filename>.csv' AS row
WITH row[0] AS <feature1>, row[1] AS <feature2>, …
RETURN <feature1>, <feature2>, …
```

`row[0]` refers to the first row in any `csv` file you load. Through the Cypher command shown above, you load it into Neo4J into a row called `<feature1>`. You change, of course, the name of the feature to something that describes the feature properly. The output is shown using the `RETURN` clause (which can be limited through `LIMIT` in case it gets too long).

Use the above Cypher command to load selected features (think pre-processing) of the `orders.csv` file. It contains 19 attributes, please import the following: `OrderID, CustomerID, OrderDate, ShipName, ShipCity, ShipCountry, ProductID, UnitPrice,` and `Quantity`. Please note that you have to convert Integer and Float values using the following functions: `toInteger(row[<numbery>])` or `toFloat(row[<number>])`. To prevent errors in case of missing `OrderID`s, please add the following at the end of the `WITH` clause: `WHERE orderID is not null.`

Store the Cypher and submit it as part of your solution.

[Please note that this Cypher command does not create any nodes or relationships! It just loads the selected features of the `csv` file.]

Assignment II – Create the Graph

Points: 5

In a next step we create the nodes and relationships.

Assignment II.A

First think about a proper schema that represents customers, products, and orders as well as their relationships. Please also think of the necessary properties to be added to the nodes and relationships.

Submit your visualization of the model. Handwritten paper submissions are ok, too ;-).

Assignment II.B

a) Create the Order nodes and properties according to the model using the `MERGE` command.

Here is an example:

```
MERGE (c:Country {countryCode: countryCode})
   ON CREATE SET c.name = countryName, p.size = countrySize
RETURN count(c)
```

This example Cypher creates nodes for countries based on the `countryCode` and assigns the respective name and size of a country as properties. It returns the number of nodes created. The values assigned to the node and properties are taken from a `csv` file similar to the one we have.

Now create the Order nodes based on the `OrderID` for our model accordingly and also add properties. Please note that you have to use the Cypher from Assignment I (`LOAD CSV FROM` …) before the `MERGE` to actually load the data from the `csv`. Otherwise, it does not work.

You can check the results using the `MATCH` clause:

```
MATCH (n)
RETURN n
```

This shows you all nodes you added to the Neo4J database.

b) Now extend the Cypher to create the Product nodes.

c) Now connect the Order nodes with the Product nodes and add the order-related parameter `Quantity`. The specific Cypher part for this looks as follows:

```
MATCH (order:Order {orderID: orderID})
MATCH (product:Product {productID: productID})
MERGE (order)-[op:CONTAINS]->(product)
      ON CREATE SET op.quantity = Quantity
```

(with variables `order` and `product` for Order and Product nodes, respectively). Adapt it to your existing Cypher and add it after the merging of the nodes.

Store the final Cypher as part of your solution.

You now have a graph, which contains orders and their related products. Submit the Cypher as a result.

OPTIONAL Assignment III – Complete the Graph

Assignment III.A

Complete the graph by creating further nodes for the Customers based on the `CustomerID`. Add the address details as parameters and relate the Customers to the Orders.

Submit the Cypher as a result.

Assignment III.B

Use the `MATCH` clause to find all Orders placed by `CustomerID` "RATTC". The syntax is described here: https://neo4j.com/docs/cypher-manual/current/clauses/match/.

Submit a screenshot of the result.