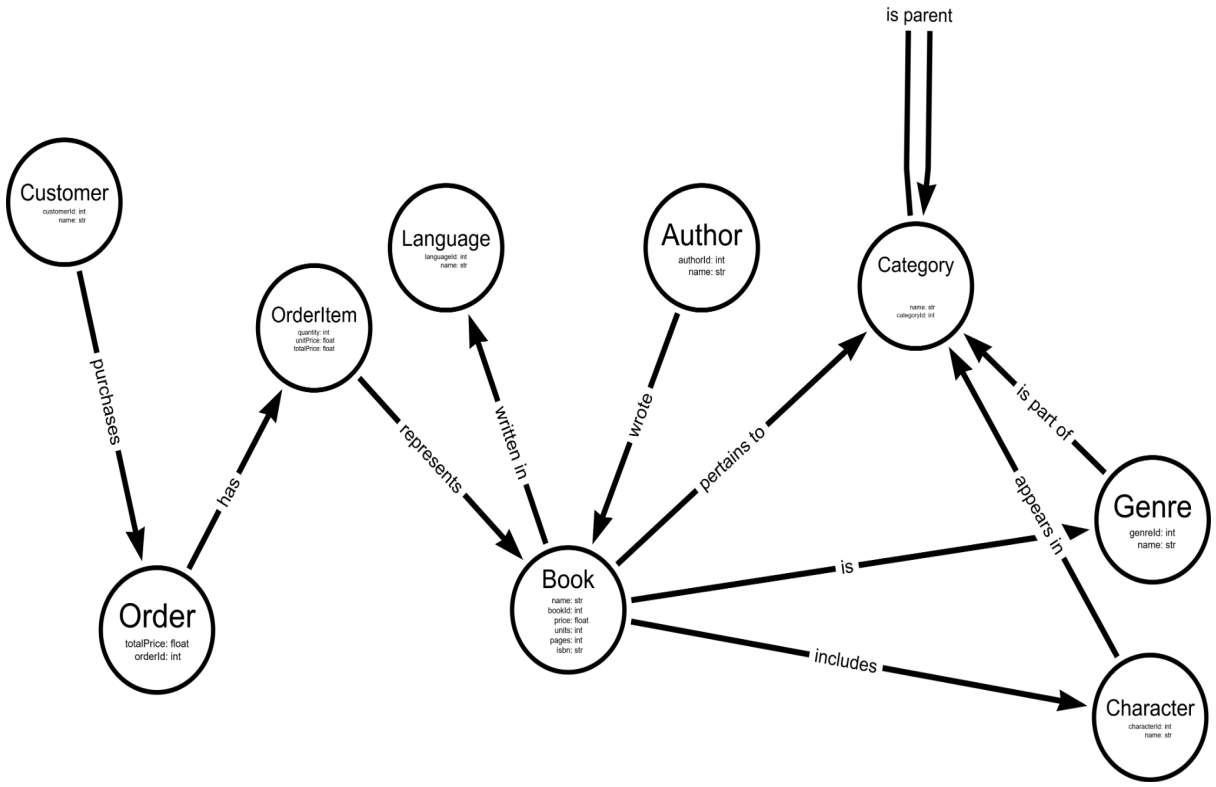
VIA University College

NSQ1 S22 Course Assignment 3

Arnau Molins Carbelo - 313294
Xavier Nadal Reales - 313287
Martí La Rosa Ramos - 313289

Question 1

We used the following model for our graph database:



Question 2

For executing the scripts it is necessary to have installed Apoc and Graph Data Science Library.

- Check **`build_populate.cypher`** for database populating.
- Check **`modify.cypher`** for modifying data.
- Check **`queries.cypher`** for querying data.

Question 3

Similar Customers

We are trying to find similar customers based on their purchase decisions to help with suggestions for next book to purchase.

- For classifying the customers based on their purchases, we could apply a community detection algorithm.
- First, we need to create a graph with Book and Customer nodes. Then we link each other based on if the customer has purchased the book.

```
CALL gds.graph.create.cypher('customer-book-purchase',
"MATCH (n)
WHERE n:Customer or n:Book
RETURN id(n) AS id, labels(n) AS labels",
"
 MATCH
(c:Customer)-[p:PURCHASES]->(o:Order)-[h:HAS]->(oi:OrderItem)-[r:REPR
ESENTS]->(b:Book)
 RETURN id(c) AS source, id(b) AS target, 'HAS_PURCHASED_BOOK' as
type
")
YIELD graphName AS graph, nodeQuery, nodeCount AS nodes,
relationshipQuery, relationshipCount AS rels
```

- By running a community algorithm with this graph we would get communities based on book purchases. An alternative would be a graph made of customers and categories/genres/characters of the purchased books.
- We execute Louvain algorithm for community detection.

```
CALL gds.louvain.stream('customer-book-purchase')
YIELD nodeId, communityId, intermediateCommunityIds
RETURN gds.util.asNode(nodeId).name AS name, communityId,
intermediateCommunityIds
ORDER BY name ASC
```

- Having tried it with our database, the results made sense. The customers that purchased the same books belonged to the same community.

Key Customers

With the system above in place, it becomes important to sell to key customers so that they can drive further sales.

```
CALL gds.graph.create.cypher('order-purchased-by-customer',
"MATCH (n)
WHERE n:Customer or n:Order
RETURN id(n) AS id, labels(n) AS labels",

"
 MATCH (c:Customer)-[p:PURCHASES]->(o:Order)
 RETURN id(o) AS source, id(c) AS target, 'PURCHASED_BY' as type
")
YIELD graphName AS graph, nodeQuery, nodeCount AS nodes,
relationshipQuery, relationshipCount AS rels
```

- With this graph we have that each order is linked to its purchaser (customer). The difference of the original graph in the database is that originally the customer points to the order, but now the order points to the customer. We are doing this because the centrality algorithm we used (PageRank), the more incoming relationships a Customer has, the more central (important) is.

```
CALL gds.pageRank.stream('order-purchased-by-customer')
YIELD nodeId, score
RETURN gds.util.asNode(nodeId).name AS name, score
ORDER BY score DESC, name ASC
```

Book Suggestions

How would you use the Graph Data Science library to give book suggestions? Explain the choice of graph and algorithm and show the code.

- In this case we want to predict a possible relationship between customer and book. We can use a topological link prediction algorithm for achieving that.

```
MATCH (c:Customer {name: 'Miquel'})
MATCH (b:Book)
```

```
RETURN gds.alpha.linkprediction.adamicAdar(c, b,
{relationshipQuery:
 "
 MATCH
(c:Customer)-[p:PURCHASES]->(o:Order)-[h:HAS]->(oi:OrderItem)-[r:REPR
ESENTS]->(b:Book)
 RETURN id(c) AS source, id(b) AS target, 'HAS_PURCHASED_BOOK' as
type
"
}) AS score, b.name as book
```

Question 4

What were the decisions taken in the modeling?

In the process of modeling the graph, we had to decide on the nodes, and most importantly, the relations the nodes would have between them, according to both the idea we had for the amazon database and a suitable model for querying later on.

The relations are all unidirectional. A customer purchases an order that has orderItems. These orderItems represent the purchased Books of an Order. These books are written in certain languages, and pertain to and are of a certain Category and Genre. They also include characters. These characters also have a relation of appearing in a Category, and the Genre are part of a Category as well. A Category can be a parent of another Category. This is a distinct feature from the past models, as now the category points to its child, and not to its parent as before. Finally, one or more authors wrote a certain book. That is the general idea behind the model.

For the attributes, we reduced a few of the ones we had on the past models, as we didn't really need them.

Why were these decisions taken?

This model is structured in a way that we think is really intuitive for everyone to understand, represents the idea behind every process in the bookstore in a logical manner and is clear and understandable when updating and querying.

What were the consequences of these decisions?

Following the intuitive manner of the model, the queries result to be simple and understandable. Given that the direction of the category relation is changed from past models, it now points to its child, when in the past it pointed to its parent, the queries are adapted in that way.

What were the difficult and easy parts of the exercise?

We believe that overall, the question about data science was the most confusing. Overall, graphs, neo4j and cypher were really interesting for all of us, so question 1 and 2 did not feel especially difficult.

Moreover, we had some problems populating the database. When using MERGE of two nodes linked to a relationship, the nodes duplicated. The solution was first MERGE the nodes separately and associate them in a variable and finally MERGE the variables with the corresponding relationship.

How does that compare to the other exercises?

We think this is the assignment we enjoyed the most. The technology feels intuitive and useful, and we already had the concepts of databases and the model of the bookstore from the past exercises.

What are the advantages and disadvantages of graph databases compared to the other database types?

Graph databases are very flexible as well, since you are able to add and drop new nodes and relations, as well as their attributes to change the data model easily. Additionally, they can be easier to visualize and more comfortable to query.

As a disadvantage, it lacks capacity for dealing with transactions and speaking of user-base, it isn't as much mature as other types of databases, so it can be pretty hard to find support.