Week 12 Assignment

Charley Ferrari

For this assignment, I have chosen to use Neo4J to store bilateral global trade data. In the included CSV, you can see a sample of the sort of data I'm looking at. Each row in the CSV includes an export country, an import country, the commodity group, and the dollar amount traded in 2013 (I only chose 2013 but this can easily be expanded to include trade as a time series.)

I used the below statements to load the data into Neo4J:

* load csv with headers from "file:C:/Users/Charley/Downloads/Courses/CUNY/SPS/Git/IS 607 Data Acquisition and Management/Week 12/Week12Assignment/countryList.csv" as countries create (a1:Country {name: countries.x})
* load csv with headers from "file:C:/Users/Charley/Downloads/Courses/CUNY/SPS/Git/IS 607 Data Acquisition and Management/Week 12/Week12Assignment/trade.csv" as trade match (ex: Country {name : trade.exportCountry}), (im: Country {name : trade.importCountry}) create (ex) -[r:Exported {Product: trade.Product, Amount: trade.Trade}]-> (im)

I chose to include only countries as nodes, and have each relationship include a particular product. The number of relationships would be multiplied by the number of years included if I decided to expand this to include trade over time.

If I were to display all relationships and nodes, the view would not be too informative. I would see all my countries, but would not be able to differentiate between the different types of relationships between them. When I click on the relationship, I only see one arbitrary relationship type.

I see this database as mainly being used for queries. Simpler queries, such as analyzing the exports of one country to another by product type, could be achieved with a query similar to the one below:

* match(a: Country{name: 'Brazil'})-[r]->(b: Country{name: 'Argentina'}) return (a.name), (r.Product), (r.Amount), (b.name)

In this case, I’m displaying exports from Brazil to Argentina by product type.

The real strength of this database however is in calculating flows via aggregation. See the below query, which calculates the trade flow between Brazil and Argentina:

* match(a:Country{name: 'Argentina'})<-[r]->(b:Country{name: 'Brazil'}) return sum((case startnode(r).name when 'Argentina' then 1 else -1 end)\*r.Amount)

In this case, I’m using a double arrow to get all directions of trade between Argentina and Brazil. In my return statement, I’m defining the direction of trade by using a case based on what the startnode is, and assigning 1 and -1 as my directions. Then, I multiply that direction by the dollar amount traded to get an aggregate balance.

I could further query within the arrow to get various types of balance. If I only wanted balance for a certain product type for example, I could enter that in the arrow. Trade flows can be difficult to represent in a relational database since the direction of trade matters in aggregation, but in Neo4J you’re able to pretty easily calculate any sort of trade balance you want.