CUNY MSDA - IS607 : DATA ACQUISITION AND MANAGEMENT

Week 13 - Project 5

A. Leopold HILLAH

##### 1. Exporting the flights data from `nycflights13 R` package into CSV files #####

```
# Exporting data from R into CSV

currdir <- getwd()

library('nycflights13')

# Identifying packages linked to nycflights package

data(package='nycflights13')


Data sets in package ‘nycflights13’:

airlines                                 Airline names.
airports                                 Airport metadata
flights                                  Flights data
planes                                   Plane metadata.
weather                                  Hourly weather data


# Loading data from identified data sets

data(airlines, airports, flights, planes, weather)

# Writing a CSV file for each data set except the weather data set

write.csv(airlines, file = "airlines.csv",row.names=FALSE, na="")
write.csv(airports, file = "airports.csv",row.names=FALSE, na="")
write.csv(flights, file = "flights.csv",row.names=FALSE, na="")
write.csv(planes, file = "planes.csv",row.names=FALSE, na="")
```

##### 2. Procedure to import the flights data using CYPHER #####

```
## 1. Create the nodes

# Create airlines
LOAD CSV WITH HEADERS FROM "file:F:/Neo4j/airlines.csv" AS row
CREATE (airline:Airline {Carrier: row.Carrier, Name: row.Name});

# Create airports
LOAD CSV WITH HEADERS FROM "file:F:/Neo4j/airports.csv" AS row
CREATE (airport:Airport {Faa: row.Faa, Name: row.Name, Lat: row.Lat, Lon:
row.Lon, Alt: row.Alt, Tz: row.Tz, Dst: row.Dst});

# Create planes
LOAD CSV WITH HEADERS FROM "file:F:/Neo4j/planes.csv" AS row
CREATE (plane:Plane {Tailnum: row.Tailnum, Year: row.Year, Type: row.Type,
Manufacturer: row.Manufacturer, Model: row.Model, Engines: row.Engines,
Seats: row.Seats, Speed: row.Speed, Engine: row.Engine});

## 2. Create indexes on the nodes


CREATE INDEX ON :Airline(Carrier);
CREATE INDEX ON :Airport(Faa);
CREATE INDEX ON :Plane(Tailnum);

## 3. Create flights and its relationships to airline, airport and plane

# Create flights and relationships

LOAD CSV WITH HEADERS FROM "file:F:/Neo4j/flights.csv" AS row
CREATE (flight:Flight {Year: row.Year, Month: row.Month, Day: row.Day,
Dep_time: row.Dep_time, Dep_delay: row.Dep_delay, Arr_time: row.Arr_time,
Arr_delay: row.Arr_delay,
Carrier: row.Carrier, Tailnum: row.Tailnum, Flight: row.Flight, Origin:
row.Origin, Dest: row.Dest, Air_time: row.Air_time, Distance: row.Distance,
Hour: row.Hour, Minute: row.Minute})

WITH *
MATCH (airline:Airline {Carrier: row.Carrier})
MATCH (plane:Plane {Tailnum: row.Tailnum})
MATCH (origin:Airport {Faa: row.Origin})
MATCH (destination:Airport {Faa: row.Dest})

MERGE (flight)-[:BELONGS]->(airline)
MERGE (flight)-[:PART_OF]->(plane)
MERGE (flight)-[:DEPARTS]->(origin)
MERGE (flight)-[:ARRIVES]->(destination);
```

##### 3. Simple CYPHER query on the data #####

```
MATCH (plane:Plane) WHERE plane:Manufacturer = "EMBRAER" RETURN plane;
```

##### 4. Advantages and disavantages of having this information in a graph database   #####

This data is well structured and may be suitably be stored in a relational database for multi-user access and security as well as interface building using standard market tools. So these are definitely disadvantages when it is stored in a graph database such as Neo4j. Moreover, graph databases store the relationships at the record level and so may end up using more storage than relational databases because of the duplications. As the data volume increases, it will not be practical to easily reshuffle the data stored in a graph databases without the operation taking a long time and without leaving dead pointers.

On the other hand, storing the data in a graph database offers advantages of flexibility, ease of querying and better accommodation for lack of structure or complex structure in the data. It also imposes a network-like structure on the data, which may bring more meaning to it, especially when the data is well connected. Graph traversal may be very fast when accessing the data. Graph databases make development faster as modeling and querying are simpler.