Neo4j Query Lab

Instructions

In this hands-on lab, we will explore basic querying in Neo4j. We will work with a slightly expanded version of our flight data set. In particular, we'll be using two data sets described here:

Flight Data

flight:	the flight identification number (this is unique)
airline:	the name of the airline
depart:	the departure airport code
arrive:	the arrival airport code
capacity:	passenger capacity
takeoff:	takeoff time (HHMM in military format)
landing:	landing time (HHMM in military format)

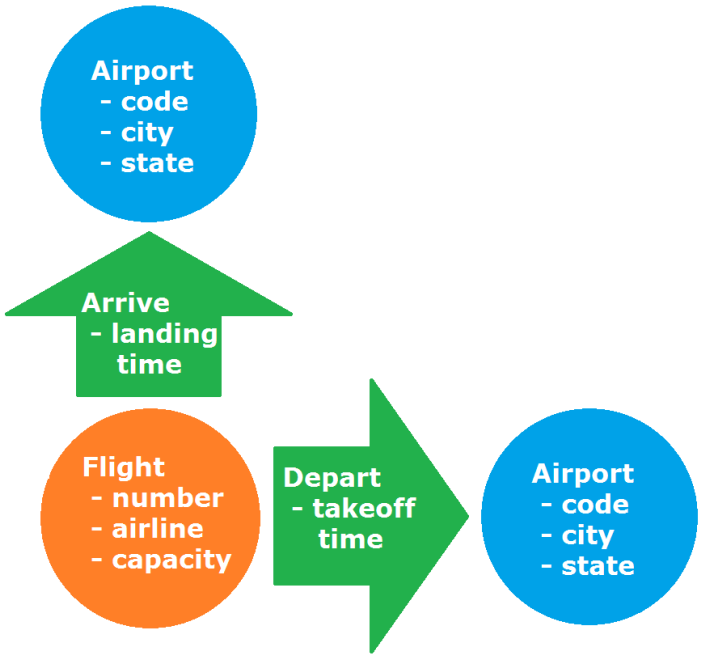
Airport Data

label:	the three-character airport code (unique)
city:	the airport's nominal city
state:	the airport's nominal state

Note that the airport data set is identical to what we used before.

Part I: Data Preparation

We will use previously studied techniques to load the data into Neo4j. In particular, we wish to end up with a data model as follows:



Complete the following steps to load the data into Neo4j:

1. Make sure your database is empty by issuing the query “match (n) return (n)” and checking that no nodes are found.

   No answer for this one… If you need to empty a database that has nodes and relationships, you can use the query:

   ```
   match (a) optional match (a)-[r]-() delete a,r
   ```

2. Use Load CSV to create the airport nodes. Note that each airport node should have a label of “Airport” and a set of three attributes: code, city, and state.

   ```
   load csv with headers from "file:C:/Users/Michael/OneDrive/CUNY/neo4j-airport-csv-raw.csv" as airports
   create (a1:Airport {label: airports.label, city: airports.city, state: airports.state})
   ```

3. Use Load CSV to create the flight nodes. Each flight node should have a label of “Flight” and should have three attributes: number, capacity, and airline.

   ```
   load csv with headers from "file:C:/Users/Michael/OneDrive/CUNY/neo4j-flight-lab-data.csv" as flights
   create (n:Flight {number: flights.flight, airline: flights.airline, capacity: flights.capacity})
   ```

4. Use Load CSV to create the arrival relationships. Each arrival relationship should have a label of “Arrive” and should have a single attribute: landing. (This attribute gives the scheduled landing time.)

   ```
   load csv with headers from "file:C:/Users/Michael/OneDrive/CUNY/neo4j-flight-lab-data.csv" as flights
   match (a: Flight {number: flights.flight}),(b:Airport {label: flights.arrive}) create (a) -[r:Arrives {landing: flights.landing}]-> (b)
   ```

5. Use Load CSV to create the departure relationships. Each departure relationship should have a label of “Depart” and should have a single attribute: takeoff. (This attribute gives the scheduled takeoff time.)

   ```
   load csv with headers from "file:C:/Users/Michael/OneDrive/CUNY/neo4j-flight-lab-data.csv" as flights
   match (a: Flight {number: flights.flight}),(b:Airport {label: flights.depart}) create (a) -[r:Departs {takeoff: flights.takeoff}]-> (b)
   ```

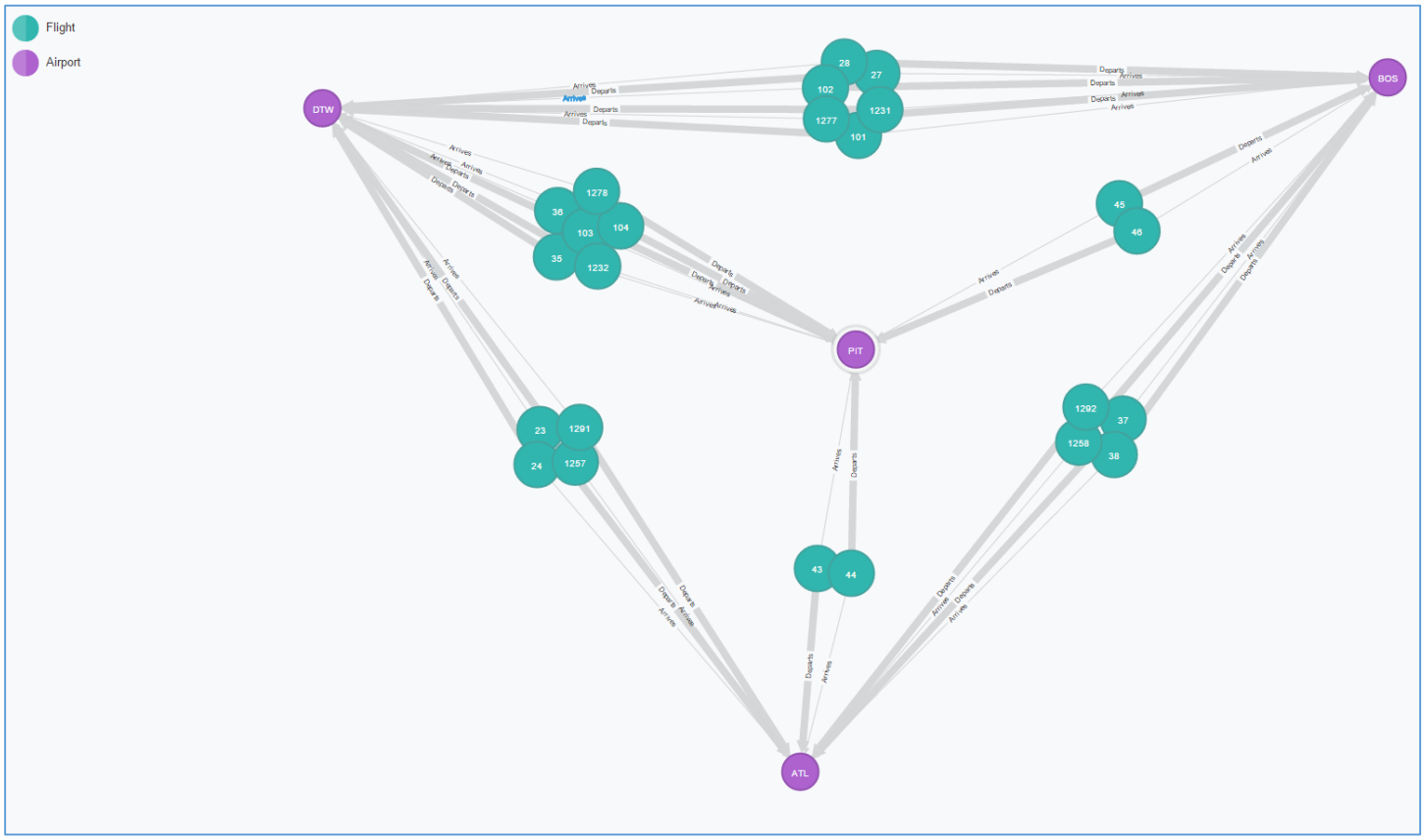
6. Run the query

   ```
   match (n) return (n)
   ```

   to ensure that the data have loaded correctly. You should see four airport nodes, 24 flight nodes, and the various attributes assigned correctly. Organize the graph in a way that makes these features clear.

   No answer for this one either…

You should see a graph that looks something like this. (I've put Pittsburgh in the middle and Detroit, Boston, and Atlanta around the outside. Your graph may look a bit different, depending on your layout choices.)



Part II: Data Analysis

Now that we have some data loaded, let's answer some basic questions. Write Cypher queries to answer each of the following:

7. Write a query that returns all airports.

   match (n:Airport) return n

8. Write a query that returns a count of all airports.

   match (n:Airport) return count(n)

9. Write a query that returns all Delta flights.

   match (f:Flight {airline: "Delta"}) return f

10. Return all flights that depart from Boston.

    match (f:Flight)-[d:Departs]-(a:Airport {label: 'BOS'}) return f

11. Return all flights that run from Detroit to Atlanta.

    match (b:Airport {label: 'ATL'})-[r:Arrives]-(f:Flight)-[d:Departs]-(a:Airport {label: 'DTW'}) return f

12. Return all flights that take off before 11 a.m.

    match (f:Flight) -[d:Departs]-() where toInt(d.takeoff) < 1100 return f

13. Return all flights with a capacity greater than 150 passengers.

    match (f:Flight) where toInt(f.capacity) > 150 return f

14. Return all flights on Delta that arrive in Boston.

    match (f:Flight {airline: 'Delta'}) -[r:Arrives]-(a:Airport {label: 'BOS'}) return f

    You could also use the following:

    match (f:Flight) -[r:Arrives]-(a:Airport) where f.airline = 'Delta' and a.label = 'BOS' return f

Part III: Data Modification

Let's make a few simple changes to our data. Perform the following tasks:

15. Suppose the Pittsburgh airport designation code has been changed from PIT to PGH. Write a query to update the airport's code.

    match (a:Airport {label: 'PIT'}) set a.label = 'PGH' return a

16. Delta has decided to renumber some of its flights. Write a query to change flight 28 to flight 29.

    match (f:Flight {number: '28'}) set f.number = '29' return f

17. Southwest has cancelled all flights between Detroit and Boston. Write a query that removes all such flights.

    You can remove flights in each direction with separate queries as follows:

    match (b:Airport {label: 'BOS'})-[r:Arrives]-(f:Flight {airline: 'Southwest'})-[d:Departs]-(a:Airport {label: 'DTW'}) delete f,r,d

    match (b:Airport {label: 'DTW'})-[r:Arrives]-(f:Flight {airline: 'Southwest'})-[d:Departs]-(a:Airport {label: 'BOS'}) remove f,r,d

Alternatively, you can remove all of them at once with:

```
match (b:Airport {label: 'BOS'})-[r]-(f:Flight {airline: 'Southwest'})-[d]-(a:Airport {label: 'DTW'})
delete f,r,d
```

Notice that this single query sets up the basic pattern but does not specify which relationship is Arrives and which is Departs.

Part IV: Clean Up

Once all of the lab has been completed, you may wish to remove the data from the database. What query removes all nodes and relationships from the database?