Dec 2023

AIRPORT ROUTES

Neo4j

M. Ahmad - M. Zeinidine - F. Chalhoub - M. Hussein

University of Saint Joseph

Contents

Abstract
This project report presents the process and findings of visualizing an airport network using the Neo4j graph database. The dataset comprises 3,504 airports and 46,400 routes, offering a comprehensive view of global connectivity. Through a series of Cypher queries, we explored various aspects of the network, such as the busiest airports, potential new routes, and average distances. This report details the dataset, the rationale behind using Neo4j, the queries executed, and the insights derived from the data.

Introduction

- Background: In the age of global connectivity, understanding the complex network of airport routes is crucial for enhancing operational efficiency and passenger experience. Graph databases provide an intuitive way to represent and analyze these networks.
- Project Objective: To effectively visualize and analyze a large-scale airport network to derive meaningful insights using the advanced graph database capabilities of Neo4j.
- Scope: The project is confined to the visualization and query analysis of airport networks, specifically focusing on route connectivity, airport activity, and geographic distributions.

Overview of Neo4j

- Introduction to Neo4j: Neo4j is known for its graph database capabilities, particularly advantageous for modeling complex, connected data. Its ability to represent graphs and query relationships is unparalleled, making it an ideal choice for visualizing our dataset.
- Neo4j vs. Other Databases: Unlike traditional relational databases, Neo4j excels in scenarios where relationships can be deeply nested and complex, offering superior performance and flexibility for queries that traverse these relationships.

About Our Dataset

- Data Source: The dataset was sourced from the Neo4j-graph-examples repository, which provides a list of airport nodes and route edges.
- Data Description: We selected the dataset because of its structured format and comprehensive data, which aligns well with Neo4j's capabilities. It includes

detailed information about airports and routes, with 3,504 airports and 46,400 routes available for analysis.

id	iata	icao	city	descr	region	runways	longest	altitude	country	continent	lat	lon
1	ATL	KATL	Atlanta	Hartsfield	US-GA	5	12390	1026	US	NA	33.6367	-84.4281
2	ANC	PANC	Anchorag	Anchorag	US-AK	3	12400	151	US	NA	61.1744	-149.996
3	AUS	KAUS	Austin	Austin Be	US-TX	2	12250	542	US	NA	30.1945	-97.6699
4	BNA	KBNA	Nashville	Nashville	US-TN	4	11030	599	US	NA	36.1245	-86.6782
5	BOS	KBOS	Boston	Boston Lo	US-MA	6	10083	19	US	NA	42.3643	-71.0052
6	BWI	KBWI	Baltimore	Baltimore	US-MD	3	10502	143	US	NA	39.1754	-76.6683
7	DCA	KDCA	Washingt	Ronald Re	US-DC	3	7169	14	US	NA	38.8521	-77.0377
8	DFW	KDFW	Dallas	Dallas/Fo	US-TX	7	13401	607	US	NA	32.8968	-97.038
9	FLL	KFLL	Fort Laud	Fort Laud	US-FL	2	9000	64	US	NA	26.0726	-80.1527
10	IAD	KIAD	Washingt	Washingt	US-VA	4	11500	313	US	NA	38.9445	-77.4558
11	IAH	KIAH	Houston	George Bu	US-TX	5	12001	96	US	NA	29.9844	-95.3414
12	JFK	KJFK	New York	New York	US-NY	4	14511	12	US	NA	40.6398	-73.7789
13	LAX	KLAX	Los Angel	Los Angel	US-CA	4	12091	127	US	NA	33.9425	-118.408
14	LGA	KLGA	New York	New York	US-NY	2	7003	20	US	NA	40.7772	-73.8726
15	MCO	KMCO	Orlando	Orlando Ir	US-FL	4	12005	96	US	NA	28.4294	-81.309
16	MIA	KMIA	Miami	Miami Int	US-FL	4	13016	8	US	NA	25.7932	-80.2906
17	MSP	KMSP	Minneapc	Minneapc	US-MN	4	11006	841	US	NA	44.882	-93.2218
18	ORD	KORD	Chicago	Chicago O	US-IL	7	13000	672	US	NA	41.9786	-87.9048
19	PBI	KPBI	West Paln	Palm Beac	US-FL	3	10000	19	US	NA	26.6832	-80.0956
20	PHX	KPHX	Phoenix	Phoenix S	US-AZ	3	11489	1135	US	NA	33.4343	-112.012
21	RDU	KRDU	Raleigh	Raleigh-D	US-NC	3	10000	435	US	NA	35.8776	-78.7875

src	dest	dist
ATL	AUS	811
ATL	BNA	214
ATL	BOS	945
ATL	BWI	576
ATL	DCA	546
ATL	DFW	729
ATL	FLL	581
ATL	IAD	533
ATL	IAH	688
ATL	JFK	759
ATL	LAX	1941
ATL	LGA	761
ATL	MCO	404
ATL	MIA	596
ATL	MSP	906
ATL	ORD	606
ATL	PBI	545
ATL	PHX	1583

- Data Preparation: We performed data cleaning and transformation to conform to the requirements of Neo4j, ensuring attributes like 'id' and 'distance' were correctly typed. To use the data, we applied the following queries to load it, add some indices and match the data from both csv files.

```
LOAD CSV WITH HEADERS FROM 'file:///airport-node-list.csv' AS row
CREATE (a:Airport {
  id: toInteger(row.id),
  iata: row.iata,
  icao: row.icao,
  city: row.city,
  descr: row.descr,
  region: row.region,
  runways: toInteger(row.runways),
  longest: toInteger(row.longest),
  altitude: toInteger(row.altitude),
  country: row.country,
  continent: row.continent,
  latitude: toFloat(row.lat),
  longitude: toFloat(row.lon)
})

CREATE (country:Country {name: a.country})
CREATE (continent:Continent {name: a.continent})
CREATE (city:City {name: a.city})
CREATE (region:Region {name: a.region})

CREATE (a)-[:IN_CITY]->(city)
CREATE (a)-[:IN_COUNTRY]->(country)
CREATE (city)-[:IN_COUNTRY]->(country)
CREATE (region)-[:IN_COUNTRY]->(country)
CREATE (a)-[:IN_REGION]->(region)
CREATE (city)-[:IN_REGION]->(region)
CREATE (a)-[:ON_CONTINENT]->(continent)
CREATE (city)-[:ON_CONTINENT]->(continent)
CREATE (country)-[:ON_CONTINENT]->(continent)
CREATE (region)-[:ON_CONTINENT]->(continent)
```

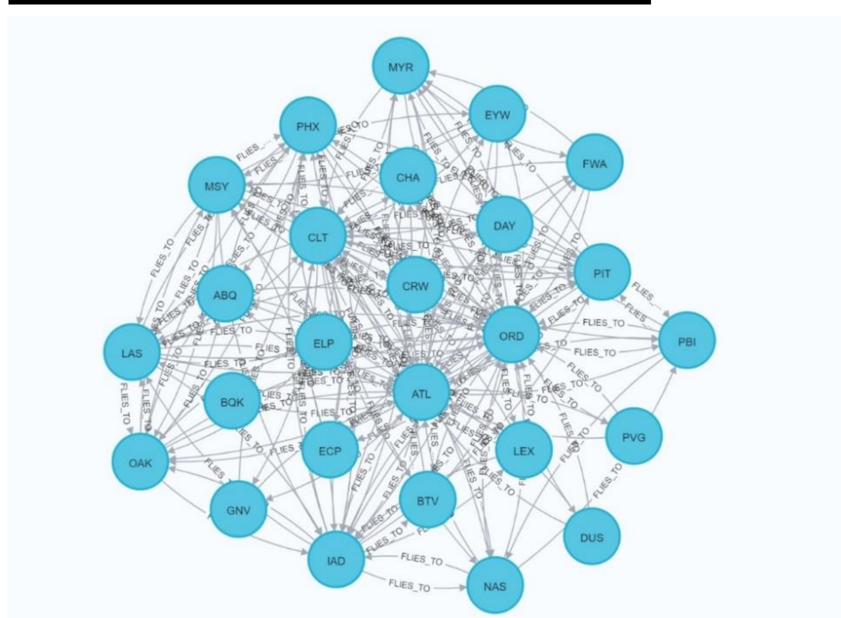
```
CREATE INDEX FOR (n:Airport) ON (n.iata);
CREATE INDEX FOR (n:Airport) ON (n.icao);
```

```
LOAD CSV WITH HEADERS FROM
'file:///iroutes-edges.csv' AS row
MATCH (
srcAirport:Airport {iata: row.src}),
(destAirport:Airport {iata: row.dest})
CREATE (srcAirport)-[:FLIES_TO {distance:
toInteger(row.dist)}]->(destAirport)
```

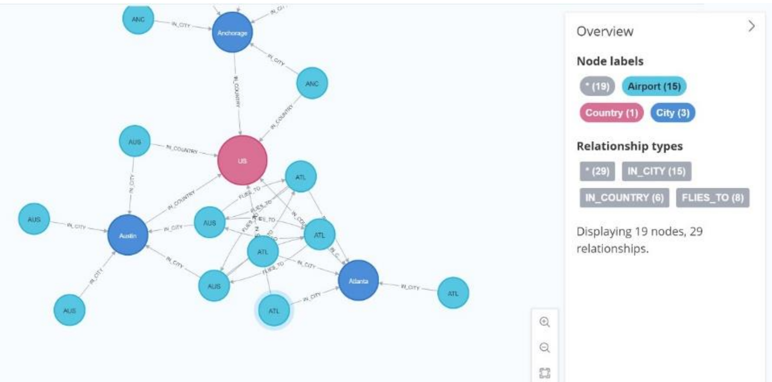
- **Data Visualization:**

To initially understand the data and test if it was imported successfully, we applied some basic queries to visualize the dataset and practice cypher on neo4j a bit.

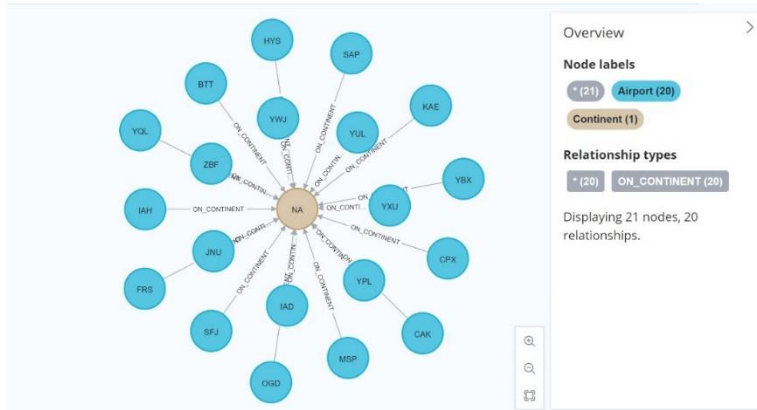
```
MATCH (a:Airport)-[r:FLIES_TO]->(b:Airport)
RETURN a, r, b LIMIT 25
```



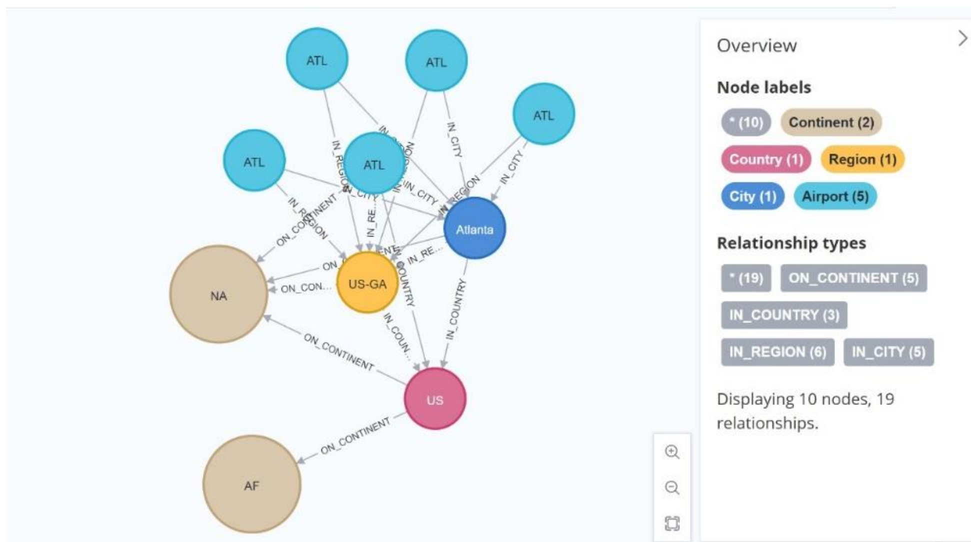
```
MATCH (a:Airport)-[r:ON_CONTINENT]->(con:Continent)

RETURN a, r, con
```




```
MATCH (a:Airport)-[r:IN_CITY]->(n:City)-[:IN_COUNTRY]-(m:Country)

RETURN a, r, n, m
```




```
MATCH (con:Continent)<-[:ON_CONTINENT]-(co:Country)<-[:IN_COUNTRY]-(r:Region)<-[:IN_REGION]-(ci:City)<-[:IN_CITY]-(a:Airport)

RETURN con, co, r, ci, a

LIMIT 10;
```



Applying Queries in Neo4j

- **Overview of Queries Used:** The process of executing queries was a significant learning curve due to unfamiliarity with the Cypher query language. Despite this challenge, we successfully ran a series of queries, which included identifying the busiest airports, potential new routes, and analyzing connectivity within and between continents.
- **Query Implementation:** The following are the queries we applied on our dataset, their prompt, and their result.

To find the busiest airports in your Neo4j database, we need to identify the airports with the most incoming and outgoing flights. This can be determined by counting the number of relationships each airport node has with other airport nodes. In your database schema, this would be the FLIES_TO relationship

```
MATCH (a:Airport)-[f:FLIES_TO]->()
WITH a, count(f) AS numberOfFlights
ORDER BY numberOfFlights DESC
RETURN a.iata, a.city, numberOfFlights
LIMIT 10;
```

```
neo4j$ MATCH (a:Airport)-[f:FLIES_TO]→() WITH a, count(f) AS numberOfFlights ORDER BY numberOfFlights DE…
```

	a.iata	a.city	numberOfFlights
1	"IST"	"Istanbul"	1842
2	"FRA"	"Frankfurt"	1842
3	"CDG"	"Paris"	1758
4	"AMS"	"Amsterdam"	1692
5	"MUC"	"Munich"	1620
6	"ORD"	"Chicago"	1584
7			

Started streaming 10 records after 14 ms and completed after 350 ms.

To identify potential new routes based on missing connections in your Neo4j database, you can look for pairs of airports that don't currently have a direct flight between them but are connected to common airports. This might suggest a demand for a direct route.

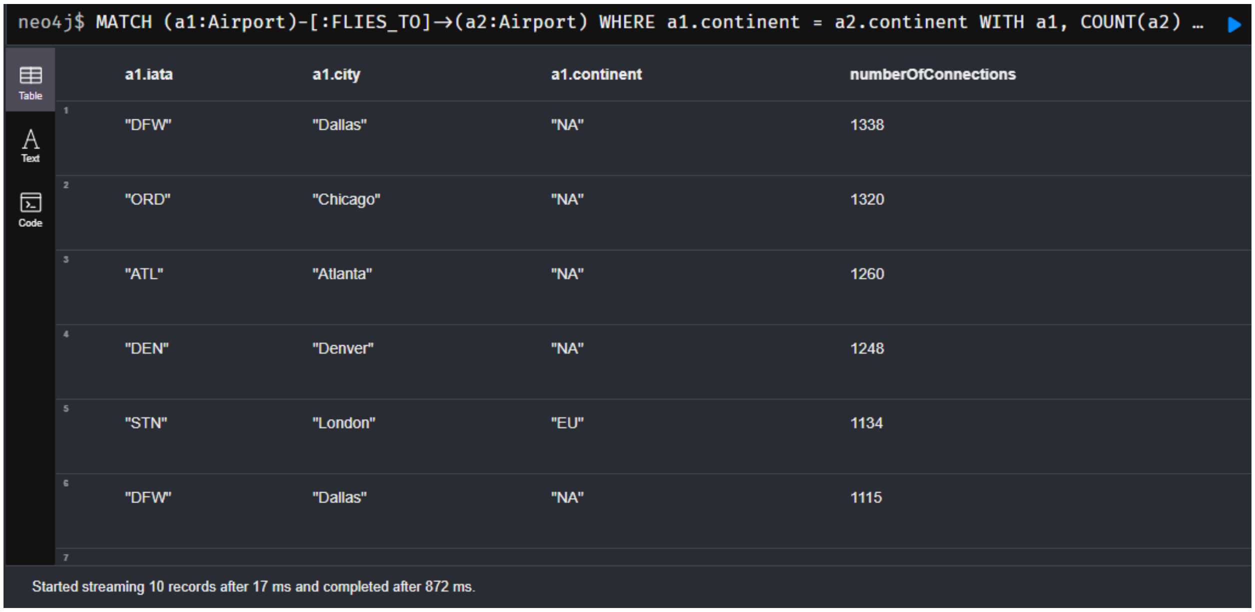
```
MATCH (srcAirport:Airport)-[:FLIES_TO]->(commonAirport:Airport)<-[:FLIES_TO]-(destAirport:Airport)
WHERE NOT (srcAirport)-[:FLIES_TO]->(destAirport)
AND srcAirport <> destAirport
WITH srcAirport, destAirport, COUNT(commonAirport) AS commonConnections
ORDER BY commonConnections DESC
RETURN srcAirport.iata, destAirport.iata, commonConnections
LIMIT 10;
```

	srcAirport.iata	destAirport.iata	commonConnections
1	"IST"	"IST"	3377
2	"FRA"	"FRA"	3377
3	"FRA"	"FRA"	3377
4	"IST"	"IST"	3377
5	"CDG"	"CDG"	3223
6	"CDG"	"CDG"	3223
7			

Started streaming 10 records after 22 ms and completed after 139436 ms.

To analyze airport connectivity within a specific continent in your Neo4j database, we can create a query that counts the number of direct flight connections each airport has within the same continent. This will give us an insight into how well-connected different airports are within a continent.

```
MATCH (a1:Airport)-[:FLIES_TO]->(a2:Airport)
WHERE a1.continent = a2.continent
WITH a1, COUNT(a2) AS numberOfConnections
ORDER BY numberOfConnections DESC
RETURN a1.iata, a1.city, a1.continent, numberOfConnections
LIMIT 10;
```



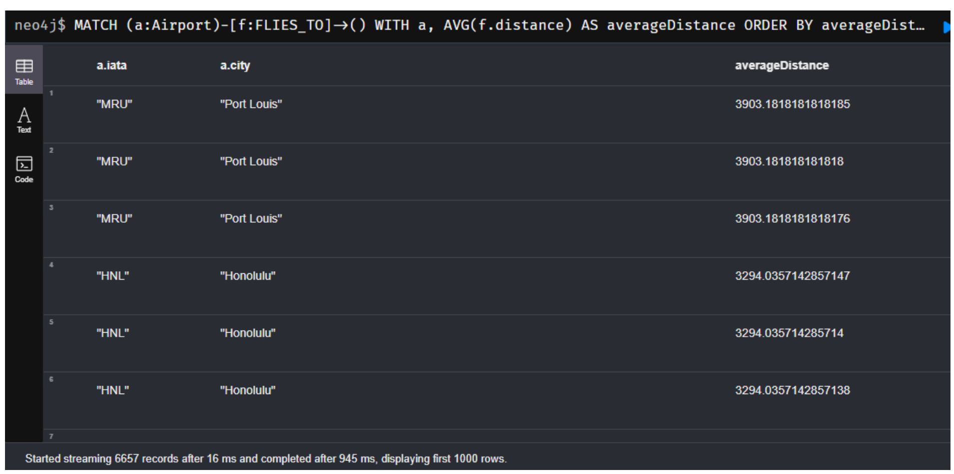
neo4j$ MATCH (a1:Airport)-[:FLIES_TO]→(a2:Airport) WHERE a1.continent = a2.continent WITH a1, COUNT(a2) …

	a1.iata	a1.city	a1.continent	numberOfConnections
1	"DFW"	"Dallas"	"NA"	1338
2	"ORD"	"Chicago"	"NA"	1320
3	"ATL"	"Atlanta"	"NA"	1260
4	"DEN"	"Denver"	"NA"	1248
5	"STN"	"London"	"EU"	1134
6	"DFW"	"Dallas"	"NA"	1115
7				

Started streaming 10 records after 17 ms and completed after 872 ms.

To calculate the average distance of routes per airport in your Neo4j database, you can use a Cypher query that aggregates the distances of all flights departing from each airport and then computes the average

```
MATCH (a:Airport)-[f:FLIES_TO]->()
WITH a, AVG(f.distance) AS averageDistance
ORDER BY averageDistance DESC
RETURN a.iata, a.city, averageDistance;
```



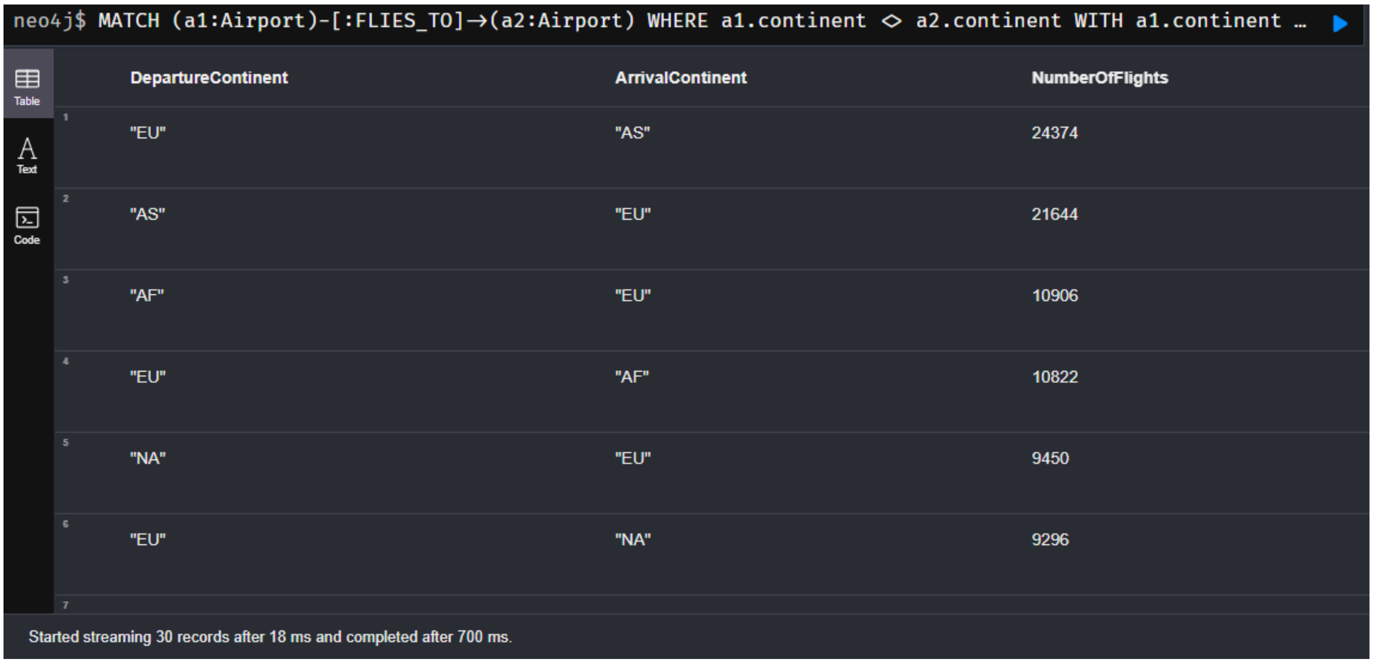
neo4j$ MATCH (a:Airport)-[f:FLIES_TO]→() WITH a, AVG(f.distance) AS averageDistance ORDER BY averageDist…

	a.iata	a.city	averageDistance
1	"MRU"	"Port Louis"	3903.1818181818185
2	"MRU"	"Port Louis"	3903.181818181818
3	"MRU"	"Port Louis"	3903.1818181818176
4	"HNL"	"Honolulu"	3294.0357142857147
5	"HNL"	"Honolulu"	3294.035714285714
6	"HNL"	"Honolulu"	3294.0357142857138
7			

Started streaming 6657 records after 16 ms and completed after 945 ms, displaying first 1000 rows.

To discover direct connections between continents in your Neo4j database, you can write a Cypher query that finds flights linking airports on different continents. This involves matching Airport nodes that have a FLIES_TO relationship and are located on different continents. Here's how you can structure this query:

```
MATCH (a1:Airport)-[:FLIES_TO]->(a2:Airport)

WHERE a1.continent <> a2.continent

WITH a1.continent AS DepartureContinent, a2.continent AS
ArrivalContinent, COUNT(*) AS NumberOfFlights

RETURN DepartureContinent, ArrivalContinent, NumberOfFlights

ORDER BY NumberOfFlights DESC;
```



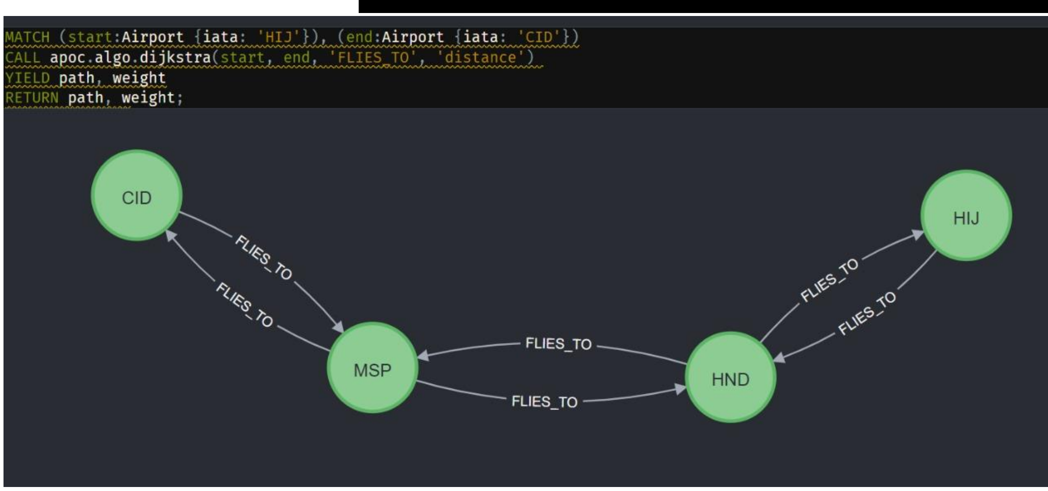
neo4j$ MATCH (a1:Airport)-[:FLIES_TO]→(a2:Airport) WHERE a1.continent <> a2.continent WITH a1.continent …

	DepartureContinent	ArrivalContinent	NumberOfFlights
1	"EU"	"AS"	24374
2	"AS"	"EU"	21644
3	"AF"	"EU"	10906
4	"EU"	"AF"	10822
5	"NA"	"EU"	9450
6	"EU"	"NA"	9296
7			

Started streaming 30 records after 18 ms and completed after 700 ms.

A sample Cypher query for finding the shortest path between two airports using Dijkstra's algorithm with APOC

```
MATCH (start:Airport {iata: 'HIJ'}), (end:Airport {iata: 'CID'})

CALL apoc.algo.dijkstra(start, end, 'FLIES_TO', 'distance')

YIELD path, weight

RETURN path, weight;
```



To find airports with the most international connections in your Neo4j database, you can write a Cypher query that counts the number of distinct foreign countries each airport has flights to. This will help identify airports that serve as major international hubs.

```
MATCH (a:Airport)-[:FLIES_TO]->(b:Airport)
WHERE a.country <> b.country
WITH a.iata AS AirportCode, a.city AS City,
COUNT(DISTINCT b.country) AS InternationalConnections
ORDER BY InternationalConnections DESC
RETURN AirportCode, City, InternationalConnections
LIMIT 10;
```

```
neo4j$ MATCH (a:Airport)-[:FLIES_TO]→(b:Airport) WHERE a.country <> b.country WITH a.iata AS AirportCode, a.city AS …
```

AirportCode	City	InternationalConnections
"IST"	"Istanbul"	115
"CDG"	"Paris"	113
"FRA"	"Frankfurt"	102
"DXB"	"Dubai"	102
"AMS"	"Amsterdam"	93
"LHR"	"London"	87

Started streaming 10 records after 22 ms and completed after 1252 ms.

To find hub airports in each continent, we'll look for airports with the highest number of direct connections (either incoming or outgoing flights) within their respective continents. This can be done using a Cypher query in Neo4j, which aggregates the number of FLIES_TO relationships for each airport and groups them by continent.

```
MATCH (a:Airport)-[:FLIES_TO]->(b:Airport)
WHERE a.continent = b.continent
WITH a.continent AS Continent, a, COUNT(b) AS Connections
ORDER BY Connections DESC
RETURN Continent, a.iata AS AirportCode, a.city AS City, Connections
LIMIT 10;
```

```
neo4j$ MATCH (a:Airport)-[:FLIES_TO]→(b:Airport) WHERE a.continent = b.continent WITH a.continent AS Continent, a,
```

Continent	AirportCode	City	Connections
"NA"	"DFW"	"Dallas"	1338
"NA"	"ORD"	"Chicago"	1320
"NA"	"ATL"	"Atlanta"	1260
"NA"	"DEN"	"Denver"	1248
"EU"	"STN"	"London"	1134
"NA"	"DFW"	"Dallas"	1115

Started streaming 10 records after 29 ms and completed after 1334 ms.

Query for the Highest Airport

```
MATCH (a:Airport)
RETURN a.iata AS IATA, a.city AS City, a.altitude AS Altitude
ORDER BY a.altitude DESC
LIMIT 1;
```

```
neo4j$ MATCH (a:Airport) RETURN a.iata AS IATA, a.city AS City, a.altitude AS Altitude ORDER BY a.alt
```

	IATA	City	Altitude
1	"DCY"	"Daocheng"	14472

Started streaming 1 records after 11 ms and completed after 36 ms.

Query for the Lowest Airport

```
MATCH (a:Airport)
RETURN a.iata AS IATA, a.city AS City, a.altitude AS Altitude
ORDER BY a.altitude ASC
LIMIT 1;
```

```
neo4j$ MATCH (a:Airport) RETURN a.iata AS IATA, a.city AS City, a.altitude AS Altitude ORDER BY
```

	IATA	City	Altitude
1	"GUW"	"Atyrau"	-72

Started streaming 1 records after 10 ms and completed after 24 ms.

Query for the Longest Route

```
MATCH (a1:Airport)-[f:FLIES_TO]->(a2:Airport)
RETURN a1.iata AS DepartureIATA, a2.iata AS ArrivalIATA, f.distance AS Distance
ORDER BY f.distance DESC
LIMIT 1;
```

```
neo4j$ MATCH (a1:Airport)-[f:FLIES_TO]→(a2:Airport) RETURN a1.iata AS DepartureIATA, a2.iata AS ArrivalIA
```

	DepartureIATA	ArrivalIATA	Distance
1	"EWR"	"SIN"	9523

Started streaming 1 records after 15 ms and completed after 802 ms.

Query for the Shortest Route

```
neo4j$ MATCH (a1:Airport)-[f:FLIES_TO]→(a2:Airport) RETURN a1.iata AS DepartureIATA, a2.iata AS ArrivalIA
```

	DepartureIATA	ArrivalIATA	Distance
1	"PPW"	"WRY"	2

Started streaming 1 records after 11 ms and completed after 527 ms.

```
MATCH (a1:Airport)-[f:FLIES_TO]->(a2:Airport)
RETURN a1.iata AS DepartureIATA, a2.iata AS ArrivalIATA, f.distance AS Distance
ORDER BY f.distance ASC
LIMIT 1;
```

Average Number of Runways per Country

```
MATCH (a:Airport)
WITH a.country AS Country,
AVG(a.runways) AS AverageRunways
RETURN Country, AverageRunways
ORDER BY AverageRunways DESC;
```

```
neo4j$ MATCH (a:Airport) WITH a.country AS Country, AVG(a.runway
```

	Country	AverageRunways
1	"NZ"	3.0799999999999987
2	"TM"	3.0
3	"LB"	3.0
4	"IM"	3.0
5	"NE"	3.0
6	"UY"	2.5

Finding Isolated Airports

```
MATCH (a:Airport)
OPTIONAL MATCH (a)-[f:FLIES_TO]->()
WITH a, COUNT(f) AS numberOfConnections
WHERE numberOfConnections <= 1
RETURN a.iata AS IATA, a.city AS City, a.country AS Country, numberOfConnections
ORDER BY numberOfConnections ASC;
```

```
neo4j$ MATCH (a:Airport) OPTIONAL MATCH (a)-[f:FLIES_TO]→() WITH a, COUNT(f) AS numberOfConnections WHERE numb
```

	IATA	City	Country	numberOfConnections
1	"TXL"	"Berlin"	"DE"	0
2	"MEB"	"Melbourne"	"AU"	0
3	"ENF"	"Enontekio"	"FI"	0
4	"WMB"	"Warrnambool"	"AU"	0
5	"PTJ"	"Portland"	"AU"	0
6	"CMF"	"Chambéry/Aix-les-Bains"	"FR"	0
7				

Airports Serving the Most Countries

```
MATCH (a:Airport)-[:FLIES_TO]->(b:Airport)
WHERE a.country <> b.country
WITH a.iata AS AirportCode, a.city AS City, COUNT(DISTINCT b.country) AS NumberOfCountries
ORDER BY NumberOfCountries DESC
RETURN AirportCode, City, NumberOfCountries
LIMIT 10;
```

```
neo4j$ MATCH (a:Airport)-[:FLIES_TO]→(b:Airport) WHERE a.country <> b.country WITH a.iata AS Ai
```

	AirportCode	City	NumberOfCountries
1	"IST"	"Istanbul"	115
2	"CDG"	"Paris"	113
3	"FRA"	"Frankfurt"	102
4	"DXB"	"Dubai"	102
5	"AMS"	"Amsterdam"	93
6	"LHR"	"London"	87
7			

Continental Connectivity

```
MATCH (a1:Airport)-[:FLIES_TO]->(a2:Airport)
WHERE a1.continent <> a2.continent
WITH a1.continent AS DepartureContinent, a2.continent AS ArrivalContinent, COUNT(*) AS NumberOfFlights
RETURN DepartureContinent, ArrivalContinent, NumberOfFlights
ORDER BY NumberOfFlights DESC;
```

```
neo4j$ MATCH (a1:Airport)-[:FLIES_TO]→(a2:Airport) WHERE a1.continent <> a2.continent WITH a1.continent
```

	DepartureContinent	ArrivalContinent	NumberOfFlights
1	"EU"	"AS"	24374
2	"AS"	"EU"	21644
3	"AF"	"EU"	10906
4	"EU"	"AF"	10822
5	"NA"	"EU"	9450
6	"EU"	"NA"	9296
7			

Started streaming 30 records after 1 ms and completed after 825 ms.

```
neo4j$ MATCH (a1:Airport)-[f:FLIES_TO]→(a2:Airport) WHERE a1.continent =
```

	Continent	AverageDistance
1	"AS"	984.3546637743998
2	"NA"	873.4501471670357
3	"EU"	803.5621445415558
4	"AF"	740.9072524407239
5	"SA"	653.7147651006736
6	"OC"	624.3350462487141

Started streaming 6 records after 15 ms and completed after 1160 ms.

Average Flight Distance by Continent

```
MATCH (a1:Airport)-[f:FLIES_TO]->(a2:Airport)
WHERE a1.continent = a2.continent
WITH a1.continent AS Continent, AVG(f.distance) AS AverageDistance
RETURN Continent, AverageDistance
ORDER BY AverageDistance DESC;
```

Most Common Runway Lengths

```
MATCH (a:Airport)
WHERE a.longest IS NOT NULL
WITH a.longest AS
RunwayLength, COUNT(*) AS
Frequency
ORDER BY Frequency DESC
RETURN RunwayLength,
Frequency;
```

```
neo4j$ MATCH (a:Airport) WHERE a.longest IS NOT NULL WITH a.longest AS RunwayLength, COUNT
```

Table | Text | Code

	RunwayLength	Frequency
1	8202	404
2	9843	340
3	6562	248
4	8530	212
5	7874	208
6	6890	180
7		

Started streaming 1585 records after 9 ms and completed after 34 ms, displaying first 1000 rows.

Cities with Multiple Airports

```
MATCH (a:Airport)
WITH a.city AS City, COUNT(*) AS
NumberOfAirports
WHERE NumberOfAirports > 1
RETURN City, NumberOfAirports
ORDER BY NumberOfAirports DESC;
```

```
neo4j$ MATCH (a:Airport) WITH a.city AS City, COUNT(*) AS NumberOfAirports WHERE NumberOfAirports > 1 R
```

Table | Text | Code

	City	NumberOfAirports
1	"London"	24
2	"San Jose"	16
3	"Melbourne"	16
4	"Santa Rosa"	16
5	"Sydney"	12
6	"Beijing"	12
7		

Started streaming 3359 records after 10 ms and completed after 69 ms, displaying first 1000 rows.

Longest Domestic Routes

```
MATCH (a1:Airport)-[f:FLIES_TO]-
>(a2:Airport)
WHERE a1.country = a2.country
AND a1 <> a2
RETURN a1.iata AS
DepartureIATA, a2.iata AS
ArrivalIATA, f.distance AS Distance
ORDER BY f.distance DESC
LIMIT 10;
```

```
o4j$ MATCH (a1:Airport)-[f:FLIES_TO]→(a2:Airport) WHERE a1.country = a2.country AND a1 <> a2 RETURN a1
```

	DepartureIATA	ArrivalIATA	Distance
1	"HNL"	"BOS"	5083
2	"HNL"	"BOS"	5083
3	"BOS"	"HNL"	5083
4	"BOS"	"HNL"	5083
5	"BOS"	"HNL"	5083
6	"HNL"	"BOS"	5083
7			

tarted streaming 10 records after 14 ms and completed after 1010 ms.

- Challenges and Solutions: Learning and debugging Cypher queries was a hurdle that we overcame through practice and consultation of Neo4j's extensive documentation. While the computational limits of personal laptops hindered our ability to display the entire network, we mitigated this by focusing on subsets of the data for analysis and visualization.

Data Visualization and Analysis

- Visualization Techniques: For visualization, we utilized Neo4j's graph visualization tools, adjusting settings for clarity, such as node captions and color coding. We tailored the graph's appearance to enhance readability and interpretability.
- Insights from the Data: Key insights include the identification of major international hubs, the discovery of the longest and shortest routes, and the recognition of potential new route opportunities.

Discussion

- Interpretation of Results: The results highlight the global nature of the aviation network and the critical role of certain airports as international hubs.
- Limitations and Future Work: The primary limitation encountered was the inability of the Neo4j interface to display more than 300 nodes at a time without performance degradation. This limitation, along with the hardware constraints of our personal computers, prevented us from visualizing the entire dataset simultaneously. For future iterations of the project, it would be beneficial to use more powerful computational resources or cloud services to handle larger datasets without performance issues.

Conclusion

Our exploration of the airport network dataset using Neo4j graph database technology provided valuable insights into global connectivity and airport activity. Despite the challenges encountered, including computational limitations and the learning curve associated with Cypher, we successfully visualized and analyzed the dataset to reveal patterns and potential opportunities in global air traffic networks.

References

- Neo4j Graph Examples: Airport Routes. Retrieved from https://github.com/neo4j-graph-examples/airport-routes/tree/main