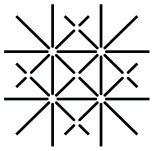


Universität Basel

Exercise Sheet 2

Distributed Information Systems (Spring Semester 2024)

Anastasiya Merkushova

Raphael Waltenpül

July 4, 2024

Exercise 2.1: Import the Data (10 points)

a) Done!

b) From Ex 2.2 and Ex 2.1.a we know that we need the following information:

2.2.a.1: *F:Flights (*year*)*

2.2.a.2: *F:Flights (*year*)* $-(F.tailNum = P.tailnum) \rightarrow$ *Planes*

2.2.a.3, 2.2.d.1, 2.2.e: *Airport* $-(F.origin = A.iata) \rightarrow$ *F:Flights*

2.2.d.1, 2.2.e *Airport* $\leftarrow (F.dest = A.iata)-$ *F:Flights*

2.2.a.3: *C:Carrier* $\leftarrow (F.uniqueCarrier = C.code)-$ *F:Flights*

2.2.b: *F:Flights (*year*)* $-(F.uniqueCarrier = C.code) \rightarrow$ *C:Carrier*

2.2.b: *F:Flights (*year*)* $-(F.uniqueCarrier = C.code) \rightarrow$ *C:Carrier* $-(G.uniqueCarrier = C.code) \rightarrow$ *G:Fuel (Gallons, USD)*

2.2.c, 2.2.e: *F:Flights (*year*)* $-(F.tailNum = P.tailnum) \rightarrow$ *Plane (Engine Type)*

2.2.d.1, 2.2.d.3: *F:Flights* $-(F.cancellationCode = R.code) \rightarrow$ *R:Cancellation*

2.2.d.2: *Airport* $\leftarrow (F.dest = A.iata)-$ *F:Flights*$-(F.cancellationCode = R.code) \rightarrow$ *R:Cancellation*

2.2.d.3: *C:Carrier* $\leftarrow (F.uniqueCarrier = C.code)-$ *F:Flights* $-(F.cancellationCode = R.code) \rightarrow$ *R:Cancellation*

From that we came up with the schema shown in fig. 1.
Due to lack of performance on relation creation, we had to add unique constraints and indexes on the following labels:

Unique constraints:

- Carrier (carrier.coder)
- Plane (plane.tailnum)
- Cancellation (cancellation.code)
- Airport (airport.iata)

Indexes:

- Flight (uniqueCarrier);
- Flight (tailNum);

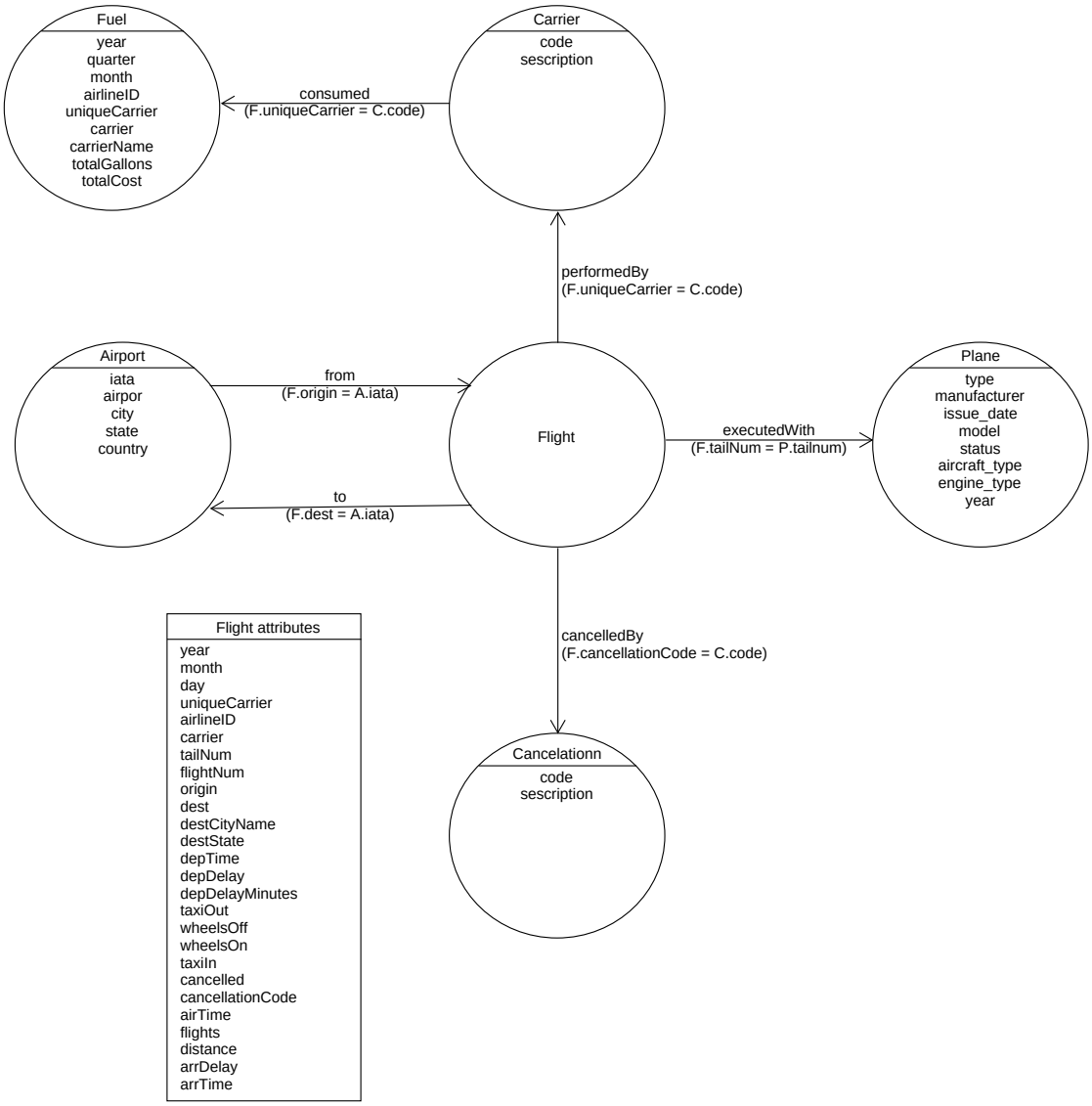


Figure 1: Schema

- Flight (dest);
- Flight (origin);
- Flight (cancellationCode);
- Airport (iata);
- Plane (tailNum);
- Carrier (code);
- Fuel (uniqueCarrier);

c) In the folder one can find the cypher solution `Schema.cyp` and a python script `srcCreateSchema.py` which automates the schema creation.

Exercise 2.2: Data Analysis (10 points)

a.1

```
MATCH (F:Flight)
RETURN labels(F), F.year, count(*)
```

Year	Count
2014	5819811
2015	5819079
2016	5617658

Table 1: Flights per Year

a.2

```
MATCH (F:Flight)-[r:executedWith]-(P:Plane)
RETURN F.year, COUNT(DISTINCT P.tailnum)
```

Year	Count
2014	4007
2015	4166
2016	4500

Table 2: Distinct planes per Year

a.3

```
MATCH (F:Flight)-[r:fromAirport]->(a:Airport)
WITH a, COUNT(*) AS numFlights
RETURN a.iata AS AirportIATA, numFlights
ORDER BY numFlights DESC
LIMIT 1
```

AirportIATA	numFlights
"ATL"	1,133,578

a.4

```
MATCH (f:Flight)
RETURN COUNT(DISTINCT f.uniqueCarrier) AS NumberOfCarriers
```

NumberOfCarriers
15

b

The three carriers with the highest fuel usage:

```
MATCH (c:Carrier)-[:consumed]->(f:Fuel)
WITH c, f.year AS year,
     SUM(toInteger(f.totalGallons)) AS totalFuelUsage,
     SUM(toInteger(f.totalCost)) AS totalCost
ORDER BY year, totalFuelUsage DESC, totalCost DESC
WITH year, COLLECT({CarrierCode: c.code,
totalFuelUsage: totalFuelUsage,
totalCost: totalCost}) AS yearStats
```

```
RETURN year, yearStats[0..3] AS topCarriers
```

Year	Carrier	Gallons (totalFuelUsage)	USD (totalCost)
2014	DL	3,265,645,505	$9,321,714,459
2014	UA	3,183,378,467	$9,042,121,024
2014	AA	2,473,463,889	$6,991,774,276
2015	DL	3,388,856,207	$7,845,050,611
2015	UA	3,215,520,613	$5,390,875,179
2015	AA	3,043,664,245	$4,955,099,986
2016	AA	3,595,720,988	$4,732,547,705
2016	DL	3,412,617,055	$5,614,257,197
2016	UA	3,261,229,135	$4,357,641,361

Table 3: Summary of top carriers' fuel usage and costs

The five carriers with the most flights per year:

```
MATCH (c:Carrier)<-[:performedBy]-(f:Flight)
WITH c, f.year AS year, COUNT(*) AS numFlights
ORDER BY year, numFlights DESC
WITH year,COLLECT({CarrierCode: c.code, numFlights:numFlights})
AS yearStats
RETURN year, yearStats[0..5] AS topCarriers
```

Year	**Carrier**	**Flights**
2014	WN	1,174,633
2014	DL	800,375
2014	EV	686,021
2014	OO	613,030
2014	AA	537,697
2015	WN	1,261,855
2015	DL	875,881
2015	AA	725,984
2015	OO	588,353
2015	EV	571,977
2016	WN	1,299,444
2016	DL	922,746
2016	AA	914,495
2016	OO	605,933
2016	UA	545,067

Table 4: Summary of top carriers' number of flights

c

```
MATCH (f:Flight)-[:executedWith]->(p:Plane)
WITH f.year AS year, p.engine_type AS engineType,
     COUNT(*) AS numFlights, SUM(f.distance) AS totalDistance
RETURN year, engineType, numFlights, totalDistance
ORDER BY year, engineType
```

Year	**Engine Type**	**Distance (totalDistance)**	**No. Flights (numFlights)**
2014	2 Cycle	130,406.0	54
2014	4 Cycle	2,511,117.0	2,695
2014	Electric	301,175.0	297
2014	None	847,564.0	1,238
2014	Reciprocating	51,253,852.0	51,557
2014	Turbo-Fan	3,694,982,772.0	4,590,522
2014	Turbo-Jet	217,394,906.0	257,545
2014	Turbo-Prop	18,845,412.0	91,287
2014	Turbo-Shaft	8,059,362.0	8,941
2015	2 Cycle	2,146,218.0	852
2015	4 Cycle	1,069,931.0	1,469
2015	Electric	358,557.0	292
2015	None	936,423.0	1,320
2015	Reciprocating	43,227,264.0	34,300
2015	Turbo-Fan	3,923,099,423.0	4,794,639
2015	Turbo-Jet	219,967,114.0	261,372
2015	Turbo-Prop	5,704,925.0	15,621
2015	Turbo-Shaft	4,963,219.0	3,655
2016	2 Cycle	2,693,964.0	1,484
2016	4 Cycle	3,766,086.0	5,112
2016	Electric	3,386,151.0	2,831
2016	None	1,180,903.0	1,384
2016	Reciprocating	80,773,015.0	68,356
2016	Turbo-Fan	3,983,590,912.0	4,834,532
2016	Turbo-Jet	230,873,526.0	265,864
2016	Turbo-Prop	14,531,116.0	13,091
2016	Turbo-Shaft	8,464,618.0	5,896

Table 5: Summary of flights by engine type

d.1

```
MATCH (f:Flight)-[:fromAirport]->(a:Airport)
WHERE f.taxiIn IS NOT NULL
WITH a, AVG(f.taxiIn) AS avgTaxiIn
RETURN a.iata AS AirportIATA, a.airport AS AirportName,
avgTaxiIn AS TaxiInAverage
ORDER BY avgTaxiIn DESC LIMIT 1
```

AirportIATA	AirportName	TaxiInAverage
"ART"	"Watertown Intl"	24.68

```
MATCH (f:Flight)-[:fromAirport]->(a:Airport)
WHERE f.taxiOut IS NOT NULL
WITH a, AVG(f.taxiOut) AS avgTaxiOut
RETURN a.iata AS AirportIATA, a.airport AS AirportName,
avgTaxiOut AS TaxiOutAverage
ORDER BY avgTaxiOut DESC LIMIT 1
```

AirportIATA	AirportName	TaxiOutAverage
"EFD"	"Ellington"	48.0

d.2

```
WITH f.year AS Year, dest.city AS City,
COUNT(*) AS CancellationCount,
f.cancellationCode AS CancellationCode
ORDER BY Year, CancellationCount DESC

// Look up the cancellation code descriptions
MATCH (c: Cancellation {code: CancellationCode})
WITH Year, City, CancellationCount, c.description AS MainReason
ORDER BY Year, CancellationCount DESC

// Collect top cancellation reasons for each year
WITH Year, COLLECT({City: City, CancellationCount:
CancellationCount, MainReason: MainReason}) AS topCancellations
RETURN Year, topCancellations[0..1] AS topCancellation
```

Year	CancellationCount	MainReason	City
2014	7592	Weather	Chicago
2015	6334	Weather	Chicago
2016	3601	Weather	Chicago

d.3

```
MATCH (f:Flight)-[:performedBy]->(c:Carrier)
WHERE f.cancelled = '1.00' AND f.cancellationCode = 'A'
WITH c.code AS CarrierCode, COUNT(*) AS CancellationCount
ORDER BY CancellationCount ASC
WITH COLLECT({CarrierCode: CarrierCode,
CancellationCount: CancellationCount})[0]
AS LeastReliableCarrier
RETURN LeastReliableCarrier.CarrierCode
AS LeastReliableCarrier,
LeastReliableCarrier.CancellationCount
AS CancellationCountLeast
```

LeastReliableCarrier	CancellationCountLeast
"WN"	19748

Table 6: Least Reliable Carrier

```
MATCH (f:Flight)-[:performedBy]->(c:Carrier)
WHERE f.cancelled = '1.00' AND f.cancellationCode = 'A'
WITH c.code AS CarrierCode, COUNT(*) AS CancellationCount
ORDER BY CancellationCount DESC
WITH COLLECT({CarrierCode: CarrierCode,
CancellationCount: CancellationCount})[0] AS ReliableCarrier
RETURN ReliableCarrier.CarrierCode AS ReliableCarrier,
ReliableCarrier.CancellationCount AS CancellationCountLeast
```

ReliableCarrier	CancellationCountLeast
"FL"	305

Table 7: Least Reliable Carrier

d.4

We assumed that we don't need the look up table for the monthes.

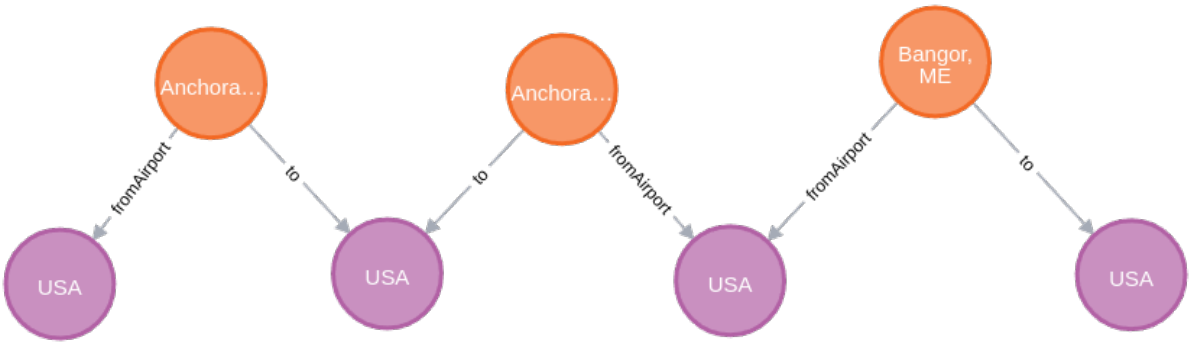
```
MATCH (f:Flight)
WHERE f.cancelled = '1.00' AND f.cancellationCode = 'B'
WITH f.month AS Month, COUNT(*) AS CancellationCount
ORDER BY CancellationCount DESC
RETURN Month, CancellationCount
LIMIT 3
```

Month	CancellationCount
2	35990
1	35182
3	15137

Table 8: Months with Most Flights Canceled due to Weather Conditions

e

MATCH path = shortestPath((start:Airport iata: 'ADK')<-[:fromAirport|to*]->(end:Airport iata: 'BGR')) RETURN path



We used UI of neo4j browser to answer these questions, because it wasn't specified to use queries here.

- 1. 3 different flights with ids: 16686692, 16360146, 2428609, from ADK to ANC, from ANC to EWR, from EWR to BGR.
- 2. AS (tailNum:N768AS), UA (tailNum: N17139), EV (tailNum: N11187)
- 3. 393.0 + 3370.0 + 1192.0 = 4955km

– 4. 58 + 429 + 150 = 637 min

– 5. 10.25

```
// Calculate average delay from ADK to ANC
MATCH (start1:Airport {iata: 'ADK'})<-[:fromAirport]
-(flight1:Flight)-[:to]->(end1:Airport {iata: 'ANC'})
WHERE flight1.cancelled = '0.00'
AND
flight1.depDelay IS NOT NULL
WITH AVG(flight1.depDelay) AS avg_delay_ADK_TO_ANC

// Calculate average delay from ANC to EWR
MATCH (start2:Airport {iata: 'ANC'})<-[:fromAirport]
-(flight2:Flight)-[:to]->(end2:Airport {iata: 'EWR'})
WHERE flight2.cancelled = '0.00'
AND
flight2.depDelay IS NOT NULL
WITH avg_delay_ADK_TO_ANC,
AVG(flight2.depDelay) AS avg_delay_ANC_TO_EWR

// Calculate average delay from EWR to BGR
MATCH (start3:Airport {iata: 'EWR'})<-[:fromAirport]
-(flight3:Flight)-[:to]->(end3:Airport {iata: 'BGR'})
WHERE flight3.cancelled = '0.00'
AND
flight3.depDelay IS NOT NULL
WITH avg_delay_ADK_TO_ANC, avg_delay_ANC_TO_EWR,
AVG(flight3.depDelay) AS avg_delay_EWR_TO_BGR

// Calculate overall average delay
WITH avg_delay_ADK_TO_ANC,
avg_delay_ANC_TO_EWR,
avg_delay_EWR_TO_BGR
RETURN (avg_delay_ADK_TO_ANC +
avg_delay_ANC_TO_EWR +
avg_delay_EWR_TO_BGR) / 3 AS overall_average_delay
```

Exercise 2.Bonus (10 points)