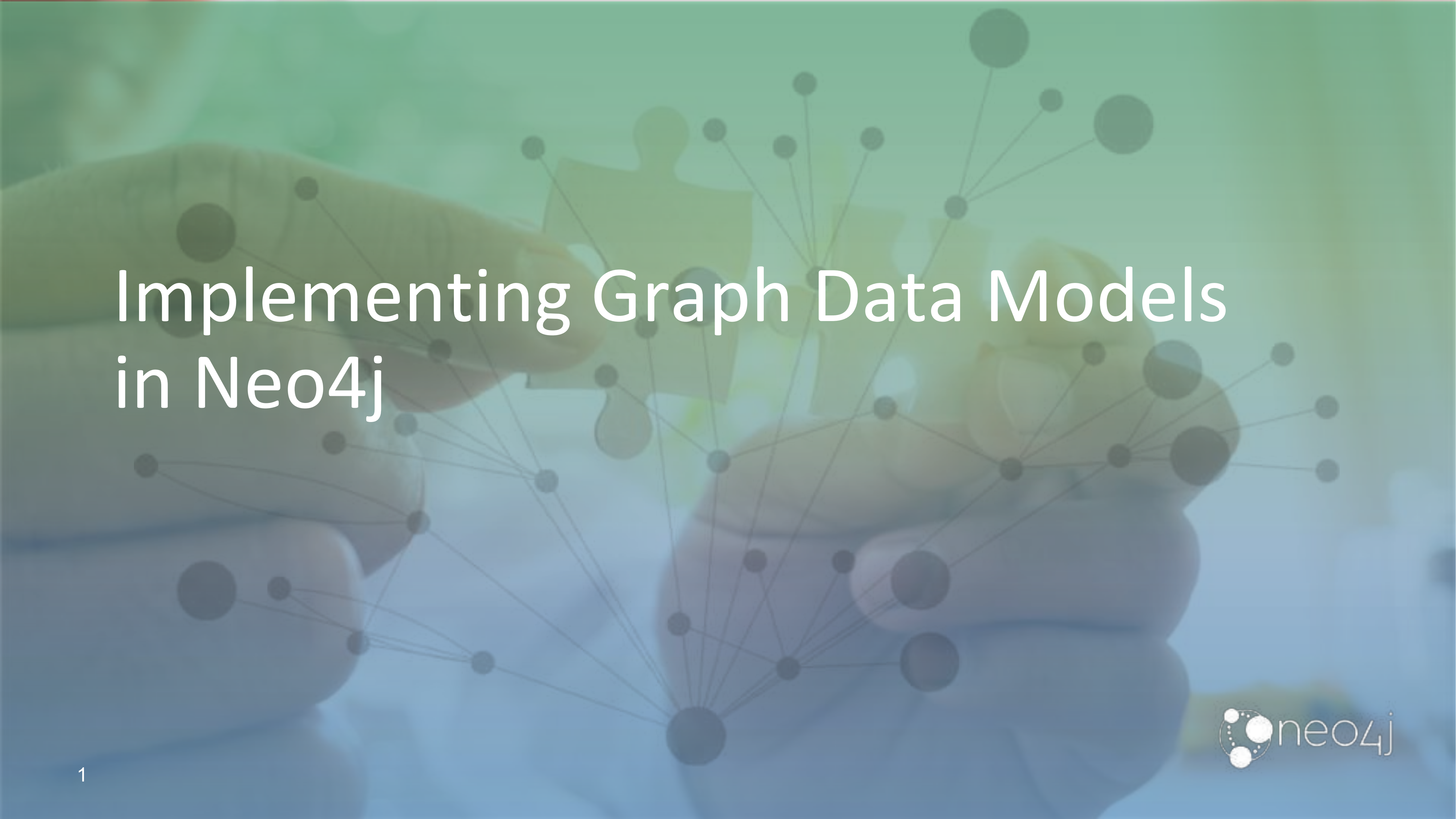
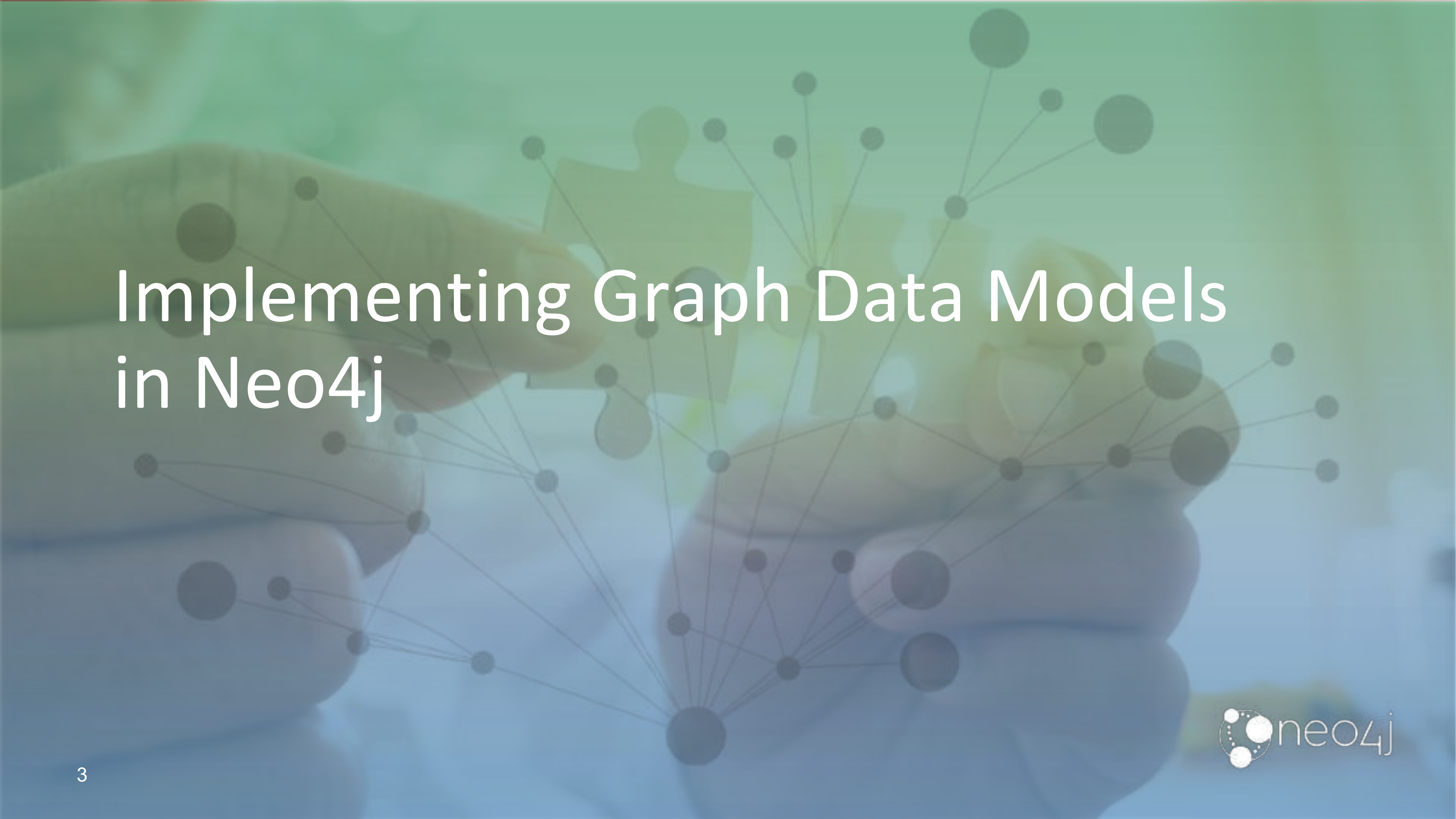


Implementing Graph Data Models in Neo4j

Lesson Overview

Modules in this course:

1. Implementing Your First Model
2. Importing Data
3. Profiling Queries
4. Refactoring Graphs



Implementing Graph Data Models in Neo4j

In This Module You’ll Learn ...

At the end of this module, you should be able to:

- Write Cypher code to implement a simple graph data model
- Confirm that the starter data is in the graph

Model Domain

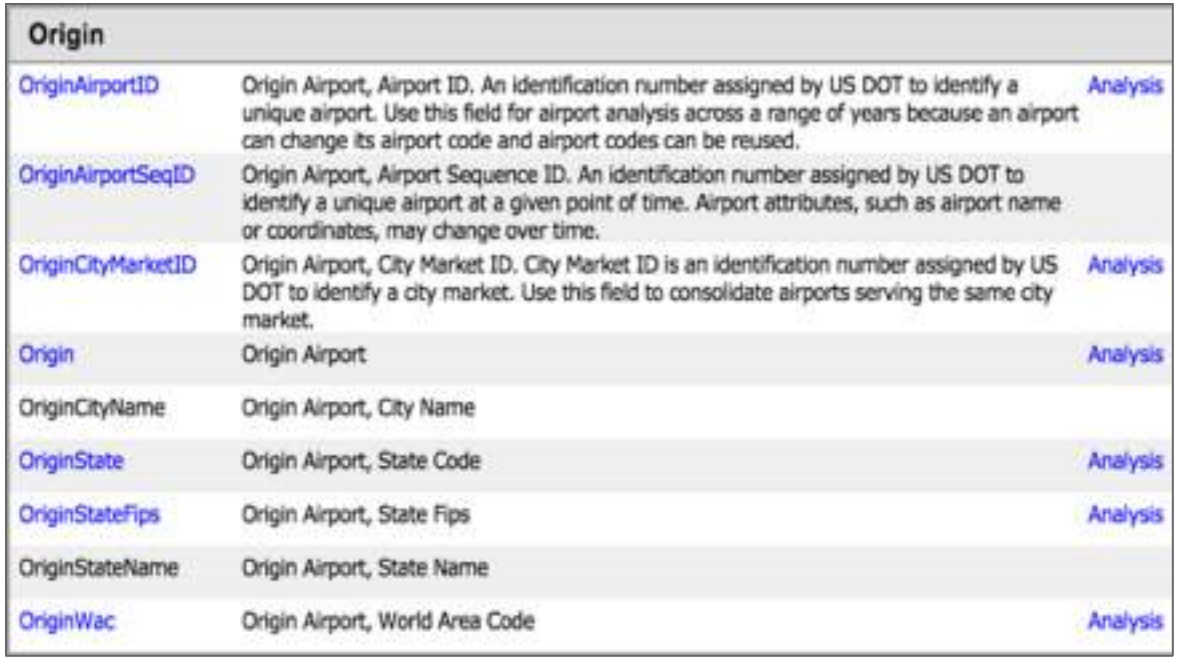
https://www.bts.gov/browse-statistical-products-and-data

United States Department of Transportation
Ask-A-Librarian
A-Z Index
Bureau of Transportation Statistics
Search BTS site
Topics and Geography
Statistical Products and Data
National Transportation Library
Newsroom
About BTS
Home

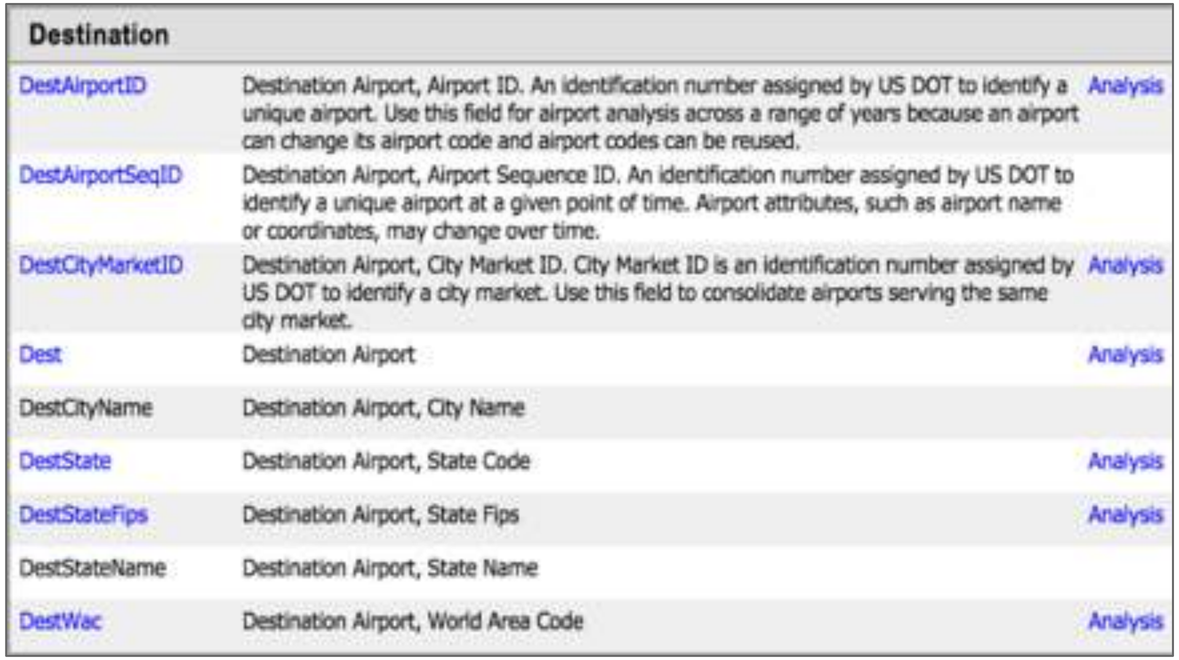
Browse Statistical Products and Data

Browse databases, surveys, reports and other statistical products created by BTS. Use BTS tools and apps to build your own statistical tables, graphics and maps.

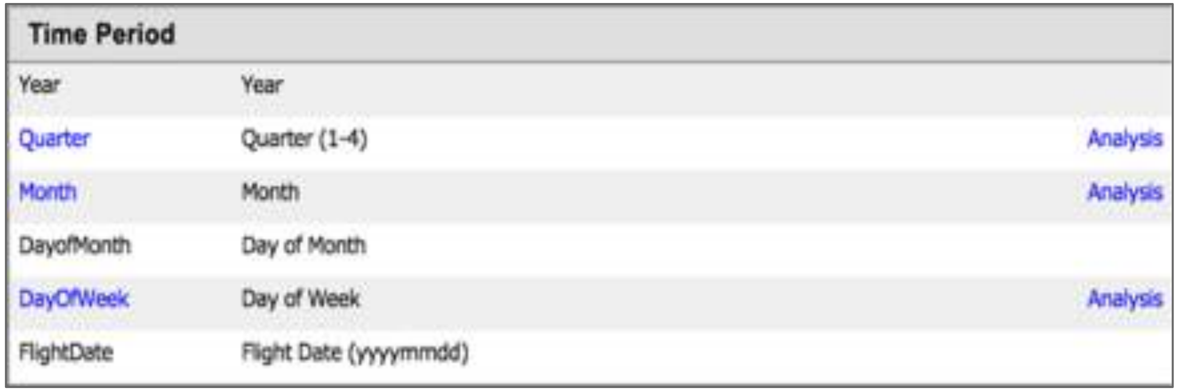
Model Data



Origin		
OriginAirportID	Origin Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.	Analysis
OriginAirportSeqID	Origin Airport, Airport Sequence ID. An identification number assigned by US DOT to identify a unique airport at a given point of time. Airport attributes, such as airport name or coordinates, may change over time.	
OriginCityMarketID	Origin Airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market.	Analysis
Origin	Origin Airport	Analysis
OriginCityName	Origin Airport, City Name	
OriginState	Origin Airport, State Code	Analysis
OriginStateFips	Origin Airport, State Fips	Analysis
OriginStateName	Origin Airport, State Name	
OriginWac	Origin Airport, World Area Code	Analysis



Destination		
DestAirportID	Destination Airport, Airport ID. An identification number assigned by US DOT to identify a unique airport. Use this field for airport analysis across a range of years because an airport can change its airport code and airport codes can be reused.	Analysis
DestAirportSeqID	Destination Airport, Airport Sequence ID. An identification number assigned by US DOT to identify a unique airport at a given point of time. Airport attributes, such as airport name or coordinates, may change over time.	
DestCityMarketID	Destination Airport, City Market ID. City Market ID is an identification number assigned by US DOT to identify a city market. Use this field to consolidate airports serving the same city market.	Analysis
Dest	Destination Airport	Analysis
DestCityName	Destination Airport, City Name	
DestState	Destination Airport, State Code	Analysis
DestStateFips	Destination Airport, State Fips	Analysis
DestStateName	Destination Airport, State Name	
DestWac	Destination Airport, World Area Code	Analysis

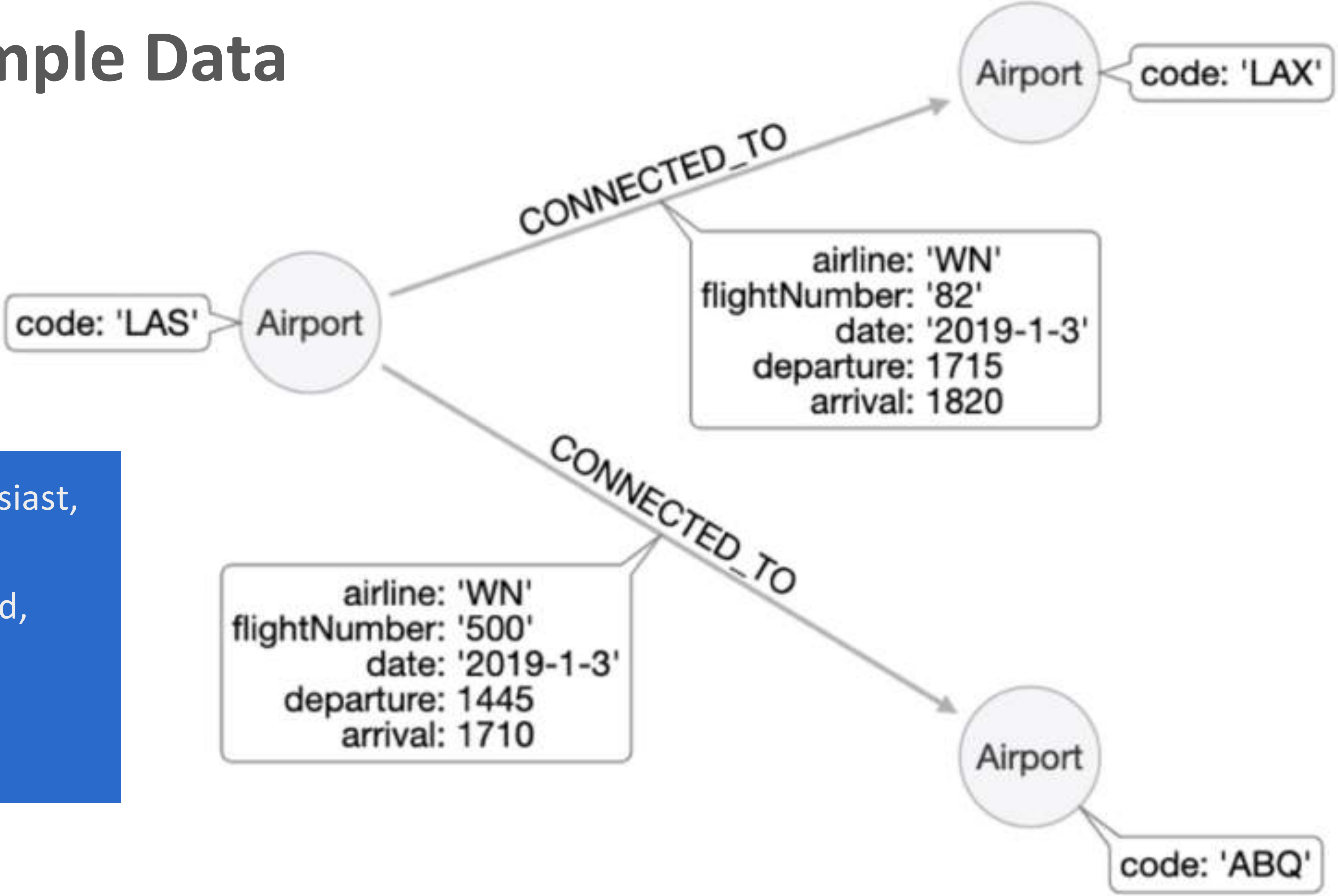


Time Period		
Year	Year	
Quarter	Quarter (1-4)	Analysis
Month	Month	Analysis
DayofMonth	Day of Month	
DayOfWeek	Day of Week	Analysis
FlightDate	Flight Date (yyyymmdd)	

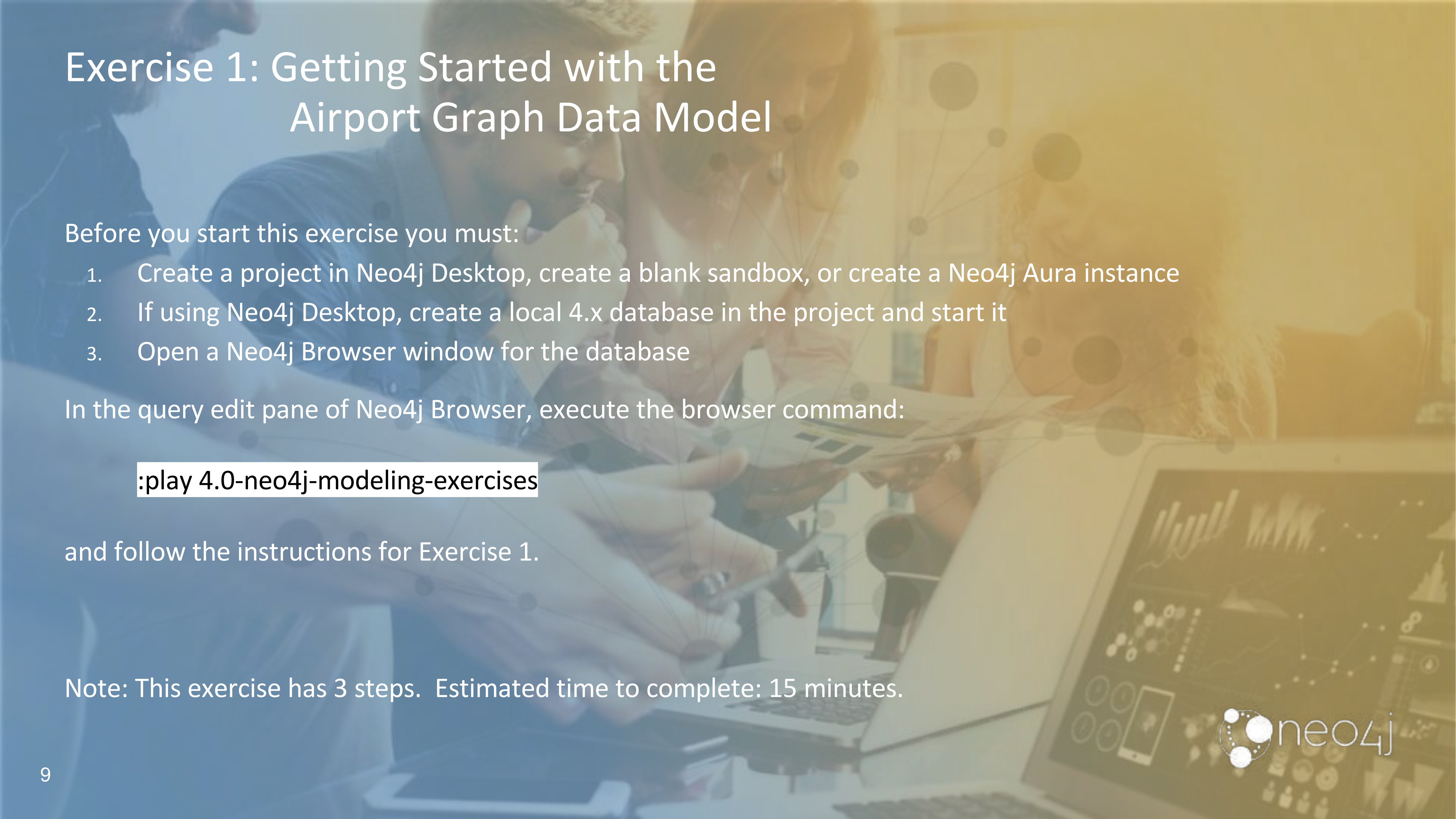
Model Application Question

As an air travel enthusiast,
I want to know how airports are connected,
so I find the busiest airports.

Model Sample Data



As an air travel enthusiast, I want to know how airports are connected, so I find the busiest airports.



Exercise 1: Getting Started with the Airport Graph Data Model

Before you start this exercise you must:

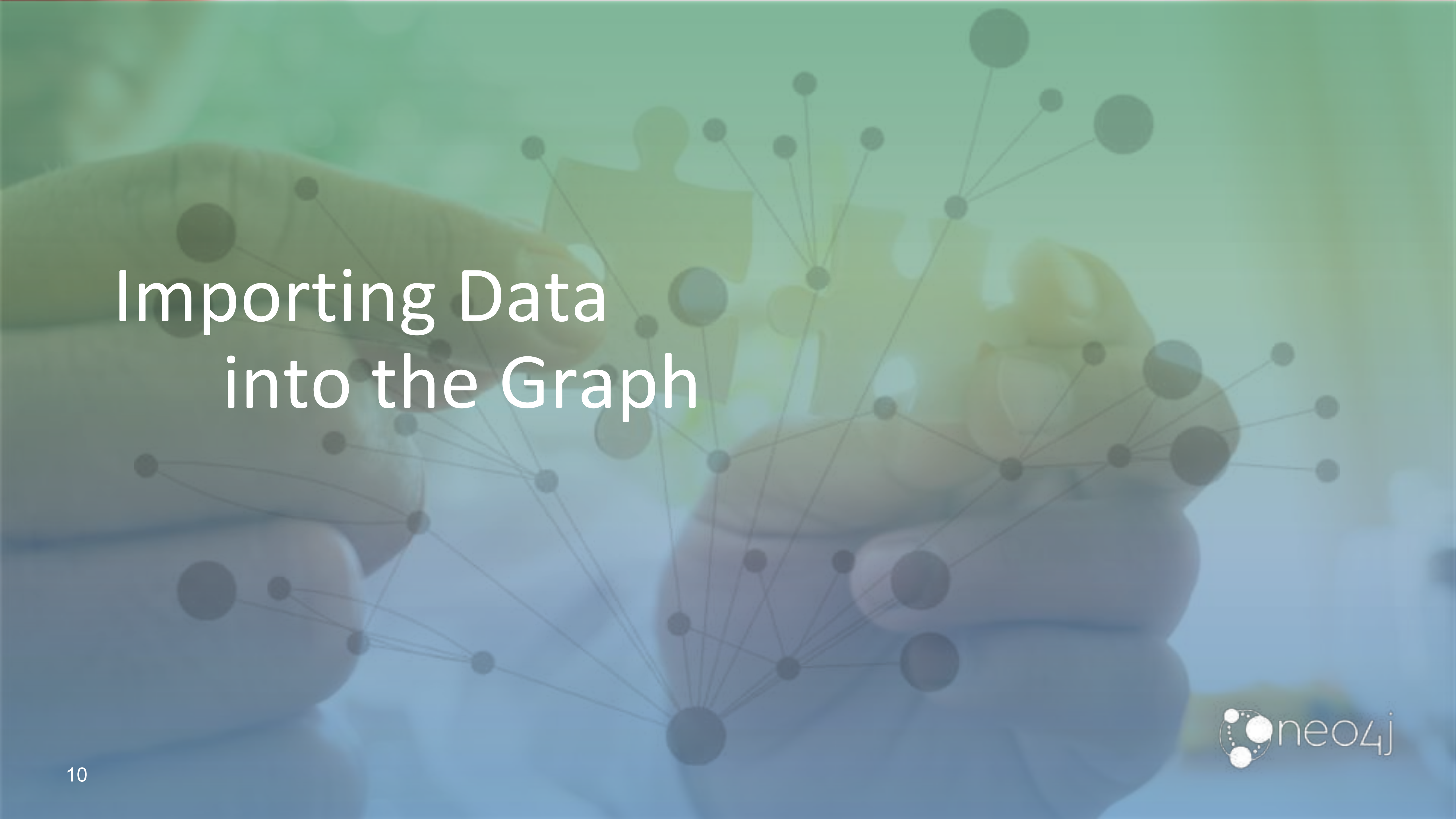
1. Create a project in Neo4j Desktop, create a blank sandbox, or create a Neo4j Aura instance
2. If using Neo4j Desktop, create a local 4.x database in the project and start it
3. Open a Neo4j Browser window for the database

In the query edit pane of Neo4j Browser, execute the browser command:

```
:play 4.0-neo4j-modeling-exercises
```

and follow the instructions for Exercise 1.

Note: This exercise has 3 steps. Estimated time to complete: 15 minutes.



Importing Data into the Graph

In This Module You’ll Learn ...

At the end of this module, you should be able to:

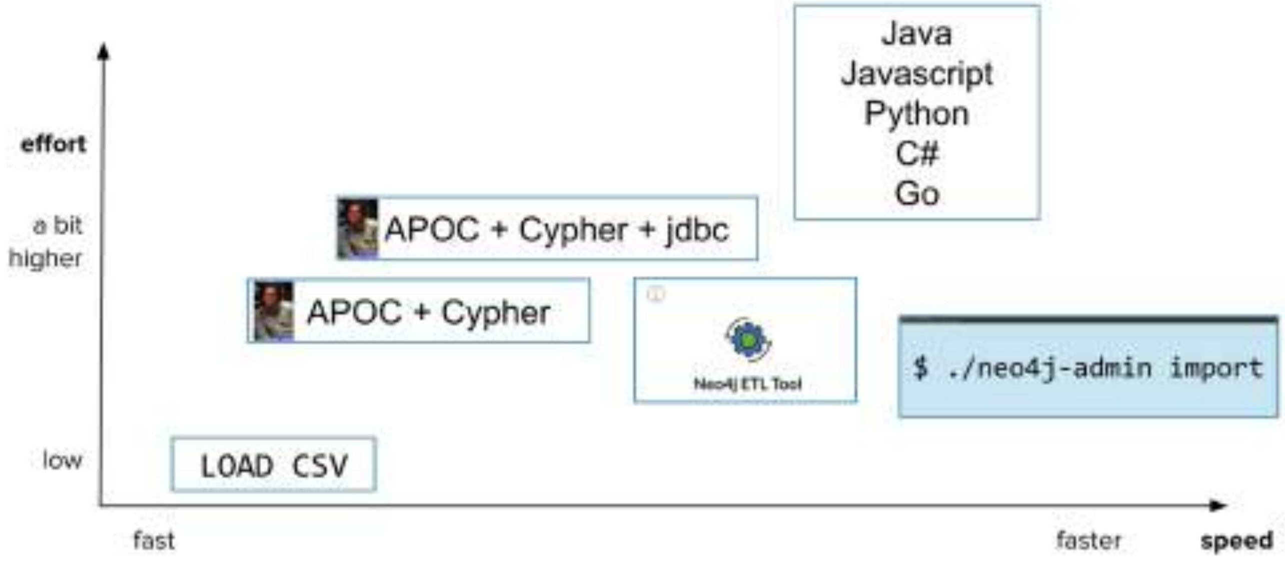
- Write Cypher code to import CSV data into a graph
- Confirm that the data has been loaded

Options for Importing Data into a Graph

There are many options for importing data into Neo4j

The option selected for importing data depends on ...

- The amount of data to be imported
- The available tools and knowledge of those tools
- The amount of time allocated to perform the import



Java
Javascript
Python
C#
Go

APOC + Cypher + jdbc

APOC + Cypher

Neo4j ETL Tool

```
$ ./neo4j-admin import
```

LOAD CSV

Import Preparation

- Names of entities (node labels)
- Names of relationships
- Names of properties for nodes and relationships
- Constraints to be defined
- Indexes required
- The most important queries

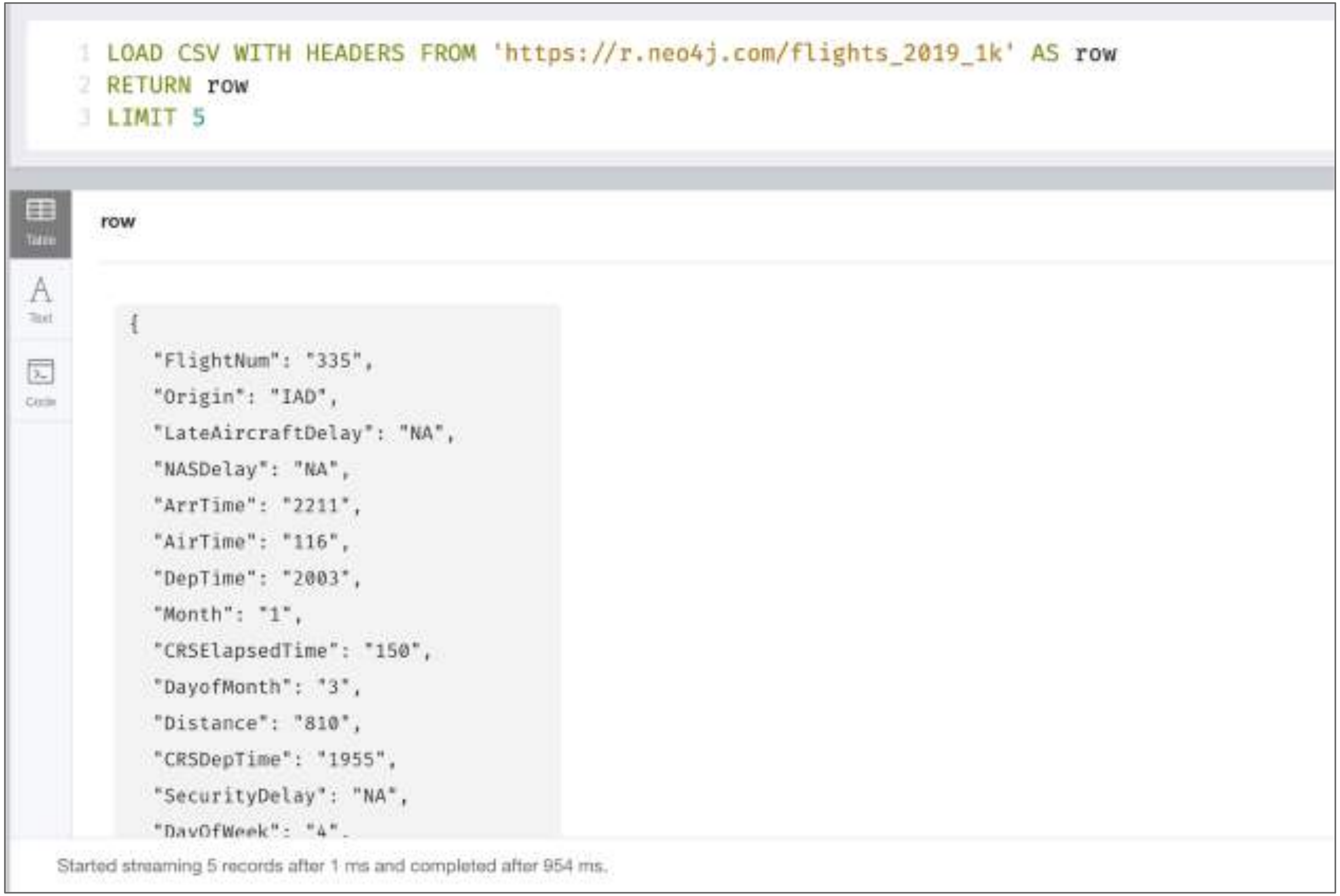
LOAD CSV Syntax

Simplified syntax for using LOAD CSV:

```
LOAD CSV     // load csv data
WITH HEADERS // optionally use first header row as keys in "row" map
FROM "url"   // file:/// file relative to $NEO4J_HOME/import or http://
AS row       // return each row of the CSV as list of strings or map
// ... rest of the Cypher statement ...
```

Use **`LOAD CSV`** for CSV files that contain fewer than 100k lines

Inspecting CSV File Data on a Network



```
LOAD CSV WITH HEADERS FROM 'https://r.neo4j.com/flights_2019_1k' AS row
RETURN row
LIMIT 5
```

row

```
{
  "FlightNum": "335",
  "Origin": "IAD",
  "LateAircraftDelay": "NA",
  "NASDelay": "NA",
  "ArrTime": "2211",
  "AirTime": "116",
  "DepTime": "2003",
  "Month": "1",
  "CRSElapsedTime": "150",
  "DayofMonth": "3",
  "Distance": "810",
  "CRSDepTime": "1955",
  "SecurityDelay": "NA",
  "DayOfWeek": "4",
```

Started streaming 5 records after 1 ms and completed after 954 ms.

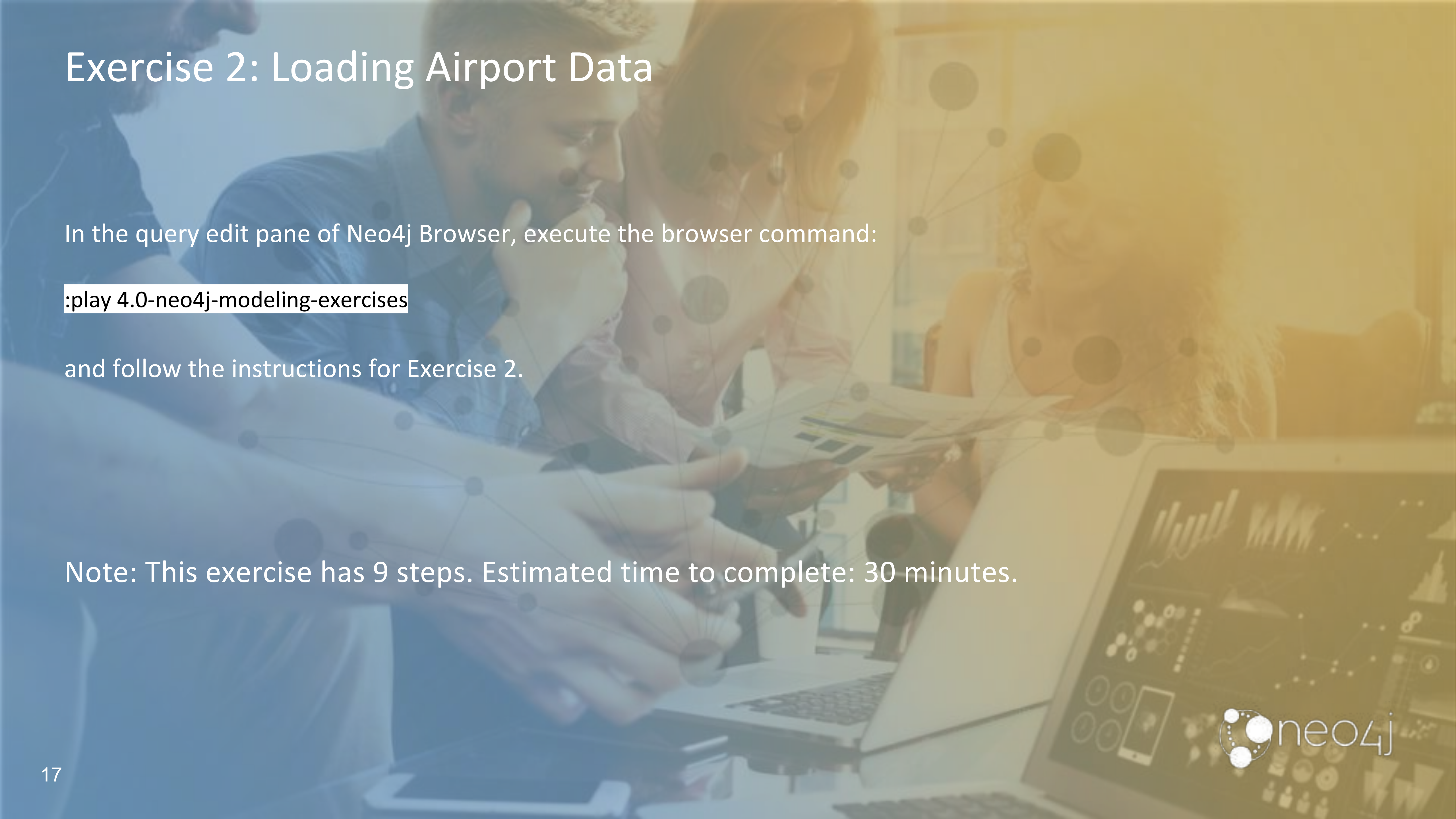
Creating Nodes and Relationships

LOAD CSV command reads rows of data from a CSV file

- It then creates nodes and relationships in the graph

```
LOAD CSV WITH HEADERS FROM 'https://r.neo4j.com/flights_2019_1k' AS row
MERGE (origin:Airport {code: row.Origin})
MERGE (destination:Airport {code: row.Dest})
MERGE (origin)-[connection:CONNECTED_TO {
  airline: row.UniqueCarrier,
  flightNumber: row.FlightNum,
  date: toInteger(row.Year) + '-' + toInteger(row.Month) + '-' +
        toInteger(row.DayofMonth)}]->(destination)
ON CREATE SET connection.departure = toInteger(row.CRSDepTime),
              connection.arrival = toInteger(row.CRSArrTime)
```

Exercise 2: Loading Airport Data



In the query edit pane of Neo4j Browser, execute the browser command:

```
:play 4.0-neo4j-modeling-exercises
```

and follow the instructions for Exercise 2.

Note: This exercise has 9 steps. Estimated time to complete: 30 minutes.

Summary

You should now be able to:

- Write Cypher code to import CSV data with Cypher
- Confirm that the data has been loaded

Profiling Queries

In This Module You’ll Learn ...

At the end of this module, you should be able to:

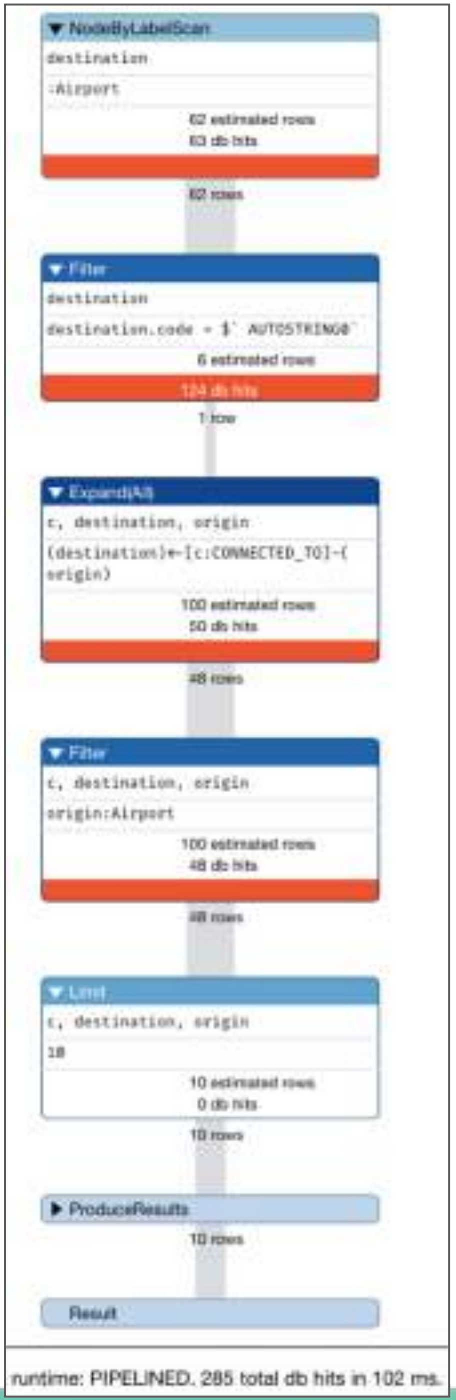
- Profile queries against the graph
- Determine if a query can be improved

Profiling a Query

```
PROFILE
MATCH (origin:Airport)-
   [c:CONNECTED_TO]->(destination:Airport)
WHERE destination.code = 'LAS'
RETURN origin, destination, c LIMIT 10
```

Code to profile a query

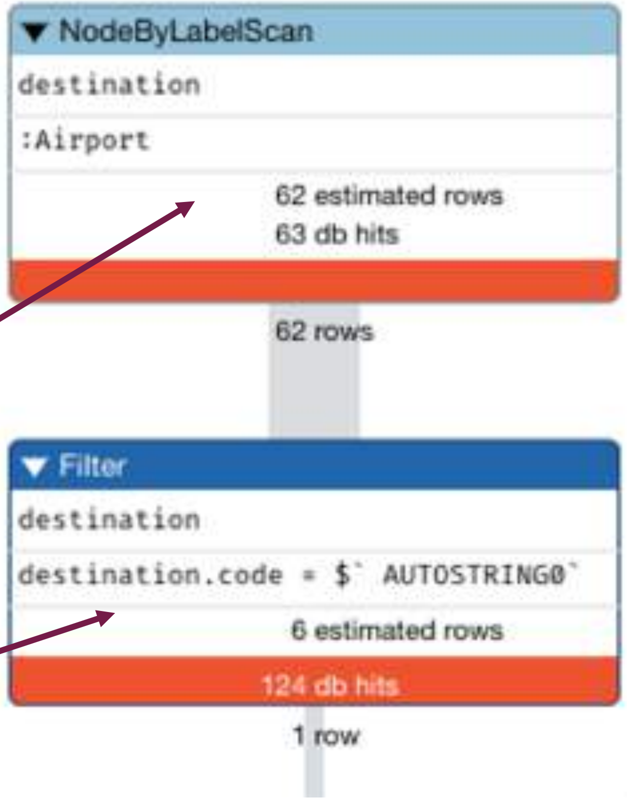
- Retrieve all connections that have a destination of *LAS*



```
▼ NodeByLabelScan
destination
:Airport
62 estimated rows
63 db hits
62 rows

▼ Filter
destination
destination.code = $` AUTOSTRING0`
6 estimated rows
124 db hits
1 row

▼ Expand(All)
c, destination, origin
(destination)<-[c:CONNECTED_TO]-(
origin)
100 estimated rows
50 db hits
48 rows

▼ Filter
c, destination, origin
origin:Airport
100 estimated rows
48 db hits
48 rows

▼ Limit
c, destination, origin
10
10 estimated rows
0 db hits
10 rows

▶ ProduceResults
10 rows

Result

runtime: PIPELINED. 285 total db hits in 102 ms.
```

Analyzing the Query Profile (1 of 3)

```
PROFILE
MATCH (origin:Airport)-
   [c:CONNECTED_TO]->(destination:Airport)
WHERE destination.code = 'LAS'
RETURN origin, destination, c LIMIT 10
```



▼ NodeByLabelScan

destination

:Airport

62 estimated rows
63 db hits

62 rows

▼ Filter

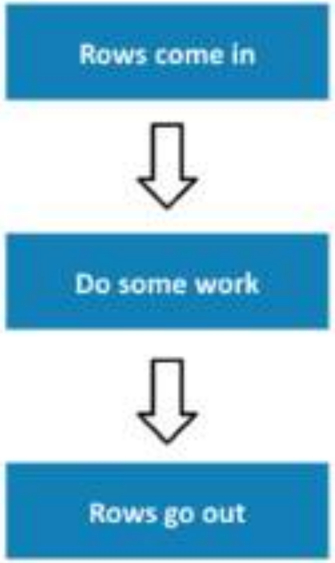
destination

destination.code = $` AUTOSTRING0`

6 estimated rows

124 db hits

1 row



Rows come in

Do some work

Rows go out

Code analysis

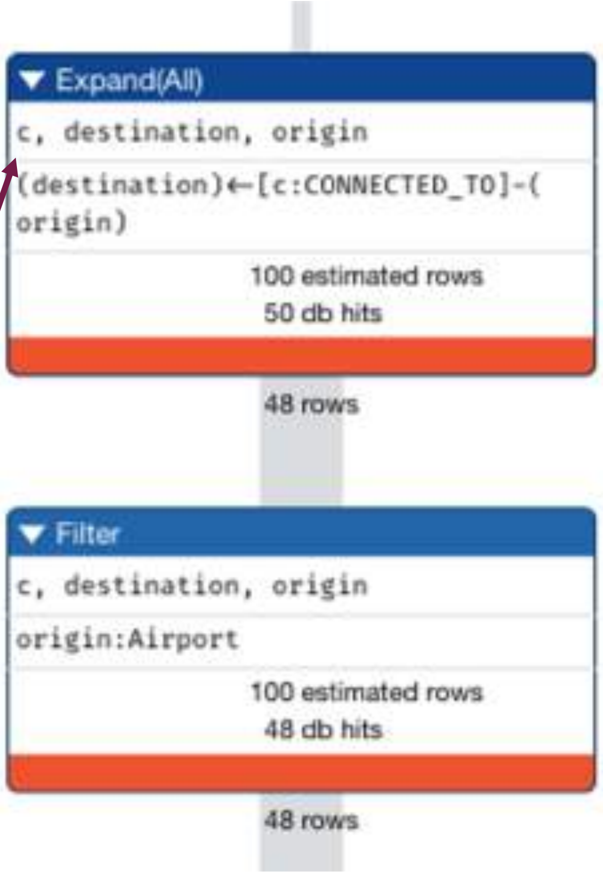
- `destination:Airport`
  - 62 total airports
- `Destination.code = ‘LAS’`
  - One airport with code LAS

Analyzing the Query Profile (2 of 3)

```
PROFILE
MATCH (origin:Airport)-
   [c:CONNECTED_TO]->(destination:Airport)
WHERE destination.code = 'LAS'
RETURN origin, destination, c LIMIT 10
```

Code analysis

- Pattern `origin-[c]->(destination)`
  - 48 nodes
- `No filter for pattern`
  - All 48 nodes pass through



c, destination, origin

(destination)←[c:CONNECTED_TO]-(origin)

100 estimated rows
50 db hits

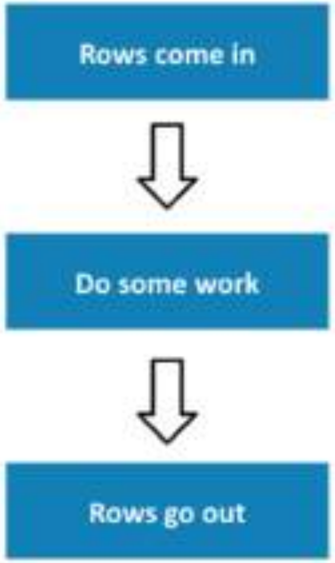
48 rows

c, destination, origin

origin:Airport

100 estimated rows
48 db hits

48 rows



Analyzing the Query Profile (3 of 3)

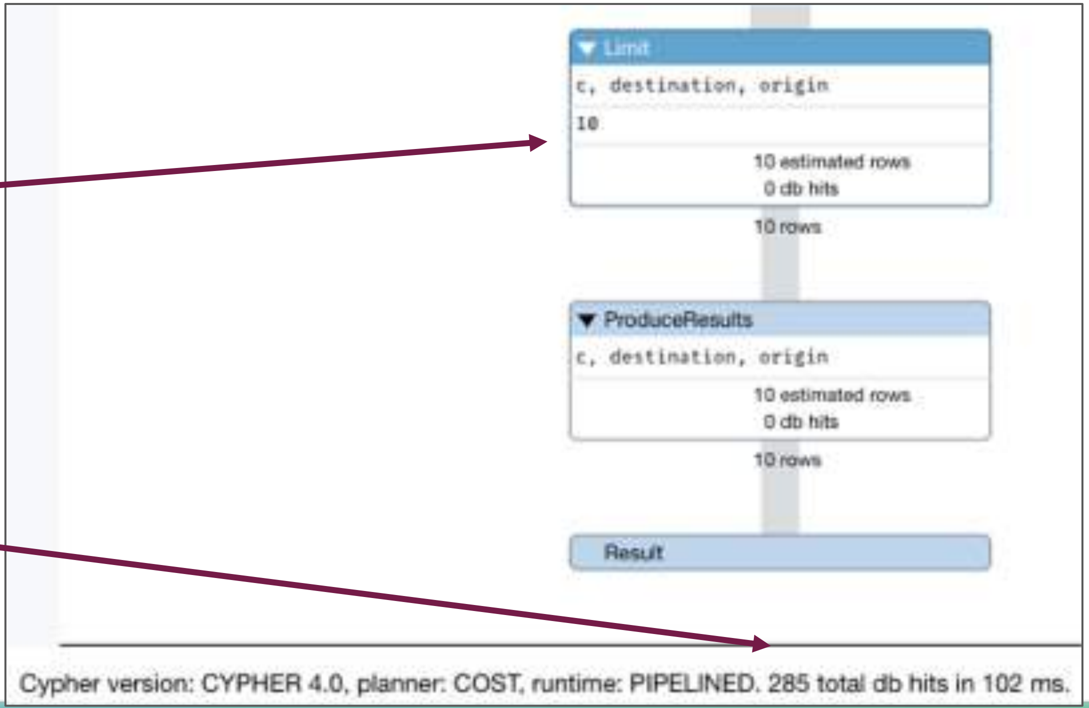
```
PROFILE
MATCH (origin:Airport)-
   [c:CONNECTED_TO]->(destination:Airport)
WHERE destination.code = 'LAS'
RETURN origin, destination, c LIMIT 10
```

Code analysis

- Product results
  - 48 nodes
- Total cost
  - 285 db hits

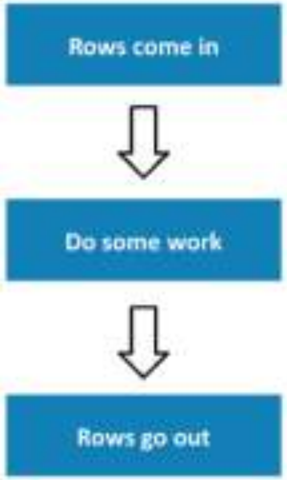
Code analysis

- LIMIT 10
- Product results
  - 48 nodes
- Total cost
  - 285 db hits



```
▼ Limit
c, destination, origin
10
                10 estimated rows
                0 db hits
                10 rows

▼ ProduceResults
c, destination, origin
                10 estimated rows
                0 db hits
                10 rows

Result
```

Cypher version: CYPHER 4.0, planner: COST, runtime: PIPELINED. 285 total db hits in 102 ms.



Rows come in

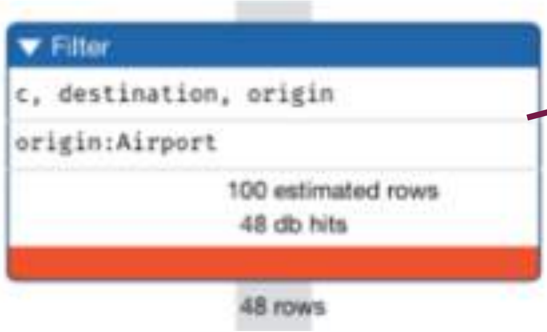
Do some work

Rows go out

Analyzing Without :Airport Filter

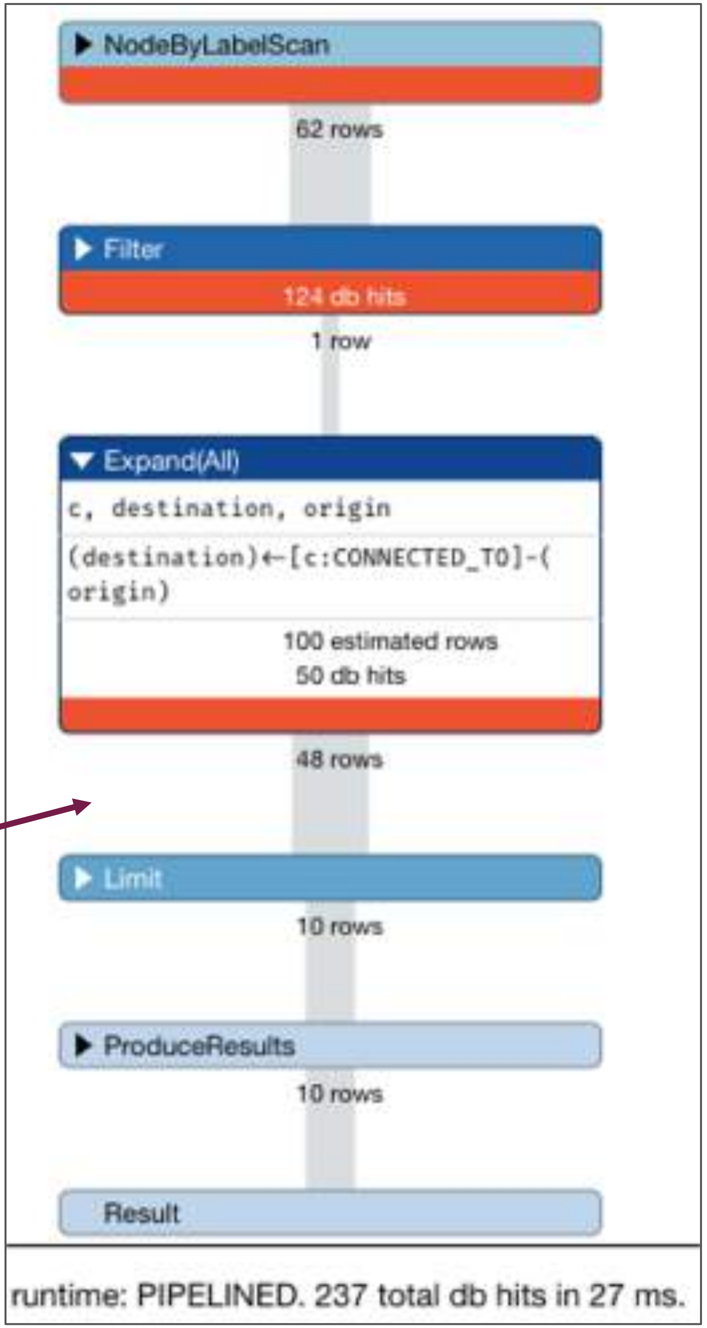
```
PROFILE
MATCH (origin)-
   [c:CONNECTED_TO]->(destination:Airport)
WHERE destination.code = 'LAS'
RETURN origin, destination, c LIMIT 10
```

:Airport label not specified

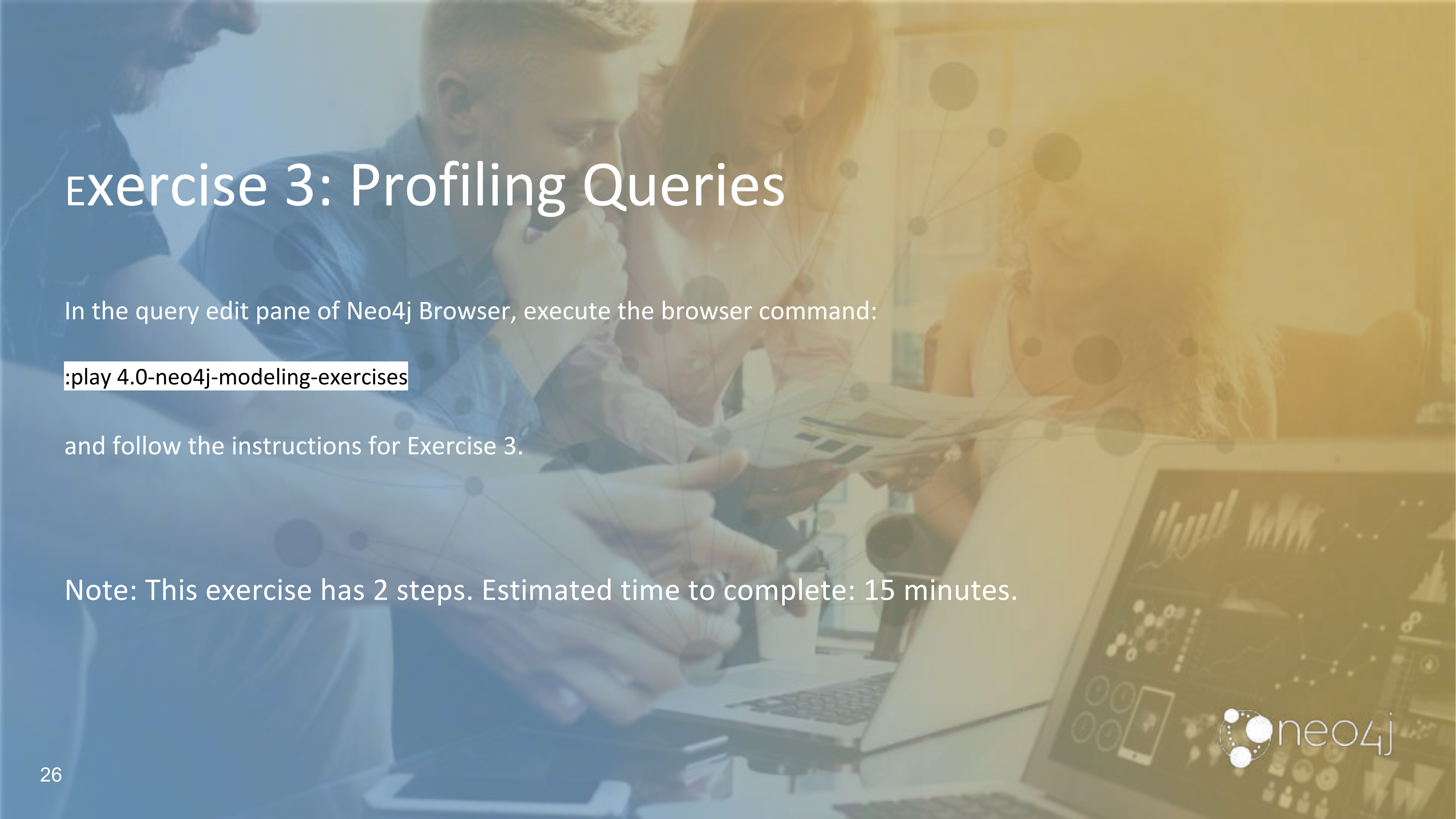


```
Filter
c, destination, origin
origin:Airport
100 estimated rows
48 db hits
48 rows
```

Without this Filter step there are 48 less db hits



```
NodeByLabelScan
62 rows
Filter
124 db hits
1 row
Expand(All)
c, destination, origin
(destination)←[c:CONNECTED_TO]-(
origin)
100 estimated rows
50 db hits
48 rows
Limit
10 rows
ProduceResults
10 rows
Result
runtime: PIPELINED. 237 total db hits in 27 ms.
```



Exercise 3: Profiling Queries

In the query edit pane of Neo4j Browser, execute the browser command:

```
:play 4.0-neo4j-modeling-exercises
```

and follow the instructions for Exercise 3.

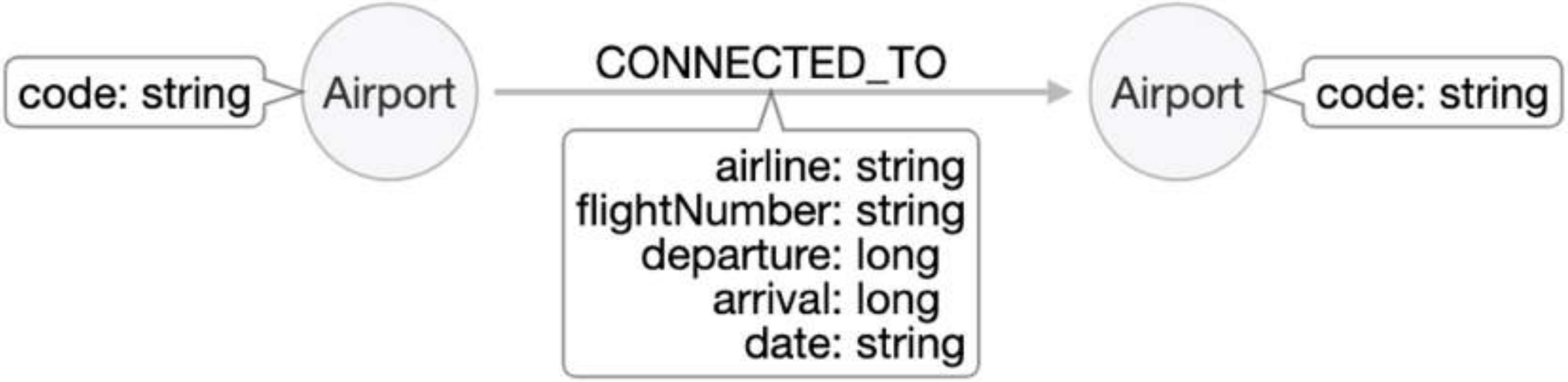
Note: This exercise has 2 steps. Estimated time to complete: 15 minutes.

Does the Model Need to be Changed?

The previous exercise asked this question:

- What are the airports and flight information for flight number 1016 for airline WN?

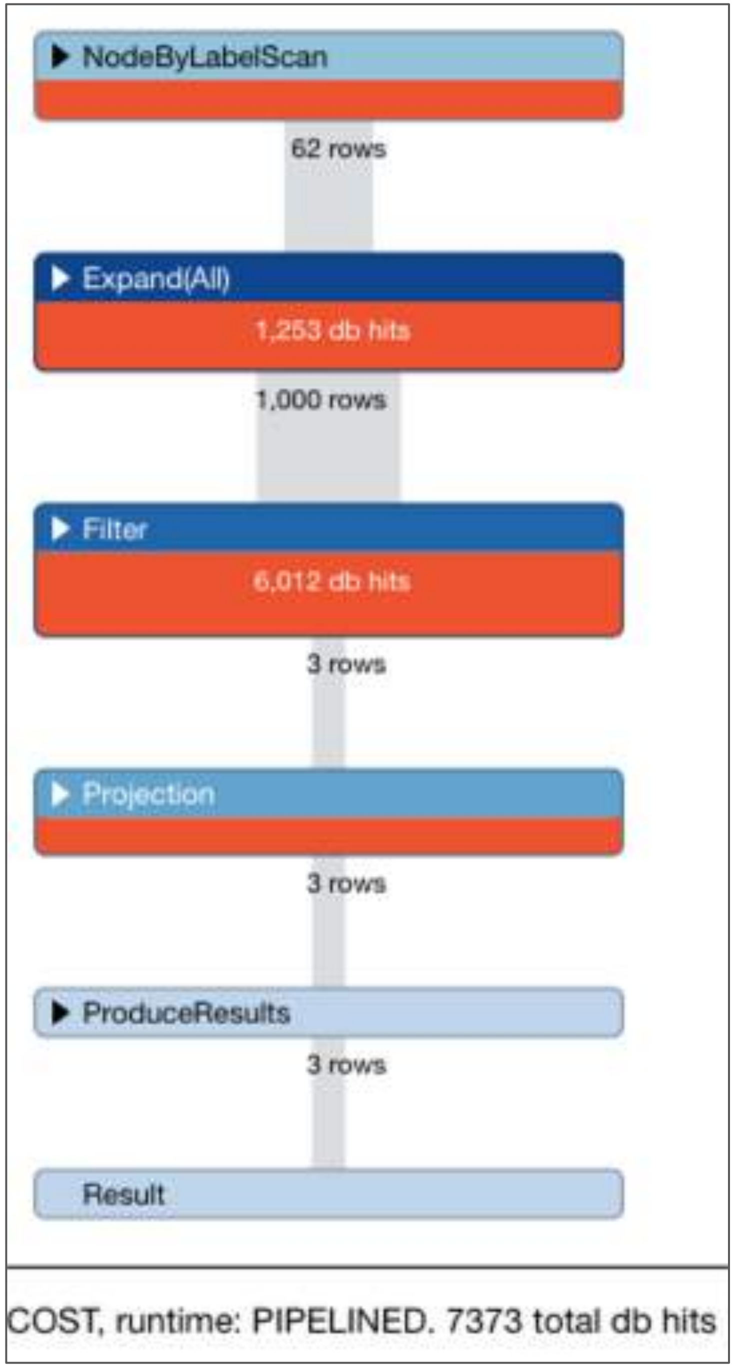
The current model:



Analyzing the Other Question

What are the airports and flight information for flight number 1016 for airline WN?

```
PROFILE
MATCH  (origin:Airport)-
       [connection:CONNECTED_TO]->
       (destination:Airport)
WHERE connection.airline = 'WN'
  AND connection.flightNumber = '1016'
RETURN origin.code, destination.code,
       connection.date,
       Connection.departure,
connection.arrival
```



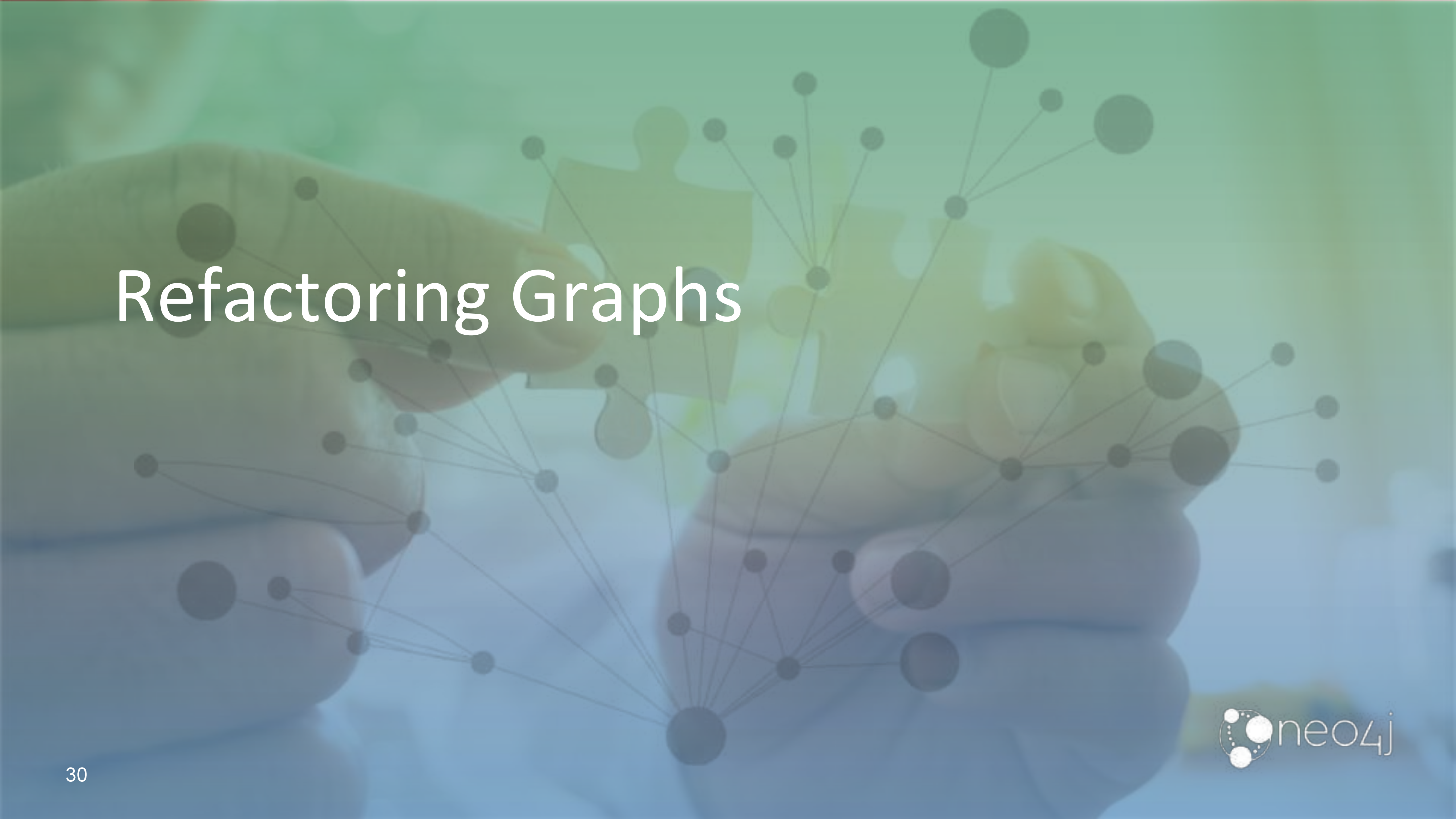
COST, runtime: PIPELINED. 7373 total db hits

Summary

You should now be able to:

- Profile queries against the graph
- Determine if a query can be improved

Refactoring Graphs



Refactoring Steps

1. Create constraints as needed
2. Execute the refactor:
   a. MATCH the data to be moved
   b. Create new nodes
   c. Create new relationships
3. Create indexes as needed
4. PROFILE all queries against the new model

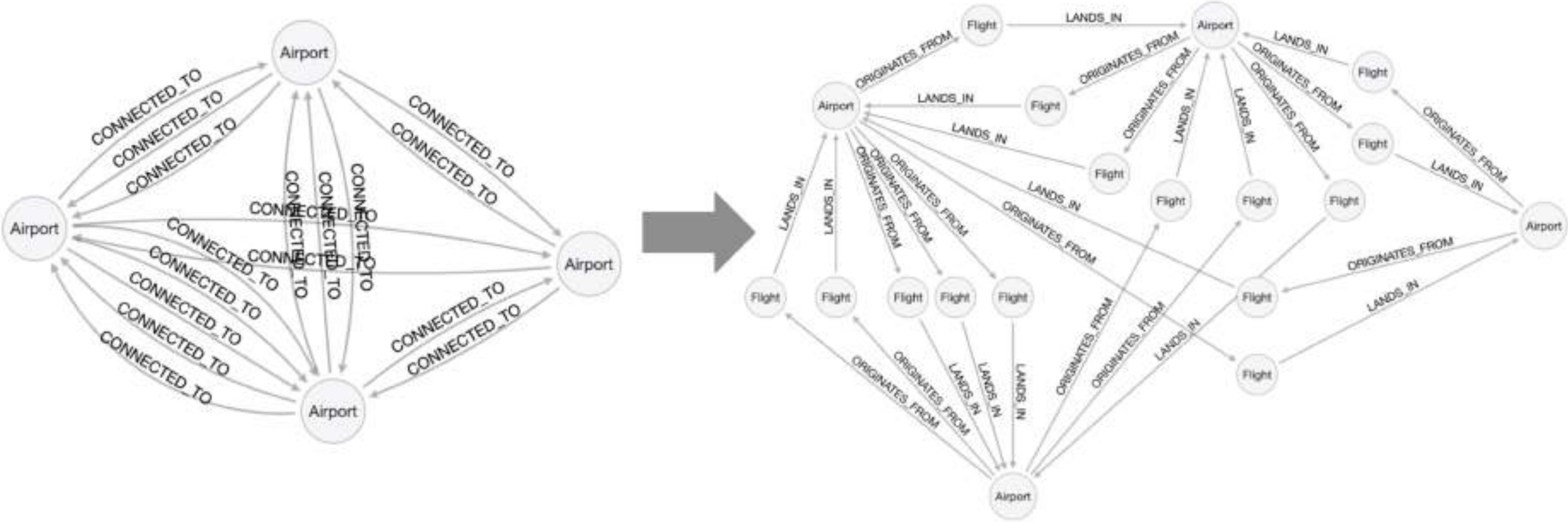
If the **new model** performs well for **all** queries

- **Delete** the **old model**
- Otherwise,
  keep **both models**

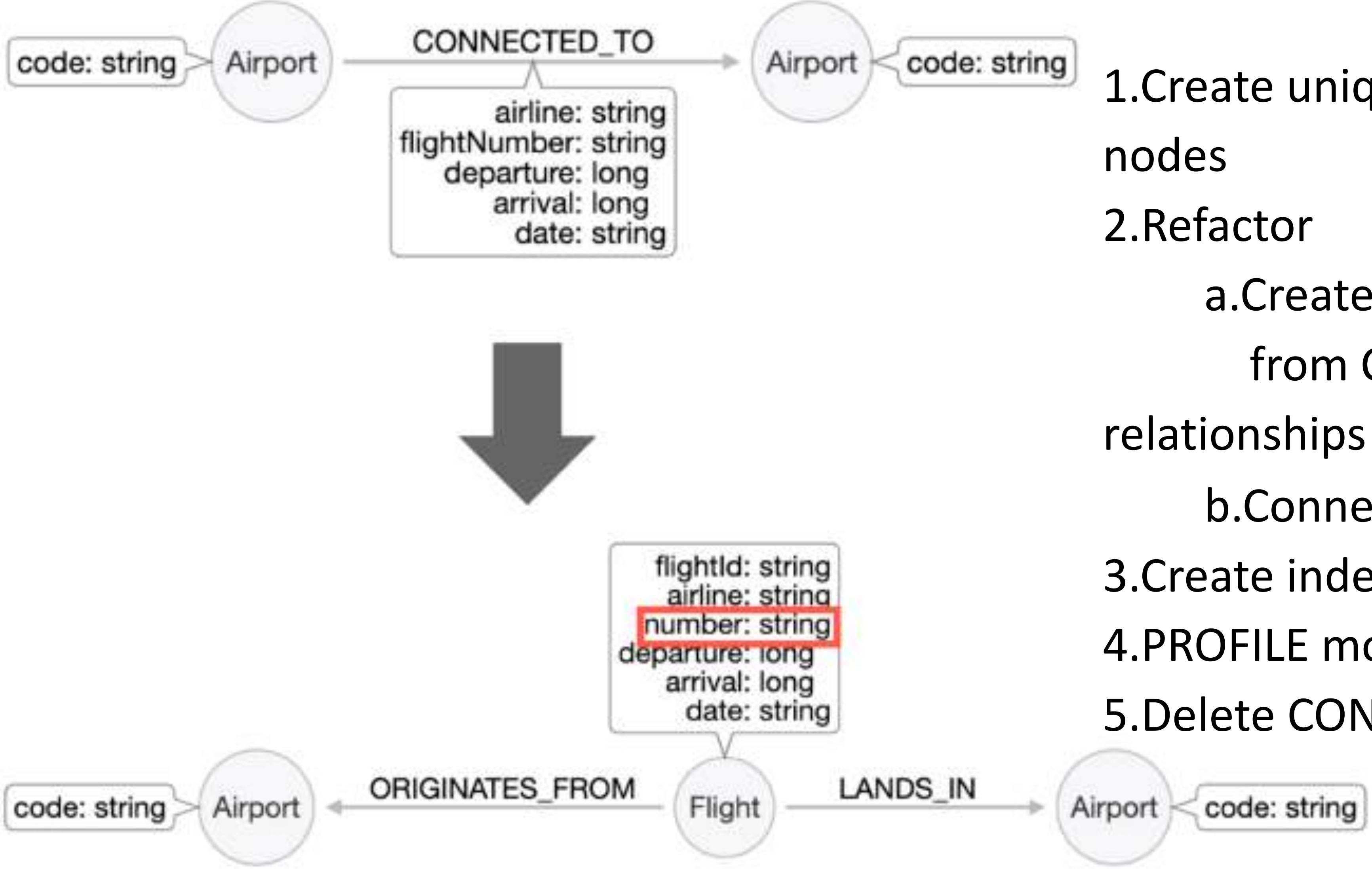
Evolving the Model

To solve this problem intermediate node Flight is introduced

- It is based upon the properties of the *CONNECTED_TO* relationship



Refactor Details



1. Create unique constraint on Flight nodes
2. Refactor
   a. Create Flight nodes from CONNECTED_TO relationships
   b. Connect Flights to Airports
3. Create index on Flight.number
4. PROFILE modified queries
5. Delete CONNECTED_TO relationships

Unique flightId

The ***Flight.flightId*** property is composed of five pieces of data

- This combination assures that all *Flight* nodes are **unique**

1. Airline
2. flightNumber
3. Code for the origin Airport
4. Code for the destination Airport
5. Date

Creating a Constraint

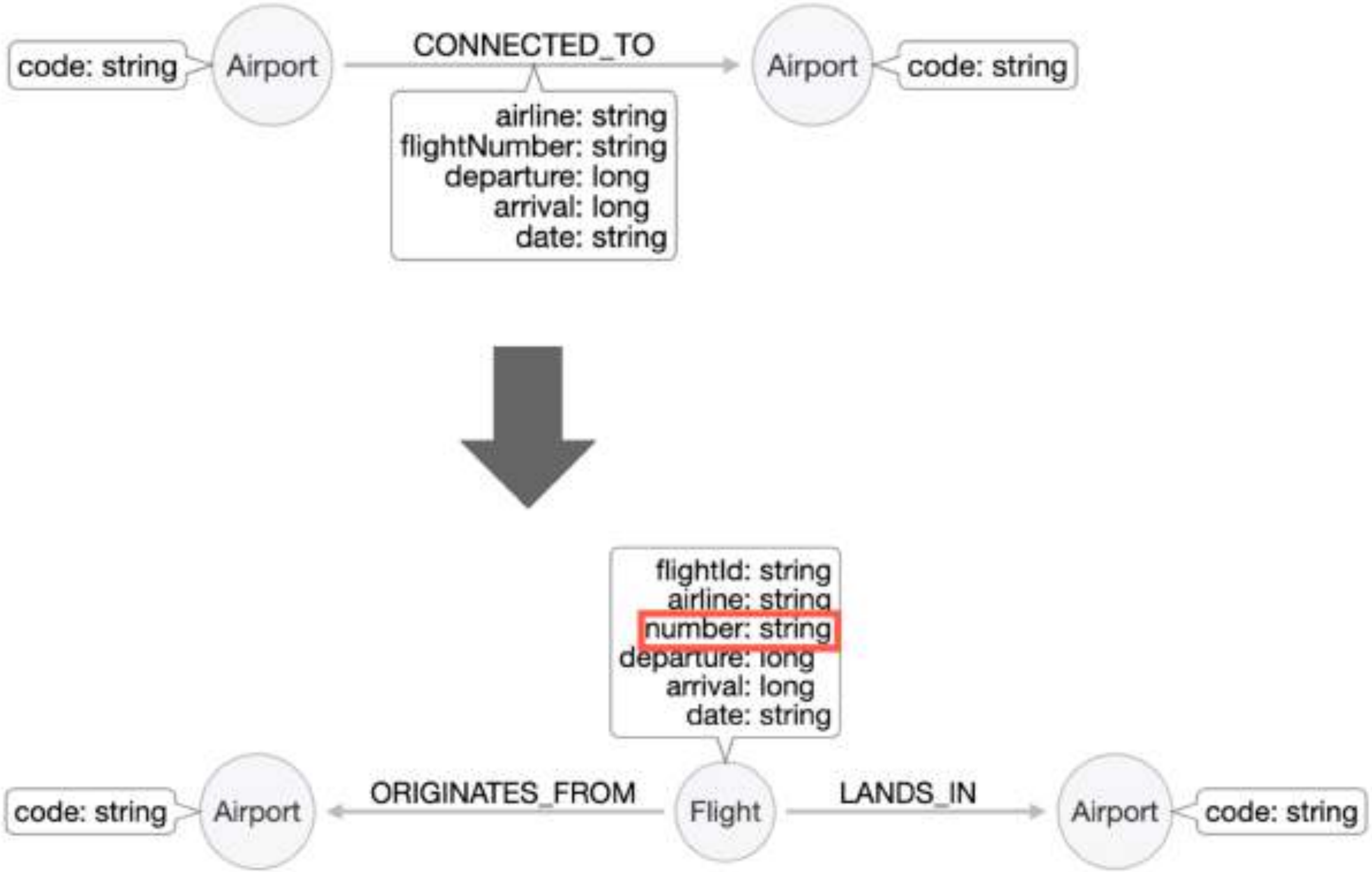
Prior to refactoring the uniqueness constraint is added

```
CREATE CONSTRAINT Flight_flightId_constraint ON (f:Flight)
       ASSERT f.flightId IS UNIQUE
```

Refactoring the Graph

```
MATCH (origin:Airport)-[connection:CONNECTED_TO]-
>(destination:Airport)
MERGE (newFlight:Flight {flightId: connection.airline +
connection.flightNumber +
       '_' + connection.date + '_' + origin.code + '_' +
destination.code })
ON CREATE SET newFlight.date = connection.date,
              newFlight.airline = connection.airline,
              newFlight.number = connection.flightNumber,
              newFlight.departure = connection.departure,
              newFlight.arrival = connection.arrival
MERGE (origin)<-[:ORIGINATES_FROM]-(newFlight)
MERGE (newFlight)-[:LANDS_IN]->(destination)
```

Two Models Exist In The Same Graph



Create an Index on Flight Number

```
CREATE INDEX Flight_number_index FOR (f:Flight) ON (f.number)
```

Profiling the Query

Question: What are the airports and flight information for flight number 1016 for airline WN?

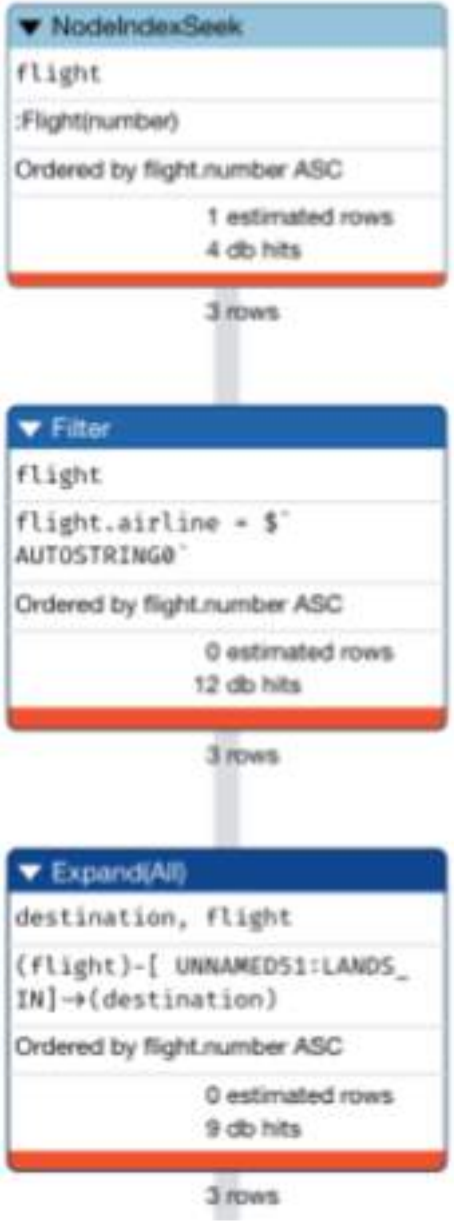
Original Query

```
PROFILE
MATCH  (origin:Airport)-[connection:CONNECTED_TO]-
>(destination:Airport)
WHERE connection.airline = 'WN' AND
connection.flightNumber = '1016'
```

New Query

```
PROFILE
MATCH (origin)<-[:ORIGINATES_FROM]-(flight:Flight)-
      [:LANDS_IN]->(destination)
WHERE flight.airline = 'WN' AND
      flight.number = '1016' RETURN origin, destination,
flight
```

Profile Result



▼ NodeIndexSeek

flight

:Flight(number)

Ordered by flight.number ASC

1 estimated rows
4 db hits

3 rows

▼ Filter

flight

flight.airline = $`AUTOSTRING0`

Ordered by flight.number ASC

0 estimated rows
12 db hits

3 rows

▼ Expand(All)

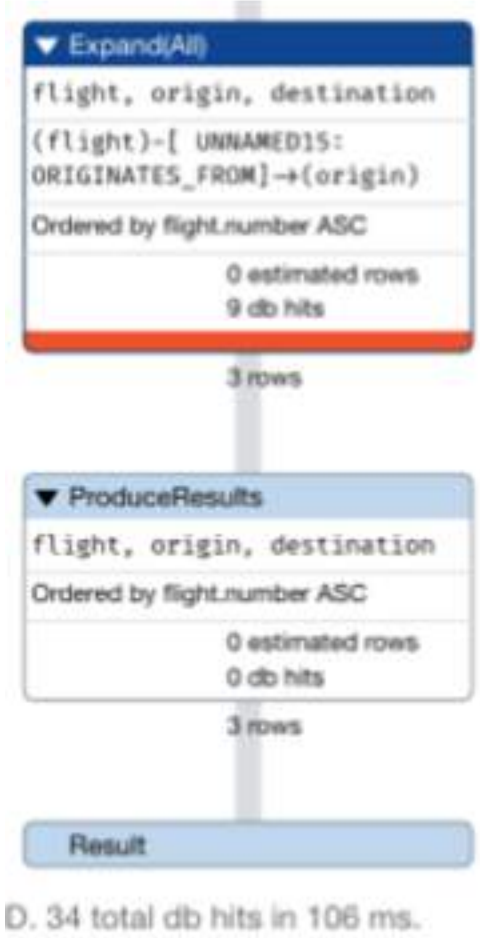
destination, flight

(flight)-[UNNAMED51:LANDS_IN]→(destination)

Ordered by flight.number ASC

0 estimated rows
9 db hits

3 rows



▼ Expand(All)

flight, origin, destination

(flight)-[UNNAMED15:ORIGINATES_FROM]→(origin)

Ordered by flight.number ASC

0 estimated rows
9 db hits

3 rows

▼ ProduceResults

flight, origin, destination

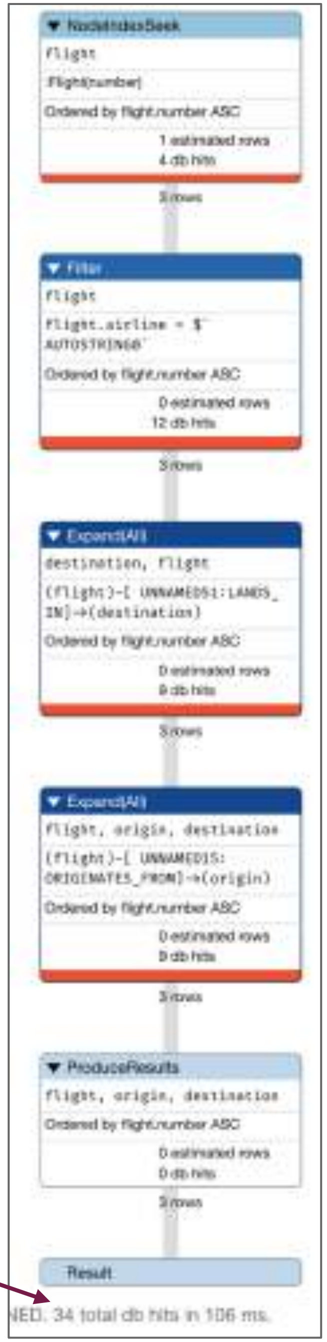
Ordered by flight.number ASC

0 estimated rows
0 db hits

3 rows

Result

D. 34 total db hits in 106 ms.



▼ NodeIndexSeek

flight

:Flight(number)

Ordered by flight.number ASC

1 estimated rows
4 db hits

3 rows

▼ Filter

flight

flight.airline = $`AUTOSTRING0`

Ordered by flight.number ASC

0 estimated rows
12 db hits

3 rows

▼ Expand(All)

destination, flight

(flight)-[UNNAMED51:LANDS_IN]→(destination)

Ordered by flight.number ASC

0 estimated rows
9 db hits

3 rows

▼ Expand(All)

flight, origin, destination

(flight)-[UNNAMED15:ORIGINATES_FROM]→(origin)

Ordered by flight.number ASC

0 estimated rows
9 db hits

3 rows

▼ ProduceResults

flight, origin, destination

Ordered by flight.number ASC

0 estimated rows
0 db hits

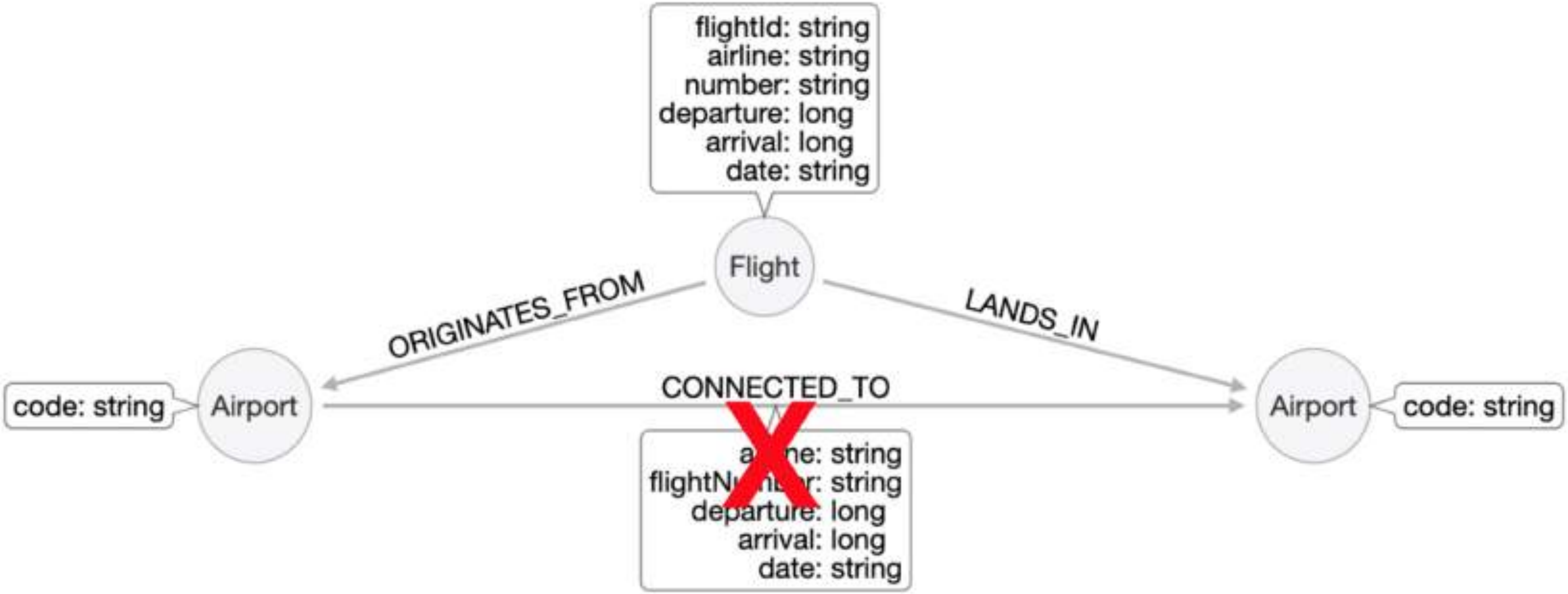
3 rows

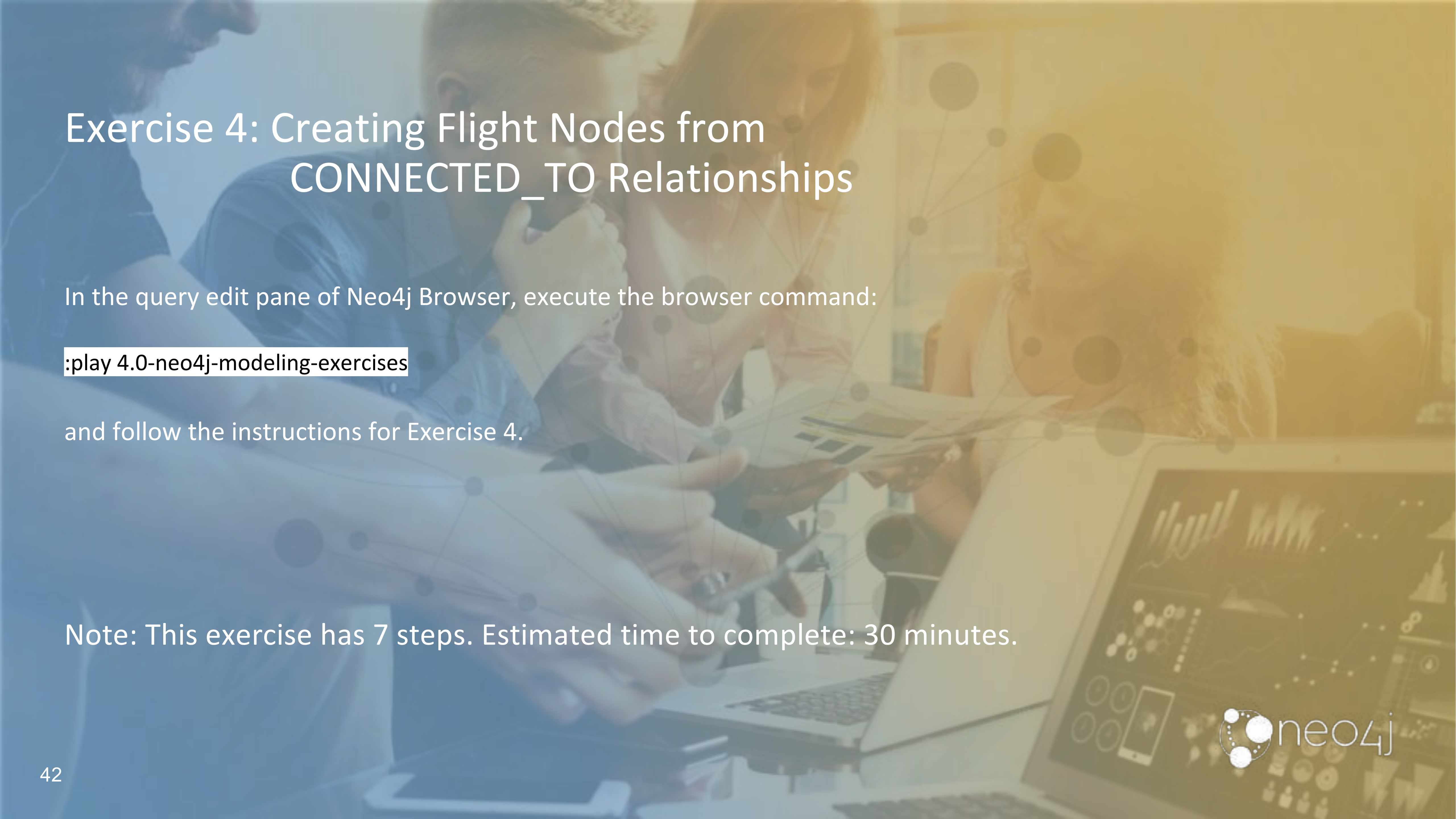
Result

NED. 34 total db hits in 106 ms.

Removing the Old Model

```
MATCH ()-[connection:CONNECTED_TO]->()
DELETE connection
```





Exercise 4: Creating Flight Nodes from CONNECTED_TO Relationships

In the query edit pane of Neo4j Browser, execute the browser command:

```
:play 4.0-neo4j-modeling-exercises
```

and follow the instructions for Exercise 4.

Note: This exercise has 7 steps. Estimated time to complete: 30 minutes.

An Additional Domain Question

Here is another question is that needs to be answered by the application:

- As a frequent traveller I want to find flights from <origin> to <destination> on <date> so that I can book my business flight

For example:

- Find all flights going from Los Angeles (LAS)
  to Chicago Midway International (MDW)
  on January 3rd, 2019.

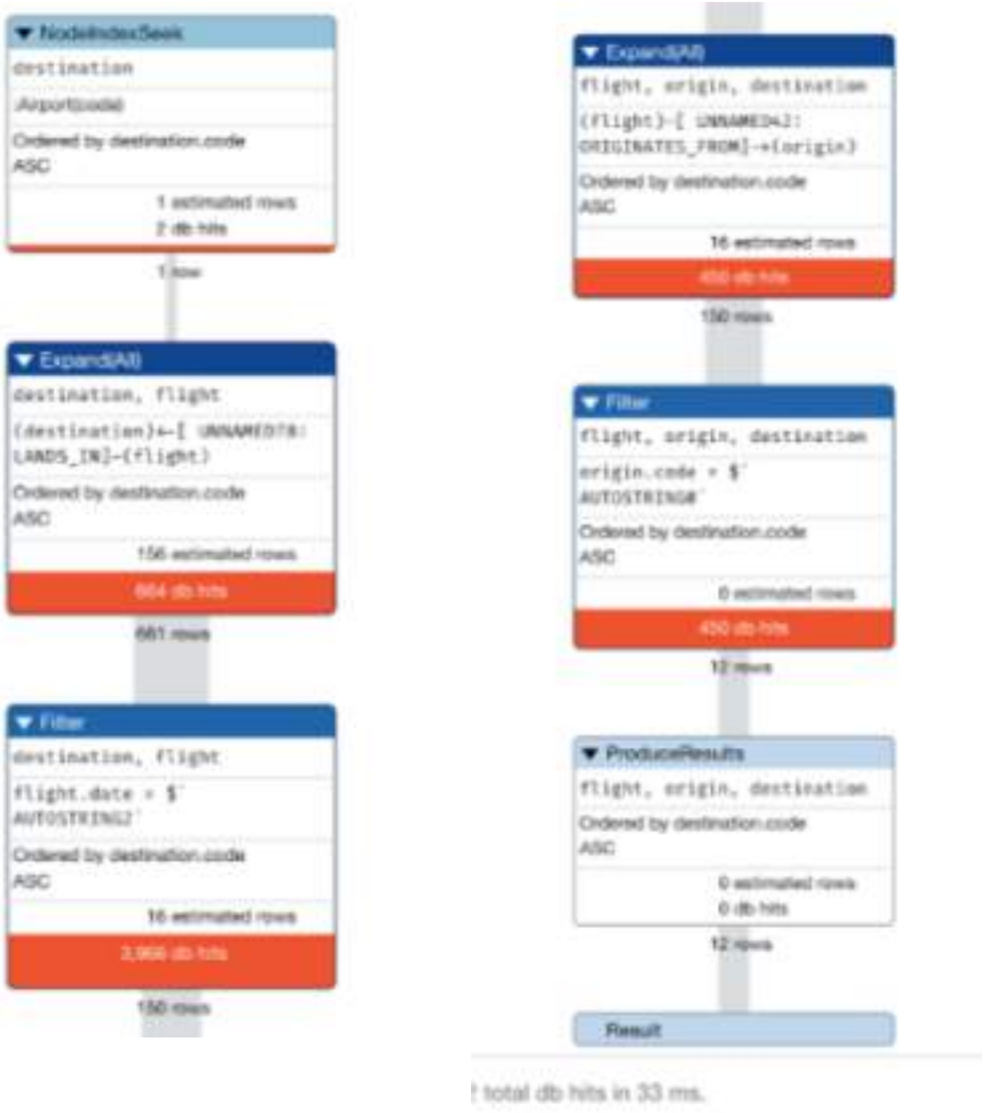
Implementing the Query

```
MATCH (origin:Airport {code: ‘LAS'})
    <-[:ORIGINATES_FROM]-(flight:Flight)-[:LANDS_IN]->
    (destination:Airport {code: 'MDW'})
WHERE flight.date = '2019-1-3'
RETURN origin, destination, flight
```

The next exercise makes use of this query

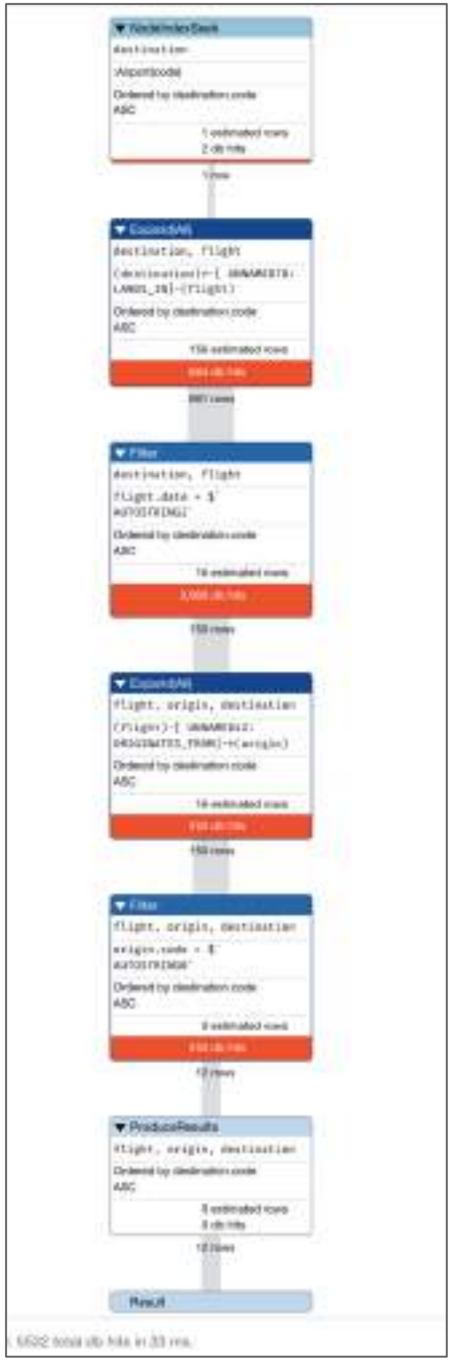
- From the US Bureau of Transportation data 10k nodes will be added to the graph

Profiling the Query

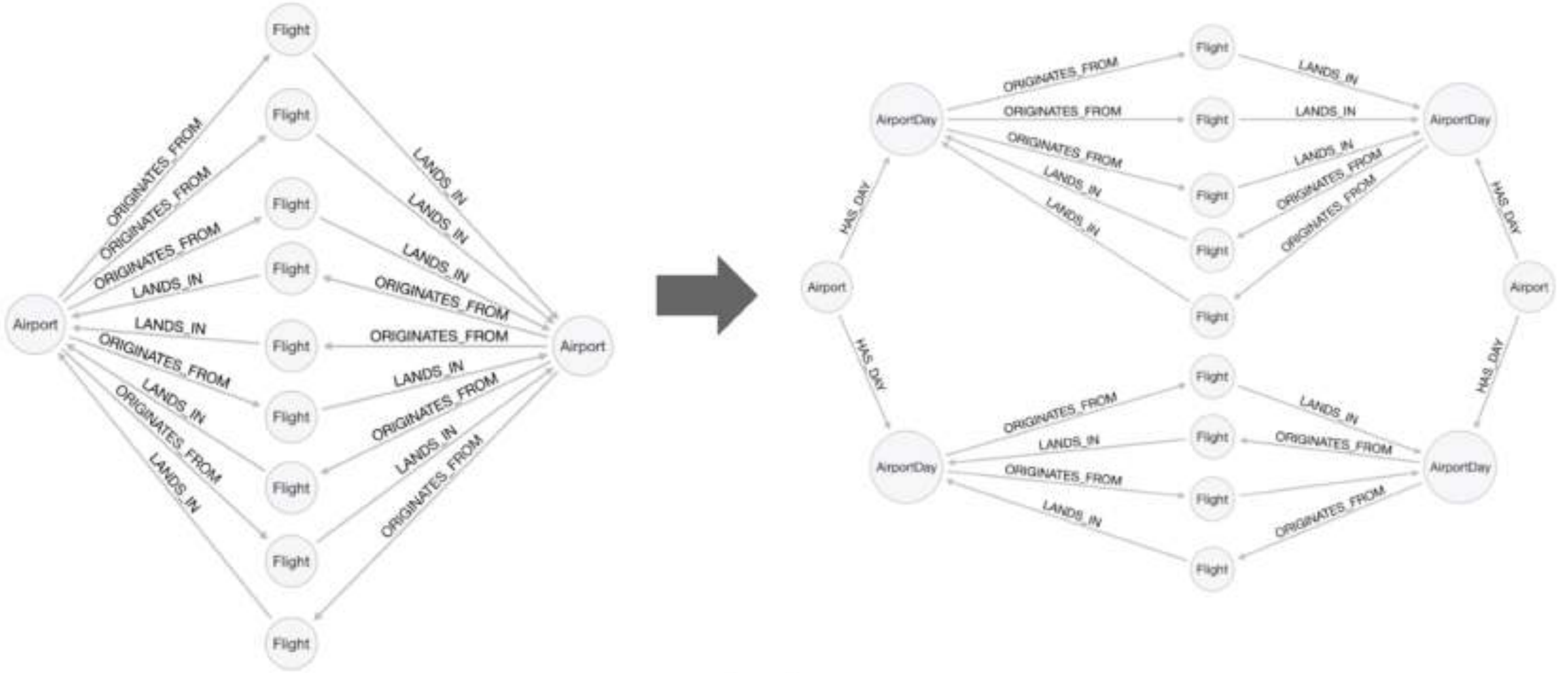


```
▼ NodeIndexSeek
destination
:Airport(code)
Ordered by destination.code
ASC
                    1 estimated rows
                    2 db hits
        1 row

▼ Expand(All)
destination, flight
(destination)<-[ UNNAMED78:
LANDS_IN]-(flight)
Ordered by destination.code
ASC
                  156 estimated rows
                    664 db hits
        661 rows

▼ Filter
destination, flight
flight.date = $`
AUTOSTRING2`
Ordered by destination.code
ASC
                   16 estimated rows
                   3,966 db hits
        150 rows

▼ Expand(All)
flight, origin, destination
(flight)-[ UNNAMED42:
ORIGINATES_FROM]->(origin)
Ordered by destination.code
ASC
                   16 estimated rows
                    450 db hits
        150 rows

▼ Filter
flight, origin, destination
origin.code = $`
AUTOSTRING0`
Ordered by destination.code
ASC
                    0 estimated rows
                    450 db hits
        12 rows

▼ ProduceResults
flight, origin, destination
Ordered by destination.code
ASC
                    0 estimated rows
                    0 db hits
        12 rows

Result

! total db hits in 33 ms.
```

The result after profiling the query that contains 10k nodes



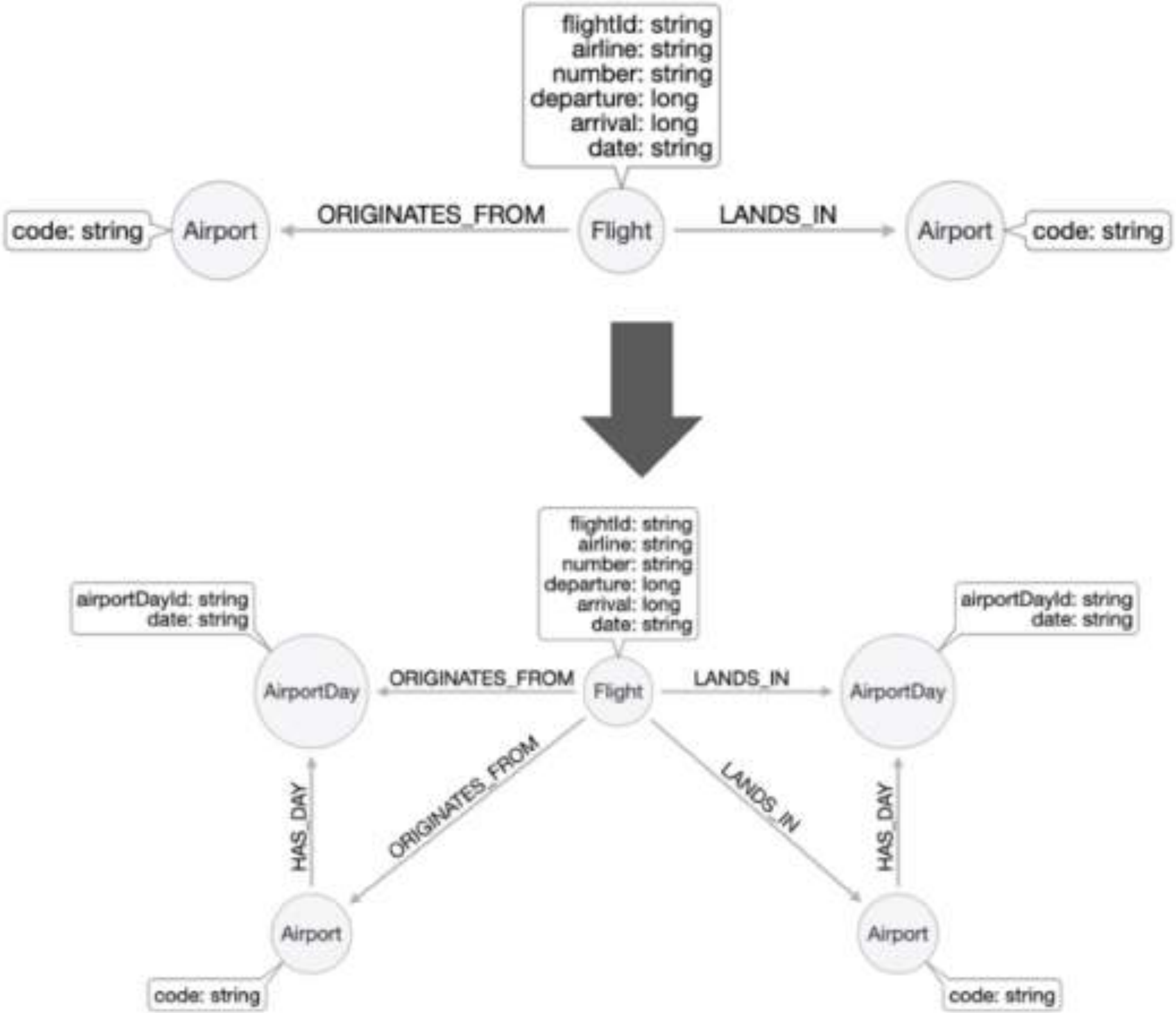
Performing Another Refactor



Note: A node is only pulled out if you are going to query through it, otherwise a property will suffice

Refactor Details

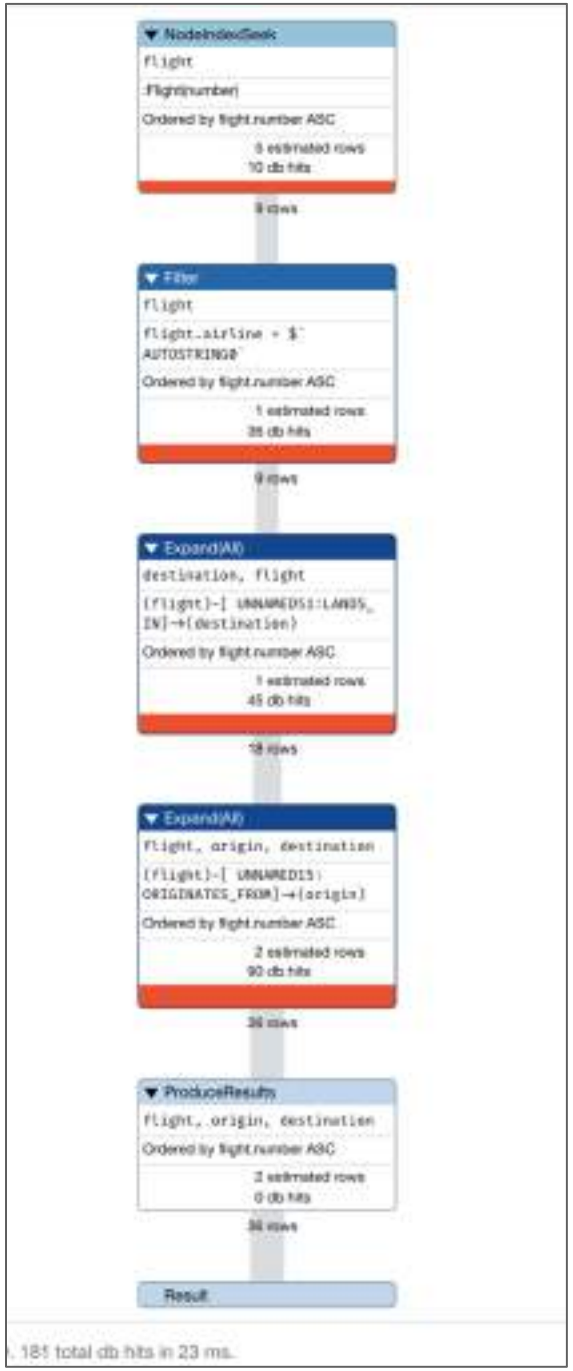
```
CREATE CONSTRAINT

AirportDay_airportDayId_constraint
  ON (a:AirportDay)
  ASSERT a.airportDayId IS UNIQUE
```



Refactor Implementation

```
MATCH (origin:Airport)<-[:ORIGINATES_FROM]-(flight:Flight)-
      [:LANDS_IN]->(destination:Airport)
MERGE (originAirportDay:AirportDay {airportDayId: origin.code +
'_' + flight.date})
SET originAirportDay.date = flight.date
MERGE (destinationAirportDay:AirportDay
      {airportDayId: destination.code + '_' + flight.date})
SET destinationAirportDay.date = flight.date
MERGE (origin)-[:HAS_DAY]->(originAirportDay)
MERGE (flight)-[:ORIGINATES_FROM]->(originAirportDay)
MERGE (flight)-[:LANDS_IN]->(destinationAirportDay)
MERGE (destination)-[:HAS_DAY]->(destinationAirportDay)
```

Profiling the First Query

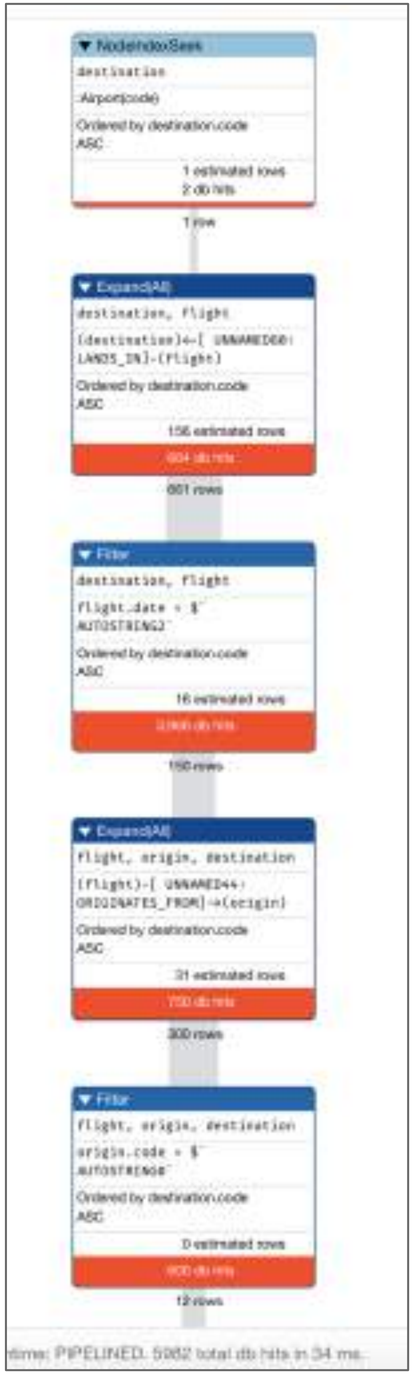


```
▼ NodeIndexSeek
flight
:Flight(number)
Ordered by flight.number ASC
5 estimated rows
10 db hits

9 rows

▼ Filter
flight
flight.airline = $`
AUTOSTRING0`
Ordered by flight.number ASC
1 estimated rows
36 db hits

9 rows

▼ Expand(All)
destination, flight
(flight)-[ UNNAMED51:LANDS_
IN]→(destination)
Ordered by flight.number ASC
1 estimated rows
45 db hits

18 rows

▼ Expand(All)
flight, origin, destination
(flight)-[ UNNAMED15:
ORIGINATES_FROM]→(origin)
Ordered by flight.number ASC
2 estimated rows
90 db hits

36 rows

▼ ProduceResults
flight, origin, destination
Ordered by flight.number ASC
2 estimated rows
0 db hits

36 rows

Result

. 181 total db hits in 23 ms.
```

```
PROFILE
MATCH (origin)<-[:ORIGINATES_FROM]-
(flight:Flight)-
      [:LANDS_IN]->(destination)
WHERE flight.airline = 'WN' AND
      flight.number = '1016' RETURN origin,
destination, flight
```

Profiling the Original Second Query

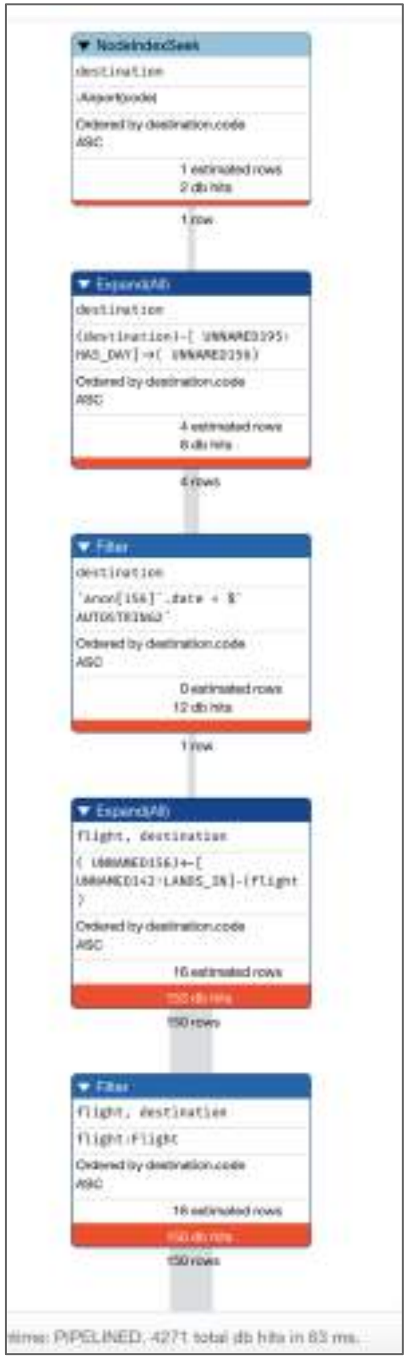
```
PROFILE MATCH (origin:Airport
{code: 'LAS'})
    <-[:ORIGINATES_FROM]-
(flight:Flight)-
    [:LANDS_IN]->
    (destination:Airport {code:
'MDW'})
WHERE flight.date = '2019-1-3'
RETURN origin, destination, flight
```



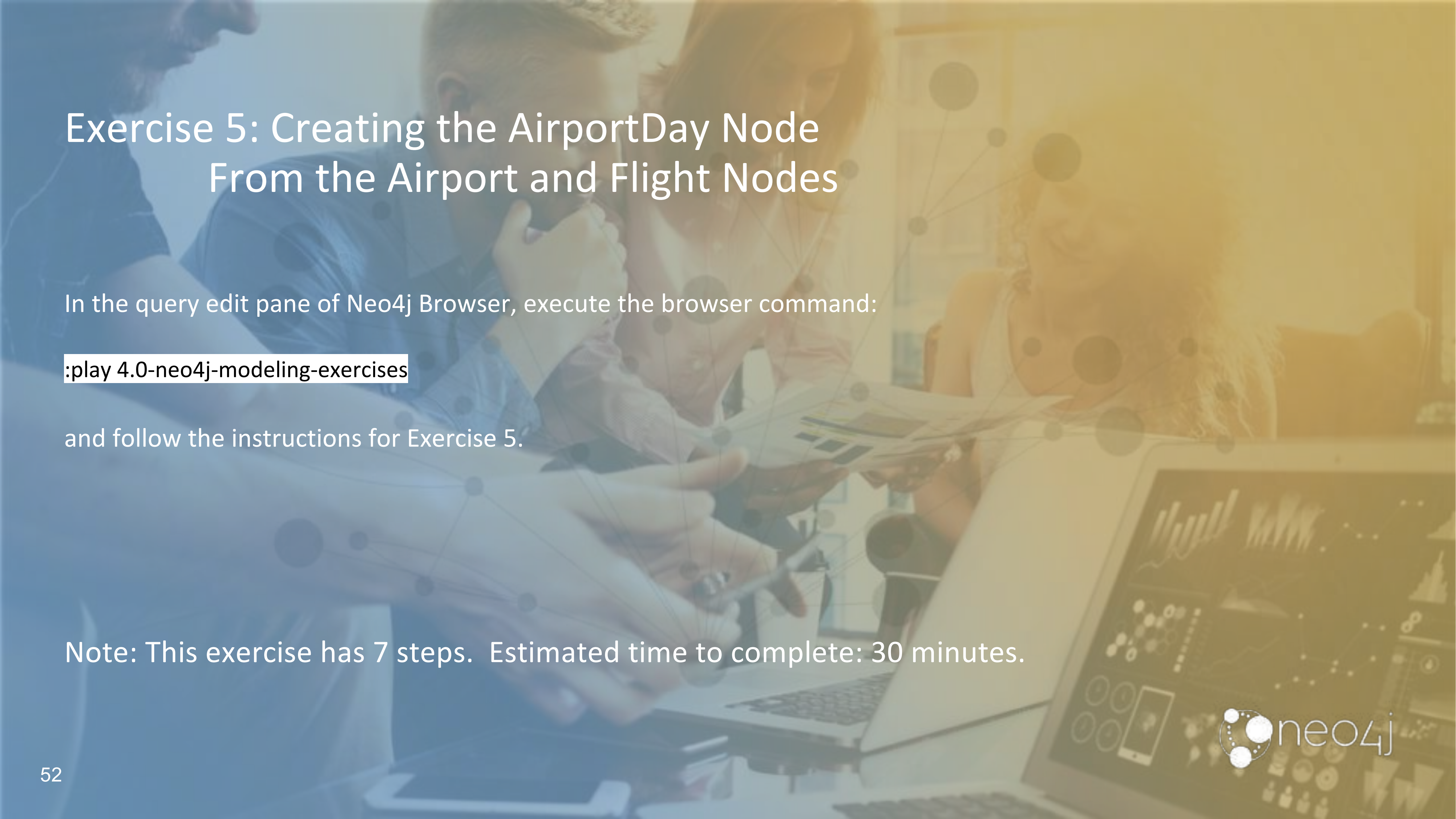
```
▼ NodeIndexSeek
destination
:Airport(code)
Ordered by destination.code
ASC
                1 estimated rows
                2 db hits
1 row

▼ Expand(All)
destination, flight
(destination)<-[ UNNAMED60:
LANDS_IN]-(flight)
Ordered by destination.code
ASC
                156 estimated rows
                604 db hits
601 rows

▼ Filter
destination, flight
flight.date = $`
AUTOSTRING2`
Ordered by destination.code
ASC
                16 estimated rows
                [illegible] db hits
150 rows

▼ Expand(All)
flight, origin, destination
(flight)-[ UNNAMED44:
ORIGINATES_FROM]->(origin)
Ordered by destination.code
ASC
                31 estimated rows
                750 db hits
300 rows

▼ Filter
flight, origin, destination
origin.code = $`
AUTOSTRING0`
Ordered by destination.code
ASC
                0 estimated rows
                600 db hits
12 rows

time: PIPELINED. 5062 total db hits in 34 ms.
```

Profiling the Revised Second Query

```
PROFILE MATCH (origin:Airport {code:
'LAS'})-
    [:HAS_DAY]->(:AirportDay {date:
'2019-1-3'})<-
    [:ORIGINATES_FROM]-(flight:Flight),
    (flight)-[:LANDS_IN]->
    (:AirportDay {date: '2019-1-3'})<-
    [:HAS_DAY]-(destination:Airport
{code: 'MDW'})
RETURN origin, destination, flight
```



```
▼ NodeIndexSeek
destination
:Airport(code)
Ordered by destination.code
ASC
                1 estimated rows
                2 db hits
        1 row

▼ Expand(All)
destination
(destination)-[ UNNAMED195:
HAS_DAY]->( UNNAMED158)
Ordered by destination.code
ASC
                4 estimated rows
                8 db hits
        4 rows

▼ Filter
destination
`anon[156]`.date = $`
AUTOSTRING2`
Ordered by destination.code
ASC
                0 estimated rows
                12 db hits
        1 row

▼ Expand(All)
flight, destination
( UNNAMED156)<-[
UNNAMED142:LANDS_IN]-(flight
)
Ordered by destination.code
ASC
                16 estimated rows
                150 db hits
        150 rows

▼ Filter
flight, destination
flight:Flight
Ordered by destination.code
ASC
                16 estimated rows
                150 db hits
        150 rows

time: PIPELINED. 4271 total db hits in 83 ms.
```



Exercise 5: Creating the AirportDay Node From the Airport and Flight Nodes

In the query edit pane of Neo4j Browser, execute the browser command:

```
:play 4.0-neo4j-modeling-exercises
```

and follow the instructions for Exercise 5.

Note: This exercise has 7 steps. Estimated time to complete: 30 minutes.

More Questions for the Model?

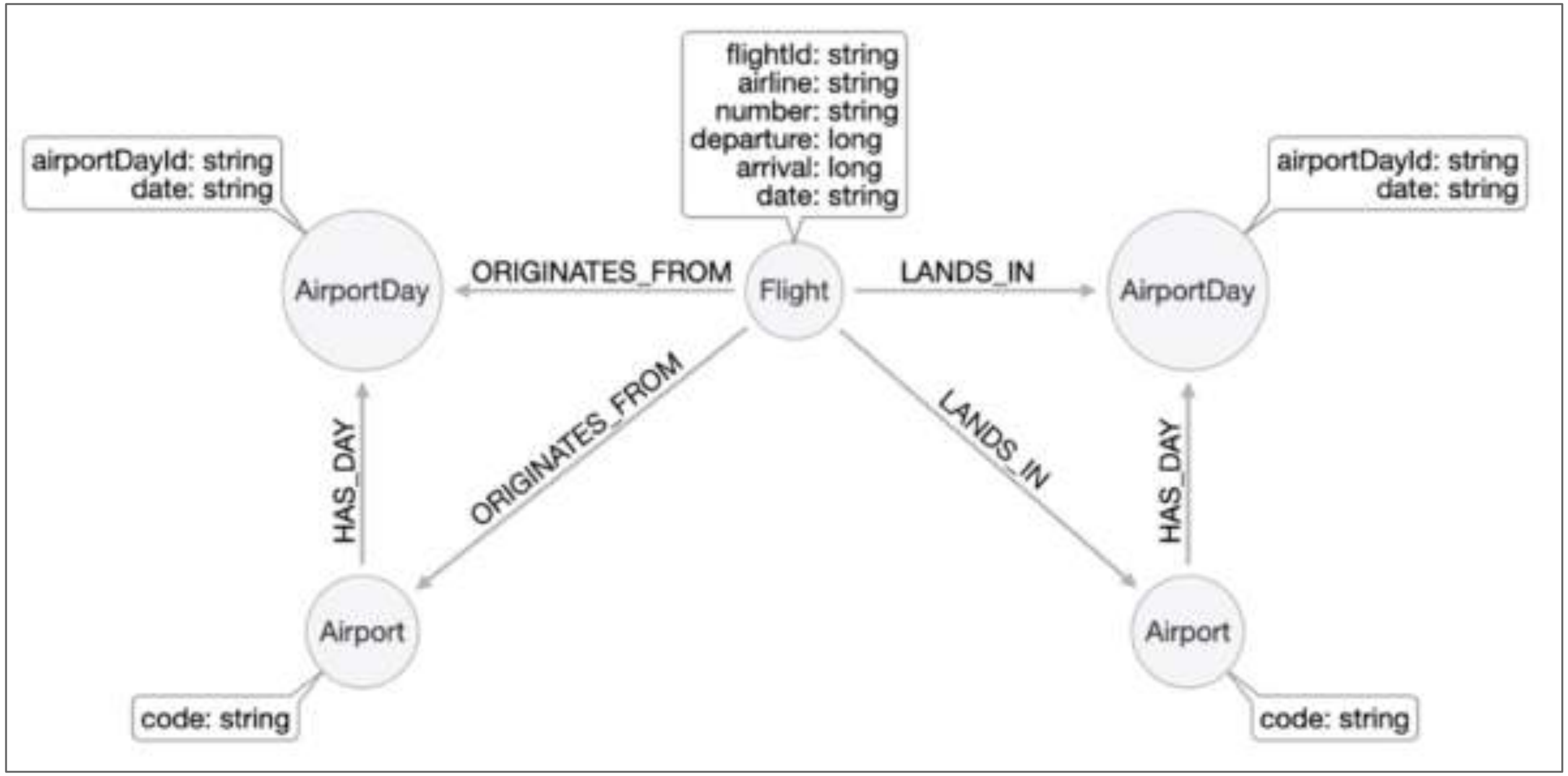
The model performs well

for these questions:

1. What are the airports and flight information for flight number xx for airline yy?
2. Find all the flights going from xx to yy on the date zz.

What if this question is added:

- Which airport has the most incoming flights?

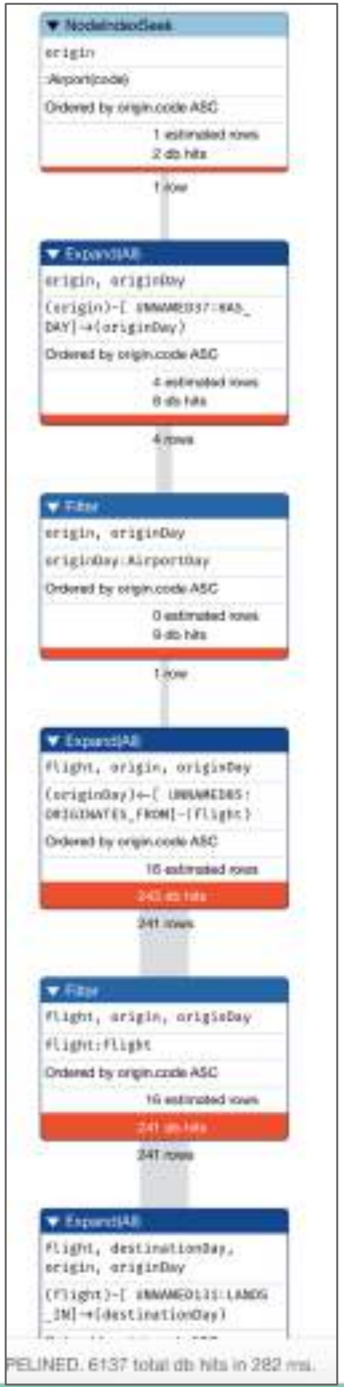


Another Question for the Model

Suppose we added this question:

- What are the flights from LAS that arrive at MDW on 2019-1-3?

```
PROFILE
MATCH (origin:Airport {code: 'LAS'})-[:HAS_DAY]->
      (originDay:AirportDay),
      (originDay)<-[:ORIGINATES_FROM]-(flight:Flight),
      (flight)-[:LANDS_IN]->(destinationDay),
      (destinationDay:AirportDay)<-[:HAS_DAY]-
      (destination:Airport {code: 'MDW'})
WHERE originDay.date = '2019-1-3' AND
      destinationDay.date = '2019-1-3'
RETURN flight.date, flight.number, flight.airline,
       flight.departure, flight.arrival
ORDER BY flight.date, flight.departure
```



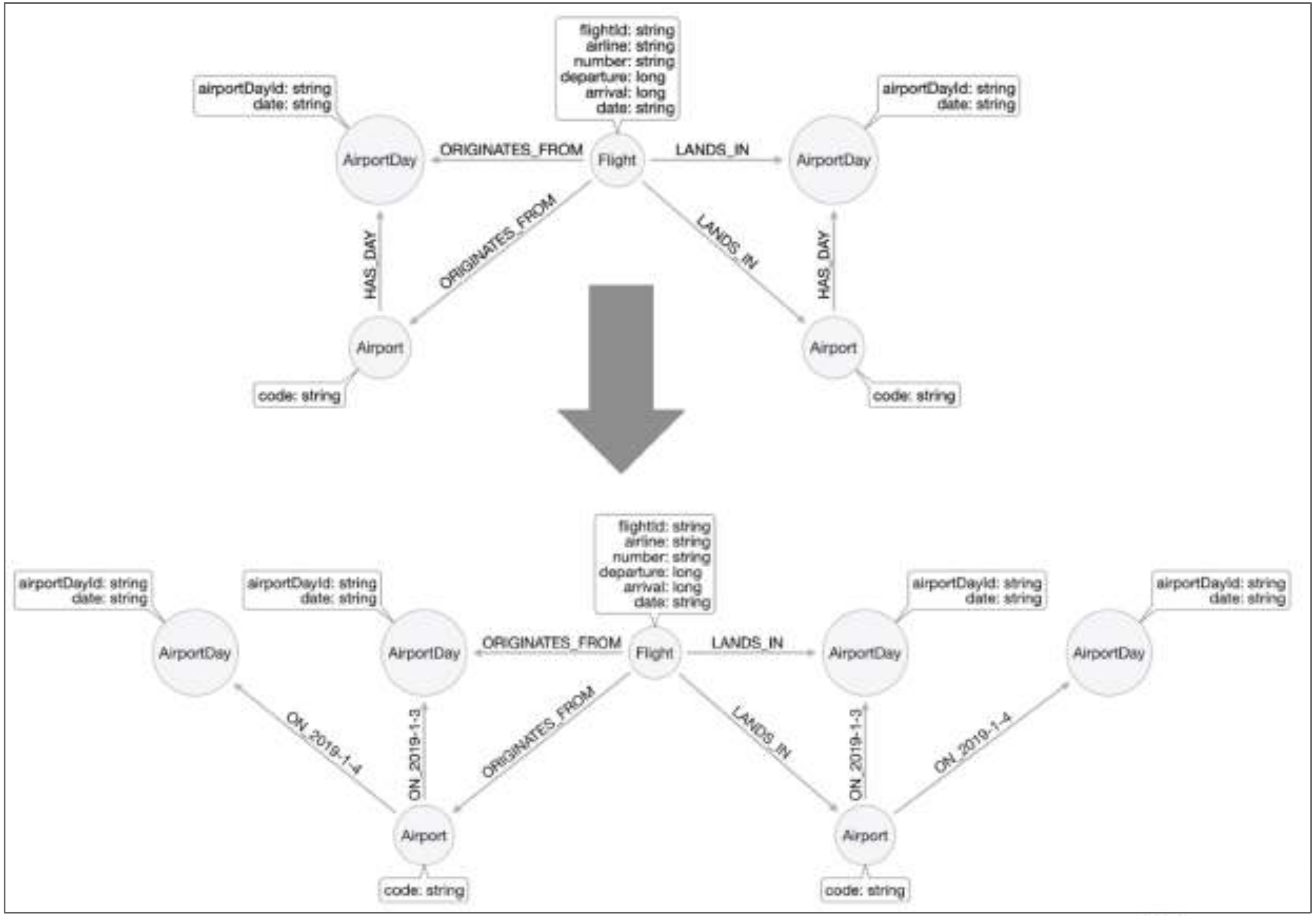
```
▼ NodeIndexSeek
origin
:Airport(code)
Ordered by origin.code ASC
                    1 estimated rows
                    2 db hits
                    1 row

▼ ExpandAll
origin, originDay
(origin)-[ UNNAMED37:HAS_
DAY]->(originDay)
Ordered by origin.code ASC
                    4 estimated rows
                    8 db hits
                    4 rows

▼ Filter
origin, originDay
originDay:AirportDay
Ordered by origin.code ASC
                    0 estimated rows
                    9 db hits
                    1 row

▼ ExpandAll
flight, origin, originDay
(originDay)<-[ UNNAMED85:
ORIGINATES_FROM]-(flight)
Ordered by origin.code ASC
                    16 estimated rows
                    243 db hits
                    241 rows

▼ Filter
flight, origin, originDay
flight:Flight
Ordered by origin.code ASC
                    16 estimated rows
                    241 db hits
                    241 rows

▼ ExpandAll
flight, destinationDay,
origin, originDay
(flight)-[ UNNAMED131:LANDS
_IN]->(destinationDay)

PELINED. 6137 total db hits in 282 ms.
```

Refactoring for Specific Relationships

A best practice for graph data modeling:

- Make relationships more specific if it helps query performance

Modify *HAS_DAY* relationship to be *ON_2019-1-3*, *ON_2019-1-4*, etc.



APOC to the Rescue!

APOC supports creating relationships based upon data in the graph

syntax:

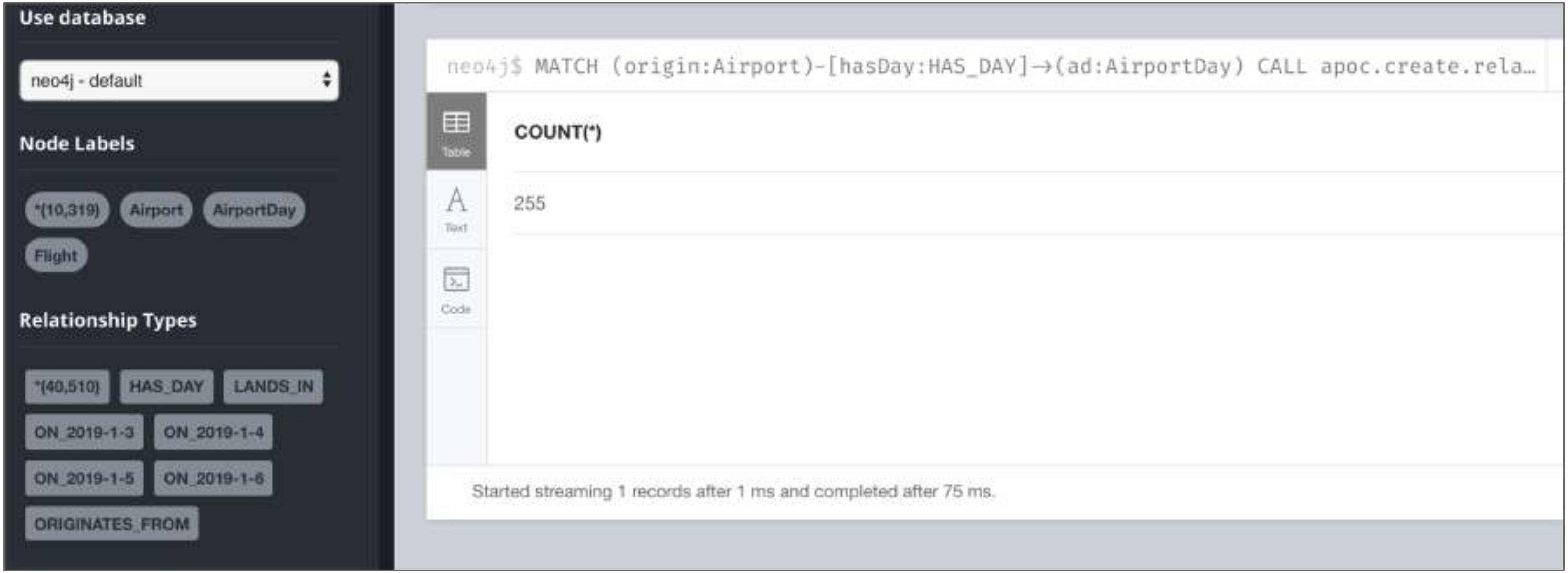
```
apoc.create.relationship(startNode(<relationship-variable>),
                         '<new-relationship-value>',
                         {<relationship-property list},
                         endNode(<relationship-variable>)
                         )
                         YIELD rel
```

Creating Specialized Relationships with APOC

Code to transform *HAS_DAY* relationships to specific relationships:

```
MATCH (origin:Airport)-[hasDay:HAS_DAY]->(ad:AirportDay)
CALL apoc.create.relationship(startNode(hasDay),
                              'ON_' + ad.date,
                              {},
                              endNode(hasDay) ) YIELD rel
RETURN COUNT(*)
```

Refactoring Result



```
neo4j$ MATCH (origin:Airport)-[hasDay:HAS_DAY]→(ad:AirportDay) CALL apoc.create.rela…
```

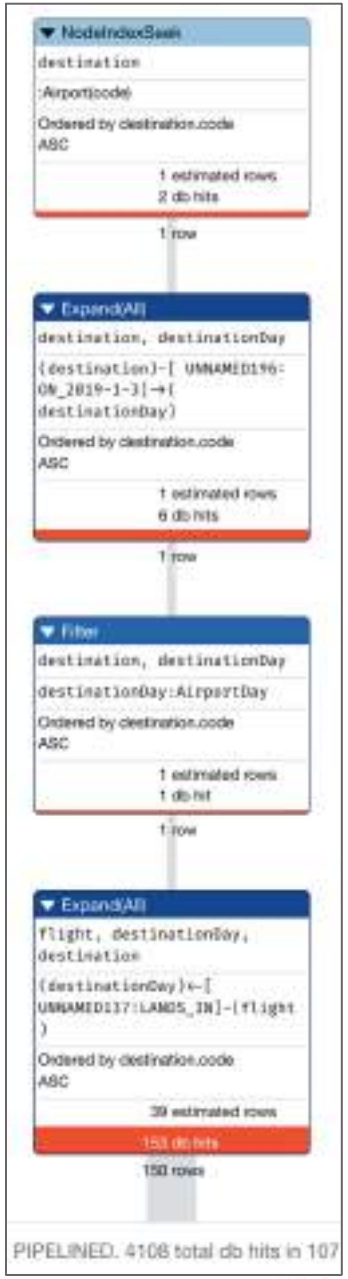
COUNT(*)
255

Started streaming 1 records after 1 ms and completed after 75 ms.

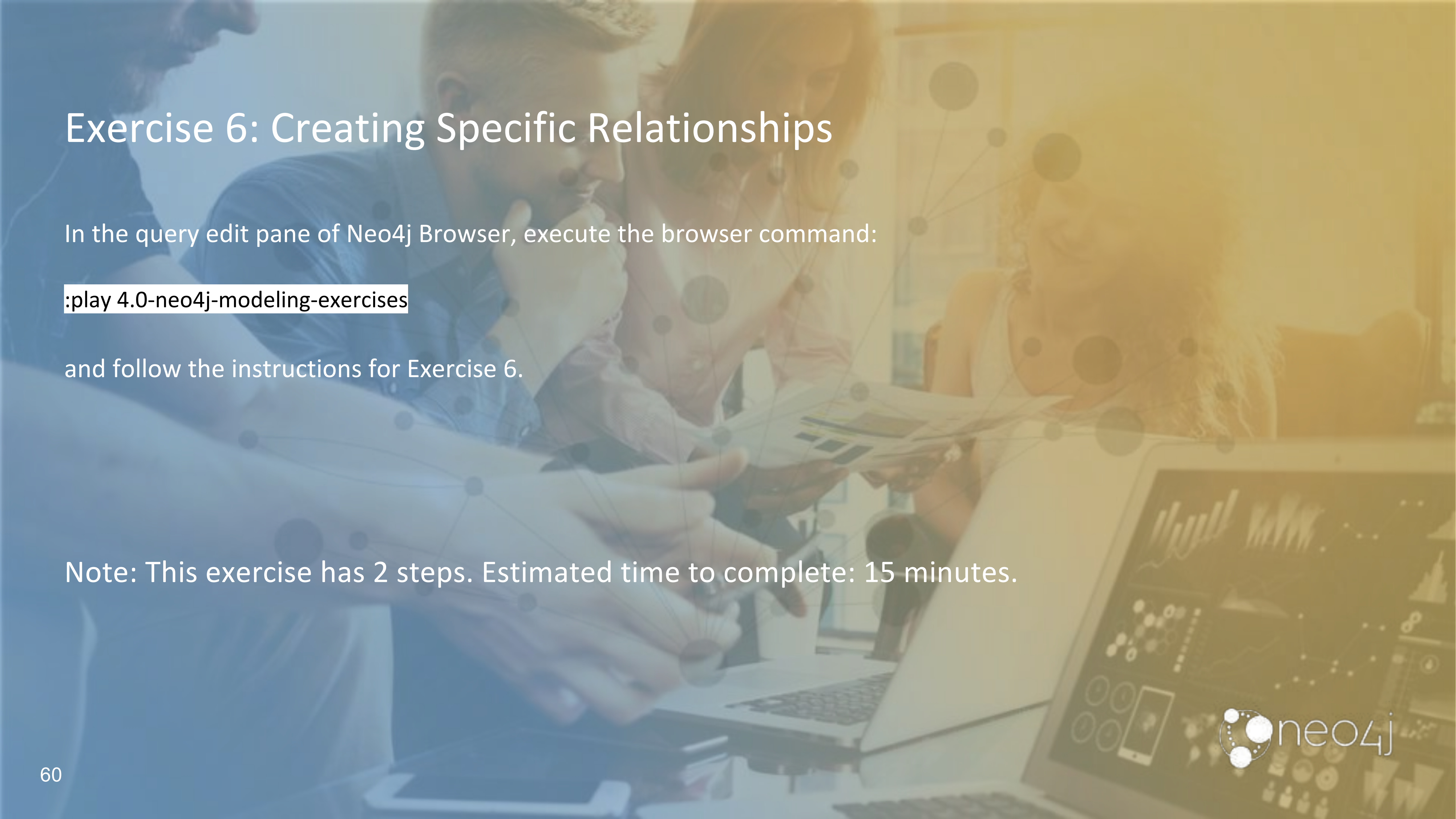
Does the Query Improve?

Rewrite the query, since the model has changed:

```
PROFILE
MATCH (origin:Airport {code: 'LAS'})-[:`ON_2019-1-3`]->
      (originDay:AirportDay),
      (originDay)<-[:ORIGINATES_FROM]-(flight:Flight),
      (flight)-[:LANDS_IN]->(destinationDay),
      (destinationDay:AirportDay)<-[:`ON_2019-1-3`]
      -(destination:Airport {code: 'MDW'})
RETURN flight.date, flight.number, flight.airline,
       flight.departure, flight.arrival
ORDER BY flight.date, flight.departure
```



```
▼ NodeIndexSeek
destination
:Airport(code)
Ordered by destination.code
ASC
                 1 estimated rows
                 2 db hits
                 1 row
▼ Expand(All)
destination, destinationDay
(destination)-[ UNNAMED196:
ON_2019-1-3]→(
destinationDay)
Ordered by destination.code
ASC
                 1 estimated rows
                 6 db hits
                 1 row
▼ Filter
destination, destinationDay
destinationDay:AirportDay
Ordered by destination.code
ASC
                 1 estimated rows
                 1 db hit
                 1 row
▼ Expand(All)
flight, destinationDay,
destination
(destinationDay)←-[
UNNAMED137:LANDS_IN]-(flight
)
Ordered by destination.code
ASC
                39 estimated rows
                153 db hits
                150 rows
PIPELINED. 4108 total db hits in 107
```



Exercise 6: Creating Specific Relationships

In the query edit pane of Neo4j Browser, execute the browser command:

```
:play 4.0-neo4j-modeling-exercises
```

and follow the instructions for Exercise 6.

Note: This exercise has 2 steps. Estimated time to complete: 15 minutes.

Refactoring Large Graphs

The heap size needs to be adapted to match, or operate in batches.

Increase these values for the server in the **neo4j.conf** file to support the need for additional memory:

- `dbms.memory.heap.initial_size=2G` (default 512m)
- `dbms.memory.heap.max_size=2G` (default 1G)

Batching the Refactoring Process

1. Tag all the nodes that need to processed with a temporary label
   - For example *Process* could be used as the temporary label

```
MATCH (f:Flight)
SET f:Process
```

1. Iterate over the subset of nodes flagged with the temporary label
   - This is done using `LIMIT`

   a. `Execute the refactoring code`

   b. `Remove the temporary label from the nodes`

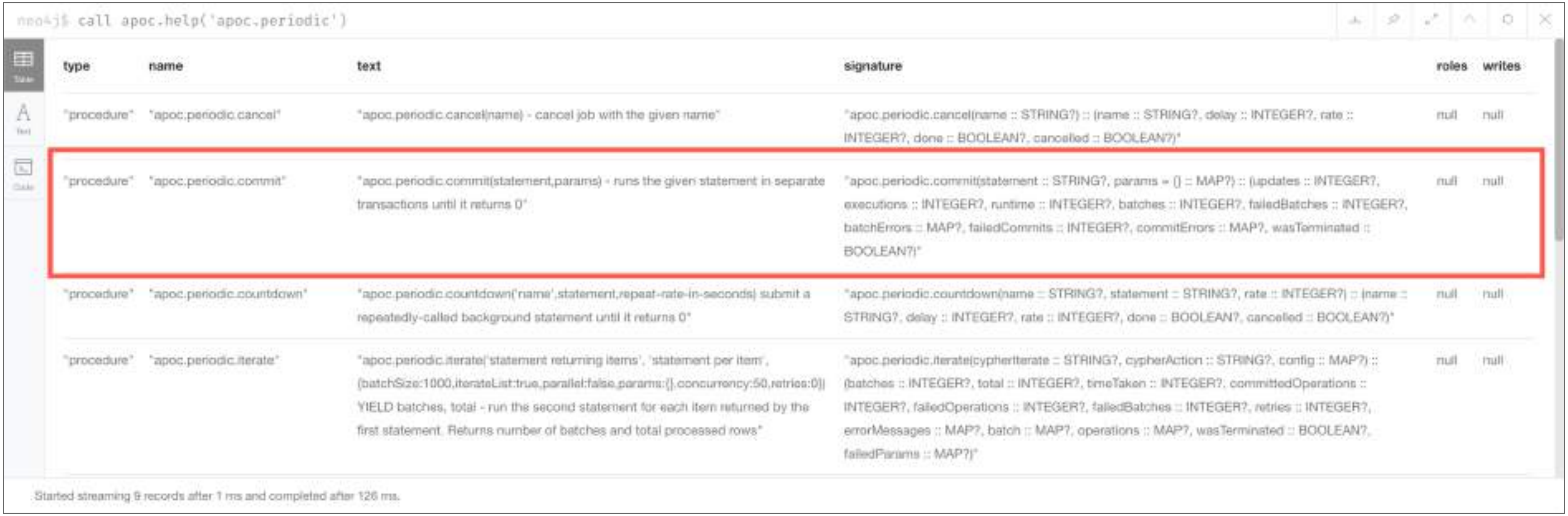
   c. `Return a count of how many rows were processed`

1. The refactoring is is done when the count reaches 0

Example Code for a Batch

```
MATCH (flight:Process)
WITH flight LIMIT 500

MATCH (origin:Airport)<-[:ORIGINATES_FROM]-(flight)-[:LANDS_IN]->(destination:Airport)

MERGE (originAirportDay:AirportDay {airportDayId: origin.code + "_" + flight.date})
ON CREATE SET originAirportDay.date = flight.date

MERGE (destinationAirportDay:AirportDay {airportDayId: destination.code + "_" + flight.date})
ON CREATE SET destinationAirportDay.date = flight.date

MERGE (origin)-[:HAS_DAY]->(originAirportDay)
MERGE (originAirportDay)<-[:ORIGINATES_FROM]-(flight)
MERGE (flight)-[:LANDS_IN]-(destinationAirportDay)
MERGE (destination)-[:HAS_DAY]->(destinationAirportDay)

REMOVE flight:Process
RETURN count(*)
```

Using apoc.periodoc.commit



```
neo4j$ call apoc.help('apoc.periodic')
```

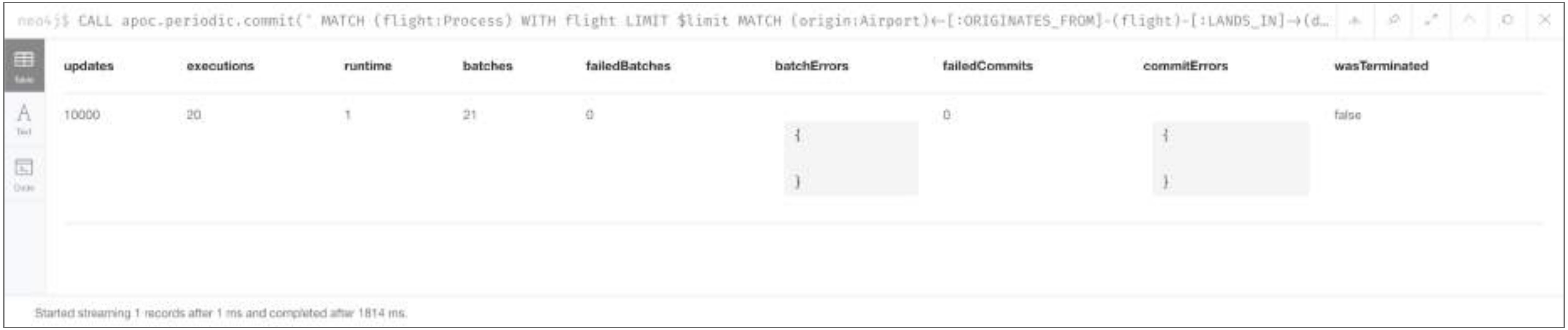
type	name	text	signature	roles	writes
"procedure"	"apoc.periodic.cancel"	"apoc.periodic.cancel(name) - cancel job with the given name"	"apoc.periodic.cancel(name :: STRING?) :: (name :: STRING?, delay :: INTEGER?, rate :: INTEGER?, done :: BOOLEAN?, cancelled :: BOOLEAN?)"	null	null
"procedure"	"apoc.periodic.commit"	"apoc.periodic.commit(statement,params) - runs the given statement in separate transactions until it returns 0"	"apoc.periodic.commit(statement :: STRING?, params = {} :: MAP?) :: (updates :: INTEGER?, executions :: INTEGER?, runtime :: INTEGER?, batches :: INTEGER?, failedBatches :: INTEGER?, batchErrors :: MAP?, failedCommits :: INTEGER?, commitErrors :: MAP?, wasTerminated :: BOOLEAN?)"	null	null
"procedure"	"apoc.periodic.countdown"	"apoc.periodic.countdown('name',statement,repeat-rate-in-seconds) submit a repeatedly-called background statement until it returns 0"	"apoc.periodic.countdown(name :: STRING?, statement :: STRING?, rate :: INTEGER?) :: (name :: STRING?, delay :: INTEGER?, rate :: INTEGER?, done :: BOOLEAN?, cancelled :: BOOLEAN?)"	null	null
"procedure"	"apoc.periodic.iterate"	"apoc.periodic.iterate('statement returning items', 'statement per item', {batchSize:1000,iterateList:true,parallel:false,params:{},concurrency:50,retries:0}) YIELD batches, total - run the second statement for each item returned by the first statement. Returns number of batches and total processed rows"	"apoc.periodic.iterate(cypherIterate :: STRING?, cypherAction :: STRING?, config :: MAP?) :: (batches :: INTEGER?, total :: INTEGER?, timeTaken :: INTEGER?, committedOperations :: INTEGER?, failedOperations :: INTEGER?, failedBatches :: INTEGER?, retries :: INTEGER?, errorMessages :: MAP?, batch :: MAP?, operations :: MAP?, wasTerminated :: BOOLEAN?, failedParams :: MAP?)"	null	null

Started streaming 9 records after 1 ms and completed after 126 ms.

Batching with APOC

```
CALL apoc.periodic.commit('
MATCH (flight:Process)
WITH flight LIMIT $limit

MATCH (origin:Airport)<-[:ORIGINATES_FROM]-(flight)-[:LANDS_IN]->(destination:Airport)

MERGE (originAirportDay:AirportDay {airportDayId: origin.code + "_" + flight.date})
ON CREATE SET originAirportDay.date = flight.date

MERGE (destinationAirportDay:AirportDay {airportDayId: destination.code + "_" + flight.date})
ON CREATE SET destinationAirportDay.date = flight.date

MERGE (origin)-[:HAS_DAY]->(originAirportDay)
MERGE (originAirportDay)<-[:ORIGINATES_FROM]-(flight)
MERGE (flight)-[:LANDS_IN]-(destinationAirportDay)
MERGE (destination)-[:HAS_DAY]->(destinationAirportDay)

REMOVE flight:Process
RETURN count(*)

',{limit:500}
)
```

Result of the Batch Processing



```
neo4j$ CALL apoc.periodic.commit(" MATCH (flight:Process) WITH flight LIMIT $limit MATCH (origin:Airport)←[:ORIGINATES_FROM]-(flight)-[:LANDS_IN]→(d…
```

updates	executions	runtime	batches	failedBatches	batchErrors	failedCommits	commitErrors	wasTerminated

10000	20	1	21	0	{
					}
						0	{
							}
								false

Started streaming 1 records after 1 ms and completed after 1814 ms.

Exercise 7: Refactoring Large Graphs

In the query edit pane of Neo4j Browser, execute the browser command:

```
:play 4.0-neo4j-modeling-exercises
```

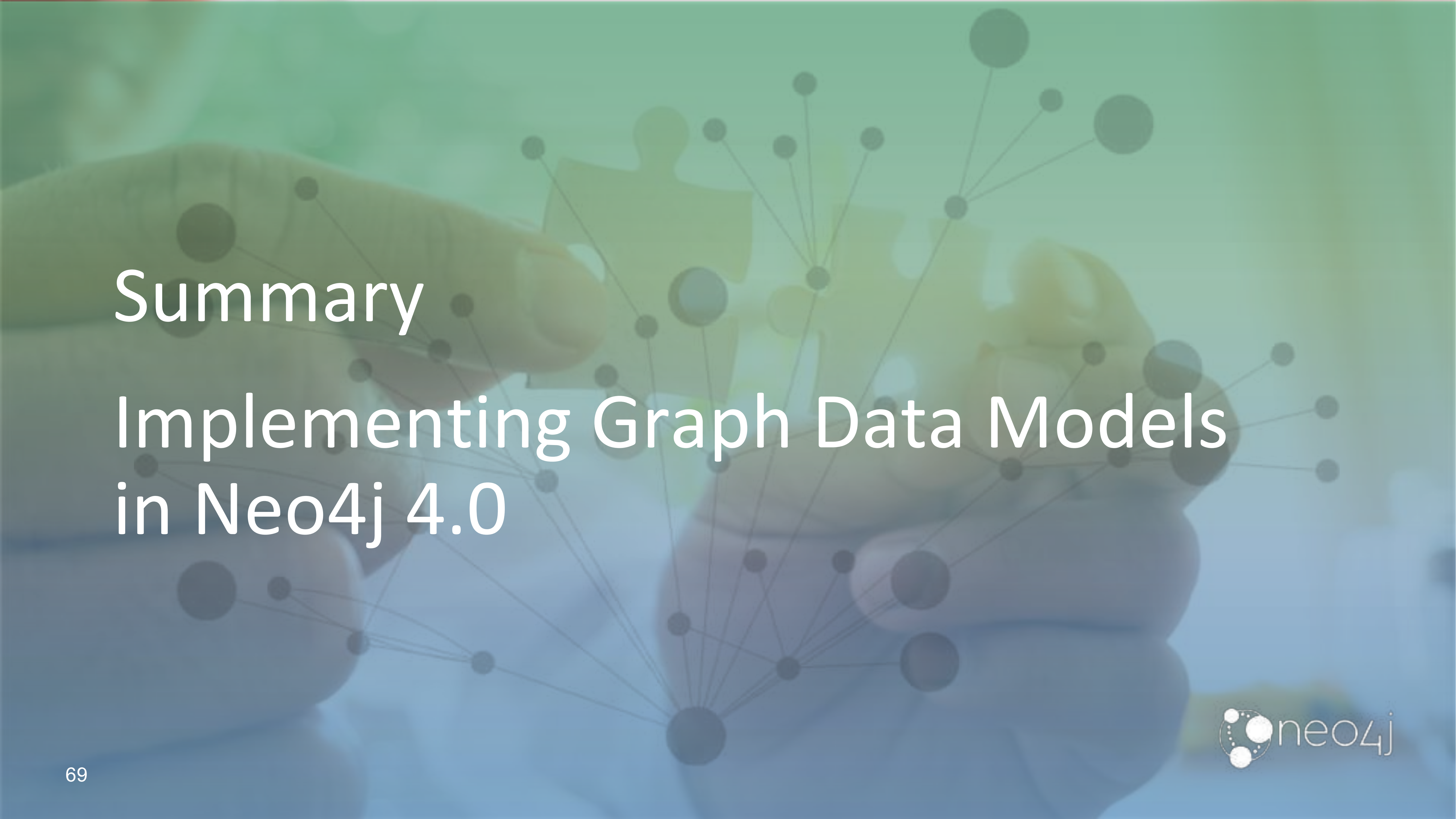
and follow the instructions for Exercise 7.

Note: This exercise has 8 steps. Estimated time to complete: 30 minutes.

Summary

You should now be able to:

- Create constraints to improve performance of node creation and queries
- Determine if a query can be improved
- Write Cypher code to refactor a data model
- Create indexes that to improve query performance
- Refactor a graph by creating intermediate nodes
- Refactor a graph by specializing relationships
- Perform batch processing when refactoring a large graph



Summary

Implementing Graph Data Models in Neo4j 4.0

Course Summary

In this course, you have learned how to:

- Implement a simple graph data model
- Import data into an existing graph data model
- Profile query performance against the implemented graph
- Refactor graph data models