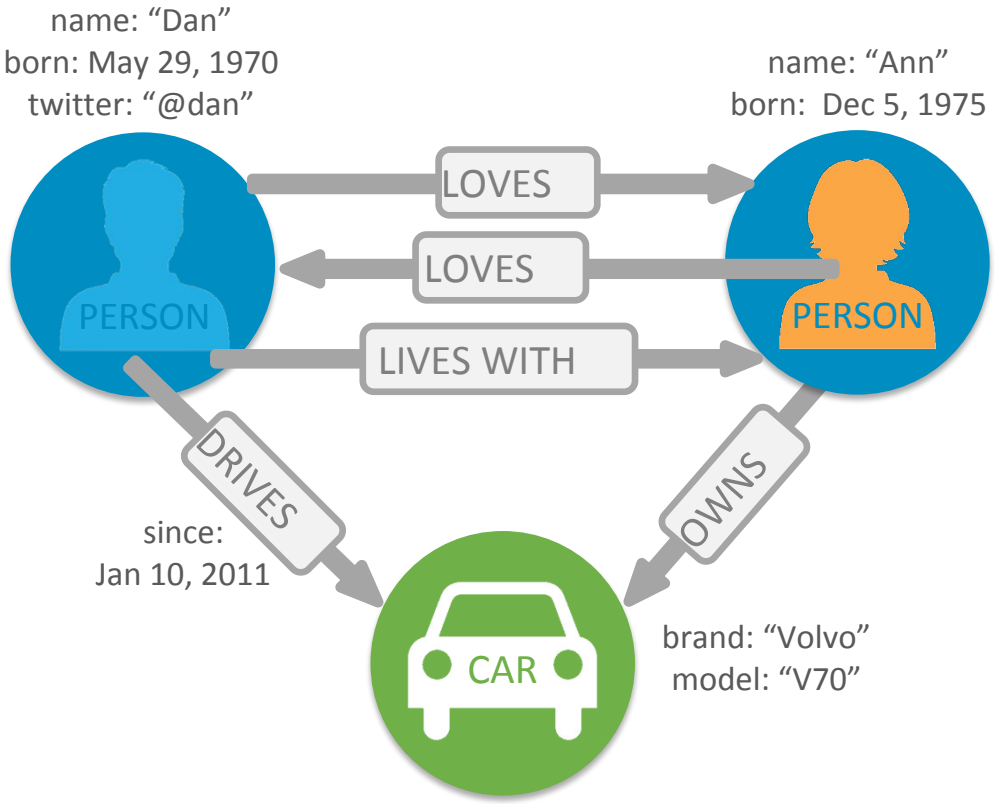
Intro to graph modeling

What are we going to do?

- Intro to the property graph model
- The airport dataset
- The modeling workflow
- Load CSV data

The labelled property graph

- Nodes
- Relationships
- Properties
- Labels



Property Graph Model Components

Nodes

- Represent the objects in the graph
- Can be *labeled*

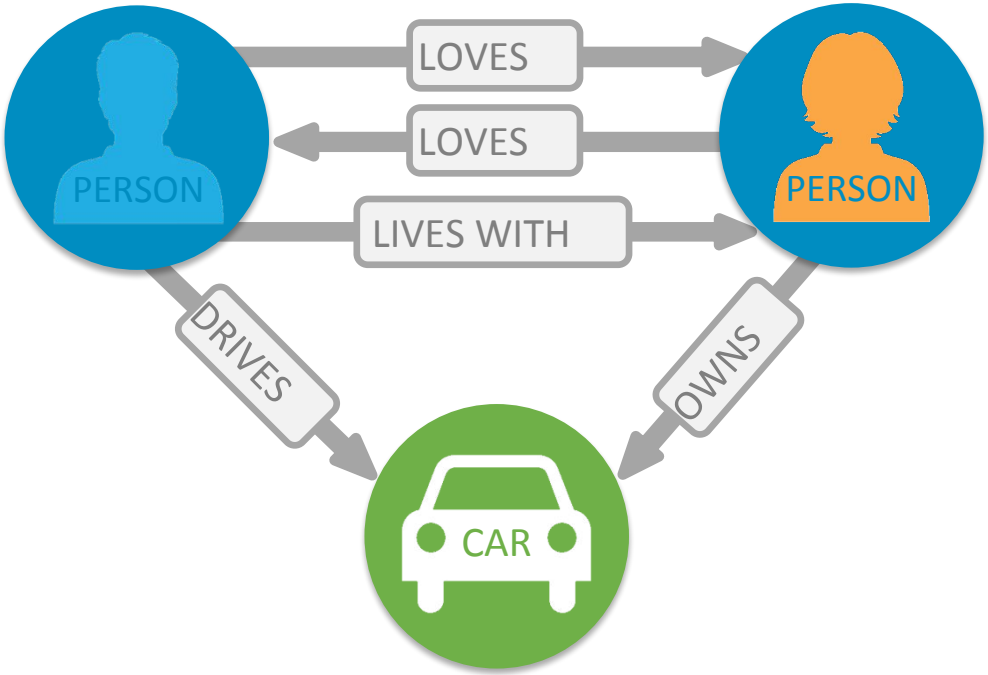
Property Graph Model Components

Nodes

- Represent the objects in the graph
- Can be *labeled*

Relationships

- Relate nodes by *type* and *direction*



Property Graph Model Components

Nodes

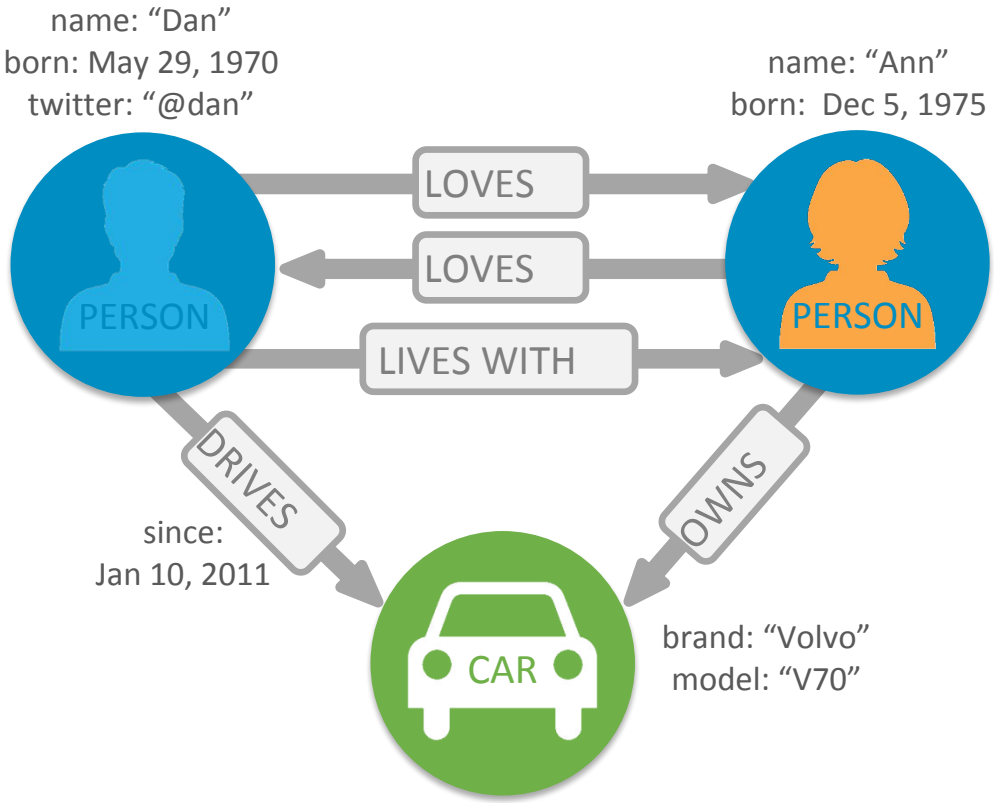
- Represent the objects in the graph
- Can be *labeled*

Relationships

- Relate nodes by *type* and *direction*

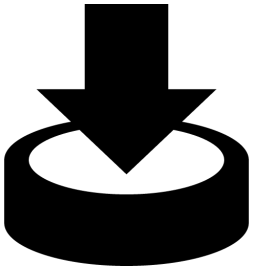
Properties

- Name-value pairs that can go on nodes and relationships.

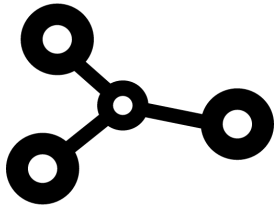


The modeling workflow

1. Derive the question



2. Obtain the data



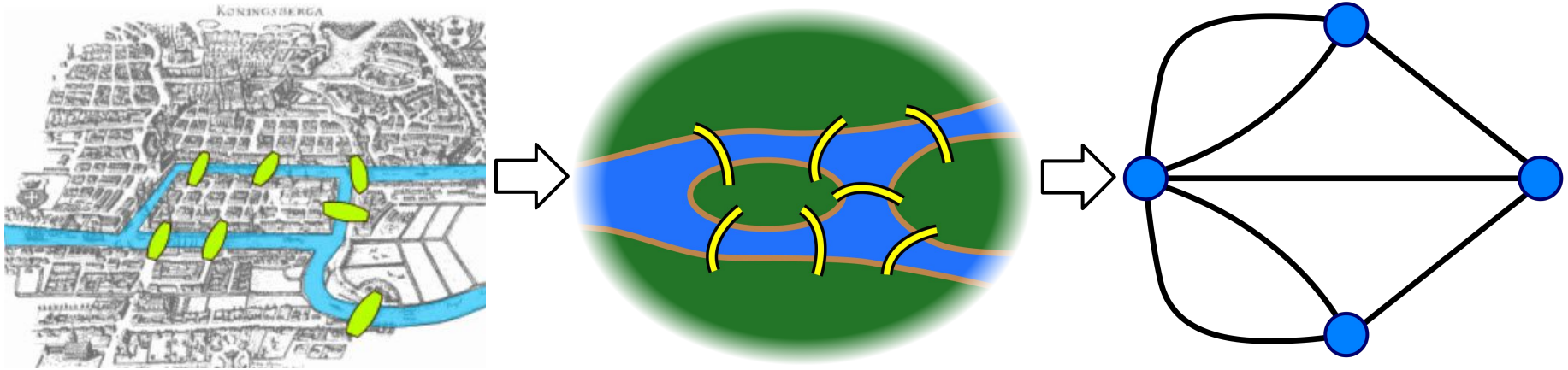
3. Develop a model

4. Ingest the data

5. Query/Prove our model

The Modeling Workflow

Models



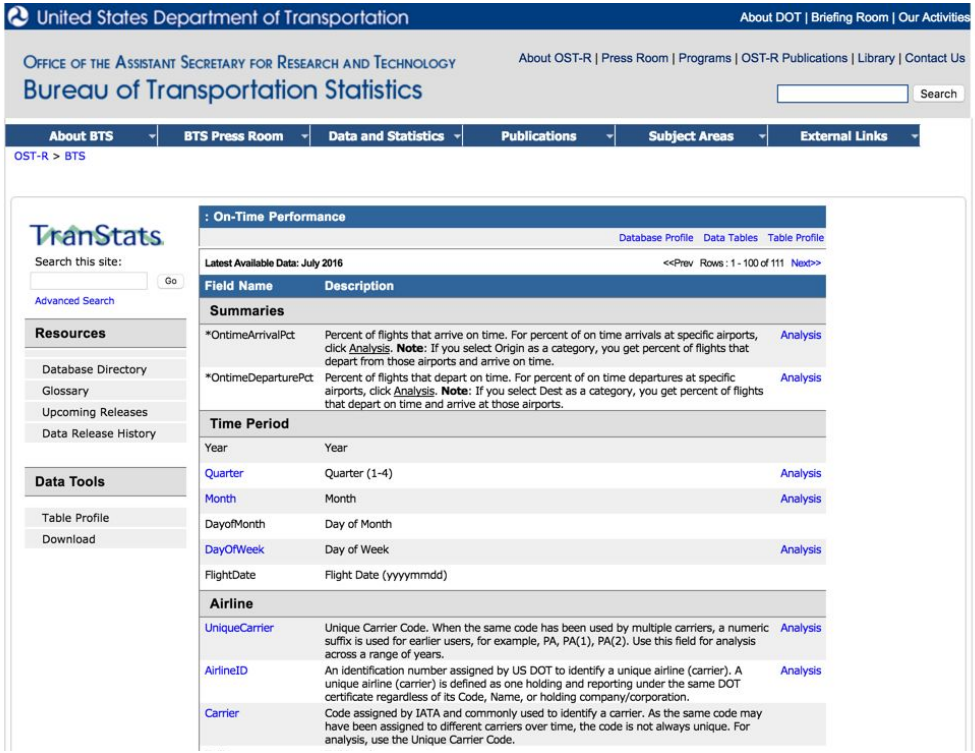
Find the most popular airports

As an air travel enthusiast

I want to know how airports are connected

So that I can find the busiest ones

The flights dataset



United States Department of Transportation

About DOT | Briefing Room | Our Activities

Office of the Assistant Secretary for Research and Technology

Bureau of Transportation Statistics

About OST-R | Press Room | Programs | OST-R Publications | Library | Contact Us

Search

About BTS | BTS Press Room | Data and Statistics | Publications | Subject Areas | External Links

OST-R > BTS

TranStats

Search this site:

Go

Advanced Search

Resources

- Database Directory
- Glossary
- Upcoming Releases
- Data Release History

Data Tools

- Table Profile
- Download

: On-Time Performance

Database Profile | Data Tables | Table Profile

Latest Available Data: July 2016 <<Prev Rows : 1 - 100 of 111 Next>>

Field Name	Description	
Summaries		
*OntimeArrivalPct	Percent of flights that arrive on time. For percent of on time arrivals at specific airports, click Analysis. **Note**: If you select Origin as a category, you get percent of flights that depart from those airports and arrive on time.	Analysis
*OntimeDeparturePct	Percent of flights that depart on time. For percent of on time departures at specific airports, click Analysis. **Note**: If you select Dest as a category, you get percent of flights that depart on time and arrive at those airports.	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)	
Airline		
UniqueCarrier	Unique Carrier Code. When the same code has been used by multiple carriers, a numeric suffix is used for earlier users, for example, PA, PA(1), PA(2). Use this field for analysis across a range of years.	Analysis
AirlineID	An identification number assigned by US DOT to identify a unique airline (carrier). A unique airline (carrier) is defined as one holding and reporting under the same DOT certificate regardless of its Code, Name, or holding company/corporation.	Analysis
Carrier	Code assigned by IATA and commonly used to identify a carrier. As the same code may have been assigned to different carriers over time, the code is not always unique. For analysis, use the Unique Carrier Code.	

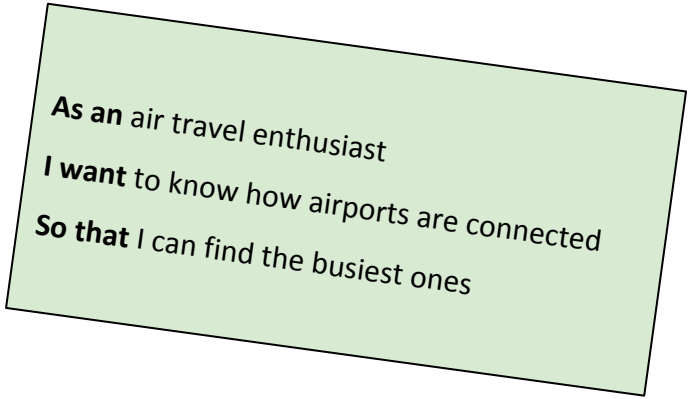
What data do we have?

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)	

Derive questions



Is Airport A connected to Airport B?

Identity entities

Is **Airport** A connected to **Airport** B?

```
airport
```

Identify relationships between entities

Is Airport A **connected to** Airport B?

```
airport CONNECTED_TO airport
```

It’s a graph!

Is Airport A **connected to** Airport B?

```
(airport)-[:CONNECTED_TO]->(airport)
```

It’s a graph!

Is Airport A **connected to** Airport B?

```
(airport)-[:CONNECTED_TO]->(airport)
```

Labels

Labels are for categorizing nodes.

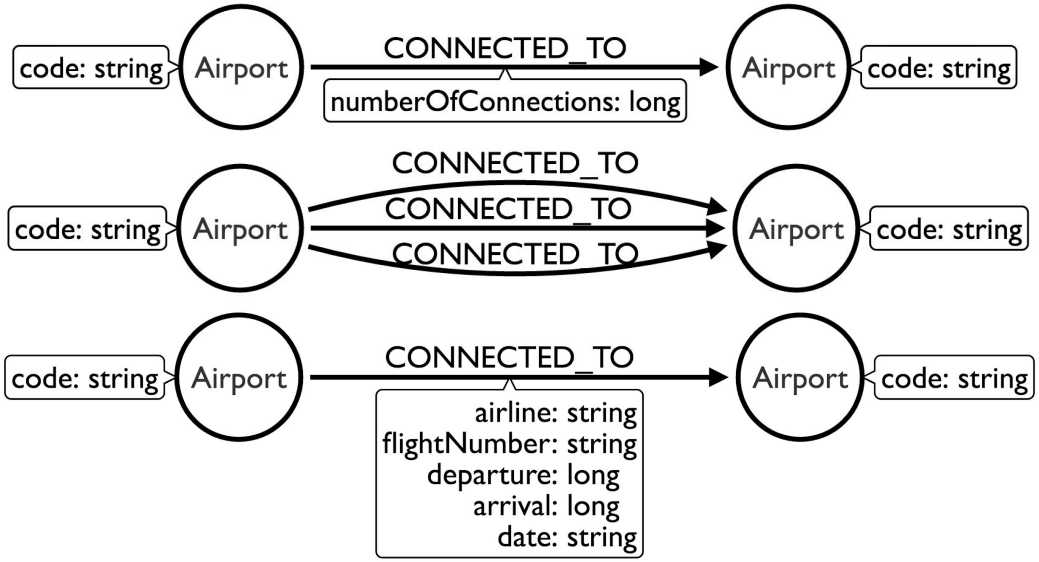
Properties

Properties are for defining attributes of nodes or relationships.

We need an airport code to uniquely identify each airport.

Modeling different connections

Busy airports will have multiple connections between them. We can model this in different ways:

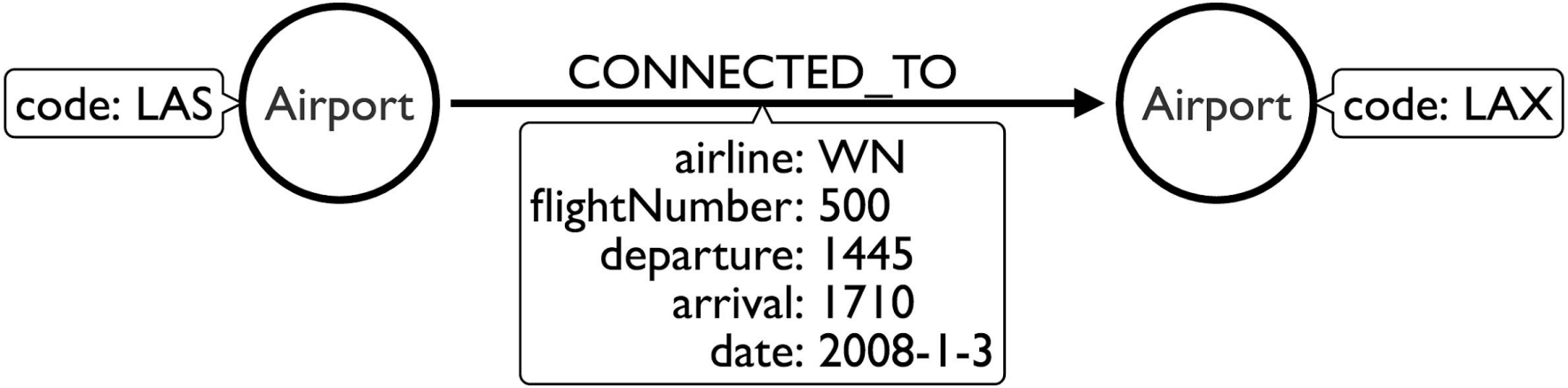


As an air travel enthusiast

I want to know how airports are connected

So that I can find the busiest ones

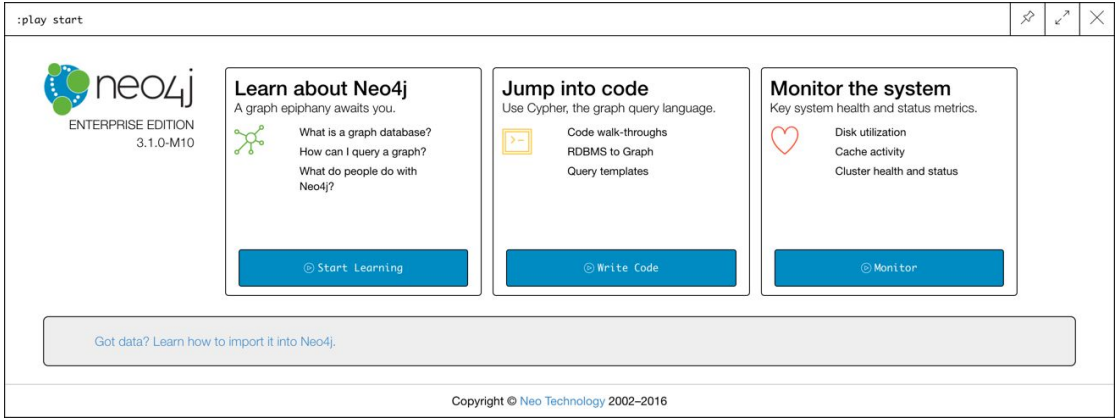
Sample data


code: LAS
Airport
CONNECTED_TO
airline: WN
flightNumber: 500
departure: 1445
arrival: 1710
date: 2008-1-3
Airport
code: LAX

Make sure you've got Neo4j running

1. Start the server.
2. It should be running on: http://localhost:7474
3. Log-in with default credentials
   - user: **neo4j** password: **neo4j**
4. Choose a **new** password

We're good to go!



Find the most popular airports

Open your browser to http://localhost:7474 and execute the following command:

```
:play https://guides.neo4j.com/modeling_airports/
```

You may also use this link which will load the CSV files on the fly:

```
:play https://guides.neo4j.com/modeling_sandbox/
```

Play the guides in your browser until you see...

The MERGE command

MERGE

The MERGE command is a combination of `MATCH` and `CREATE`. If the pattern already exists then it will return it; if not it will create it.

We can MERGE nodes...

```
MERGE (las:Airport {code: "LAS"})
RETURN las
```

...or relationships...

```
MATCH (las:Airport {code: "LAS"})
MATCH (lax:Airport {code: "LAX"})
MERGE (las)-[:CONNECTED_TO]->(lax)
```

...or both

```
MERGE (las:Airport {code: "LAS"})
MERGE (lax:Airport {code: "LAX"})
MERGE (las)-[:CONNECTED_TO]->(lax)

MERGE (las:Airport {code: "LAS"})-[:CONNECTED_TO]->(lax:Airport
{code: "LAX"})
```

Continue playing the guide

Continue the guide in your browser

The modeling workflow

The modeling workflow

As an air travel enthusiast

I want to know how airports are connected

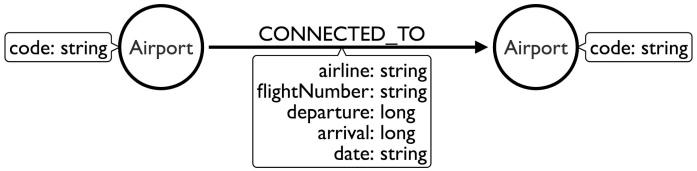
So that I can find the busiest ones

```
MATCH (origin:Airport {code: "LAX"})
      -[flight:CONNECTED_TO]->
      (destination:Airport {code: "LAS"})
RETURN origin, destination, flight
LIMIT 10
```

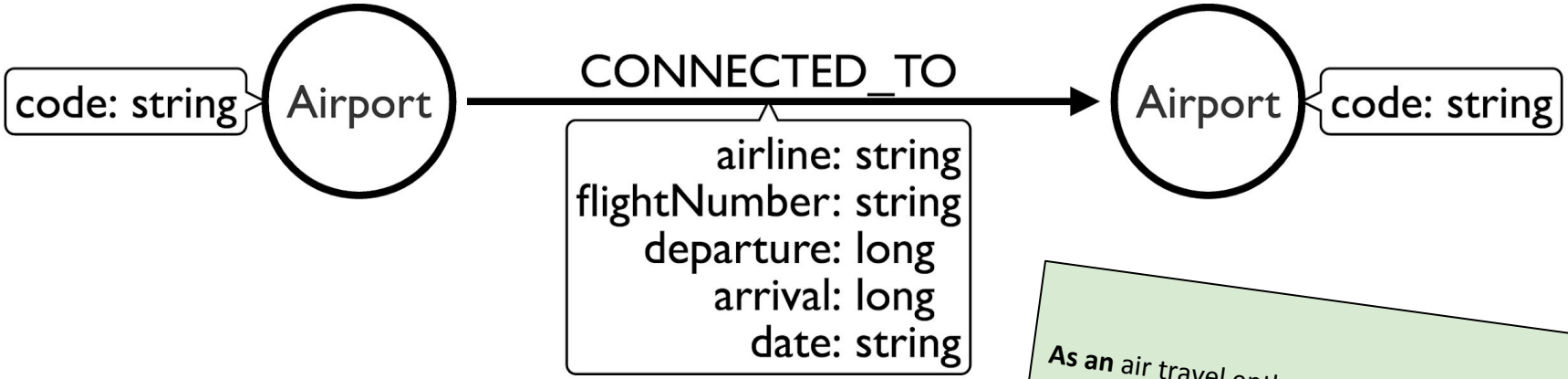
```
CREATE (lax:Airport {code: "LAX"})
CREATE (las:Airport {code: "LAS"})
CREATE (las)-[connection:CONNECTED_TO {
    airline: "WN",
    flightNumber: "82",
    date: "2008-1-3",
    departure: 1715,
    arrival: 1820}]->(lax)
```

Is **Airport** A connected to **Airport** B?

Origin	Dest	FlightNum
IAD	TPA	335
IAD	TPA	3231
IND	BWI	448
IND	BWI	3920



Modeling airports

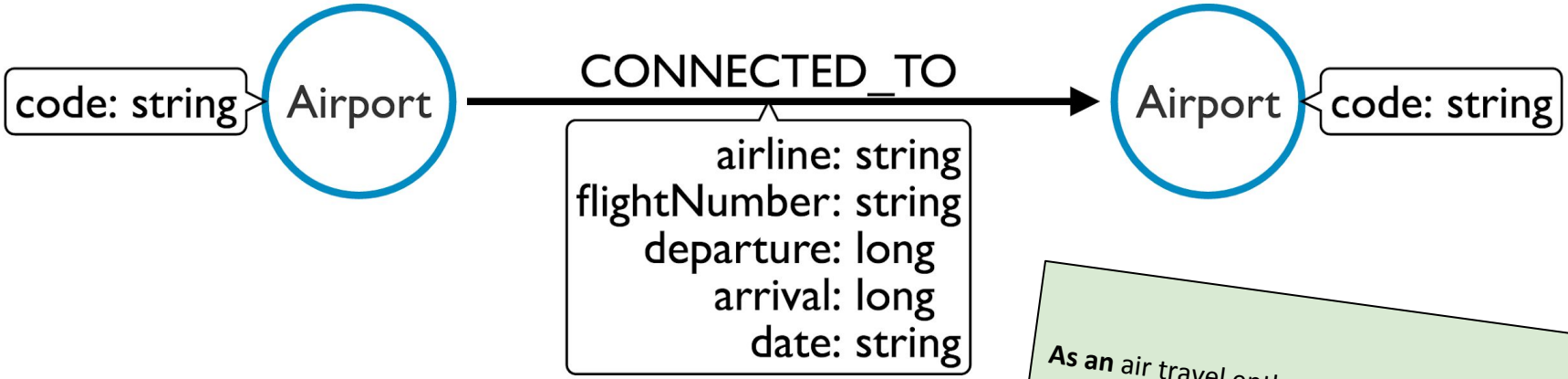


As an air travel enthusiast

I want to know how airports are connected

So that I can find the busiest ones

Nodes

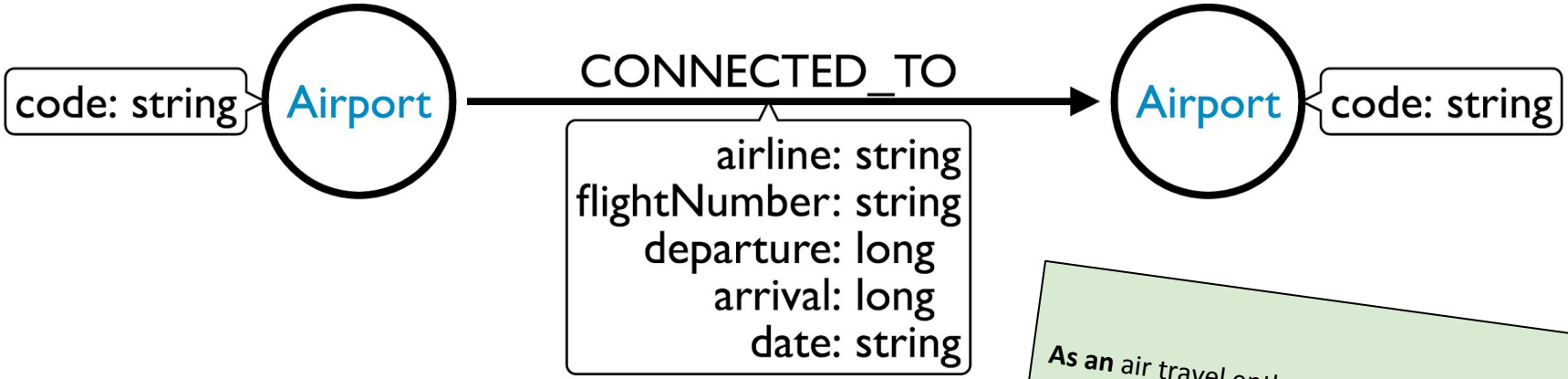


As an air travel enthusiast

I want to know how airports are connected

So that I can find the busiest ones

Labels

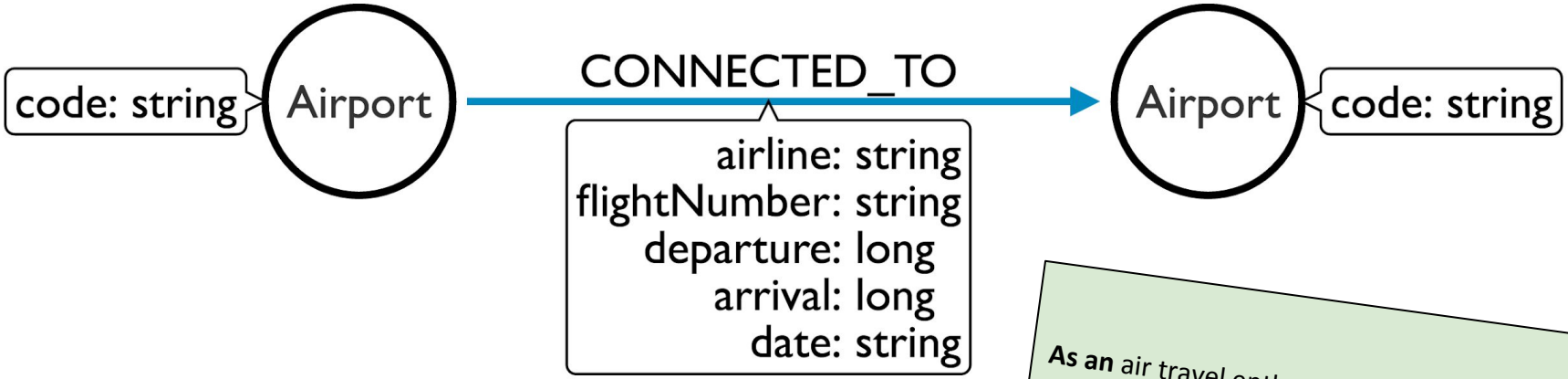


As an air travel enthusiast

I want to know how airports are connected

So that I can find the busiest ones

Relationships

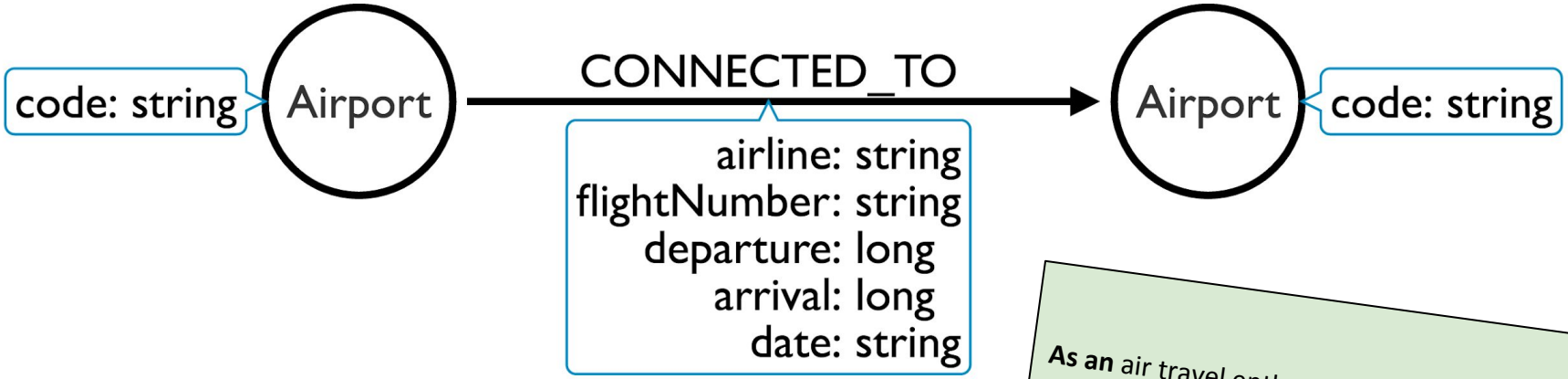


As an air travel enthusiast

I want to know how airports are connected

So that I can find the busiest ones

Properties



As an air travel enthusiast

I want to know how airports are connected

So that I can find the busiest ones

Loading CSV data

Load CSV

A clause in the Cypher query language that lets us iterate over CSV files and create graph structures based on the data contained in each row.

Load CSV

```
LOAD CSV      // load csv data
WITH HEADERS // optionally use first header row as keys in "row" map
FROM "url"    // file:// URL 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 ...
```

Load CSV

```
[USING PERIODIC COMMIT] // optionally batch transactions
LOAD CSV      // load csv data
WITH HEADERS // optionally use first header row as keys in "row" map
FROM "url"    // file:// URL relative to $NEO4J_HOME/import or http://
AS row        // return each row of the CSV as list of strings or map
[FIELDTERMINATOR ";"] // optionally alt. delimiter
// ... rest of the Cypher statement ...
```

Consider this CSV file

Origin	Dest	FlightNum
IAD	TPA	335
IAD	TPA	3231
IND	BWI	448
IND	BWI	3920

Create nodes

```
LOAD CSV WITH HEADERS FROM "file:///flights.csv" AS row

CREATE (:Airport {code: row.Origin})

CREATE (:Airport {code: row.Dest})
```

Origin	Dest	FlightNum
IAD	TPA	335
IAD	TPA	3231
IND	BWI	448
IND	BWI	3920

Create relationships

```
LOAD CSV WITH HEADERS FROM "file:///flights.csv" AS row

CREATE (origin:Airport {code: row.Origin})

CREATE (destination:Airport {code: row.Dest})

CREATE (origin)-[:CONNECTED_TO {flightNumber: row.FlightNum}]->(destination)
```

Origin	Dest	FlightNum
IAD	TPA	335
IAD	TPA	3231
IND	BWI	448
IND	BWI	3920

Find existing nodes and relationships

```
LOAD CSV WITH HEADERS FROM "file:///flights.csv" AS row

MATCH (origin:Airport {code: row.Origin})

MATCH (destination:Airport {code: row.Dest})

MATCH (origin)-[:CONNECTED_TO {flightNumber: row.FlightNum}]->(destination)

...
```

Origin	Dest	FlightNum
IAD	TPA	335
IAD	TPA	3231
IND	BWI	448
IND	BWI	3920

Update existing nodes and relationships

```
LOAD CSV WITH HEADERS FROM "file:///flights.csv" AS row

MATCH (origin:Airport {code: row.Origin})

MATCH (destination:Airport {code: row.Dest})

MATCH (origin)-[c:CONNECTED_TO {flightNumber:row.FlightNum}]->(destination)

SET c.airline = row.UniqueCarrier
```

Origin	Dest	FlightNum	UniqueCarrier
IAD	TPA	335	WN
IAD	TPA	3231	WN
IND	BWI	448	WN
IND	BWI	3920	WN

Idempotently create nodes and relationships

```
LOAD CSV WITH HEADERS FROM "file:///flights.csv" AS row

MERGE (origin:Airport {code: row.Origin})

MERGE (destination:Airport {code: row.Dest})

MERGE (origin)-[:CONNECTED_TO {flightNumber:row.FlightNum}]->(destination)
```

Origin	Dest	FlightNum
IAD	TPA	335
IAD	TPA	3231
IND	BWI	448
IND	BWI	3920

Let’s give it a try

Continue the guide in your browser

End of Module
Intro to Graph Modeling

Questions ?

Prepare your Neo4j graph.db directory

Copy folders

- `import`
- `plugins`

from **USB Stick**

to the **`default.graphdb`** folder

(or **`$NEO4J_HOME`**)

http://bit.ly/gov-graph