An Overview of Neo4j 

By Red Team (Colin & Kyle)

What we did

Kyle: General overview, similarities/differences with Sql, differences with MongoDB, large scale model, and writing the paper.

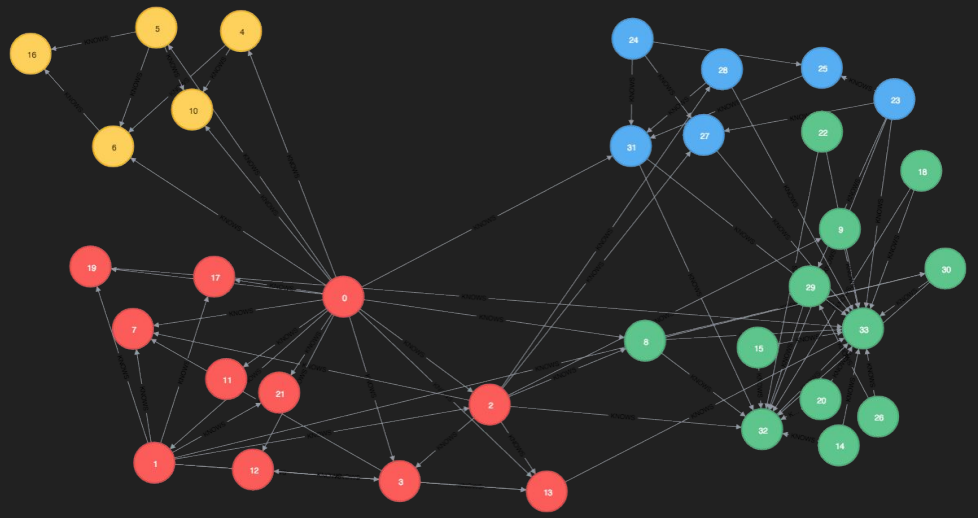
Colin: Ne04j features, advantages/disadvantages of Neo4j, similarities with MongoDB, large scale model, libraries/tools, and writing the paper.

Why we did it

- We thought it would be easier than a project
- Schema-free databases
- To learn more about Neo4j and graphical databases (which looked cool)

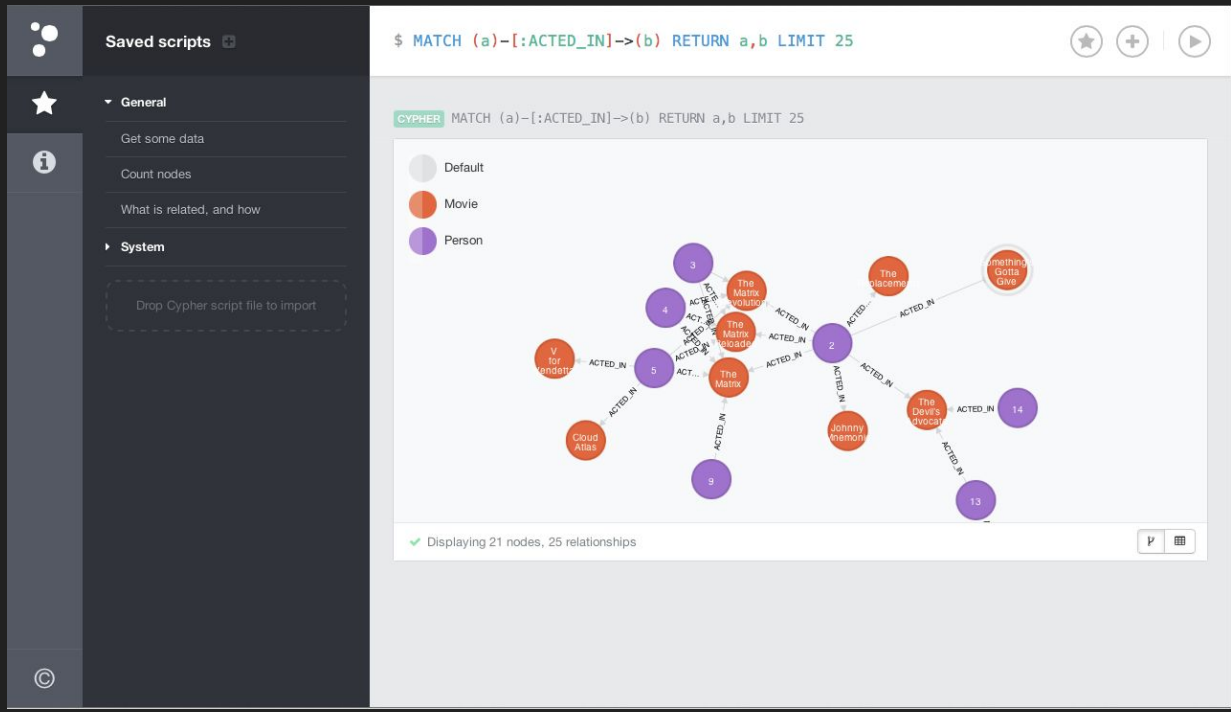
General Facts about Neo4j

- Developed by Neo4j Inc.
- J stands for Java (implementation language)
- Most popular graph database technology, 21st most popular overall
- Used by many large companies, particularly those involved in social media



Neo4j's U.I.

- Command driven client
- Frame code view
- Other features
- Neo4j is a combinations of RELP, a lightweight I.D.E. , and graph visualization



The Anatomy of a Cypher Query

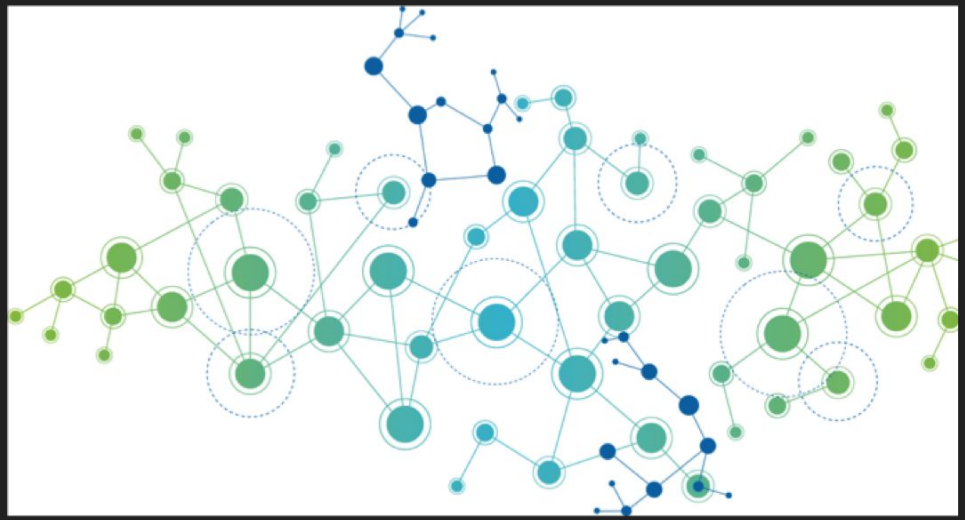
- Match is like select in sql
- nodeName
- Label
- Property
- Relationships
- Return

```
MATCH (nodeName:Label{property: “value”})-[property: “value”] -(nodeName:Label{property: “value”})

Return nodeName.property
```

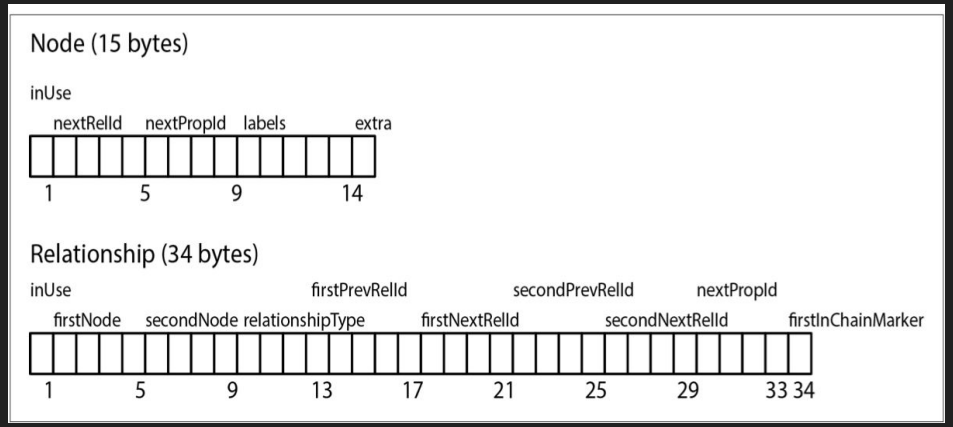
Neo4j Features

- Data can be visualized in graph model or be displayed in a table
- Supports importing and exporting as CSV files
- Regular expressions
- Constraints
- Supports user defined functions and procedures (triggers)
- Schema free (optional)
- Indexes
- Wide array of built in functions



Neo4j Internal Processes

Native Graph Storage



- Disc storage!
- Store Files (different types)
- Fixed size records

Neo4j Internal Processes (cont.)

Native Graph Processing

- Index Free Adjacency
- Nodes are “Hubs”
- Chains of Information
- Very Efficient

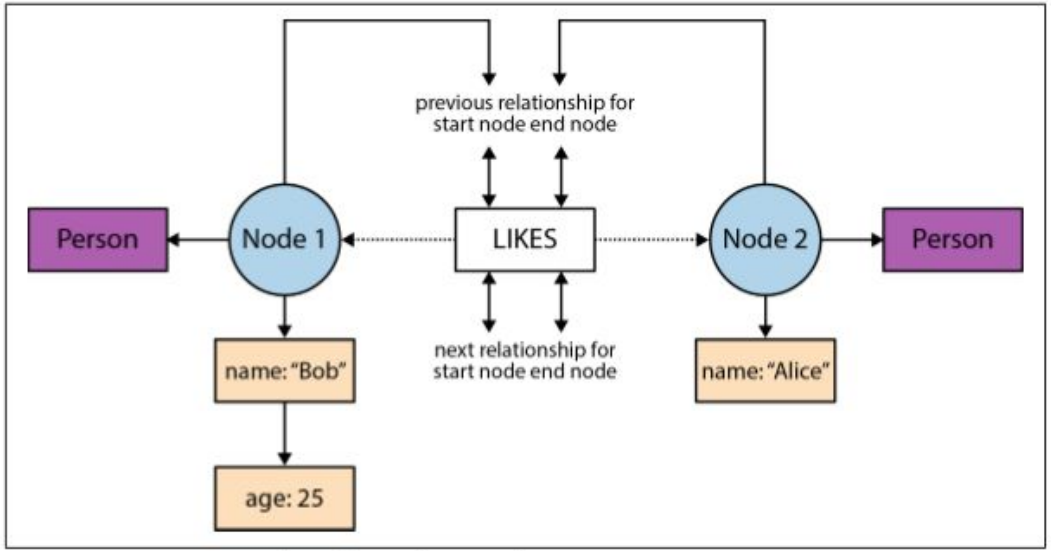


Figure 6-5. How a graph is physically stored in Neo4j

Similarities/Differences with MariaDB

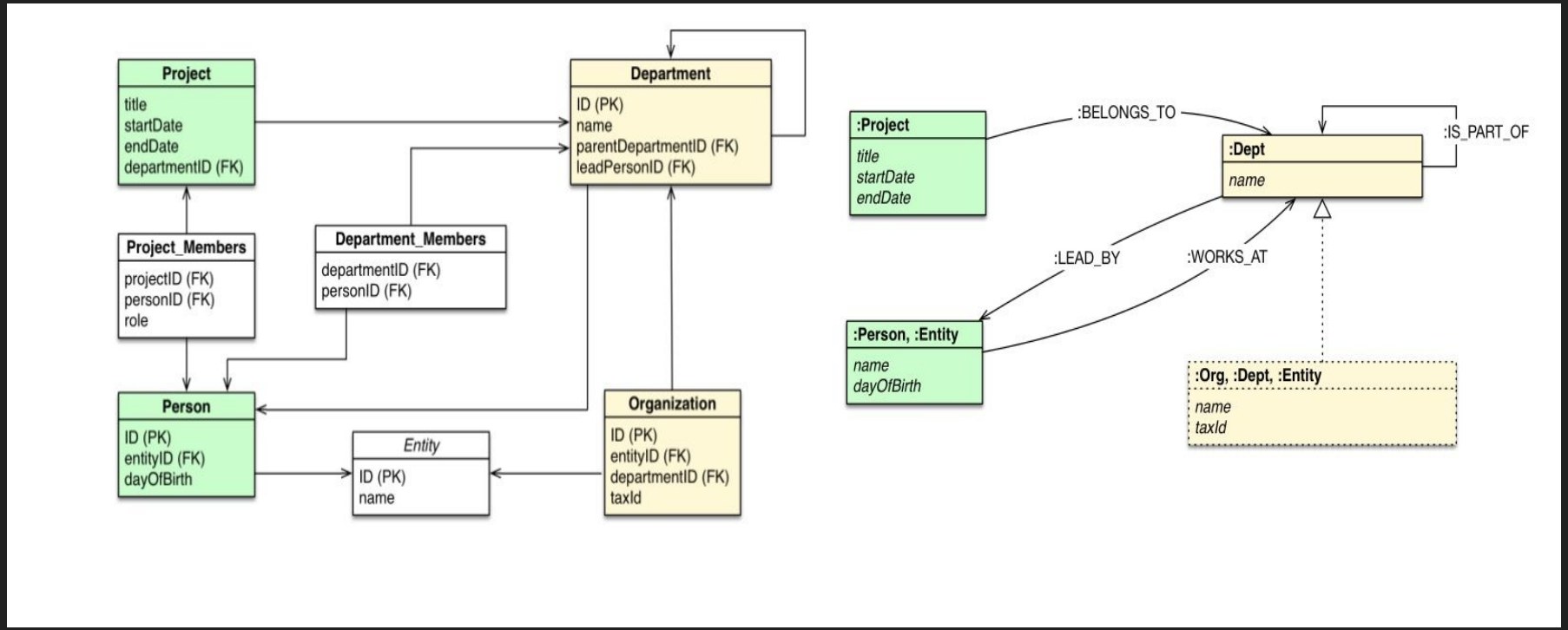
Similarities

- Query structure
- Functions
- Procedures
- Triggers
- Transactions
- Indexes
- Schema

Differences

- Match
- Relationships
- Types
- Joins
- Efficiency
- Sharding
- Query optimizations

Differences Continued



Differences Continued

SQL Statement

```
SELECT name FROM Person
LEFT JOIN Person_Department
  ON Person.Id = Person_Department.PersonId
LEFT JOIN Department
  ON Department.Id = Person_Department.DepartmentId
WHERE Department.name = "IT Department"
```

Cypher Statement

```
MATCH (p:Person)<-[:EMPLOYEE]-(d:Department)
WHERE d.name = "IT Department"
RETURN p.name
```

Comparisons with MongoDB

Similarities

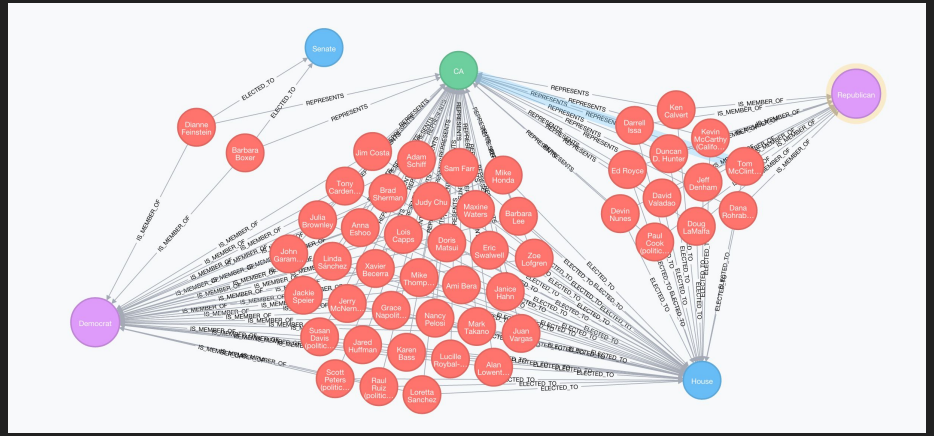
- NoSQL
- Open Source
- Close release Date
- Same operating Systems
- Key Value Pairs
- CSV Imports/Exports
- Eventual and Immediate Consistency
- Most popular of each type of system
- Both are “schema-free” but can have structure added by using constraints

Differences

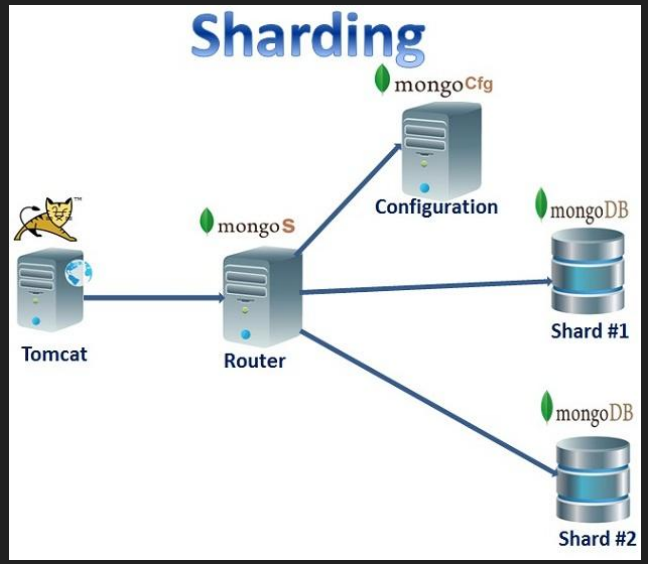
- Data representation
- Sharding
- Mapreduce
- Data replication
- Transactions
- Foreign Keys
- Triggers

Advantages

- Schema-Free [Optional] (very flexible)
- ACID compliant (very reliable)
- Excellent at retrieving interrelated data
- Great for visualizing data



Disadvantages



Does not support sharding (relies on vertical scaling)

Probably less support available from an industry perspective

Might not be the best fit for business transaction applications

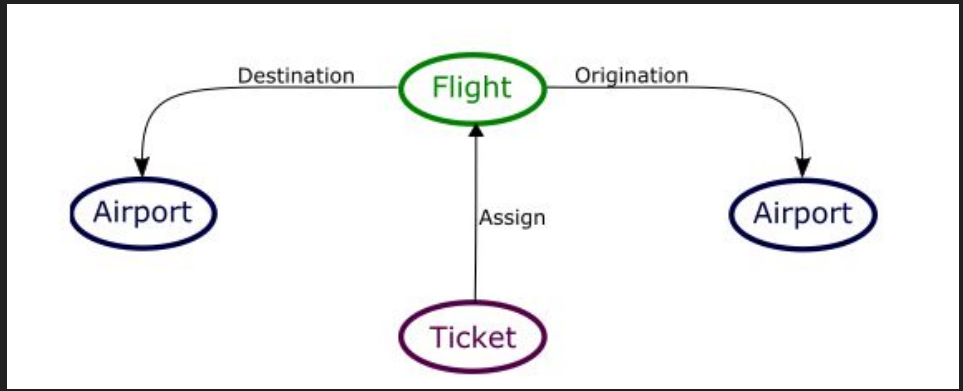
Less efficient at performing data analytics (Aggregation)

No native support for audit trails, versioning

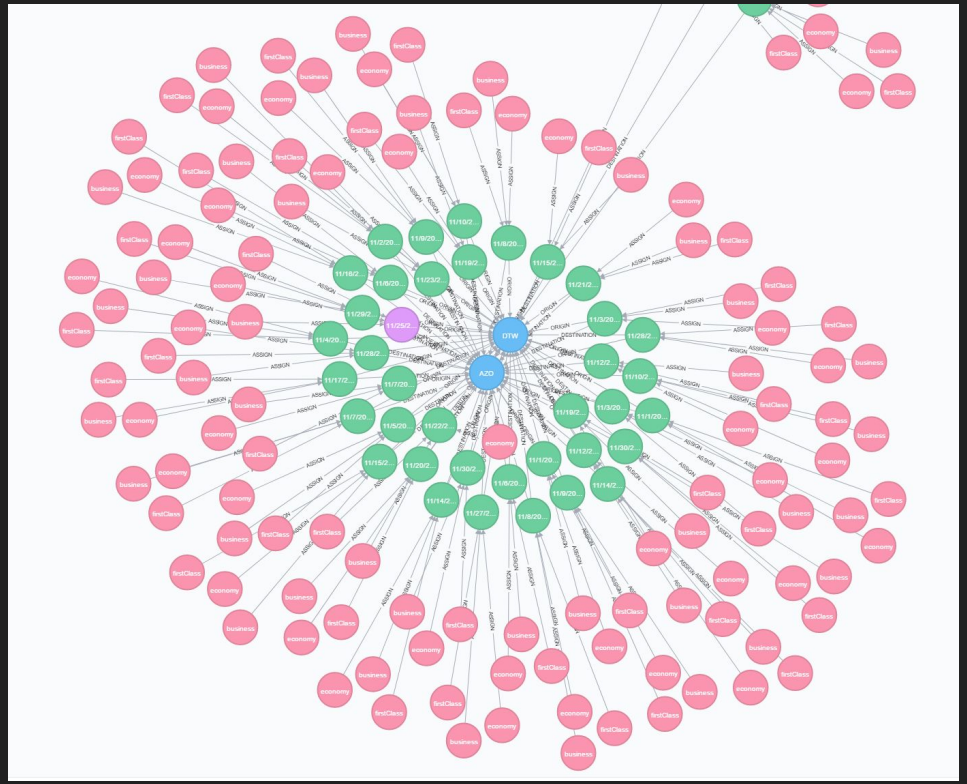
Flight Data Model

Entities: Flights, Airports, Tickets

Relationships: Flights have destination and origination airports, Tickets are assigned to flights



Flight Model (cont.)



- 1,294 Nodes, 1,520 Relationships
- Imported using Cypher (convenient)
- Labels, Properties, Relationships, Nodes

Flight Model Queries

Pattern Comprehension

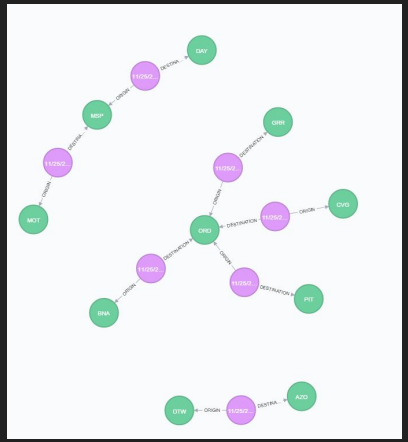
```
MATCH (t:Ticket)
RETURN
[(t)-[:ASSIGN]->(f:Flight) WHERE f.airline CONTAINS '19977'
| f.distance * 1.60934]
as km_distance
```

Assigning Labels

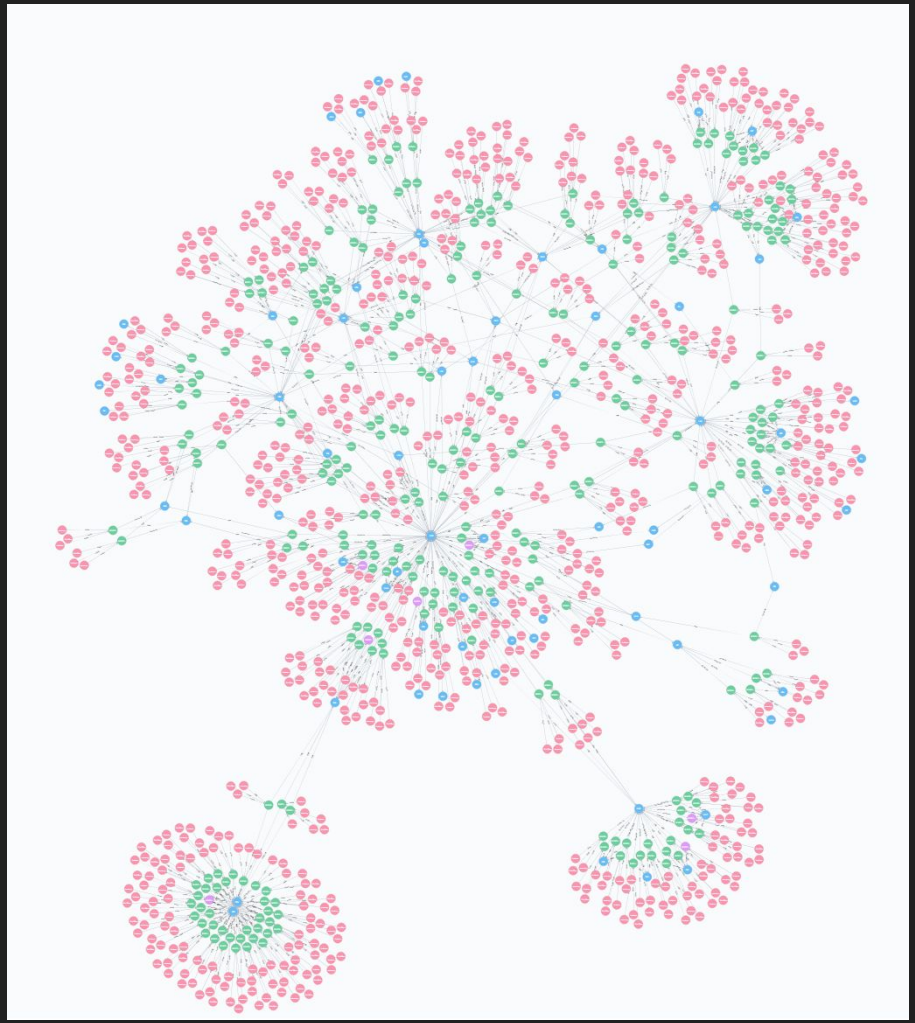
```
MATCH (f:Flight) WHERE f.date =~ '11/30/2015.*'
SET f :Thanksgiving
RETURN f
```

Joins and Aggregation

```
MATCH (f:Flight)-[:ORIGIN]-(a:Airport)
WITH f
MATCH (t:Ticket)-[:ASSIGN]-(f)-[:DESTINATION]-(d:Airport)
RETURN d.name AS destination_airport, avg(t.price) AS
average_flight_distance ORDER BY average_flight_distance DESC;
```



destination_airport	average_flight_distance
"HNL"	4538.083333333334
"SAN"	2323.616666666667
"EWR"	2158.863636363637
"BOS"	1795.0750000000003
"SNA"	1776.8833333333334
"SFO"	1640.2893939393944
"PDX"	1578.25
"IAD"	1528.802380952381
"SEA"	1317.0111111111112
"MIA"	1225.2333333333333
"IAH"	1210.5447916666667



How it worked

- Researching
- Timings / Importing
- Data Model
- Paper