Crash Course in Neo4j

Lju Lazarevic lju@neoj4.com
@ellazal
Erik Nygren erik.nygren@neo4j.com

Agenda

Part 1:

- Graph databases overview and their application

Part 2:

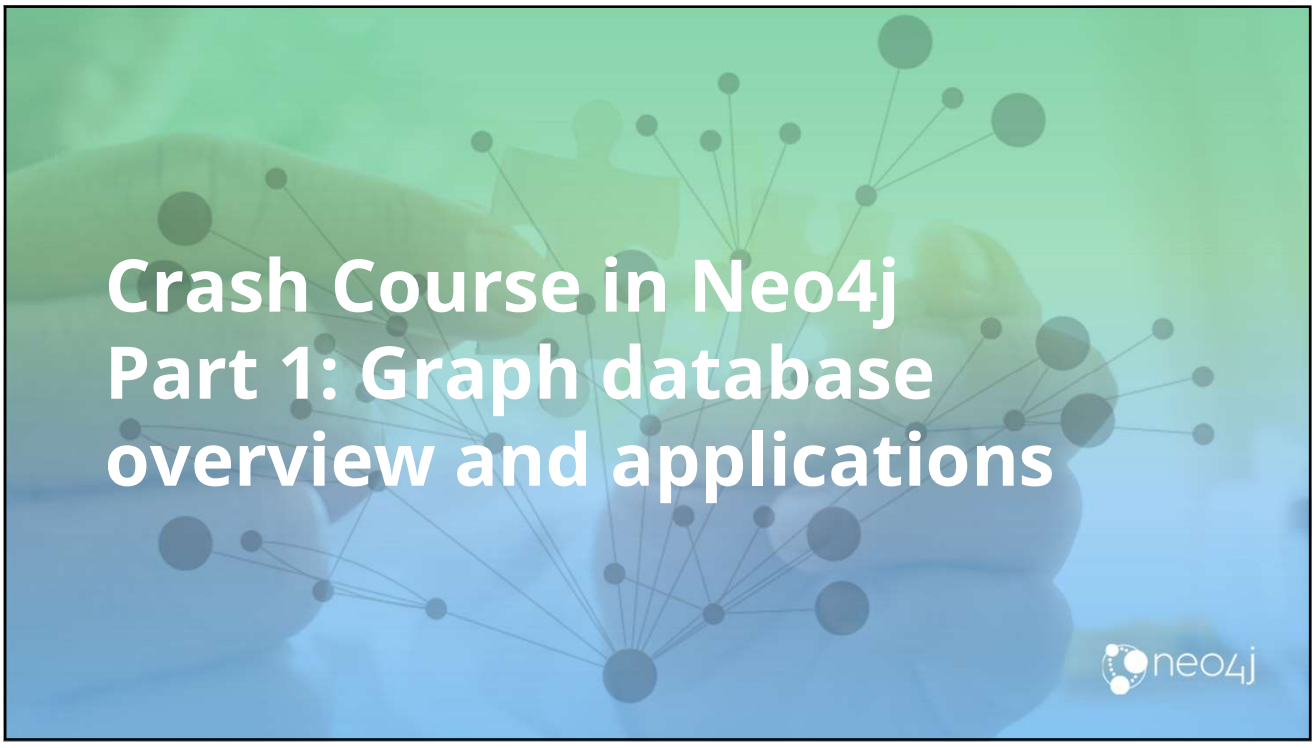
- Neo4j under the hood
- Introduction to graph data modelling
- Introduction to Cypher
- Overview of Neo4j Graph Algorithms

Part 3:

- Hands-on session
- Getting data in and out
- Neo4j and Apache Spark

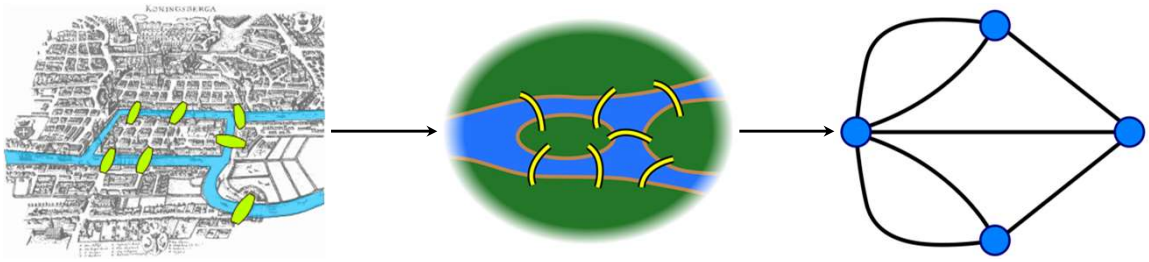
The “Rules”

- Do ask questions!
  - (we might have to park some if we get time-constrained)
- Do have a go!
  - Group questions
  - Hands-on exercises
- Do have a look at the recommended reading!
  - We’re only going to scrape the surface today



What is a graph?

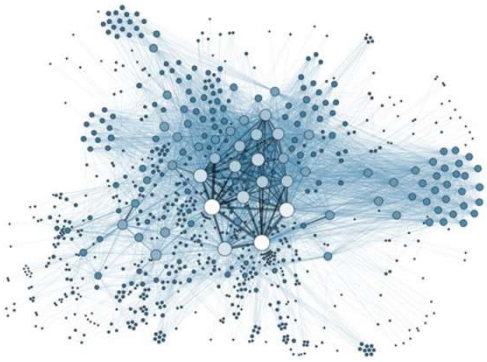
A **graph** is set of discrete objects, each of which has some set of relationships with the other objects



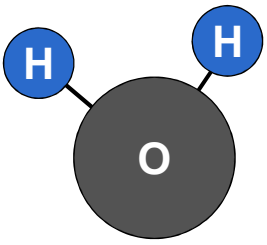
5 *Seven Bridges of Konigsberg problem. Leonhard Euler, 1735*

Anything can be a graph

the Internet



a water molecule

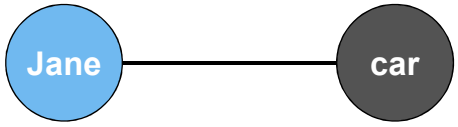


6

Graph components

Node (Vertex)

- The main data element from which graphs are constructed
  A node without relationships is permitted. A relationship without nodes is not.
- A waypoint along a traversal route

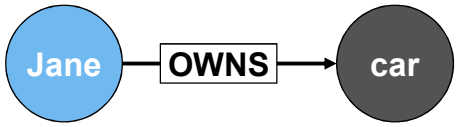


7

Graph components

Node (Vertex)

- The main data element from which graphs are constructed
  A node without relationships is permitted. A relationship without nodes is not.
- A waypoint along a traversal route



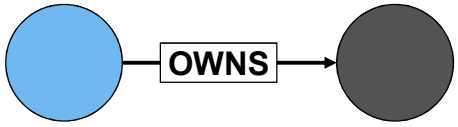
Relationship (Edge)

- A link between two nodes. May contain:
  - Direction
  - Metadata; e.g. *weight* or *relationship type*

8

Label property graph database

Node (Vertex)
Relationship (Edge)

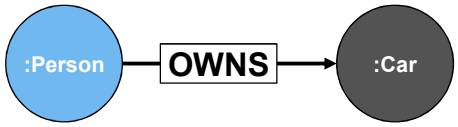


9

Label property graph database

Node (Vertex)
Relationship (Edge)
Label

- Define node category (optional)

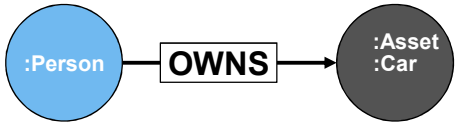


10

Label property graph database

Node (Vertex)
Relationship (Edge)
Label

- Define node category (optional)
- Can have more than one



11

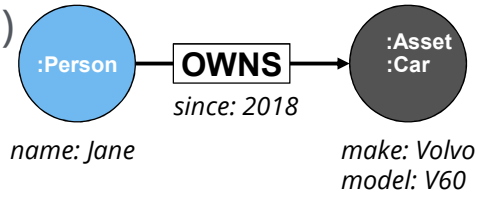
Label property graph database

Node (Vertex)
Relationship (Edge)
Label

- Define node category (optional)
- Can have more than one

Properties

- Enrich a node or relationship
- No need for nulls!



12

Why do we need graphs ?

- Technology works with a **representation of reality**
- Reality is about **things being connected to other things**
- Reality seldom comes in **discrete tables**
- Reality does not stop at the border of a data silo
- A graph is a **better approximation** of reality !

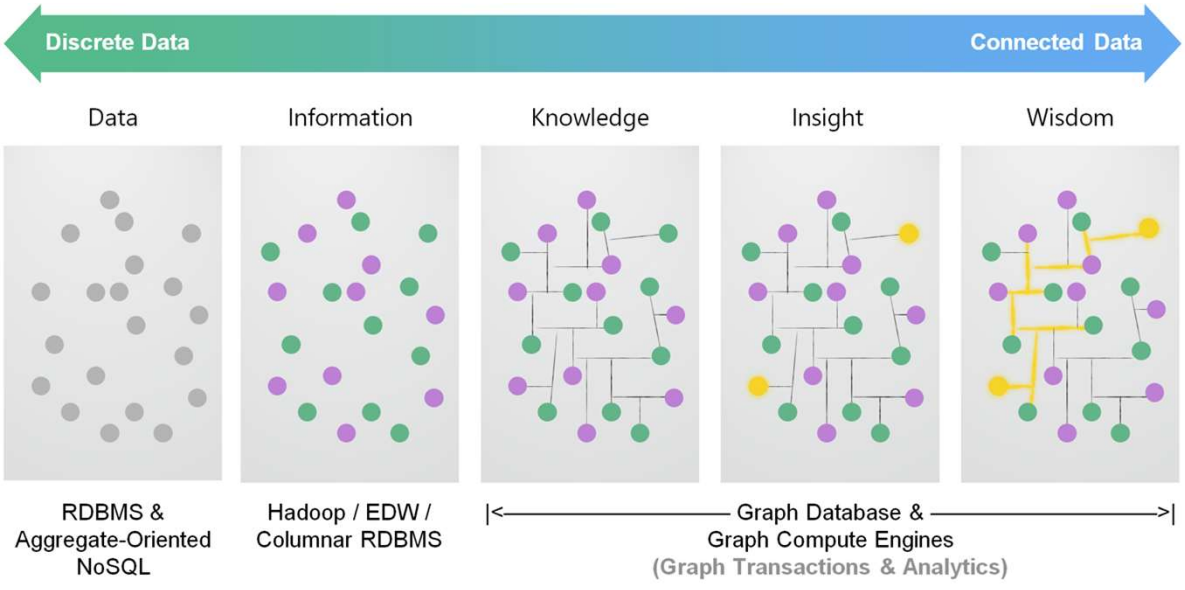


Illustration by David Somerville based on the original by Hugh McLeod (@gapingvoid)

Customer to claims policy information

Table 1

Customer				
CustID	**Customer**	**Address**	**Email**	**Alternate**
1	John Doe	23 Links Lane	john@gmail.com	
2	Joe Bloggs	1 Guildford high St	jbloggs@gmail.com	
3	Susan Smith	4 Oak Street London	ssmith@gmail.com	
4				

Table 1-1

Policy	
PolicyID	**CustID**
1234	1
5678	1
…..	….
5588	99

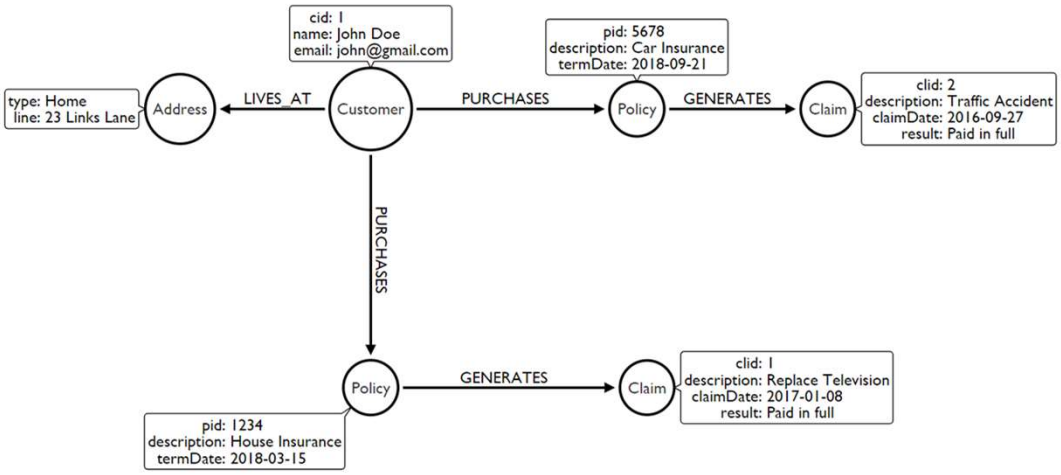
Table 1-3

LineItem			
PolicyID	**Description**	**Term Date**	**ClaimsID**
1234	House Insurance	15th March 2018	1
5678	Car Insurance	21st Sept 2018	2
….			…
5588	765	18th Dec 2018	999

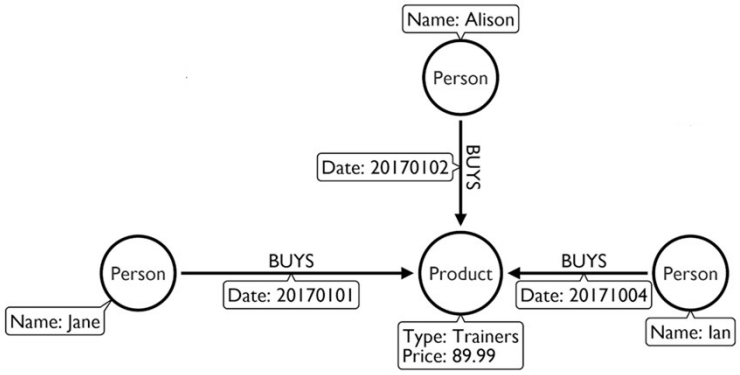
Table 1-2

Claims			
ClaimsID	**Description**	**Date of Claim**	**Result**
1	Replace Television	8th January 2017	Paid in full
2	Traffic Accident	27 Sept 2016	Paid in full
…			
987	Roof Leaking	25th April 2012	Dispute

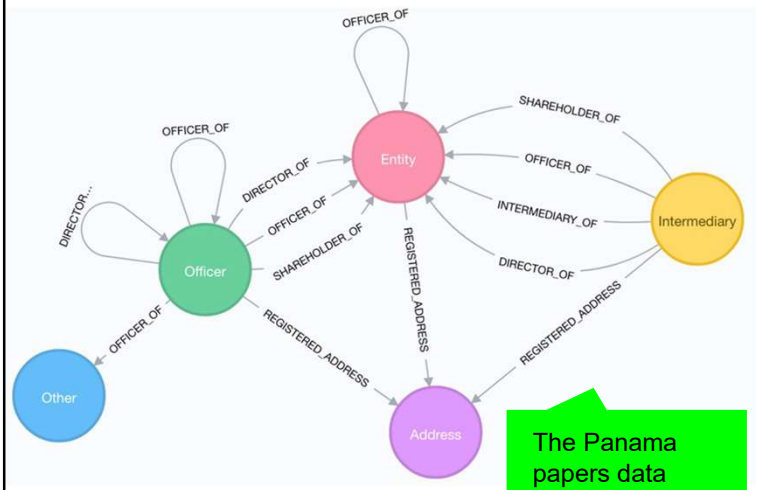
What might this look like in a graph?



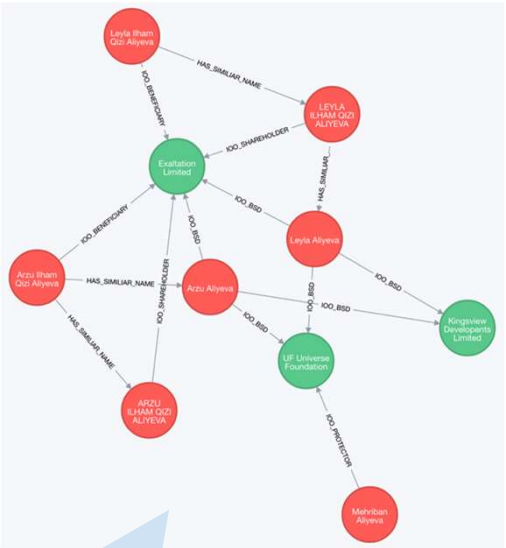
Follow the flow - buying trainers



Finding fraud



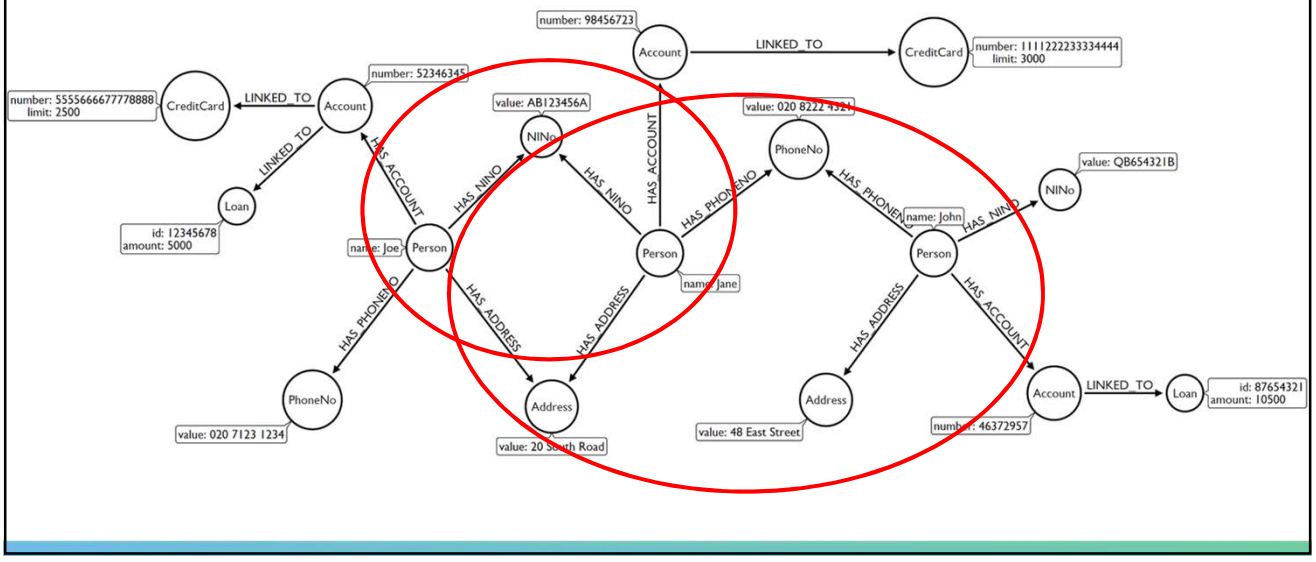
The Panama papers data model...



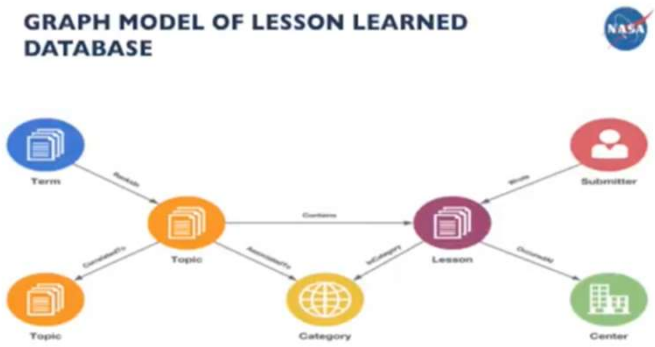
...and dealing with entity resolution

18

Finding fraud rings



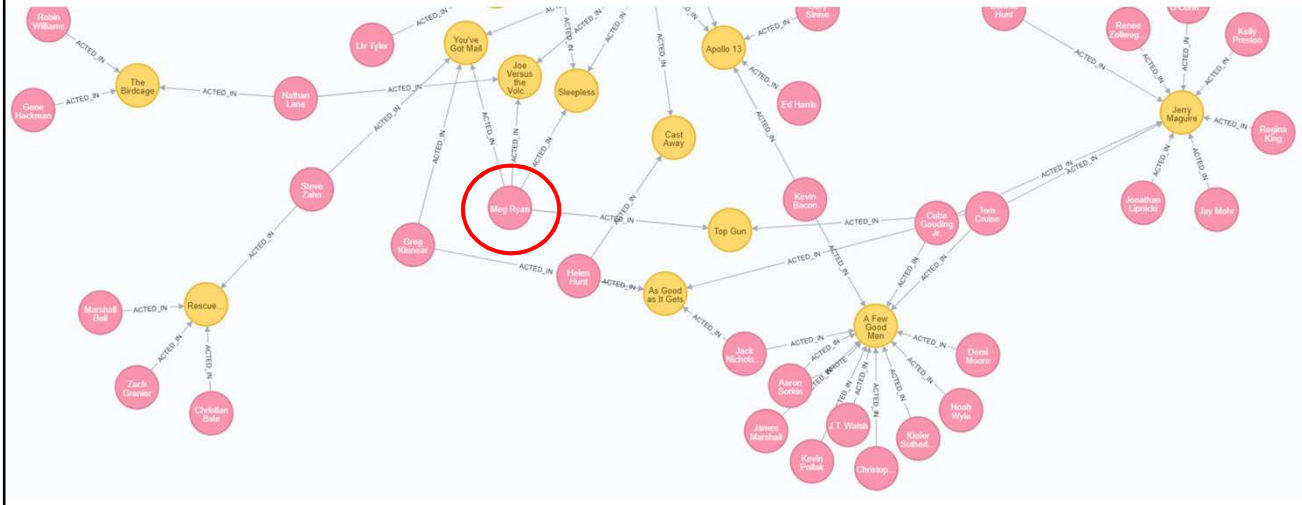
NASA Knowledge graphs



NASA's knowledge management graph model.

"using Neo4j someone from our Orion project found information from the Apollo project that prevented an issue, saving well over two years of work and one million dollars of taxpayer funds"

Friends of friends



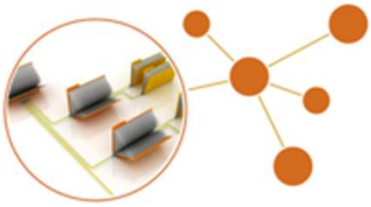
...or co-actors of co-actors

Graphs are everywhere...

Impact analysis

Logistics and routing

Recommendations



Access control

Fraud analysis

Social network

Getting set up

A couple of options:

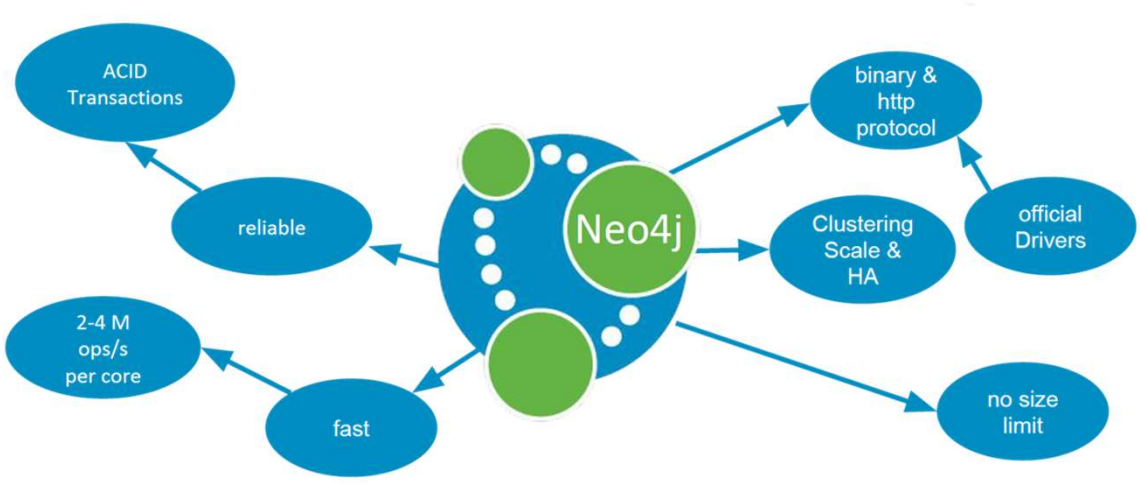
- Neo4j Desktop - preferred
  - neo4j.com/download
- Sandbox - no install option
  - neo4j.com/sandbox-v2
  - We will use the **Blank Sandbox** option
  - Will not have options for APOC or Graph Algorithms

Set-up support during the first break

Crash Course in Neo4j Part 2: Neo4j under the hood, Data Modelling and Cypher

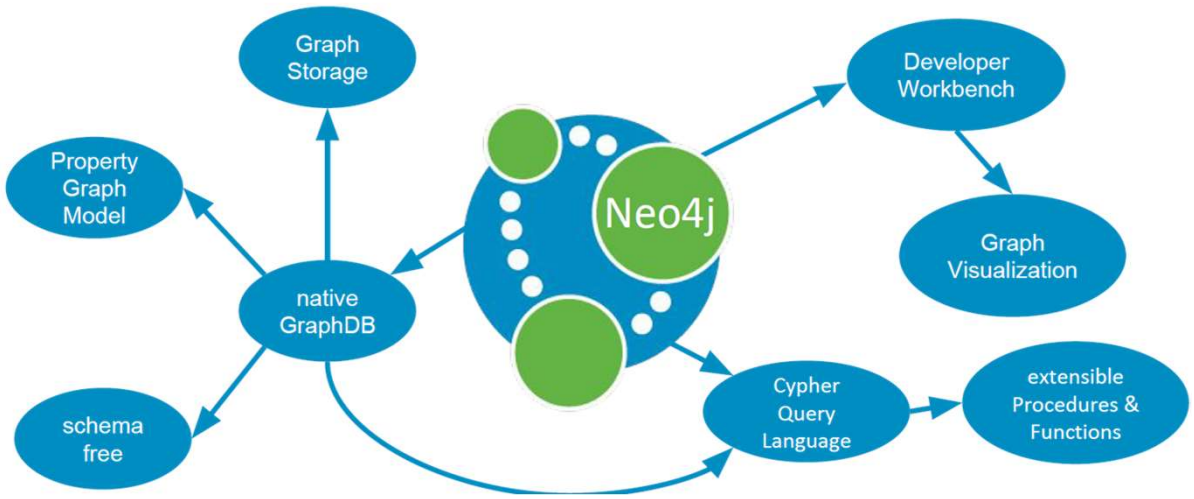
neo4j

Neo4j is a database



26

Neo4j is a graph database



27

Language and driver support

- Cypher to access the database
- Server-side extensions to access the database
- Out-of-the-box drivers via **bolt** protocol:
  - Java
  - JavaScript
  - Python
  - C#
  - Go
- Neo4j community contributions for other languages

28

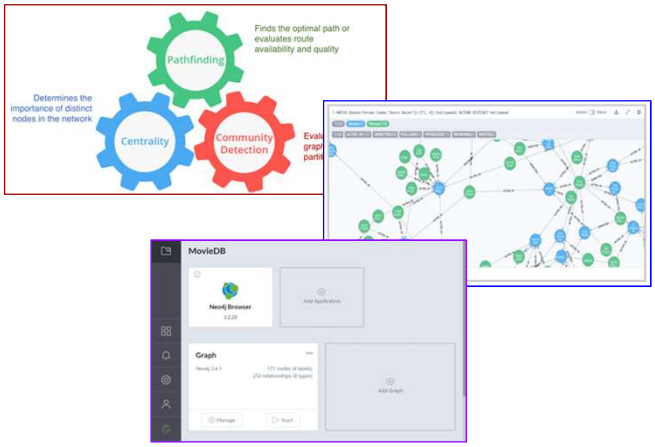
Libraries and tools

Out-of-the-box libraries:

- APOC
- Graph Algorithms
- GraphQL

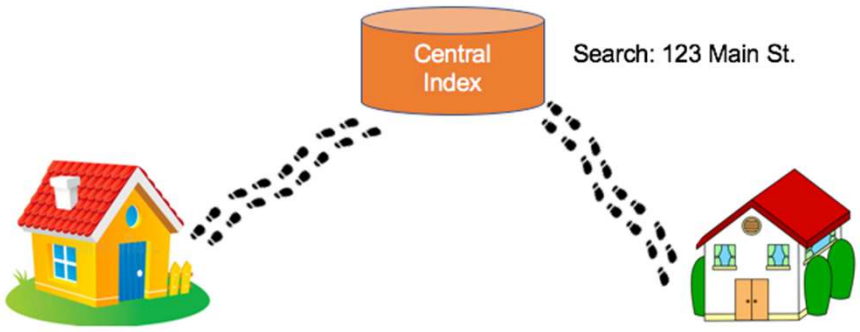
Tools:

- Browser
- Desktop
- Bloom
- ETL



The Neo4j community has also made many contributions!

Index-free adjacency in two minutes



My house

Anne's house

30

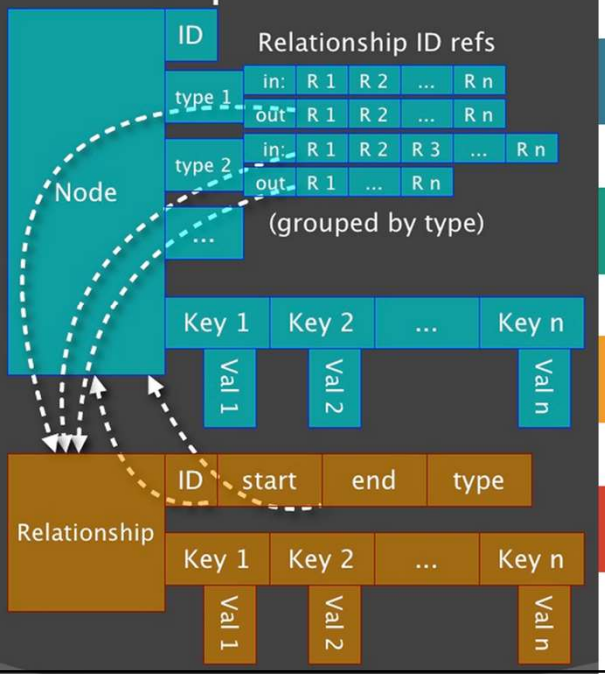
Index-free adjacency in two minutes

My house

Anne's house

31

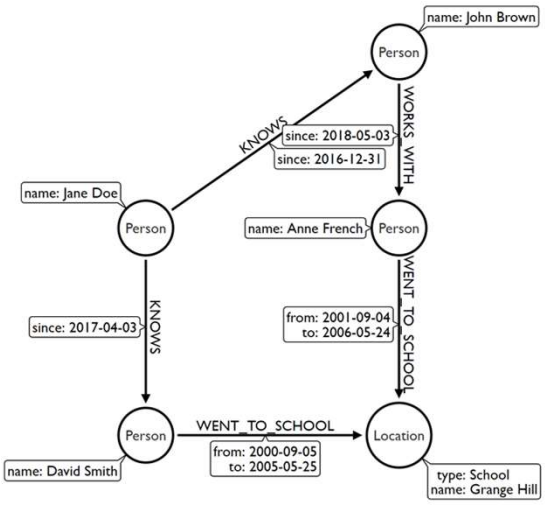
What we put in cache



Neo4j Secret Sauce

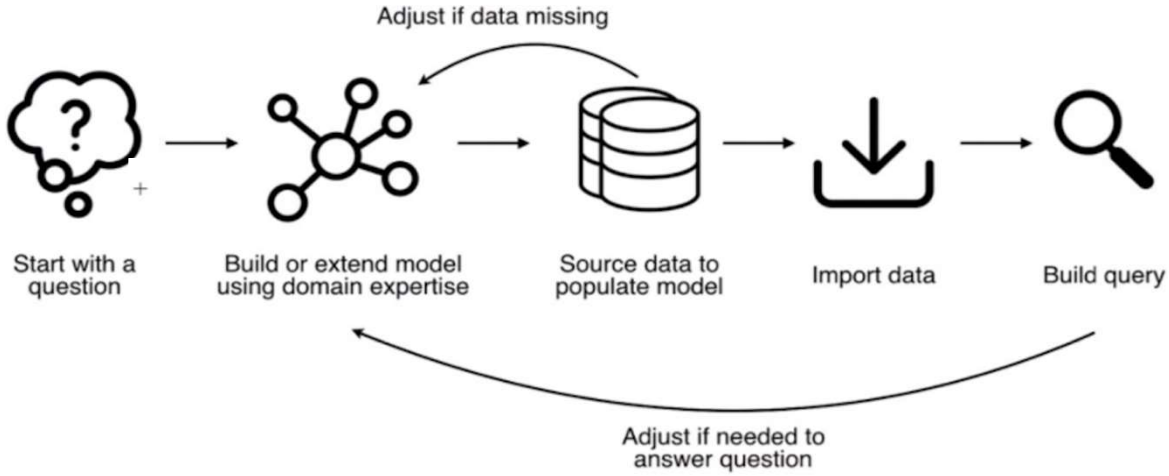
1. Pointers instead of Lookups
2. Fixed Sized Records
3. "Joins" on Creation
4. Spin Spin Spin through this data structure

In practice, what does this mean?



- How are Jane and Anne connected?
- Might John and David know each other?
- Who went to Grange Hill?

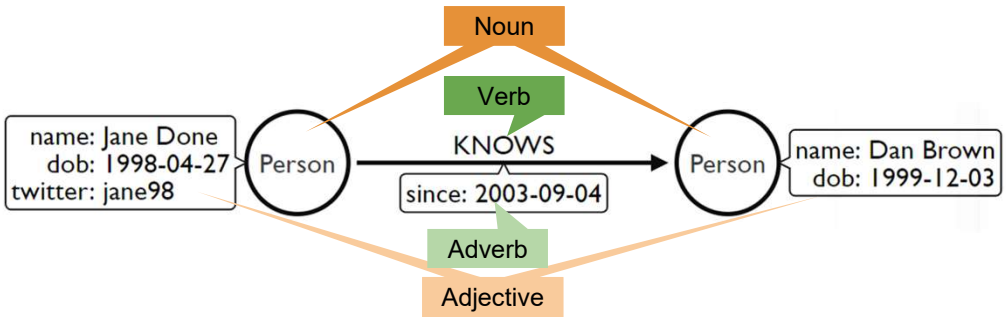
The graph data modeling and implementation process



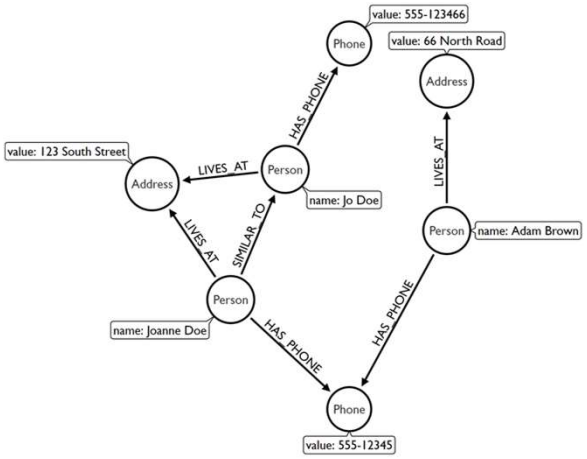
34

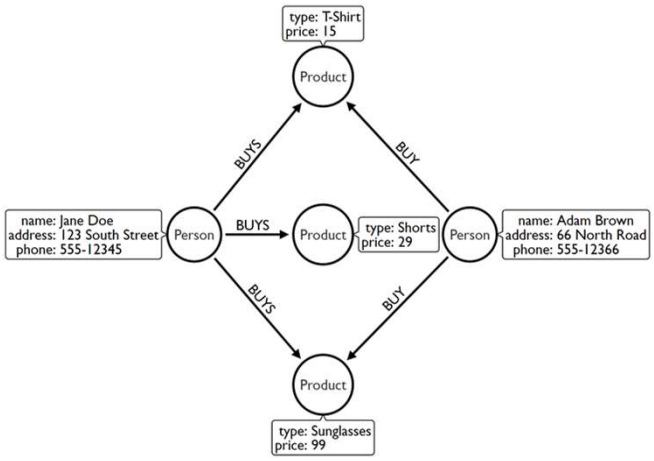
Some basic graph modelling rules…

- If it's a verb, it's probably a relationship
- If it's a noun, it's probably node label
- If it's an adverb/adjective, it's probably a property



I am looking for fraud rings….





I am looking for recommendations….

Modelling emails

Your turn:

- Think about what happens during email exchanges
- How might we model this as a graph?

Some hints:

- Think about what questions we might ask
- 'Generate' a tiny data set
- Identify what the elements might be

Arrows:

- www.apcjones.com/arrows/

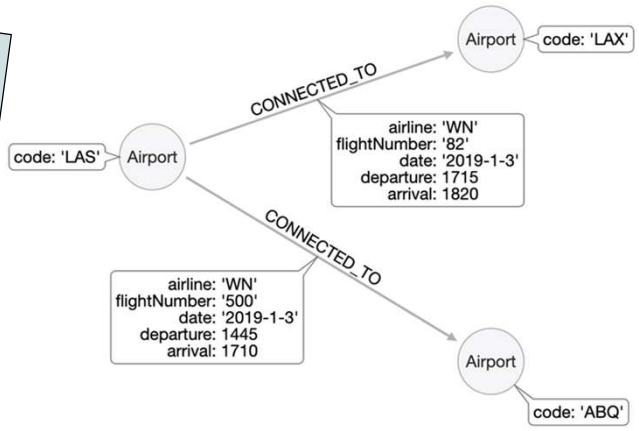
Domain question for our model

As an air travel enthusiast
I want to know how airports are connected
So that I can find the busiest ones

38

Initial data model with sample data

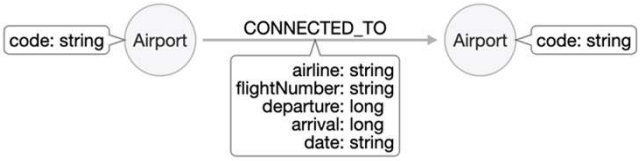
As an air travel enthusiast
I want to know how airports are connected
So that I can find the busiest ones



39

Do we need to change the model?

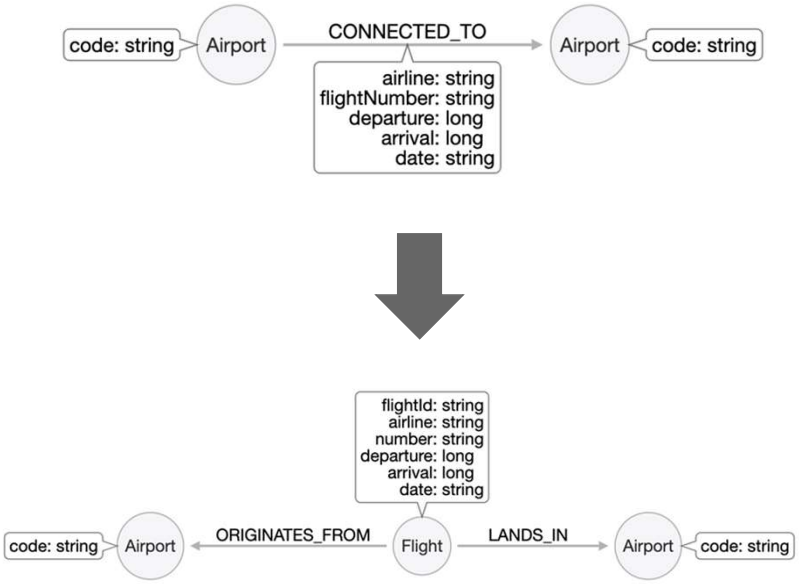
Question: What are the airports and flight information for flight number 1016 for airline WN?



Why are we changing the model?
- No indexes on relationship properties
- We have to scan every single Airport node!

40

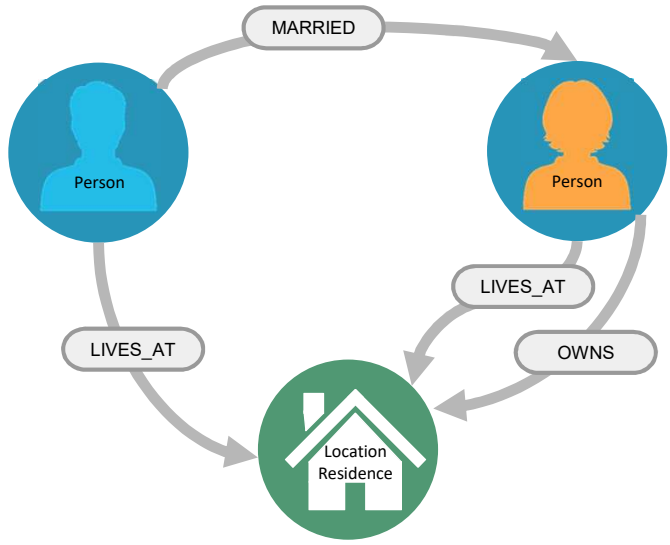
Steps: Refactoring the model



41

What is Cypher?

- Declarative query language
  - Focuses on **what**, not how
- Uses keywords such as **MATCH**, **WHERE**, **CREATE**
- Runs on the database server
- ASCII art to represent nodes and relationships
  - () - Nodes
  - [] - Relationships
  - {} - Properties



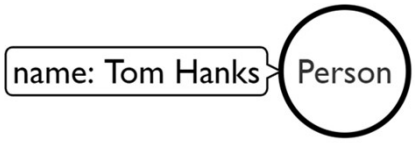
42

Use MATCH to retrieve nodes

```
//Match all nodes
MATCH (n)
RETURN n;

//Match all nodes with a Person label
MATCH (n:Person)
RETURN n;

//Match all nodes with a Person label and property name is 'Tom Hanks'
MATCH (n:Person {name: 'Tom Hanks'})
RETURN n;
```



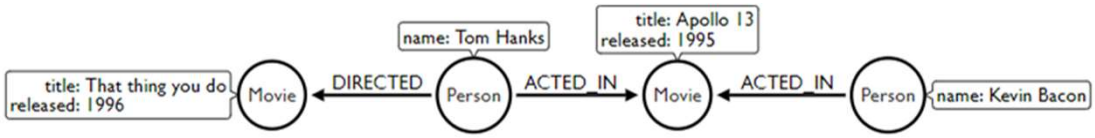
Use MATCH and properties to retrieve nodes

```
//Return nodes with label Person and name property is 'Tom Hanks' - Inline
MATCH (p:Person {name: 'Tom Hanks'}) //Only works with exact matches
RETURN p;

//Return nodes with label Person and name property equals 'Tom Hanks'
MATCH (p:Person)
WHERE p.name = 'Tom Hanks'
RETURN p;

//Return nodes with label Movie, released property is between 1991 and 1999
MATCH (m:Movie)
WHERE m.released > 1990 AND m.released < 2000
RETURN m;
```

Extending the MATCH




```
//Find all the movies Tom Hanks acted in
MATCH (:Person {name:'Tom Hanks'})-[:ACTED_IN]->(m:Movie)
RETURN m.title;

//Find all the movies Tom Hanks directed and order by latest movie
MATCH (:Person {name:'Tom Hanks'})-[:DIRECTED]->(m:Movie)
RETURN m.title, m.release ORDER BY m.release DESC;

//Find all of the co-actors Tom Hanks has ever worked with
MATCH (:Person {name:'Tom Hanks'})-->(:Movie)<-[:ACTED_IN]-(coActor:Person)
RETURN coActor.name;
```

Use relationships to retrieve results

```
//Find nodes with an ACTED_IN relationship
MATCH (p)-[:ACTED_IN]->()
RETURN p

//Find Person nodes with an ACTED_IN or DIRECTED_IN relationship
MATCH (p:Person)-[:ACTED_IN|DIRECTED]->()
RETURN p

//Find Person nodes who don't have an ACTED_IN relationship
MATCH (p:Person)
WHERE NOT (p)-[:ACTED_IN]->()
RETURN p
```

Variable relationship hops

```
//Find nodes up to 2 hops from Tom Hanks
MATCH path = (:Person {name:'Tom Hanks'})-[*0..2]-(:Person)
RETURN path;

//Find the shortest path between Tom Hanks and Tom Cruise
MATCH path = shortestPath((hanks:Person {name:'Tom Hanks'})-[*]-
(cruise:Person {name:'Tom Cruise'}))
RETURN path;
```

WARNING!

Be very careful with unbounded, undirected relationship traversals!

Expanding patterns, collections and counts

```
//Find me all actors that have acted and directed movies
MATCH (p:Person)-[:ACTED_IN]->(), (p)-[:DIRECTED_IN]->()
RETURN p.name;

//Find all reviewers of actor directors and their movies
MATCH (a:Person)-[:ACTED_IN|DIRECTED]->(m:Movie)<-[:REVIEWED]-(r:Person)
RETURN m.title AS Title,
       COLLECT(a.name) AS `Actor/Director`,
       COLLECT(r.name) AS Reviewer;

//Find all Movies and reviewers with ratings less than 50
MATCH (m:Movie)<-[r:REVIEWED]-(p:Person) WHERE r.rating < 50
RETURN m.title AS Title, r.rating AS Rating, p.name AS Reviewer;
```

CREATE

```
//Create a person node called 'Tom Hanks'
CREATE (p:Person {name:'Tom Hanks'});

//Create an ACTED_IN relationship between 'Tom Hanks' and 'Apollo 13'
MATCH (p:Person {name:'Tom Hanks'}), (m:Movie {title:'Apollo 13'})
CREATE (p)-[:ACTED_IN]->(m);

//Create the pattern of 'Tom Hanks' ACTED_IN 'Apollo 13'
//This will create the entire pattern, nodes and all!
CREATE (:Person {name:'Tom Hanks')-[:ACTED_IN]->(:Movie {title:'Apollo 13});
```

MERGE

- Similar to an upsert
  - If the pattern doesn't exist, it will be created
  - If the pattern does exist, it will be MATCHed
- Good MERGE practice
  - Only merge on the unique property
  - Always set other properties after the initial MERGE

MERGE

```
//Merge a person node called 'Tom Hanks'
MERGE (p:Person {name:'Tom Hanks'});

//MERGE an ACTED_IN relationship between 'Tom Hanks' and 'Apollo 13'
MATCH (p:Person {name:'Tom Hanks'}), (m:Movie {title:'Apollo 13'})
MERGE (p)-[:ACTED_IN]->(m);

//MERGE the pattern of 'Tom Hanks' ACTED_IN 'Apollo 13'
//If the identical pattern below doesn't exist, it will be created!
MERGE (:Person {name:'Tom Hanks'})-[:ACTED_IN]->(:Movie {title:'Apollo 13});
```

MERGE

```
//If creating Person node 'Lju', set favourite colour to 'Red'
MERGE (lju:Person {name:'Lju'})
ON CREATE SET lju.favColour = 'Red';


//If Person node 'Lju' exists, update favourite food to 'Chocolate'
MERGE (lju:Person {name:'Lju'})
ON MATCH SET lju.favFood = 'Chocolate';
```

Cypher can be case-sensitive

Case sensitive:

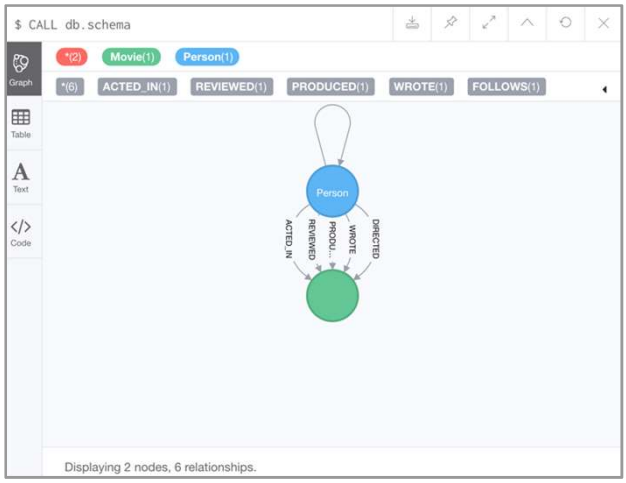
- Node labels
- Relationship types
- Property keys
- Property values (where relevant)
- Function and procedures

Not case sensitive:

- Cypher key words (e.g. MATCH, NOT, ORDER BY, etc.)

Examine the data model

```
CALL db.schema
```



54

Your turn - Cypher

- Match a Person called John Doe
- Find FRIENDS_OF John Doe
- COUNT John Doe's direct acquaintances

Your turn - Cypher

- Match a Person called John Doe

```
MATCH (p:Person {name: "John Doe"})
RETURN p
```

- Find FRIENDS_OF John Doe

```
MATCH (:Person {name: "John Doe"})-[:FRIENDS_OF]-(friend:Person)
RETURN friend
```

- COUNT John Doe's direct acquaintances

```
MATCH (:Person {name: "John Doe"})--(acq:Person)
RETURN COUNT(acq)
```

Graph Algorithms

Pathfinding & Search

Finds optimal paths or evaluates route availability and quality

Centrality / Importance

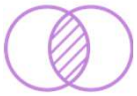
Determines the importance of distinct nodes in the network

Community Detection

Detects group clustering or partition options

Estimates the likelihood of nodes forming a future relationship

Link Prediction



Similarity

Evaluates how alike nodes are

Neo4j Graph Algorithms

Pathfinding & Search

- *Parallel Breadth First Search**
- Parallel Depth First Search
- *Shortest Path**
- Single-Source Shortest Path
- All Pairs Shortest Path
- Minimum Spanning Tree
- A* Shortest Path
- Yen's K Shortest Path
- K-Spanning Tree (MST)
- Random Walk

Centrality / Importance

- Degree Centrality
- Closeness Centrality
- CC Variations: Harmonic, Dangalchev, Wasserman & Faust
- Betweenness Centrality
- Approximate Betweenness Centrality
- *PageRank**
- Personalized PageRank
- ArticleRank
- Eigenvector Centrality

Community Detection

- *Triangle Count**
- Clustering Coefficients
- *Connected Components (Union Find)**
- *Strongly Connected Components**
- *Label Propagation**
- Louvain Modularity – 1 Step & Multi-Step
- Balanced Triad (identification)

Similarity

- Euclidean Distance
- Cosine Similarity
- Jaccard Similarity
- Overlap Similarity
- Pearson Similarity

Link Prediction

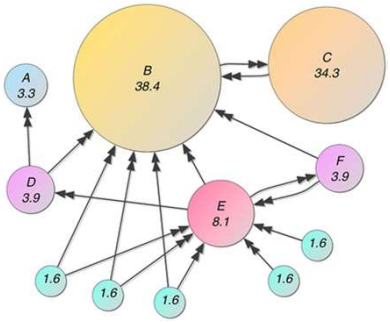
- Adamic Adar
- Common Neighbors
- Preferential Attachment
- Resource Allocations
- Same Community
- Total Neighbors

neo4j.com/docs/graph-algorithms/current/

** Available in GraphFrames*

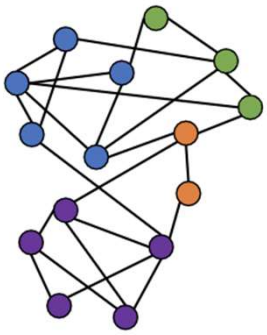
PageRank algorithm

- Use when
  - Anytime you're looking for broad influence over a network
  - Many domain specific variations for differing analysis, e.g. Personalized PageRank for personalized recommendations
- Examples:
  - Twitter Recommendations
  - Fraud Detection



Louvain modularity

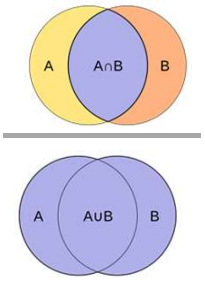
- Use when
  - Community Detection in large networks
  - Uncover hierarchical structures in data
- Examples
  - Money Laundering
  - Protein-Protein-Interactions



Jaccard similarity

- Use when
  - Computing pair-wise similarities
  - Accommodates vectors of different lengths
- Examples
  - Recommendations
  - Disambiguation

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$



Crash Course in Neo4j Part 3: Getting data in and out, graph algorithms and connecting to Spark

You turn - the Movie Database

Your turn - a little bit extra...

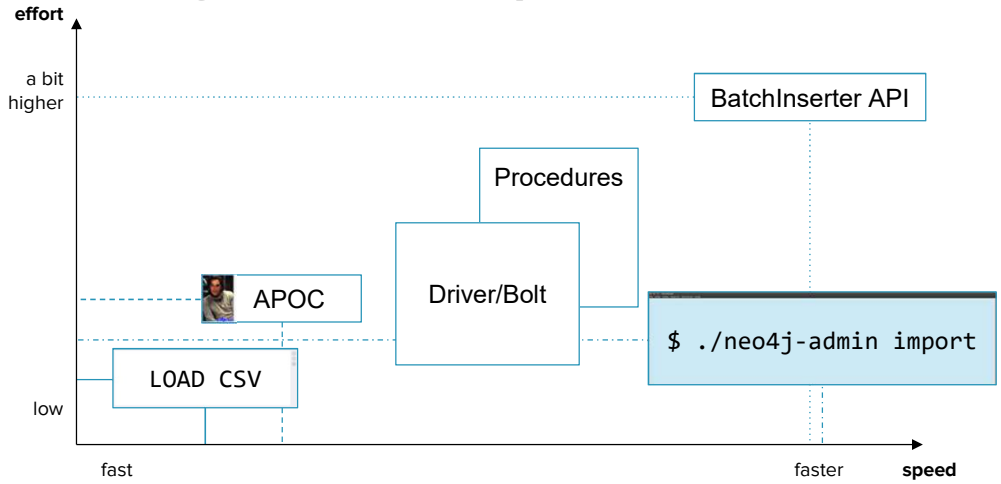
PageRank the movie db (Neo4j Desktop only - graph algorithms plugin installed)

(Copy and paste from: tinyurl.com/y5b6qnhy)

```
CALL algo.pageRank.stream(
        'MATCH (p:Person) RETURN id(p) AS id',
        'MATCH (p1:Person)-->()<--(p2:Person)
                RETURN distinct id(p1) AS source, id(p2) AS target',
    {graph:'cypher'})
YIELD nodeId, score
RETURN algo.asNode(nodeId).name AS name, score ORDER BY score DESC
```

What happens if you select the pattern to ACTED_IN relationship only?

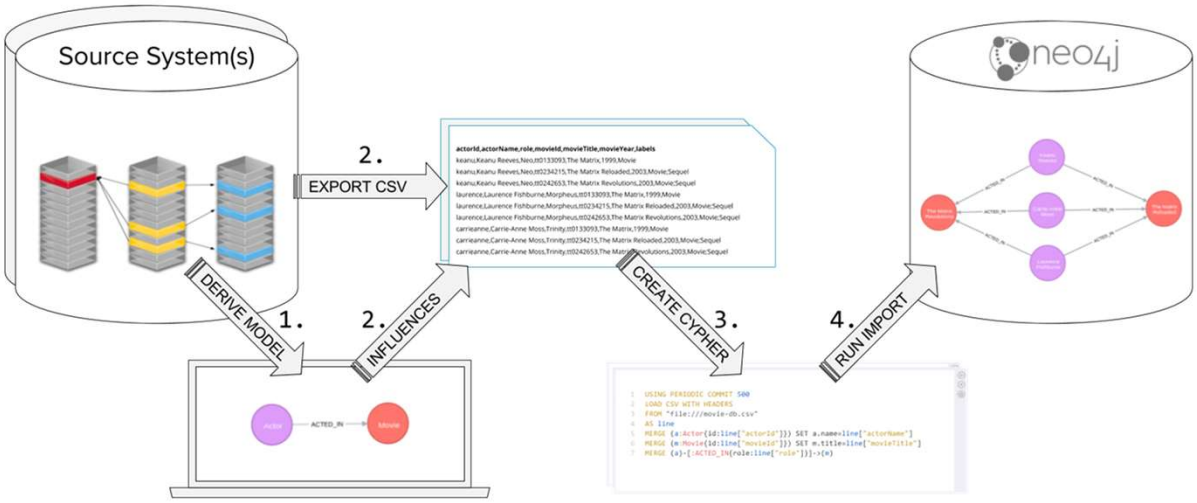
Loading data - what options do we have?



Cypher and LOAD CSV

- Probably the simplest method
- Initial import or update
- Database is online during import i.e. transactional!
- Create indexes upfront
- The cluster is being synchronized automatically

Cypher and load CSV import approach



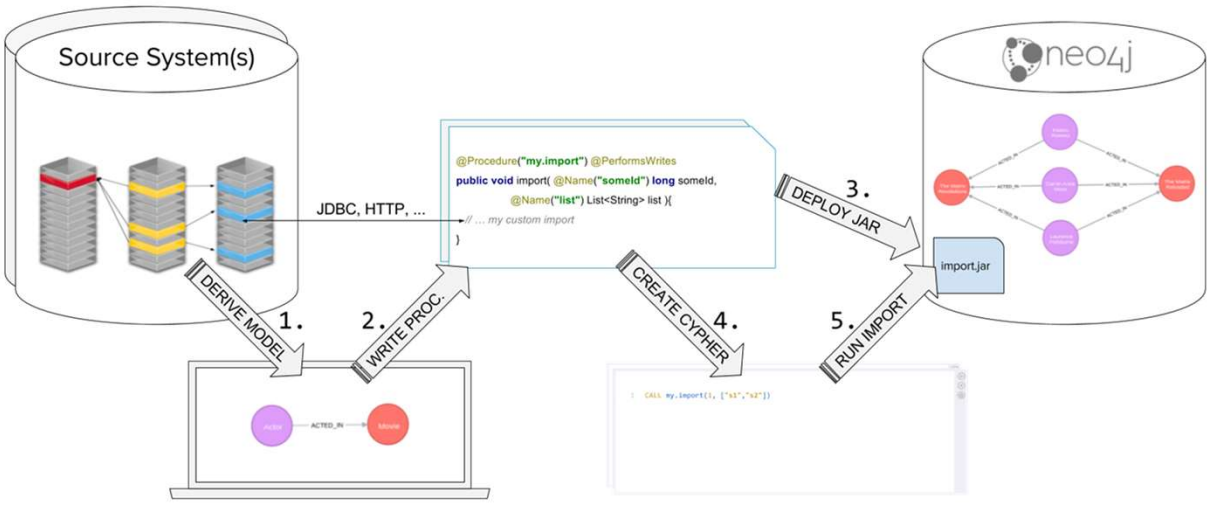
Cypher and APOC

- Iterate / batching
- Plenty of procedures and functions
- GraphML
- JDBC
- .. and others (e.g. XML, JSON, ...)

Procedures

- Extension of the Neo4j server
  - Will be deployed as .jar file to the plugins folder
- Database is online during import, transactional!
- Make use of one of our APIs for graph processing
  => Performance
- Fine grained user/role concept
- The cluster is being synchronized automatically

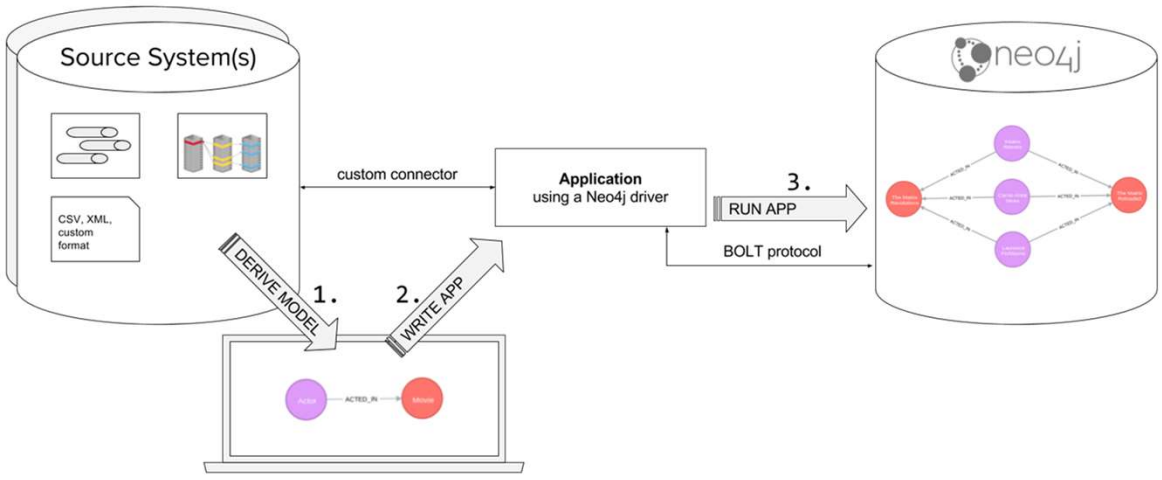
Procedure import approach



Driver via Bolt

- Drivers for many languages available
  - .NET, Java, JavaScript, Python, Go are officially supported
- Transactional processing
- Batching
- Parallelization possible

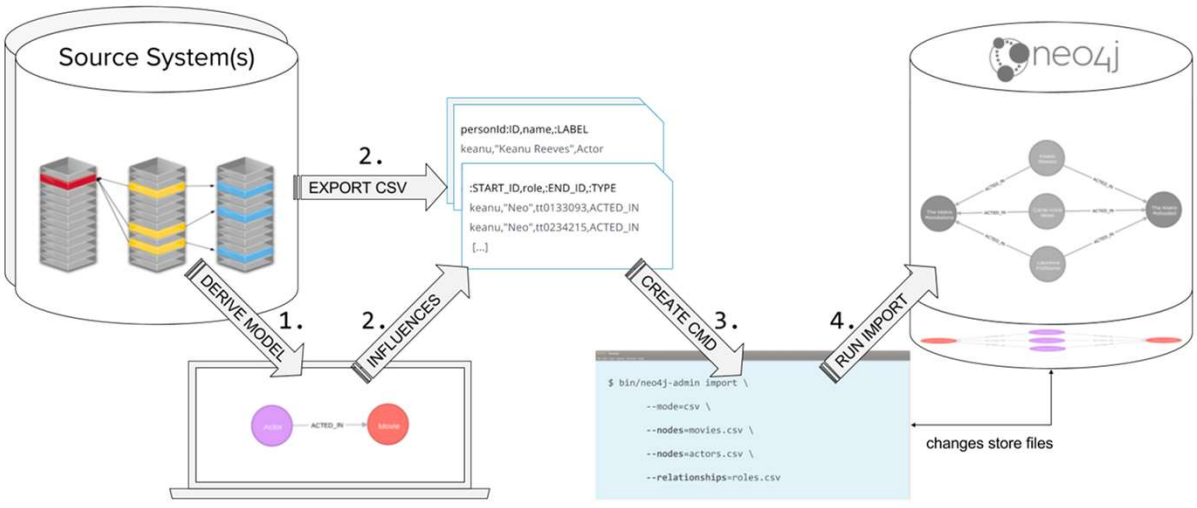
Bolt import approach



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

- Fastest method (w.r.t. writes/second)
- Initial import; a new database is being created
- Database is offline during import
- No need to create indexes in advance
- The cluster needs to be synchronized after the import

Neo4j-admin import import approach



Graphs in Spark and Neo4j

Spark is an **immutable data** processing engine

- Distributable computational analysis over massive data sets
- Does not natively represent relationships
- Analytical operations

Neo4j is a **native transactional CRUD** database

- Has optimized in-process parallel graph algorithms
- Uses native graph data representation
- Real-time, transactional operations

Passing data between Neo4j and Spark

- Discussed data loading approaches, these can be extended to Spark:
  - Flat file import/export via neo4j-admin import
  - Use of the drivers
  - Use of procedures
  - And so forth
- Neo4j - Spark connector
- Neo4j Morpheus

Neo4j Spark Connector

- Community contribution
- Uses the binary Bolt protocol
- Offers Spark 2.0 APIs for:
  - RDD
  - DataFrames
  - GraphX
  - GraphFrames

https://github.com/neo4j-contrib/neo4j-spark-connector

Morpheus: SQL + Cypher in one session

Graphs and tables are both useful data models

- Finding paths and subgraphs, and transforming graphs
- Viewing, aggregating and ordering values

The **Morpheus** project parallels Spark SQL

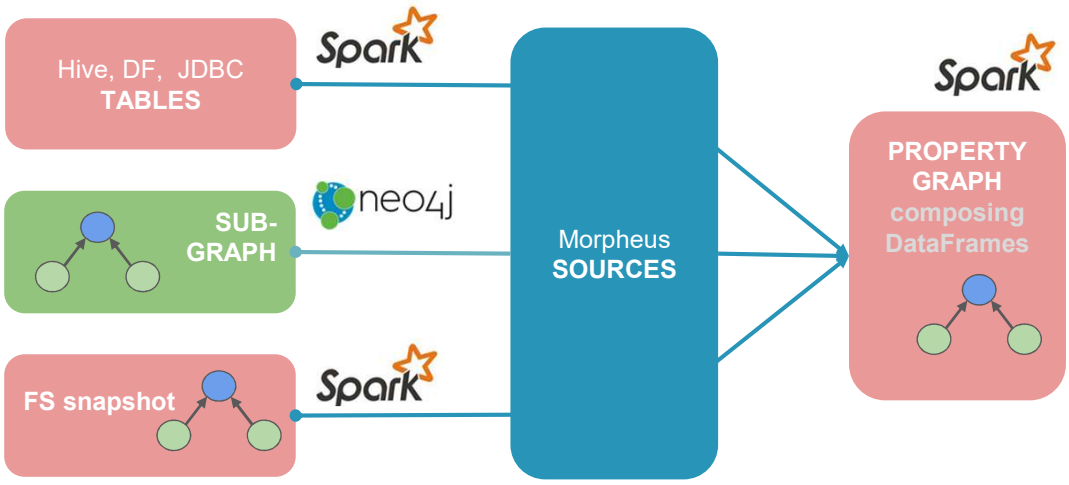
- `PropertyGraph` type (composed of DataFrames)
- Catalog of graph data sources, named graphs, views,
- Cypher query language

A `CypherSession` adds graphs to a `SparkSession`

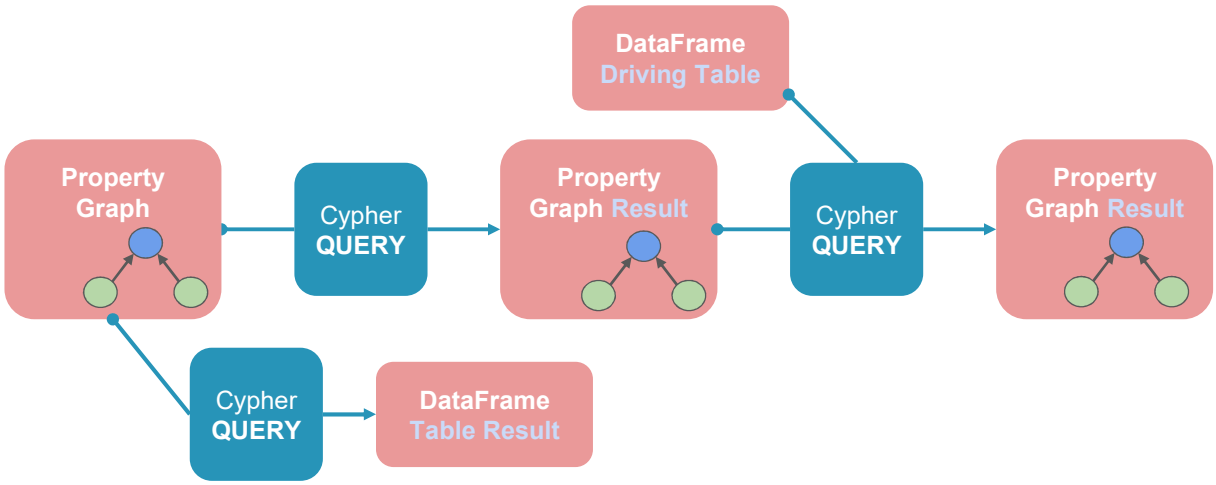
What is Morpheus used for?

- Data integration
  - Integrate (non-)graphy data from multiple, heterogeneous data sources into one or more property graphs
- Distributed Cypher execution
  - OLAP-style graph analytics
- Data science
  - Integration with other Spark libraries
  - Feature extraction using Neo4j Graph Algorithms

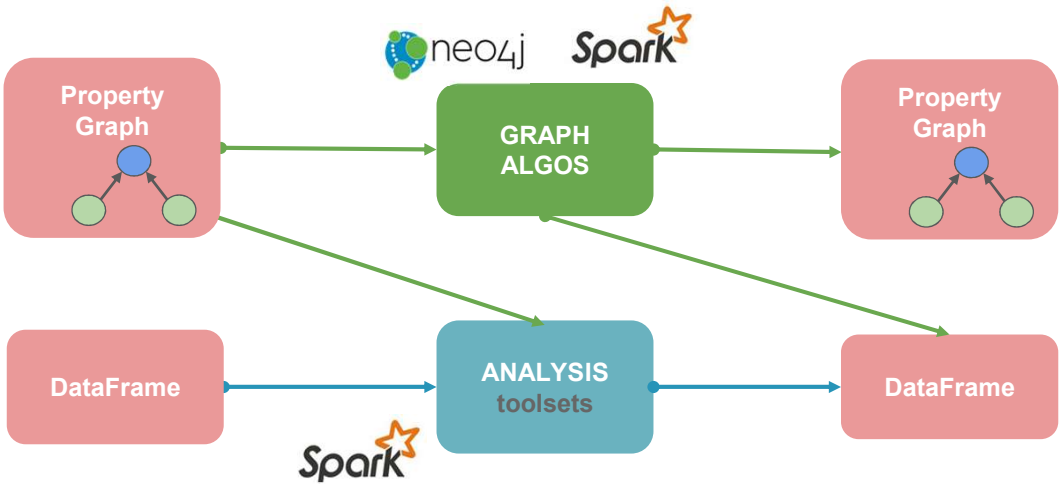
Morpheus creates Spark Graphs …



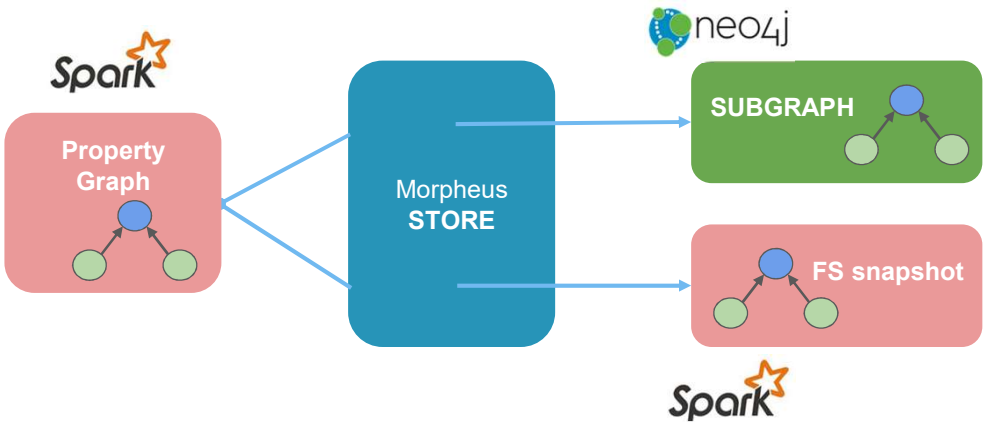
… wrangles Spark Graphs …



… analyses graphs in Spark and Neo4j …

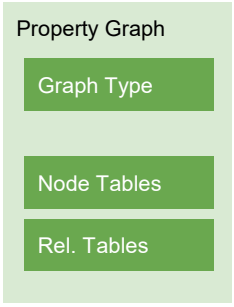


… and stores Spark Graphs

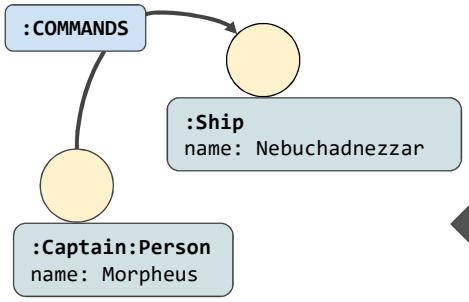


"Tables for Labels"

- In Morpheus, PropertyGraphs are represented by
  - **Node Tables** and **Relationship Tables**
- Tables are represented by DataFrames
  - Require a fixed schema
- Property Graphs have a **Graph Type**
  - Node and relationship types that occur in the graph
  - Node and relationship properties and their data type



"Tables for Labels"



:Captain:Person

id	name
0	Morpheus

:Ship

id	name
1	Nebuchadnezzar

:COMMANDS

id	source	target
0	0	1

```
Graph Type {
 :Captain:Person (
    name: STRING
 ),
 :Ship (
   name: STRING
 ),
 :COMMANDS
}
```

Constructing graphs

Input: a property graph
Output: a property graph

```
FROM GRAPH socialNetwork
MATCH (p:Person)-[:FRIEND*2]->(foaf)
WHERE NOT (p)-[:FRIEND]->(foaf)
CONSTRUCT
  CREATE (p)-[:POSSIBLE_FRIEND]->(foaf)
RETURN GRAPH
```

Language features available in Morpheus

Querying multiple graphs

Input: property graphs
Output: a property graph

```
FROM GRAPH socialNetwork
MATCH (p:Person)
FROM GRAPH products
MATCH (c:Customer)
WHERE p.email = c.email
CONSTRUCT ON socialNetwork, products
      CREATE (p)-[:IS]->(c)
RETURN GRAPH
```

Language features available in Morpheus

Creating graph views

Input: property graphs
Output: a property graph

```
CATALOG CREATE VIEW youngFriends($inGraph){
  FROM GRAPH $inGraph
  MATCH (p1:Person)-[r]->(p2:Person)
  WHERE p1.age < 25 AND p2.age < 25
  CONSTRUCT
    CREATE (p1)-[COPY OF r]->(p2)
  RETURN GRAPH
}
```

Language features available in Morpheus

Spark Graph

Property Graph Model
Cypher queries
Graph Algorithms

Coming in Spark 3.0!

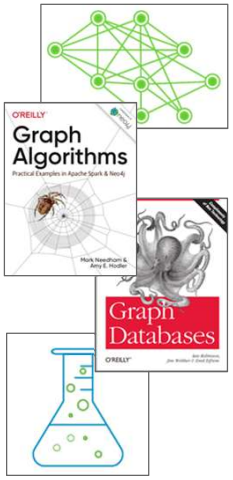
Spark Project Improvement Proposal

- SPARK-25994 Spark Graph for Apache Spark 3.0
  - Property Graphs, Cypher Queries, and Algorithms
- Cypher-compatible Property Graph type based on DataFrames
- Replaces GraphFrames querying with Cypher
- Reimplements GraphFrames/GraphX algos on the Property Graph type
- Implementation is based on Spark SQL
- Provide Scala, Python and Java APIs

Resources and further reading

Learn and try

- Neo4j getting started guide: neo4j.com/developer/get-started/
- Cypher reference card: neo4j.com/docs/cypher-refcard/current/
- Getting started sandboxes: neo4j.com/sandbox-v2/
- GraphGist: neo4j.com/graphgists/



Books

- Graph Databases book: neo4j.com/graph-databases-book/
- Graph Algorithms book: neo4j.com/graph-algorithms-book/

Other

- Neo4j Morpheus: github.com/opencypher/cypher-for-apache-spark
- Neo4j Labs: neo4j.com/labs/

Thank you!

Our contact details:

- lju@neo4j.com @ellazal
  medium.com/@lju
- erik.nygren@neo4j.com linkedin.com/in/eriknygrens