

# Knowledge Graphs 2021: The great (graph) convergence

**Jesús Barrasa**  
Neo4j



# Jesús Barrasa

Director of SE EMEA

@Barrasadv / [jesus@neo4j.com](mailto:jesus@neo4j.com)



PhD in RDB to RDF mapping (2007)

6 Years at Ontology (UK)

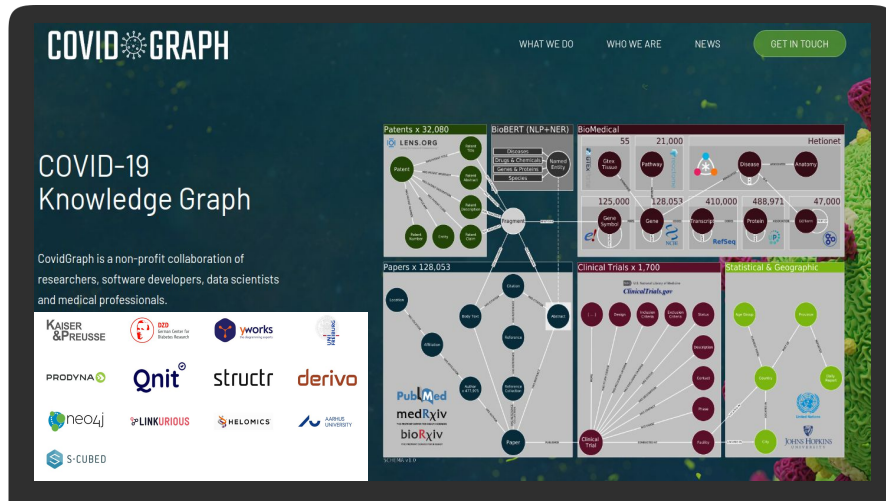
2Y Stint at data virtualization: Denodo

Last 6 years: Neo4j

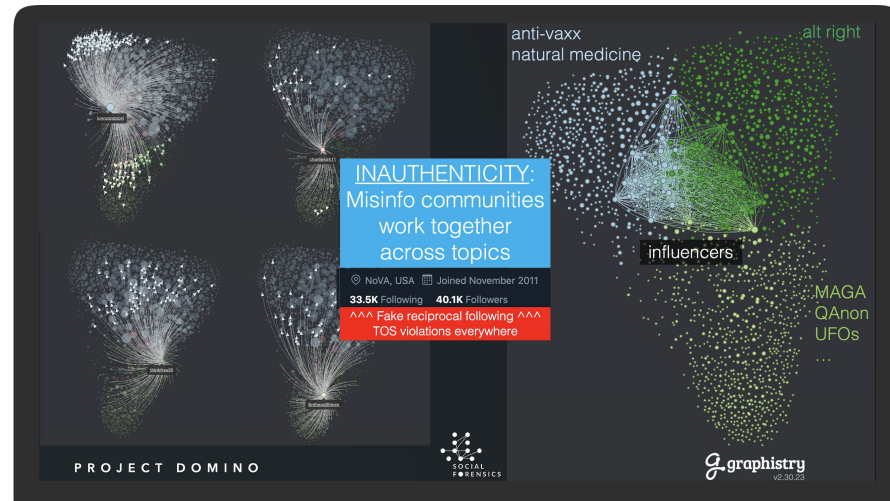
The neosemantics project

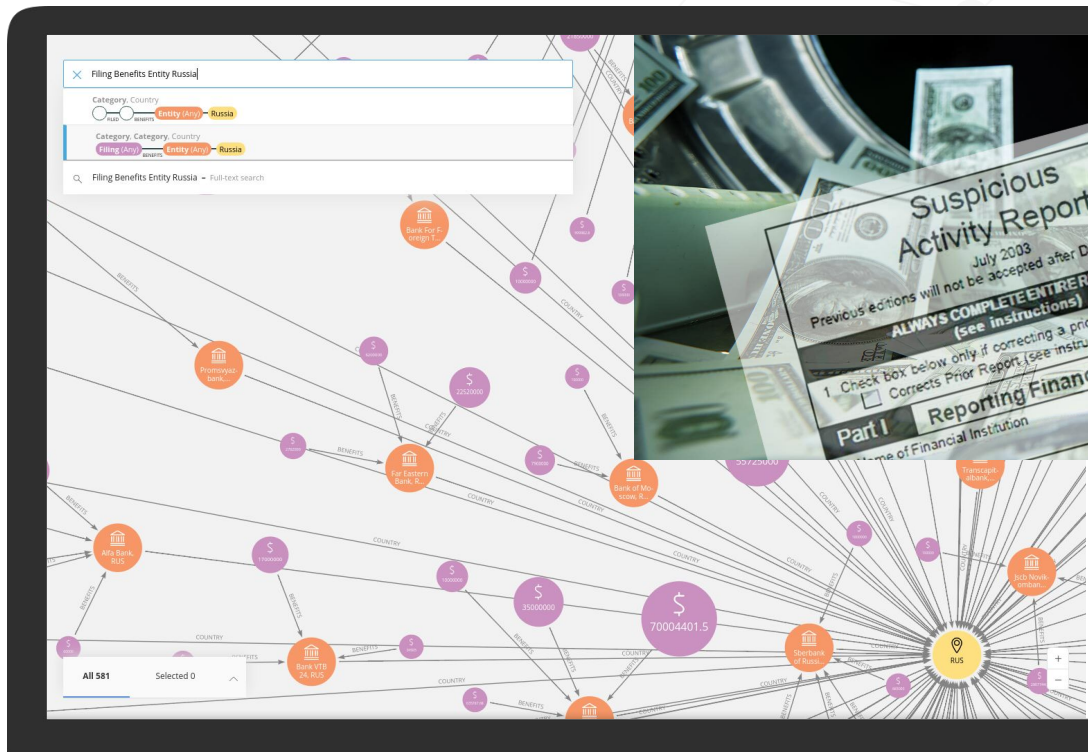
# Graphs4Good in the times of COVID

## Covid\*Graph



## Project Domino





<https://www.icij.org/investigations/fincen-files/global-banks-defy-u-s-crackdowns-by-serving-oligarchs-criminals-and-terrorists>

# The Great graph Convergence



# converge verb



Save Word

con·verge | \ kən-ˈvərj ˌ \

**converged; converging**

## Definition of *converge*

*intransitive verb*

- 1 : to tend or move toward one point or one another : come together : MEET  
// *converging* paths  
// Police cars *converged* on the accident scene.
- 2 : to come together and unite in a common interest or focus  
// Economic forces *converged* to bring the country out of the recession.
- 3 : to approach a limit as the number of terms increases without limit  
// the series *converges*



1

Huge interest in  
**graph ML**

2

Renewed interest in  
**RDF/ semantics**

3

Commercial  
Graphs **market  
growth**

1

Huge interest in  
graph ML





# 2

Renewed interest in  
**RDF/ Semantics**



**Gartner.**

## How to Build Knowledge Graphs That Enable AI-Driven Enterprise Applications

- Take an agile approach to ontology and knowledge graph development to decrease time to value.
- Support a minimum viable graph (MVG) approach by incorporating machine learning techniques.

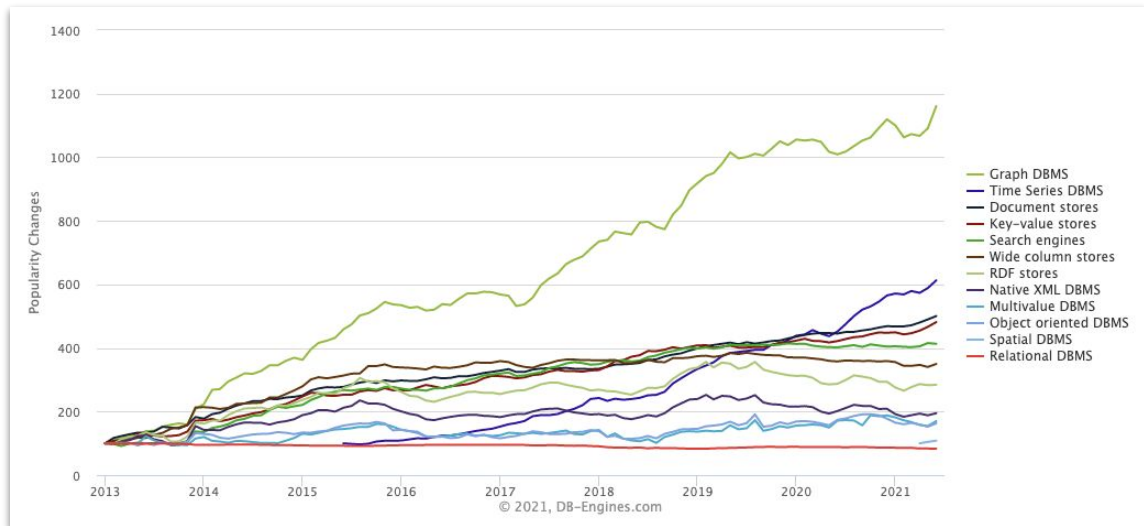
## Gartner Identifies Top 10 Data and Analytics Technology Trends for 2021

### Trend 8: Graph Relates Everything

Gartner predicts that by 2025, graph technologies will be used in 80% of data and analytics innovations, up from 10% in 2021, facilitating rapid decision making across the organization.

# 3

Commercial  
Graphs **market  
growth**



[https://db-engines.com/en/ranking\\_categories](https://db-engines.com/en/ranking_categories)

# The trends come from different directions



## Interoperability

And explicit semantics, of course, but targeted interoperability: ER

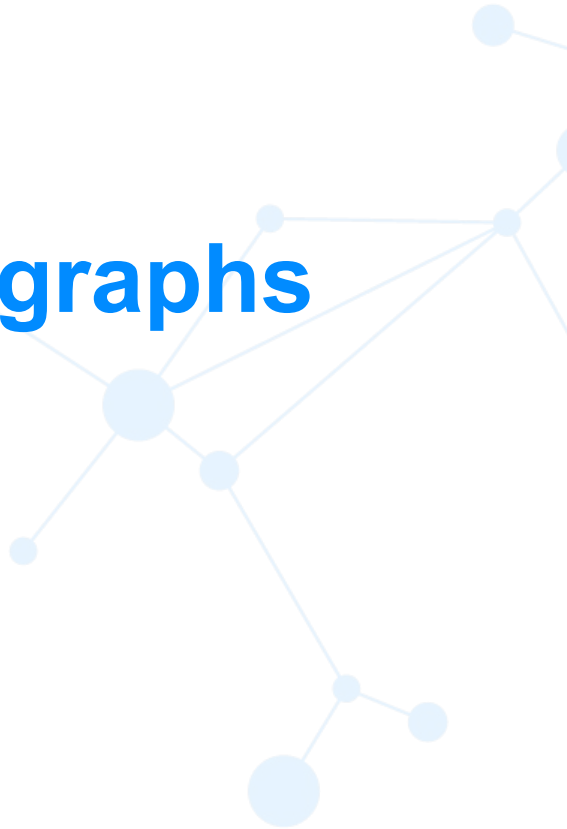
## Better predictions

Graph features turn out to be significantly more predictive than attributes

## Graph management

Shortest path to building a graph based solution both operational or analytical

# Context: How do property graphs relate to RDF graphs?





**GRAPH =  
VERTICES + EDGES**

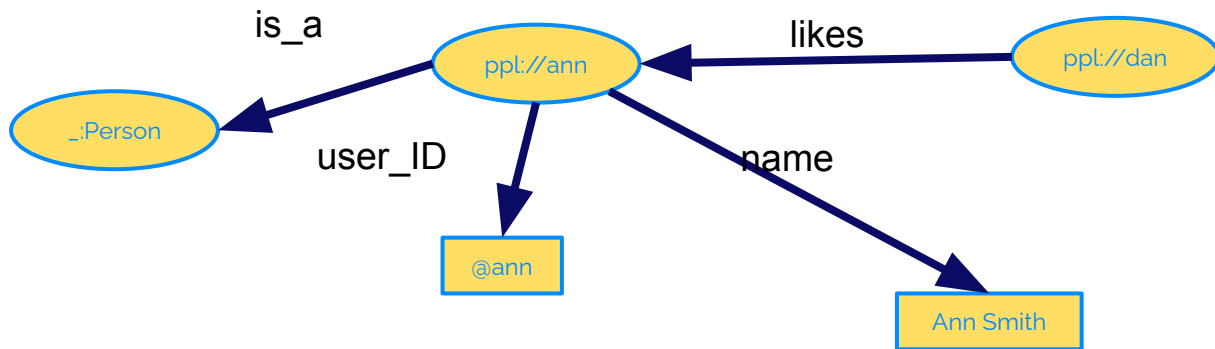
# RDF statements (triples)

ppl://ann is a person

ppl://ann user ID is @ann

ppl://ann name is Ann Smith

ppl://dan likes ppl://ann



Vertices

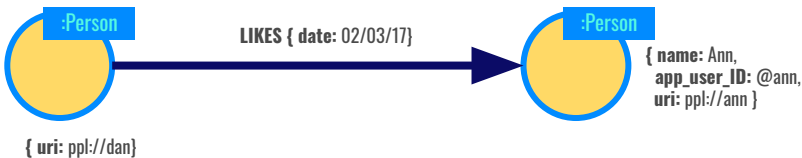
Edges

# PG connected objects (with properties)

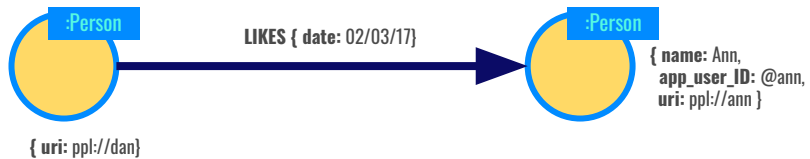
There is a person that is described by her name: Ann, her user ID: @ann and a globally unique identifier: <ppl://ann>

There is another person with a unique identifier: <ppl://dan>

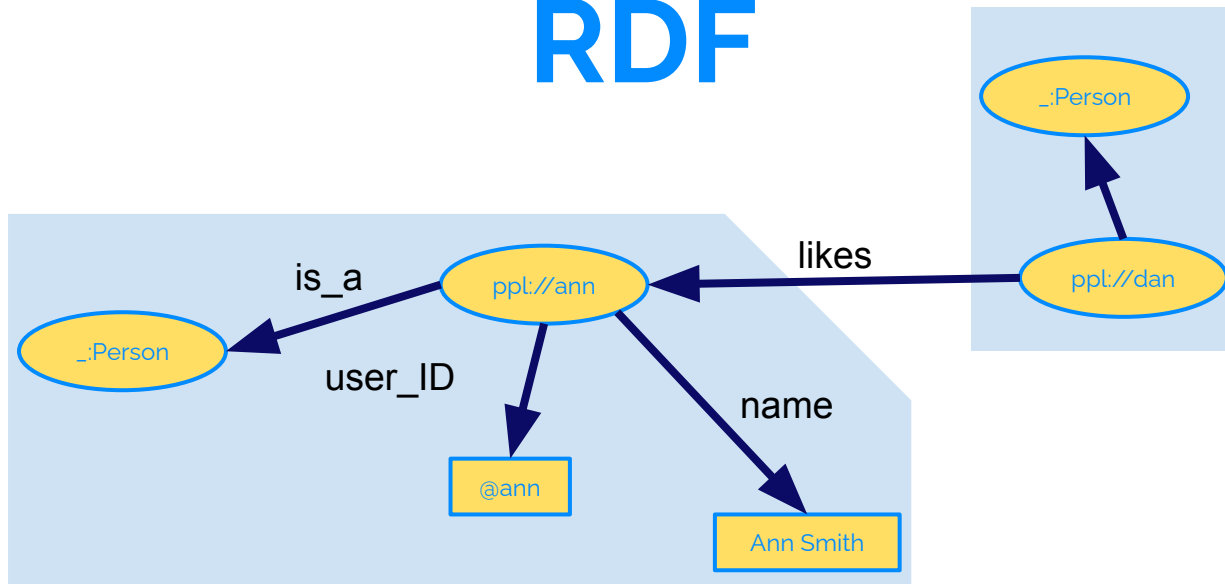
Dan likes Ann



# PG



# RDF





**Any RDF graph  
can be  
automatically  
imported into a  
PG DB in a  
lossless manner**

**Any Property  
Graph can be  
automatically  
serialised as  
RDF (or RDF\*)  
in a lossless  
manner.**



# So there are really no diffs?

## Just minor ones

- Property Graphs identify relationships (edges) uniquely
  - (some kind of native implementation of the singleton property  
<http://dl.acm.org/citation.cfm?id=2567973>)
- Multivalued properties are arrays/collections

## Context: How do property graphs relate to RDF graphs?

### SPARQL

```
prefix ms: <http://myschma.me/>
prefix rdf: <http://www[... ]#>
```

```
SELECT ?who
{
  ?a a ms:Person .
  ?a ms:name ?asName .
  FILTER regex(?asName, 'Ann')
  ?who ms:likes ?a .
}
```

### Cypher

```
MATCH (who)-[:LIKES]->(a:Person)
WHERE a.name CONTAINS 'Ann'
RETURN who
```

A query: Who likes this person named Ann?

**Context: How do property graphs relate to RDF graphs?**

**Integrity:** ACID?

**Storage:** Native, RDB, NoSQL

**Clustering:** Consistency level

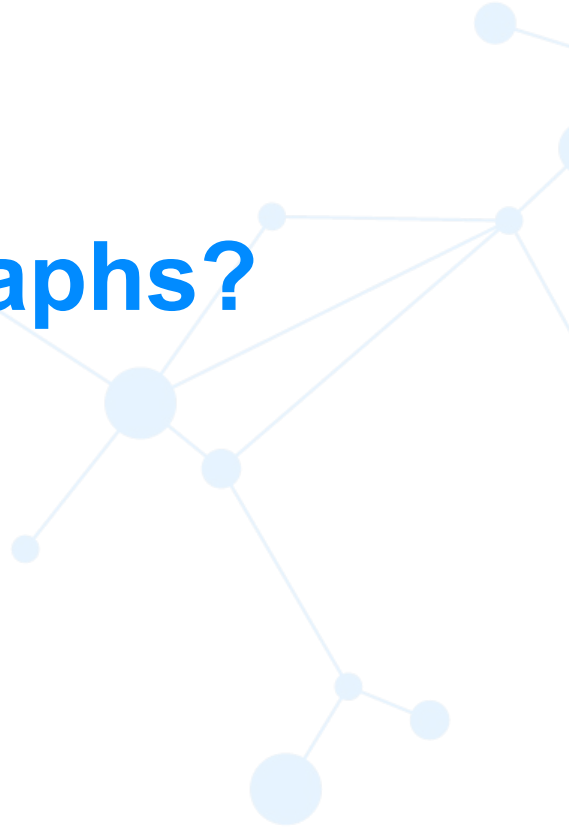
**Workloads:** Deep traversals, Algorithms.

**Licensing & Support**

**Open/closed Source**

**Tech stack / integrations / architectures**

# Are we still Knowledge Graphs?



# A KG is...

It's up to you really... here's an idea

|         | <i>Persisted in a DB</i> | <i>Described as RDF</i> | <i>Integrates disparate data</i> | <i>Models master data</i> | <i>Explicit &amp; formal semantics</i> | <i>Queryable via SPARQL</i> | <i>A graph!</i> | <i>At scale!</i> | <i>Collaborative</i> | <i>...</i> | <i>KG</i> |
|---------|--------------------------|-------------------------|----------------------------------|---------------------------|--|-----------------------------|-----------------|------------------|----------------------|------------|-----------|
| def#1   | 1                        | 1                       | 0                                | 0                         | 1                                      | 1                           | 1               | 1                | 0                    |            | 1         |
| def#2   | 0                        | 1                       | 1                                | 0                         | 1                                      | 0                           | 1               | 0                | 0                    |            | 1         |
| def#3   | 1                        | 1                       | 0                                | 0                         | 1                                      | 0                           | 1               | 0                | 0                    |            | 1         |
| ...     |                          |                         |                                  |                           |  |                             |                 |                  |                      |            |           |
| def#183 | 1                        | 0                       | 0                                | 0                         | 0                                      | 0                           | 1               | 1                | 0                    |            | 0         |

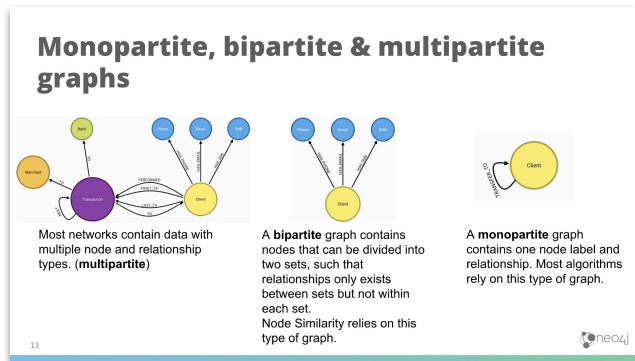
# How are Knowledge Graphs built with Neo4j?



# KG construction



Purely pragmatic: Start from the end.  
Keep usage in mind... Iterate



Query perf considerations. Model evolution, etc... change!

Then remodelling for analytics.  
Monopartite, bipartite... views on KG



# Observation#1

## KG construction is largely an engineering task

- Important (huge) investment in modelling trainings
- (change!) Model Refactoring...
- Things are different on the consuming side (graphs help, at least via visual exploration)

# Observation#2

## Model reuse is (close to) non-existent

- There's no such thing as a library of public Property Graph models
- What does a PG “ontology” look like? -> The multilayered network example.

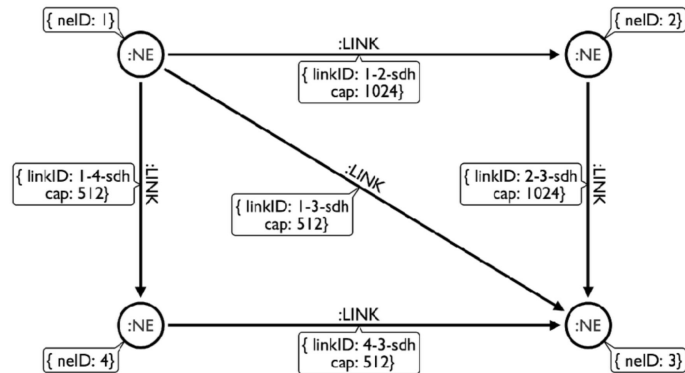
# The social network

## Route oriented Model



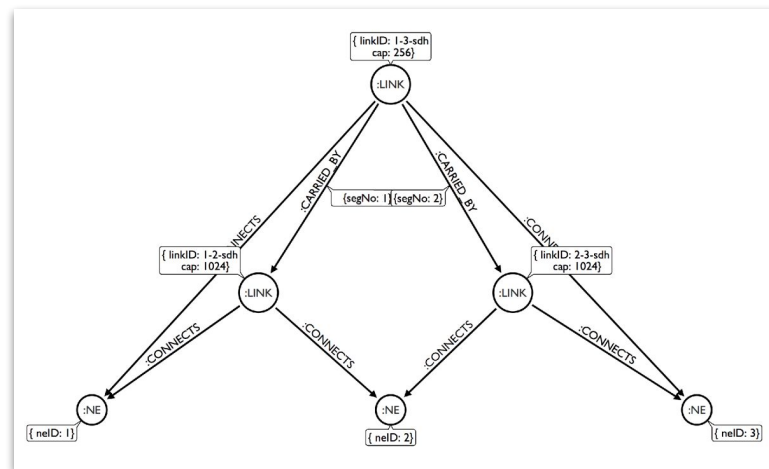
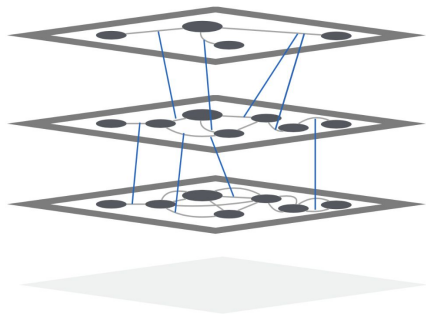
Shortest/most efficient path from A to B

Find diverse routes between A and B

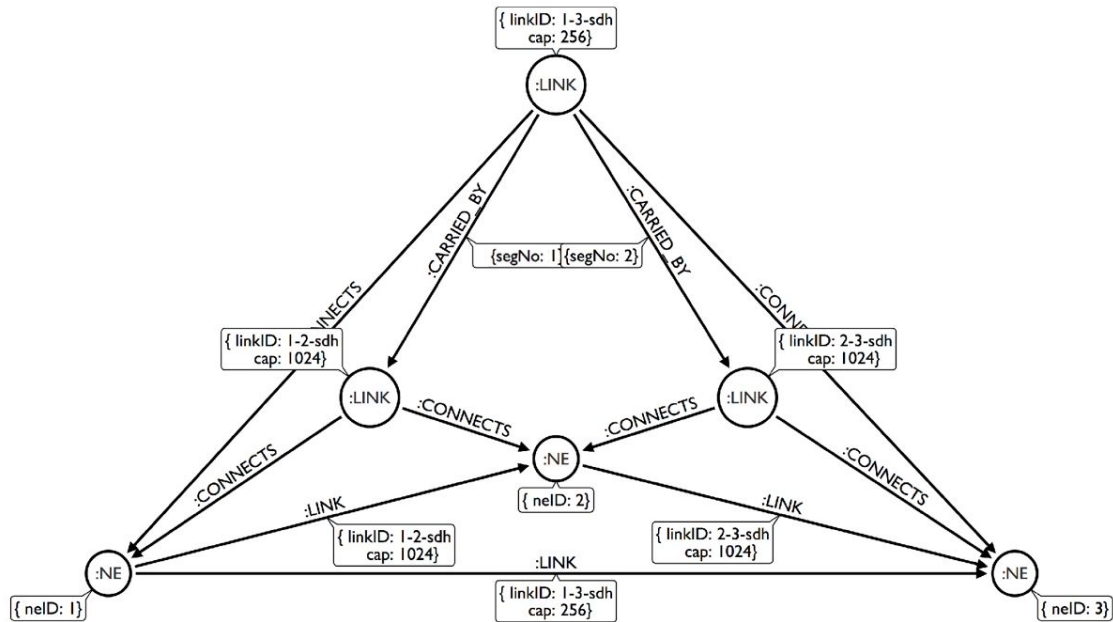


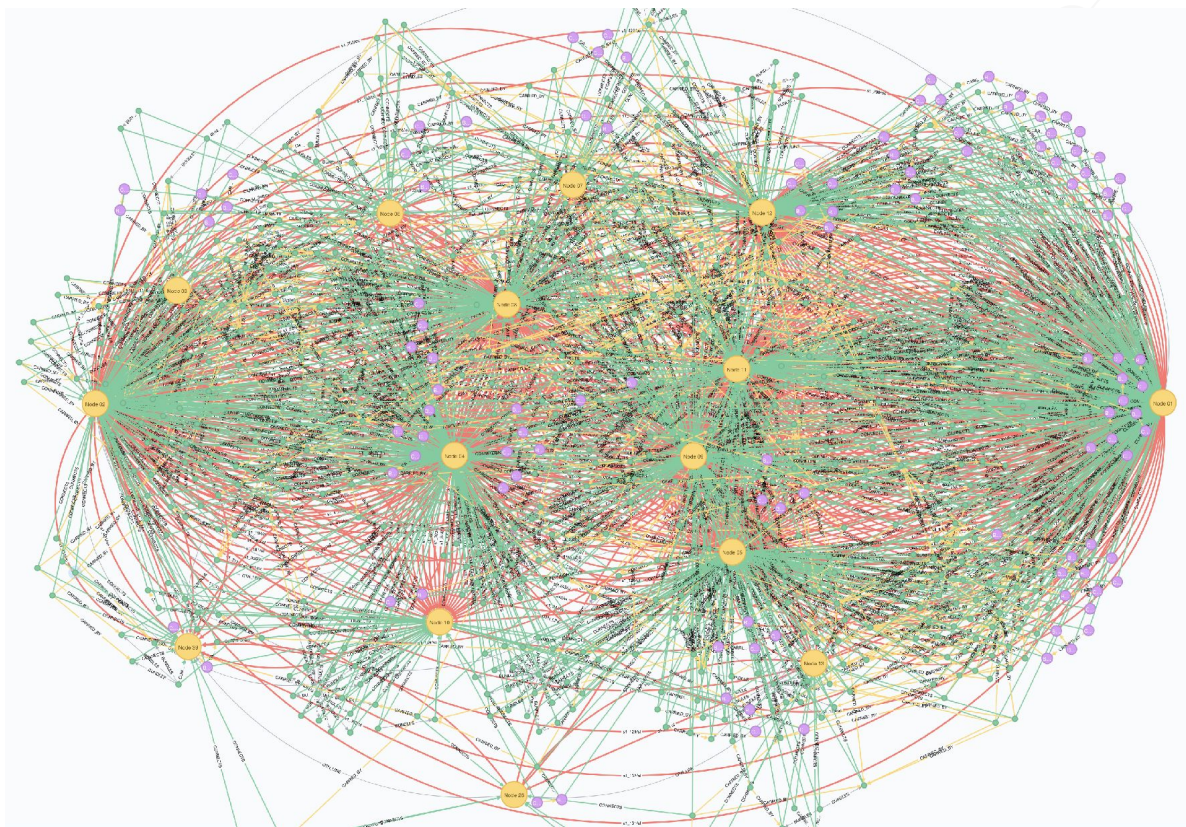
# The organization

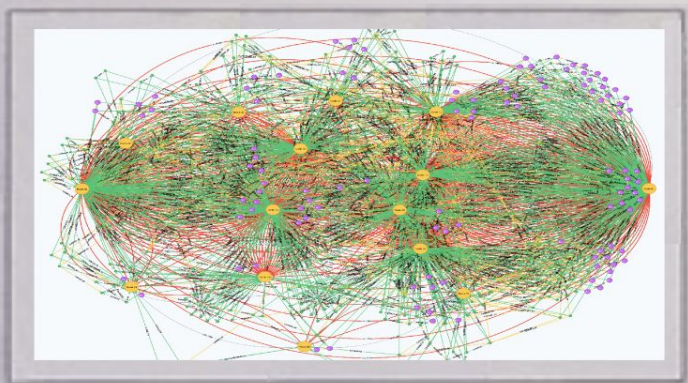
## Dependency oriented Model



# Dual Model







# The loaders and the “inferencing” on the model

```
CALL apoc.load.(json|xml|csv|...) ...  
    MERGE (aN:NE { neId: $aNeId}), (zN:NE { neId: $zNeId})  
    MERGE (aN) <-[:CONNECT]-(l:Link)-[:CONNECT]->(zN) WITH l
```

```
CALL nm.spof("123-sdh","317-sdh")  
CALL nm.disjoint(2,"n-1","n-7", <max-cost>)  
CALL nm.rca(["593-sdh","627-sdh","114-sdh",...])  
...
```



# Observation#3

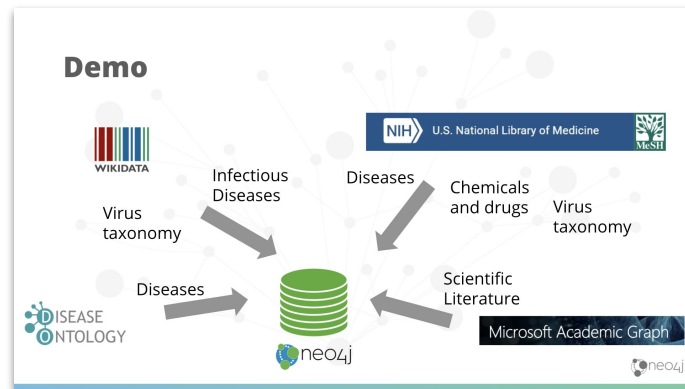
## KG construction is augmented by automation (Graph Algos, ML...)

- Taxonomies “learnt” from the data.
  - Similarity algorithms
- Formalised as overlay ontologies
- Used to drive query expansion, similarity analysis, recommendation...

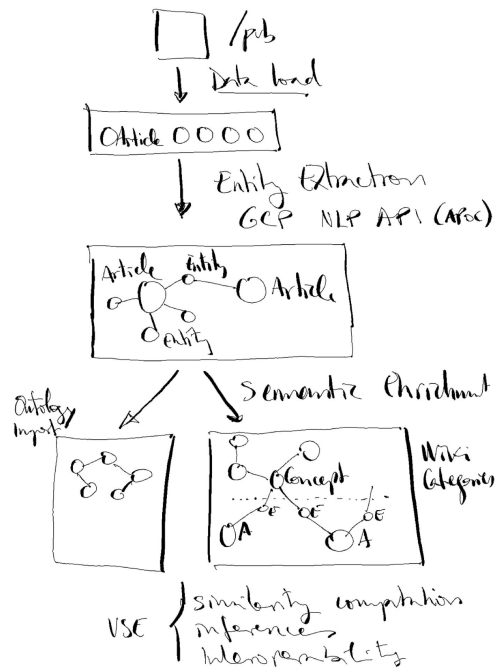
# Observation#4

## (private) KG construction use fragments of public KGs

- Wikidata
- Public ontologies



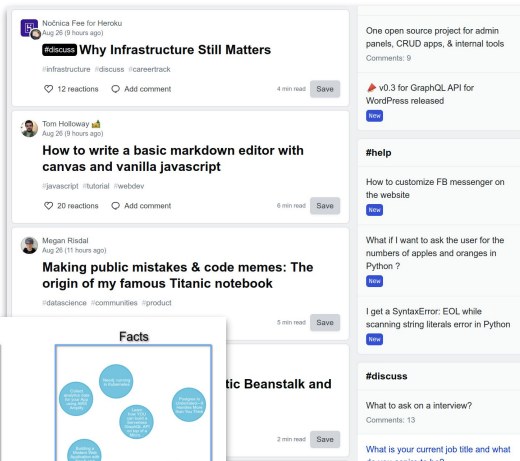
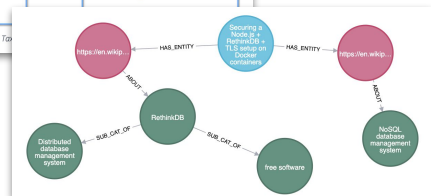
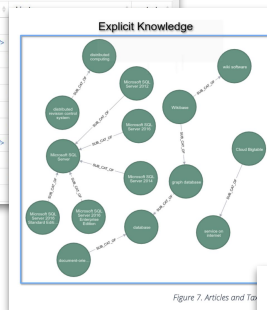
# Example: Tutorial- Building a KG using NLP and Ontologies



```

1 prefix neo: <neo4j://neo4j>
2 CONSTRUCT {
3   item a neo:Category ; neo:subCatOf ?parentItem .
4   item neo:name ?label .
5   ?parentItem a neo:Category ; neo:name ?parentLabel .
6   ?article a neo:WikipediaPage ; neo:about ?item .
7 }
8 WHERE
9   (
10    item (with {P11[with {P270}] id:Q4298814 .
11    item with {P11[with {P270}] ?parentItem .
12    item rdfs:label ?label .
13    filter(lang(?label) = "en")
14    ?parentItem rdfs:label ?parentLabel .
15    filter(lang(?parentLabel) = "en")
16  )
17  OPTIONAL {
18    ?article schema:about ?item ;
19    ?item schema:isPartOf "https://en.wikipedia.org/" .
20  }
21 )

```



<https://neo4j.com/developer/graph-data-science/build-knowledge-graph-nlp-ontologies/>

# n10s: RDF ~~vs~~ and PG



# Observation#5

**There's no 'standard' property graph serialisation format. Wait a minute...**

- Cloning a subgraph had to be done using scripting (cypher).
- RDF (kind of) did the job. RDF\* makes it a lot easier

# Observation#6

**There's a lot of valuable (RDF) graph data out there.**

- Do I really need to flatten it before I make it a graph again in my DB?
- RDF endpoints + SPARQL CONSTRUCT is your friend.

# n10s: The **bidirectional** conversion

Take a triple...

- object is a Literal

↳ a node with a property

(deal with datatypes, lang tags, multivals, etc.)

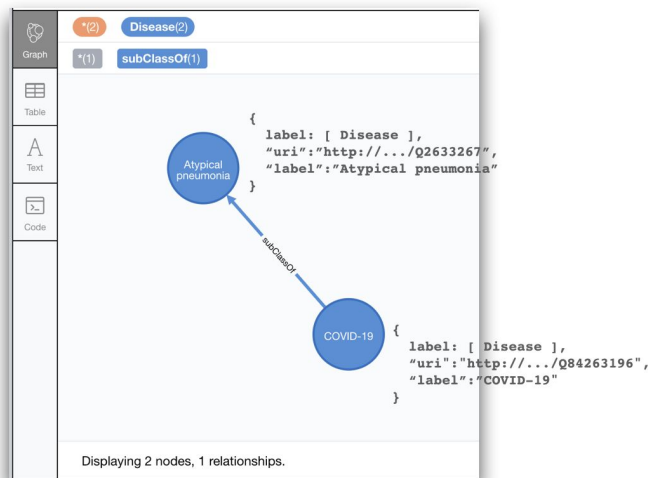
- object is a resource

↳ two nodes and a relationship

- predicate is *rdf:type*

↳ a node with a label (**optionally**)

```
<http://www.wikidata.org/entity/Q84263196> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.wikidata.org/category/Disease> .  
<http://www.wikidata.org/entity/Q84263196> <http://www.w3.org/2000/01/rdf-schema#label> "COVID-19" .  
<http://www.wikidata.org/entity/Q84263196> <http://www.w3.org/2000/01/rdf-schema#subClassOf> <http://www.wikidata.org/entity/Q2633267> .  
<http://www.wikidata.org/entity/Q2633267> <http://www.w3.org/1999/02/22-rdf-syntax-ns#type> <http://www.wikidata.org/category/Disease> .  
<http://www.wikidata.org/entity/Q2633267> <http://www.w3.org/2000/01/rdf-schema#label> "Atypical pneumonia" .
```





# Enter neosemantics: n10s

n10s is a plugin that enables the **use of RDF in Neo4j**

- **Import and store RDF data in Neo4j** in a **lossless** manner
- On-demand **export** property graph **data from Neo4j as RDF**
- model **mapping**
- Graph **data validation** based on SHACL shapes /constraints
- (limited) **Inferencing**



# https://neo4j.com/labs/neosemantics

neo4j/labs/neosemantics

Graph-Semantics: Import/Export RDF from Neo4j: Model mapping, inferencing and more... If you like it, please ★

448 commits · 8 branches · 0 packages · 11 releases · 2.0

neosemantics (n10s)

neosemantics is a plugin that enables the use of RDF in Neo4j. RDF is a W3C standard model for data interchange. Some key features of n10s are:

- Store RDF data in Neo4j in a business manner (imported RDF can subsequently be exported without losing a single triple in the process).
- On-demand export property graph data from Neo4j as RDF.
- Model validation based on the W3C SHACL language
- Import of Ontologies and Taxonomies in OWL, SKOS, RDFS...

Other features in NSMNTX include model mapping and inferencing on Neo4j graphs.

➔ User Manual and Blog ➔

➔ Check out the complete user manual with examples of use. ➔

➔ Blog on neosemantics (and more). ➔

Installation

You can either download a prebuilt jar from the releases page or build it from the source. If you prefer to build, check

neo4j Labs

Neo4j Labs / Neosemantics

## neosemantics (n10s): Neo4j RDF & Semantics toolkit

neosemantics (n10s) is a plugin that enables the use of RDF and its associated vocabularies like (OWL, RDFS, SKOS and others) in Neo4j. RDF is a W3C standard model for data interchange.

You can use n10s to build integrations with RDF-generating / RDF-consuming components. You can also use it to validate your graph against constraints expressed in SHACL or to run basic inferencing.

### Availability & Installation

neosemantics runs as an extension to your Neo4j database. Downloading the appropriate release for your Neo4j database into the plugins folder adds n10s to any Neo4j installation.

The n10s GraphApp will help you get started with neosemantics

### Functionality Includes

- Import/Export of RDF and RDF\* in multiple formats (Turtle, N-Triples, JSON-LD, RDF/XML, TriG and N-Quads, Turtle\*, TriG\*)
- Model mapping on import/export
- Import and export of Ontologies/Taxonomies in different vocabularies (OWLSKOS, RDFS)
- Graph validation based on SHACL constraints
- Basic inferencing

### Relevant Links

- Support: Neo4j Online Community
- Authors: Jesús Barrasa, Adam Cowley (GraphApp)
- Releases: <https://github.com/neo4j-labs/neosemantics/releases>
- Source: <https://github.com/neo4j-labs/neosemantics>
- Docs: <https://neo4j.com/docs/labs/nsmntx/current/>
- GraphApp: Install from your Neo4j Desktop via <https://install.graphapp.io>

Neosemantics  
Neo4j Labs

An interactive guide to the Neosemantics RDF toolkit

Install

neosemantics(n10s) User Guide

This is the user guide for Neosemantics 4.0, authored by the Neo4j Labs Team.

The guide covers the following areas:

- Chapter 1: Introduction — An introduction to the Neosemantics RDF toolkit for Neo4j.
- Chapter 2: Installation — Instructions for installing Neosemantics.
- Chapter 3: Configuring Neo4j in use — Setting up your Neo4j graph to work with RDF data.
- Chapter 4: Importing RDF data — A detailed guide to importing RDF data.
- Chapter 5: Inferencing RDF — A detailed guide to inferencing RDF data.
- Chapter 6: Importing RDF data — A detailed guide to importing RDF data.
- Chapter 7: Inferencing RDF data — A detailed guide to inferencing RDF data before we import it into Neo4j.
- Chapter 8: Importing RDF data — A detailed guide to importing RDF data.
- Chapter 9: Mapping graph results — A detailed guide to mapping transformations to RDF as it's imported into Neo4j.
- Chapter 10: Inferencing graph results — A detailed guide to inferencing and reasoning.
- Chapter 11: Inferencing/Reasoning — A detailed guide to inferencing and reasoning.
- Chapter 12: Neosemantics Repository — An overview of all procedures and functions in the library.
- Chapter 13: Projects using Neosemantics — A list of projects using n10s.
- Appendix A: Migrating from Neosemantics 3.x — A guide for Neosemantics 3.x users.

Videos & Tutorials

- 36. Build a Knowledge Graph
- Using Graph Studio to Import RDF Data from a Graph
- FIBO in Neo4j: Apply...
- Neosemantics (n10s) ...
- Ontologies in Neo4j: S...
- How Semantic is Your ...

## n10s in some key figures

5 Yrs

First commit in April  
2016. **14 contributors**  
50% are Neo4j staff.

24.8K

**Downloads** of  
neosemantics as of  
June 5th 2021.

526★

Top 20 actively  
maintained RDF  
projects on GitHub

# n10s community

The screenshot shows the Neo4j community forum interface. At the top, there's a header with the Neo4j logo and a navigation bar with links like 'Discussions', 'Do you know the answer?', 'Developer Guides', 'Graph Academy', 'Developer Blog', and 'Friendly Chat'. Below this, there's a section for 'Integrations' with a sub-section for 'Linked Data, RDF, Ontology'. The main content area displays a list of topics under the heading 'Topic'. The first topic is 'About the Linked Data, RDF, Ontology category' with a description and a link to read more. Other topics include 'N10s inferences union/intersections, complements', 'Support for running SPARQL into graph', 'Export procedure that returns serialized RDF', and 'How to check if graph is empty?'. Each topic entry shows the number of replies, views, and the time since it was posted.

| Topic  | Replies | Views | Activity |
|--|---------|-------|----------|
| * About the Linked Data, RDF, Ontology category  | 0       | 573   | Sep '19  |
| N10s inferences union/intersections, complements | 2       | 23    | 3d       |
| Support for running SPARQL into graph            | 0       | 10    | 8d       |
| Export procedure that returns serialized RDF     | 1       | 35    | 11d      |
| How to check if graph is empty?                  | 2       | 35    | 13d      |

<https://community.neo4j.com/c/integrations/linked-data-rdf-ontology/162>

The screenshot shows the GitHub repository page for 'neo4j-labs/neosemantics'. The repository has 37 issues, 11 pull requests, and 8 actions. The 'Issues' tab is selected, showing a list of open issues. The issues are filtered by 'is:issue is:open'. The list includes issues like 'Schema export lacks DatatypeProperties', 'Schema export flaw', 'Is it possible to import a directory', 'Adding non mapped data to imported rdf data', 'Documentation 'showOnlyMapped'', 'SHACL datatype restriction problem', 'Will there eventually be neo4j support for n10s?', 'SHACL validation before deletion of nodes', 'request n10s.onto.import.fetch require a relative path and relative to the Neo4j import directory', 'Load class annotations in ontology import', and 'custom prefix on existing neo4j data'. Each issue entry shows the issue number, title, and the time since it was opened.

| Issue   | Opened                  | By        |
|---|-------------------------|-----------|
| 71 Open Schema export lacks DatatypeProperties  | #224 opened 19 days ago | thollebig |
| Schema export flaw  | #223 opened 24 days ago | thollebig |
| Is it possible to import a directory  | #219 opened on 15 Apr   | alilik    |
| Adding non mapped data to imported rdf data   | #215 opened on 10 Mar   | mklopp    |
| Documentation 'showOnlyMapped'  | #214 opened on 10 Mar   | mklopp    |
| SHACL datatype restriction problem  | #213 opened on 3 Mar    | palandiom |
| Will there eventually be neo4j support for n10s?  | #212 opened on 1 Mar    | sbailin   |
| SHACL validation before deletion of nodes   | #211 opened on 25 Feb   | palandiom |
| request n10s.onto.import.fetch require a relative path and relative to the Neo4j import directory | #208 opened on 17 Feb   | Dcanzano  |
| Load class annotations in ontology import   | #207 opened on 16 Feb   | tanavarro |
| custom prefix on existing neo4j data  | #205 opened on 21 Jan   | ssilwal29 |

<https://github.com/neo4j-labs/neosemantics/issues>

## N10s satellite projects: rdflib-neo4j

```
import rdflib

# create a neo4j backed Graph
g = rdflib.Graph(store='Neo4j')

# set the configuration to connect to your Neo4j DB
theconfig = {'uri': "neo4j://localhost:7687", 'database': 'rdfstore', 'auth': {'user': "neo4j", 'password': "password"}}

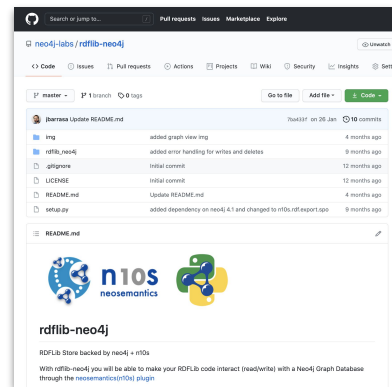
g.open(theconfig, create = False)

g.load("https://raw.githubusercontent.com/jbarrasa/datasets/master/rdf/music.nt", format="nt")

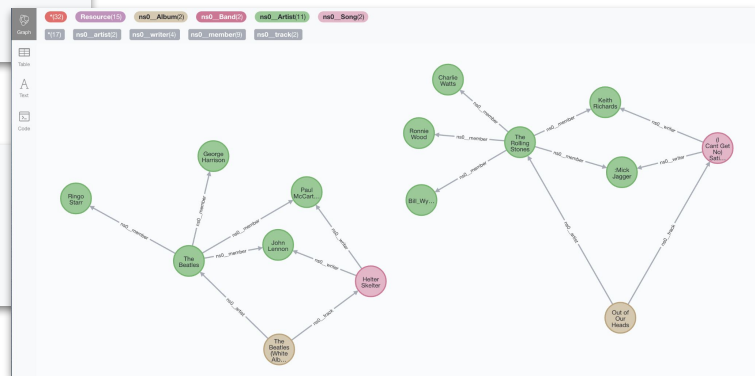
# For each foaf:Person in the store, print out their mbox property's value.
print("--- printing band's names ---")
for band in g.subjects(rdflib.RDF.type, rdflib.URIRef("http://neo4j.com/voc/music#Band")):
    for bandName in g.objects(band, rdflib.URIRef("http://neo4j.com/voc/music#name")):
        print(bandName)
```

```
MUSIC = rdflib.Namespace("http://neo4j.com/voc/music#")
fm = rdflib.URIRef("http://neo4j.com/indiv#Fleetwood_Mac")

g.add((fm, rdflib.RDF.type, MUSIC.Band))
g.add((fm, MUSIC.name, rdflib.Literal("Fleetwood Mac")))
```



## An RDFLib Store backed by neo4j + n10s



# Observation#7

## **We are convinced of the value of making semantics explicit**

- Automation is central to the creation of ontologies. Without it it's close to pointless.
- Making the data smarter

# Observation#8

**But the sad reality is that the understanding of semantics outside the academic community is very poor**

- People get SHACL validations... but not OWL inference
- Maybe that's the right path?



# The SHACL approach

- SHACL core
- DASH Constraints (<http://datashapes.org/constraints.html>)
  - dash:coExistsWith**
  - dash:subSetOf**
- Defined by TopQuadrant, supported by Ontotext

# Takeaways

*Updated from the version presented  
based on Christophe Debruyne's  
question ;-)*

- Let's keep making data smarter together. Your contribution is welcome
  - Successful precedents: eccenca
- KG adoption in industry is in its infancy, we have a great future ahead of us.
- Call to action: be curious and spin up a Neo4j sandbox(\*) and do RDF!

## “It's all graphs!”

Juan Sequeda

(\*) <https://sandbox.neo4j.com/>



# Thank you!

