Knowledge Graphs 2021: The great (graph) convergence

Jesús Barrasa
Neo4j
Jesús Barrasa
Director of SE EMEA
@Barrasadv / 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
The Great graph Convergence
converge  verb

Save Word

con-verge  |  \kən-\vər\ |  
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
Huge interest in graph ML
2

Renewed interest in RDF/ Semantics

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
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
ppl://ann is a person
ppl://ann user ID is @ann
ppl://ann name is Ann Smith
ppl://dan likes ppl://ann
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
Ann, app_user_ID: @ann, uri: ppl://ann

PG

LIKES { date: 02/03/17 }
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](http://dl.acm.org/citation.cfm?id=2567973)
- Multivalued properties are arrays/collections
Context: How do property graphs relate to RDF graphs?

**SPARQL**

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

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

**Cypher**

```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

<table>
<thead>
<tr>
<th>def#1</th>
<th>def#2</th>
<th>def#3</th>
<th>...</th>
<th>def#183</th>
</tr>
</thead>
<tbody>
<tr>
<td>Persisted in a DB</td>
<td>Described as RDF</td>
<td>Integrates disparate data</td>
<td>Models master data</td>
<td>Explicit &amp; formal semantics</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

...
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

```
[linkID: 1-3-sdh  
cap: 256]

[segNo: 1] [segNo: 2]

[linkID: 1-2-sdh  
cap: 1024]

[linkID: 2-3-sdh  
cap: 1024]

[netID: 1]

[linkID: 1-2-sdh  
cap: 1024]

[netID: 2]

[linkID: 1-3-sdh  
cap: 256]

[netID: 3]
```

Neo4j, Inc. All rights reserved 2021
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

n10s: RDF vs and PG
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)

https://jbarrasa.com/2016/06/07/importing-rdf-data-into-neo4j/
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
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

https://community.neo4j.com/c/integrations/linked-data-rdf-ontology/1

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

Neo4j, Inc. All rights reserved 2021
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", 'pwd': ""}}
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")))
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
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](http://datashapes.org/constraints.html))
  - dash:coExistsWith
  - dash:subSetOf

• Defined by TopQuadrant, supported by Ontotext
Takeaways

- 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!