Edge-labelled graphs and property graphs - to the user, more similar than different

Conference or Workshop Item

How to cite:
Warren, Paul and Mulholland, Paul (2022). Edge-labelled graphs and property graphs - to the user, more similar than different. In: KGSWC 2022: Knowledge Graphs and Semantic Web. 4th Iberoamerican Conference and third Indo-American Conference, 21-23 Nov 2022, Madrid, Spain.

For guidance on citations see FAQs.

https://creativecommons.org/licenses/by-nc-nd/4.0/

Version: Supplementary Material
Edge-labelled graphs and property graphs

- to the user, more similar than different

Paul Warren and Paul Mulholland
Knowledge Media Institute, The Open University, U.K.
{paul.warren, paul.mulholland}@open.ac.uk
Knowledge graphs

Two paradigms

• Edge-labelled graphs
  – based on concept of triple
  – standardized by W3C (RDF and SPARQL)
  – considerable theoretical understanding

• Property graphs
  – edges and nodes can have properties with literal values
  – originating from proprietary standards
    • now openCypher and draft GQL standard
  – widely used in commercial applications
An empirical study

• Objectives
  – compare ease of use
  – identify modelling preferences
    • and whether these differed between paradigms
  – understand difficulties in querying

• Using
  – RDF-star and SPARQL-star (Blazegraph implementation)
    • extensions to RDF and SPARQL to facilitate reification
  – Cypher (Neo4J)
Overview of study

• Between-participants
  – RDF*/SPARQL*  N = 26
  – Cypher  N = 18
• RDF*/SPARQL* participants more relevant experience than Cypher participants
  – controlled for in statistical analysis
• Modelling questions
  – asked participants to rank models
• Querying questions
  – asked participants to identify whether queries were correct or incorrect
• Five sections
  – modelling question followed by querying questions
Nodes versus literals

Sophie works for CreativeCo, which is located in London. Brian works for BigCo, which is located in York. Diane works for AcmeCo, which is located in York.

**Required queries:** Where is the company located for which Brian works? Who works for a company located in the same town as Brian’s company?

**Cypher models**

1. CREATE ((name: ‘Sophie’)) <- [worksFor] -> ((name: ‘CreativeCo’, located: ‘London’)).
   ((name: ‘Brian’)) <- [worksFor] -> ((name: ‘BigCo’, located: ‘York’)).
   ((name: ‘Diane’)) <- [worksFor] -> ((name: ‘AcmeCo’, located: ‘York’)).

2. CREATE ((name: ‘Sophie’)) <- [worksFor] -> ((name: ‘CreativeCo’)) -[(located)] -> ((name: ‘London’)).
   ((name: ‘Brian’)) <- [worksFor] -> ((name: ‘BigCo’)) -[(located)] -> ((name: ‘York’)).
   ((name: ‘Diane’)) <- [worksFor] -> ((name: ‘AcmeCo’)) -[(located)] -> ((name: ‘York’)).

**RDF models**

1. Sophie :worksFor :CreativeCo.
   CreativeCo :located ‘London’.
   Brian :worksFor :BigCo.
   BigCo :located ‘York’.
   Diane :worksFor :AcmeCo.
   AcmeCo :located ‘York’.

2. Sophie :worksFor :CreativeCo.
   CreativeCo :located ‘London’.
   Brian :worksFor :BigCo.
   BigCo :located ‘York’.
   Diane :worksFor :AcmeCo.
   AcmeCo :located ‘York’.

Preference to represent cities as nodes - but significantly less for Cypher than RDF

N.B. lines represent 95% confidence intervals
Class hierarchies are straightforward in RDF
- use predicate to represent subsumption

In Cypher, labels used for classes
- no natural way of creating hierarchies
- need to represent labels as strings

Compare classes as strings (e.g. Cypher labels) with classes as nodes

Preference to represent classes as nodes
- even in Cypher (i.e. rather than labels)
- although significantly less for Cypher than RDF

‘Neo4j generally pushes for the use of labels as “types”, but the path query for recursive subgroups is going to be awkward ...’
Querying class hierarchies

Query: What are the names of the individuals which are mammals?

classes as strings (models 1)

Cypher queries

| Query                                                                 | Result |
|----------------------------------------------------------------------|--------|
| MATCH (x), [y: {subGroupOf*} -> [[id: 'Mammal']]] WHERE x.id IN label(y) RETURN x.name | ✔      |
| MATCH (x) {subGroupOf*} -> [[id: 'Mammal']]] RETURN x.name             | ✗      |
| MATCH (x) { WHERE 'Mammal' IN label(x) RETURN x.name                   | ✗      |

SPARQL queries

| Query                                                                 | Result |
|----------------------------------------------------------------------|--------|
| SELECT ?name WHERE ( ?name :typeOf ?grouppname . ?grouppname :subGroupOf :Mammal . FILTER(COUNTSTR(?grouppname) > 0)) | ✔      |
| SELECT ?name WHERE ( ?name :subGroupOf :Mammal )                      | ✗      |
| SELECT ?name WHERE ( ?name :typeOf :Mammal )                          | ✗      |

classes as nodes (models 2)

Cypher queries

| Query                                                                 | Result |
|----------------------------------------------------------------------|--------|
| MATCH (x) {subGroupOf*} -> [[id: 'Mammal']]] RETURN x.name             | ✗      |
| MATCH (x) { WHERE 'Mammal' IN label(x) RETURN x.name                   | ✔      |

SPARQL queries

| Query                                                                 | Result |
|----------------------------------------------------------------------|--------|
| SELECT ?name WHERE ( ?name :subGroupOf :Mammal )                      | ✗      |
| SELECT ?name WHERE ( ?name :typeOf :Mammal )                          | ✗      |
| SELECT ?name WHERE ( ?name :typeOf :subGroupOf :Mammal )              | ✔      |

mean accuracy

2.43 / 3 2.89 / 3

Querying question answered significantly more accuracy for model 2, representing classes as nodes, than model 1, using string labels
- consistent with participants’ preference for model 2
- no difference in performance between the paradigms
Reverse predicates

Adrian works as a lawyer for TransportCo. In addition, he works as an advisor for ArtsCo. Clare works as an accountant for TransportCo.

Query: For which companies does Adrian work and what role does he have in each company?

Cypher model

```
(1) CREATE (a {name: 'Adrian'})
    (a) :worksFor {role: 'lawyer'}-> (b {name: 'TransportCo'}),
    (a) :worksFor {role: 'advisor'}-> (b {name: 'ArtsCo'}),
    (a) :worksFor {role: 'accountant'}-> (b)

(3) MATCH (m-[e:worksFor]->(n{name: 'Adrian'}))
    RETURN n.name, e.role
```

RDF* model

```
(1) <<:Adrian:worksFor :TransportCo>> :role 'lawyer'.
<<:Adrian:worksFor :ArtsCo>> :role 'advisor',
<<:Clare:worksFor :TransportCo>> :role 'accountant'.

(3) SELECT ?company ?role
    WHERE {
        ?role^:role<<:Adrian:worksFor ?company>>
    }
```

Cypher question recognized as correct significantly more frequently than SPARQL* query

- reverse arrow may be more intuitive than ^
- but this effect not observed in other, possibly less complex, queries
- needs more investigation
Results

• Where models analogous, similar preferences in the two paradigms
  – even where Cypher ‘style’ might encourage a different preference
• Preference for representing class hierarchies as connected nodes
  – rather than labels needing string representation
  – suggests extending Cypher to enable hierarchies of node labels; and edge types
    • to enable query-time processing (c.f. rdfs:subClassOf and rdfs:subPropertyOf
• Little difference in identifying correct / incorrect queries
  – suggests interpretability of two paradigms similar
    • possibility of preference for Cypher <- over SPARQL ^
  – study with timing information might reveal differences
Conclusions

• Each paradigm can learn from the other
  – node and edge properties may enable rich structures to be created relatively easily
  – but edge-labelled graphs (RDF) enables query-time reasoning

• Consider how closely they can be brought together
  – see Hartig ‘Reconciliation of RDF* and property graphs’:
    https://arxiv.org/abs/1409.3288
  – Stardog offers Property Graph syntax to describe RDF*
    https://www.stardog.com/blog/property-graphs-meet-stardog/
    • enables property values to be nodes as well as literals