Graph? Yes! Which one? Help!

Ora Lassila, Michael Schmidt, Brad Bebee, Dave Bechberger, Willem Broekema, Ankesh Khandelwal, Kelvin Lawrence, Ronak Sharda, and Bryan Thompson

Amazon Web Services

“Squaring the Circle on Graphs” @ SEMANTiCS 2021
September 2021
Outline

Introduction: graph interoperability, RDF, LPGs, RDF-star, etc.

1G Model

Interoperability challenges

Next steps
Graph interoperability

Amazon Neptune
- managed, cloud-based graph database service
- supports RDF (SPARQL) and LPG (Gremlin & openCypher)

User has to choose either RDF or LPG
- this choice also determines which query languages are available
- the choice is not always easy, and is hard to reverse later

RDF vs. LPG
- RDF offers a formal model, LPG not so much
- RDF is seen as “academic”, and developers tend to prefer LPG
- different strengths and weaknesses
Graph interoperability

What if we did not have to choose between RDF and LPG?

What if we could use Gremlin over RDF, or SPARQL over LPG?

Interoperability: single graph (meta)model, free use of any query language

• we are not interested in “qualified” interoperability where one metamodel is *implemented* using the other

RDF-star is a step towards having LPG features in RDF
1G model ("one graph to rule them all")

Basic representational unit: a **statement**

![Diagram](src → label → value → sid)

Every statement has a unique identifier *sid*

In RDF, statements map to SPO triples with a unique identifier

For LPG, we get a uniform mechanism for both edges and properties

None of this implies a particular implementation
Example

“Alice knows Bob since 2020”

- Alice \(\xrightarrow{\text{knows}}\) Bob \(\text{sid}_1\)
- Alice \(\xrightarrow{\text{name}}\) “Alice” \(\text{sid}_2\)
- Bob \(\xrightarrow{\text{name}}\) “Bob” \(\text{sid}_3\)
- \(\text{sid}_1 \xrightarrow{\text{since}} 2020 \text{sid}_4\)
Example analyzed

Possible representations in

• RDF – without the “since” statement, unless we use reification
• RDF-star – that’s what RDF-star is for
• LPG – “name” statements are vertex properties, and the “since” statement is an edge property

1G mapped to “lower-dimensional” models

We do not (yet) provide a full formalization of 1G, but offer it more as a framework for discussion

:Alice :knows :Bob .
:Alice :name “Alice” .
:Bob :name “Bob” .


(name: “Alice”)

(name: “Bob”)

since: 2020
Interoperability challenges

1) Edge properties, multiple edge instances, reification
2) Triples vs. graph abstraction
3) Datatype alignment
4) Graph partitioning
5) Graph merging, external identifiers
6) Lack of formal foundation
7) Update query semantics
Edge properties, multiple edge instances, reification

\[ \text{Alice} \xrightarrow{\text{knows}} \text{Bob} \quad \text{sid}_1 \]

\[ \text{Alice} \xrightarrow{\text{knows}} \text{Bob} \quad \text{sid}_2 \]

\[ \text{sid}_1 \xrightarrow{\text{since}} 2020 \quad \text{sid}_3 \]

\[ \text{sid}_2 \xrightarrow{\text{since}} 2021 \quad \text{sid}_5 \]

\[ \text{sid}_1 \xrightarrow{\text{statedBy}} \text{NYTimes} \quad \text{sid}_4 \]

\[ \text{sid}_2 \xrightarrow{\text{statedBy}} \text{Guardian} \quad \text{sid}_6 \]
Edge properties, multiple edge instances, reification

Alice knows Bob \(\text{sid}_1\)  

Alice knows Bob \(\text{sid}_2\)  

\(\text{sid}_1\) since 2020 \(\text{sid}_3\)  

\(\text{sid}_2\) since 2021 \(\text{sid}_5\)  

\(\text{sid}_3\) statedBy NYTimes \(\text{sid}_4\)  

\(\text{sid}_5\) statedBy Guardian \(\text{sid}_6\)
Triples vs. graph abstraction

RDF:
- vertices “do not exist” (infinite space of identifiers, no internal structure)
- `rdf:type`

LPG:
- vertices and edges are structured objects
- labels
- sets of vertices and edges are disjoint
Datatype alignment

RDF:
• compound datatypes are constructed using graph elements (e.g., rdf:List)
• we could use the literal datatype mechanism to introduce compound datatypes
  e.g., “[1, 2, 3]”^^onegraph:List

LPG:
• compound datatypes are “delegated” to implementation language(s)
• no formal foundation
Graph partitioning

RDF:
• named graphs
• triples extended to quads, no semantic theory

LPG:
• no standard mechanism
• edge properties could be used, but what about vertices?

1G:

In RDF, named graphs are sometimes used as containers
• this prevents their use for graph partitioning (for versioning, maintenance, etc.)

LPG users would like containers too…

1G gives you both
Graph merging, external identifiers

RDF:
- strong external identifiers
- formally specified merge semantics

LPG:
- no standard mechanism
- compounded by the “multiple edge instances” dilemma
Lack of formal foundation

LPG:

- no formal semantics for the graph metamodel (except *post hoc*)
- no formal semantics or query algebra for query languages
- semantics often defined either via example or by an implementation
- ambiguous
Update query semantics

Read queries can be mapped from 1G to “lower-dimensional” models

Update queries make this much harder

• “invisible” dimensions
• ambiguity

Example:
• should (in SPARQL) deleting 
  ：Alice :knows :Bob
  delete both $sid_1$ and $sid_2$?
• what about the assertions about $sid_1$ (and $sid_2$)…?
Next steps

Users prioritize standards and interoperability

Data integration is seen as the dominant use case

Interoperability is something the entire graph industry needs to address to make adoption easier

Is the idea of “compatibility subsets” viable…?

We now seek support from the broader community to look into these issues
Thank you!

ora@amazon.com
Twitter: @oralassila