Toward Easier RDF

W3C Graph Workshop, Berlin, March 4-6, 2019

David Booth, Yosemite Project <david@dbooth.org>
Christopher G Chute, Johns Hopkins University <chute@jhu.edu>
Hugh Glaser, Seme4 Ltd. <hugh@glasers.org>
Harold Solbrig, Johns Hopkins University <solbrig@jhu.edu>

Your background in the main topic areas of the workshop
We have been working with RDF and teaching it for over 10-15 years, during which we have collected various observations about the barriers to its adoption.

Which topic you would like to lead discussion on
Making RDF easier to use.

Links to related supporting resources

- Github site for discussing Easier RDF: https://github.com/w3c/EasierRDF
- Public email discussion about making RDF easier: https://lists.w3.org/Archives/Public/semantic-web/2018Nov/0036.html
- Defining N-ary Relations on the Semantic Web: https://www.w3.org/TR/swbp-n-aryRelations/

Position statement

The value of the RDF has been demonstrated in a wide variety of applications over its ~20 year lifespan. The abstract model that underlies RDF makes a lot of sense. RDF offers globally unique and potentially resolvable identifiers -- URIs/IRIs; it works well as a universal information representation, allowing any sort of source data to be represented; it has a solid foundation in formal logic and supports automated inference; and it supports distributed extensibility.

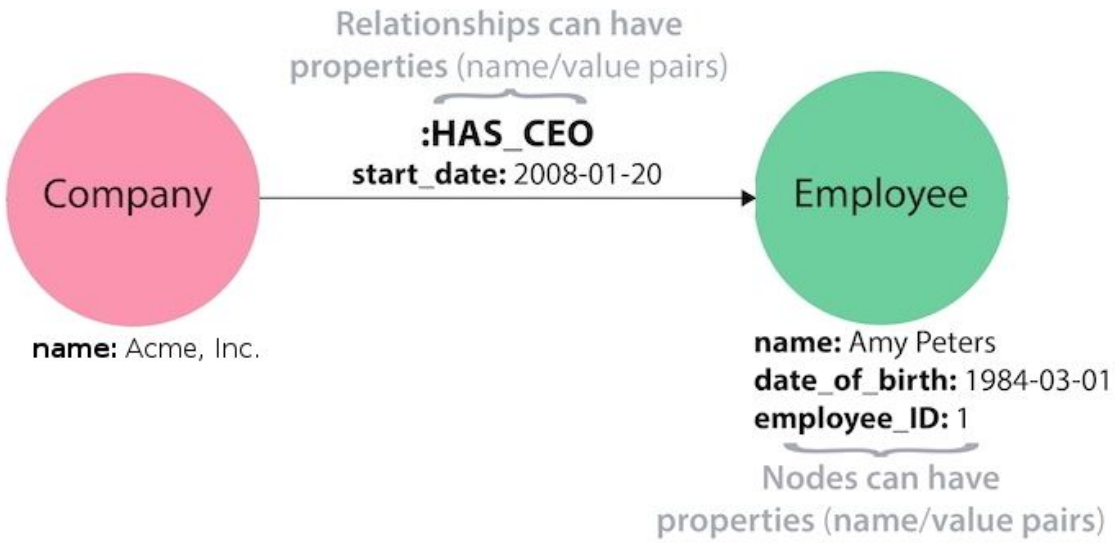
On the other hand, RDF, *as implemented today*, has proven to be too hard to use by *average* software developers -- middle 33% of ability -- who are not RDF experts and have other priorities and deadlines. The current implementations of the underlying abstraction behind RDF has inhibited its adoption. We have collected observations about some of the things that makes RDF implementations hard to use. We recently launched a public discussion and github site (see above links) to collect ideas for making the RDF ecosystem easier. Areas of concern include (among others):

- Support for n-ary relations and property graphs
- Support for SPARQL-friendly lists
- Problems around blank node usage
- RDF canonicalization

Use of RDF does not only involve the RDF standard. It involves the entire RDF ecosystem of related standards, tools and documentation. Some other more recently popular graph databases have a considerably lower entry barrier than RDF. Although much of the lower entry barrier can be explained by more polished, integrated tooling and documentation, their data representations differ from RDF in ways that also contribute to ease-of-use. We would like to both learn from their successes, and make RDF as friendly as possible for users of these other graph representations. Standardization of graph data in RDF would provide not only an opportunity to bridge between graph representations, but also to make the RDF ecosystem more broad-based and easier to use.

Example property graph relation

Consider this example property graph relation (adapted from the Neo4J introduction https://neo4j.com/developer/graph-database/):



A :HAS_CEO relation is expressed between a subject Company :acme and an object Employee Amy Peters, and a property (start_date: 2008-01-20) is attached to the :HAS_CEO relation. RDF has no native support for attaching properties to subject-object relations, but the effect can be achieved in RDF in a number of ways. The examples below illustrate some of the possibilities. They are not intended as concrete proposals, but as examples for discussion.

As an n-ary relation

One way to represent the above example in RDF would be to follow an n-ary relation pattern, in which the properties of a subject-object relation become additional participants in an n-ary relation. One example (in Turtle/TRIG syntax, namespaces omitted for brevity):

```
:acme a :Company ;
  :name "Acme, Inc." .
:amyPeters a :Employee ;
  :name "Amy Peters" ;
  :date_of_birth: "1984-03-01^^xsd:date" ;
  :employee_ID: 1 .

:acme :HAS_CEO [         # :HAS_CEO acts like an N-ary relation
    a :N_ary_relation ;
    rdf:value :amyPeters ;
    :start_date: "2008-01-20" ] .
```

As a named graph

Another approach would be to use a named graph (in TriG syntax, namespaces omitted for brevity):

```
:acme a :Company ;
  :name "Acme, Inc." .
:amyPeters a :Employee ;
  :name "Amy Peters" ;
  :date_of_birth: "1984-03-01^^xsd:date" ;
  :employee_ID: 1 .

:g1 { :acme :HAS_CEO :amyPeters . }  # :g1 is a named graph
:g1 a :Labeled_property ;
  :start_date: "2008-01-20" .
```

As RDF reification

A third approach would be to use RDF reification:

```
:acme a :Company ;
  :name "Acme, Inc." .
:amyPeters a :Employee ;
  :name "Amy Peters" ;
  :date_of_birth: "1984-03-01^^xsd:date" ;
  :employee_ID: 1 .

:s1 a :Labeled_property ;   # :s1 is the reified relation
    rdf:subject :acme ;
    rdf:predicate :HAS_CEO ;
    rdf:object :amyPeters ;
    :start_date: "2008-01-20" .
```

Many other variations are possible. If one such idiom were adopted as a standard way to represent property graph data, then:

- property graphs could be fully compatible with RDF;
- RDF newcomers would have clear guidance about how to represent n-ary relations and labeled properties -- especially those who are already familiar with property graphs;
- RDF tools could start recognizing and supporting the adopted idiom; and
- eventually a higher-level RDF syntax could be developed that directly supports the adopted idiom and compiles it down to the standard RDF 1.1 model, further simplifying RDF usage.

This is one example of why we believe efforts to standardize graph data using RDF could help simplify RDF usage, in addition to bridging between graph representations.