ontotext

ONTOTEXT ANSWER:

You probably saw our performance comparison on the topic and are wondering if more performant is always better. That question has the classic answer: “It depends”.

Let’s look at the three main modes of reification and what they offer.

First, a short introduction to the concept itself. RDF is based on triples. As an abstract knowledge representation model, it does not differentiate between data and metadata. Each edge is the connection between two concepts. In a property graph, the connection is a concept in itself. Therefore, if you want to model quadruples or more complex relationships, which store both the data (triple) and its metadata as a single datapoint, you have to normalize the connection somehow. This is where RDF graphs and property graphs differ – property graphs natively support more complex relationships. Reification is how you rectify this deficiency.

Standard Reification and N-ary relations

Standard reification and N-ary relationships are easy to conceptualize. Suppose you have a triple such as:

```
:a :hasSpouse :b
```

:hasSpouse is the relationship – or edge – here. If you want to add metadata to it, you turn the whole triple into a relationship.

```
:Relationship1 a :Relationship ;
    :partner1 :a ;
    :partner2 :b ;
    :startYear “1999” .
```

That works – and it would work on any triplestore, even if it isn’t SPARQL 1.1 compliant, but only implements the older 1.0 standard. But now you have extra triples. Furthermore, you cannot perform inference on this data out of the box, as it doesn’t fit standard inference models.

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

Named Graphs

SPARQL 1.1 introduces named graphs (or contexts), which turn triples into quintuples. We can model the relationship between **a** and **b** as a separate graph.

```
:a :hasSpouse :b :Relationship1 .
:Relationship1 :startYear “1999” .
```

That’s much simpler than standard reification. And you can perform inference on it out of the box. However, GraphDB isn’t optimized for named graph storage, so you should turn the “context index” on. And, your performance would be roughly the same as with standard reification. Furthermore, this may lead to confusion if your data uses named graphs anyway.

RDF-star and SPARQ-star

RDF-star and SPARQL-star are extensions on the SPARQL 1.1 standard. As extensions, they are not supported by the majority of the RDF databases. However, they offer much terser expressions for the complex relationships in a database:

```
<< :a :hasSpouse :b >> :startYear “1999” .
# Or, even more complex, perhaps a and b remarried after a while and
had different children in the two marriages...
<< << << :a :hasSpouse :b >> :startYear “1999” >> :endYear “2005” >>
:hasChild :c .
<< << :a :hasSpouse :b >> :startYear “2007” >> :hasChild :d .
```

RDF-star offers full SPARQL support on its nested triples and has substantially better performance than the other methods of reification. It is both faster and takes up less storage. The drawback is that you cannot have inference, unless if you also materialize the triples as “plain relationships”, which negates some of the performance benefits.

```
<< :a :hasSpouse :b >> :startYear “1999” .
:a :hasSpouse :b .
```

RDF star with inference

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com

ontotext

Conclusion

Ultimately, RDF-star is the better choice when performance is a major factor and inference isn't, or when the complexity of the embeddings is high. Named graphs are good for inference and in cases when they are not used by the database for other purposes. Standard reification is the fallback option that neither performs well, nor works well with inference. However, it may be the only choice if named graphs see heavy usage in the dataset and you are trying to work with a dataset that is not RDF-star enabled, or with SPARQL 1.0 (which doesn't support graphs).

Did this help you solve your issue? Your opinion is important not only to us but also to your peers.

Help us get better: review GraphDB on Gartner

Copyright: Ontotext New York | Sofia | Basel
Reach out to us: info@ontotext.com