How many stars do you see in this constellation?

Fabrizio Orlandi ✉, Damien Graux ✉, and Declan O'Sullivan

ADAPT SFI Centre, Trinity College Dublin, Ireland
{orlandif,grauxd,declan.osullivan}@tcd.ie

Abstract. With the increase of Knowledge Graphs available on the Web came the need of characterising them, adding for example provenance information. To facilitate adding statement-level metadata, an RDF syntax extension has recently been proposed to the Semantic Web community: RDF*. In this article, we examine the current coverage of RDF* by SPARQL engines. We identify a few issues arising even when performing simple operations such as counting the number of statements. We raise awareness on these issues and derive new research directions for the community.

1 Introduction

During the past decades, the number of linked datasets has rapidly increased, see for instance the current state of the Linked Open Data cloud[1]. These datasets are structured following the W3C standard Resource Description Framework (RDF) [8] and share knowledge on various domains, from the generalist ones such as DBpedia [1] or WikiData [12] to the most specialised ones, *e.g.* SemanGit [7].

The increasing number of these available sources of information, which can also be updated on a regular basis, implies that metadata characterising the datasets themselves is needed to help users find the correct pieces of information. For instance, provenance [10], versioning [3] and ownership of facts could be recorded on a statement-level and added to the datasets [2]. Hence, there is a need for expressing statements about statements, or statement-level metadata.

Technically, there exist several methods to express statements over a set of RDF triples. One might consider the various reification methods designed by experts in the community. However, these strategies lead, in practice, to an extensive amount of RDF triples generated [9], *e.g.* standard reification requires the use of four additional triples for each reified statement. In order to help users generating and maintaining their RDF statements of statements, Hartig et al. introduced the RDF* syntax [6,5]. This syntactical extension of the RDF standard has been received with enthusiasm by the Semantic Web community[2] and is now implemented by a few SPARQL engines.

[1] As of March 2019, the LOD-cloud gathers more than 1 200 datasets sharing more than 16 000 links. https://lod-cloud.net/

[2] The *W3C Workshop on Web Standardization for Graph Data* (2019) has set a direction to bridge the Semantic Web and Property Graph communities together indicating RDF* as a viable option. https://www.w3.org/Data/events/data-ws-2019/index.html

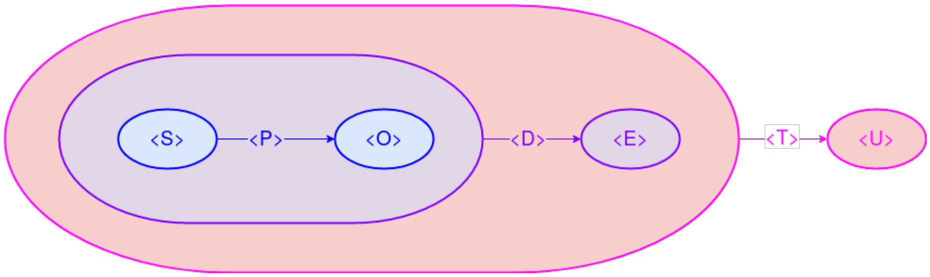

Fig. 1. A simple RDF* constellation of triples with nested "stars".

In this article, we explore the internal representations of RDF* data inside three SPARQL engines. Our simple test suite shows that the community should establish stricter guidelines and standardised methods to deal with RDF* datasets, as it appears that the considered engines are not treating this extension in a uniform way.

2 RDF*, Internal representations and Star counting

In 2014, Hartig & Thompson introduced the RDF* extension for RDF [6]. It allows data providers to shape statements about RDF graphs in an intuitive manner, while still being compliant with the RDF standard syntax. To do so, the RDF graph to be characterised should be encapsulated between double angular brackets and can act either as subject or object of another triple. In other words, RDF* graphs have the following generic structure: `(Subj|<<Graph>>) Pred (Obj|<<Graph>>)`. Moreover, the RDF* syntax allows data providers to nest their characterisations. Simultaneously, Hartig & Thompson proposed to extend the accepted syntax of SPARQL (the RDF associated query language) to allow users to extract this additional data level at query time.

While, in general, RDF* and SPARQL* syntax and semantics are described in [5], SPARQL* syntax is based on Pérez et al.'s algebraic SPARQL syntax [11]. A more detailed formalization of SPARQL*, and extension of the full W3C specification of SPARQL 1.1 [4], is available in [6]. However, as the W3C has not yet standardised the RDF*/SPARQL* syntax and semantics, a certain degree of freedom is taken by the community that does not have very strict guidelines to adhere to. This freedom of choice is especially true regarding the techniques used by the RDF* stores to internally represent datasets. In this case, a relevant example is the one regarding "Redundancy in RDF* Graphs" (*cf.* Section 2.4 in [5]) where implementation techniques are explicitly left free to choose how to deal with redundancy in order to achieve performance gains.

In order to illustrate this variety of possible representations, let's consider the simplest RDF* dataset: `<< <s> <p> <o> >> <d> <e>`; and let's try to answer the following question: "once stored, what should `SELECT (count(*) as ?c) WHERE {?s ?p ?o}` return?". One possible solution is to consider the reification method where actually **six** triples are to be created. Another one is to use the singleton property method [9] which would lead to **four** triples created. Finally, a more naïve one could be to answer **two**, as there are two predicates (`<p>` and `<d>`) in the dataset. Moreover, as the extension allows "stars" to be nested in one another, we can also consider the graph depicted in Figure 1, nesting two levels and corresponding to the following statement:

```
<< << <S> <P> <O> >> <D> <E> >> <T> <U> .
```

With this statement, we can as well ask the same simple question: "how many triples?". In the rest of our analysis, we are going to test such queries with SPARQL engines claiming to support RDF*/SPARQL*.

3 Comparing SPARQL* engines' results

Currently, only a few RDF management systems claim to cover the RDF* extension and natively support SPARQL* queries. Stardog[3] and Blazegraph[4] are among them. In addition, as a common basis of comparison, we use the tool provided by Hartig[5] which is built on top of Jena. In order to review their capabilities, we design the simple dataset which follows. It is composed of 4 statements: 1 classic RDF triple, 2 basic RDF* statements and 1 nested RDF* statement (as the one presented in Figure 1).

```
<s1> <p1> <o1> .
<< <s2> <p2> <o2> >> <d2> <e2> .
<< << <s3> <p3> <o3> >> <d3> <e3> >> <t3> <u3> .
<< <s4> <p4> <o4> >> <d4> <e4> .
```

One star – First, we only load in the systems the first two statements (Lines 1-2). Both Stardog and Blazegraph return that 3 entities were created during the loading phase. The `ExecuteSPARQLStar` tool can only query, on-the-fly, data stored in an RDF* file (*e.g.* using Turtle*). Then, we ran the following queries:
Q1: `Select * Where{?s ?p ?o}`.
Q2: `Select * Where{<<?s ?p ?o>> ?d ?e}`.
The obtained results, for both the engines and the Jena-based tool, were:
R_{Q1}: `{<s1>,<p1>,<o1>}{<s2>,<p2>,<o2>}{<< <s2><p2><o2> >>,<d2>,<e2>}`.
R_{Q2}: `{<s2>,<p2>,<o2>,<d2>,<e2>}`.
This implies that all systems consider the RDF-Graph of an RDF* statement as a subject and, in case of *Q1*, counted **3** triples (which corresponds to the naïve counting idea we had in the previous section).

Nested star – For the second round of experimentation we loaded all the 4 statements presented above. We used again the same queries *Q1*, *Q2* and added *Q3*: `Select * Where{<<<<?s ?p ?o>> ?d ?e>> ?t ?u}`.
The results are displayed in Table 1. We observe that both Blazegraph and the

[3] `Stardog` version 7.1.2 https://www.stardog.com/
[4] `Blazegraph` version 2.1.5 https://blazegraph.com/
[5] `ExecuteSPARQLStar` version 0.0.1 https://github.com/RDFstar/RDFstarTools

`Select * Where ...`	Stardog	Blazegraph	ExecuteSPARQLStar
`{?s ?p ?o}`	s1 p1 o1, s2 p2 o2, s3 p3 o3, s4 p4 o4, {s2 p2 o2} d2 e2, {s4 p4 o4} d4 e4, {s3 d3 e3} t3 u3, s3 d3 e3	s1 p1 o1, s2 p2 o2, s3 p3 o3, s4 p4 o4, {s2 p2 o2} d2 e2, {s4 p4 o4} d4 e4, {s3 p3 o3} d3 e3, {{s3 p3 o3} d3 e3} t3 u3	s1 p1 o1, s2 p2 o2, s3 p3 o3, s4 p4 o4, {s2 p2 o2} d2 e2, {s4 p4 o4} d4 e4, {s3 p3 o3} d3 e3, {{s3 p3 o3} d3 e3} t3 u3
`{<<?s ?p ?o>> ?d ?e}`	s2 p2 o2 d2 e2, s4 p4 o4 d4 e4, s3 d3 e3 t3 u3	s2 p2 o2 d2 e2, s4 p4 o4 d4 e4, s3 p3 o3 d3 e3, {s3 p3 o3} d3 e3 t3 u3	s2 p2 o2 d2 e2, s4 p4 o4 d4 e4, s3 p3 o3 d3 e3, {s3 p3 o3} d3 e3 t3 u3
`{<<<<?s ?p ?o>> ?d ?e>> ?t ?u}`	No results	s3 p3 o3 d3 e3 t3 u3	s3 p3 o3 d3 e3 t3 u3

Table 1. Behaviours with nested RDF* statements.

Jena-based tool present the same behaviour as the one observed in the previous test. Inversely, Stardog cannot deal correctly with nested RDF* statements. It is visible that Stardog "flattens" one level of the encapsulation as *Q3* does not offer any result and `s3 d3 e3` is present as a result of *Q1*, instead of the expected `{s3 p3 o3} d3 e3`.

Findings – As explained prior, the goal of our study is **not** to analyse the engines' performances, but rather to comparatively examine how they internally represent simple RDF* statements[6]. Despite the simplicity of the experiments, we find multiple syntax anomalies: *(i)* Stardog needs its own syntax based on curly-brackets (the RDF* angular-brackets often raise some exceptions); *(ii)* Blazegraph cannot deal with spaces at some places and raises errors when "`Select *`" is used (all the variables need to be specifically listed); *(iii)* the ExecuteSPARQLStar tool doesn't return the subject column[7] when there is an RDF* triple at the subject place in the clauses. More importantly, in addition to the syntactical errors, we discover that the three engines do not have the same internal representations. Further, it appears that Stardog's representation leads to errors, as it is "flattening" the nested statements.

4 Conclusions

Leveraging the simple example of counting the "stars" inside simple nested RDF* statements, we would like to alert the community on the current divergences that are appearing in the domain of RDF* storage and management. According to our observations, the tested versions of the engines diverge when dealing with nested statements. That is why we would not recommend to use nested RDF* statements in production systems yet, at least until a form of agreement in representing them has been reached. Indeed, as RDF* can drive the community forward and bridge the gap between the Semantic Web and Property Graph worlds, reaching an early agreement is of paramount importance as compatibility between engines would allow them to communicate and, in turn, enable features

[6] More details are available from: https://github.com/dgraux/RDFStarObservatory

[7] Similarly to Blazegraph, instead of using "`Select *`", users need to specify all the variables in the `Select` clause.

such as query federation. Our goal is to continue our systematic exploration of RDF* engines, paying attention to their behaviours and performances when dealing with more complex SPARQL* queries and with richer datasets.

Acknowledgments

This research was conducted with the financial support of the European Regional Development Fund and the European Unions Horizon 2020 research and innovation programme under the EDGE Marie Skłodowska-Curie grant agreement No. 713567 at the ADAPT SFI Research Centre at Trinity College Dublin. The ADAPT SFI Centre for Digital Media Technology is funded by Science Foundation Ireland through the SFI Research Centres Programme and is co-funded under the European Regional Development Fund (ERDF) Grant # 13/RC/2106.

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: A nucleus for a web of open data. In: The semantic web. Springer (2007)
2. Frey, J., Müller, K., Hellmann, S., Rahm, E., Vidal, M.E.: Evaluation of metadata representations in RDF stores. Semantic Web **10**(2), 205–229 (2019)
3. Frommhold, M., Piris, R.N., Arndt, N., Tramp, S., Petersen, N., Martin, M.: Towards versioning of arbitrary RDF data. In: Proceedings of the 12th International Conference on Semantic Systems. pp. 33–40 (2016)
4. Harris, S., Seaborne, A., Prud'hommeaux, E.: Sparql 1.1 query language. W3C recommendation **21**(10), 778 (2013)
5. Hartig, O.: Foundations of RDF* and SPARQL* (an alternative approach to statement-level metadata in RDF). In: 11th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW) (2017)
6. Hartig, O., Thompson, B.: Foundations of an alternative approach to reification in RDF. arXiv preprint arXiv:1406.3399 (2014)
7. Kubitza, D.O., Böckmann, M., Graux, D.: SemanGit: A linked dataset from git. In: International Semantic Web Conference. pp. 215–228. Springer (2019)
8. Manola, F., Miller, E., McBride, B., et al.: RDF primer. W3C recommendation **10**(1-107), 6 (2004)
9. Nguyen, V., Bodenreider, O., Sheth, A.: Don't like RDF reification? Making statements about statements using singleton property. In: Proceedings of the 23rd international conference on World wide web. pp. 759–770 (2014)
10. Orlandi, F., Passant, A.: Modelling provenance of DBpedia resources using Wikipedia contributions. Journal of Web Semantics **9**(2), 149–164 (2011)
11. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and complexity of SPARQL. ACM Transactions on Database Systems (TODS) **34**(3), 1–45 (2009)
12. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Communications of the ACM **57**(10), 78–85 (2014)