On the Role of the GRAPH Clause in the Performance of Federated SPARQL Queries

David Chaves-Fraga[1], Claudio Gutierrez[2], and Oscar Corcho[1]

[1] Ontology Engineering Group, Universidad Politénica de Madrid
[2] Computer Science Department & CIWS, Universidad de Chile

Abstract. Federated SPARQL queries give unified answers from multiple and distributed SPARQL endpoints. A good example may be the collection of stops from different transport companies in the same city to create a route planning application. The performance of the evaluation of these types of queries is usually poor, a fact that makes difficult their use in real-life applications that need good performance requirements.
In this paper we present a preliminary analysis on the improvement that can be achieved by using the GRAPH clause in federated SPARQL queries. The main goal is to reduce the search space of such queries by setting the NAMED GRAPH to the graph pattern where the corresponding patterns should be evaluated. We perform a preliminary comparison between a federated query and a rewriting that uses systematically the GRAPH clause. These experiments show that the inclusion of the GRAPH clause may only improve performance of query evaluation between 5% and 10%. These early findings suggest that, although the GRAPH clause may indeed play a role in speeding up federated SPARQL queries, hurdles are yet to be overcome when using the GRAPH clause as named graphs are semantically ambiguous.

Keywords: SPARQL, federated queries, graph, performance

1 Introduction

In recent years several public SPARQL endpoints have been made available on the Web containing billions of RDF triples that can be queried [4]. A way to query multiple SPARQL endpoints with a single query is by building a federated query. The definition of federated query is specified in the SPARQL 1.1 recommendation [5] and its syntax, semantics and evaluation are described on [3].

While these queries are useful for this purpose, their main disadvantage is their low performance, hence their use on real applications is limited. Other approaches are being currently applied, like getting the data from APIs, doing a manual data merge or joining data at the client side. Even so, federated queries are still an interesting approach to query RDF data sources exposed via SPARQL endpoints. Finding ways to improve their performance is one of the current research topics in the field.

A SPARQL endpoint is conceptualized as shown in Figure 1: one default-graph and from 0 to N NAMED GRAPHS. Each of these NAMED GRAPHS

contains a set of triples and has a unique identifier (URI). In a SPARQL query it is possible to indicate the graph where a particular part of a query (e.g. a pattern) will be evaluated completely or partially, by using the GRAPH clause with the graph URI. We hypothesize that using this feature in a systematic form may reduce the search space when evaluating federated SPARQL queries. The goal of this paper is to add evidence to this claim.

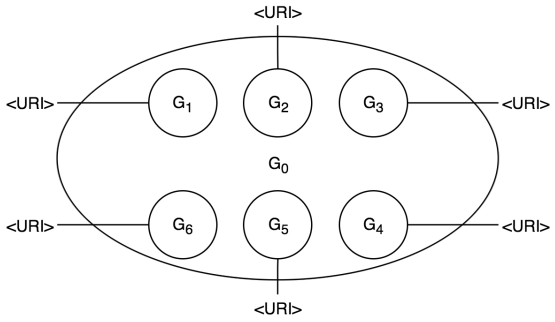

Fig. 1. SPARQL endpoint graphical conceptualization

The paper is organized as follows: Section 2 presents some of the related work on the conceptualization of SPARQL endpoints and federated queries. Section 3 describes our proposal about systematically using the GRAPH clause in federated queries. Section 4 shows a preliminary evaluation and presents some conclusions and future areas of work to be explored.

2 Related work

In this section we describe related work on two topics. On the one hand, the SPARQL endpoint conceptualizations; on the other hand, several optimizations that have been proposed in the area of federated SPARQL queries.

The SPARQL endpoint conceptualization is not an easy task. Hernández and Gutierrez [6] deal with current problems of the notion of the dataset in SPARQL that is defined as a set $\{G_0, (u_1, G_1), .., (u_n, G_n)\}$ where G_0 is called default graph and each G_i is a RDF graph with an associated URI (u_i). They try to identify the possibilities and the constraints of a dataset in a (federated) SPARQL query. There are further analysis on this topic in [2], in the specification of SPARQL [5] and in the documentation of federated SPARQL [11].

There are several works in the literature on federated SPARQL queries that try to optimize the performance of query evaluation [1][13], so as to make them applicable to real cases. The current approach in federated SPARQL queries deals essentially with the problem of source selection [14][12]. A different approach is detailed in [7], where the authors include the GRAPH clause in the queries to support graph-level access during the source selection step. By using the SERVICE clause in the queries the problem of source selection is automatically solved, but other interesting problems show up, like the availability of SPARQL endpoints in federated queries [4] or how to query heterogeneous resources following W3C standards like R2RML [10].

3 Using GRAPH clauses in federated SPARQL queries

In this section we propose an approach to improve federated query evaluation by systematically using the GRAPH clause to rewrite a query into a semantically equivalent one.

When the GRAPH clause is used in a SPARQL query, only triples saved into that GRAPH clause will occur in the resulting SPARQL bindings. We want to compare the performance between a federated SPARQL and a semantically equivalent query that makes use of the GRAPH clause to direct the evaluation. The definition of what is a graph pattern and its semantics and syntax can be found in [9]. Clearly the main challenge here is to identify the graph patterns that will only retrieve data from a unique GRAPH in the SPARQL endpoint. In this preliminary report we will assume this hypothesis, because we would like to show that it is actually possible to rewrite queries using this idea and this method add efficiency in the evaluation. We are aware that the possibility to identify this pattern is linked to information about sources. Once all the patterns are identified, one can rewrite the query in order to direct precisely the pattern evaluations to the relevant sources. The rationale behind this approach is that our experience shows that in a query usually there are many graph patterns that bind results from only one of the graphs in the SPARQL endpoint.

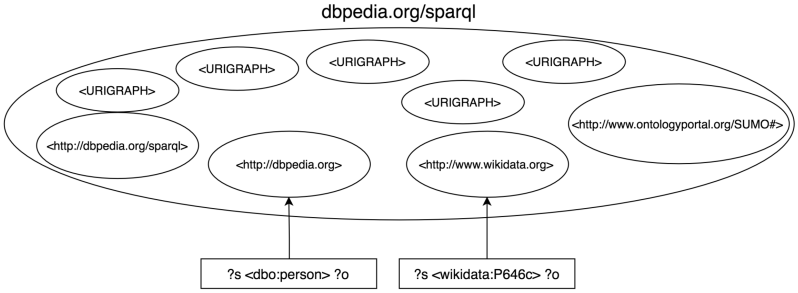

Fig. 2. Example our approach with DBpedia SPARQL endpoint

In Figure 2 we show graphically what does this approach mean for examples using the DBpedia SPARQL endpoint[3]. When evaluating a query with a graph pattern (?s <dbo:person> ?o), the data returned will belong to the graph <http://dbpedia.org>. Similarly, a pattern of the form (?s <wikidata:P646c> ?o) retrieves data only from graph <http://wikidata.org> in the same SPARQL endpoint. Thus would be interesting to be able to explicit this information in the query to be issued, so that the server can have more information and optimize the retrieval.

An example of how this transformation from a standard query into on which uses the GRAPH clause with this purpose is shown in the queries 1.1 and 1.2 below. The expectation is that the search space of a federated SPARQL query will be reduced and hence performance of query evaluation will improve.

[3] https://dbpedia.org/sparql

Listing 1.1. Federated query

```
SELECT * WHERE {
  SERVICE <S1> { GP1 }
  SERVICE <S2> { GP2 }
  SERVICE <S3> { GP3 }
}
```

Listing 1.2. Federated query with the GRAPH clause

```
SELECT *
FROM NAMED <S1-G2>
FROM NAMED <S2-G3>
FROM NAMED <S3-G1>
WHERE {
  SERVICE <S1> {
    GRAPH <S1-G2> {GP1} }
  SERVICE <S2> {
    GRAPH <S2-G3> {GP2} }
  SERVICE <S2> {
    GRAPH <S3-G1> {GP3} }
}
```

4 Preliminary evaluation

We performed some preliminary experiments to test our approach. We built federated queries over multiple real SPARQL endpoints[4,5,6,7] where their graph patterns only retrieve data from one of the graphs of each SPARQL endpoint. The queries were executed over the public real instances of the SPARQL endpoints. This first set of results, shown in Table 1, were obtained after 5 executions per each query and approach. We computed the average of the executions in milliseconds with the disabled cache on the browser. We also used a local Virtuoso 7.2.4 to send the queries over the real SPARQL endpoints.

	SERVICE	GRAPH
Q1	14630	**13510**
Q2	4360	**3220**
Q3	2880	**1980**
Q4	**1690**	1730
Q5	9120	**8560**

Table 1. Preliminary results

The results obtained after the experimentation reflect that a federated query that includes the GRAPH clause does not improve relevantly the performance of a federated query, as one could hypothesize. We only got an improvement in the queries evaluation time between 5% and 10%. The reason of this is that the conceptual information that a SPARQL endpoint provides about its graphs is not the same as the physical storage organization. If a query over a public SPARQL endpoint is made to list all the graphs of that SPARQL endpoint

[4] https://dbpedia.org/sparql
[5] https://kb.3cixty.com/sparql
[6] http://data.linkedmdb.org/sparql
[7] https://github.com/dachafra/federated-graph-queries

and their related triples (an example of DBpedia[8] is provided) conceptually the information that the user receives is that this number of triples are stored into those graphs. The reality, however, is that usually all the triples are stored in the same place and thus it does not make relevant difference the approach we are proposing. As reflected on Virtuoso documentation[9], it currently puts the triples in a single table with the graph URI as a key part although they are already working on an improvement to store triples in their graphs. Testing our approach with this future improvement of Virtuoso (or other triple store that includes this feature) could prove that including the GRAPH clause in federated SPARQL queries improves their performance.

4.1 Conclusion: The role of the GRAPH clause in SPARQL

After this short experiment one could ask: What is the role of the GRAPH clause in SPARQL? We know that it is useful from a semantic point of view, that is, when a graph pattern matches triples that are saved in multiple graphs and the user only wants the triples from one of them. But we have identified issues that are not evident when using a GRAPH clause: (i) semantic ambiguity; (ii) offers a view of how the data is conceptually ordered but not physically; (iii) is a SPARQL clause that seems to not allow any type of optimization.

One of the important problems related with the graphs of a SPARQL endpoint is the correctness of their characterization. When an open data portal or a controlled vocabulary is developed, the format of the URIs is highly structured. Also, only by analysing the URI itself, one usually can get some interesting information about what type of data they are storing[10,11]. If this feature were taken into account at the moment of building the graphs of a SPARQL endpoint, it might be useful for the users to have an idea of what type of triples are loaded in each graph. In summary, the role of the GRAPH clause needs to be clarified in order to be really useful as a construct of the language.

The problems presented generate different lines for future research, such as creating a federated benchmark that includes the SERVICE and GRAPH clauses and do the evaluation in a controlled environment, applying our approach to other triple stores or carrying out semantic equivalence between open data portal and a SPARQL endpoint. It would also be very useful to develop a way for automatically linking unique graph patterns with their pertained graphs in a SPARQL endpoint using an index or other current solutions like Loupe[8]. This problem is reflected in some inconsistencies between the specification and real-life implementations. For example, although the syntax of federated SPARQL 1.1 does not allow to include GRAPH clauses in a federated query, Virtuoso and Apache Jena can execute the approach presented in this note.

[8] https://github.com/dachafra/federated-graph-queries/blob/master/DBpediaGRAPHS.csv
[9] https://virtuoso.openlinksw.com/rdf/
[10] http://ec.europa.eu/transport/facts-fundings/statistics/pocketbook-2013_en
[11] http://ec.europa.eu/eurostat/web/products-datasets/-/rail_go_quartal

Acknowledgements. The research leading to this paper has been partially done in the context of the FP7 Marie Curie IRSES SemData project (http://www.semdata-project.eu/), grant agreement No PIRSES-GA-2013-612551, and it is also supported by the 4V Spanish national project (TIN2013-46238-C4-2-R) .

References

1. Maribel Acosta, Maria-Esther Vidal, Tomas Lampo, Julio Castillo, and Edna Ruckhaus. Anapsid: an adaptive query processing engine for SPARQL endpoints. *The Semantic Web–ISWC 2011*, pages 18–34, 2011.
2. Renzo Angles and Claudio Gutierrez. SQL nested queries in SPARQL. In *AMW*, 2010.
3. Carlos Buil-Aranda, Marcelo Arenas, Oscar Corcho, and Axel Polleres. Federating queries in SPARQL 1.1: Syntax, semantics and evaluation. *Web Semantics: Science, Services and Agents on the World Wide Web*, 18(1):1–17, 2013.
4. Carlos Buil-Aranda, Aidan Hogan, Jürgen Umbrich, and Pierre-Yves Vandenbussche. SPARQL web-querying infrastructure: Ready for action? In *International Semantic Web Conference*, pages 277–293. Springer, 2013.
5. Steve Harris, Andy Seaborne, and Eric Prudhommeaux. SPARQL 1.1 query language. *W3C recommendation*, 21(10), 2013.
6. Daniel Hernández and Claudio Gutierrez. Disentangling the notion of dataset in SPARQL. In *Alberto Mendelzon International Workshop on Foundations of Data Management*, page 213, 2015.
7. Yasar Khan, Muhammad Saleem, Aftab Iqbal, Muntazir Mehdi, Aidan Hogan, Axel-Cyrille Ngonga Ngomo, Stefan Decker, and Ratnesh Sahay. Safe: policy aware SPARQL query federation over RDF data cubes. In *Proceedings of the 7th International Workshop on Semantic Web Applications and Tools for Life Sciences, Berlin, Germany, December 9-11, 2014.*, 2014.
8. Nandana Mihindukulasooriya, Raúl García-Castro, and Miguel Esteban-Gutiérrez. Linked data platform as a novel approach for enterprise application integration. In *Proceedings of the Fourth International Conference on Consuming Linked Data-Volume 1034*, pages 146–157. CEUR-WS. org, 2013.
9. Jorge Pérez, Marcelo Arenas, and Claudio Gutierrez. Semantics and complexity of SPARQL. *ACM Transactions on Database Systems (TODS)*, 34(3):16, 2009.
10. Freddy Priyatna, Carlos Buil Aranda, and Oscar Corcho. Applying SPARQL-DQP for federated SPARQL querying over google fusion tables. In *Extended Semantic Web Conference*, pages 189–193. Springer, 2013.
11. Eric Prudhommeaux, Carlos Buil-Aranda, et al. SPARQL 1.1 federated query. *W3C Recommendation*, 21, 2013.
12. Muhammad Saleem and Axel-Cyrille Ngonga Ngomo. Hibiscus: Hypergraph-based source selection for SPARQL endpoint federation. In *European Semantic Web Conference*, pages 176–191. Springer, 2014.
13. Muhammad Saleem, Axel-Cyrille Ngonga Ngomo, Josiane Xavier Parreira, Helena F Deus, and Manfred Hauswirth. Daw: Duplicate-aware federated query processing over the web of data. In *International Semantic Web Conference*, pages 574–590. Springer, 2013.
14. Maria-Esther Vidal, Simón Castillo, Maribel Acosta, Gabriela Montoya, and Guillermo Palma. On the selection of SPARQL endpoints to efficiently execute federated SPARQL queries. In *Transactions on Large-Scale Data-and Knowledge-Centered Systems XXV*, pages 109–149. Springer, 2016.