Applying SPARQL-DQP for federated SPARQL querying over Google Fusion Tables

Freddy Priyatna[1], Carlos Buil Aranda[2], and Oscar Corcho[1]

[1] Ontology Engineering Group, Facultad de Informática, UPM, Spain
{fpriyatna,ocorcho}@fi.upm.es
[2] Department of Computer Science, PUC Chile
cbuil@ing.puc.cl

Abstract. Google Fusion Tables (GFT) is a data management, integration and visualization service provided by Google. Users can upload their structured data, integrate it with other people's data, and visualize it on various tools provided, such as Google Maps, charts or graphs. Every GFT table constitutes a data silo that is not commonly linked to other data sources. A way to enable data to be linked and reused is by exposing it as (virtual) RDF dataset (for instance, using R2RML) and to query it using SPARQL. In this work, we present a system that exposes GFT tables as SPARQL endpoints, enabling federated SPARQL queries with data from other SPARQL endpoints.

Keywords: Google Fusion Tables, SPARQL, DQP, R2RML

1 Introduction

Announced in 2009, Google Fusion Tables [5] is a data management service provided by Google to help users in working with their structured data. Users can upload their CSV data and host in the cloud on Google infrastructure such as Google Bigtable for scalability purposes. Hosting data in the cloud brings several benefits, such as enabling the data to be shared with other users, to be merged with other people's data, and even allowing other users to collaborate by giving read/write permission. Google also integrates various visualisation tools so that users can share their data easily in various forms, such as on maps (using Google Maps), charts, or graphs. While the number of GFT tables users or available tables is not publicly advertised, it is believed that there are 400 millions active GMail users[3], and each of them potentially can use this service and contribute their own tables.

As aforementioned, data hosted in GFT tables can be merged with other's people data. However, this can work only among GFT tables. Seen from outside the service, GFT tables consistute data silos that are not commonly linked to other data sources.

In this work, we present a system that exposes GFT tables as SPARQL endpoints by using R2RML [4], and we use SPARQL-DQP to demonstrate that

[3] http://www.quora.com/Gmail/How-many-total-active-Gmail-users-are-there

federated queries can be evaluated over them as if they were RDF datasets, thus taking benefit of the ease and scalability provided by the Google infrastructures while at the same time exploting the power of querying over RDF enabled sources. The remainder of the paper is as follows: Section 2 presents the architecture of our system and in Section 3 we see how the system works by using some examples. Finally in Section 4 we close the paper by giving the conlusion and the future work.

2 Architecture

The general architecture of the system is shown in Figure 1.

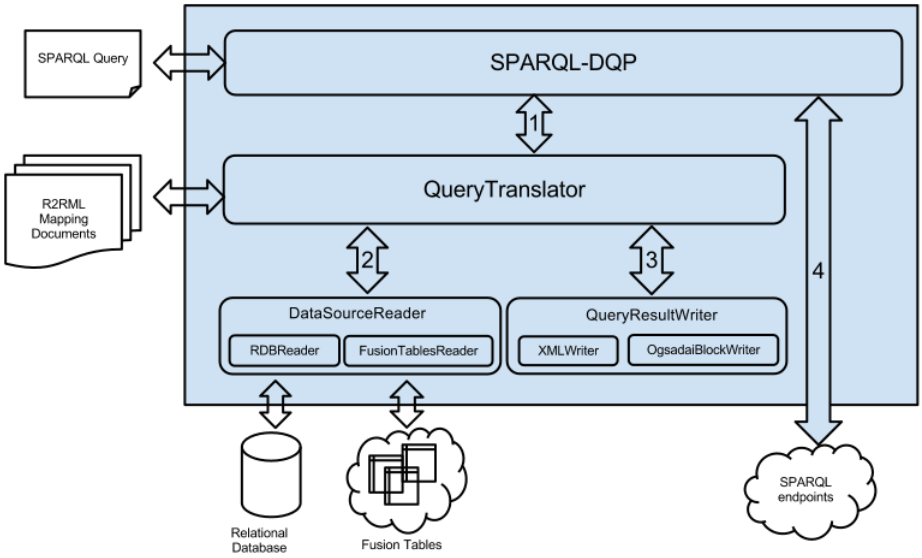

Fig. 1. Architecture of the system

A federated SPARQL query sent by the user is received by SPARQL-DQP [2]. SPARQL-DQP is a SPARQL 1.1 Federated Query system based on OGSA-DAI/DQP system [1]. The SPARQL 1.1 Federated Query specification [6] defines a SERVICE keyword as the location of the SPARQL endpoint that corresponds to the part of the query. By using SPARQL-DQP, we enable the SPARQL queries to query data not just from Fusion Tables, but also from available public SPARQL endpoints such as the ones in Linked Open Data (LOD) cloud. The user query is a normal SPARQL 1.1 query in which the SERVICE keyword may identify either a remote SPARQL endpoint or the R2RML mapping file that is used in the query translation process. The following processes occur when the system receives a SPARQL query:

- SPARQL-DQP divides the query into queries to be evaluated in each remote endpoints, as explained in [2], and sends each of them to the Query Translator (1) or directly to the corresponding SPARQL endpoint (4).
- The Query Translator translates (2) the SPARQL query from the user into an SQL query using the algorithm defined in [3]. That new SQL query is sent to the Fusion Table service by the Data Source Reader which will also retrieve the results.
- These results will be converted into the internal SPARQL-DQP representation and streamed into the main data workflow (3). This data workflow is managed by SPARQL-DQP [2].

3 Example

We now explain the process when the system receives the SPARQL query *"give me all the members of the Ontology Engineering Group coming from a country whose capital is Madrid"*, shown in Listing 1.1.

Listing 1.1. Example of Federated SPARQL Query

```
1  SELECT ?n ?c
2  WHERE {
3    SERVICE <http://mappingpedia.linkeddata.es/mappings/fusiontables/
4      1pQBGUqR_g-j1WQavu-Fi1wGS7jsdRxomGc0DxMI/oegmembers.ttl>
5    {
6      ?m rdf:type foaf:Person.
7      ?m foaf:name ?n.
8      ?m ex:hasCountry ?c.
9    }
10   SERVICE <http://DBpedia.org/sparql> {
11     ?c <http://dbpedia.org/property/capital> <http://dbpedia.org/resource/Madrid>.
12   }
13 }
```

That query asks for data about the members of the OEG research group (name and country) and also asks for data about Spain. For that, the query contains two SERVICE calls, one to the DBpedia SPARQL endpoint (asking about the country resource) and another one to a Google Fusion table containing the data about the members of this research group.

The part of the query that is addressed at the Fusion Table is sent to the Query Translator along with the mapping document specified in the SERVICE URI. The Query Translator component translates the SPARQL query into a SQL query using the mapping information specified in the mapping document. The SPARQL and resulting SQL query can be seen in Listing 1.2. That SQL query is then sent to Fusion Table Reader component, which uses Google Fusion Table API to execute the query and process the results which are streamed into the SPARQL-DQP data workflow.

Meanwhile the SERVICE call to DBpedia is executed in parallel and SPARQL-DQP will join the results with the ones obtained from the GFT table query

execution. All these remote query executions and data transfers form a single data workflow which is managed by SPARQL-DQP for finally send the results back to the user.

A video showing the example can be seen at this URL : `http://www.youtube.com/watch?v=qrTBJbnTHnc`.

Listing 1.2. Query that is addressed at the GFT table

```
1  -- SPARQL query sent to Query Translator
2  SELECT ?n ?c
3  WHERE {
4      ?m rdf:type foaf:Person.
5      ?m foaf:name ?name.
6      ?m ex:hasCountry ?c.
7  }
8
9  -- SQL query produced by Query Translator
10 SELECT name, country
11 FROM 1pQBGUqR_g-j1WQavu-Fi1wGS7jsdRxomGc0DxMI
12 WHERE name NOT EQUAL TO ""
13   AND country NOT EQUAL TO ""
```

4 Conclusion and Future Work

In this demo we will present a system that combines two features: first, exposing Google Fusion Tables as SPARQL endpoints using R2RML mapping documents in order to map the values from Fusion Tables as virtual RDF datasets; second, querying these virtual RDF datasets together with public SPARQL endpoints by using SPARQL-DQP.

In the current system, the user has to put the mapping document URL manually in the SPARQL query. In the future work, we will store those mapping documents in our triples stores and enable a SPARQL endpoint. In that way, instead of providing the URL of the mapping document, a user can just ask "get me all the mapping documents that map the concept foaf:person".

Acknowledgments: Freddy Priyatna and Oscar Corcho have been supported by the PlanetData (FP7-257641) and myBigData (TIN2010-17060) projects. Carlos Buil-Aranda has been supported by the CONICYT/FONDECYT project number 3130617.

References

1. M. Alpdemir, A. Mukherjee, A. Gounaris, N. Paton, P. Watson, A. Fernandes, and D. Fitzgerald. OGSA-DQP: A service for distributed querying on the grid. *Advances in Database Technology-EDBT 2004*, pages 3923–3923, 2004.
2. C. Buil-Aranda, M. Arenas, and O. Corcho. Semantics and optimization of the SPARQL 1.1 federation extension. *The Semanic Web: Research and Applications*, pages 1–15, 2011.
3. A. Chebotko, S. Lu, and F. Fotouhi. Semantics preserving SPARQL-to-SQL translation. *Data & Knowledge Engineering*, 68(10):973–1000, 2009.

4. S. Das, S. Sundara, and R. Cyganiak. R2RML: RDB to RDF mapping language. 2012.
5. H. Gonzalez, A. Halevy, C. S. Jensen, A. Langen, J. Madhavan, R. Shapley, and W. Shen. Google fusion tables: data management, integration and collaboration in the cloud. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 175–180. ACM, 2010.
6. E. Prudhommeaux and C. Buil-Aranda. SPARQL 1.1 federated query. *World Wide Web Consortium, Proposed Recommendation (November 2012)*, 2012.