Efficient Mobile Querying of Distributed RDF Sources

Elien Paret[1], William Van Woensel[1], Sven Casteleyn[2], Beat Signer[1], Olga De Troyer[1]

[1]Vrije Universiteit Brussel, Pleinlaan 2, 1050 Brussels, Belgium
{Elien.Paret, William.Van.Woensel, Beat.Signer, Olga.DeTroyer}@vub.ac.be
[2]Universitat Politècnica de València, Camino de Vera S/N, 46007, Valencia, Spain
Sven.Casteleyn@upv.es

Recent advancements in mobile devices and the omnipresence of wireless connectivity have turned handheld devices into powerful mobile web clients. This evolution has made the wealth of online Semantic Web data accessible to mobile devices. To support this new opportunity, we present a semantic technology-based client-side solution to efficiently query large amounts of distributed online RDF sources. The main idea is to continuously extract and manage metadata from RDF sources and to store this metadata in a local Source Index Model (SIM). At query execution time, the SIM is consulted to identify and assemble potentially relevant sources. In contrast to other approaches [1-3], we do not rely on any query endpoints. Our solution pays attention to the computational limitations of mobile devices by querying only relevant sources and thereby improving the overall query execution time. We do not aim to replace existing mobile query engines, but rather build on them to efficiently and transparently query large sets of online RDF sources.

The first step of our solution involves the creation of a SIM as highlighted in Fig. 1. Each time we receive a new source reference from an application, the source is downloaded and relevant metadata is extracted and stored in the SIM. Since RDF is a predicate-based formalism, we base our approach on the presence of predicates in the data sources to filter out irrelevant data sources similar to [1,2]. We further aim to exploit the semantic information embedded in RDF documents in terms of the type of the predicate subjects (i.e. domain). RDF sources and SPARQL queries often use this domain information to detail triples and restrict triple patterns. We consider two different SIM variants: the first variant stores only the found predicates while the second variant manages the found predicates together with their domains. Note that the second SIM variant still supports queries with unspecified predicate domains.

The second step of our approach consists in resolving an application query over the combined data set of relevant sources as shown in Figure 2. First, the query analyzer extracts the metadata of the given SPARQL query that is compatible with the data stored in the SIM (predicates or predicates with their domains). The triple patterns occurring in the WHERE clause of a SPARQL query are used to restrict the query results in the graph pattern matching process. Therefore, we need to examine the WHERE clause in order to identify potentially relevant sources. Subsequently, the extracted query metadata is matched to the metadata managed by the Source Index Model. The matching is performed for each triple pattern. Sources that contain predicates (and subject types) referenced in one or more query triple patterns are also

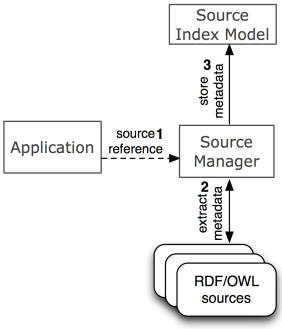

Fig. 1. RDF indexing

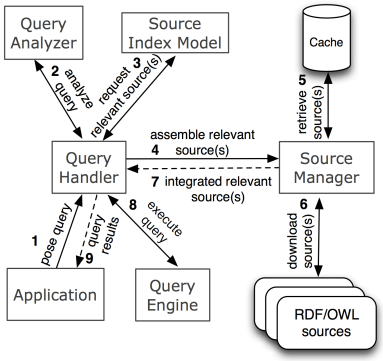

Fig. 2. Query handler

included in the final dataset. In this way, queries that are not solvable by a single data source can potentially be answered by a combination of data sources. Once the relevant sources are identified, they are assembled and the query engine, for example Androjena, executes the query on the collected set of relevant data sources. To validate our solution, we conducted a series of performance experiments on a Sony Ericsson Xperia X10 device with 567 MB memory and a 1 GHz processor. Four distinctive queries have been executed on datasets of 50, 100, 250 and 500 RDF sources. These RDF files had an average size of 3.7 kB and were automatically generated using random resource types and properties from a number of ontologies describing, for example, buildings, shops and products. For the dataset of 500 sources, the use of the first SIM variant resulted in an average query response time of 20198 ms, the use of the second SIM variant resulted in an average of 8415 ms whereas 26099 ms were necessary if no SIM was used.

We have presented a client-side query service for the efficient querying of distributed RDF sources in mobile settings. Our solution is based on a Source Index Model and does not require any query endpoints. A major challenge was to find the right balance between the amount of stored index data and the resulting maintenance overhead on the one hand, and the number of potentially filtered irrelevant data sources on the other hand. Our validation confirms that we can achieve a significant speed-up in querying distributed RDF data sources while ensuring that our solution scales well with growing sets of data sources.

Acknowledgement: Sven Casteleyn is supported by an EC Marie Curie Intra-European Fellowship for Career Development, FP7-PEOPLE-2009-IEF, N° 254383.

References

[1] B. Quilitz and U. Leser, Querying Distributed RDF Data Sources with SPARQL, *Proc. of ESWC 2008*, Tenerife, Spain, June 2008.

[2] H. Stuckenschmidt, R. Vdovjak, J. Broekstra and G.-J. Houben, Towards Distributed Processing of RDF Path Queries, *International Journal of Web Engineering and Technology*, 2(2/3), 2005.

[3] Z. Kaoudi, K. Kyzirakos and M. Koubarakis, SPARQL Query Optimization on Top of DHTs, Proc. of ISWC 2010, Shanghai, China, 2010.