SemWIQ – Semantic Web Integrator and Query Engine

Andreas Langegger, Wolfram Wöß

{al|wolfram.woess}@jku.at

Abstract: One of the most popular applications of Semantic Web technology is the integration of data from distributed locations over the Web. With wrappers, screen scrapers, and information extraction tools it is possible to access, merge, and reason over RDF data from various different, heterogeneous sources. SPARQL can be used to access RDF data in a declarative manner and as will be shown, it can also be used to virtually integrate distributed RDF graphs.

In this contribution a mediator-wrapper based middleware is shown which virtually integrates distributed, heterogeneous data sources based on RDF and a slightly modified SPARQL algebra. Schema information available in RDF Schema and OWL vocabularies is used for query federation and optimization. Compared to traditional data integration systems, SemWIQ can fully exploit the semantics of available RDF data. Additionally, RDF has many interesting properties which provide advanced query capabilities based on OWL DL inference.

1 Introduction

A data integration middleware usually uses a rich data model to represent data originating from diverse data sources in a common global scheme. The Resource Description Framework is actually rather simple, but together with the concepts of IRIs and namespaces it can be arbitrarily extended. With the RDF Schema and Ontology layers it allows for meta-modelling and is therefore well suited as the core data model of such a middleware. A very common approach for data integration systems is the mediator-wrapper architecture [Wie92]. While a mediator is the core component which provides transparent access to the integrated data, several wrappers map local data schemes to the global schema and translate data accordingly. With most of the basic tools available for working with RDF, OWL, and SPARQL, it is already possible to build such a data integration system very quickly. There also exist several tools for wrapping structured data into RDF, scrape and extract RDF from the Web or mine document collections.

Based on the requirements and application scenario, there are many different ways how to build a data integration system based on RDF and SPARQL. The most popular approach is probably to *query a consolidated data set*. Within this approach all data is dumped into a global graph at the mediator before queries are executed against it. This is very fast of course, because all data is available in a central data store at one location. But the consolidated data set may not be up-to-date and moreover, the approach is only useful for small to medium-sized graphs because all data has to be transfered to the mediator first. To avoid this, data access can also be carried out on-the-fly during query execution.

Such an approach could be called *queries over a federated data set*. This actually reflects the named graphs extension to RDF, respectively the capability of SPARQL to execute queries over multiple graphs. The mediator implementing the SPARQL protocol may link named graphs to remote data sources and try to fetch data from unknown graph IRIs from the Web. However, usually the complete remote graph will be downloaded to the mediator resulting in a bad query execution performance. An interesting extension of SPARQL implemented in Jena is the `Service`[1] operator, which may be used to delegate sub-plans to remote SPARQL endpoints. In both cases, the user must explicitly specify the IRIs of graphs or SPARQL endpoints in the query which does not enable virtual data integration. A third approach, called *queries over a virtual data set* is the core concept of SemWIQ. It is a virtual data integration system which federates global queries into local sub-queries and optimizes the distributed plan based on static rules and cost estimates. This approach provides a completely transparent and already optimized data access to distributed SPARQL endpoints. Together with RDF wrappers it may be used to integrate heterogeneous data sources based on RDF and SPARQL. For relational database systems D2R-Server [CBRC06] is used as a wrapper. A generic wrapper for table-oriented file-based data (CSV, Excel, etc.) is already planned and further wrappers are expected to become available in future by the community.

2 Description of SemWIQ

The basic architecture of SemWIQ is the traditional mediator-wrapper approach as proposed in [Wie92]. A detailed description of SemWIQ can also be found in [LWB08]. The query execution process will be described briefly as follows. Clients connect to the mediator and request data by submitting *global* SPARQL queries. Classes and properties in such queries may refer to any vocabulary that is used to describe available data in remote data sources. The query parser validates the query and parses the lexical string into an algebraic operator tree. The operator tree is then expanded by the federator (Sect. 2.1) and optimized afterwards (Sect. 2.2). Finally, the query engine executes the global plan. Several sub-plans will be executed against remote endpoints which may be RDF wrappers like D2R-Server. The results can be delivered in various formats like the SPARQL Query Results XML Format, JSON, or as an RDF graph.

2.1 Query Federation

SemWIQ uses a concept-based data integration approach. This means, that all data must be defined as instances of some class. A global query in SemWIQ is defined just by E', which is a slightly modified SPARQL algebra expression. By contrast to the original definition of SPARQL, the data set DS is not part of the query (it is virtually defined by the registry catalog) and the only query form R currently allowed is `SELECT`.

[1] `http://seaborne.blogspot.com/2007/07/basic-federated-sparql-query.html`

For E', the `Graph` operator was dropped, the special `Service` operator of ARQ2 was added (it sends the underlying sub-plan as a back-serialized SPARQL query to a remote endpoint) and the basic graph pattern (BGP) is restricted as follows: $BGP' = V \times (I \cup V) \times (RDF\text{-}T \cup V)$, where V is the set of query variables, I the set of IRIs, $RDF\text{-}T$ is the set of $RDF\text{-}Terms = (I \cup RDF\text{-}L \cup RDF\text{-}B)$, $RDF\text{-}L$ the set of literals, and $RDF\text{-}B$ the set of blank nodes. Thus, the subject must always be variable and may never be concrete[2]. Further, each subject in a query must be typed, i.e. being V_s the set of all subject variables, t a triple pattern, and I_C the set of RDF or OWL classes, the following condition must hold: $\forall v_s \in V_s \; \exists t(v_s, \text{rdf:type}, c), c \in I_C$. If the vocabulary used in the query (respectively for data descriptions) contains OWL restrictions on properties between nodes, the federator may also infer types of linked resources. For the data source selection process, the federator requires at least one asserted or inferred type for each subject variable. Selecting data sources means, expanding the original operator tree and inserting `Service` operators, joins and unions accordingly. A BGP with multiple subject variables will be split into multiple joined BGPs, and each of these BGPs is further replaced by a union of new BGPs for each data source. This algorithm (explained in [LWB08]) finally leads to a global query plan which is usually very large and requires global plan optimization.

2.2 Global Query Plan Optimization

Global query plan optimization is currently based on declarative rules. The RETE-based *JBoss Rules* engine is used and several rules are used to move down filters and projections, move any unary operator beyond `Service`, and merge filter expressions. Similar to functional rules introduced in [GML88], this has the advantage of being extensible without the need to alter optimizer code if new methods for optimizing queries are introduced. Since BGPs are always executed locally at remote SPARQL implementations, the optimization of BGPs (reordering triple patterns) is currently ignored. In future, when endpoints may provide more detailed information about their capabilities and indices, the BGP operator may at least provide better cost estimates. Anyway, a remote query engine is usually responsible for BGP optimization.

However, within a distributed query processor like SemWIQ, shipping data for distributed join processing is usually much more costly than local query processing. Compared to local query processing, where the cost model is mainly influenced on access methods (sequencial vs. index scan), in a first step the global cost model is only based on the size of RDF tuples returned by endpoints. The global plan optimizer is currently implemented based on such a cost model and will be presented in a forthcoming paper. A combination of functional rules and an (iterative) dynamic programming (IDP) algorithm for global join reordering has already been presented in [LMHDKELWJY97]. Indeed, most of the research from the database community is already applicable to SPARQL query processing. Additionally, because RDF incorporates schema information and semantics, it is assumed that there are much more possibilities for efficient distributed query optimization.

[2]Possibly, in a future version SemWIQ is extended to support `DESCRIBE` queries and IRIs for subjects.

For a query optimizer, it is usually required to provide or generate statistics of the available data sets. Because current wrapper implementation neither provide endpoint descriptions, nor such statistics, a statistics monitor which can be attached locally is currently developed. To achieve maximum compatibility and ease of use, this component will be an autonomous deamon running on the remote host and accessing the endpoint locally through the SPARQL protocol. Another approach would be to create an in-process extension for specific wrappers like D2R and export the statistics as a special named graph. However, the first approach seams more promising and reusable for diverse, also non-Java wrapper implementations.

The statistics monitor will provide statistics like the number of instances for all classes used, frequencies of properties used, as well as the distribution of property values at the data sources as compressed histograms.

3 Related Work

Data integration is already a mature field of research. Early work already started just after the first database systems had been introduced. One example of early work based on the mediator-wrapper architecture [Wie92] was the TSIMMIS project [SCHGMJH+94]. However, using ontologies for the global data model is fairly new. Because RDF is very extensible based on global IRIs and namespaces, it is well suited for a decentralized, scalable data integration system. A more current and similar project is DARQ (distributed ARQ) [BQ06]. Data source selection in DARQ is only based on properties. Ontological meta data which would have interesting properties for query optimization and also advanced query capabilities are not used. It remains also rather unclear, how optimization is done in DARQ (it seems it is currently limited to reordering triples in Basic Graph Patterns).

4 Conclusion

In this contribution a middleware for virtual data integration based on RDF and a slightly modified version of SPARQL is presented. It can be used to integrate distributed, heterogeneous data sources which may be described by various RDF and OWL vocabularies. The system is based on the mediator-wrapper architecture. For relational database systems, wrappers like D2R-Server can already be used, other wrappers are expected to become available in future. The most important components of the mediator: the federator and the global plan optimizer have been described. A detailed description of the currently implemented static transformation rules based on JBoss Rules as well as the cost-based global plan reordering algorithm will be provided in a forthcoming paper. To further optimize distributed SPARQL queries, several extensions are necessary. Beside the implementation of the statistics monitor mentioned, other extensions involve the implementation of a special two-phase semi-join operator commonly used in distributed query processing as well as the support for row blocking to transfer multiple RDF tuples at once in a batch manner.

The cost model for the IDP-based global plan reordering process will later incorporate cost of local operations as for example low-level data access based on different indices available. The middleware provides a good basis for further research towards data integration with Semantic Web technologies and distributed SPARQL query optimization. Processing distributed SPARQL queries is regarded to have further impact on the Semantic Web community in future.

Acknowledgements

The work described in this paper is supported by the *Austrian Grid Project*, funded by the Austrian BMBWK (*Federal Ministry for Education, Science and Culture*) under contract GZ BMWF-10.220/0002-II/10/2007.

References

[BQ06] Bastian Quilitz. DARQ – Federated Queries with SPARQL. `http://darq.sourceforge.net/`, 2006. Last visit: Dec 12, 2007.

[CBRC06] Chris Bizer and Richard Cyganiak. D2R Server – Publishing Relational Databases on the Semantic Web. In *5th International Semantic Web Conference*, 2006.

[GML88] Guy M. Lohman. Grammar-like functional rules for representing query optimization alternatives. In *SIGMOD '88: Proceedings of the 1988 ACM SIGMOD international conference on Management of data*, pages 18–27, New York, NY, USA, 1988. ACM Press.

[LMHDKELWJY97] Laura M. Haas, Donald Kossmann, Edward L. Wimmers, and Jun Yang. Optimizing Queries Across Diverse Data Sources. In *Proceedings of the 23th International Conference on Very Large Databases*, pages 276–285, Athens, 1997. VLDB Endowment, Saratoga, Calif.

[LWB08] A. Langegger, W. Wöß, and M. Blöchl. A Semantic Web middleware for Virtual Data Integration on the Web. In *Proceedings of the European Semantic Web Conference 2008, Tenerife*, 2008.

[SCHGMJH+94] Sudarshan Chawathe, Hector Garcia Molina, Joachim Hammer, Kelly Ireland, Yannis Papakonstantinou, Jeffrey D. Ullman, and Jennifer Widom. The TSIMMIS Project: Integration of heterogeneous information sources. In *16th Meeting of the Information Processing Society of Japan*, pages 7–18, Tokyo, Japan, 1994.

[Wie92] Gio Wiederhold. Mediators in the Architecture of Future Information Systems. *Computer*, 25(3):38–49, 1992.