SPARQL–RW: Transparent Query Access over Mapped RDF Data Sources

Konstantinos Makris[‡] Nikos Bikakis[†¥] Nektarios Gioldasis[‡] Stavros Christodoulakis[‡]

[‡] TUC/MUSIC Lab | Technical University of Crete | Greece

[†] National Technical University of Athens | Greece [¥] IMIS Institute | "Athena" Research Center | Greece

[makris, nektarios, stavros]@ced.tuc.gr, bikakis@dblab.ntua.gr

ABSTRACT

The *Web of Data* is an open environment consisting of very large, inter-linked RDF datasets from various domains (e.g., *DBpedia*, *GeoNames*, *ACM*, *PubMed*, etc.) accessed through SPARQL queries. Establishing interoperability in this environment has become a major research challenge. This paper presents SPARQL–RW (SPARQL–ReWriting), a framework which provides transparent query access over mapped RDF datasets. The SPARQL–RW provides a generic method for SPARQL query rewriting, with respect to a set of predefined mappings between ontology schemas. To this end, it supports a set of rich and flexible mapping types and it is proved to provide semantics preserving queries.

Keywords

SPARQL query rewriting, Linked Data, Ontology mapping, Interoperability, Semantic Web Databases, Web of Data.

1. INTRODUCTION

The *Web of Data* is an environment that allows publishing data on the Web, in structured, linked, and standardized ways. It is comprised by a great number of very large inter-linked RDF datasets from various domains (e.g., *DBPedia*, *ACM*, *PubMed*, *BBC Music*, *GeoNames*, *Flickr*, etc.), and initiatives like the *Linked Open Data*, *Open Government* and *Linked Life Data* have played a major role towards its development.

In this environment, it is very common for several datasets to describe the same or overlapped domains. A plethora of such examples can be given, starting from the *DBpedia*, *YAGO*, *WordNet* and *Freebase* cross-domain datasets. Taking it a step forward, we notice several other overlapping datasets, like the *ACM*, *IEEE*, *DBLP* and *ePrints* in the domain of publications, *PubMed*, *GeneID*, *Drug Bank* and *Gen Bank* in life science, *GeoNames*, *Linked GeoData* and *Geo Linked Data* in the geographic domain, as well as *Last.FM*, *MySpace*, *BBC Music* and *Music Brainz* in the domain of media. Numerous other examples can be obtained from the Web of Data graph.

Considering that data providers and consumers need to have the ability to use their preferred schema in this kind of setting, it becomes obvious that systems supporting transparent querying over different datasets are essential components for a great number of Web of Data applications. Although many state of the art applications (e.g., LDIF [4], SPARQL++ [5], Mosto [6]) are focused on the RDF data exchange/transformation problem, to the best of our knowledge, there is no system supporting transparent querying over mapped RDF data sources.

In this paper, we present the SPARQL–RW (SPARQL–ReWriting) Framework. The SPARQL–RW provides a generic method for SPARQL query rewriting, with respect to a set of predefined mappings between ontology schemas. It supports a set of rich and flexible mappings types formally described using *Description Logics* (*DL*) and it is proved to provide semantics preserving queries.

Formally, let a source ontology O_S, a target ontology O_T and a set of mappings $\mathbf{M}$ between O_S and O_T. Our framework takes as input a SPARQL query Q_S expressed over O_S, and rewrites it to a *semantically correspondent* SPARQL query Q_T (expressed over O_T) with respect to $\mathbf{M}$. We have formally evaluated [16] the soundness and completeness of the proposed rewriting method with respect to the set of mapping types supported by our framework.

2. FRAMEWORK OVERVIEW

The architecture of the SPARQL–RW Framework is presented in Fig. 1. Our working scenario involves ontologies, as well as a set of predefined mappings between them. Our system exploits these mappings in order to rewrite an initial SPARQL query Q_S expressed over the source ontology, to a semantically correspondent SPARQL query Q_T, expressed over the target ontology.

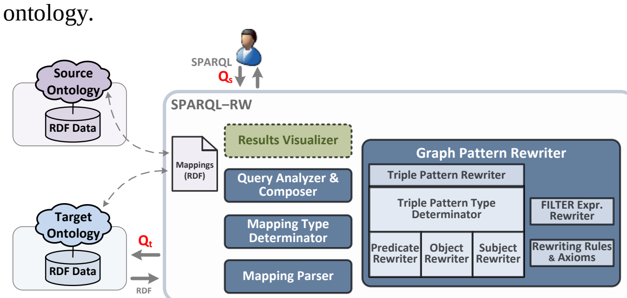

Fig. 1. The System Architecture

The system is divided into 4 basic components: (a) *Query Analyzer & Composer*, that analyzes the input SPARQL query and also composes the rewritten one; (b) *Mapping Parser*, that parses the predefined mappings; (c) *Mapping Type Determinator*, that identifies the type of each mapping in order to be exploited by the rewriting process; (d) *Graph Pattern Rewriter*, that rewrites the *Graph Pattern* of the input SPARQL query based on the specified mappings**.** Finally, for demonstration purposes, we have also integrated a *Results Visualizer* component which is responsible for the results presentation.

2.1 Mapping Model

In this section, we outline the mapping model adopted by the SPARQL−RW Framework in the context of SPARQL query rewriting.

Our attempt is to identify and support the set of mapping types which can be exploited by the SPARQL query rewriting process. This task is highly dependent to the SPARQL expressiveness. For instance, a mapping containing aggregates would be meaningless, since aggregates cannot be represented in the current SPARQL.

The proposed mapping model supports a highly expressive set of mapping types. To this end, it provides a grammar in order to describe these mapping types, as well as a formal definition of their semantics expressed in DL. Below we outline a fragment of the SPARQL−RW mapping capabilities.

In order to define the supported mapping types we introduce the following four basic notions: (a) the *Class Expression*; (b) the *Object Property Expression*; (c) the *Datatype Property Expression*; and (d) the *individual*. The above notions form the basis of our mapping model and result to *n:m* cardinality mappings, using either *equivalence* ($\equiv$) or *subsumption* ($\sqsubseteq$, $\sqsupseteq$) relationships.

Regarding ontology classes, a *Class Expression* from the source ontology can be mapped to a Class Expression from the target ontology. As Class Expression we denote any complex expression between classes, using *union* ($\sqcup$) and *intersection* ($\sqcap$) operations. A Class Expression can be restricted to the values of one or more *Property Expressions* (i.e., complex expression between object/datatype properties) using binary and unary predicates. Moreover, it is possible for a Class Expression to be restricted on a set of individuals having object/datatype property values with a specific relationship between them.

Regarding ontology object properties, an *Object Property Expression* from the source ontology can be mapped to an Object Property Expression from the target ontology. As Object Property Expression we denote any complex expression between object properties using *union* ($\sqcup$), *intersection* ($\sqcap$), *composition* ($\circ$) and *inverse* ($-$) operations. Any Object Property Expression can be restricted on its domain/range values using a Class Expression to define the applied restrictions.

Similarly, a *Datatype Property Expression* from the source ontology can be mapped to a Datatype Property Expression from the target ontology. As Datatype Property Expression we denote any complex expression between datatype properties using *union* ($\sqcup$) and *intersection* ($\sqcap$) operations, as well as *composition* ($\circ$) operations between object/datatype properties. Although Datatype Property Expressions can be restricted on their domain values with the same way as Object Property Expressions, their ranges can be restricted on data values only, using various unary predicates.

Finally, an individual from the source ontology can be mapped to an individual from the target ontology.

As noted before, we have formally described the semantics of the aforementioned mapping types using DL [16]. Since our query rewriting method is based on these mapping types, we provide no limitation on the language used for the mapping representation. As a result, any mapping language that supports the above mapping types (or a fragment of them) can be used. Additionally, we do not provide any limitation regarding the mapping discovery task, which can be performed either manually or automatically.

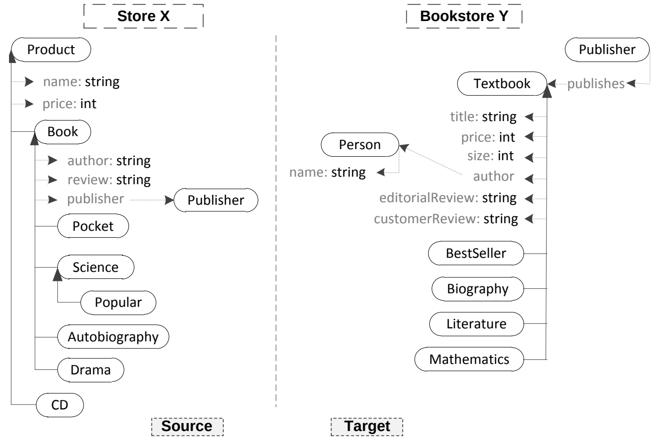

Fig. 2. The source schema `"Store X"` (Left Side) and the target schema `"Bookstore Y"` (Right Side)

2.1.1 Mapping Examples

In most real-world situations, an ontology schema is mapped to more than one ontology schemas. However, for the sake of simplicity but without loss of generality, in this section we consider two small ontology schemas, in order to present a set of mapping cases and thus, outline a fragment of the SPARQL−RW mapping capabilities.

Let the two hypothetically autonomous partners, `Store X` and `Bookstore Y`. `Store X` is a store providing information for its selling products (e.g., books, CDs, etc.) and `Bookstore Y` is a bookstore providing information for its book collections. In our example, `Store X` is considered to be the source ontology O_S, while `Bookstore Y` the target ontology O_T. Fig. 2 illustrates the structure of the two aforementioned ontology schemas.

Generally, several mappings of different types can be considered between `Store X` and `Bookstore Y`. Starting from class mappings, we say that the class `Popular` can be mapped to the intersection of the class `BestSeller` with the class `Mathematics` (μ_1). This mapping emerges from the fact that the class `Popular` seems to describe `Mathematics` individuals which are also of type `BestSeller`.

$$\mu_1\text{: src : Popular} \equiv \text{trg : BestSeller} \sqcap \text{trg : Mathematics}$$

Similarly, the class `Pocket` can be mapped to the class `Textbook` restricted on its size property values (μ_2), since the class `Pocket` seems to describe `Textbook` individuals having a specific value for the property `size` (e.g., less than or equal to 14 cm).

$$\mu_2\text{: src : Pocket} \equiv \text{trg : Textbook} \sqcap \exists \text{trg : size.}_{\leq 14}$$

Apart from class mappings, mappings between object/datatype properties can be also identified. For instance, the property `name` seems to subsume the property `title` (μ_3), while the object property `publisher` can be mapped to the inverse of the object property `publishes` (μ_4), since the binary relations described by the property `publisher` correspond with the inverse binary relations described by the property `publishes`.

$$\mu_3\text{: src : name} \sqsupseteq \text{trg : title}$$

$$\mu_4\text{: src : publisher} \equiv \text{trg : publishes}^{-}$$

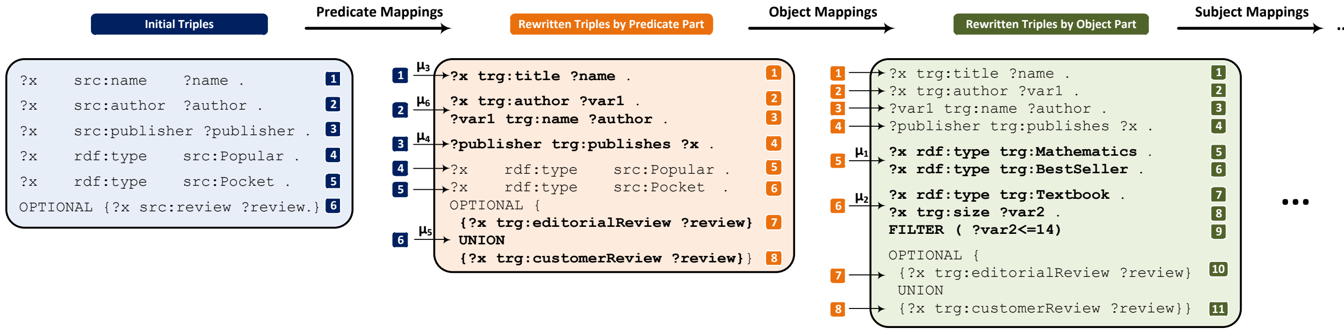

Fig. 3. SPARQL Graph Pattern Rewriting Process Example

Apart from these trivial property mappings, more complex ones can be also identified. For instance, the datatype property `review` can be mapped to the union of the datatype properties `editorialReview` and `customerReview` (μ_5), since the binary relations described by the property review correspond with the binary relations described by the properties `editorialReview` and `customerReview`.

$$\mu_5\text{: src : review} \equiv \text{trg : editorialReview} \sqcup \text{trg : customerReview}$$

Similarly, the datatype property `author` from the source ontology can be mapped to the composition of the object property `author` with the datatype property `name` from the target ontology (μ_6). This mapping emerges from the fact that the binary relations described by the datatype property `author` from the source ontology correspond with the binary relations provided by connecting the `Textbook` individuals to the `name` property values of the class `Person`.

$$\mu_6\text{: src : author} \equiv \text{trg : author} \circ \text{trg : name}$$

2.2 Query Rewriting

The SPARQL query rewriting process lies in the query's graph pattern rewriting and is performed by the *Graph Pattern Rewriter* component. The rewritten query is produced by replacing the initial query's graph pattern with the rewritten graph pattern. Any variables appearing in the initial query's graph pattern appear also in the rewritten graph pattern. The rewriting process is independent of the query type (i.e., `Select`, `Ask`, etc.), the SPARQL solution sequence modifiers (i.e., `Order By`, `Distinct`, etc.) and the SPARQL algebra operators (i.e., `Union`, `Optional`, etc.).

A SPARQL graph pattern consists of triple patterns, filters and SPARQL operators. Triple patterns may refer either to *data* (e.g. relationships between instances) or *schema* (e.g., relationships between classes and/or properties) information (i.e., *Data Triple Patterns* and *Schema Triple Patterns*) [16]. The *Triple Pattern Type Determinator* sub-component, identifies the type of each triple pattern. Based on this type, the *Triple Pattern Rewriter* sub-component rewrites triple patterns using a three-step procedure by exploiting mappings for each triple pattern's part (i.e., subject, predicate, object). The rewriting is performed by the *Subject*, *Predicate* and *Object Rewriter* sub-components by applying a set of rewriting rules [16] according to the type of the mapping which is exploited each time (*Rewriting Rules & Axioms* sub-component). The rewriting rules applied to Data Triple Patterns arise directly from the DL semantics defined for every different mapping type, while the rewriting rules applied to Schema Triple Patterns, are based on a set of common inference axioms. Filter expressions that may occur in the input query are rewritten by the *FILTER Expression Rewriter* component, using trivial expression-based rules. The rewriting rules have been formally presented in [16].

In Fig. 3, we outline a simple SPARQL graph pattern rewriting example, where the graph pattern of an initial query Q_S posed over the `Store X` ontology is rewritten to a semantically equivalent graph pattern, in order for the rewritten query Q_T to be expressed over the `Bookstore Y` ontology (Fig. 2). The rewriting process exploits the mappings (i.e., μ_1, μ_2, etc.) specified in Section 2.1.1.

2.2.1 Semantics Preservation

In this section, we outline the process that we followed in order to formally evaluate the soundness and completeness of the proposed query rewriting method. Since we are working in the context of different mapped datasets; the resulted query is heavily relied to the mappings which have been exploited by the rewriting method. As a result, any statement related to the soundness and completeness of our method should also consider the mapping semantics. In the rest of this section, we formally define the term "semantics preserving", and we outline the process that we followed in order to formally evaluate our method.

Let $[[\cdot]]_D$ be a graph pattern evaluation function which takes a graph pattern expression and an RDF dataset D and returns a set of graph pattern solutions, as defined in [14].

Moreover, let D_S and D_T be the RDF datasets of a source and a target ontology, respectively. Similarly, let D_U be the RDF dataset which is produced by *merging* [15] the D_S and D_T datasets using a set of mappings $\mathbf{M}$.

Definition 1. (Semantics Preserving Rewriting). Let *tp* be a triple pattern and *rp* the graph pattern resulted from one step rewriting of *tp* with respect to a mapping $\mu \in \mathbf{M}$. The rewriting step performed for *tp*, with respect to the mapping μ, is *semantics preserving* iff the evaluation of *tp* and the evaluation of *rp* over D_U, preserve the semantics of mapping μ.

In other words, let $\mathbf{V}$ be the common variable set between *tp* and *rp*. The relationship $\mathbf{R}$ (i.e., $\equiv$, $\sqsubseteq$, $\sqsupseteq$) that holds for the mapping used in the rewriting step, should also hold between $[[tp]]_{D_U}$ and $[[rp]]_{D_U}$ projected on $\mathbf{V}$.

$$\pi_V([[tp]]_{D_U})\ \mathbf{R}\ \pi_V([[rp]]_{D_U}),\text{ where }\mathbf{R}\in\{\equiv,\sqsubseteq,\sqsupseteq\} \qquad \square$$

Following the above definition and using the mapping type semantics that we have defined, along with the SPARQL semantics, we have formally proved [16] that every rewriting step that we perform in order to rewrite an initial SPARQL query, is semantics preserving.

3. IMPLEMENTATION & DEMONSTRATION

In what follows we provide technical information about the implementation of our system and we outline the demonstration scenario.

3.1 Implementation

The SPARQL–RW Framework has been implemented using Java 2SE as a software platform, and the Jena framework for SPARQL query manipulation. The SPARQL–RW Framework is a part of the *Semantic Query Mediation Prototype Infrastructure* (*SQMPI*) developed in the *TUC/MUSIC* Lab. Additionally, the *SQMPI* has been integrated with our XS2OWL [13] and SPARQL2XQuery [12] frameworks, in order to support integration and interoperability between the XML and the Semantic Web environments [17].

Finally, regarding the mapping representation and encoding, we utilize the *EDOAL* language (*Expressive and Declarative Ontology Alignment Language*)[1], since it is expressive enough, in order to cover all the different mapping types that our framework supports.

3.2 Demonstration Outline

In this section, we outline the scenario employed to demonstrate the applicability of the SPARQL–RW Framework.

In our demonstration scenario, except from a discussion regarding the major technical and theoretical challenges we faced throughout the development of the SPARQL–RW Framework, attendees will be able to have an in depth experience of mapping different RDF/S – OWL schemas, express queries over their corresponding data and observe the query rewriting process via an interactive user interface.

In more detail, attendees will be able to (a) select an ontology set, between various overlapping ontologies; (b) specify mappings between the previously selected ontologies; (c) view/modify the specified mappings in order to observe the affection on the rewriting process; (d) specify SPARQL queries based on the source ontology, in order to be rewritten, with respect to the predefined mappings, to semantically equivalent SPARQL queries (valid over the target ontology); (e) have a thorough look on the SPARQL query rewriting, via the system interface which provides interactive step-by-step navigation to the rewriting procedure; (f) evaluate both the initial query and the rewritten query over the source and target ontologies respectively, in order to inspect the returned results.

4. RELATED WORK

Our work can be related to several research fields, including (semantic) data integration, schema mediation, ontology mapping and query rewriting. Among the aforementioned categories we consider the fields of ontology mapping and query rewriting as the most relevant to our work.

Ontology mapping, has received extensive attention by the Semantic Web community especially in the tasks of mapping discovery and mapping representation. This paper does not contribute to neither of these tasks. Our focus is on the specification of those types of ontology mappings which can be exploited by the SPARQL query rewriting process (i.e., can be supported by the SPARQL expressiveness).

Regarding SPARQL query rewriting, few published studies examine the problem of posing a SPARQL query over different RDF datasets. An approach [9] which comes closer to ours, with some of its parts based on a preliminary description of our work [10], proposes a method that exploits transformations between RDF graphs in order to perform SPARQL query rewriting. Compared to our method, this approach seems to restrict the mappings expressiveness and also the supported query types.

Finally, some recent efforts address the problem of federated SPARQL query evaluation over linked data [1][2][3], while others [7][8] examine the problem of query rewriting using views in semantic web databases.

5. CONCLUSIONS

Systems supporting transparent querying over different datasets managed by different organizations and accessed through SPARQL are essential for many Web of Data applications. Such a system was presented in this paper. The SPARQL–RW Framework supports SPARQL query rewriting with respect to a set of predefined mappings between ontologies. Using this infrastructure, users can express SPARQL queries based on their own OWL–RDF/S schema and automatically access data across a federation of RDF resources over the Web.

Acknowledgment. This work was partially supported by the FP7 project Natural Europe (Project Ref. No 250579, Area CIP-ICT-PSP.2009.2.5 – Digital Libraries), where we investigated how we can apply the *SQM Prototype Infrastructure* in the context of the Natural Europe federation.

6. REFERENCES

[1] Schwarte A., Haase P., Hose K., Schenkel R., Schmidt M.: "FedX: Optimization Techniques for Federated Query Processing on Linked Data". In ISWC 2011.

[2] Quilitz, B., Leser, U.: "Querying distributed RDF data sources with SPARQL". In ESWC 2008.

[3] Hartig O., Bizer C., Freytag J.C.: "Executing SPARQL queries over the web of linked data". In ISWC 2009.

[4] Schultz A., Matteini A., Isele R., Bizer C., Becker C.: "LDIF - Linked Data Integration Framework". In 2nd International Workshop on Consuming Linked Data 2011.

[5] Polleres A., Scharffe F., Schindlauer R.: "SPARQL++ for Mapping Between RDF Vocabularies". In ODBASE 2007.

[6] Rivero C. R., Hernandez I., Ruiz D., Corchuelo R.: "Generating SPARQL Executable Mappings to Integrate Ontologies". In ER 2011

[7] Goasdoue F., Karanasos K., Leblay J., Manolescu I.: "View selection in semantic web databases". In PVLDB, 5(1), 2012.

[8] Le W., Duan S., Kementsietsidis A., Li F., Wang M.: "Re-writing queries on SPARQL views". In WWW 2011.

[9] Correndo, G., Salvadores, M., Millard, et al.: "SPARQL Query Rewriting for Implementing Data Integration over Linked Data". In 1st Int. Workshop on Data Semantics 2010.

[10] Makris K., Bikakis N., Gioldasis N., Christodoulakis S.: "Towards a mediator based on OWL and SPARQL". In WSKS 2009.

[11] Makris K., Gioldasis N., Bikakis N., Christodoulakis S.: "Ontology Mapping and SPARQL Rewriting for Querying Federated RDF Data Sources". In ODBASE 2010.

[12] Bikakis N., Gioldasis N., Tsinaraki C., Christodoulakis, S.: "Querying XML Data with SPARQL". In DEXA 2009.

[13] Tsinaraki C., Christodoulakis S.: "Interoperability of XML Schema Applications with OWL Domain Knowledge and Semantic Web Tools". In ODBASE 2007.

[14] Perez J., Arenas M., Gutierrez C.: "Semantics and Complexity of SPARQL". ACM Trans. Database Syst. (TODS) 34(3) 2009.

[15] Noy N.F., Musen M.A.: "PROMPT: Algorithm and tool for automated ontology merging and alignment". In: AAAI/IAAI 2000.

[16] Makris K., Gioldasis N., Bikakis N., Christodoulakis S.: "SPARQL Rewriting for Query Mediation over Mapped Ontologies". T.R. 2010 *http://www.music.tuc.gr/reports/SPARQLREWRITING.PDF*

[17] Bikakis N., Tsinaraki C., Gioldasis N., Stavrakantonakis I., Christodoulakis S.: "The XML and Semantic Web Worlds: Technologies, Interoperability and Integration. A survey of the State of the Art" In Semantic Hyper/Multi-media Adaptation: Schemes and Applications, Springer 2012 (to appear).

[1] *http://alignapi.gforge.inria.fr/edoal.html*