SPARQL Query Mediation for Data Integration

Xiaoqing Zheng
School of Computer Science
Fudan University
zhengxq@fudan.edu.cn

Xitong Li
Sloan School of Management
Massachusetts Institute of Technology
xitongli@mit.edu

Stuart E. Madnick
Sloan School of Management
Massachusetts Institute of Technology

ESD-WP-2012-02 January 2012 esd.mit.edu/wps

SPARQL Query Mediation for Data Integration

Xiaoqing Zheng
Xitong Li
Stuart Madnick

Working Paper CISL# 2011-06

November 2011

Composite Information Systems Laboratory (CISL)
Sloan School of Management, Room E62-422
Massachusetts Institute of Technology
Cambridge, MA 02142

SPARQL Query Mediation for Data Integration

Xiaoqing Zheng
School of Computer Science
Fudan University
zhengxq@fudan.edu.cn

Xitong Li
Sloan School of Management
Massachusetts Institute of Technology
xitongli@mit.edu

Stuart E. Madnick
Sloan School of Management
Massachusetts Institute of Technology
smadnick@mit.edu

Abstract

The Semantic Web provides a set of promising technologies to make sophisticated data integration much easier, because data on the semantic Web is allowed to be connected by links and complex queries can be executed against the dataset of those linked data. Although the Semantic Web techniques offer RDF/OWL to support schematic mappings between diverse data sources, large-scale data integration is still severely hampered by various types of data-level semantic heterogeneity among the data sources. In the paper, we show that SPARQL queries that are intended to execute over multiple heterogeneous data sources can be mediated automatically.

1. Introduction

For several decades relational technology has been the most successful data management solution for business information processing. New requirements for data modeling and integration are creating needs beyond the modeling and processing capabilities of the relational technology. The relational model works well for semantically homogenous worlds in which all views of the same entity are consistent because it is based on first-order logic (FOL) and everything is required to be consistent under FOL. However, the real world in which we live is complicated and usually inconsistent. Due to design autonomy there can exist multiple views of an entity for different purposes. As a result, the conflict between the inherent semantic heterogeneity of databases to be integrated and the semantic homogeneity of relational model assumption leads to a great increase in the cost of designing, developing and maintaining relational integration solutions [1].

The integration of heterogeneous databases requires identification and reconciliation of the conflicts between schema objects that describe similar or equivalent concepts. The schema of the relational model often represents an informal agreement between the users and developers in a singular task-specific environment. Thus, the schemas are usually not designed to be shared or reused, which also makes them difficult to be integrated [2]. The Semantic Web provides promising technologies to make such integration much easier due to the ability and flexibility of its abstract data model, the Resource Description Framework (RDF). Relational databases can be accessed by their RDF views through mapping of relational database to RDF [3] [4]. Notice that although it might be easy to add columns or tuples to relational tables, it can be very troublesome to add integrity constraints or delete columns from the tables. In contrast, it is quite easy to add or delete nodes in RDF graphs and to merge two RDF graphs (can be considered as two schemas) into one, these are features which make the RDF model more suited to integrate or combine different data sources. The ontology layer of the Semantic Web stack also offers the facility to create an overarching contextualization that supports mappings between related schemas.

Using native RDF data, or data has been exported through their RDF views, the SPARQL query language can be used as an integration tool to pose rich queries on the dataset that integrates data from different sources. Note that the data export does not necessarily mean physical conversion of the data, and the data can be generated on-the-fly from relational database or other non-RDF sources at query time. However, much existing RDF data takes the form of simple values (or literals) for properties such as lengths, prices, etc. The contextually-dependent information such as unit and format is often omitted, and there is no standard for how to represent the context information by using RDF. Since each data source and the potential receiver of queries over the dataset may operate with a different context, it is generally not safe to make the assumption that anyone accessing the dataset will understand the units and formats being used. A typical example might be the extraction of price information from the databases: but is the price in Dollars or Yuan (if dollars, is it US dollars or Hong Kong dollars), does it include taxes, does it include shipping?

This paper presents an approach to automatically detecting and reconciling data-level semantic conflicts among multiple RDF data sources and rewriting naive SPARQL queries issued against those RDF sources so that correct results can be returned to end users. The data conflicts handled are formatting, unit, encoding, scale inconsistencies, etc. Unless the values are correctly and dynamically converted into a uniform presentation, operations on dataset can lead to wrong answers (see Section 2). The proposed solution also allows users (we call receivers) to specify a context with respect to their queries so that the query can be framed with data in different presentations from that in the RDF data sources.

2. Motivational Examples

DBpedia[1] is a community effort to extract structured information from Wikipedia and to make this information available on the Semantic Web. The DBpedia knowledge base allows you to ask interesting queries, like "Give me the rivers that flow into the Rhine and are longer than 50 kilometers". The SPARQL query can be formulated as shown on the left of Figure 1.

NAÏVE SPARQL QUERY

```
1: PREFIX dbo: <http://dbpedia.org/ontology/>
2: SELECT    ?river    ?length
3: WHERE { river dbo:type :River ;
4:                       dbo:length ?length ;
5:                       dbo:riverMouth ?mouth ;
6: FILTER ( ?length >= 50 ) .
7: FILTER regex(?mouth, "rhine", "i") .    }
```

CONTEXT: RECEIVER

- *Length is expressed in kilometers.*

CONTEXT: SOURCE

- *Length is expressed in meters.*

Figure 1. DBpedia example

The SPARQL query, however, does not take into account the fact that both receiver and source operate with different contexts. Specifically, the user works with kilometers, but the lengths of rivers are recorded using meters in the source. The answer returned would be wrong unless Line 6 of the query is transformed into "FILTER (?length >= 50000)". It is still not enough because the answers returned should be further transformed to the context of the receiver.

[1] http://wiki.dbpedia.org/

Consider another simple example of finding the countries whose GDP is more than 50 trillion Chinese Yuan (CNY) against the Semantic Web version of the CIA Factbook[2] database as shown in Figure 2. The query will return almost all of the countries on the planet because the GDPs are reported in US dollars with a scale-factor of 1 in the CIA Factbook database and 50 dollars is a very tiny number comparing to the GDP of any country. For this query it is not enough to just rewrite 50 into 50×10^{12} as we did in the DBpedia example since the context of the receiver differs from the source not only in the scales but also in the currencies.

NAÏVE SPARQL QUERY

```
PREFIX fk:
   <http://www4.wiwiss.fu-berlin.de/factbook/ns#>
PREFIX db:
   <http://www4.wiwiss.fu-berlin.de/factbook/resource/>
SELECT   ?country   ?gdp
WHERE {
   ?country fk:GDP_purchasingpowerparity ?gdp .
   FILTER ( ?gdp >= 50 ) }
```

CONTEXT: RECEIVER

- *Currency is CNY with a scale-factor of trillion.*

CONTEXT: SOURCE

- *Currency is USD with a scale-factor of 1.*

Figure 2. CIA Factbook example

There is only one source involved in the above two examples. If multiple data sources originating from different contexts are brought together and we can pose queries on the whole dataset, the situations would become much more complex and many semantic conflicts would occur. The semantic conflicts would occur when the constraints typed by receivers are tested against data sources or the values from different sources need to be compared.

NAÏVE SPARQL QUERY

```
1:    PREFIX foaf: <http://xmlns.com/foaf/0.1/>
2:    PREFIX xbrl: <http://www.xbrl.org/>
3:    SELECT   ?pname     ?bprofit     ?percent
4:    WHERE {
5:      GRAPH ?graph1
6:        { ?comp  foaf:name ?pname ;
7:                 xbrl:branch ?branch ;
8:                 xbrl:profit ?pprofit . }
9:      GRAPH ?graph2
10:       { ?branch xbrl:profit ?bprofit . }
11:     LET (?percent := ( ?bprofit / ?pprofit )) .
12:     FILTER ( ?percent >= 0.50 ) .
13:     FILTER ( ?bprofit >= 20000 ) . }
```

CONTEXT: RECEIVER

- *Currency is USD with a scale-factor of 1,000.*

CONTEXT: SOURCE1 (USA)

- *Currency is USD with a scale-factor of 10,000.*

CONTEXT: SOURCE2 (Japan)

- *Currency is JPY with a scale-factor of 10,000.*

CONTEXT: SOURCE3 (China)

- *Currency is CNY with a scale-factor of 10,000.*

Figure 3. XBRL example

Suppose there are three data sources that record the financial information about companies from United States, China, and Japan, using different currencies and scale-factors as shown on the right of Figure 3. In this example, all the data sources are included as named graphs. The query is to find the companies where more than 50% of their profits come from their oversea branches and the profits of the branches are greater than twenty million USD. Line 11 of the query is used to calculate the percentage of the profits of oversea branches to the total profits of their parent companies. The variables ?bprofit and ?pprofit might be bound to the values from different sources. How the values bound to ?bprofit and ?pprofit are transformed so that the two transformed values

[2] http://www4.wiwiss.fu-berlin.de/factbook/

conform to the same context and can be used to calculate correct percentage depends on which sources are matched by the patterns in the query.

SPARQL language can be used to express queries across diverse data sources, and the different parts of query patterns can be matched against different data sources. If the GRAPH keyword is followed by a variable (like XBRL example), the variable will range over all the named graphs (or data sources) in the query's dataset, which makes the SPARQL query language a powerful integration tool, unlike SQL in which we need to indicate the tables from which data is to be retrieved and the full attribute names.

The earlier DBpedia example shows that some constraints typed by users should be transformed to comply with assumptions in the data source context. In the CIA Factbook example the data might vary in two or more aspects (in that case, currency and scale) so there must be more than one conversion of the data. The XBRL example shows that this conversion should be made in dynamic way, depending on which data source is accessed. Besides, the answers should be further transformed so that they conform to the context of the receiver.

3. System Architecture

To resolve the problems illustrated in the above examples, we develop a SPARQL query mediation system in which the semantics of data (of those recorded in a source, or expected by a receiver) can be explicitly represented in the form of a set of *context statements*. As shown in Figure 4, queries submitted to the system will be intercepted by a *Context Mediator*, which rewrites the naïve SPARQL query to a *mediated SPARQL query* and passes it to *SPARQL query engine*. Then, the *SPARQL query engine* dispatches subqueries to individual sources and undertakes conversions when data are exchanged between two data sources. Eventually, the correct answers are provided to the receiver.

Context ontology is a collection of generic concepts, which provides a common type system for describing data semantics exchanged between disparate sources. As shown in Figure 5, the concepts have some attributes, called *modifiers*, which serve as annotations that make explicit the semantics of data in different aspects. The context ontology defines what types of modifiers apply to which concepts. The context ontology could be seen as a knowledge representation framework, which is used to define the contexts of receivers and data sources. Following the XBRL example, the user needs to create a *context instance* of the concept "*MonetaryValue*" by the following two statements:

```
US_dollar_ thousand   hasScale   "1000"^^xsd:integer .
US_dollar_ thousand   hasCurrency   "USD"^^xsd:string .
```

where "US_dollar_thousand" is used to identify the instance, "hasScale" and "hasCurrency" are two modifiers, and "1000" and "USD" are values of the modifiers. The above statements explicitly indicate that any data value associated with this instance is in USD currency with a scale-factor of thousand. Such instances will be stored in a *context pool*, and can be reused by others. Then, the user can declare that for the XBRL application if a data value is "MonetaryValue" type (profit belongs to this type), he assumes that it is reported in USD currency with a scale-factor of thousand by the statement below.

```
xbrl   monetaryValue   USD_dollar_ thousand .
```

The contexts of the data sources can be defined in a similar way, and the only difference is that the attribute of data sources (not the name of application such as "xbrl") should be associated with the context instance.

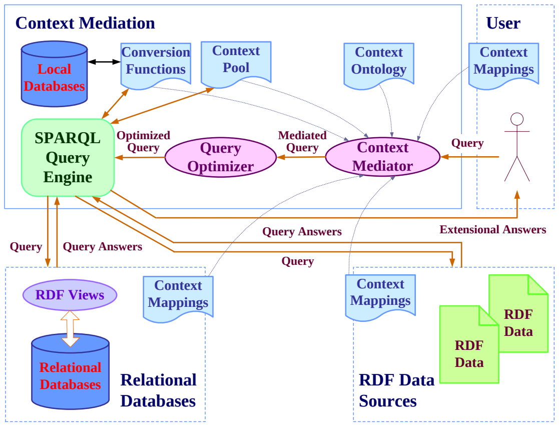

Figure 4. System architecture

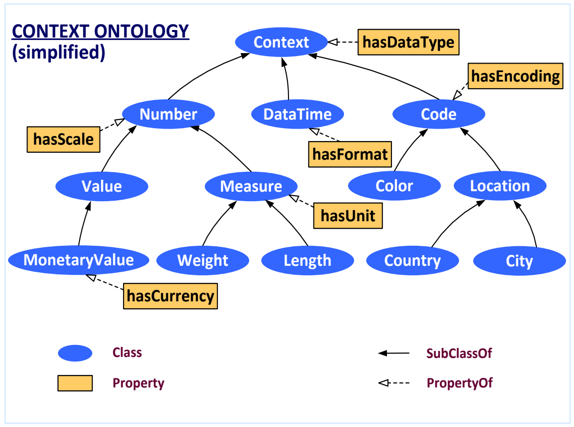

Figure 5. An illustration of context ontology

The preceding statements are not yet sufficient for resolving the conflicts of data present in disparate contexts. *Conversion functions* are introduced for each modifier to achieve conversions between different contexts. A function cvt_modifier(*mvs*, *mvt*, *vs*) returns the equivalent value *vt* that complies with assumptions in the target context for the source value *vs*, where *mvs* and *mvt* are two distinct values of the modifier in the source and target contexts respectively. We have algorithms that use atomic functions (simple conversions such as conversion between USD and CNY or between scale 1,000 and scale 10,000) to be automatically composited to construct composite conversions (to convert from 10,000 CNY to 1,000 USD.). As it can be seen in the CIA Factbook example, the data vary in two aspects: currency and scale. In the mediated query, a composite function will be automatically constructed to reconcile those two conflicts through a series of invocations on the conversion functions defined on each conflicting aspect.

4. Query Mediation

The context mediator uses query rewriting technique to undertake the role of detecting and reconciling potential conflicts at the time a query is submitted. The following clauses of queries will be rewritten to the corresponding forms that all semantic conflicts, when detected, are resolved:

- SELECT: the answers returned should be further transformed so that they conform to the context of the receiver.
- WHERE: the constraints typed by the receiver should be transformed to comply with assumptions in the data source contexts.
- FILTER and FUNCTION: one or more arguments might be transformed so that all the arguments conform to the same context.

The mediation process starts by converting the naïve query (i.e. query ignoring semantic differences between sources and receivers) into its well-formed query. Then, all the semantic conflicts are detected and the corresponding conversion functions will be constructed. Finally, the naive query is rewritten into a logically equivalent mediation query. This work is inspired by a long-standing research project, Context Interchange strategy [5]. We use Jena[3] to implement the query mediation system and demonstrate its feasibility. It also can easily be plugged in to other existing SPARQL query engines to take advantage of their state of the art query optimization and execution techniques, because query mediation is separated from query optimization and execution in our system and the mediated queries encode all the necessary data transformations. We had tested the solution with the real data from CIA Factbook and DBpedia.

5. Conclusion

With this system, the users are not burdened with the diverse data semantics in data sources, all of which are declared in the context representation component and automatically taken into consideration by the mediator. A user in any context is able to issue queries over any set of data sources in other contexts as if they were in the same context. The idea for employing context ontology and conversion rules for resolving semantic heterogeneity might be not new, but our work highlights the data-level semantic heterogeneity among various linked RDF data sources, a challenge which has not been fully recognized by the Linked Data pioneers. With the proposed mediation system, multiple data sources with disparate contexts can be integrated by SPARQL query mediation, in which case the users can just construct queries in their own contexts with no need to understand the potential semantic conflicts. The system supports and allows the data sources to dynamically participate in or quit the system and sets target for facing more open environment. Adding or removing a data source is accomplished by adding or removing the corresponding context declarations, which does not require any changes to the mediator or query processers. Most importantly, the sources are not required to make any change or commit to any criteria; they only need to record data semantics declaratively. Another possible use of the system is to transform or standardize data from multiple sources for data warehouse generation. When combining data from different sources with appropriate context definitions, the system can help rewrite the conversion rules and quickly adapt to the change of the data sources.

Acknowledgement: This work was supported, in part, by a grant from the National Natural Science Foundation of China (No. 60903078).

References

[1] Brodie, M.L. The power and limits of relational technology in the age of information ecosystems. Talk at MIT CSAIL Seminar, December 9, 2010.

[2] George, D. Understanding structural and semantic heterogeneity in the context of database schema integration. *Journal of the Department of Computing, UCLAN*, 4, (2005). 29-44.

[3] Keio. Data integration on Semantic Web, 2006. Retrieved July 26, 2011, from http://www.w3.org/People/Ivan/CorePresentations/DataIntegration/Slides.html.

[4] Sahoo, S.S., Halb, W., Hellmann, S., Idehen, K., et al. A survey of current approaches of mapping of relational database to RDF, 2009. Retrieved April 25, 2011, from http://www.w3.org/2005/Incubator/rdb2rdf/RDB2RDF_SuveyReport.pdf.

[5] Goh, G.H., Bressan, S., Madnick, S., Siegel, M. Context Interchange: New Features and Formalisms for the Intelligent Integration of Information. *ACM Transactions on Information Systems*, 17, 3, (1999), 270-293.

[3] http://jena.sourceforge.net/