Query Processing in a Three-Level Ontology-Based Data Integration System

João C. Pinheiro
Universidade Federal do Ceará
Department of Computing, Brazil
Department of Informatics - IFMA
– São Luís, MA – Brazil
joaoslz@lia.ufc.br

Vânia M. P. Vidal
Universidade Federal do Ceará
Department of Computing, Brazil
vvidal@lia.ufc.br

Marco A. Casanova
Department of Informatics – PUC-Rio
– Rio de Janeiro, RJ – Brazil
casanova@inf.puc-rio.br

José A. F. Macêdo
Universidade Federal do Ceará
Department of Computing, Brazil
jose.macedo@lia.ufc.br

Fábio A. M. Porto
Laboratório Nacional de Computação Científica – LNCC
– Rio de Janeiro, RJ – Brazil
fporto@lncc.br

Eveline R. Sacramento
Universidade Federal do Ceará
Department of Computing, Brazil
Fundação Cearense de Meteorologia e Recursos Hídricos - Brazil
eveline@lia.ufc.br

1. INTRODUCTION

According to Lenzerini [13], the main components of a data integration system (DIS) are: the schema of the mediated view; the schemas of the sources where the real data are stored; and the mapping that specifies the correspondences between the mediated schema and the source schemas.

One of the problems resulting from data integration is how to specify such mappings, and two basic approaches have been proposed with this objective [13]. The first one, called global-as-view (GAV), requires that to every element of the mediated schema, a view over the data sources is associated, so that its meaning is specified in terms of the data stored at the sources. The second one, called local-as-view (LAV), requires the mediated schema to be specified independently from the sources. In turn, the sources are defined as views over the mediated schema.

These mappings can be classified according to their accuracy in: sound, exact or complete [13]. A view may be sound, when all the data it provides satisfies the corresponding element of the mediated schema, but there may be additional data satisfying the element not provided by the view. A view is complete, when not all the data it provides satisfies the corresponding element of the mediated schema, but all data satisfying the element is provided by the view; and a view is exact when all the data it provides satisfies the corresponding element of the mediated schema, and all data satisfying the element of the mediated schema is provided by the view [6].

The other problem resulting from data integration is how to use the obtained mappings to answer correctly the queries posed on the mediated schema [13]. Typically, mappings are used to define how to translate data from a data source into another, preserving the semantics of the data or; alternatively, to rewrite a query posed on a data source into an equivalent query over another data source. Furthermore, no matter whether the query is issued or whether the data is shared, the semantic differences between the data sources need to be reconciled. So, reconciling semantic heterogeneity is a key issue in any data integration architecture.

The problem of semantic heterogeneity is exacerbated when

ABSTRACT

In this paper, we present a three-level ontology-based framework for effectively designing GAV data integration systems. In our approach, the mediated schema is represented by a domain ontology, which provides a conceptual representation of the application. Each local source is described by an application ontology, whose vocabulary is restricted to be a subset of the vocabulary of domain ontology. The three-level architecture permits dividing the mapping definition in two stages: local mappings and mediated mappings. Due to this architecture the problem of query answering can also be broken into two steps. First, the query is decomposed, using the mediated mappings, into a set of elementary sub-queries expressed in terms of the application ontologies. Then, these sub-queries are rewritten, using the local mappings, in terms of their local sources schemas. This paper focuses on a method for query processing that addresses the problem of efficient query answering. Our approach is illustrated by an example of a virtual store mediating access to online bookstores.

Categories and Subject Descriptors
H.2 [Database Management]: Distributed Databases - query processing, database integration, XML/XSL/RDF.

General Terms
Algorithms, Design.

Keywords
Semantic Web, ontologies, data integration, schema mappings, query processing, RDF, SPARQL.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

Copyright 2010 ACM 978-1-4503-0421-4/10/11...$10.00.
dealing with semi-structured data, due to its flexibility in adding new attributes and, consequently, generating schema variations. From a data integration perspective, ontologies provide a possible approach to address the problem of semantic heterogeneity [23]. In this sense, the idea is to use ontologies to formally describe the semantics of the data sources.

Recent research has also used ontologies for specifying the mediated schema in the context of data integration [8][9][12][17]. An important challenge in ontology-based data integration systems is the problem of rewriting a query specified in terms of the domain ontology into sub-queries that can be answered by individual data sources.

In this article, we present an ontology-based framework for integrating data provided by multiple heterogeneous data sources. The mediated schema is represented by a domain ontology, defined in RDF/OWL format, which provides a conceptual representation of the domain. The semantics of each data source is described in an application ontology, also in RDF/OWL format, using a subset of the shared vocabulary of the domain ontology. This format allows us focusing on the logical structure of the date, in contrast to the structural format of XML or the storage format of the relational databases.

Then we show that application ontologies, simplify the definition of the mediated mappings, thereby facilitating our query rewriting process. In our work, a SPARQL query is first posed on the domain ontology. In order to define a generic and automatic approach for processing distributed SPARQL queries, we adopt the SPARQL algebraic formalism presented in [1] for both defining the mediated mappings, and for query processing.

The main contribution of this work is a method for query answering a SPARQL query submitted over a domain ontology into sub-queries over multiples data sources using a set of mappings expressed in an algebraic formalism.

The remainder of this article is structured as follows. Section 2 introduces the basic definitions used in this paper. Section 3 describes our ontology-based framework for data integration. Section 4 describes the running example. Section 5 presents our query answering method and illustrates it with an example. Section 6 presents related work. Finally, Section 7 contains the conclusions and discusses future research.

2. BASIC DEFINITIONS

RDF Data Model: RDF [14] is a general model language, optimized for data sharing and interchange. The easiness of data interchange arises from some characteristics of this language, like the RDF graph structure, the simple structure of the basic units of these graphs, and the global namespace provided by the use of IRIs (Internationalized Resource Identifiers).

In RDF, all data items encode knowledge facts, and they are represented in the form of RDF triples (subject, predicate, object). Predicates encode binary relationships between a subject and an object, and they are labeled with IRIs. Each IRI and literal has a global scope. The use of global names is critically important, because it means that the triples can always be merged without name translations. Since each part of the statement in a graph can be used without translation, entire graphs can be transported and combined without any translation, which is a great advantage when exchanging data.

Indeed, since RDF triples need no translation when moving from one system to another, they are valid in any context. These triples are completely self-contained assertions of information, and as such, they are independent one from one another. This independence means that their order is irrelevant.

OWL: The Web Ontology Language (OWL) [2] describes classes and properties in a way that facilitates machine interpretation of Web content. The description of OWL is organized as three dialects: OWL Lite, OWL DL and OWL Full.

An OWL schema is a collection of RDF triples that use the OWL vocabulary. A concept of an OWL schema is a class, datatype property or object property defined in the schema. The vocabulary of the schema is the set of concepts defined in the schema (a set of IRIs). The scope of a property name is global to the OWL schema, and not local to the class indicated as its domain.

In this work we use OWL Lite. It supports named classes, datatype and object properties, subclasses, and individuals. The domain of a datatype or object property is a class. As property restrictions, the dialect admits minCardinality and maxCardinality, with the usual meaning; and InverseFunctionalProperty, which resemble the notion of a key in databases, for object properties.

SPARQL Query Language: SPARQL (SPARQL Protocol and RDF Query Language) [18] is a W3C standard recommendation. It is a declarative query language that allows extracting data from RDF graphs based on a graph pattern matching, whose basic constructs are triple patterns. An example of a triple pattern is (?p, ?s, ?o). As an example, consider the following query (applied over the Publishers application ontology presented in Figure 5):

```
PREFIX pub: <http://example.org/publishers/>
SELECT ?name, ?phone
FROM <publishers.owl>
WHERE {
  ?p s:name ?name .
  OPTIONAL ( ?p s:phone ?phone )
  FILTER regex(?country, "USA") .
}
ORDER BY ?name ?phone
```

Figure 1. Simple SPARQL query.
in [7], offer means to group a set of datasets in a graph, and then refer to this graph using an IRI. A SPARQL query may specify the dataset by using a FROM clause (as a default graph), or a FROM NAMED clause to name the referring RDF dataset. If there is no FROM clause, but there is one or more FROM NAMED clauses, then the dataset includes an empty RDF graph as a default graph.

For example, the SPARQL query in Figure 2 asks for the name and phone of the publishers, and also their publisher into two RDF datasets (Amazon and Publishers) using FROM NAMED clauses. These datasets (application ontologies) are presented in Figure 5.

```sparql
PREFIX am: <http://example.org/amazon/>
PREFIX pub: <http://example.org/publishers/>
SELECT ?pb am:name ?name
WHERE{
  GRAPH <amazon.owl>{
    ?pb am:hasPub ?pb .
    ?pb am:name ?name .
  }
  GRAPH <publishers.owl>{
    ?pub pu:name ?name .
  }
}
```

**Figure 2. Named Graph SPARQL query.**

### 2.3 SPARQL Algebra

Arenas et al. [1] developed an algebra for SPARQL. This algebra defines the semantics of a SPARQL graph pattern. Every SPARQL query string is mapped to a SPARQL algebra expression. The authors also conduct an extensive analysis of the semantics and the complexity of SPARQL, focusing on two operators of SPARQL: UNION and OPTIONAL. In the following, we provide an algebraic formalization of the core fragment of SPARQL over RDF.

Let $I$ be the set of IRIs, $L$ be the set of RDF Literals, $B$ be the set of blank nodes, and $T = I \cup L \cup B$ be the set of RDF terms called *SPARQL Algebra Expressions*. There exists an infinite set of variables $V$ disjoint from $T$.

**Definition 2.1:** A SPARQL graph pattern expression can be defined recursively as follows:

1. A Basic Graph Pattern (or BGP) is a graph pattern expression.
2. If $P_1$ and $P_2$ are graph patterns, then the expressions $(P_1 \text{ AND } P_2)$, $(P_1 \text{ OPT } P_2)$, and $(P_1 \text{ UNION } P_2)$ are graph pattern expressions (called join graph pattern, optional graph pattern, and union graph pattern, respectively).
3. If $P$ is a graph pattern expression and $R$ is a SPARQL built-in condition, then the expression $(P \text{ FILTER } R)$ is a graph pattern expression.
4. If $P$ is a graph pattern expression and $X \in I \cup L \cup V$, then $(X \text{ GRAPH } P)$ is a graph pattern expression.

### 3. A FRAMEWORK FOR ONTOLOGY-BASED DATA INTEGRATION

In this section, we present our mediated environment. As we said before, our approach uses ontologies for formally describing the data sources, and also the mediated schema. Figure 3 describes the main components of the proposed mediated environment, adapted from [21].

The mediated schema is represented by a domain ontology (DO), which provides a conceptual representation of the domain (a global shared vocabulary) and a set of constraints over the domain ontology. Each local source schema is described by an application ontology (AO) whose vocabulary is restricted to be a subset of the vocabulary of the domain ontology.

In our work, the application ontologies help breaking the query answering problem, and they are a notational convenience to divide the definition of the mappings into two stages: mediated mappings and local mappings. The mediated mappings define each element of the domain ontology as an algebraic query over one or more application ontologies; whereas the local mappings define the concepts of the application ontologies in terms of the elements of their corresponding local source schemas. Application ontologies enable the identification and the association of semantically corresponding concepts, so they are useful for enhancing tasks such as information discovery and retrieval, and also data integration. Finally, the *Inverse Functional Axioms* play a key role to deal with data integration, as we show in the next section.

We adopt OWL Lite [2] as the ontology language to represent the DO and the AOs. We use SPARQL query language [18] for posing queries on the domain ontology. SPARQL has been chosen, as it is a recent W3C recommendation query language for RDF. In this work, we rewrite an initial query submitted over the DO using an algebraic formalism of SPARQL [1]. The mediated mappings are also represented in this algebraic formalism, whereas the local mappings are represented by a set of correspondent assertions. The local source schemas are accessed via wrappers, like the ones presented in [3][4], which export the local data into RDF/OWL format.

The following definition formally introduces the notion of a mediated environment.

**Definition 3.1:** A mediated environment is a 5-tuple $ME = (DO, S_1, AOs, \gamma, \gamma')$, $k=1,...,n$, where:

- DO is a domain ontology, which represents the mediated schema. We assume that the classes and properties in DO are...
\( C_1, \ldots, C_u \) and \( P_1, \ldots, P_v \), respectively.

- for each \( k = 1, \ldots, n \),
  - \( S_k \) is a local source schema.
  - \( AO_k \) is an application ontology, which formally describes the local source schema \( S_k \). The vocabulary of \( AO_k \) is a subset of the vocabulary of DO. We adopt namespace prefixes to distinguish the occurrence of a symbol in the DO vocabulary from the occurrence of the same symbol in the vocabulary of \( AO_k \). We assume that: For each class \( C_i \) (or property \( P_j \)) in the vocabulary of DO, we denote the occurrence of \( C_i \) (or \( P_j \)) in the vocabulary of \( AO_k \) by \( AO_k;C_k \) (or \( AO_k;P_k \)).
  - \( \gamma \) is a set of correspondence assertions, called *local mappings*, relating the concepts of \( AO_k \) with the elements of \( S_k \). We consider that the views defined by the local mappings are *exact*.

- \( \gamma \) is the *mediated mappings*. The mapping \( \gamma \) is defined in the GAV approach: to each concept \( C \) of DO, we associate a SPARQL algebra expression \( \rho(C) \) over the application ontologies; \( \rho(C) \) is defined as the union of \( \rho(C_k) \) over the application ontologies; \( \rho(C) \) is defined as the union

### 4. THE RUNNING EXAMPLE

In this section, we describe how our mediated environment allows the integration of data originated from autonomous, heterogeneous and distributed data sources, using an example of a virtual store mediating access to online booksellers. We assume that the user provides a domain ontology, and that we have two application ontologies describing data about Amazon and eBay virtual stores, and a third application ontology describing book publishers.

Figure 4 shows a conceptual representation of the Sales domain ontology. We use the namespace prefix "s:" to refer to the vocabulary of this domain ontology. For example, "s:Book" is defined as a datatype property with domain "s:Product" and range "string", "s:Book" is declared as a subclass of "s:Product", and "s:hasPub" is defined as an object property with domain "s:Book" and range "s:Publ".

Figure 5 shows a conceptual representation of the application ontologies. We use the namespace prefixes "am:，“eb:” and “pub:” to refer to the vocabularies of Amazon, eBay and Publishers application ontologies, respectively. In Figures 4 and 5, we use a simplified UML version of class diagrams instead of RDF, for readability reasons.

Figure 6 shows the *inverse functional axioms* of the Sales domain ontology. Based on the *inverse functional axioms* of the DO, one can automatically identify links [5] relating entities in different application ontologies. For example, axiom A2 specifies that given an instance \( X \) of "am:Publ" and instance \( Y \) of "am:Publ", if \( X \) and \( Y \) have the same name, then \( X \) and \( Y \) represent the same entity of the real world. More formally: am:Publ(X), am:name(X, n), pub:Publ(Y), pub:name(Y, n) \( \rightarrow \) same-as (X, Y). So, the axiom A2 virtually specifies links relating instances of "am:Publ" and "pub:Publ". As we show next, the identification of links plays an important role in the generation of the mediated mappings.

**The mediated mappings** is defined based on a set of correspondence assertions which relates DO concepts with AO concepts. A correspondence assertions has the form: \( q_0 \leftarrow q_1 \), where \( q_0 \) is a query over the DO and \( q_1 \) is a query over the applications ontologies. Intuitively, an assertion \( q_0 \leftarrow q_1 \) specifies that the concept represented by the query \( q_0 \) over the applications ontologies correspond to the concept in the DO represented by the query \( q_1 \). We define two types of assertions:

1. **Homogeneous assertion**, in which \( q_0 \) is over a single application ontology. These assertions are automatically generated based on the vocabulary of the application ontologies. Figure 7 shows the homogeneous mappings referring to our case study.

2. **Heterogeneous assertion**, in which \( q_0 \) is over two application ontology. These assertions are automatically generated based on the inverse functional axioms and homogeneous mappings. Figure 8 shows the heterogeneous mappings referring to our case study. For example, the assertion A27 is generated from assertion A22 and domain axiom A2.

The mediated mappings is defined in the GAV approach: to each concept \( C \) of DO, we associate a SPARQL algebra expression \( \rho(C) \) over the application ontologies; \( \rho(C) \) is defined as the union
of the queries \( q_A \) in the assertions for \( C \). For example, the query for the class \( s:country \) is given by:

\[
\rho(s:country) = (\{?pub pub:country ?country\} \cup (\{?pub am:name ?name\} \land (\{?pub1 pub:name ?name\} \land (\{?pub1 pub:country ?country\}))
\]

### Class Assertions:

1. \( (?p rdf:type s:Product) \rightarrow (\{?p rdf:type am:Product\}) \)
2. \( (?p rdf:type s:Product) \rightarrow (\{?p rdf:type eb:Product\}) \)
3. \( (?p rdf:type s:Book) \rightarrow (\{?p rdf:type am:Book\}) \)
4. \( (?p rdf:type s:Book) \rightarrow (\{?p rdf:type eb:Book\}) \)
5. \( (?p rdf:type s:Music) \rightarrow (\{?p rdf:type am:Music\}) \)
6. \( (?p rdf:type s:Music) \rightarrow (\{?p rdf:type eb:Music\}) \)
7. \( (?p rdf:type s:Publ) \rightarrow (\{?p rdf:type am:Publ\}) \)
8. \( (?p rdf:type s:Publ) \rightarrow (\{?p rdf:type eb:Publ\}) \)
9. \( (?p rdf:type s:Publ) \rightarrow (\{?p rdf:type eb:Publ\}) \)

### Property Assertions:

10. \( (?p s:title ?title) \rightarrow (\{?am am:title ?title\}) \)
11. \( (?p s:title ?title) \rightarrow (\{?eb eb:title ?title\}) \)
12. \( (?p s:description ?description) \rightarrow (\{?eb:description ?description\}) \)
13. \( (?p s:priceAmaz ?priceAmaz) \rightarrow (\{?am:price ?priceAmazon\}) \)
14. \( (?p s: priceEbay ?priceEbay) \rightarrow (\{?eb:price ?priceEbay\}) \)
15. \( (?p s:hasPub ?pub) \rightarrow (\{?am:hasPub ?pub\}) \)
16. \( (?p s:hasPub ?pub) \rightarrow (\{?eb:hasPub ?pub\}) \)
17. \( (?p s:name ?name) \rightarrow (\{?am:name ?name\}) \)
18. \( (?p s:name ?name) \rightarrow (\{?eb:name ?name\}) \)
19. \( (?p s:name ?name) \rightarrow (\{?pub pub:name ?name\}) \)
20. \( (?p s:address ?address) \rightarrow (\{?am:address ?address\}) \)
21. \( (?p s:address ?address) \rightarrow (\{?eb:address ?address\}) \)
22. \( (?p s:country ?country) \rightarrow (\{?am:country ?country\}) \)
23. \( (?p s:phone ?phone) \rightarrow (\{?am:phone ?phone\}) \)
24. \( (?p s:homepage ?homepage) \rightarrow (\{?am:homepage ?homepage\}) \)

### Diagrams

**Figure 7.** Homogeneous Correspondence Assertions.

**Figure 8.** Heterogeneous Correspondence Assertions.

**Figure 9.** Mapping tree for \( \rho(s:country) \).

Due to space limitations, the local sources schemas and the local mappings are not presented here. The work in [22] presents a strategy to automatically generate application ontologies, local and mediated mappings, considering a domain ontology, a set of local source schema, and the result of the matching between each local schema and the domain ontology.

### 5. The Proposed Query Answering Method

The ultimate goal of a data integration system is to answer queries posed by the user in terms of the mediated schema. In this section, we sketch our query answering method (more details can be found in [16]). The query answering method basically unfolds domain ontologies into application ontology terms using the previous specified mediated mappings. Each mediated mapping is parsed and represented as a mapping tree such as illustrated in Figure 9, which represents the mediated mapping with respect to country domain ontology concept. The input of our method is the submitted query using domain ontology vocabulary and the set of mappings tree.

The proposed query processing strategy consists of four steps, summarized as follows:

1. **Translation.** Firstly, the user poses a SPARQL query in terms of a domain ontology. This initial query is turned into a parse tree representing the structure of the query in a useful way.
2. **Unfolding.** The obtained parse tree is expanded into a new parse tree by using the mediated mappings, which are a combination of sub-queries over the relevant application ontologies. Each sub-query aims at extracting data from a single application ontology. All information that is relevant for answering the query is discovered in this step.
3. **Optimization.** This step attempts to find the most desirable operator evaluation tree for generating the execution plan, according to the following objectives: (i) the potential gain in performance from having several nodes processing different parts of the query in parallel, and (ii) the costs of data transmission over the network, in order to reduce the amount of data transferred.
4. **Evaluation.** The sub-queries over the application ontologies are rewritten in terms of their corresponding local source schemas, with the help of the local mappings; and then they are delegated to evaluation at the local data sources. The results of these sub-queries are returned to the mediator.
where the final result is combined to properly answer to the
query according to the execution plan. Pipeline evaluation can
be used to produce an efficient query processing. The main
goal of a pipeline evaluation is to eliminate the need of
shipping, storing, and retrieving foreign relations and/or
intermediate results during the processing of joins.

We now give an example of how a SPARQL query is processed.

Example: Consider a query Q over the Sales domain ontology
that asks for titles and names of their publishers; for all books
whose publishers’ country is USA. Figure 10 shows this query in
SPARQL syntax:

```sparql
PREFIX s:<http://example.org/sales/>
PREFIX rdf:<...>
SELECT ?title, ?name
WHERE
  ?p s:hasPub ?pub .
  ?pub rdf:type s:Publ .
  ?pub s:country ?country
  FILTER regex(?country, "USA") .
}
```

Figure 10. Query in SPARQL syntax.

Each processing step of the SPARQL query Q is illustrated as follows:

1. Translation. The query, written in SPARQL, is parsed, that is,
turned into a query tree representing the structure of the
query as illustrated in Figure 11.

2. Unfolding. This step can be summarized as follows:

   a. Each domain ontology term in the parse tree is substituted
      by its corresponding mediated mapping. The final result
      of unfolding operation is the tree illustrated in Figure 12.

   b. Then, we restructure the query tree by applying the
distributive property over union (Figure 13), in order to
separate and express each query over a single application
ontology. The query tree in Figure 13, consists of tree sub-
queries Q1, Q2 and Q3, over the Amazon, eBay and
Publishers application ontologies, respectively.

```sparql
PREFIX s:<http://example.org/sales/>
PREFIX rdf:<...>
SELECT ?title, ?name
WHERE
  ?p s:hasPub ?pub .
  ?pub rdf:type s:Publ .
  ?pub s:country ?country
  FILTER regex(?country, "USA") .
}
```

Figure 11. Parse tree for query Q.

Figure 12. Query tree after the unfolding

The generated query tree represents the following SPARQL query
illustrated in Figure 14(a). The query tree is transformed into a
query plan tree showed in Figure 14(b). The final query plan tree
specifies how the results of sub-queries, which aim at extracting
data from single application ontology, can be combined to
properly answer to the query. However, there is no information
about the order of execution, which is defined by the optimization
step.

```sparql
Q' = { ?title, name },
  ?p rdf:type s:Book
  ?p s:hasPub ?pub
  ?pub rdf:type s:Publ
  ?pub s:country ?country
  FILTER regex(?country, "USA") .
} UNION
  ?p rdf:type s:Book
  ?p s:hasPub ?pub
  ?pub rdf:type s:Publ
  ?pub s:country ?country
  FILTER regex(?country, "USA") .
} UNION
  ?p rdf:type s:Book
  ?p s:hasPub ?pub
  ?pub rdf:type s:Publ
  ?pub s:country ?country
  FILTER regex(?country, "USA") .
} UNION
```

Figure 13. Query tree after applying distributive operation
over union.

Figure 14. (a) SPARQL query in the algebraic formalism
representing the query tree in Figure 13 (b) Query plan tree.
3. Optimization. In the third stage, the query plan tree is analyzed and a query plan is generated. Note that, in Figure 14(b), the independent queries Q1 and Q2, can be executed in parallel, whereas Q2, and Q3 are dependent since the result of Q3 will be used to select the country names on Publishers ontology and Q2 is executed with a selection clause of Q3.

To reduce the amount of data transferring, the join operation is implemented using the semi-join [11] strategy, as follows:

(i) Q3 extracts the name of the publisher located in the USA;
(ii) Q2 is executed with a selection clause based on the name of the publisher obtained in the previous step.

Then it is done a union with the result of Q1 to compose the final result.

4. Evaluation. This step can be summarized as follows: (i) the sub-queries defined over the application ontologies are rewritten in terms of their corresponding local source schemas, with the help of the local mappings; and (ii) the result of such sub-queries are returned to mediator, where the final result is built according to the optimized execution plan.

6. RELATED WORK

Research in distributed SPARQL query processing only started recently. Two known projects have used SPARQL to provide integrated access to distributed RDF data sources: DARQ [19] and SemWIQ [12]. The first one, DARQ (acronym for Distributed ARQ), provides a single interface for querying distributed query services. The DARQ engine decomposes a SPARQL query into sub-queries, then ships these sub-queries to their corresponding query services, and finally integrates the results of such sub-queries. However, this approach has some limitations currently, as for example, it only supports queries with bound predicates; what means that unions and left-outer-joins (Optional Pattern Matching) operations are not supported yet, requiring the user to explicitly supplies a catalog (a service description) that includes, for instance: the total number of triples, the cardinalities and the selectivities. However, these statistics are no longer representative when the corresponding data are changed. Similarly to the DARQ approach, the SemWIQ (acronym for Semantic Web Integrator and Query Engine) [12] system contains a mediator service that transparently distributes the execution of SPARQL queries.

These approaches are not accompanied by mappings described in SPARQL algebra. This prevents the use of their technique to automatically decompose algebraic queries over multiple data sources, and the consequent composition of such query results. In contrast, our approach, adopts a SPARQL algebraic formalism to define the mappings and it ensures that a semantic query is properly rewritten over the data sources.

In [9], it is provided an ontology-based approach to the integration of heterogeneous XML documents that transforms heterogeneous XML sources into local RDF ontologies, which are then merged into a RDF global ontology. Note that this work does not define an approach for the unification of the results that are returned from different data sources. There are also many efforts in using the research results of querying general RDF graph models. Among these graph data model proposals, the work presented in [10] describes an object consolidation algorithm, which analyses inverse functional properties, and that is used to identify and merge equivalent instances in a RDF dataset.

Calvanese et al. [8] present an algorithm for answering queries submitted over a data integration system. It follows the GAV approach and assumes that the views associated to the elements of the mediated schema are sound. However, in this approach, the query processing is more complex than in traditional GAV systems, as the presence of integrity constraints in the mediated schema, implies in the need of taking the semantics of such constraints into account during query execution. This algorithm uses integrity constraints and mappings for, respectively, inferring additional information in the query (query expansion) and rewriting the query over the sources. This way, extracting information in this approach is similar to query answering with incomplete information, which is a difficult task.

Calvi et al. [6] present an algorithm for query rewriting using GAV sound views that takes into account a set of foreign key constraints over the mediated schema. This work also considers the problem of query answering with incomplete information, and it also adopts the strategy of query expansion. It shows that the presence of constraints in the mediated schema can make the query processing difficult, even in the GAV approach.

7. CONCLUDING REMARKS AND FUTURE WORK

In this paper, we presented a framework for ontology-based integration of distributed and heterogeneous data. This framework takes a SPARQL query submitted over a domain ontology, and rewrites it into sub-queries over multiple data sources. The query’s result is obtained by the proper combination of data obtained from these sub-queries. We have illustrated, through an example, how it allows the combination of data from different sources, thus overcoming some limitations of other approaches that worked with distributed query processing using SPARQL.

As a future work, we intend to formally prove that our query rewriting process is correct, using the adopted algebraic formalism. We also intend to investigate the data fusion problem, which consists in identifying and merging equivalent instances provided by multiple RDF datasets. We will implement and evaluate the proposed query answering method. In addition, we plan to study optimization step in depth and define heuristics to help generate cost-effective plans. For reaching this objective, we need to evaluate, in a near future, the proposed framework in a real world setting. These experiments will provide empirical evidences of the scalability of our query answering approach using sets of very large ontologies.

8. ACKNOWLEDGMENTS

This research has been partially supported by the grant 620207/2008-6 from CNPq – Brazil.

9. REFERENCES


Recommendation. Available at: http://www.w3.org/TR/owl-ref/.


