Peer-to-Peer Semantic Integration of XML and RDF Data Sources

Isabel F. Cruz, Huiyong Xiao, and Feihong Hsu

Department of Computer Science
University of Illinois at Chicago, USA
{ifc, hxiao, fhsu}@cs.uic.edu

Abstract. Peer-to-Peer (P2P) data management systems combine traditional schema-based integration techniques with the P2P infrastructure. In this paper, we propose a P2P data management framework named PEPSINT that semantically integrates heterogeneous XML and RDF data sources, using a hybrid architecture and a global-as-view approach. Our focus is on the query processing techniques over heterogeneous data. Queries in PEPSINT are expressed in XQuery and in RDQL. We consider two types of queries, depending on whether the query is first posed on the super peer or on one of the peers.

1 Introduction

The Semantic Web has been proposed to add semantics to web content and to enable interoperability among heterogeneous data sources. Both Extensible Markup Language (XML) and Resource Description Framework (RDF) can be used to represent information on the Web. However, there exists a wide gap between the two languages, since RDF data has *domain structure* (the concepts and the relationships between concepts) while XML data has *document structure* (the hierarchy of elements) [11].

An example is shown in Figure 1, in which the RDF schema R explicitly specifies two concepts, `Book` and `Publisher`, as well as the `publishedBy` relationship. Figure 1 also shows two XML schemas S_1 and S_2. Each of these XML schemas contains two concepts: `book` and `author` (equivalently denoted by `article` and `writer` in S_2). Conceptually, these two XML schemas are quite similar. Structurally speaking, however, they are very different: S_1 (book-centric schema) has the `author` element nested under the `book` element, whereas S_2 (author-centric schema) has the `article` element nested under the `writer` element.

Furthermore, the wide diversity of possible XML schemas for a single conceptual model also results in wide diversity for the XML queries. For instance, a user who wants to "List all the publications" from two data sources corresponding to S_1 and S_2 may write the XML path expressions, respectively, as `/books/book/@booktitle` and `/writers/writer/article/@title`. We notice that although the two XML path expressions refer to semantically equivalent concepts, they follow two distinct XML paths. In contrast, schemas defined on

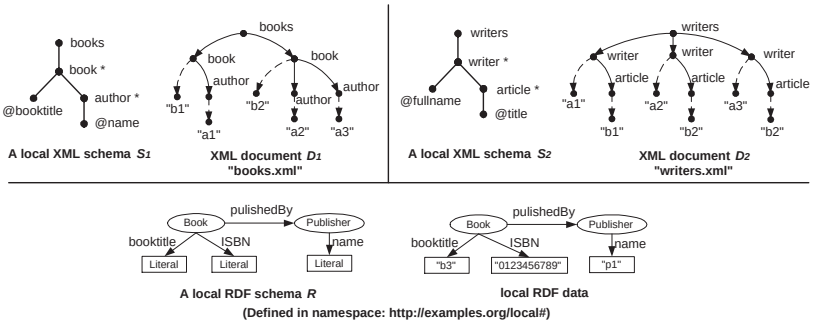


Fig. 1. An example of heterogeneous XML and RDF data sources.

the conceptual level (known as *conceptual schemas* or ontologies) are *flat* in document structure, and therefore the user can formulate a query without considering the structure of the source (we refer to such queries as *conceptual queries*). RDF Schema (RDFS), DAML+OIL, and OWL are examples of languages used to create conceptual schemas.

There are currently several attempts to use *conceptual schemas* [1, 2, 8, 9] and *conceptual queries* [6, 7] to overcome the problem of structural heterogeneities among XML sources. In this paper, we propose a framework called PEPSINT (PEer-to-Peer Semantic INTegration framework) to semantically integrate heterogeneous XML and RDF data sources in a P2P environment. We discuss the architecture of PEPSINT, and present a solution for semantic integration and query processing in the P2P heterogeneous environment. In brief, we make the following contributions in this paper:

- We propose a P2P schema-based data management framework, PEPSINT, built on a *hybrid* P2P architecture, in which the global RDF ontology (constructed using the global-as-view approach [13]) in the *super peer* behaves not only as a central control point over the *peers* but also as a mediator for query translation from peer to peer.
- For the purpose of semantic integration, we propose an approach that preserves the domain structure of RDF and the document structure of XML. Specifically, the semantic integration of XML and RDF data sources is implemented at the schema level (through the schema matching process) and at the instance level (through the query answering process).
- We also provide a set of query rewriting algorithms that can propagate a user's query across the heterogeneous XML or RDF data sources in PEPSINT. In our framework, mappings connect the peer to the super peer, thus making query processing within the network transparent to a user in any peer.

The paper is organized as follows. Section 2 gives a review of related work. In Section 3 we describe the architecture of PEPSINT and its main components. Section 4 discusses schema-based integration of RDF sources and (structurally

dissimilar) XML sources. Query processing in PEPSINT is covered in Section 5. Finally, we draw conclusions and discuss future work in Section 6.

2 Related Work

The research community has, to date, produced several P2P data management systems that aim to enable interoperability among distributed heterogeneous data sources.

The **Edutella** project [15] provides an RDF-based metadata infrastructure for P2P networks based on the JXTA framework [10]. In Edutella, connections between peers are encoded into a network topology known as the *Edutella super-peer topology*, which is similar to the hybrid architecture used in PEPSINT. A Datalog-based query exchange language called RDF-QEL is proposed to serve as a common query interchange format. Thus a wrapper translates local query languages such as SQL and XPath into RDF-QEL. Edutella does not support XML sources directly, though the RDF data sources may be serialized in XML format.

PeerDB [16] is an agent-based P2P data management system where each peer holds a relational database. The metadata for relations that are sharable with other peers is specified in a local *export dictionary*. Unlike PEPSINT, there are no established mappings between peers. Thus, query reformulation between peers in PeerDB is assisted by agents through a *relation-matching strategy*; this is a process of matching the metadata between relations in different peers. XML and RDF data are not considered in the current implementation of PeerDB.

SEWASIE [4] is another agent-based P2P system that aims to integrate Information Nodes (SINodes), where each node acts as an autonomous mediator-based system. It contains two types of agents: *query agents* that are responsible for query processing and answering; and *brokering agents* (peers) that handle the mappings between nodes. Each brokering agent directly controls at least one SINode and handles the creation and maintenance of semantic relationships among concepts from different information nodes in the system. SEWASIE does not currently support RDF data sources.

Hyperion [3] proposes an architecture for a P2P data management system for relational databases (one stored at each peer). Similarly to PEPSINT, *mapping tables* and *mapping expressions* (mapping tables that allow variables) are used to store connections between local schemas in peers. A *query manager* uses the mapping tables and mapping expressions to rewrite a query posed in terms of the local schema; the rewriting process produces a query that is run over the schema of acquainted peers. Unlike PEPSINT, only relational data sources and relational queries are supported by Hyperion.

The **Piazza** system [11] is a P2P data management system that, like PEPSINT, supports interoperation of both XML and RDF data sources. Furthermore, both systems preserve document structure of XML sources during interoperation of these sources. The differences from PEPSINT are: (1) Piazza is based on the pure P2P architecture in which peers are connected directly, whereas PEPSINT

is built on top of a hybrid architecture with a super peer containing the global ontology. This is a tradeoff between efficiency and autonomy [4]. (2) Piazza uses a (declarative) XQuery-based mapping language for mediating between nodes, whereas PEPSINT utilizes mapping tables to store schema correspondences, which we believe results in easier construction and maintenance of mappings. (3) The Piazza system achieves its interoperability in a low-level (syntactic) way, i.e., through the interoperability of XML and the XML serialization of RDF. For this reason, the user has to write an RDF query in terms of an XQuery. The query rewriting in Piazza is based on pattern matching between an XQuery expression and the mappings. In contrast, PEPSINT supports RDF queries at the conceptual level (RDQL), as well as XQuery. Query translation is realized by a collection of query rewriting algorithms.

3 The PEPSINT Architecture

There are two types of P2P architectures [14]: the *pure P2P architecture*, in which no central point of control exists and peers are autonomous but can communicate directly with each other; and the *hybrid P2P architecture* that contains at least one central point of control. The global control point(s) maintain either network control or the references to the remaining peers. Based on the hybrid P2P architecture, PEPSINT contains two types of peers: the *super peer*, containing the global RDF ontology, and the *peers*, containing local schemas and local data sources. Each peer represents an autonomous information system and connects with the super peer by establishing P2P mappings. As shown in Figure 2, the PEPSINT architecture has four main components.

XML to RDF wrapper. Since XML is characterized by having a hierarchical document structure while RDF has a flat document structure, it is hard for the user to directly map a local XML schema to the global RDF ontology. To solve this problem, an *XML to RDF wrapper* is used to transform the XML schema into a local RDF schema, which is then mapped to the global ontology. This is a process that conceptualizes the XML elements into RDF concepts while keeping their nesting information (by using a specialized RDF property).

Local XML and RDF schemas. The local XML and RDF schemas residing in peers contain both data and metadata. For the purpose of semantic integration, we represent a local RDF schema as a labeled digraph (from now on referred to as *RDF schema graph*). The domain structure is explicitly represented by labeled vertices (concepts) and labeled arcs (relationships between concepts). Likewise, a local XML schema is represented as a labeled tree (from now on referred to as *XML schema tree*) that specifies nesting relationships between labeled vertices (elements).

Global RDF ontology. The global RDF ontology in the super peer is a virtual mediated schema integrated from distributed local RDF schemas (using the global-as-view approach [13]). In PEPSINT, the global ontology has two roles: (1) It provides the user with a uniform and complete view of data sources in the distributed peers; and (2) it serves as a mediator for query translation from

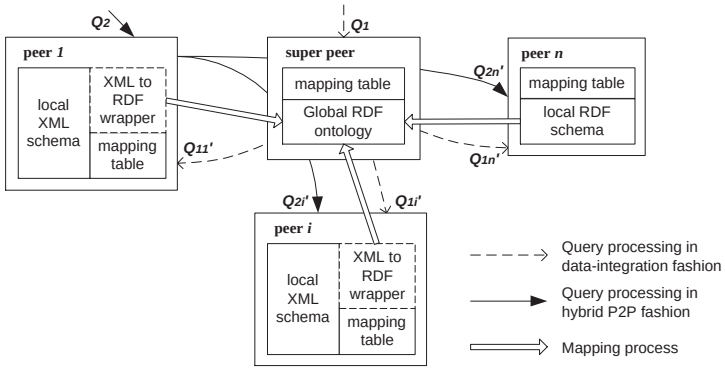


Fig. 2. The PEPSINT architecture.

one peer to other peers. The global RDF ontology is a fairly simple ontology—it does not contain high-level axioms, such as those available to DAML+OIL or OWL.

Mapping table. A mapping table stores mappings between local schemas and the global ontology. We use *XML path expressions* to represent the elements contained in an XML schema, and *RDF path expressions* to represent the concepts and relationships in an RDF schema.

The operation of PEPSINT can be divided into two phases: *mapping (or design) phase* and *query (or runtime) phase*, as respectively indicated by the hollow arrowed lines and the solid and dashed arrowed lines in Figure 2. To realize semantic integration of XML and RDF data sources, domain structure and document structure must be preserved in both phases.

1. Mapping phase. Whenever a new peer joins the PEPSINT network, the peer gets registered and indexed in the super peer by establishing mappings from its local schema to the global ontology. The mappings are established through a process of *schema matching* [1] and stored in the mapping table of the peer. During the process of schema matching, the global ontology is extended by integration of the local schemas. As previously mentioned, the domain structure and document structure of local schemas are encoded in the mappings.

2. Query phase. PEPSINT provides two query processing modes. (1) In the *data-integration mode*, the user poses a query (*source query*) on the global ontology in the super peer, which is then reformulated into multiple subqueries (*target queries*) over the XML and RDF sources in the peers (one subquery for each source). By executing the target queries and integrating their results, the system returns an answer to the user at the site of the super peer. (2) In the *hybrid P2P mode*, the user can pose a source query on the local XML or RDF

[1] Schema matching is a basic problem in many database application domains, and currently it must be performed manually. A taxonomy covering most of the existing approaches to schema matching has been devised [17].

source in some peer. Locally, the query will be executed on the local source to get a *local answer*. Meanwhile, the source query is reformulated into a target query over every other peer through transitive mappings (compositions of mappings from the original peer to the super peer and mappings from the super peer to the other target peers). By executing the target query, each peer returns an answer to the original peer, called the *remote answer*. The local and remote answers are integrated and returned to the user at the site of the originating peer.

Query translation is achieved by using the mappings in conjunction with a collection of query rewriting algorithms. We discuss the mapping and query phases in greater detail in Section 4 and Section 5, respectively. Running examples based on the schemas in Figure 1 will be used for illustration.

4 Mapping Process

In PEPSINT, the data sources residing at the peers may be either XML data modeled by an XML schema language (e.g., XML Schema) or else RDF data whose classes and properties are described using RDF Schema (RDFS). As previously mentioned, mappings between local schemas and the global ontology are established by the schema matching process during the registration of a peer to the super peer. The key operation in this process is the preservation of the domain structure of RDF sources and the document structure of the XML sources.

4.1 Mapping a local RDF schema to the global RDF ontology

Schema matching takes the global RDF ontology G (in the super peer) and a local RDF schema R (in the peer) as the inputs and returns a set of mappings M between the elements of G and the elements of R as the output. Meanwhile, the global ontology is updated by merging or adding metadata from the local RDF schema.

Elements in an RDF schema include *concepts* and *roles* (also known as *classes* and *properties* in RDFS terminology). When matching the local RDF schema with the global RDF ontology, for each element p_L in the local RDF schema, if there already exists in the global ontology a semantically equivalent element p_G, the two elements will be merged and a correspondence such as (p_L, p_G) will be generated. Otherwise, the element p_L will be copied into the global ontology as p_G, and a correspondence (p_L, p_G) will be generated as well. We define a group

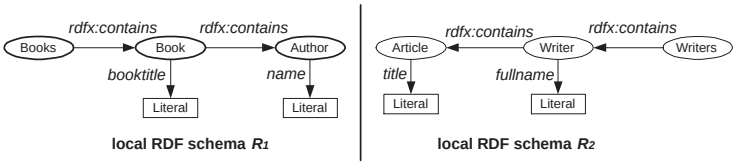
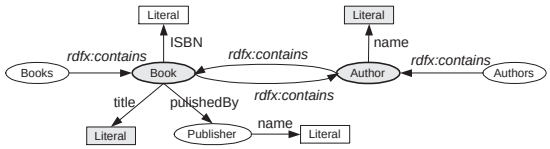


Fig. 3. RDF schemas transformed from the local XML schemas in Figure 1



Global RDF ontology *G* (defined in namespace: http://examples.org/global#)

RDF path expressions in G	**RDF path expressions in** R	**XML path expressions in** S_1	**XML path expressions in** S_2
Books	–	/books	–
Book	Book	/books/book	/writers/writer/article
Book.title	Book.booktitle	/books/book/@booktitle	/writers/writer/article/@title
Book.ISBN	Book.ISBN	–	–
Book.publishedBy	Book.publishedBy	–	–
Publisher	Publisher	–	–
Publisher.name	Publisher.name	–	–
Authors	–	–	/writers
Author	–	/books/book/author	/writers/writer
Author.name	–	/books/book/author/@name	/writers/writer/@fullname

Fig. 4. The global RDF ontology and its mapping table.

of operations on the ontology to implement schema matching between two RDF schemas, e.g., *merging of classes*, *merging of properties*, *merging of relationships between classes*, and *copying a class and/or its properties*. A concrete example is given in our previous work [9].

4.2 Mapping a local XML schema to the global RDF ontology

By transforming the participating local XML schema into a local RDF schema, we can convert the problem of matching an XML schema with the global ontology into the problem of matching an RDF schema with the global ontology, which is discussed in Section 4.1.

The schema transformation is carried out by the XML to RDF wrapper. The XML to RDF wrapper converts XML attributes and simple elements to RDF properties; it converts XML complex elements to RDF classes. The wrapper also encodes the element-attribute relationship and the element-subelement relationship in XML schema respectively as the *class-to-literal* relationship and the *class-to-class* relationship in the resulting RDF schema.

We choose to define a new, specialized RDF property *rdfx:contains* (the prefix *rdfx* stands for the new name space "`http://pepsint.org/rdfx#`") to explicitly denote nesting relationships. In particular, given that two XML elements e_i (parent element) and e_j (child element) are respectively converted into two RDF classes, c_i and c_j, the property *rdfx:contains* of c_i is then generated to connect c_i to c_j. Figure 3 shows the resulting local RDF schemas R_1 and R_2 that are respectively converted from the two XML schemas S_1 and S_2 shown in Figure 1. Finally, the global ontology G integrated from S_1, S_2 and R (in Figure 1) and its mapping table are shown in Figure 4. The grayed concepts or roles are the ones merged from local sources. We notice that both the *rdfx:contains* property in G

and the mappings in the mapping table encode the document structure of XML sources, so that either of them can be exploited for tracking XML document structure in future query translations.

5 Query Processing

5.1 Assumptions

For the simplicity of discussion, we make the following assumptions.

1. We assume the mappings from a local schema to the global ontology are total, one-to-one mappings. On the other hand, the mappings from the global ontology to the whole set of local schemas are total but not one-to-one mappings, since a concept in the global ontology might be merged from multiple concepts of different local schemas (as a result of schema matching). The mappings from the global ontology to a single local schema are one-to-one but they may be partial mappings, which means a query run at a local source may result in an incomplete answer.

2. We also assume that XML queries conform to a subset of XQuery [5], which we call PXQuery (*Partial XQuery*) in this paper. PXQuery consists of a non-nested FLWR expression that includes four clauses: `for`, `let`, `where`, and `return`; the `where` clause may only contain comparison operators. Other limitations of PXQuery include: (1) Only a single XML document is involved in the query; (2) No new XML fragments are introduced in the query; (3) The path expressions contained in the clauses only use child axes; (4) No type declarations, functions, `order` clauses, and predicate filters are used.

3. To represent RDF queries, we use RDQL, which uses an SQL-like syntax [12]. RDQL consists of the following clauses: `SELECT`, `FROM`, `WHERE`, `AND`, and `USING`. We assume only comparison operators are used in the AND clause of the RDQL query. The `FROM` and `USING` clauses are not the focus of our attention since they are not involved in query translation.

For the sake of convenience, we associate a PXQuery query Q with (V_{Q^R}, V_{Q^W}, C_Q), where V_{Q^R} and V_{Q^W} are the two sets that respectively contain all XML path expressions in the `return` clause and in the `where` clause, and C_Q contains the constraints whose items are in the form of vRc, where $v \in V_{Q^W}$, c stands for a constant, and R is a comparison operator (e.g., $=$, $<$, $>$, $\leq$, $\geq$, and $\neq$). Likewise, we also associate an RDQL query Q with a triple (P_{Q^S}, P_{Q^W}, C_Q), where P_{Q^S} and P_{Q^W} respectively contain all RDF path expressions in the `SELECT` clause and in the `WHERE` clause, and C_Q contains the constraints whose items are in the form of pRc, where $p \in P_{Q^W}$, c stands for a constant, and R is a comparison operator.

5.2 Query answering in data integration mode

Query answering in data integration mode includes the following steps. We use a running example for illustration.

1. Analyzing the source RDQL query to convert it from a string to a triple $Q_{in} : (P_{Q_{in}^S}, P_{Q_{in}^W}, C_{Q_{in}})$. In order to get the RDF path expressions in $P_{Q_{in}^S}$ and $P_{Q_{in}^W}$, we have to match the triple patterns (specified in the `WHERE` clause) with the RDF graph corresponding to the local RDF schema. $C_{Q_{in}}$ contains all the constraints specified in both the triple patterns of the `WHERE` clause and the `AND` clause. Because of space limitations, we ignore the detailed process of pattern matching in this paper.

Example 1. To "find the publications written by `a1`", the user poses a query over the global ontology as shown below on the left hand side (the prefix `go` stands for the name space "`http://examples.org/global#`", where the global ontology is defined). The resulting Q_{in} elements are listed on the right hand side.

```
SELECT ?title
WHERE (?book, <go:title>, ?title),
      (?book, <rdfx:contains>, ?author),
      (?author, <go:name>, ?name)
AND (?name eq "a1")
```

$P_{Q_{in}^S}$={`Book.title`}
$P_{Q_{in}^W}$={`Book`, `Book.title`, `Author`, `Author.name`}
$C_{Q_{in}}$={(`Author.name`, `eq`, `"a1"`)}

2. Rewriting the source query into target subqueries over the RDF or XML sources, by applying the query rewriting algorithm: `RDQL2RDQL` or `RDQL2PXQuery` (once for each source), which utilizes mapping information stored in the mapping table of Figure 4. The output Q_{out} of a query rewriting in algorithm is a triple of the form $(P_{Q_{out}^S}, P_{Q_{out}^W}, C_{Q_{out}})$ for the RDF source or $(V_{Q_{out}^R}, V_{Q_{out}^W}, C_{Q_{out}})$ for the XML source. From Q_{out}, we can compose the target query that is executable over the local source. Below is the result of this step for Example 1.

For the local RDF source R:
$P_{Q_{out}^S}$={`Book.booktitle`}, $P_{Q_{out}^W}$={`Book`, `Book.booktitle`}, $C_{Q_{out}}$={}.
The target RDF query is:

```
SELECT ?booktitle
WHERE (?book, <lo:booktitle>, ?booktitle)
```

For the local XML source S_1:
$V_{Q_{out}^R}$={`/books/book/@booktitle`}, $V_{Q_{out}^W}$={`/books/book`, `/books/book/@booktitle`, `/books/book/author`, `/books/book/author/@name`},
$C_{Q_{out}}$={`/books/book/author/@name`, `=`, `"a1"`}.
The target XML query is:

```
for $book in doc("books.xml")/books/book
where $book/author/@name = "a1"
return $book/@booktitle
```

For the local XML source S_2:
$V_{Q_{out}^R}$={`/writers/writer/article/@title`}, $V_{Q_{out}^W}$={`/writers/writer/article`, `/writers/writer/article/@title`, `/writers/writer`, `/writers/writer/@fullname`},
$C_{Q_{out}}$={`/writers/writer/@fullname`, `=`, `"a1"`}.
The target XML query is:

```
for $writer in doc("writers.xml")/writers/writer
where $writer/@fullname = "a1"
return $writer/article/@title
```

3. Building an answer to the source query (on the global ontology G) by assembling the fragment results returned from local sources. We need to

not only *union* the fragments (returned from different sources) while removing identical records, but also *join* the records based on some common key attribute. In addition, *null* values will be filled into the records that just partially cover queried attributes. The result of an RDQL query is a table containing URIs or string constants corresponding to the path expressions in the `SELECT` clause. For example, the answer to the query of Example 1 is a table containing a single tuple (`"b1"`), which is the union of results from S_1 and S_2. The record (`"b3"`) returned from R is filtered out since the target query over R loses the query constraints in query rewriting, caused by the partial mappings from G to R (i.e., R has no correspondence for the class `Author` in G).

5.3 Query answering in hybrid P2P mode

We only focus on the case of translating a source query in PXQuery from a peer to all the other peers, since the translation of a source RDQL query is similar to what is done in data integration mode (except for the transitive mappings). Query answering in hybrid P2P mode includes the following steps.

1. Analyzing the source PXQuery query to convert it from a string to a triple $Q_{in} : (V_{Q_{in}^R}, V_{Q_{in}^W}, C_{Q_{in}})$.

Example 2. To "list all the publications", the user poses a query (over the local source S_1) as shown below on the left hand side. The resulting Q_{in} components are listed on the right hand side.

```
for $book in doc("books.xml")/books/book
return $book
```

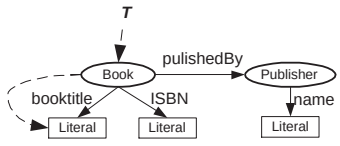
$V_{Q_{in}^R}$={`/books/book`}
$V_{Q_{in}^W}$={}, $C_{Q_{in}}$={}

2. Rewriting the source query into a target query over all the other connected RDF or XML sources, by utilizing the query rewriting algorithm: `PXQuery2RDQL` or `PXQuery2PXQuery` (once for each source) and the transitive mappings between the original data source and the target data source. The output of the query rewriting algorithm is a triple $Q_{out} : (V_{Q_{out}^R}, V_{Q_{out}^W}, C_{Q_{out}})$ for the target XML data source or $(P_{Q_{out}^S}, P_{Q_{out}^W}, C_{Q_{out}})$ for the target RDF data source.

An XML query must take into account the document structure of the XML source. The answer to an XML query is returned as a set of subtrees, each of which is rooted from one of the queried nodes (i.e., vertices in V_{Q^R}). For instance, the answer to the XML query in Example 2 is the subtree rooted from `book` in S_1 (see Figure 1). Therefore, the query rewriting algorithm also outputs a tree T with its children being the resulting subtrees of the answer. The result of this step by following Example 2 is shown below.

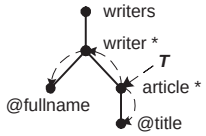
For the local RDF source R:
$P_{Q_{out}^S}$={`Book`}, $P_{Q_{out}^W}$={}, $C_{Q_{out}}$={}.
The target RDF query is:

```
SELECT ?book, ?title
WHERE (?book, <lo:booktitle>, ?title)
```



For the local XML source S_2:
$V_{Q^R_{out}}$={/writers/writer/article},$V_{Q^W_{out}}$={}, $C_{Q_{out}}$={}.
The target XML query is:

```
for $writer in doc("writers.xml")/writers/writer
   for $article in $writer/article
return
   <book booktitle="{$article/@title}">
      <author name="{$writer/@fullname}"/>
   </book>
```



3. Building an answer to the source query (against the original data source) by computing the union of the local answer (returned from the original queried peer) and the remote answers (returned from remote peers). To construct the remote answers, different methods are used for queries that target XML sources versus queries that target RDF sources. In the former case, because RDQL cannot represent document structure, the remote answer is built by organizing (based on the structure specified by T) the instances returned from executing the target RDQL query. Whereas in the latter case, the remote answer is formed by simply executing the target PXQuery query that already represents the same structure as specified by T. For Example 2, the final answer to the source query is shown below, where the three resulting lines come from the local sources S_1, S_2, and R, respectively.

```
<book booktitle="b1"> <author name="a1"> </book>
<book booktitle="b2"> <author name="a2"> <author name="a3"> </book>
<book booktitle="b4"> </book>
```

6 Conclusions and Future Work

In this paper, we propose a P2P schema-based data management framework called PEPSINT. This framework aims to semantically integrate distributed heterogeneous XML and RDF data sources. We discuss the construction of the architecture, maintenance of mappings, and query processing in PEPSINT. In particular, semantic integration is implemented at schema-level through the schema matching process and at instance-level through the query answering process. A key aspect in these two processes is the preservation of domain and document structure, which is realized by extending the RDF metadata space and providing a set of query rewriting algorithms. Because of this preservation, the user query can be correctly propagated across the heterogeneous XML and RDF data sources in PEPSINT, so that information access within the network is transparent to the user.

As for future work, we will: (1) Develop a proof of correctness for the query process. (2) Design and implement a semantic web application (e.g., for bibliographic data exchange) in PEPSINT to validate and evaluate the system. (3) Do a performance comparison of PEPSINT with other P2P data management systems.

References

1. B. Amann, C. Beeri, I. Fundulaki, and M. Scholl. Ontology-Based Integration of XML Web Resources. In *Proceedings of the 1st International Semantic Web Conference (ISWC 2002)*, pages 117–131, 2002.
2. B. Amann, I. Fundulaki, M. Scholl, C. Beeri, and A. Vercoustre. Mapping XML Fragments to Community Web Ontologies. In *Proceedings of the 4th International Workshop on the Web and Databases (WebDB 2001)*, pages 97–102, 2001.
3. M. Arenas, V. Kantere, A. Kementsietsidis, I. Kiringa, R. J. Miller, and J. Mylopoulos. The Hyperion Project: From Data Integration to Data Coordination. *SIGMOD Record*, 32(3):53–38, 2003.
4. S. Bergamaschi, F. Guerra, and M. Vincini. A Peer-to-Peer Information System for the Semantic Web. In *Proceedings of the International Workshop on Agents and Peer-to-Peer Computing (AP2PC2003)*, July 2003.
5. S. Boag, D. Chamberlin, M. F. Fernández, J. R. D. Florescu, and J. Siméon. XQuery 1.0: An XML Query Language. http://www.w3.org/TR/xquery, W3C Working Draft, August 2003.
6. S. D. Camillo, C. A. Heuser, and R. S. Mello. Querying Heterogeneous XML Sources through a Conceptual Schema. In *Proceedings of the 22nd International Conference on Conceptual Modeling (ER2003)*, pages 186–199, 2003.
7. Y. Chen and P. Revesz. CXQuery: A Novel XML Query Language. In *Proceedings of International Conference on Advances in Infrastructure for Electronic Business, Science, and Medicine on the Internet (SSGRR 2002w)*, 2002.
8. I. F. Cruz and H. Xiao. Using a Layered Approach for Interoperability on the Semantic Web. In *Fourth International Conference on Web Information Systems Engineering (WISE'03)*, pages 221–232, Rome, Italy, December 2003.
9. I. F. Cruz, H. Xiao, and F. Hsu. An Ontology-based Framework for Semantic Interoperability between XML Sources. In *Eighth International Database Engineering & Applications Symposium (IDEAS 2004)*, July 2004. (To appear).
10. L. Gong. JXTA: A Network Programming Environment. *IEEE Internet Computing*, 5(3):88–95, May 2001.
11. A. Y. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: Data Management Infrastructure for Semantic Web Applications. In *Proceedings of the 12th International World Wide Web Conference (WWW2003)*, pages 556–567, 2003.
12. HP Labs. RDQL - RDF Data Query Language. http://www.hpl.hp.com/semweb/rdql.htm.
13. M. Lenzerini. Data Integration: A Theoretical Perspective. In *Proceedings of the 21st ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS 2002)*, pages 233–246, Madison, Wisconsin, June 2002. ACM.
14. G. Moro, A. M. Ouksel, and C. Sartori. Agents and Peer-to-Peer Computing: A Promising Combination of Paradigms. In *Proceedings of the 1st International Workshop of Agents and Peer-to-Peer Computing (AP2PC2002)*, pages 1–14, 2002.
15. W. Nejdl, B. Wolf, C. Qu, S. Decker, M. Sintek, A. Naeve, M. Nilsson, M. Palmér, and T. Risch. EDUTELLA: A P2P Networking Infrastructure Based on RDF. In *Proceedings of the 11th International World Wide Web Conference (WWW2002)*, 2002.
16. W. S. Ng, B. C. Ooi, K. Tan, and A. Zhou. PeerDB: A P2P-based System for Distributed Data Sharing. In *Proceedings of the 19th International Conference on Data Engineering (ICDE 2003)*, pages 633–644, 2003.
17. E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.