XML Interoperability

Laks V. S. Lakshmanan*
Department of Computer Science
University of British Columbia
Vancouver, BC, Canada
laks@cs.ubc.ca

Fereidoon Sadri†
Department of Mathematical Sciences
University of North Carolina at Greensboro
Greensboro, NC, USA
sadri@uncg.edu

ABSTRACT
We study the problem of interoperability among XML data sources. We propose a lightweight infrastructure for this purpose that derives its inspiration from the recent semantic web initiative. Our approach does not require either source-to-source or source-to-global mappings. Instead, it is based on enriching local sources with semantic declarations so as to enable interoperability. These declarations expose the semantics of the information content of sources by mapping the concepts present therein to a common (application specific) vocabulary, in the spirit of RDF. In addition to this infrastructure, we discuss tools that may assist in generating semantic declarations, and formulation of global queries and address some interesting issues in query processing and optimization.

1. INTRODUCTION
Interoperability and data integration are long standing open problems with extensive research literature. Much of the work in the context of federated databases focused on integrating schemas by defining a global schema in an expressive data model and defining mappings from local schemas to the global one. More recently, in the context of integration of data sources on the internet, the so-called global-as-view and local-as-view paradigms have emerged out of projects such as TSIMMIS [20] and Information Manifold (IM) [12]. All of these have primarily concerned themselves with relational abstractions of data sources. Recently, the advent of XML as a standard for online data interchange has much promise toward promoting interoperability and data integration. But XML, being a syntactic model, in itself cannot make interoperability happen automatically. Two main challenges to overcome are: (i) data sources may model XML data in heterogeneous ways, e.g., using different nestings or groupings or interchanging elements and attributes, and (ii) sources may employ different terminology, a classic problem even in multi-database interoperability. While earlier approaches to integration can be extended to handle XML, they suffer from the shortcoming of requiring a commonly agreed upon global schema. Can we interoperate without a global schema?

Indeed, there are a few recent proposals that do overcome the need for a global schema – Halevy et al. [7] and Miller et al. [16]. We will discuss them in detail in Section 5. In a nutshell, both of these approaches rely on source to source mappings. One problem here is that requiring such mappings for all pairs is too tedious and cumbersome. In order to mitigate this, one can merely insist, as [7] does, that the graph of pairs with available mappings be connected. A second problem is that when a source \(s_i\) is mapped to source \(s_j\), if \(s_j\) does not have some of the concepts present in \(s_i\), then they will be lost. E.g., \(s_i\) may include the ISBN for all its books while \(s_j\) may not.

Independently of all this, there has been a lot of recent excitement around the semantic web initiative, coming with its own host of technologies such as resource description framework (RDF) and ontology description languages such as DAML, OWL, and XTM [19]. The promise of semantic web is to expose the semantics of the information content of web resources (including text, audio, video, etc.) using common transparent vocabulary thus taking the web to a higher semantic level, enabling easy exchange of data and applications. Can we leverage these developments and create a lightweight scalable infrastructure for interoperability? We believe the answer is yes and take the first steps toward this creation here.

Our thesis is that each source(’s administrator) interested in participating in data and application sharing should “en-
We make the following contributions in this paper.

- We describe a lightweight infrastructure based on local semantic declarations for enabling interoperability across data sources (Section 3).
- These declarations are mapping rules that map data items in a source to a common vocabulary, inspired by RDF. Checking the validity of these mappings involves checking certain key constraints. We illustrate this issue and show how this checking can be done in general (Section 3).
- A user query, when translated against the data sources will in general lead to exponentially many inter-source queries and a linear number of intra-source queries. The former are more expensive and their elimination and optimization is an important problem. We show under what conditions inter-source queries can be eliminated altogether. We also develop a generic technique for optimizing inter-source queries (Section 4).

Proofs are given in the full paper [10]. Optimizing inter-source queries requires us to infer key and foreign key constraints that hold for the predicates defining the vocabulary, given the source schema (e.g., XML schema or DTD) and the mappings. While previous work has addressed the derivation of key constraints, we develop an algorithm for inferring foreign key constraints. Due to space restrictions we could not include this algorithm in this paper. Interested readers are referred to the full paper [10].

2. A MOTIVATING EXAMPLE

Consider a federation of catalog sales stores maintaining their merchandise information as shown in Figure 2.

Consider the query “For each state, list the state information and all (distinct) items available in warehouses in that state.” How can we express this query? There are two sources of complexity in writing this query (say in XQuery):

- For each source, the user has to write source specific “access code” involving XPath expression specific to it.
- The query, in general, is not the union of three intra-source queries, as we must consider combinations of state/warehouse pairs in one source with warehouse/item pairs in others. As a result, the resulting query is a complex union of a large number of “joins” across sources in addition to the three intra-source queries.

In addition to the obvious burden on the user for composing such queries, without further knowledge, they are hard to optimize. Note that even if we were to employ source-source mapping techniques of [7] or [16], the second of these difficulties does not go away. The reason is that not every concept present in a source may have a counterpart in the source it is mapped to.

What we would ideally like is for the user to be able to write such queries easily, preferably by referring to some common vocabulary infrastructure created for the application on hand. The infrastructure should alleviate the burden on the user of source specific access code writing. Besides, we want the query expression to be simple to compose and to comprehend. Finally, we would like query optimization to be handled by the system.

The main idea behind our approach is based on the observation that the resource description framework (RDF) [18] is a mechanism for specifying metadata about a source that includes, not only metadata such as the author and creation date, but also the semantic concepts present therein. RDF normally employs predicates over subjects and values for describing such concepts. E.g., suppose we have the following common vocabulary (predicates) defined for describing catalog sales applications:

\[ \text{item-name, item-description, item-warehouse, warehouse-city, warehouse-state} \]

Each predicate takes two arguments where the arguments are required to play specific roles. E.g., \textit{item-name} takes two arguments where the first argument should be an identifier of an item whereas the second should be an identifier of a name, or a literal name string itself. The roles of the (two) arguments of the other predicates is self-evident from the predicate names. This is in line with RDF convention where a URI is associated with each subject, property (predicate), and value (or object). For lack of space, we do not show the RDF specs. In fact, in the spirit of semantic web, we may expect that in addition to the semantic marking up of each source, there may be additional ontological mappings (expressed in languages such as OWL [15]) that relate concepts in a class hierarchy as well as specify additional constraints on them. In this paper, we do not consider ontologies. A final point is that an RDF-based semantic marking up for a large source can be tedious. Furthermore, storing the RDF marking up explicitly is redundant and wastes considerable space. To address this concern, we can: (i) write a transformation program in, say XSLT that transforms an XML document into the required RDF markup specs, and (ii) make use of tools that assist in the writing of the XSLT program.

We expect users (local source administrators) will create such transformations or mappings with the aid of tools. In the paper, for simplicity, in place of the complex syntax of XSLT, we use a fragment of XPath together with predicates to express the mappings. An example mapping program for the data sources of Figure 2 appears in Figure 3.

Global queries can now be formulated using these vocabulary predicates. No knowledge of local sources is needed for this task.

**Example 2.1 (A GLOBAL QUERY).** Revisit the query “For each state, list the state information and all (distinct) items available in warehouses in that state.” For simplicity, we assume different sources are using the same domain for item ids (such as the manufacturer’s item id).\(^1\) We can formulate this query as follows:

\(^1\)In general, local sources may use their own id domains.
and processing in Section 4.

The rest, optimization and execution of query, is the respon-
sibility of the system. We will discuss query optimization
for $S$ in distinct-values(warehouse-state/tuple/state)
return <state> {$S} </state>

for $X$ in warehouse-state/tuple[state=$S$]
    $Y$ in item-warehouse/tuple[warehouse=$X$/warehouse]
return
</item> <id> {distinct($Y/item)} </id> <item> <state>

Note that the only information needed to formulate this
query is the knowledge that the federation contains the
vocabulary predicates item-warehouse listing items and the
warehouses for each item, and warehouse-state listing ware-
houses and their respective states. Also note that, in gen-
eral, each source in the federation may have none, some, or
all of the predicates specified in the global query.

The rest, optimization and execution of query, is the respon-
sibility of the system. We will discuss query optimization
and processing in Section 4.

3. LOCAL SOURCE DECLARATIONS

The main idea behind our approach is to provide a sim-
ple yet powerful model for local information sources that
makes interoperability possible. Any approach to interop-
erability and integration is ultimately dependent on stan-
dards and common vocabularies (or higher level declarations
that relate vocabularies to each other). In our approach,
we assume the availability of application specific standard ontologies. Such an ontology specifies a number
of simple classes and predicates (often called concepts and
properties) that are needed for the modeling of the appli-
cation. In such cases, a mapping (for example between local item
ids and manufacturer’s item ids) should be available to re-
late different id domains in order to make interoperability
possible.

In addition, an ontology could state axioms involving rel-
ationships between classes and between predicates. For brevity, we do not discuss these in this paper.

In addition, an ontology could state axioms involving rel-
ationships between classes and between predicates. For brevity, we do not discuss these in this paper.

For example, the predicate item-warehouse takes two
arguments. The first argument should be an identifi-
cation for an item. The second argument should be an
identifier for a warehouse.

Information about identifiers. For example, for a predi-
cate person-name where the first argument is an identi-
fier for person, the ontology contains information on pos-
sible identifiers for people, such as, ssn (social security number in the US), sin (social security number
in Canada), etc.

Additional type information for arguments. For ex-
ample, in the predicate person-salary, the ontology
specifies that the type information for salary must spec-
ify additional information such as currency, and period
(e.g. annual, monthly, bi-weekly).

To provide maximum flexibility, standard predicates should
be the simplest possible. Each predicate defines a relation-

Figure 2: Schema of three sources.

Figure 3: Mappings.
ship between a number of concepts, and is not divisible further into simpler predicates. Hence, following RDF, we expect the ontology will mainly employ simple binary predicates.

A local source exists in its native model (XML data, relational, spreadsheet, etc.). To make interoperability possible, the user (or local database administrator) should (1) choose the set of standard predicates that model the source contents from the relevant ontology, and (2) provide mappings (or transformation rules) that map source data to the standard predicates.

### 3.1 Mappings

Once the set of predicates for a source has been determined, mappings from XML data (or other source types) to predicates should be established. The language we use to specify XML-to-predicate mappings is based on (a subset of) XPath and is similar to mapping languages (also called transformation rules) in the literature (For example, [6]). The mapping for a binary predicate \( p \) has the following form

\[
p(\$X, \$Y) \leftarrow \text{path1 } \$G, \text{path2 } \$X, \text{path3 } \$Y.
\]

where \( \$X \) and \( \$Y \) correspond to the arguments of \( p \). \( \$G \) is called the glue variable, and is used to restrict the \( (\$X,\$Y) \) pairs to have the same \( \$G \) ancestor element in the document.

We envision tools to assist users (or database administrators of local sources) in defining the mappings. The graphical interface displays the structure of the XML document as a graph. For a simple binary predicate, three nodes in the XML schema graph should be selected by the user to determine (1) and (2): Variables associated with each of the arguments, respectively, and (3): The glue variable. Normally, the node specifying the glue variable is the least common ancestor of the first two nodes that specify the arguments. Sometimes, gluing may be accomplished by equating values of elements related to the first two argument elements.

#### Example 3.1 (Mapping Tool). In the schema graph of source 1 (below) user selects the three nodes \( \text{id, @id, and warehouse} \) for argument1, argument2, and glue, respectively. The following mapping is then generated by the tool.

\[
\text{item-warehouse} (\$I, \$W) \leftarrow \text{source1/store/warehouse } \$X, \$X/item/id \$I, \$X/@id \$W.
\]

store

warehouse

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
</table>

The mapping tool can also assist the user by performing certain type and role checking on the arguments of the predicate, and verifying or deriving integrity constraints for the predicates. The extent to which the tool can succeed in these tasks depends on the richness of local source (native) declarations of types and constraints. For example, a rich declaration for XML data would use a powerful schema declaration language (such as XML Schema) and contain elaborate integrity constraints satisfied by source data.

#### Example 3.2. In the predicate \( \text{warehouse-state} \), the ontology may specify that the first argument should be an identifier for warehouses, and should be the key for the predicate. These requirements can be verified by the tool if adequate declarations exist for the sources. For example, in the DTD of source 1 the \( \text{@id} \) attribute of \( \text{warehouse} \) is of type \( \text{Id} \), hence it is an identifier for \( \text{warehouse} \). Alternatively, if XML Schema was used, the requirements could be derived from the following constraint

\[
(e, (\text{source1/store/warehouse}, \{\text{@id}\}))
\]

using techniques of [6].

We have studied the problem of deriving foreign key constraints (or referential integrity constraints) for the predicates from XML declarations and mapping rules [10]. These constraints play a very important role in query optimization in this approach. Davidson et al. [6] have studied the problem of deriving or verifying functional dependencies from XML key constraints. Their algorithms can be used in the mapping tool for the verification of mappings correctness by checking argument roles and constraints that should be satisfied by the predicates.

### 4. QUERY TRANSLATION, PROCESSING, AND OPTIMIZATION

Our simple model of (mostly binary) predicates significantly simplifies the task of query formulation. The federation consists of local sources, where the content of each source has been declared by a set of standard predicates and mappings to them. The global content is the collection of all local predicates. Given a predicate \( p \), a source \( i \) may contain a fragment of \( p \), which we denote by \( p_i \). In general, fragments of the same predicate at different sources may contain overlapping data. We refer to the (conceptual) union of all fragments \( p_i \) of a predicate \( p \) as the global predicate \( p \).

A query is formulated in terms of global predicates and predicate fragments. The query processor architecture is shown in Figure 1. A naive implementation approach is to materialize these predicates at the coordinator using the local mapping specifications, and then execute the query using materialized predicates. This approach will be very inefficient in general.

A better approach involves expanding the query by replacing each global predicate by the union of its fragments. The outcome, in general, consists of a collection of local (i.e., intra-source) and inter-source queries, the results of which should be combined to obtain the answer to the query. The coordinator is in charge of query expansion, query preparation and transformation (to local native schemata), and coordination of sub-query execution at different sources. It then collects results of the sub-queries and combines them to form the final answer.

A local sub-query is one where all the predicate fragments specified in the query belong to the same source. A local query can be rewritten in the native model of the source, and executed locally at that source. We will demonstrate this with an example below.

---

\[\text{Example 3.2.}\text{ In the predicate \( \text{warehouse-state} \), the ontology may specify that the first argument should be an identifier for warehouses, and should be the key for the predicate. These requirements can be verified by the tool if adequate declarations exist for the sources. For example, in the DTD of source 1 the \( \text{@id} \) attribute of \( \text{warehouse} \) is of type \( \text{Id} \), hence it is an identifier for \( \text{warehouse} \). Alternatively, if XML Schema was used, the requirements could be derived from the following constraint using techniques of [6].}\]

We have studied the problem of deriving foreign key constraints (or referential integrity constraints) for the predicates from XML declarations and mapping rules [10]. These constraints play a very important role in query optimization in this approach. Davidson et al. [6] have studied the problem of deriving or verifying functional dependencies from XML key constraints. Their algorithms can be used in the mapping tool for the verification of mappings correctness by checking argument roles and constraints that should be satisfied by the predicates.

---

\[\text{Example 3.1 (Mapping Tool). In the schema graph of source 1 (below) user selects the three nodes \( \text{id, @id, and warehouse} \) for argument1, argument2, and glue, respectively. The following mapping is then generated by the tool.}\]

\[
\text{item-warehouse}(\$I,\$W) \leftarrow \text{source1/store/warehouse } \$X, \$X/item/id \$I, \$X/@id \$W.
\]

---

\[\text{store}\]

\[\text{warehouse}\]

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
</table>

The mapping tool can also assist the user by performing certain type and role checking on the arguments of the predicate, and verifying or deriving integrity constraints for the predicates. The extent to which the tool can succeed in these tasks depends on the richness of local source (native) declarations of types and constraints. For example, a rich declaration for XML data would use a powerful schema declaration language (such as XML Schema) and contain elaborate integrity constraints satisfied by source data.

---

\[\text{Example 3.2. In the predicate \( \text{warehouse-state} \), the ontology may specify that the first argument should be an identifier for warehouses, and should be the key for the predicate. These requirements can be verified by the tool if adequate declarations exist for the sources. For example, in the DTD of source 1 the \( \text{@id} \) attribute of \( \text{warehouse} \) is of type \( \text{Id} \), hence it is an identifier for \( \text{warehouse} \). Alternatively, if XML Schema was used, the requirements could be derived from the following constraint using techniques of [6].}\]

We have studied the problem of deriving foreign key constraints (or referential integrity constraints) for the predicates from XML declarations and mapping rules [10]. These constraints play a very important role in query optimization in this approach. Davidson et al. [6] have studied the problem of deriving or verifying functional dependencies from XML key constraints. Their algorithms can be used in the mapping tool for the verification of mappings correctness by checking argument roles and constraints that should be satisfied by the predicates.

---

\[\text{Example 3.1 (Mapping Tool). In the schema graph of source 1 (below) user selects the three nodes \( \text{id, @id, and warehouse} \) for argument1, argument2, and glue, respectively. The following mapping is then generated by the tool.}\]

\[
\text{item-warehouse}(\$I,\$W) \leftarrow \text{source1/store/warehouse } \$X, \$X/item/id \$I, \$X/@id \$W.
\]

---

\[\text{store}\]

\[\text{warehouse}\]

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>description</th>
</tr>
</thead>
</table>

The mapping tool can also assist the user by performing certain type and role checking on the arguments of the predicate, and verifying or deriving integrity constraints for the predicates. The extent to which the tool can succeed in these tasks depends on the richness of local source (native) declarations of types and constraints. For example, a rich declaration for XML data would use a powerful schema declaration language (such as XML Schema) and contain elaborate integrity constraints satisfied by source data.

---

\[\text{Example 3.2. In the predicate \( \text{warehouse-state} \), the ontology may specify that the first argument should be an identifier for warehouses, and should be the key for the predicate. These requirements can be verified by the tool if adequate declarations exist for the sources. For example, in the DTD of source 1 the \( \text{@id} \) attribute of \( \text{warehouse} \) is of type \( \text{Id} \), hence it is an identifier for \( \text{warehouse} \). Alternatively, if XML Schema was used, the requirements could be derived from the following constraint using techniques of [6].}\]

We have studied the problem of deriving foreign key constraints (or referential integrity constraints) for the predicates from XML declarations and mapping rules [10]. These constraints play a very important role in query optimization in this approach. Davidson et al. [6] have studied the problem of deriving or verifying functional dependencies from XML key constraints. Their algorithms can be used in the mapping tool for the verification of mappings correctness by checking argument roles and constraints that should be satisfied by the predicates.

---
In an inter-source query, fragments from different sources are specified. We present various techniques to the translation, optimization, and processing of inter-source queries (Section 4.2). Processing these queries normally involves data transmission between sources so optimizing inter-source queries is important.

Example 4.1 (A Local Query). Let's consider the global query Q of Example 2.1. The local sub-query obtained from Q for source 1 is obtained by substituting predicate fragments warehouse-state-1 and item-warehouse-1 for the corresponding predicates in Q.

The local query is then transformed by substituting its variable declarations by variable declarations obtained from mappings for source 1 (see Section 2 for mappings):

\[
\text{for } S \text{ in distinct-values(source1/store/warehouse/state) return <state>}
\{\{S\}\}
\text{for } Z \text{ in source1/store/warehouse[@state=\$S],}
\text{\$W \text{ in source1/store/warehouse[@id=\$Z/@id]}}
\text{\{distinct(\$W/item/id)\} </id> <item>}
\text{</item> <item>}
\text{</state>}
\]

The last step for local queries is native query optimization, which may be carried out by the coordinator, the source, or both. In this example, the optimizer realizes \$Z and \$W are the same, and eliminates one to get the final local query to be executed by the local source.

4.1 When can Inter-source Processing be Eliminated?

A global query often involves more than one predicate. If there are \( n \) sources and the query involves \( m \) predicates, then the expansion of the query results in \( O(n^m) \) queries. Only \( O(n) \) of these queries are local queries. The rest are expensive inter-source queries.

We are interested in characterizing cases where inter-source queries can be eliminated. If we can eliminate all inter-source queries, then we have reduced the number of queries to \( O(n) \), instead of \( O(n^m) \). In the remainder of this section we will study a simple case where the global query involves the joining of two binary predicates. Results of this section can be extended to more general cases.

Let the global query involve the join of predicates \( p(A, B) \) and \( q(B, C) \) on argument \( B \). Our Example 2.1 was of this type, joining item-warehouse and warehouse-state on the warehouse argument. In general, predicate fragments from different sources contain overlapping data. Often a distinct clause must be used in the query to eliminate multiplicities that result from this data duplication.

Consistency Condition: We say a predicate \( p \) satisfies the consistency condition, provided: (i) whenever a constraint \( p(\alpha) \Rightarrow p(\beta) \) holds in \( p \), then it holds for every fragment \( p_i \); (ii) if for fragments \( p_i \) and \( p_j \) we have \( t_\alpha \in p_i, t_\beta \in p_j \), and \( t_\alpha = t_\beta \), then \( t_i(\beta) = t_j(\beta) \).

Condition (i) states that when the semantics of a predicate \( p \) dictates that a constraint (key or functional dependency) should hold in \( p \), then all fragments should satisfy this constraint. Further, by Condition (ii), data in different fragments should be consistent in the sense that two facts (or tuples) in different fragments with the same \( \alpha \) component should also have the same \( \beta \) component when \( p(\alpha) \Rightarrow p(\beta) \) holds.

The following theorem characterizes the cases where inter-source joins are not needed. Let \( Q \) be a query involving the join of \( p(A, B) \) and \( q(B, C) \), and \( Q \) also has the distinct clause to eliminate duplicates. Let \( k \) be a source.

Theorem 4.1. No inter-source processing involving \( p_k \) is needed if and only if the following conditions hold: (1) The key constraint \( B \rightarrow C \), and the consistency condition hold for \( q \), and (2) There is a foreign key constraint from \( p_k \) to \( q_k \) on the attribute \( B \).

Proof is given in the full paper. Note that in this theorem we are assuming only local information in the form of key and foreign key constraints, plus a consistency condition which is basically a statement of correctness of data across sources. If other forms of constraints are permitted, the theorem can be extended by relaxing the conditions. An extension to this theorem is also given in the full paper.

4.2 Efficient Inter-source Processing (When Needed)

Let's consider a query involving the join of predicate \( p \) and \( q \), and concentrate on the inter-source processing of \( p_i \bowtie q_j \), \( i \neq j \). We will discuss several approaches. The optimization process, in general, will need a cost-based approach which estimates the cost of different approaches and selects the best.

A Naive Algorithm. A naive approach for calculating \( p_i \bowtie q_j \), \( i \neq j \) is to first materialize one of the predicates locally and send it to the other site. Then only transform query variable declarations on the second predicate to its corresponding local document, and execute the query (using the first predicate and the second document). For example, \( p_i \) can be materialized at site \( i \) and shipped to site \( j \). Then query is transformed to one that uses predicate \( p_i \) and the document at site \( j \). This approach may be more efficient than the first approach since the volume of data transferred may be significantly lower (predicate \( p_i \) versus the whole document at site \( i \)).

If certain conditions hold, we are able to use the semi anti-join approach discussed below in the calculation of inter-source joins to avoid generation of duplicate answers, and thereby further optimize query processing.

Assume the global query involves join of predicates \( p(A, B) \) and \( q(B, C) \). Conditions under which semi anti-join approach can be used in the calculation of inter-source joins are: (1) The global query has a distinct qualifier, and (2) The key constraints \( A \rightarrow B \) and \( B \rightarrow C \) and the consistency condition hold for \( p \) and \( q \).

The semi anti-join technique: When calculating inter-source join \( p_i \bowtie q_j \) as a step towards calculating \( p \bowtie q \), proceed as follows:

1. Let the partial result of \( p \bowtie q \) calculated so far be \( r \). Ship \( r[A^\alpha] \) to site \( i \).
2. At site \( i \), generate \( p'_i = \{t \mid t \in p_i, t(A) \notin r[A]\} \).
3. Ship \( p'_i \) to site \( j \).
(4) Calculate \( p_j \rightarrow a_j \).
(5) Combine the result with \( r \) to obtain the updated partial result.

In the full paper [10], we present a variation on the semi-anti-join algorithm that may be more efficient in some cases.

5. RELATED WORK

Data integration has received significant attention since the early days of databases. Most approaches to data integration involve deriving a global schema, which integrates local schemata. Then mappings are provided between the global schema and local schemata. There are two general approaches, local as view where local sources are considered as views of the global; and global as view where rules are provided to derive global data from local sources. There are a number of excellent surveys and discussions on data integration and related topics such as [4, 8, 9, 11, 17].

A more recent approach attempts at providing data sharing by mapping sources to each other. These approaches eliminate the need for deriving the global schema, and hence eliminate a substantial overhead in system design and creation. The approach of [7] is particularly attractive as it does not require schema mappings between every pairs of sources. Rather, schema mappings can be composed to derive new mappings. Hence, it suffices for a source to provide a mapping to one other source, preferably the one closest to its structure. Every source-pair need to be connected by a sequence of these peer-to-peer mappings. This approach works well when sources have more or less the same information content. Otherwise some concepts in a source may not have counterparts in another source, and the mapping will lose these concepts. The loss will be further propagated by mapping compositions. The Clio project has made significant contributions to the problems of data and schema integration from various source types (legacy, relational, object-oriented, semistructured) [2, 14, 16]. Their approach is based on providing tools to map any two schemas to each other, and hence it can be used for source-source as well as source-global approaches. Amman et al. [1] propose mappings of XML sources into a global schema using the local as view approach and propose algorithms for query rewriting and optimization.

Our approach differs from most of the previous approaches in that we eliminate the need for source-source or source-global mappings. Instead, we provide simple but powerful declarations that are local to each source. Global interoperability is then obtained by using the standard vocabulary predicates declared for each source. Effective query optimization in our approach depends on rich schema and consistency rules declaration for local sources, and the ability to derive key and foreign key (or referential integrity) constraints for the standard predicates. There has been a lot of recent interest in the declaration of constraints for XML data, and derivation of constraints when XML data is stored as relational. Buneman et al. study the question of constraints declaration for XML [5]. The problem of propagating XML key constraints to relational is studied in [6]. They provide algorithms to determine, given XML key constraints and XML to relational transformation rules, whether a given FD is implied by the XML constraints. We utilize these results, as well as our results reported in the full paper [10], to derive key and foreign key constraints that are used in query optimization.

6. CONCLUSION

The vision behind our project is to create a lightweight scalable infrastructure for interoperating over XML data sources. There, we presented an approach that derives its inspiration from the semantic web initiative, specifically RDF, to semantically mark up the information contents of data sources using application specific common vocabulary, by means of mapping rules in a language like XSLT. User queries are expressed over this vocabulary and are translated into source specific access code and optimized. We addressed issues of eliminating or optimizing the costly inter-source queries. The full paper addresses inference of integrity constraints over the vocabulary predicates, across the mapping rules. We plan to implement these ideas in a prototype system. Our ongoing work addresses various exciting technical challenges arising from this preliminary investigation.

7. REFERENCES