Efficient Query Processing in an Ontology-Based Mediation System

Aluno: João Carlos Pinheiro\textsuperscript{1,2}
Orientadora: Vânia Maria Ponte Vidal\textsuperscript{2}

Doutorado em Ciência da Computação
Início: Março de 2006
Término (previsto): Fevereiro de 2011
Etapas concluídas: Exame de qualificação

\textsuperscript{1}Departamento de Computação – Universidade Federal do Ceará (UFC)
Fortaleza, CE – Brasil

\textsuperscript{2}Departamento Acadêmico de Informática
Instituto Federal de Educação, Ciência e Tecnologia do Maranhão (IFMA)
São Luís, MA – Brasil

\{joaoslz, vvidal\}@lia.ufc.br

\textbf{Abstract.} Ontologies have been extensively used to model domain-specific knowledge. Recent research has used ontologies for specifying the mediated schema in the context of data integration. In this work we propose an ontology-based approach for integration of XML data sources. 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. Each application ontology is translated into an XML schema, which constitutes the schema of the XML view exported by the application ontology’s local source. Furthermore, mediated mappings define concepts and properties of the domain ontology in terms of concepts and properties of the application ontologies. Local mappings specify the correspondences between each XML view schema and its local source schema.

In our approach, the process of answering a query posed on the mediated view consists of four steps: (i) Semantic rewriting. Based on the mediated mappings, the initial query is decomposed into a set of elementary sub-queries over the application ontologies, and it is generated the semantic execution plan (SEP), which specifies how the results for the sub-queries are combined to the final result. (ii) XML translation. The sub-queries resulting from the previous step are translated into queries over the XML views, and the SEP is translated to an XML algebra expression. (iii) Optimization. This step tries to find a near to optimal final execution plan (FEP). (iv) Evaluation. The sub-queries over the XML views are rewritten in terms of their local source schemas, with the help of the local mappings. The results of the XML views sub-queries are returned to the mediator, where the final result is built according to the final execution plan.

\textit{Keywords and phrases:} data integration, schema mappings, ontologies, query processing, query optimization.
1. Introduction

Data integration is the problem of combining the data residing at different sources, and providing the user a unified view (mediated view) of these data [Lenzerini 2002], which can be queried by the users in a transparent way. In this work, we assume that the mediated view is virtual and that there is a set of source schemas describing the local data sources.

According to [Lenzerini 2002], the main components of a data integration system are: the schema of the mediated view, the schemas of the sources where real data are stored, and the mapping that specifies the correspondences between the local sources and the mediated schema. The two main problems resulting from data integration are: defining the mediated view (how to specify the mappings) and query answering (how to use the mapping to answer correctly the queries posed on the mediated view schema).

Data integration systems with a mediator-wrapper exist since several years. But these earlier systems are based on either relational or object-oriented models, which have not evolved to the standard format of the Web. Given that XML has evolved to the format of choice for exposing data over the web, some recent works propose XML-based approaches for designing a data integration system [Essid et al. 2004; Manolescu et al. 2001]. However, the versatility of XML as a data model and the expressive power of XML query languages can lead to a complex integration architecture. We need to design the XML mediated view and then discover the mappings between the XML mediated view schema and the XML source schemas. However, the specification of these mappings is labor intensive and error prone, representing over half of the effort spent in a typical data integration scenario. As the meaning of the data to be integrated is weak, it is not possible to automate the mapping generation process. So, we need semantic information about the local sources to be integrated.

Recent research has used ontologies for specifying the mediated schema in the context of data integration [Calvanese et al. 2007; Lehti and Fankhauser 2004; Amann et al. 2002]. The main reason to build an ontology-based data integration system is to provide high-level services to the clients of the information system [Poggi et al. 2008]. Clients express their queries in terms of the ontology, and the system should reason about the ontology and the mappings and should translate the request into suitable queries posed over sources. Reasoning is used to infer the mappings and to determine whether existing ontology concepts (describing the local sources) are a match for the user’s query.

In this work we propose an ontology-based approach for integration of XML data sources. In our approach, the mediated schema is represented by a domain ontology, which provides a conceptual representation of the application, and the data sources can be queried through exported XML views. We address the problem of query answering, i.e., how to compute the answer to queries posed in terms of the domain ontology. For this purpose, the system should be able to reformulate the query in terms of a suitable set of queries posed to the exported XML views. These queries are then shipped to the sources, and the results are assembled into the final answer.

The remainder of this article is structured as follows. Section 2 describes our ontology-based framework for integration of data. Section 3 describes the proposed approach with the help of an example. Section 4 presents our query processing approach. Section 5 presents related work. Finally, Section 6 presents the conclusions.

2. A Framework for Ontology-Based Data Integration

Our approach uses ontologies for both semantic descriptions of the sources and as the global schema language. Figure 1 describes the main components of the proposed mediated environment. The mediated schema is represented by a domain ontology (DO), which
provides a conceptual representation of the application domain (a global shared vocabulary). Each local source schema is described by an application ontology (AO) whose vocabulary is restricted to be a subset of the vocabulary of DO. The global ontology consists of the union of the application ontologies, and a set of axioms that define inter-ontology properties. Each application ontology is translated by means of a straightforward schema transformation process into an XML schema, which constitutes the schema of the XML view exported by the application ontology’s data source. The mediated mapping defines the concepts and properties of the domain ontology in terms of the vocabularies of the global ontology, whereas the local mappings specify the correspondences between an XML view schema and its local source schema.

![Diagram](image)

Figure 1. Ontology-based Architecture for XML Data Integration.

In our approach, the global ontology plays a key role in order to deal with data integration. It is constituted by the union of application ontologies, which are a notational convenience to divide the definition of the mappings into two stages: the definition of the mediated mapping and the definition of the local mappings. Also, the global ontology is constituted by the inter-ontology classes and properties to find out in other application ontologies which related concepts can provide the other relevant information.

To represents ontologies and mappings, we adopt a family of logics called Description Logics (DL) [Calvanese et al 1998, Casanova 2009]. Additionally, we use XSPARQL query language [DERI Galway 2009] for posing queries on the domain ontology. The following definition formally introduces the notion of the proposed mediated environment.

**Definition 2.1:** (Mediated Environment) A mediated environment is a 7-tuple

\[ \text{ME} = (\text{DO}, S_k, \text{AO}_k, V_k, \gamma_k, \text{GO}, \gamma, k=1,...,n) \]

- DO is a domain ontology, which represents the mediated schema. We assume that the classes and properties in DO are \( C_1, ..., C_n \) and \( P_1, ..., P_v \).
- for each \( k=1,...,n \),
  - \( S_k \) is a local source schema
  - \( \text{AO}_k \) is an application ontology, which describes exactly the data source \( S_k \). The vocabulary of \( \text{AO}_k \) 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 \( \text{AO}_k \). We assume that:
    - Classes and properties in DO are \( C_1, ..., C_n \) and \( P_1, ..., P_v \), respectively. So, 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 \( \text{AO}_k \) by \( \text{AO}_k:C_i \) (or \( \text{AO}_k:P_j \))
    - *(Domain Disjointness Assumption)* for any interpretation \( \xi_i \) and \( \xi_j \) for the alphabet of \( \text{AO}_k \) and alphabet of \( \text{AO}_k \), \( \xi_i \) and \( \xi_j \) have disjoint domains, for each \( i, j \in [1,k] \), with \( i \neq j \)
o $V_k$ is a XML schema, which is the XML translation for $AO_k$. Due to the standardization of both syntaxes, the translation is nonambiguous and therefore straightforward.

o $\gamma$ is a set of correspondence assertions, called a local mapping, relating the elements of $V_k$ with elements of $S_k$.

- GO is the global ontology, which consists of the union of the application ontologies $AO_k$, $k=1,...,n$, and a new set of inter-ontology classes and properties, introduced by definition.

- $\gamma$ is the mediated mapping, which defines (some of) the $\gamma$ defines the classes and properties of DO in terms of the classes and properties of GO, and is such that:

  1. for each $i=1,...,u$, the mapping $\gamma$ contains a definition of the form

     $$C_i \equiv c_1 \sqcup \ldots \sqcup c_n$$

     (1)

     where $c_k$ is a class of GO, $k=1,...,n$.

  2. for each $j=1,...,v$, the mapping $\gamma$ contains a definition of the form

     $$P_j \equiv p_1 \sqcup \ldots \sqcup p_m$$

     (2)

     where $p_k$ is a property of GO, $k=1,...,m$.

3. Case Study

In this section, we explain in more detail our mediated environment through a data integration example.

Local Sources Schemas. Consider four relational databases ($BD_1$, $BD_2$, $BD_3$ and $BD_4$) whose schemata are shown in Figure 2. The first data source provides information about patients and their pathologies. The second data source provides information about patients, consultations and diagnosis. The third data source provides general information about pathologies. Finally, the forth data source provides information about medical doctors belonging to the Regional Council of Medicine.

![Figure 2. Relational schemas of $BD_1$, $BD_2$, $BD_3$ and $BD_4$.](image)

Domain and Application Ontologies. In our approach, we assume that the user provides the domain ontology, and that there is an application ontology described with the shared vocabulary of the domain ontology, for each local source schema. Figure 3 shows the domain ontology Patients, which provides a suitable vocabulary covering the main concept of our restricted medical domain. Figure 4 shows the application ontologies for the local source schemas in Figure 2, with the same concepts of the domain ontology.

![Figure 3. Domain ontology Patients.](image)

82
Exported XML Views. The application ontologies are translated into XML schemas, which constitutes the schemas of the exported XML views (see Figure 5). Because of the standardization of both syntaxes, the translation is unambiguous and therefore straightforward.

Semantic Mappings. Figure 6 illustrates axioms of the global ontology that represent definitions of inter-ontology properties, which are properties that relate concepts in different application ontologies. For example, the axiom in line 2 defines that the property $A_2$:description is obtained by the composition of $A_1$:CID10, $A_3$:CID10 and $A_0$:description. Figure 7 shows the mediated mappings that define the concepts and properties of the domain ontology in terms of the vocabularies of the application ontologies. Due to space limitation, the local mappings for the XML views are omitted here.
4. Query Processing

In this section we give an overview on how a XSPARQL query posed on the domain ontology is processed. We use an adaptation of the methodologies proposed in [Figueiredo 2007], which is constituted by four steps: semantic rewriting, XML translation, optimization, and evaluation, summarized as follows:

- **Semantic rewriting.** In this step, the query is decomposed into a set of elementary sub-queries over the AOs that are relevant to the query. The mediated mappings in conjunction with reasoning are used to generate the semantic execution plan (SEP), which is a combination of sub-queries over relevant application ontologies using joins and unions.

- **XML translation.** The sub-queries resulting from the previous step are translated into queries over the XML views, and the SEP is translated into an XML algebra expression which is a combination of XQuery sub-queries using joins and unions. This step is simple, because the AOs are based on the same syntaxes of the XML views.

- **Optimization.** This step tries to find a near to optimal final execution plan (FEP). In our strategy, we consider the goal of reducing the amount of data transfer as an optimization criterion in choosing the final execution plan. We make use of semijoin operation [Elmasri and Navathe 2003], to reduce the communication costs.

- **Evaluation.** After the generation of the final execution plan, sub-queries over the XML views are rewritten in terms of their local source schemas with the help of the local mapping. The results of the XML views sub-queries return to the mediator where the final result is built according to the final execution plan.

**Example 4.1:** Consider a XSPARQL query [DERI Galway 2009] \( Q \) in Figure 8 that asks some information about Patients.

\[
Q = \text{for } \text{Patient } \text{sp1} \text{from } <\text{Patients.rdf}>
\]

\[
\text{return } <\text{patient}>
\]

\[
\text{sp1/SSN, sp1/name, sp1/birthDate}
\]

\[
\text{for } \text{Pathology } \text{sp2} \text{from } <\text{Patients.rdf}>
\]

\[
\text{where } \{ \text{sp1/hasDisease = sp2} \}
\]

\[
\text{return } <\text{pathology}>
\]

\[
\text{sp1/hasDisease/CID10, sp2/description, sp1/hasDisease/diagnosisDate}
\]

\[
</\text{pathology}>
\]

\[
</\text{patient}>
\]

**Figure 8:** Semantic Query \( Q \) in XSPARQL.

- **Semantic rewriting.** \( Q \) is decomposed into sub-queries \( Q_1, Q_2 \) and \( Q_3 \) shown in Figure 9 (a) over the application ontologies using the mediated mappings. The semantic execution plan in Figure 9 (b) specifies how the results for the queries are combined into the final answer.

\[\text{SEP - let } \text{sp1/SSN, sp1/name } \text{from } <\text{Q1.rdf}>
\]

\[\text{return } \{ \text{for } \text{sp1/hasDisease } \text{from } <\text{Q1.rdf}>
\]

\[\text{return } \langle\text{patient}\rangle
\]

\[\text{sp1/SSN, sp1/name, sp1/birthDate}
\]

\[\text{for } \text{sp1/hasDisease/CID10, sp1/hasDisease/diagnosisDate}
\]

\[</\text{patient}>
\]

\[</\text{Q1.rdf}>\]

\[\text{SEP - let } \text{sp2/SPID, sp2/description } \text{from } <\text{Q2.rdf}>
\]

\[\text{return } \{ \text{for } \text{sp2/hasDisease } \text{from } <\text{Q2.rdf}>
\]

\[\text{return } \langle\text{pathology}\rangle
\]

\[\text{sp2/SPID, sp2/description, sp2/hasDisease/CID10, sp2/hasDisease/diagnosisDate}
\]

\[</\text{pathology}>
\]

\[</\text{Q2.rdf}>\]

\[\text{SEP - let } \text{sp3/SPID, sp3/description } \text{from } <\text{Q3.rdf}>
\]

\[\text{return } \{ \text{for } \text{sp3/hasDisease } \text{from } <\text{Q3.rdf}>
\]

\[\text{return } \langle\text{pathology}\rangle
\]

\[\text{sp3/SPID, sp3/description, sp3/hasDisease/CID10, sp3/hasDisease/diagnosisDate}
\]

\[</\text{pathology}>
\]

\[</\text{Q3.rdf}>\]

\[\text{SEP - let } \text{sp1/SSN, sp1/name } \text{from } <\text{Q1.rdf}>
\]

\[\text{and } \text{sp2/SPID, sp2/description, sp2/hasDisease/CID10, sp2/hasDisease/diagnosisDate}
\]

\[</\text{Q1.rdf}>\]

**Figure 9:** (a) Sub-queries \( Q_1, Q_2, Q_3 \); (b) The Semantic Execution Plan for \( Q \).
- XML translation. The sub-query \( Q_1, Q_2 \) and \( Q_3 \) are translated to XQuery \( Q_1', Q_2' \) and \( Q_3' \) in Figure 10(a). The SEP is translated in the following XML algebra expression:

\[
Q' = ((Q_1' \Join \text{patient\text{/}pathology\text{/}CID10=\text{pathology}\text{/}CID10} \cdot Q_3) \cup Q_2'),
\]

where the union (U) and join (J) operators are simpler forms of the Union (U) and Left-Outer-Nest-Value-Join (LONVL) operators of the TLC algebra [Paparizos et al. 2004].

- Optimization. Figure 10(b) shows a graphical and intuitive representation of the final execution plan. To reduce the amount of data transfer, the join operation is implemented using the semijoin strategy, as follows: First, \( Q_1' \) and \( Q_2' \) are executed in parallel, increasing the efficiency of the queries processing. Then, \( Q_3' \) is executed with a selection clause based on the codes of pathologies in the result of \( Q_1' \). Then, the results of \( Q_1' \) and \( Q_3' \) a joined, and its result is combined (using union) with the result of \( Q_2' \).

![Diagram](image)

Figure 10. (a) XQuery sub-queries \( Q_1 \), \( Q_2 \) and \( Q_3 \); (b) Final Execution Plan for \( Q \).

- Evaluation. Each sub-query \( Q_i' \) is rewritten in terms of their local source schemas using the local mapping, and it is executed in a given local data source. The data extracted from the local data sources will be encoded in XML view format, where the resulting XML data will be used to populate the domain ontology. XPARQL already provided a solution for translation XML to RDF/OWL [DERI Galway 2009].

5. Related Work

[Lehti and Fankhauser 2004], explores the use of an ontology language (OWL) for the definition of the global schema and as language for semantic mapping between the local sources schemas and the global schema. It also introduces a query language for OWL (SWQL) and describes how a query posed against an OWL global schema is translated to a XQuery query over the local data sources. However, this work does not deals with the problem of integrating and querying more than one source at a time.

In [Cruz et al. 2004], is provide an ontology-based approach to the integration of heterogeneous XML documents, transforming the heterogeneous XML sources into local RDF ontologies, which are then merged into an RDF global ontology. This work does not define an approach for the unification of the results which are returned from different data sources and does not deal with query optimization.

In [Poggi et al. 2008; Calvanese et al. 2007], is presented an ontology-based data integration system, called MASTRO-I. In this system, the global schema is specified in terms of an ontology in DL-LiteA [Calvanese et al. 2008] and the source schema is the schema of a relational database, resulting from the federation of a set of heterogeneous data sources. This approach consists in defining the global schema as a set of views over local schemas [Lenzerini 2002]. Queries are posed in terms of the global schema, and they are answered by suitably reasoning on the global schema, and exploiting the mappings to
access data at the sources. MASTRO-I addresses the problem of query rewriting over the global schema; but it is not concerned with the task of building the source schema, neither rewriting this query over the data sources, which are managed by a data federation tool.

6. Concluding Remarks

In this work, we have presented an approach for an ontology-based mediation system to integrate XML data. To the best of our knowledge, this is the first solution that uses an ontology as global schema to integrate XML data provided by multiple data sources and address the problem of efficient query processing to integrate XML data provided by multiple data sources.

This thesis intends to go beyond a simple query mediation system, taking advantage of reasoning in conjunction with mediated mappings to infer additional relations between concepts, opening possibilities for more sophisticated queries, based not only on syntax, but also on meaning. The thesis proposal will be presented next August.

References


