Data Model Integration

Hassana NASSIRI\textsuperscript{1, a} Mustapha MACHKOUR\textsuperscript{2, a} Mohamed HACHIMI\textsuperscript{3, b}
\textsuperscript{a} Laboratory of the Computing Systems and Vision
\textsuperscript{b} Laboratory of Engineering Sciences
University Ibn Zohr, Agadir, Morocco
\textsuperscript{1} hassana.nassiri@edu.uiz.ac.ma
\textsuperscript{2} machkour@hotmail.com
\textsuperscript{3} m.hachimi@uiz.ac.ma

ABSTRACT

There has been significant recent interest in data integration and querying heterogeneous data sources. Thus, in our work, we aim to develop a system for querying databases regardless of the nature of their model, especially XML and relational data model as they are increasingly related in practice. Due to that we choose to make them the first models under study in this contribution. In fact, the relational model is the most dominated data model in most organizations, and it has utility widely used to manage and maintain a large volume of data. At the same time, XML is increasingly becoming the lingua franca of data interchange and has received considerable attention due to its multiple benefits. Besides, each one of these models has its specific query languages, and it will be important to ensure a flexible way to access information represented by both technologies. Thus, this paper addresses the problem of accessing data independently of the model used, by using a unique query language. Since users and even developers may not be familiar with multiple query language syntax at a time, we need to facilitate accessing data by making one single query in any query language enough to retrieve data from any data model. As a start, the models under study are the relational and XML (eXtensible Markup Language) and we aim to integrate others especially the most used and famous ones. Why exactly these two models? In one hand, Relational databases are still very essential and critical infrastructure in most organizations and have utility widely used to manage and maintain a large volume of data. On the other hand, XML has received considerable attention due to its multiple benefits, especially as it is auto-descriptive, extensible and usable in all fields of applications. Last, they are complementary in practice.

There have been many attempts to query XML and relational data. Moreover, with XML becomes the lingua franca of data interchange increasingly, various research has been done to query XML Database using relational database system \cite{1} \cite{2}. Furthermore, others have been focused on designing general systems to manage XML among other data formats \cite{3}. Such approaches have great opportunities as it has some limitations as well \cite{4}. Our purpose, therefore, is to define a system for querying data stored in different models, such as the relational and XML with any query language of these models. That means that users do not need to know each query language of each data model, one query language is enough and will meet the purpose, even if the language is none of the corresponding ones of the model in question. Hence, our proposal system will be independent of the data model and the query language as well.

The remainder of this work is organized as follows. Section 2 introduces some terms related to our zone. Section 3 describes some related work and compares our approach to them. Section 4 explains the main idea of our...
approach. Finally, Section 5 summarizes our contribution.

2 RELATED WORK

Recent RDBMS such as Oracle support some kind of uniform querying of mixed relational and XML data. As mentioned before, Oracle XMLDB technology extends the possibilities of the relational database by offering all the features of an XML database and offers an independent structure for the storage and management of XML data. Furthermore, there has been various work related to XML and relational database and variant studies trying to figure out a link between these models in order to efficiently store and query data, through varied approaches: XML views over Relational, using RDBMS to store XML data and query rewrite and translation. According to the systems studied in literature and based on our understanding of them, these approaches focused on querying data via diverse directions: Relational to XML sense or XML to Relational sense by a translation tool to transform XML queries into SQL or the opposite.

In relational to XML sense, here are some approaches that go with it: the ROX (Relational over XML) project at IBM [13], presents a way to efficiently support Relational over XML and the SQL to XQuery translation approach, and discuss the feasibility of querying natively stored XML data through SQL interfaces. It is an approach that provides access to relational data, based on SQL along relational views over XML. Likewise, The BEA AquaLogic Data Services Platform [14], a unified, service-oriented, XML-based view of data from heterogeneous information sources, which can be queried using XQuery. It proposes a framework that can transform SQL statements to XPath expressions. Also, [15] examines how XML data can be queried using XPath or SQL, and introduces a framework where SQL statements can be transformed to XPath queries that enable users to access XML and relational database through SQL. Again, [16] designs an SQL interface for XML databases that can convert SQL queries to XPath expressions and extract data from XML documents. And, [17] proposes a framework for converting SQL join (Left, Right, and Full) queries into XPath expressions to allow users to access XML database through SQL queries only.

In the other sense, i.e. XML to relational, some techniques are discussed here: [18] discusses the manner to support XML ordered data model using a relational database system, by encoding order as a data value. It proposes three order encoding methods and algorithms for translating ordered XPath expressions into SQL using these encoding methods. In the same token, [19] presents a way to process queries over XML by RDBMS through mapping XML documents/schema to RDBMS schema and use XQuery to retrieve XML documents. It introduces a system to store in and retrieve XML documents from a relational database system. XML documents are translated into tables in the relational database and stored in a shredded schema, and could be queried by the query language XQuery. The input is the user’s query, then the XQuery expressions are translated into SQL queries. The results are transformed into XML documents and returned to the user as output. Whereas, [20] presents a methodology for integrating heterogeneous data sources, including relational and tree-structured data sources, under an XML global schema, which is implemented in the Agora data integration system [21], and explains their approach in translating XQuery to SQL. Moreover, [22] discusses the Query translation in the presence of recursive data schemas and provides algorithms for rewriting an XPath query into an equivalent XPath query over a recursive DTD. [23] addresses two issues: the translation of XQuery expressions to SQL statements and the development of efficient execution strategies of the resulting queries. The proposed techniques target a relational implementation but it can be used within native XML system too. Additionally, [24] presents BLAS [25] a Bi-LAbeling based XPath processing System, as a generic and efficient system for XML storage and XPath query processing by leveraging relational databases. Also, it represents algorithms for translating XPath query to SQL query.

What makes our approach different and new is that it works on double sense, we mean that there is a way to query Relational over XML and vice versa. In other words, it is possible for users
to access and extract data with either SQL or XPath from heterogeneous models (Relational and XML case), so they do not need to control the use of the two different types of query languages to retrieve data on different database systems. It is an independent system on either data model and query language as well. On the other hand, our approach is valid with or without the need of storing XML in RDBMS also it eliminates the need of additional learning of another language to manipulate hybrid data such as SQL/XML, just SQL or XPath can do the task.

3 PROPOSED SYSTEM

3.1 Objective

Each database query language is specific to a particular data model, for example, SQL to extract data from the relational and XPath or XQuery for querying XML. Then, it is difficult for those users to retrieve data because they need the correspondent query language. Hence, in order to overcome this and make it easier for them to get what they want with less effort, we aim to make one query - no matter what the query language can be - sufficient to retrieve data even if it is none the correspondent data model. Also, the user does not have to store or manage XML data using RDBMS also no need to make any physical changes at the databases level.

We discussed here a way to handle the problem of integrating relational and XML data to support both XPath and SQL queries. Figure 1 explains this goal, it shows that with SQL we can extract data in relational and in XML Model, the same as with XPath.

3.2 Challenges

Many defies may make the task difficult, the most challenging aspect that needs to be addressed is how to efficiently bridge semantics gaps between these technologies. ‘Table 1’ shows some of these differences in brief.

Table 1. XML and Relational differences

<table>
<thead>
<tr>
<th>Relational</th>
<th>XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Regular structure</td>
<td>Heterogeneous structure</td>
</tr>
<tr>
<td>Flat data</td>
<td>Nested elements on several levels</td>
</tr>
<tr>
<td>The order has no importance</td>
<td>The Order has an importance</td>
</tr>
<tr>
<td>Static schemas</td>
<td>Schemas tend to be more extensible</td>
</tr>
<tr>
<td>Always have a schema</td>
<td>May or may not have a schema</td>
</tr>
</tbody>
</table>

3.3 The infrastructure

The groundwork for the bulk of our approach involves two phases, as presented in Figure 2. In the first phase, we generate a universal Query language (UQL), where the input is the user Query. In the second one, we identify the query through the Common Model of query specification (CMQS).

The Initial query can be written with either SQL or XPath. UQL presents our Intermediate Query Language (IQL). Why using an IQL? Because it can operate on the lowest level semantics, can increase possibilities to use more
transformations and optimizations, can be an aid to switch between several query languages, and conversion between two languages will be through it.

CMQS is the abstract layer where we specify queries against the specific data model to extract data.

As shown in Figure 3, the procedure begins with one query, which will be decomposed into a set of sub-queries, each query interrogates the suitable model resulting from an answer. Then, the answers to all these queries are recomposed to form an answer to the initial query.

![Figure 3. Retrieving data from heterogeneous models](image)

In addition, we will be able to provide the user the answer according to the query language used in the first place. For instance, if the initial query was made using SQL, then the result will be displayed in a tabular form.

4. CONCLUSION & FUTURE WORK

We try to extract data with both SQL and XPath. In fact, even if query languages are specific to a particular data model, we will be able to query a data model with the database query language of the other (with the non-corresponding Query language of the concerning model). Hence, we have built a system that can extract data independently from the query language and the storage model of data.

The relational model is the most data model used to manage data for years. Similarly, XML is rapidly becoming more and more popular and its importance as a standard format for the exchange of information with a management more and more powerful of the documents is very remarkable in the last years. This creates the need of building some bridge between the two. Due to that, we choose to make them the first models under study in this contribution.

In the future work, we will aim to integrate further data models to our system, so it can be independent of any data model as possible as we can, at least for the most known ones.

REFERENCES


