04-FE

IEEE Sixth International Symposium on

Multimedia Software Engineering

13-15 December 2004 Miami, Florida, USA

IEEE COMPUTER SOCIETY

IEEE

With support of Florida International University, Florida, USA
IEEE/CS Technical Committee on Computational Intelligence
IEEE/CS Technical Committee on Multimedia Computing
IEEE/CS Technical Committee for Services Computing

Proceedings

IEEE Sixth International Symposium on Multimedia Software Engineering

Miami, Florida

December 13 – 15, 2004

Supported by

IEEE Computer Society Technical Committee on Computational Intelligence
IEEE Computer Society Technical Committee on Multimedia Computing
IEEE Computer Society Technical Community for Services Computing

IEEE COMPUTER SOCIETY
http://computer.org
Los Alamitos, California
Washington • Brussels • Tokyo

A Flexible and Effective XML Storage and Retrieval System[1]

Li Yang
Department of Computer Science
State University of West Georgia
Carrollton, GA 30118, USA
lyang@westga.edu

Naphtali Rishe
High Performance Database Research Center
School of Computer Science
Florida International University
Miami, FL 33199, USA
rishen@cs.fiu.edu

Abstract

As XML becomes increasingly pervasive on the web, it is necessary to provide a good solution to storing and retrieving huge amount of XML data that includes not only textual data, but also multimedia documents in XML format. Using the Sem-ODB, a multimedia database system, as the underlying repository, we show how the XML storage and retrieval issue can be tackled in a flexible and effective way. We present a prototype that effectively maps and stores XML data into a Sem-ODB through the use of a meta-schema-based approach. Our approach is flexible and effective because (1) users are given control over the resulting mapping scheme via the manipulation of the KnowledgeBase; (2) it greatly reduces the number of joins generated in the final Sem-SQL query when translating an XQuery query.

1. Introduction

Recently XML (eXtensible Markup Language) has increasingly become the *de facto* standard in representing and exchanging data that include not only textual but also multimedia data (image, audio, video, etc.) over the Internet. In an XML document, the content is independent of its data presentation, which brings a lot of flexibility in exchanging data and rendering them in customized ways. For instance, the result of a query against a multimedia document can be adapted to reflect different presentation contexts according to user preferences, capabilities of physical devices (WAP phones, PDAs, PCs), etc. At the same time, how to efficiently store and retrieve XML data is an open issue in both industry and the research community. A plethora of approaches have been proposed and implemented [12, 11, 5, 4, 3, 14]. Most of the systems generate a target schema before storing XML data. However, the schema generated is often either a generic one that does not naturally reflect the structure of XML data or a fixed one that cannot be changed once it is produced. In addition, excessive join operations are frequently used to translate long path expressions of XML queries. Furthermore, none of them take into consideration the multimedia data types in their work.

In this paper, we describe a prototype that is being built at the High-Performance Database Research Center (HPDRC) [7] at Florida International University, which provides a solution to this issue using the Semantic Binary Object-Oriented Database System (Sem-ODB). The Sem-ODB, developed at HPDRC, is a multimedia spatial database system that is capable of storing textual data as well as remotely sensed and graphic data such as maps, and aerial imagery data, among others. As a fully functional multi-user, multimedia object-oriented DBMS, Sem-ODB has been successfully deployed for highly complex applications such as applications intended for storage and processing of large amounts of Earth science observations and the Terrafly Geographic Information System (GIS) [10]. The Sem-ODB's high-level data model, Sem-ODM [8], which features simple constructs, multi-valued attributes, explicit relationship description, inheritance and surrogates (object ids), and its navigation-oriented query language, Sem-SQL [9], suggest a natural and expressive approach to tackling this problem.

The system we are building stores XML documents conforming to DTDs [1] into Sem-ODB and provides an XQuery [2] facility for users to query the XML data. Our approach is distinguished by three features. Firstly, a meta-schema based mapping approach is used for coping with data model heterogeneity and schema heterogeneity. Thus, the mapping between XML and Sem-ODM is not hard-coded in the system. Secondly, the mapping information is made accessible and updateable via the use of KnowledgeBase. Unlike the other systems, where mapping information is hidden from end users, who have no control over the mapping process, our system allows users to query and update the mapping information so that they can specify more appropriate mapping schemes when such needs arise. Finally, it reduces the number of join operations involved when translating an XQuery query into a Sem-SQL query, whereas a common problem of relational approaches in translating XML queries into SQL is that the number of join operations is proportional to the length of the path expressions [12].

1 This material is based on work supported by the National Science Foundation under Grants No. HRD-0317692, EIA-0320956, and EIA-0220562.

The remainder of this paper is organized as follows. A brief overview of the Sem-ODM and Sem-SQL is given in section 2. Section 3 describes the overall system architecture with the details of two major components, DTD Mapping component and XQuery Query Translator. Section 4 concludes the paper and points out the future work.

2. Overview of Sem-ODM and Sem-SQL

Sem-ODM, a conceptual and high level data model, is the underlying data model of Sem-ODB. Two constructs, *category* and *relation*, are used to describe a Sem-ODM. *Categories* (represented as rectangles) are like *Entities* in the Entity Relationship (ER) model, except that the *Attributes* in the ER model are represented as *relations* in Sem-ODM. Binary *relations* (represented as solid arrows) between two categories are used to represent the association between them.

Semantic SQL (Sem-SQL) was adopted from traditional SQL92 and incorporated with some advanced concepts, such as the navigation operator and inverse relation operator, which significantly reduce the length of a complex query and provide an easier query facility.

3. System architecture

An overall system architecture is illustrated in Figure 1. The system first maps a DTD into a Semantic Schema, then loads and stores XML documents conforming to the DTD into a Sem-ODB database. When a user issues an XQuery to the system, the query gets translated into Sem-SQL and sent to the underlying Sem-ODB database. After data is retrieved, it is converted to XML format and returned to the user. Each component and its sub-components are further explained as follows.

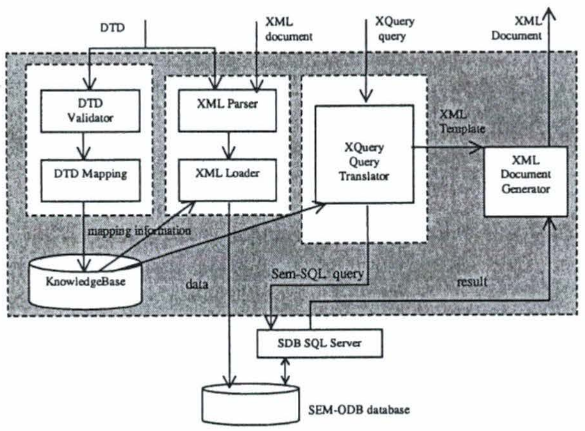

Figure 1. System Architecture

DTD Validation and Mapping Component consists of two sub-components, *DTD Validator* and *DTD Mapping Module*. The former takes a user-provided DTD and performs validation against the common DTD syntax. The latter takes the output of the DTD Validator and then performs the schema mapping from a DTD to a Semantic Schema and stores the mapping information and the meta-schemas of DTDs and corresponding Semantic Schemas in the KnowledgeBase. The mapping approach is further explained in section 3.1.

KnowledgeBase (KB): This is a repository used to store the meta-schema information of schemas (such as the subschema shown in Figure 2), including meta-schemas of DTDs and Semantic Schemas, and the mapping information between them. A semantic database is used as the KB in our system.

XML Processor: Its subcomponent, *XML Parser,* takes an XML document and validates it against its DTD; then *XML Loader* loads the XML data into the underlying Sem-ODB if it passes the validation.

XQuery Query Translator: This component translates a query specified in XQuery into an equivalent Sem-SQL query and sends it to the SDB SQL Server, which is a query engine evaluating Sem-SQL queries, for execution. Section 3.2 details the translation scheme for FLWOR expressions.

XML Document Generator: This component reconstructs the data retrieved from the Sem-ODB database into a readable XML document using the XML template extracted from the input XQuery and schema information stored in the KB, and returns the result to users. We use a variant of the sorted outer union approach [11] to structure the data for an easy tagging process.

3.1. DTD mapping component

To deal with the schema and data model heterogeneity, we utilize a *meta-schema based approach* in converting a DTD into a Semantic Schema. The basic idea of our meta-schema based approach is capturing the meta-data of both DTDs and Sem-ODM Semantic Schemas, and then mapping the basic constructs of a DTD to their counterparts in a Semantic Schema, while preserving the structure and semantic information of the DTD as much as possible.

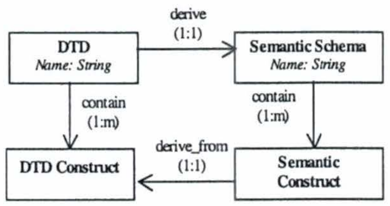

Figure 2. Sub-schema Representing the Mapping between Sem-ODM and DTD

Figure 2 shows the Sem-ODM representation of the mapping sub-schema between DTD and Sem-ODM Semantic Schema. Each DTD construct, either an element or attribute, is mapped into one Semantic Construct, which could be either a category or relation. In our study, we further extract the meta-data of both DTD Construct and Semantic Construct and capture the sub-schemas of both constructs. Space constraints prevent a detailed description. Interested readers are referred to [13].

After extracting the meta-data, we apply some mapping rules to transform a DTD construct into a Semantic Schema construct. The basic idea of the rules is to map the majority of elements into categories. Some special elements (e.g. ANY and PCDATA) are mapped into categories if they do not have any parent element, or are shared by multiple parent elements, or appear in their only parent element multiple times, and are otherwise mapped into attributes. The insight here is to inline a sub-element as an attribute of its parent element if it does not appear in its parent element multiple times to reduce the number of categories created. The attributes in a DTD are mapped as relations in a Semantic Schema. Additionally, we map the relationships between sub-elements and their parent elements to relations of two categories corresponding to the elements in DTD. Because the actual multimedia data are not contained in the XML document, an extra attribute needs to be created for storing them. This can be accomplished by the designer at the end of the mapping process when she tunes the resulting schema. For instance, Figure 3 shows a DTD example that is extracted from [12] and slightly modified and used as the DTD running example throughout this paper. There is a *pic* element for each *author* whose attribute *fileref* contains an image file location of the author.

```
<!ELEMENT publication (article| monograph)*>
<!ELEMENT article (title, author*, contactauthor)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT contactauthor EMPTY>
<!ATTLIST contactauthor authorID IDREF #IMPLIED>
<!ELEMENT monograph (title, author, editor)>
<!ELEMENT editor (monograph*)>
<!ATTLIST editor name CDATA #REQUIRED>
<!ELEMENT author (name, addr, pic)>
<!ATTLIST author id ID #REQUIRED>
<!ELEMENT name (first?, last)>
<!ELEMENT first (#PCDATA)>
<!ELEMENT last (#PCDATA)>
<!ELEMENT addr ANY>
<!ELEMENT pic EMPTY>
<!ATTLIST pic fileref CDATA >
```

Figure 3. DTD Example

Figure 4 (a) illustrates the resulting Sem-ODM Semantic Schema after applying the mapping rules to element *author* and its sub-elements of the example DTD of Figure 3. For clarity, the corresponding part for other elements in Figure 3 is not shown here. Basically, *article*, *title*, *contactauthor*, *author*, *name* and *pic* are mapped into categories, while *addr*, *first* and *last* are mapped to attributes, and the parent-child element relationships are mapped to relations between categories. Notice that an attribute *data* of *Binary* data type is created in the schema representing the image of the author. The number in parentheses represents the ordering of each relation. Attribute *order* is used to represent the document order of each element.

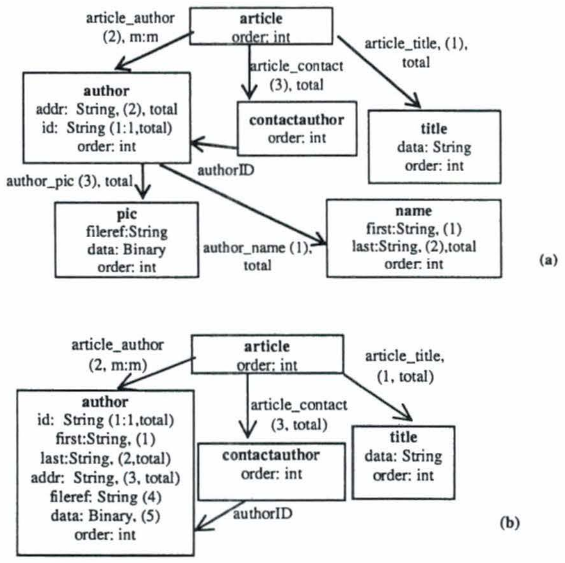

Figure 4. Semantic Schema Representation of (a) author and its Sub-elements; (b) User-Preferred Mapping

Because of the extraction of the meta-schemas of both data models, the mapping information is not hard-coded in an application; rather it is generated dynamically during the mapping process and kept in the KB for future query translation and result reconstruction phase. This enables user-defined mappings and brings a lot of flexibility in generating user-friendly transformations. For instance, if one feels that all the information about an *author* should be included as its attributes, she can do so by adjusting the mapping scheme stored in the KB as long as the change does not violate the semantic constraints of the XML documents. In this case, *first*, *last* and *pic* will be inlined as attributes of *author*, and relation *author_name* and *author_pic* and category *name* and *pic* will be dropped. Consequently, the ordering information has to be adjusted. For instance, the order of *addr* will become 3 instead of 2, since *front* and *last* have an ordering number 1 and 2, respectively. Figure 4 (b) shows the user-preferred mapping.

3.2. XQuery query translator

An XQuery query is first simplified before being translated as per normalization rules proposed in [6] to reduce its complexity and to ease the Sem-SQL translation process. The result of the process is a simplified XQuery query. We then consider the translation schemes on the principal expressions of XQuery, including path expressions, FLWOR expressions, functions, etc. For clarity, we only present the translation scheme on FLWOR expressions in this paper. Interested readers may refer to [13] for more details.

Table 1 (a) shows an informal representation of the FLWOR expression. Note that l_1, l_2, o_1, o_2, *and* g_i $(i=1..n)$ represent expressions. Except g_i, which appears to construct the result documents and denotes the *ith* expression appearing in the return clause, the rest of the notation is self-explanatory; (b) shows a high-level translation scheme to Sem-SQL. Our preliminary experiment proves the argument that our approach greatly reduces the number of joins generated in the final Sem-SQL query.

Table 1. (a) Informal Representation of a FLWOR Expression (b) FLWOR to Sem-SQL Translation Scheme: the result is ordered by the ordering specifications $o_1$and o_2 in the *order by* clause and document order of h_i

```
for      $x1 in Exp1,  $x2 in Exp2
let      $a1 :=.... ,  $a2: =....
where    l1 and l2
order by o1, o2
return   g1     g2......  gn                              (a)
```

- Expand g_i with its descendant information: insert all the descendant information into appropriate positions in a template which contains the *return* clause of the FLWOR, so that we know what expressions are needed to be evaluated in the *return* clause. For instance, *return /author* should be translated into *return <author id = $u> <name> <first>$f</first> <last> $l</last> </name> <addr> $addr</addr> </author>*. In this way, we might get more expressions than those present in the original FLWOR expression. Suppose we will have *m* expressions in the form of h_i $(i = 1..m)$ after inserting the descendants of g_i $(i = 1..n)$, where $m \geq n$.
- Search the KB for the starting categories S_1, S_2 based on Exp_1 and Exp_2.
- Look up the Path Table *PT* to find all the paths that are necessary for evaluating o_1, o_2, l_1, l_2 and h_i $(i = 1..m)$.
- Check the KB to generate the navigation paths for all the above XQuery paths. Suppose they are *o1_path, o2_path, l1_path, l2_path, hi_path (i=1..m)*.
- Create the following Sem-SQL statement for the FLWOR expression

```
select 0, o1_path, o2_path, h1_order, h1_path, null, ...,null
from S1, S2                                          m+1
where Predicate( l1_path) AND Predicate(l2_path)
union all
select 1, o1_path, o2_path, h2_order, null,h2_path, ...,null
from S1, S2
where Predicate( l1_path) AND Predicate(l2_path)     m+1
.....................
union all
select m-1, o1_path, o2_path, hm_order,null, null, ...,hm_path
from S1, S2                                           m+1
where Predicate( l1_path) AND Predicate(l2_path)
orderby 2,3,4                                              (b)
```

4. Conclusion and future work

In this paper, we have described an approach to tackling the XML storage and retrieval issue using Sem-ODB, a multi-media object-oriented database system. The system we present here effectively maps and stores XML data into a Sem-ODB via the meta-schema approach. The mapping approach is flexible since users are given control to some degree over the resulting mapping scheme via manipulation of the KnowledgeBase. The approach also greatly reduces the number of joins generated in the final Sem-SQL query.

We have implemented all the components except XQuery Query Translator and XML Document Generator. Future work will include the completion of the remaining components and extensive experiments on large XML documents for performance evaluation. To be able to provide metadata-based, content-based, and semantic-based multi-media data query capability, components accomplishing those tasks need to be incorporated. In addition, the manipulation of the KB is still done manually via writing code using SDB java APIs; we will automate this process and provide a user-friendly GUI in the future.

References

[1] Bosak, J., Bray, T., Connolly, D., Maler, E., et. al., "W3C XML Specification DTD", http://www.w3.org/XML/1998/06/xmlspec-report.html.

[2] Boag, S., Chamberlin, D., et al (Eds.), "XQuery 1.0: An XML Query Language", W3C Working Draft, November 2002, http://www.w3.org/TR/xquery/.

[3] Deutsch, A., Fernandez, M., Suciu, D., "Storing Semistructured Data with STORED", *Proceedings of ACM SIGMOD*, 1999, pp. 431-442.

[4] Kappel, G., Kapsammer, E., Rausch-Schott, S., Retschitzegger, W., "X-Ray -Towards Integrating XML and Relational Database Systems", *Proceedings of ER'00*, pp. 323-338.

[5] Lee, D., Chu, W.W., "CPI: Constraints-Preserving Inlining Algorithm for Mapping XML DTD to Relational Schema", *Journal of DKE*, 39(1), 2001, pp. 3-25.

[6] Manolescu, I., Florescu, D., Kossmann D., "Pushing XML queries inside relational databases", Tech. Report No. 4112, INRIA, January 2001.

[7] Rishe, N., Sun W., Barton D., Deng Y., et al., " Florida International University High Performance Database Research Center", *SIGMOD Record*, 24(3),1995, pp. 71-76.

[8] Rishe, N., *Database Design: The Semantic Modeling Approach*, McGraw-Hill, 1992.

[9] Rishe, N., "Semantic SQL", Internal Document, High-Performance Database Research Center, School of Computer Science, Florida International University, 1998.

[10] Rishe, N., "TERRAFLY: A High-Performance Web-based Digital Library System for Spatial Data Access", *ICDE'01* (Demo), 2001, pp.17-19.

[11] Shanmugasundaram, J., Shekita, E. J., Barr, R., Carey, M. J., et al., "Efficiently Publishing Relational Data as XML Documents", *Proceedings of VLDB'00*, pp. 65-76.

[12] Shanmugasundaram, J., Tufte, K., et al., "Relational Databases for Querying XML Documents: Limitations and Opportunities", *Proceedings of VLDB'99*, pp.302-314.

[13] Yang, L., "XML Storage and Retrieval Using the Semantic Binary Object-Oriented Database System (Sem-ODB)", PhD Dissertation, Florida International University, 2003.

[14] Yoshikawa, M., Amagasa, et al., "XRel: A Path-Based Approach to Storage and Retrieval of XML Documents Using Relational Databases", *ACM Transactions on Internet Technology*, 1(1), 2001, pp. 110-141.