

Transformation Rules from Semi-structured XML Documents to Database Model

Y. Badr**, M. Sayah*, F. Laforest**, A. Flory**

* Lebanese University
Faculty of Sciences II
P.O. BOX 90 656 Jdeideh
Lebanon
marguerite.sayah@ul.edu.lb

**LISI - INSA de Lyon
20 av. Albert Einstein
69621 Villeurbanne Cedex
France

{ youakim.badr, frederique.laforest, andre.flory }@insa-lyon.fr

Abstract

XML semi-structured documents provide a flexible and natural way to capture data. More often, users exchange documents with others or reuse their data in a wide range of tasks. A direct consequence is a growing need to manage semi-structured documents and to gain conventional database functionality. The aim of our research is the development of a comprehensive capturing and mapping data system that ensures flexible and well-adapted information capture and at the same time efficient information retrieval. A transformation process allows the mapping between the semi-structured document model and the traditional database model. It handles data extraction and data restructuring and deals with data mapping rules to feed traditional databases.

1. Introduction

In a multitude of MIS applications, database forms used in capturing data, are too limiting and constraining. They correspond to a rigid database schema, where data is restricted to a number of fields with limited size and predefined data type. Although traditional database query languages allow an easy and efficient information retrieval, the flexibility in capturing data remains an important criterion in evaluating software applications. A more natural and flexible way to capture information is the use of electronic documents. We briefly mention two types of documents and highlight some of their common features and differences: structured documents, and semi-structured documents.

Structured documents describe data structure by using tags and provide semantic to their contents. They are characterized by fine-grained data of small independent units of information. In capturing data, structured documents are less constraining than database forms but still too limiting.

Semi-structured documents are in between structured and free text documents. They are characterized by larger grained data, usually tagged paragraphs of free text, where each paragraph contains relevant information for a domain of interest. Currently, they are in wide use because of being more flexible in capturing data than database forms and structured documents, and easier than free text documents in managing information. SGML [1] and XML [2] are two meta languages used to define tags for structured and semi-structured documents.

The aim of our research is the development of a comprehensive capturing and mapping data system that ensures flexible and well-adapted information capture and at the same time efficient information retrieval. In our novel system data is captured paragraph by paragraph.

We propose a new approach to query semi-structured documents, which consists in extracting structured information from free text paragraphs, fill in a domain-oriented database, then apply traditional query languages on the database. [6] presents a step in this direction; it considers information extraction as simple pattern extraction. We now propose a more elaborate technique based on a transformation process.

Section 2 of our paper discusses related work in querying XML documents and mapping documents into database, section 3 presents the architecture of our system, section 4 describes the transformation processor and illustrates it through transformation rules, section 5 concludes our work, it describes the prototype we are currently implementing and proposes future extensions.

2. Related works

Many query languages for extracting and restructuring the contents of XML documents have been proposed, but none is recommended by W3C [7]. Some extend traditional languages [5], others are inspired by XML [4, 8] or designed for semi-structured data and can be used for XML [3, 9, 10]. In fact, the data expressed in XML is strikingly

similar to semi-structured data. [11] and [12] highlight common features and differences of some of XML query languages. Most of current query languages do not support join, aggregation, or abstract data types. Others do not offer manipulation languages to insert, delete, and update elements in collections of documents. In addition, XML queries do not evaluate efficiently and are hard to formulate

Another trend to query and manage data in XML structured documents consists of mapping data to a traditional database model. This type of mapping is provided by many systems. STORED [13] is a query language that automatically generates data-mining techniques to map data. It closely relates to Lorel [3], UnQL [9], and struQL [10] and applies only to structured data sources that have regular structure with few outliers. Within the same context, an extension of OQL, OQL-Doc [14], allows to query documents with a structure obeying some grammar. The disadvantage of this approach is that users have to switch between two data models. Sometimes, the data is inefficiently stored and missing data corrupts relations.

Currently there is an increasing use of semi-structured documents. These documents contain tagged paragraphs that hide relevant information. NoDoSE [15] provides a semi-automatic system to extract data. It consists of a general structure-mining algorithm that completely supports text and HTML documents and partially semi-structured documents. Based on an application ontology that describes a domain of interest, [16, 17, 18, 19] formulate rules to extract relevant information. This approach is limited to relatively small ontological model and small set of constants. The extraction phase is also purely syntactic and doesn't support semantics of natural language processing.

In our research, we are studying a comprehensive capturing system based on semi-structured documents. We propose a novel approach to map semi-structured document model to database model, and use traditional query languages to retrieve data.

Unlike studied mapping techniques [13,15,16,17], our approach is more general. It supports information extraction from free text and feeds traditional databases.

3. System Architecture & Overview

Our system provides a capturing interface for a domain of interest based on semi-structured documents. It hinges on a transformation processor that allows the mapping between the semi-structured document model and the traditional database model. This processor applies rules to extract relevant information from semi-structured documents, to re-structure them in order to produce structured XML documents, and to feed the database. Figure 1 illustrates the system architecture.

The Capturing User Interface, which is a graphical interface, provides the user with a list of DTD. Each DTD

presents an activity in our domain of interest and describes paragraphs to include in the semi-structured document instance. Once the user chooses a DTD, he fills its paragraphs with natural free text. The CUI then generates an instance of semi-structured document and calls the transformation processor.

The transformation process maps data from semi-structured documents to database. It comprises two phases. The first phase is based on transformation rules to structure extracted data according to a database schema. As a result, an instance of structured document is generated containing only relevant information. The second phase relies on data mapping rules to transfer data from structured document instances to the database.

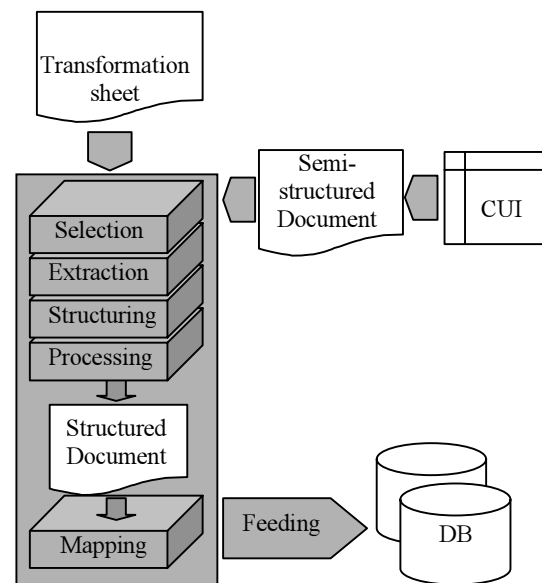


Figure 1 : System architecture

4. Transformation processor

The transformation processor is the core of our system. It is based on a transformation sheet of rules of different types. It reads both an XML document and its DTD, looks for the DTD transformation sheet, executes rules, and produces a new structured document that conforms to a predefined DTD corresponding to the database schema.

4.1. Transformation sheet overview

An important aspect of our system is the use of the transformation sheet. The transformation sheet is an abstract model that describes the transformation mechanism. It contains rules of five different types:

- Selection rules to select elements efficiently.

- Extraction rules to extract relevant information.
- Processing rules to manipulate extracted information.
- Structuring rules to reorder the structure and generate new elements into the output document.
- Data mapping rules to feed database with data.

Each DTD has an associated transformation sheet. The processor uses it either to construct documentary fund or to fill a traditional database. In the later case, the output contains intrinsic information and convenient structure to facilitate the database feeding. The transformation sheet is strongly inspired by the XSLT [20] style sheet; it uses XML syntax with extensions to implement our rules. The motivation for taking this trend is to build a system based on XML background.

In the transformation sheet, rules are within the transformation templates. A transformation template has a selection rule specifying elements it applies to and a list of processing and structuring rules to execute when the selection rule is matched. A transformation template has the following syntax:

```
<transform match="pattern">
    content
</transform>
```

Each transformation template is defined by a *transform* element. The attribute *match* and its value specify the selection rule. The content has restructuring and processing rules as sub-elements whose execution results in the target document.

The transformation processor manipulates the XML document as a tree of nodes. It scans the semi-structured document tree looking through each sub-tree in turn. As each tree in the XML document is read, the processor compares it with the selection rule of each transformation template. When the processor finds a tree that matches a selection rule's pattern, it executes rules inside the transformation template and tags returned values into the target document.

4.2. Selection rules

A selection rule is a match pattern that identifies a particular element, or any part of the document. To write powerful match patterns, we use the XPath language [21]. We illustrate the selection rule syntax as an attribute-value pair of transformation template.

```
<transform match='XPath_Expression'>
    content
</transform>
```

The selection rule *match= XPath_Expression* allows a convenient syntax to match elements in semi-structured documents. The content is a sequence of extraction and restructuring elements.

4.3. Extraction rules

Each element in a semi-structured document is a tagged paragraph of free text that contains relevant information to be extracted. An extraction rule is inserted in the transformation template. It returns multi-slot values and has the following syntax:

```
<extract pattern='extraction_rule'>
    <slot name='slot_i' />      i=0,1,2 ...
</extract/>
```

Extraction_rule is an entry key in the dictionary of text extraction rules. Its label is used to invoke the extraction component.

We mention that processing natural language does not represent the aim of our work. Our framework is a new research area at the intersection of processing natural language, XML and traditional databases.

4.4. Structuring rules

Structuring rules are used to tag relevant information and insert them into the target structured document. They are mainly conceived to put the data in a format that is easy to feed the traditional database. The syntax used to express structuring rule is:

```
<element name = 'element_name' > content </element>,
where element_name is a new element in the target
document tagging the content.
```

To add attributes with values to new elements, a structuring rule has the following syntax:

```
<attribute name = 'attribute_name' >
    attribute_value
</attribute >
```

4.5. Processing rules

Processing rules apply operations to extracted information or elements. The result is tagged and inserted into the target document. The basic categories of operations are string manipulation, arithmetic and aggregation operations etc ...

A processing rule has the following syntax:

```
<value-of select = 'expression' />, where the expression is
evaluated and its value is inserted in the target document.
```

4.6. Filling in an existing database

In order to fill in a database with data from documents, it is necessary to map the document structure to the database structure [23]. Using the data-specific object model [24], the document is treated as a tree of objects (a view of classes); generally, element types correspond to classes, and attributes and terminal elements (PCDATA) correspond to properties. This model maps directly to

databases using traditional object-relational mapping techniques [25] or SQL3 objects [26]. Figure 2 shows the syntax of the mapping rules.

```
<ClassMap>
  <ElementType Name="element_in_SD_instance" />
  <ToClassTable>
    <Table Name="table_in_database" />
  </ToClassTable>
  <PropertyMap>
    <Attribute Name="sub-element_in_SD_instance" />
    <ToColumn>
      <Column Name="column_in_table" />
    </ToColumn>
  </PropertyMap>
</ClassMap>
```

Figure 2: Extract of data mapping rules

5. Conclusion

In this article, we have presented an approach joining document model and database model. Semi-structured documents provide flexibility to capture free text, while traditional databases retrieve data efficiently. Our approach hinges on transformation rules to transform semi-structured documents into structured documents and on data-mapping rules to fill a database with data out of structured documents. Furthermore, the stored data can be used for statistical studies and decision-support. We also believe that mapping a semi-structured document model to a database model is a subject of substantial research attention. We are currently implementing our prototype on top of existent XML tools. We intend also to use the WHISK [22] learning system, which learns from training instances and builds a dictionary of extraction patterns.

Bibliography

- [1] ISO, Information Processing – Text and office Systems- Standard Generalized Markup Language, International Organization for Standardization, Rdf No. ISO 8879:1986,1986
- [2] World Wild Web Consortium, Extensible Markup Language (XML)1.0, W3C recommendation 10-February-1998, URL:<http://www.w3c.org/XML/>
- [3] Lorel S. Abitaboul, D. Quass, J. Widom, and J.L. Wiener. The lorel query language for semi-structured data. International Journal on Digital Libraries1997
- [4] J. Robie, XQL: XML Query Language, URL:<http://www.ibiblio.org/xql/xql-proposal.html> 1999.
- [5] A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. XML-QL: a query Language for XML. International WWW conference 1999.
- [6] F. Laforest, A. Flory. The Electronic Medical Record: Using Documents For Information Capture. In Medical Informatics Europe, Hannover, Germany, August 2000.
- [7] The World Wide Web Consortium (W3C) Home page, URL:<http://www.w3c.org/>
- [8] W3C, XML Extensible Stylesheet Language (XSL), URL: <http://www.w3.org/Style/XSL>, In Proc. WWW8 Conf., 1999.
- [9] P. Buneman, S. Davidson, G. Hillebrand, and D. Succiu. A query language and optimization techniques for unstructured data (UnQL). In Proc. SIGMOD Conf., 1996.
- [10] M. Fernandez D. Florescu J. Kang A. Levy D. Suciu. Catching the boat with Strudel: experiences with a web-site management system (StruQL). In Proc of ACM-SIGMOD International Conference on Management of Data , 1998
- [11] A. Bonifati, S.Ceri. Comparative Analysis of five XML Query Languages. SIGMOD Record, vol. 29 No. 1, March 2000.
- [12] M. Fernandez, J. Simeon, P. Wadler. XML Query Languages and exemplars. URL: [http:// www-db.research.bell-labs/user/simeon/xquery.html](http://www-db.research.bell-labs/user/simeon/xquery.html)
- [13] A. Deutsch M. Fernandez D. Suciu , Storing semistructured data with STORED, In Proc of the ACM SIGMOD International Conference on Management of Data , 1999.
- [14] S. Abiteboul, S. Cluet, V. Christophides, T. Milo, G. Moerkotte, and J. Simeon. Querying Documents in Object Databases, 1996.
- [15] B. Adelberg, NoDoSE: a tool for semi-Automatically Extracing Structured and Semistructured Data from Text Documents. Proceeding of ACM SIGMOD international conference on Management of data, 1998, pages 283-294.
- [16] D. Embley, D. Campewll, R. Smith, S. Liddle. Ontology-Based Extraction and Structuring of Information from Data-Rich Unstructured Documents.
- [17] D. Embley, D. Campewll, R. Smith, S. Liddle. A Conceptual-Modeling Approach to Extracting data from the Web.
- [18] D. Fensel, J Angele, S. Decker, S. Schnutt, S. Staab, R. Studet A. Witt. On2broker: semantic based access to information sources at the WWW. Institute of AIFB, Univ of Katlstuhe, Germany.
- [19] Flogic M. Kifer, G. Lausen, J. Wu: Logical Foundations of Object-Oriented and Frame-Based languages, ACM journal, 38(3): 620-650, 1991
- [20] W3C, XSL Transformation Language (XSLT), URL:<http://www.w3.org/TR/xsl>
- [21] W3C, XPath Version 1.0, URL:<http://www.w3.org/TR/xpath>
- [22] S. Soderland, Learning Information extraction rules for semi-structured and free text (WHISK) Journal of Machine Learning 1998.
- [23] Modeling Relational Data in XML URL:<http://www.extensibility.com/tibco/resources/modeling.htm>
- [24] Scott W. Ambler, Mapping Objects to Relational Databases, URL:<http://www.ambyssoft.com/mappingObjects.pdf>
- [25] Mark L. Fussell, Foundations of Object-Relational Mapping, URL:<http://www.chimu.com/publications/objectRelational/>
- [26] D. Brashear, SQL Reference Page URL:<http://www.contrib.andrew.cmu.edu/~shadow/sql.html>.