INFORMATICA, 2010, Vol. 21, No. 4, 505–519

From XML to Relational Models

Elena CASTRO, Dolores CUADRA, Manuel VELASCO

Computer Science Department, Carlos III University of Madrid
Avda. Universidad, 30, 28911 Leganes, Madrid, Spain
e-mail: {ecastro, dcuadra, velasco}@inf.uc3m.es

Received: July 2008; accepted: July 2010

Abstract. For many businesses and organizations, the achievement of interoperability has proven to be a highly desirable goal. However, without efficient schema mapping mechanisms or models that allow for the storage and management of information from several distinct systems, the goal of interoperability is impossible to attain. Due to the role of XML as a standard in information exchange, considerable research has been undertaken to find effective methods or algorithms for the conversion from XML to database models. This paper reviews leading research in the field – focusing particularly on three novel approaches taken – and proposes an original schema mapping mechanism using a conceptual model which, due its higher level of abstraction, maximizes the preservation of semantics.

Keywords: inlining, mapping, data model semantics, data model, schema, metaschema.

1. Introduction

Since the rise of Extensible Markup Language (XML) as a *de facto* standard in data exchange, the W3C has proposed various models for the management of XML document collections (W3C, 2007). While such collections had traditionally been managed using Document Type Definitions (DTDs), XML Schema is currently the most widely used model since its formalism offers developers a set of tools for concept definition with more semantics.

XML documents can be categorized as data-centric XML (almost entirely structured) or document-centric XML (semi-structured or unstructured) and there exist four different proposals for their storage and management: using a file system, a native XML database, a relational database management system or an object-oriented database management system (Pokorny, 2009). While most technologies currently provide mechanisms for dealing with XML data, no model mapping currently exists – due to the differences in model structures – that is able to completely preserve the semantics of the original. Given the importance of relational database technology and advanced data management to ensure data availability, concurrency and confidentiality, it is necessary to analyze this loss of semantics and to propose a new, robust mapping mechanism.

An XML document collection usually follows a fixed structure given by a DTD or an XML Schema. In spite of the fact that DTD formalism has become obsolete, a good deal

of legacy data nevertheless exists which requires processing. As a result, it is necessary to analyze methods and techniques capable of handling it.

The aim of this study is to review the principle algorithms for the mapping between XML data (and/or their underlying schemata) and relational databases (Florescu and Kossman, 1999). In Section 3 of the paper, three recent algorithms are analyzed and a new proposal is introduced. Sections 4 and 5 develop this new proposal more fully and present related experimental results. The paper ends with sections for conclusions as well as proposals for future lines of research.

2. Background

Mapping between XML and relational databases (or, for that matter, any other technology) can be achieved in two ways: by mapping from XML to the other model or vice versa. While this paper confines itself to an analysis of the former process, some space is nevertheless dedicated to the question of original XML document reconstruction raised in the latter process.

An important technological gap in mapping XML data to relational databases is that the mapping algorithms generally focus on DTD structure, ignoring the semantic aspects of these schemas. In order to bridge this gap, a number of different approaches have been taken.

In the mapping of XML data to relational databases, one of the more noteworthy attempts has been STORED (Semi-Structured to Relational Data; Deutsch *et al.*, 1999), according to which data-mining techniques are employed to obtain information from which, in turn, rules are derived for mapping XML to relational databases. In addition, some authors (Bourret, 2001) have proposed algorithms that combine natural language techniques in order to derive transformation rules; whereas others have proposed algorithms that do not use data-mining or natural language techniques, but rather focus strictly on DTD structure (Shanmugasundaram *et al.*, 1999; Bohannon *et al.*, 2002; Tatarinov *et al.*, 2002; Elmasri and Navathe, 2006). Recently, additional approaches have proposed methods whereby semantics are preserved reusing existing algorithms in conjunction with regular grammars (Lee, 2002a, 2002b, 2003). Due to the importance of this last, mixed technique, it has been included among the algorithms examined more carefully in the following section of this study.

Regarding mapping based on conceptual models, McBrien and Poulovassilis (2002) propose an intermediate graph model to facilitate conversion. Liu and Ling (2000), on the other hand, use an object-oriented formalism in mapping HTML data to schemas, though their study does not consider XML data directly. Finally, Conrad *et al.* (2000) use UML to try to model XML more easily.

Another interesting approach is that of Dos Santos Mello and Heuser (2001) who propose a mapping between DTD-dependent XML data collections, complete DTDs and conceptual schemata using ORM/NIAM and EER model functionalities. To achieve this aim and obtain a canonical representation of the conceptual schemata, they generate a set

of conversion rules for the DTD structure and heuristics for various semantic interpretations of this structure. In addition, an ontology is employed as a layer in the processing of the query system, interacting with the user for the validation and retrieval of these schemata. While the authors take mixed models, cardinality constraints and exclusive elements into account, they nevertheless fail to distinguish between constructors and employ the user validation in an attempt to make up for the fact.

Another novel approach is that proposed by Atay *et al.* (2007) in which DOM and SAX are used to write an algorithm for mapping XML to relational data while taking into account their underlying schemata. This algorithm will be discussed in greater detail in the following section.

3. Advanced Research: Mapping Between XML and Database Models

Since the late 1990s, numerous mechanisms have been developed and published for the automatic translation of XML data into relational databases. Following the brief review from the previous section of this paper, one may generally classify such mechanisms as those that are regular-grammar-based, graph-based or work with mixed mechanisms. In the subsections that directly follow, the most representative proposals of each of these classifications are presented and discussed.

Due to the hierarchical nature of XML and the unordered nature of relational models, model mechanisms need to employ robust algorithms when mapping the former to the latter. Additionally, a majority of these algorithms works with XML data (XML documents) rather than with their underlying schemas, inevitably resulting in the loss of domain semantics during mapping (Castro, 2004). For this reason, the subsections below in this paper limit themselves to reviewing recent research techniques solely for the mapping between DTDs or XML Schema and relational models.

In order to facilitate comparison between the models discussed below, a common example (Fig. 1) is used regarding a DTD for information about the staff of an advertising agency. While, due to the specific aims of this paper, conversion of the DTD to an XML Schema has not been done here, it is nevertheless important to note that all information represented in a DTD could be so converted.

3.1. *A Comparative Study of Three Mapping Algorithms*

3.1.1. *XSchema*

This algorithm proposed by Mani and Lee (2002) achieves conversion from XML schemas or DTDs to relational schemas using regular tree grammar (RTG) theory.

If we assume that an RTG is a 4-tuple $G = (N; T; S; P)$, where

- N is a finite set of non-terminals;
- T is a finite set of terminals;
- S is a set of start symbols (S being a subset of N);
- P is a finite set of production rules of the form $X \rightarrow Y$, where Y may be a non-terminal symbol, a terminal symbol or a mixture of non-terminal and terminal symbols.

```
<!ELEMENT Address (Street*, City, ZipCode?, Country) >
<!ATTLIST Main_address (yes | no) "yes" >

<!ELEMENT Street (#PCDATA) >
<!ELEMENT City (#PCDATA) >
<!ELEMENT ZipCode (#PCDATA) >
<!ELEMENT Country (#PCDATA) >
<!- <People> ->
<!ELEMENT staff (Person*) >

<!ELEMENT Person (NamePerson, Email*, Address?) >
<!ATTLIST Person
personID ID #REQUIRED
role (A | B | C | D) #REQUIRED
>
<!ELEMENT Email (#PCDATA) >
```

Fig. 1. DTD fragment.

Mapping to relational schemas can be established after considering an intermediate step in which an XSchema is denoted by a 6-tuple $X = (E; A; M; P; r; S)$, where

- E is a finite set of the schema elements;
- A is a set of the schema attributes of a specified element;
- M is a function receiving an element as input and giving its composition as output;
- P is a function similar to M, but receiving an attribute as input and giving its characteristics as output;
- r is a finite set of root elements;
- S is a finite set of integrity constraints for the XML document.

Applying this definition to the DTD of Fig. 1, therefore, the following grammar (Fig. 2) is obtained.

Having reached this point in the procedure, the authors then perform inlining to add attributes to each element and generate the relational schema (relational graph).

```
E = {Person, NamePerson, Email, Address, Street, City,
     ZipCode, Country}
A(Person) = {personID, role}
M(Person) = ( NamePerson, Email*, Address? )
P(PersonID) = (ID, not null, Desc)
P(role) = (string, not null, (A | B | C | D))
.........................................
r = { Person, Address }
S = {Name, email -> Person's primary key, Street, ZipCode,
     Country ->
Address's primary key}.
```

Fig. 2. Figure 1 DTD grammar.

Taking this example into account, therefore, it can be deduced that the selection of relational constraints, such as primary keys, is not an automatic process and requires an expert user in the domain of the initial schema. Additionally, IDREFS attributes are always foreign keys. In the case of complex content models which may present several possibilities for mixing elements, the algorithm creates a table for each possibility. Finally, element order is taken into account, adding a field indicating the position of the element in the schema.

The mechanism proposed by the authors is sufficiently robust and complete. However, due to the large number of tables generated, the mechanism in some cases may also produce non-normalized relational schemata and require user intervention.

3.1.2. *Bell Labs: LegoDB*

Proposed by Bohannon *et al.* (2002) and offering an alternative to the use of regular grammars, the design of the LegoDB algorithm is based on the following three principles: cost-based searches, logical/physical independence and the reuse of existing technology. As inputs, the algorithm requires the XML document, the DTD or XML Schema, a set of statistical data about the former and a representative workload (i.e., the most frequent queries). After these requirements have been met, an intermediate schema is built and a search is run among several relational technologies in order to identify the optimal technology.

Following the procedure from the preceding paragraph, Fig. 3 displays the intermediate schema built for the DTD of Fig. 1.

As demonstrated in Fig. 3, the algorithm mixes elements with their attributes to build a type, introduces data types for all elements and takes into account the elements' cardinality. Having generated an intermediate schema, a relational technology is chosen using the workload and the intermediate schema is mapped to a relational schema applying five basic rules:

```
Type Person = person          [personId [Integer],
                              role [string],
                              namePerson,
                              Email {0, n},
                              Address*]
Type namePerson = [string]
Type Email = [string]
Type Address = address        [main [string],
                              Street {0,n},
                              City [string],
                              ZipCode {0,n},
                              Country]
type Street = [string]
type ZipCode = [integer]
type Country = [string]
```

Fig. 3. Code generated by LegoDB.

- For each type, a relation *Rt* is created.
- For each relation *Rt*, a primary key is automatically created.
- The (primary) parent key is exported to all child keys to maintain foreign keys.
- Each element included in a type is mapped to a column in the relation *Rt*.
- All required attributes are considered not null in the relational schema.

As a final analysis of the LegoDB mechanism, therefore, the following points must be stressed. First, as can be observed in the preceding discussion, LegoDB attributes do not work with IDREFS. Additionally, the automatic creation of primary and foreign keys results in semantic loss in the final schema. Finally, original element order is not taken into account.

3.2. *ODTDMap*

Proposed by Atay *et al.* (2007), ODTDMap differs from the algorithms discussed in the previous subsections in that its application is limited solely to the mapping between DTDs and relational schemata.

After first transforming a DTD into a more simplified version, the ODTDMap algorithm uses directed and ordered graphs to produce an intermediate schema with which the final conversion is carried out. As shown in Fig. 4, nodes represent elements and attributes whereas edges denote parent-child relationships. Below is a rendering of the Fig. 1 DTD following the ODTDMap procedure.

As a next step, the algorithm includes the children in the parent only if the minimum cardinality of the former is one. After this inlining process, a set of rules is defined to transform the intermediate schema into a relational schema. While this rules set takes the order of the DTD and IDREFS into account, some drawbacks of the mechanism nevertheless include the production of a non-normalized relational schema (due to the inlining process) and the inability to preserve all the semantics of the original (due to the simplification of the original DTD).

Having presented three different mapping algorithms, Table 1 summarizes the principle characteristics and drawbacks discussed for each.

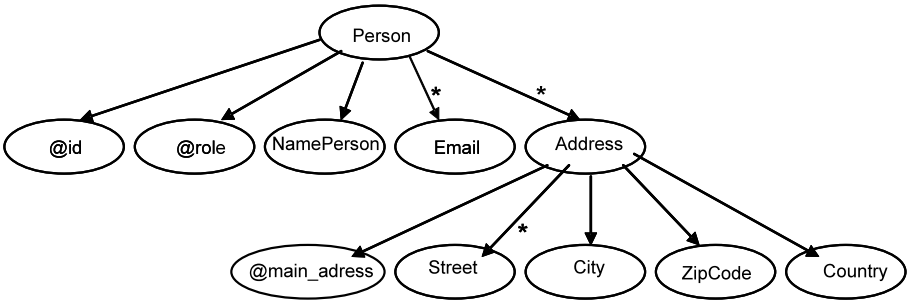

Fig. 4. Graph for Fig. 1 DTD.

Table 1
Principle characteristics and drawbacks of mapping algorithms

	Characteristics	Drawbacks
XSchema	Uses regular grammars to generate an intermediate schema	Non-normalized relational schemata
LegoDB LegoDB	Uses statistical data to generate an intermediate schema	Semantic loss; order not considered
ODTDMap ODTDMap	Uses a directed and ordered graph as an intermediate schema	Semantic loss; non-normalized relational schemata

4. Preserving Semantics in Schema Conversion

One does not engage in hyperbole by saying that perhaps all existing algorithms for mapping between XML and relational models are domain-specific. Thus, in order to attain a generalized mechanism capable of managing any schema, independent of the schema's underlying domain, an in-depth study of these algorithms is necessary. In addition, schema repositories ought to be employed to facilitate schema reuse and construction. Taking these factors into consideration, what follows is a discussion of an original mechanism allowing for the storage, management and recovery of any scheme, regardless of its underlying domain.

As all models consist of a set of constructors and a set of rules, it seems relatively easy to establish a parallelism between any two models. The only question, then, where parallelism is concerned is that of the semantics associated with each model. Due to the scarcity of the semantics in logical models, parallelism must be established between conceptual models. For this reason, a class diagram in UML is a promising candidate to facilitate this comparison, especially when taking into account the simplicity with which schemas modelled in UML can be mapped to relational databases.

Figure 5 illustrates the intermediate schema proposed for achieving this mapping described above. The translation for XML schemata is very similar. To make the schema easier to understand for the reader, stereotypes have been used as proposed in Ambler (2003).

Each DTD constructor has been mapped into classes and the relationships between which have been translated into associations in a UML diagram. A particularly important question to consider at this point of the procedure is how to capture complex content models which, in their structures, may contain multiplicities (“,”) and/or sequences (“|”). Sub-elements as well as parameter entities of a fixed element may appear with a cardinality of zero or one time (“?”), one and only one time (“ ”), one or more times (“+”) or zero or more times (“∗”). Conversely, sub-elements and parameter entities may form a chain with nesting levels using logical operators AND (“,”) and OR (“|”). All of these characteristics have been collected through two multivalued properties called “Frequency”. In order to capture complex content model semantics, an algorithm has been developed (briefly described below) for these properties, where

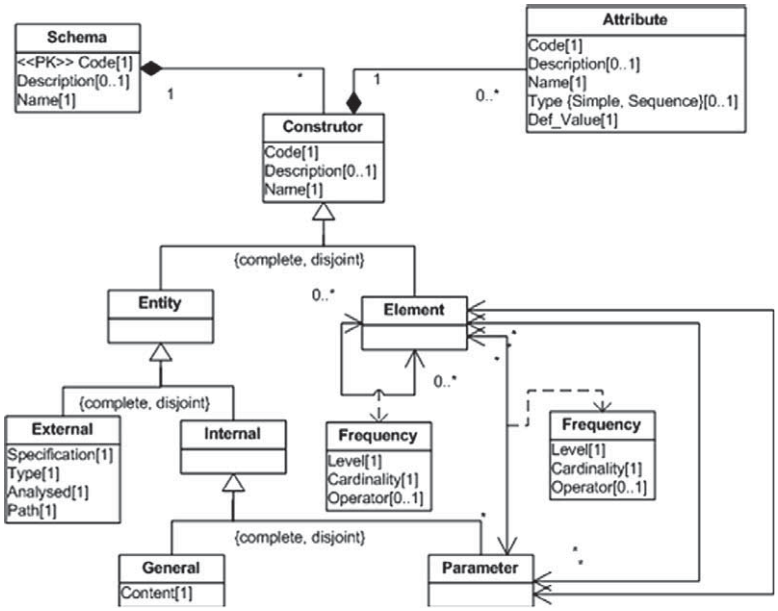

Fig. 5. Metaschema approach.

- $L = \{l_j\}_{j=1,\ldots,n}$, is a finite set of nested levels for the main element. A level is defined as the role a specific element plays in its content model (i.e., the number of possible cardinalities for an element). This set is represented by the property "Level" in the design.
- $Le_i = \{l_n - k\}$, where k is the number of open brackets to the left of the sub-element e_i, showing the first nested level for each element.
- $Card = \{$"?", "1", " $*$ ", " $+$ "$\}$, the set of all possibilities for the appearance of an element in a content model (where "1" denotes a one and only one cardinality). This set is defined as a domain "D_Cardinality" on top of the property "Cardinality" in the design.
- $Op = \{$"|", ",", "*Null*", " "$\}$, is the set made up of logical operators AND (",") and OR ("|") in a DTD specification. "*Null*" denotes the non-existence of one operator in the final nested level of an element, whereas the empty value " " reflects the case where an operator is not required to catch any other operator for a particular complex element. *Op* is also the set of values that may appear in the property "Operator" and is defined in the "D_Operator" domain.

In the proposed mechanism, the number of levels for each sub-element Le_i is calculated first. Next, the cardinality and operator for each sub-element e_i are calculated taking into account the situation of these values to the right of the sub-element. As an example (equally valid for parameter entities), suppose one element e with sub-elements $e_i,\ i = 1, \ldots, 5$, nested as follows:

$$\langle !\ \text{ELEMENT e}\ (((\text{e}_1{*}|\ \text{e}_2{+})?,\ \text{e}_3?,\ \text{e}_4) + |\text{e}_5)\rangle.$$

- $L=\{I_1\}_{I=1,\ldots,4}=\{1,2,3,4\}$
- $Le_1\{I_n-k\}=\{4\text{-}3,4\text{-}2,4\text{-}1\}$
- Le_1=1=>Cardinality(e_1)=*,Operator(e_1)=|=>e_1*|
- Le_1=2=>Cardinality(e_1)=?,Operator(e_1)=,=>(e_1*|e_2+)?,
- Le_1=3=>Cardinality(e_1)=+,Operator(e_1)=|=>((e_1*|e_2+)?,e_3?,e_4)+|
- Le_1=4=>Cardinality(e_1)=1,Operator(e_1)=Null=>((((e_1*|e_2+)?,e_3?,e_4)+|e_5)

Fig. 6. Algorithm for complex content models.

Figure 6 shows the value properties for the e_1 sub-element.

The algorithm presented above allows one to capture the semantics associated with mixed and complex content models and, as shown in Fig. 5, to obtain a semantic repository for web schemata. In addition, due to the fact that the metaschema captures the order of any element or "Code" attribute and of the ROWID during the transformation to the relational schema, the mechanism allows for the retrieval of the original schema and the reuse of schema fragments during the building of new schemata. In short, the use of a conceptual model or metaschema appears to be a good mechanism for mapping between DTDs or XML schemas and relational databases insofar as the transformation produces a normalized solution.

5. Experimental Study

In order to demonstrate the soundness of the proposal presented in the previous section of this paper, two questions must be taken into account: (1) does the output schema capture the same semantics as the original schema and (2) is the original schema retrievable after the translation? While both questions seem rather similar, the former refers specifically to the preservation of the semantics intrinsic to a DTD (e.g., order, complex content models, ID and IDREFS, etc.), whereas the latter refers directly to the efficiency (in terms of structure normalization) of the final relational schema.

What follows is a discussion of the experimental application of these questions to the mechanism presented in Section 4 of this paper. The first subsection outlines this experiment while the second subsection presents and analyses the results obtained.

5.1. *Experiment Description*

The efficiency and optimization of a system depends not only on the technology used, but also on a good application design (since good design leads to proper implementation and avoids the typical deficiencies of systems that are not based on normalized schemata). In this respect, one of the advantages of relational systems is that they are easy to optimize due to their high-level relational operations. The system detailed in this paper was developed using Oracle 10g, chosen here for its importance among RDBMSs currently on the market.

In this experiment, a repository of thirteen free, non-commercial DTDs with different sizes, levels of complexity and purposes was used such that, through query analysis,

the ability of the proposed algorithm (1) to preserve original DTD semantics and (2) to function efficiently could be clearly demonstrated. Table 2 below presents each of the DTDs from this repository along with their respective purpose and file size.

In addition to the creation of the DTD repository, several queries were designed to simulate end-user requests. These queries can be classified into two groups: complete and partial queries.

Complete queries are all those related to the obtainment of a complete DTD and could be useful in cases where a user must retrieve a predefined schema. To achieve this goal, a query was implemented to retrieve any schema together with its constructors and attributes.

Partial queries allow for the retrieval of DTD fragments for their reuse in a new schema. These queries can be further classified into two additional groups: simple queries (i.e., queries for isolated elements in the DTD) and complex queries (i.e., queries for DTD constructors with their content models). In the present experiment, simple queries were designed to measure response time in the retrieval of simple constructors, while complex queries were designed to retrieve complex content model elements.

All RDBMSs have an optimizer module selecting the best data access path for a query depending on the style in which the query is written. However, in this experiment where queries are not very difficult and their operators are not overly complex, different trials run with several versions of the same sentence do not yield significantly different results.

Table 2
DTDs stored in repository

DTD name	DTD purpose	DTD size (bytes)
Records.dtd	DTD modelling a record collection	576
Log.dtd	DTD providing an XML-formatted log message	2,517
Message.dtd	DTD modelling electronic messages	633
Music.dtd	DTD for Music Markup Language (MusicML), an XML application displaying musical notation on web pages	3,141
NewsML.dtd	DTD for News Markup Language (NewsML), an XML standard for the tagging of news stories and related documents (developed by the International Press Telecommunications Council)	13,964
Novel.dtd	Simple XML DTD for marking up novels	1,453
Osd.dtd	Vocabulary (Open Software Description Format) for describing software packages and their dependencies for heterogeneous clients	1,267
Play.dtd	DTD for Shakespearian plays	1,138
Pml.dtd	DTD for Portal Markup Language (PML) describing portal related data and metadata and allowing the exchange of information across portals of the same or different manufacturers	9,064
Preferences.dtd	DTD for preference tree	1,256
Tstmt.dtd	DTD for wills and testaments	1,074
Xcard.dtd	DTD for electronic business cards	1,104
Xmi.dtd	DTD for XML Metadata Interchange	3,376

Finally, in this experiment, especially taken into account were the parameters of response time and catalogue statistics. That said, however, due to the fact that response times change with the number of records requiring processing rather than with the complexity of the query, parameters between queries were not compared here.

5.2. *Experiment Results*

In order to gauge the utility of the proposed algorithm with regard to the two criteria described in Section 5.1, the present experiment took into account the number of records retrieved from the repository and the corresponding retrieval times (measured in milliseconds) for each query. The results from this experiment for complete queries suggest that schema size is directly related to retrieval time (i.e., the larger the schema, the longer the retrieval time); while for partial queries searching for constructors or schema fragments, retrieval time does not appear to have been much affected by the size of the DTD. In order to clearly demonstrate that that these results do not indicate a lack of system efficiency, the average response time for each query was calculated and is presented below in Fig. 7.

Of the three queries with the highest response times according to Fig. 7, Query 1 had requested a complete schema, Query 3 had requested parameter entities included in elements and Query 6 had requested the complex content model. Additionally, these three queries had more joins than the rest, resulting in a greater number of tables that required accessing.

Finally, to determine whether the complexity of the schemata is an indicator of system efficiency (particularly in cases where the size of a DTD did not affect retrieval), each schema was assigned a number representing its relative complexity. In order to do so, all constructors and their relationships for each particular schema were taken to account, as were the ways and the number of operations necessary to achieve each one. Schema elements, therefore, were assigned a corresponding weight depending on these complexity factors where elements received a weight of zero (since every schema had to have at least one element), while content elements were accorded the greatest weight (due to their nested structure). As a last step, the total complexity of each individual schema was

Queries	Averages
Query 1	223.230769
Query 2	200.769231
Query 3	232.076923
Query 4	134.615385
Query 5	163.384615
Query 6	236.769231
Query 7	145.30762

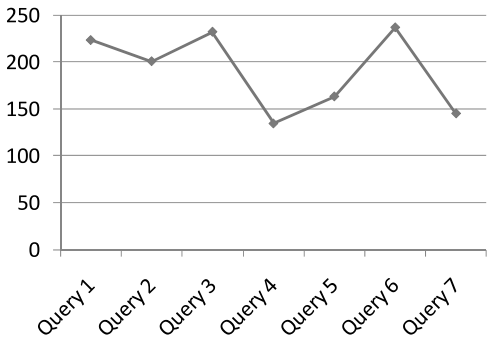

Fig. 7. Average query response times.

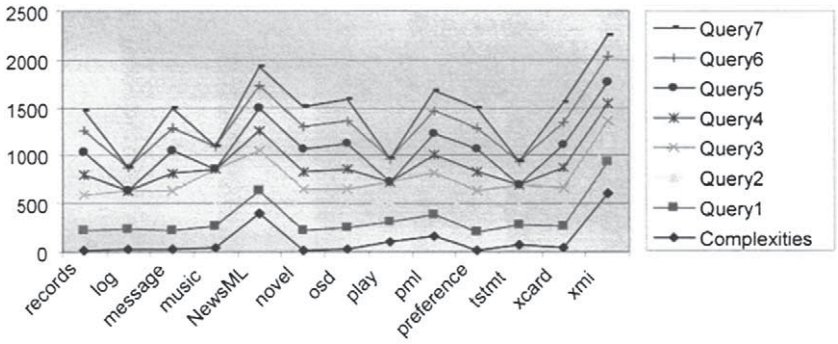

Fig. 8. Cost relative to DTD complexity.

calculated as the sum of the complexity weights given to each of the items and elements in the schema. In this way, a cost for each constructor in each schema was obtained.

Following this step, complexity measurements and query response times were compared for each corresponding schema. As evidenced by Fig. 8, it can be concluded that schema complexity directly affects retrieval time, insofar as increases in the former resulted in proportional increases in the latter. That said, and following other criteria, DTD fragment retrieval time was quite acceptable, taking an average of 185 milliseconds. Therefore, the relational schema appears to be efficient with regard to the number of tables and joins. In addition, complete queries retrieved schemas as well as their complete original semantics.

It seems obvious that the size and the complexity of a DTD may influence response times. At the beginning of this section it was noted that the experiment should serve to demonstrate the robustness of the proposed algorithm by taking the following two points into consideration: the possibility of retrieving a complete DTD with full capture of its original semantics and the efficiency of the relational database generated. As regards the former and using complete queries, any DTD can be reconstructed in the same order and with the same structures (in the same position) by the ROWID of each record. As regards the latter, the number of tables generated in transforming the conceptual schema of Fig. 5; following the methodology presented in Elmasri and Navathe (2006) ensures a normalized result that positively affects schema efficiency.

6. Conclusions

This paper presents a review of recent research on mapping between XML and relational databases. In any XML formalism or schema, the order in which elements are written is fundamental for the reconstruction of the original document; however, the relational model does not take this order constraint into account. Thus, this paper has analyzed how algorithms can be designed to accurately preserve this order and the original semantics.

Furthermore, this paper presented an overview of three recent algorithms and a new proposal for handling XML models. This proposal emphasized the need to capture as

many original semantics as possible (including the DTD order of the constructs), as well as the importance of obtaining a normalized relational schema. The benefits offered by the proposal were demonstrated in the discussion of experimental results, with special attention being paid to the storage and management of DTD schemas.

In the final section of this paper, the need for future research efforts in the open field of XML storage and management will be briefly discussed.

7. Future Work and Trends

Further research on mapping between XML and relational models is necessary for the acquisition of a reliable automatic mapping mechanism. Currently, with the help of data models and meta-constructors (the latter permitting the representation of the constructors of any model (Atzeni *et al.*, 2005; Bernstein *et al.*, 2000), experts are researching techniques for automatic conversion between any two models.

Nevertheless, and as repeated throughout this paper, any mapping between two models must allow for the reconstruction of the original schema and, moreover, it should be reusable. This reusability means not only the possibility to retrieve fragments of schemata or a complete schema, but also the original semantics. Moreover, the reusability is linked as the attainment of the long-awaited interoperability. However, without the use of ontologies or controlled vocabularies (Gomez-Pérez and Benjamins, 2002; Magdalenic and Radosevic, 2009), this goal is not being met.

Focusing specifically on the proposal presented in Sections 4 and 5 of this paper, it could be productive to develop a detailed study around a mechanism for the inclusion of XML document collections in their underlying and stored schemata.

Acknowledgements. This study is the result of work undertaken as part of the "Software Process Management Platform: modeling, reuse and measurement". Project number: TIN2004/07083.

References

Ambler, S.W. (2003). *Agile Database Techniques*. Wiley, New York.

Atay, M., Chebotko, A., Liu, D., Lu, S.,Fotouhi, F. (2007). Efficient schema-based XML-to-relational data mapping. *Inform. Syst.*, 32(3), 458–476.

Atzeni, P., Cappellari, P., Bernstein, P.A. (2005). ModelGen: model independent schema translation (demo). In: *Intenational Conference on Data Engineering, ICDE*.

Bernstein, P.A., Levi, A.Y., Pottinger, R.A. (2000). A vision for management of complex models. *Microsoft Research Technical Report MSR-TR-2000-53*.

Bohannon, P., Freire, J., Roy, P., Siméon, J. (2002). From XML schema to relations: a cost-based approach to XML storage. Bell Labs. In: *Data Engineering, Proceedings 18th International Conference on on Data Engineering (ICDE'02)*, pp. 64–75.

Bourret, R. (2001). Mapping DTDs to databases. Retrieved January 21, 2003. `http://www.XML.com/lpt/a/2001/05/09/DTDodbs.html`.

Castro, E. (2004). *An approach to the documental structures reusability in the frame of the semantic Web*. Doctoral thesis. Carlos III University of Madrid, Spain.

Conrad, R., Scheffner, D., Freytag, J.C. (2000). XML conceptual modeling using UML. In: *19th International Conference on Conceptual Modeling (ER'2000)*, pp. 558–571.

Deutsch, A., Fernandez, J., Suciu, D. (1999). Storing semistructured data with STORED. In: *Proceedings ACM SIGMOD International Conference on Management of Data*, Philadephia, PA USA, pp. 431–442

Dos Santos Mello R., Heuser, C.A. (2001). A ruled-based conversion of a dtd to a conceptual schema. Retrieved February 2, 2003. `http://citeseer.nj.nec.com/500727.html`.

Elmasri, R., Navathe, B. (2006). *Fundamentals of Database Systems*, 5th ed., Addison-Wesley, Boston.

Florescu, D., Kossman, D. (1999). Storing and querying XML data using RDMBS. *IEEE Data Eng. Bull.*, 22(1), 27–34.

Gomez-Pérez, A., Benjamins, V.R. (2002). Knowledge engineering and knowledge management: ontologies and the semantic web. In: *13th International Conference on Knowledge Engineering and Knowledge Management, EKAW'2002*. LNAI, Vol. 2473, Sigüenza, Spain. Springer, Berlin, pp. 114–121.

Lee, D., Mani, M., Chu, W.W. (2002a)). NeT & CoT: Inferring XML schemas from relational world. In: *Proceedings of the 18th International Conference on Data Engineering*.

Lee, D., Mani, M., Chu, W.W. (2002b). NeT & CoT: translating relational schemas to xml schemas using semantic constraints. In: *Proc. 11th ACM Int. Conf. on Information and Knowledge Management*, McLean, VA, USA, pp. 282–291.

Lee, D., Mani, M., Chu, W.W. (2003). Schema conversion methods between XML and relational models. In: *Knowledge Transformation for the Semantic Web, Frontiers in Artificial Intelligence and Applications*, Vol. 95, IOS Press, Amsterdam, pp. 1–17.

Liu, M., Ling, T.W. (2000). A conceptual model for the web. In: *Proceedings 19th International Conference on Conceptual Modeling (ER'2000)*, pp. 225–238.

Magdalenic, I., Radosevic, D. (2009) Dynamic generation of web services for data retrieval using ontology. *Informatica*, 20(3), 397–416.

Mani, M., Lee, D. (2002). XML to relational conversion using theory of regular tree grammars. In: *Proceedings of the 28th VLDB Conference*, pp. 81–103.

McBrien, P., Poulovassilis, A. (2001). A semantic approach to integrating XML and structured data sources. In: *13th Conference on Advanced Information Systems Engineering*.

Pokorny, J. (2009). XML in enterprise systems. *Informatica*, 20(3), 417–438

Shanmugasundaram, J., Tufte, K., He, G. (1999). Relational databases for querying XML documents: limitations and opportunities. In: *Proceedings of the 25th VLDB Conference*, pp. 302–314.

Tatarinov, I., Viglas, S.D., Beyer, K., Shanmugasundaram, J., Shekita, E., Zhang, Ch. (2002). Storing and querying ordered XML using a relational database system. In: *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*, pp. 204–215.

W3C (2007). The world wide web consortium. Retrieved March 1, 2007. `www.w3c.org`.

E. Castro received a MSc in mathematics from the Complutense University of Madrid in 1995. Since 1998, she has been working in the Advanced Databases Group in the Computer Science Department at the Carlos III University of Madrid. In 2004, she obtained a PhD in information science from the Carlos III University of Madrid. She currently works as associate professor and is currently teaching relational and object oriented databases. Her research interests include database conceptual and logical modeling, advanced database CASE environments, information and knowledge engineering, natural language processing and E-learning fields.

D. Cuadra received the MSc in mathematics from Universidad Complutense of Madrid in 1995. In 1997, she joined the Advanced Databases Group, at the Computer Science Department of Carlos III University of Madrid, where she currently works as associate professor. In 2003, she obtained the PhD degree in computer science from Carlos III University of Madrid. Her research interests include advanced database technologies, spatio-temporal databases and their applications to Situation management. She has been working in Computer Science Department at Purdue University of West Lafayette (Indiana) for nearly a year, where she has applied her research in spatio-temporal database. Apart from this, she is working on several national research projects regarding human-computer interaction, specifically on cognitive models for natural interaction systems.

M. Velasco received the MSc in computer science from Universidad Complutense of Madrid in 1992. In 1993, he joined the Software Engineering Group, at the Computer Science Department of Carlos III University of Madrid. In 1998 he obtained the PhD degree in computer science in the Artifitial Intelligence Department from Universidad Politecnica of Madrid. Since 1999 he works as associate professor at Carlos III University of Madrid. His research interests include software reuse, software testing and spatio-temporal databases. Apart from this, he is working on several national research projects at SEL (Software Engineering Laboratory).

XML schemų atvaizdavimas į reliacinius modelius

Elena CASTRO, Dolores CUADRA, Manuel VELASCO

Kuriant daugelį veiklos ir organizacijų sistemų vienu iš svarbiausių siekiamų tikslų yra jų interoperabilumas. Tačiau šio tikslo neįmanoma pasiekti, neturint vienų schemų atvaizdavimo į kitas efektyvaus mechanizmo arba tokių modelių, kurie sudaro galimybes saugoti ir tvarkyti informaciją keliose skirtingose sistemose. Kadangi XML yra informacijos mainų standartas, yra atlikta gana daug įvairių tyrimų bandant sukurti efektyvius metodus bei algoritmus XML schemoms transformuoti į duomenų bazių modelius. Straipsnyje apžvelgiami svarbiausi šios srities tyrimai, pagrindinį dėmesį skiriant trims naujausiems tyrimams, ir siūlomas originalus koncepciniu modeliu grindžiamas schemų atvaizdavimo mechanizmas, kuris savo aukšto abstrakcijos lygio dėka maksimizuoja semantikos išsaugojimą.