2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops

Managing Schema Evolution in Hybrid XML-Relational Database Systems

Abdullah Baqasah and Eric Pardede
Department of Computer Science and Computer Engineering
La Trobe University, Melbourne VIC 3086
Email: {ambaqasah@students.; E.Pardede@}latrobe.edu.au

Abstract

Many applications and data enterprises need to change their data content and structure from time to time. The schema evolution in database systems has been studied well by the database community and several projects have been proposed for supporting schema modifications in XML and relational models individually. However, these studies do not investigate well the evolutionary process in hybrid XML-relational systems. In this paper we will investigate the schema evolution for current hybrid XML-relational systems. We will propose a taxonomy for schema evolution in a way that preserves a schema after data evolves. Finally, we proposed a general model which will be evaluated through a case study.

1. Introduction

XML (eXtensible Markup Language) is merging as a standard de facto format for representing data over the web. On the other hand, relational data model will continue to serve as dominant technology for storing and querying a structured data. As a consequence, commercial DBMS (Database Management System) vendors have long recognized the demand for hybrid XML-relational systems. Majority of them start introducing a new datatype, `XMLType`, to facilitate native handling of XML data in their databases.

While major database vendors such as IBM DB2 9.5 [8], Oracle XML DB [14], and Microsoft SQL Server 2005[12] support XML data within their relational products, few industry and research projects have ventured to suggest scenarios and methodologies for such a new hybrid design environment.

During the time, many of these hybrid XML-relational systems require modifications in their data and schema as well. The problem of data modification is well studied by research community and it is solved by a temporal database [18], whereas, modifications on the schema, especially, in the hybrid models, have not been addressed yet and represents a serious needs in the industry.

To cope with the problem of schema evolution, this paper will firstly study the existing schema evolution solutions in the hybrid XML-relational systems. Then, it will describe the general architecture for managing schema evolution in the hybrid-XML relational systems. Finally, a taxonomy of schema evolution operations in such the hybrid systems will be introduced with an appropriate case study.

Roadmap. Section 2 introduce key concepts for XML-relational systems and schema evolution. Section 3 surveys related work. Section 4 discusses the evolution process in a hybrid model which leads to creating a major architecture for modeling schema evolution in Section 5. Section 6 introduces our classifications for schema evolution operations in the hybrid DB model and applies it in the case of an invoice system). We will conclude the paper in Section 7.

2. Background

2.1. Hybrid XML-Relational Model

Many of today's database vendors support RDBMSs as a major solution for data management. The relational model [1] represents the database as a collection of relations. Each relation contains a set of tuples and attributes. In the relational model, updates are controlled by DBMS. For example, changing attribute's datatype would be prevented by the DBMS because data stored in that column is already propagated to a certain datatype.

XML as a new data storage model has the following characteristics: (1) it is suitable for storing semi-structured and unstructured data, (2) it is self-describing, and (3) its portable and compatible with different platforms and applications. Thus XML model is utilized to store and retrieve data in certain cases.

Hybrid model is a system that supports XML and relational data in the same environment. It is becoming more significant nowadays for several reasons: (1) data of an organization is no longer be exclusively relational, (2) some organizations necessitate some of their data to be hybrid [7] and changeable, (3) challenges for organizations to come up with the changing market and their requirements [13].

Figure 1 describe two different methods to support hybrid XML-relational system. The first one,

978-0-7695-4019-1/10 $26.00 © 2010 IEEE
DOI 10.1109/WAINA.2010.127

which is supported by most commercial database systems, shreds XML data into relational tables. The second one stores XML data as character large objects (CLOBs) in the relational systems [3]. It seems that mapping between two models is straightforward process, however it involves many issues such as when XML node is said to be an attribute in the table and when it should be a new relational table.

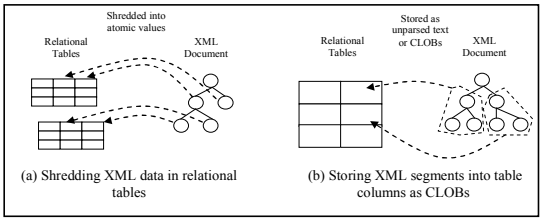


Fig.1. Two methods for storing XML data in RDBMSs.

There are various work dedicated to investigate the storage and management issue of hybrid XML-relational data. In this paper, we will focus on one area that has not been fully explored, the schema evolution in this database system.

2.2. Data Schema Evolution

Many database applications such as CAD systems and medical records require updates in the course of time. While relational structure can be generally modified by the 'ALTER' SQL statement, in the XML documents we cannot clearly distinguish between updating values and altering document schema because updates of XML documents may involve updates of values and changes of structures [10]. Database schema need to change for several reasons such as user requirement changes and the rapid changes in database references [15]. For the XML model, XML query language XQuery [23] is used to update the document and schema.

To avoid ambiguity, two definitions should be introduced at the beginning; "schema modification" and "schema evolution".

Schema Modification as denoted by [18] is introduction of changes into the schema of a populated database. For example, addition, deletion of types in XML model and changing relations between types. *Schema Evolution* as denoted by [18] is incorporation of changes into the schema of a database without loss of data. Examples of such evolution cases are adding or deleting data elements.

In this paper, we will focus on the management of schema evolution in hybrid XML-relational system only.

3. Related Work

3.1. Schema Evolution in Relational Model

Schema evolution in the relational model is investigated by the research community from different perspectives. The earlier work done by [20] discusses the reorganization of the database relations in an efficient manner. It tackles two problems that could occur in this reorganisation: suspension of database and inefficient storage.

Few years later, a classification of schema evolution is proposed by [16]. It describes the most common modifications to a database schema by classifying operationally changes into four major groups: domain/attribute evolution, relation evolution, attribute-relation assignment evolution and schema transaction support.

Another work is done by [17] and discusses data modeling issues, architectural issues, and query language issues that relates to the supporting of schema evolution in relational system.

3.2. Schema Evolution in XML Model

In XML data model, document schema is stored externally. For this reason, when we modify the schema, the XML documents associated with it have to be adapted. The more recent work in the XML schema evolution [11] shows how schema evolution can be realized on a conceptual model by using an approach called CoDEX.

For XML schema evolution in the native format, [19] has created a framework called the XML Evolution Manager (XEM), to manage the evolution of DTDs and XML documents. The main idea of XEM approach is to represent DTD as a graph with several types of vertices: vertices-elements, vertices-constraints and vertices-data. Elementary modifications in this approach are given in [19].

The work in [2] extended the other work which has defined an axiomatic model for XML-database schema evolution by including the DTD semantic constraints in the evolution model.

3.3. Schema Evolution in Hybrid XML-Relational Model

Hybrid XML-Relational systems as explained before deal with two different data models, therefore, the problem of creating a new schema to describe the data or even semantic changes should be considered for both XML and relational data models.

In order to manage the schema evolution in such hybrid systems we should know firstly how hybrid systems are constructed, and that is discussed in the next section. Mapping between two different models, which "specifies how data instances of one schema correspond to data instances of another" [21], is also significant for evolving the schema.

Some XML-enabled DBMSs as an example of hybrid model support XML schema evolution in their systems. Table 1 list schema evolution and validation support in the major commercial DBMSs. In Oracle XML DB, for instance, two kind of schema evolution are supported: Copy-based schema evolution and In-place schema evolution [6].

Table 1. Schema evolution support in commercial system (table adapted from [6])

DBMS	Type of schema	Validation	Schema Evolution
SQL Server	Subset XML Schema	total	no
DB2	XML Schema	total	partially
Oracle 11g	XML Schema	Partial and total	yes

4. Modeling Hybrid XML-relational databases

The significance of managing hybrid XML-relational data cannot be overlooked. Nowadays online transactions and a serious amount of business data is stored in relational format and utilize XML as a data exchange format. Different techniques have been developed to manage XML besides relational data. Based on W3C, there are two main categories for storing and querying XML into relational RDBMS: *structure-mapping approaches* and *model-mapping approaches* [22].

4.1. Model-Mapping Approaches

4.1.1. Edge approach: In this approach [3] we store XML data graphs in a table called "Edge Table". The Node table schema takes this form:

EdgeTable (Source, Ordinal, Label, Flag, Target, Value)

It is a simple scheme to maintain edges individually in one table. Therefore, it needs to concatenate edges to form the path of user queries. Because this approach is designed for schemaless XML documents, XML schema evolution techniques that proposed by [19] and [5] cannot be employed with this approach.

4.1.2. Node Approach: It is developed by [25], in which all internal nodes (elements and attributes) are stored in a table called "Node". The schema of the table is as follows.

NodeTable (Label, Start, End, Level, Flag, Value)

The key idea here is using a triple attributes (start, end, level) which represents the start and end point, and the level of the node.

4.1.3. Path-Based Approach: This method, also called "XRel" approach, was presented by [24] where XML data graphs are stored in four tables.

Path (PathId, PathExp)
Element (DocId, PathId, Start, End, Ordinal)
Text (DocId, PathId, Start, End, Value)
Attribute (DocId, PathId, Start, End, Value)

The key feature of this approach is that instead of maintaining edge information in the schema, a region maintains a containment relationship [9]. The concept region is defined as a pair of "start" and "end" point of a node.

The advantage of this approach is the ability of identifying weather a node is in a path of another node by using joins. However, lack of an efficient schema definition in this approach is disadvantage because XML documents that rely on DTDs, or any other schema languages, may be invalidated when the data is returned to its original XML format.

4.1.4. XParent approach: Same as "XRel" approach, however, it needs another table called "Ancestor" table which maintains "ancestor-descendent" relationship.

Ancestor (Did, Ancestor, Level)

In this table "Did" is a data-path identifier, "Ancestor" and "Level" maintains "ancestor-descendent" relationships.

4.2. Structured-Mapping Approaches

There is one approach in this context, which is called *DTD approach*.

All approaches that are described earlier, store and query XML documents which do not conform to a specific DTD or XML-Schema. Such XML documents called general (schemaless) data. However, some applications especially for those relying on specific data classification require storing DTD for its XML documents.

The DTD approach, investigated by [4] converts XML query to SQL in order to retrieve a certain records. The schema of the database is firstly mapped. Then, the data are mapped to the relational tables.

5. Proposal

In this section we propose our evolution model depending on the hybrid XML-relational existing systems. In most cases, there are three types of schemas already exist in the system: relational, XML and XML stored in relational tables. Figure 2 shows different possibilities for schema changes in the hybrid model.

5.1. Proposed Architecture

Figure 3 shows two types of changes on the data. The first, document-based changes, is in the form of XQuery statements to update the schema. The second, table-based changes, is the relational updates such as ALTER statement in SQL.

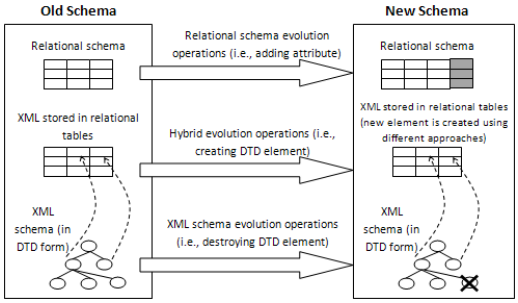


Fig 2. Schema changes scenarios in hybrid model

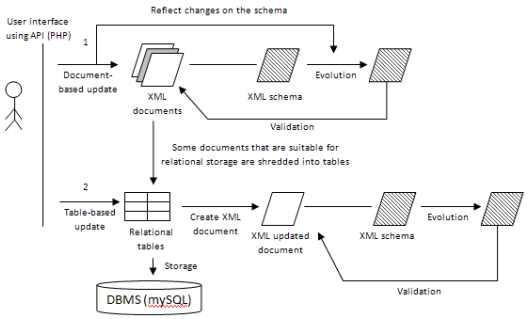


Fig 3. Proposed architecture for hybrid system evolution

When a user makes the evolution by persisting a portion of the data in a specific model, the system will apply one of the previous mentioned approaches to transfer data from one model to another model. Then, if some XML documents already stored in relational tables, it is difficult to infer the schema of populated data because Edge, Node, or Path-based approaches map XML documents regardless of the schema. Thus, the only way to achieve an efficient evolution is by using DTD approach which maps the schema first then the XML documents associated with the schema.

In the following sections we introduce taxonomies that are proposed by [16] for the relational model and [19] for the XML model.

5.2. A taxonomy of Relational and XML Schema Evolution Operations

At the designing time, a number of design criteria are taking into account in order to manipulate schema evolution efficiently. For relational subsystem, the evolution modifications rely on the ER diagram which is related mainly with the relational model. Sample classification is given in table 2.

We also classify XML schema evolution model. First, the evolution operations for XML schema are listed, and of course, XML documents because the self-describing XML documents are a very important part that should take place in the evolution process. The taxonomy of DTD and XML data change primitives is proposed by [19]. Sample classification is given it table 3.

Table 2. Relational Schema Evolution Sample Operations

Entity Evolution Operation	**Description**
Add entity	Add a new entity (table) with unique name using SQL *create table* statement.
Change weak entity to regular entity	The identifying attributes of the parent entity are demoted.
Attribute Evolution Operation	**Description**
Add attribute to relationship	Add a new attribute to the corresponding relationship.
Rename attribute	Rename the corresponding attribute.
Relationship Evolution Operation	**Description**
Add relationship between two or more entities	Create a relationship between two or more entities that do not have relationships.
Delete relationship	Delete a relationship between two or more entities that do have relationships.

Table 3. XML Schema /DTD Evolution Sample Operations

DTD evolution operation	**Description**
Create DTD element	Create target element with name and content type
Flatten group	Flatten group in target element
XML document evolution operations	**Description**
Add data element	Add element node to another element node as a child
Destroy data attribute	Destroy attribute with name in target element node

5.3. A taxonomy of XML-in-relational Schema Evolution Operations

Operations for XML data, that is stored in the relational DBMS, are significant; hence many applications update their XML documents without concerned about XML schema. If users aim to perform some data changes the system must follow a certain rules in order to preserve the consistency of XML document. A set of primitive changes operations and their implications are produced in table 4. Figure 4 is used to show the operations

Table 4: XML-Relational Schema Evolution Sample Operations

Operation	**Description**	**Conditions**
Create data element	Create target element node as a new tuple with label, type (element or leave node) and value if it is leave node (Fig. 4(a)).	Data element is created if it is already defined in the DTD or XML-Schema.
Add data element	Add element node to another element node as a child by updating all fields in the tuple that represent the relationship between a parent & child elements (Fig. 4(b)).	Data element is added if it is already defined as a subelement in the parent element definition in the DTD or XML-Schema.
Destroy data attribute	Destroy attribute by deleting it from the table and thus, all fields that represent the hierarchy of the tree should be updated (Fig. 4(c)).	Data attribute should be deleted from the attribute list defined in DTD or XML-Schema, or the property should be changed.

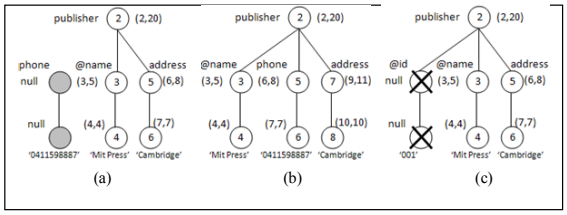


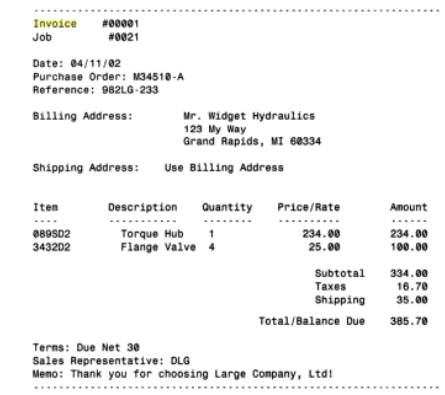
Fig 4. Evolution operations example

6. Case Study: Invoice System

6.1. Overview

In our experiment, we use the invoice system as a suitable usecase for hybrid model. Although, it has a

few entries (XML nodes), our aim is to test the correctness not the performance of the data model. Figure 5 shows the invoice sample motivating some key issues such as: (1) how atomic data should be stored in the data repositories? (2) should data be stored in relational, XML, or both? (3) how the data can evolve in the case of storing as relational, XML or both?



Invoice #00001
Job #0021

Date: 04/11/02
Purchase Order: M34510-A
Reference: 982LG-233

Billing Address: Mr. Widget Hydraulics
123 My Way
Grand Rapids, MI 60334

Shipping Address: Use Billing Address

Item	Description	Quantity	Price/Rate	Amount
089SD2	Torque Hub	1	234.00	234.00
3432D2	Flange Valve	4	25.00	100.00
			Subtotal	334.00
			Taxes	16.70
			Shipping	35.00
			Total/Balance Due	385.70

Terms: Due Net 30
Sales Representative: DLG
Memo: Thank you for choosing Large Company, Ltd!

Fig. 5. Invoice hybrid XML-relational example

By understanding business artifacts and application requirements there are different ways to store the invoice information. Storing it as hybrid XML-relational data would be the best solution because some data with high schema variability and sparse data should ideally be persisted as XML.

The XML representation of the invoice example is shown in figure 6 where each independent entity is presented as a node. However, some information is suitable for relational model such as invoice number, data, billing and shipping address, and purchase order because there is no need to store it as XML. Moreover, some data is derivative such as total, so it does not need to be stored as XML. The enhanced XML document in figure 7 would have items information which is more likely to persist as XML due to its schema variability.

```
<invoice number="00001" job="0021">
  <date>04/11/02</date>
  <billingAddress>
    <company>Mr. Widget</company>
    <address>123 My Way</address>
    <city>Grand Rapids</city>
    <state>MI</state>
    <zip>60334</zip>
  </billingAddress>
  <shippingAddress useBilling="yes"/>
  <PO PONumber="M34510-A" reference="982LG-233"/>
  <item code="089SD2">
    <description>Torque Hub</description>
    <quantity>1</quantity>
    <price_rate >234.00</price_rate >
    <amount>234.00</amount>
  </item>
  <item code="3432D2">
    <description>Flange Valve</description>
    <quantity>4</quantity>
    <price_rate>25.00</price_rate >
    <amount>100.00</amount>
  </item>
  <subtotal>334.00</subtotal>
  <tax taxRate=".05">16.70</tax>
  <charge type="shipping">35.00</charge>
  <total>385.70</total>
  <term>Due Net 30</term>
  <rep name="DLG" dept="sales"/>
  <message>Thank you for choosing … </message>
</invoice>
```

Fig. 6. Invoice XML document example.

```
<?xml version="1.0" encoding="UTF-8" ?>

<items>
 <item code="089SD2">
  <description>Torque Hub</description>
  <quantity>1</quantity>
  <price_rate>234.00</price_rate>
  <amount>234.00</amount>
 </item>
 <item code="3432D2">
  <description>Flange Valve</description>
  <quantity>4</quantity>
  <price_rate>25.00</price_rate>
  <amount>100.00</amount>
 </item>
</items>
```

Fig 7 Enhanced Invoice XML document example.

6.2. Analysis

In this subsection we qualitatively analyze different evolution scenarios to the current system. Modifications for the schema are divided into two major categories; document-based changes and table-based changes as shown in figure 3. While updates in the first category are well studied by many research projects such as [5, 10, 19], the latter category would have more investigation by researchers and data designers. The aim of the analysis is to show how our approach would facilitate working with schema evolution in the hybrid model.

6.2.1. Document-Based Updates occurs in the XML schema such as adding or deleting element and attributes. Scenarios for our case study can be shown as follows:

- *Adding new element* "supplier".
  DTD Definition: `<!ELEMENT item (description, quantity, price_rate, amount, supplier+)>` `<!ELEMENT supplier (#PCDATA)>`.
  Implications: Suppliers data should be added after "amount" element in all associated XML documents, and each "item" element should have at least one "supplier" child element. After the evolution process schema should be validated against all associated documents.
- *Deleting element* "supplier".
  DTD Definition: `<!ELEMENT supplier (#PCDATA)>` should be removed and "item" element definition should be modified to `<!ELEMENT item (description,quantity, pricae_rate, amount)>`.
  Implications: All "supplier" elements should be removed from the associated XML documents. And revalidation is required.

6.2.1. Table-Based Updates occurs to the schema of XML that are stored in relational tables. The following scenarios are suggested:

- *Adding new element* "customer_no".
  In this scenario, evolution steps differ depending on the approach used for storing XML documents. Our results indicate that "Edge", "Node", "Path-Based" and "XParent" approaches require updates for all XML elements that are stored in tables, whereas "DTD" approach does not require updates for the propagated documents.
- *Deleting element* "PONumber".
  This operation sometimes involves updates for the table structure in "DTD" approach. However, updates in this case depend on whether we want to delete "PONumber" from all "invoice" elements or some of them. In other words, delete definition of "PONumber" from the schema file or decrease the "PONumber" quantifier so that "+" becomes "*". At the end, the last case does need requires updates of table structure when using "DTD" approach.

Table 4 compares between different hybrid approaches in applying sample schema evolution operations. Note that "schema updates" column indicates whether XML schema need to be updated or not.

Table 4. Hybrid Approaches in Schema Updates Comparison

XML storage app.	Table Data Updates	Table Structure Updates	Schema Updates
Adding new element			
Edge	yes	no	yes
Node	yes	no	yes
Path-Based	partially yes	no	yes
XParent	partially yes	no	yes
DTD	no	partially yes	yes
Deleting existing element			
Edge	yes	no	no
Node	yes	no	no
Path-Based	yes	no	no
XParent	yes	no	no
DTD	yes	partially yes	yes

7. Conclusion and Future Work

In this paper we discuss the schema evolution for hybrid XML-relational systems. A general structure and a taxonomy for evolution operations in different format and implications for each operation are provided.

Managing schema evolution in the XML data that are stored in relational DBMS is well studied. However, storing relational data in the XML format would be an area of study as a future work in the schema evolution. In addition, more sophisticated schema evolution operations have to be considered and applied with real usecases.

8. References

[1] E. Codd, "A relational model of data for large shared data banks," Commun. ACM, vol. 13, pp. 377-387, 1970.

[2] S. Coox, and A. Simanovsky, "Regular expressions in XML schema evolution," vol. 1, pp. 24-38, 2004.

[3] D. Florescu, and D. Kossmann, "Storing and querying XML data using an RDMBS," IEEE Data Eng. Bull., vol. 22, pp. 27-34, 1999.

[4] G. Gou, and R. Chirkova, "Efficiently querying large XML data repositories: A survey," IEEE Trans. on Knowl. and Data Eng., vol. 19, pp. 1381-1403, 2007.

[5] G. Guerrini, M. Mesiti, and D. Rossi, "Impact of XML schema evolution on valid documents," in WIDM 2005.

[6] G. Guerrini, and M. Mesiti, "XML schema svolution and sersioning: current approaches and future trends," in E.Pardede (ed.), "Open and Novel Issues in XML Database Aplplications", 2009.

[7] HIPAA. Health Insurance Portability and Accountability Act of 1996, http://www.legalarchiver.org/hipaa.htm

[8] IBM, Update XML in DB2 9.5, http://www.ibm.com/-developerworks/db2/library/techarticle/dm-0710nicola/

[9] H. Jiang, "Path materialization revisited: an efficient storage model for XML data," in ADC 2002.

[10] M. Klettke, H. Meyer, and B. Hansel, "Evolution: the other side of the XML update coin," ICDE Workshops, 2005.

[11] M. Klettke, "Conceptual XML schema evolution - The CoDEX approach for design and redesign," in BTW 2007.

[12] Microsoft, XML Support in Microsoft SQL Server 2005, http://msdn.microsoft.-com/en-us/library/ms345117.aspx

[13] M. Moro, L. Lim, and Y.-C. Chang, "Challenges on modeling hybrid XML-Relational databases," in E.Pardede (ed.), "Open and Novel Issues in XML Database Aplplications", 2009.

[14] Oracle XML DB, http://www.oracle.com/technology-/tech/xml/xmldb/index.html

[15] S. Ram, and G. Shankaranarayanan, "Research issues in database schema evolution: the road not taken," 2003.

[16] J.F. Roddick, N.G. Craske, and T.J. Richards, "A taxonomy for schema versioning based on the relational and entity relationship models," in ER 1994.

[17] J.F. Roddick, "A survey of schema versioning issues for database systems," Information and Software Technology, vol. 37, pp. 383-393, 1995.

[18] A.A. Simanovsky, "Data schema evolution support in XML-relational database systems," Programming and Computer Software, vol. 34, pp. 16-26, 2008.

[19] H. Su, D. Kramer, and E. Rundensteiner, "XEM: XML evolution management," Worcester Polytechnic Inst., 2002.

[20] J. Takahashi, "Hybrid relations for database schema evolution," in COMPSAC 1990.

[21] Y. Velegrakis, "Mapping adaptation under evolving schemas," in VLDB 2003.

[22] W3C, "Extensible Markup Language (XML) 1.0 - W3C Recommendation 10-February-1998," 1998, Available from: http://www.w3.org/TR/1998/REC-xml-19980210.

[23] W3C, "XQuery 1.0: An XML Query Language," http://www.w3.org/-TR/xquery/, 2007.

[24] M. Yoshikawa, " XRel: a path-based approach to storage and retrieval of XML documents using relational databases," ACM TOIT 1, pp. 110-141, 2001.

[25] C. Zhang, "On supporting containment queries in relational database management systems," in SIGMOD 2001.