Updating XML Using Object-Relational Database

Pensri Amornsinlaphachai, M. Akhtar Ali, and Nick Rossiter

University of Northumbria at Newcastle, UK
{pensri.amornsinlaphachai,akhtar.ali,nick.rossiter}@unn.ac.uk

Abstract. Presently, the area of updating XML is immature since XQuery has not provided update features. Thus this area has not been investigated as fully as it should have been. Moreover existing researches focus on updating native XML database so that everything must be created from scratch. Furthermore, an XML document is often treated as a database by keeping all data in one document, leading invariably to data redundancy. Such redundancy in XML documents can lead to data inconsistency and low performance when updates are performed. Therefore, we exploit the power of traditional database systems, which are fully developed to update XML documents. We present a mechanism to link non-redundant data kept in multiple XML documents. The data is held in an object-relational database (ORDB) and an update language is proposed, an extension to XQuery, which is translated into SQL for updating XML data stored in an ORDB. Finally, we present a technique to propagate the changes in an ORDB to XML documents.

1 Introduction

The emergence of XML as an effective standard for representation of (semi-) structured data on the Web has motivated a host of researches in the area related to XML such as storing [6], publishing [5], querying [1], and updating [9] XML documents. In the area of querying XML documents, several query languages, such as Lorel, XQL, and XQuery have been proposed and implemented while in the area of updating XML documents, several researchers pay attention to designing update languages such as XUpdate [11], SiXDML [8], and XML Update Extension [9] of which only a few have been implemented such as XUpdate. However, these update languages can perform only simple updates. For example, they may update an XML document without checking constraints and they cannot perform joins between documents in update commands. This indicates that at present the research in this area is underdeveloped.

Our research concentrates on developing a methodology to update linked XML documents. Our motivation comes from three reasons as follows. Firstly, research in the area of updating XML is not fully developed since XQuery, a standard from W3C, has not provided update features. However, there is a suggestion from W3C [3] for the imminent arrival of an update version in XQuery. Secondly, when updates are made directly on XML documents in the form of

M. Jackson et al. (Eds.): BNCOD 2005, LNCS 3567, pp. 155–160, 2005.

native XML database, many other tasks need to be performed such as preserving constraints. However, developing the mechanism for handling this work from the current starting point may take a long time. Thirdly, an XML document is usually treated as a database keeping all data in one document; thus data redundancy can occur. This redundancy may lead to data inconsistency and poor performance when updates are performed. To reduce data redundancy, data is sometimes kept separately in several documents. However, presently, this means that joins between XML documents in update commands cannot be performed.

In our methodology, we update XML documents via ORDB and let the database engine handle the preservation of constraints; thus structure and constraints of XML are mapped to an ORDB. To solve the problem of data redundancy, data is kept in several separated documents. These documents will be linked together by a mechanism called 'rlink'. This mechanism is then mapped to an ORDB. We propose an XML update language, which is an extension to XQuery. The proposed update language is translated into SQL to update XML data stored in an ORDB. Finally, the change in an ORDB is propagated to XML documents.

For the rest of the paper, we investigate issues relating to the design of our methodology. Section 2 describes how XML documents are mapped onto an ORDB. Section 3 presents our XML update language and its translation into SQL. Section 4 describes how changes are propagated into original XML documents. Preliminary conclusions and future work are discussed in section 5.

2 Mapping XML Documents

To update XML documents via traditional databases, XML must be mapped onto a database. We map XML onto an ORDB by using a shredding approach since hierarchical structures as well as constraints of XML can be represented in an ORDB. Presently, according to published work [6, 7], full mapping of XML structures and constraints onto ORDBs cannot be fully achieved due to limited constraints-handling capabilities in existing object-relational database management systems (ORDBMSs). Therefore, we propose new mapping rules and apply some existing rules [6] that are practicable using available ORDB technologies.

We use three features of ORDBs in our mapping rules: *abstract data type*, *object table* and *nested table*. Some of our rules are as follows. Firstly, elements having only one complex child-element are mapped to object tables, and their complex child-elements are mapped to abstract data type fields. Secondly, complex elements which have `*` or `+` occurrence and have siblings are mapped to nested tables if they comply with the following conditions: (a) all of their children are simple elements and all attributes have no type `IDREF`(s), (b) they have no references to other elements and no references from other elements to them, and (c) they have no recursive structure. Thirdly, complex elements which have `?` or `1` occurrence, have a sibling and have children all of which are simple elements are mapped to abstract data type fields. Fourthly, complex elements which do not correspond to the above rules are mapped to object tables. Fifthly,

for parent-child relationship and recursive structure with `?` or `1` occurrence, the primary key of the table of parent-element is mapped to a table of child-element. Finally, for recursive structure with `+` or `*` occurrence, a separate table is created to store the primary keys of tables of a parent-element and a child-element. For attributes and simple elements, rules are similar to the work of [6].

For associating the relationship between elements from different XML documents, an *rlink* mechanism is used to provide information to identify which documents and/or elements are linked to others. Although this may be extended to XLink the main purposes of XLink and rlink are different. Mapping rlink mechanism to ORDB is the same as mapping IDREF(s). If an element referred by `IDREF` or occurrence of elements containing rlink is `1` or `?`, the primary key of the table of a referred element is mapped to a table of a referring element. If an element is referred by `IDREF`s or occurrence of elements containing rlink is `+` or `*`, a separate table is created to keep primary keys of tables of a referring element and a referred element.

Most of XML constraints can be mapped onto ORDB constraints; however, a cardinality constraint is unavailable in any (O)RDBMSs. Therefore, we add a method for preserving this constraint when updates are performed.

3 XML Update Language and Its Translation

Our XML update language is adapted from the update language proposed, but not yet implemented, by Tatarinov et al. [9], and is based on the syntax of XQuery [10]. The syntax of our language is shown in Fig. 1.

When compared with existing XML query languages, XQuery is the most powerful, providing many features [4, 10]. Moreover, since XQuery is a functional language and SQL is a declarative language, this translation cannot be performed in a straightforward manner. In our research, five important constructs of the update language are inherited from XQuery: FLW(R—I—D), conditional expression, quantifier, aggregate functions and user-defined functions. These constructs are translated into SQL using four techniques: update/delete join commands, rewriting rules, graph mapping and optimisation. At here, only the first three techniques are presented while optimisation is presented in [2].

Update/delete join commands: In the SQL standard, joins in update/delete commands cannot be performed; however, translating XML update commands can produce a join of several tables. Thus we will translate XML update commands into update/delete join commands and then rewrite these commands in SQL. Syntax of the commands is shown in Fig. 2.

Rewriting rules: There are six categories of rewriting rules: For-Let-Where-Replace-Insert-Delete (FLWRID) expression, aggregate function, quantifier, conditional expression, (non-recursive) user-defined function and SQL rewriting rules. The first five categories are classified according to features of the update language. These rules will rewrite update commands as SQL functions. Such

```
(ForClause | LetClause)+
WhereUpdateClause|IfUpdateClause
```

where each clause is:

```
ForClause         ::= For $var in XPathExp(,$var in XPathExp)*
LetClause         ::= Let $var := XPathExp(,$var := XPathExp)*
WhereUpdateClause ::= WhereClause? UpdateClause
WhereClause       ::= Where Condition
UpdateClause      ::= DeleteClause|ReplaceClause|InsertClause
DeleteClause      ::= Delete node WhereClause? (,Delete node WhereClause?)*
ReplaceClause     ::= Replace node with content WhereClause?
                      (, Replace node with content WhereClause?)*
InsertClause      ::= Insert content Into node (Before|After condition basedon XPath)?
                      (,Insert content Into node (Before|After condition basedon XPath)?)*
IfUpdateClause    ::= If Condition Then UpdateClause
                      (ElseIf Condition Then UpdateClause)*
                      (Else UpdateClause)?
```

Fig. 1. Syntax of XML Update Language

Syntax of joins in update command	Syntax of joins in delete command
`Update` *table whose fields will be updated* `From` *all related tables* `Set` *field1 =value1, field2 = value2,* `Where` *Condition;*	`Delete` *table whose data will be deleted* `From` *all related tables* `Where` *Condition;*

Fig. 2. Syntax for Update/Delete Join Commands

functions are sometimes conceptual, i.e., the function serves a purpose not currently existing in SQL. The last category is SQL rewriting rules, which rewrite update/delete join commands to SQL commands.

In translating XML update commands by using the rewriting rules, all clauses of the commands must be rewritten as SQL functions, which are used to group update clauses and their conditions together since one update command can consist of several update clauses, and each update clause can have its own conditions. These update clauses are grouped together using `funcNo`, a parameter of every SQL function. A `funcNo` of `0` for ForClause, LetClause, and WhereClause of the update command means that these clauses will be shared clauses of an UpdateClause. Each update clause will have its own `funcNo`, being a sequential number starting from 1. The update clause and its own condition(s) will have the same `funcNo`. Some of the SQL functions used are shown in Fig. 3.

Some functions have the parameter `value|:funcNo` (literal or variable) since the value in the predicate or in an insert or update command is sometimes not a

```
1. select(node, funcNo)        2. insert (node, value | :funcNo, funcNo)
3. delete(node, funcNo)        4. update(node, value | :funcNo, funcNo)
5. where | logical-operator (node, comparison-operator, value|:funcNo, funcNo)
```

Fig. 3. Examples of some SQL Functions

constant value but may come from selecting a value in other nodes. Hence in this case, `:funcNo` has the same value as the `funcNo` of `select()` function. Details of rewriting rules including additional rules for translating recursive functions into SQL can be found in [2].

Graph mapping: Graph mapping is used to determine the type of a node and hence which SQL functions can be performed on the structure of the ORDB, obtained as a result of mapping XML documents.

The process of graph mapping starts from creating a graph whose nodes correspond to nodes in SQL functions. The graph is then mapped into the database schema graph (a graph representing database schema) to identify which node is table, nested table, abstract data type field or simple field. Foreign keys for joins between tables are added to the graph. The SQL functions are then mapped to the graph. Then the graph may be split into several sub-graphs. The number of sub-graphs corresponds to the number of update operations performed on different tables. Finally, the (sub-)graphs are optimised and SQL commands or update/delete join commands are generated from the (sub-)graphs.

4 Propagating the Change in ORDB to XML Documents

The purpose of propagating the change in ORDB to XML documents is to reflect the change of data. Usually updating affects only some small parts of the documents; thus propagating the change is performed on only the affected parts. We use values of primary keys (PKs) or RowIDs of updated data in ORDB to indicate which elements should be updated. The PKs in ORDB originate from ID attributes. For elements which do not provide ID attributes, the values of RowIDs, which are automatically generated by the database system, are recorded to appropriately typed elements at the stage of populating data into the tables. Hence the values of these RowIDs can be indicated by the values of the RowIDs kept in ORDB.

When data in the ORDB is updated, the table name, PKs and values of PKs of the updated data will be returned and then the paths in the XML update command are converted to XPath expressions. The conditions in XPath expressions are based on the returned objects to indicate the positions in XML documents which will be updated. Since XPath has no capability for updating, we propose functions which serve as operators for updating XML documents.

5 Preliminary Conclusion and Future Work

As stated earlier, research in the area of updating XML is not fully developed. Thus we propose a potential way for updating XML via traditional databases. However, the mapping of XML onto simple RDB structures loses structural clarity, while object-oriented databases (OODBs) have limitations in representing constraints. Hence we map from XML to an ORDB. To eliminate redundancy, non-redundant data are kept in multiple documents and are linked by an rlink mechanism, mapping to an ORDB. We proposed an XML update language and

techniques to translate XML update language into SQL. Finally, the change in ORDB is propagated to XML documents. A major benefit of updating XML through (O)RDB is that the task of preserving constraints can be pushed to the database engine.

In further work, we will first investigate how to handle the order of elements in XML documents when elements are inserted or deleted. Then we implement the translation of the update language and propagate the change in an ORDB to XML documents. Finally, we will conduct a performance comparison of updating one XML document containing redundant data via an ORDB in the manner of native XML database with that of updating linked XML documents containing non-redundant data via an ORDB. For the future work, we will propose mapping XML to an ORDB based upon XML Schema and focus on updating the structure of XML via ORDB and handling concurrency aspects such as lock levels.

References

1. Abiteboul, S., Quass, D., McHugh, J., Widom, J., Winer, J.: The Lorel query language for semistructured data. *In* Proceedings of Int. Journal on Digital Libraries. (1997) 68–88
2. Amornsinlaphachai, P. and Rossiter, N. and Ali, A.: Translating XML update language into SQL. http://computing.unn.ac.uk/pgrs/cgpa2/. (2004)
3. Chamberlin, D.: Influences on the Design of XQuery. XQuery from experts: A Guide to the W3C XML Query Language. Addison-Wesley. (2003) 143
4. Chamberlin, D.: XQuery from experts: A guide to the W3C XML query language. Addison-Wesley. (2003)
5. Fernandez, M., Kadiyska, Y., Suciu, D., Morishima, A., Tan, W.: SilkRoute: A framework for publishing relational data in XML. ACM Transactions on Database Systems. (2002) 1–55
6. Klettke, M., Meyer, H.: Managing XML Documents in object-relational databases. Computer Science Department. University of Rostock, Germany. (1999)
7. Rahayu, J.W., Pardede, E., Taniar, D.: On using collection for aggregation and association relationships in XML object-relational storage. ACM Symposium on Applied Computing. Nicosia, Cyprus. (2004)
8. Shamkante, B., Navathe, S.: A proposal for an XML data definition and manipulation language. VLDB Conference. Hongkong.(2002)
9. Tatarinov, I., Ives, Z., Halevy, A.Y., Weld, D.S.: Updating XML. SIGMOD Conference. Santa Barbara. (2001) 413–424
10. W3C: XQuery: An XML Query Language. http://www.w3c.org/TR/xquery. (2003)
11. XMLDB: XUpdate. http://www.xmldb.org/xupdate/xupdate-wd.html (2002)