Tittle

By: Aileen Posadas, Dihia Boulegane, Minh Nguyen, Carlos Perez

Outline or agenda

- Introduction(context, problematic, motivation, approach)
- Related work.
- XML data model .
- The operations.
- Mapping between XML and relational databases.
- Implementation strategies.
- Evaluation.
- Conclusions.

Context

- As XML role has expanded beyond its original domain as a semantics preserving markup language for online documents.
- XML document repositories available and XML publishing capabilities have been added to the latest relational database systems from Oracle, IBM, and Microsoft.
- The World Wide Web Consortium is in the process of developing a standard for this XML query language(XQuery)
- The database research community has struggles in addressing the challenges of providing XML views of relational databases.

Problem

- XML is not a full-featured data exchange format.
- The users are restricted to update XML documents.
- In order to fully evolve XML into a universal data representation and sharing format:
  - Users must be able to updates capabilities to XML documents (for modifying, propagating changes through XML views)
  - There must be develop techniques to process XML documents.

Motivation

- XML nowadays it is one of the default format for interchanging data between heterogeneous systems.
- Great interest in languages and systems for expressing queries over XML data due the straightforward access to data.
- Relational database engines will provide standardized, integrated support for querying and publishing XML views of databases.
- Making XML into a full-featured data exchange format is to support not only queries, but **updates** over XML content.

Approach

1. Propose a set of constructs for expressing updates (ordered and unordered) XML data model.
2. Map these constructs into the syntax of the XQuery language.
3. Describe implementation techniques over a relational database system.

Also is consider:

- Describe extensions to the proposed standard XML query language, XQuery, to incorporate the update operations.
- Consider alternative methods for implementing update operations when the XML data is mapped into a relational database.
- Review key concepts in mapping between XML and relational databases.

Contribution

- Propose a set of primitive operations for modifying the structure and content of an XML document.
- Present update extensions to the World Wide Web Consortium's proposed XQuery standard query language.
- Provide algorithms for implementing XML update capabilities within an XML repository based on a relational database (RDB) system, and describe how them can also be applied to the problem of updating RDB through XML views.
- Study techniques for updating complex XML structure mapped to a relational system.
- Provide an analysis of the performance of our different update strategies using a number of workloads and document structures.

Related work

- People focus on developing query language like XQuery *[2]* rather than ***updating*** XML data
- eXcelon XML *[17]*: repository which supports XML updates, plus simple ***insertions*** and ***deletions***.
- Lorel *[1]* supports ***insertion*** and ***deletion*** into Lore data graph, then migrated to XML data *[12]*, without ***update*** functionality.
- Object-oriented systems support *object assignment*, ***insertion*** and ***deletion*** from a collection, but not ***update***.
- Problems of storing XML data into relational database system [10,7,4] and extracting an XML view from relational database system [9,15,3] has been previously mentioned.

XML Update fundamentals

Data model

For this paper a simplified version of the World Wide Web's Query Data mode is used. Xquery uses this data model l which views an XML document as a node-labeled tree with references.

There is also the need to refer to different kinds of XML contents: The contents refered to in the paper are objects, attributes, IDREFS (named order list of IDs), elements and PCDATA.

XML Update fundamentals

Update operations

- delete(child): If the child is a member of a target object is removed
- rename(child,name): If the child is a non.PCDATA member of the target object iti is given a new name.
- insert(content): Inserts new content into target. (Can be PCDATA, element, attribute or reference)
- insertbefore(ref,content): Only for ordered execution. If ref is a child element then content must be an element or PCDATA.
- replace(child, content): Atomic replace operation
- sub-Update (patternMarch, predicates, updateOp): Invokes a nes pattern-matching operation over the input.

XQuery extensions for Update

Mapping of operations in XQuery language syntax

- Deletion: Removes a node from a child.
- Insertion: Introduce a constructor for new attributes and references.
- Replacement: It has the same effect as inserting an item before another one and deleting it. It is just sometimes convenient to have it a single atomic operation.

Storing XML in relations

XML repositories are mostly constructed over relational database systems. Another symilar source of queryable XML data are XML mediators placesd over existing relational databases.

The mediator generates a hierachical XML view of existing databases.

Several methods have been proposed to map XML to SQL but in general they produce create excessive fragmentation .

- The edge approach works with documents that don’t have a DTD. The shared inlining method exploit a DTD to better cluster parents and child methods.
- The shared Inlining method exploits a DTD to better cluster parent and child elements. The DTD provides the information when the inlining is possible.

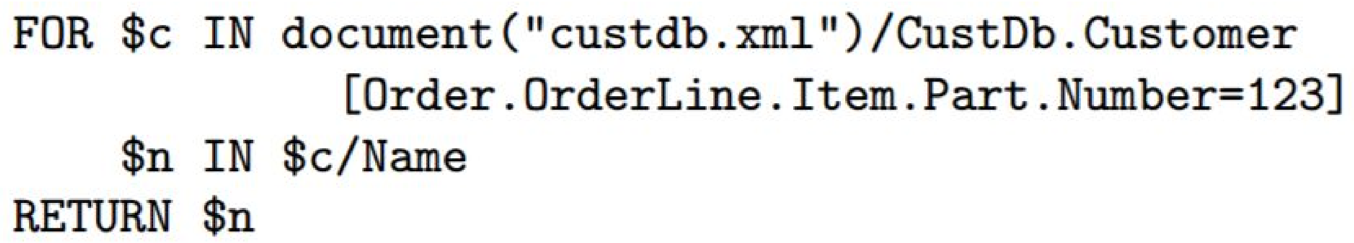
Storing XML in relations

XML Results as Outer Unions

- When an XML structure is stored across multiple tables there are a number of possible ways of returning the results. There are different techniques to do this.
- The one this paper uses is called Outer Union : To simplify the job of reconstructing the XML document at the cient, output tuples are sorted so that child element data comes after parent data and child elements. Instead of separately joining the relations Customer, Order and OrderLine

Access Support Relations for Path Expression Evaluation

- Access Support Relation is an effective method for speeding up the evaluation of path expression in object-oriented and semi-structured database.



```
FOR $c IN document("custdb.xml")/CustDb.Customer
              [Order.OrderLine.Item.Part.Number=123]
    $n IN $c/Name
RETURN $n
```

- With this sample query, normally we need 3 joins.
- If we have ASR path at root, it only takes 2 joins to finish.

UPDATING STORED XML

- Ideally, an update in XML should be translated into a single SQL query for better performance in large-scale optimizations.
- However, it requires multiple SQL queries to update an XML document at several levels of hierarchy.
- Other problems: replication of tuples to reserve connectivity of keys, correctness assurance of translation…
-

Information about the solution

Minh Nguyen

Technical details about the solution

Information about the solution

Minh Nguyen

Technical details about the solution

Experimental evaluation

- **Description of the environment**
  - The tested were run on a single computer to avoid network
  - No transaction were commited to avoid reloading the data

Experimental evaluation

- **Dataset :**
  - Two different ways : **bulk** and **random**
  - **3** different datasets:
    - **Fixed synthetic data**
    - **Randomized synthétic data**
    - **Real life data**

Experimental evaluation

- **Dataset :**
  - Two different ways : **bulk** and **random**
  - **3** different datasets:
    - **Fixed synthetic data**
    - **Randomized synthétic data**
    - **Real life data**

Experimental evaluation

- **Effect of Access Support Relations**
  - This parameter affects updates and all
  - Improves performances in case Fanout is small
  - Helpful in case of long paths expressions

Experimental evaluation

- **Evaluation of Delete**
  - Bulk is often better than Random because of the number of SQL statements enerated
  - **Bulk:** per-statement outperforms
  - **Random :** per-tuple doesn’t increase with the increase on data size, per-sttement slows

Experimental evaluation

- **Evaluation of insert**
  - Copying element from-to the same file
  - Random : 10 random subtrees
    - If data is smal $\longrightarrow$ tuple-method outperforms
    - If data is large $\longrightarrow$ table method is the best
  - Bulk : copy all the document $\longrightarrow$ Table method is the best

Experimental evaluation

- **Evaluation of update**
  - And update can be expressed as an insert and delete primitie
  - The best performance is the combination of fastest insert - fastest delete
    - **Insert** : table-based
    - **Delete** : per-tuple

Conclusion

- This work allows to express and execute changes on XML data as extension to XQuery
- **Futur work :**
  - Typechaking during opérations
  - Deterministic result for reachable references