Summary

Context:

Thanks to the continuous development that XML have know, its role has become wider, as it is now a standard for interchanging data between heterogeneous systems. Therefore, more attention has been gained by languages and systems expressing queries over XML data, regardless of the source.

Introduction:

Update capabilities are very essential in the sense that they afford a greater control over the data. Especially, being able to modify XML documents, propagate changes through XML views or even expressing and transmitting changes to documents.

The suggested language extensions allows different update constructs, namely: Insertions, deletions and assignments at multiple levels within a hierarchy. In addition, these methods are mapped into the syntax of the XQuery language, a WWW Consortium standard. Updates over ordered and unordered XML data model are both supported.

Motivation:

- Making XML into a full-featured data exchange format
- Support not only queries, but also updating over XML content including derived XML Schema
- Allowing users to specify updates to XML documents
- Developing more efficient techniques in processing XML documents.

Contribution:

The research paper has several contributions:

- Proposing basic operations to modify an XML document structure and content.
- Presenting update extensions to XQuery.
- Providing algorithms for implementing XML update capabilities within an XML repository based on a relational database system.
- Provide an analysis of the performance of our different update strategies.

Related works:

While many aspects related to XML has been studied extensively, the topic of updating XML hasn't received much attention. (Extensions to Lorel, eXcelon XML repository, Object oriented systems)

Problem

What can be done and improved to easily manipulate a XML document ?
How can we map those updates to the XQuery language ?

Methodology/Solutions

1. **Define primitive opérations**

We need to define primitive operations for adding it to the XQUERY language. So we define the following primitive operations :

- delete
- rename
- insert
- ...

2. Define syntax as a xquery extensions

We define the syntax of the updates queries inspired by the Xquery syntax, the FOR … LET … WHERE … UPDATE structure for a general pattern. Then we define the pattern of each primitives operations.

3. Storing XML in relations

- Focus on the inlining approach to transform the XML document into a queryable XML data source
- Display XML results as Outer Unions using “wide” tuples

4. Updating stored XML

We detail :

1. **XML Deletion techniques :**
    - Trigger-Based Delete
    - Per-Tuple vs. Per-Statement Triggers
    - Cascading Delete
    - ASR-Based Delete
2. **XML Insertion techniques :**
    - Tuple-Based Insert
    - Table-Based Insert
    - ASR-Based Insert

Experimental results

We summarise in the upcoming section the battery of tests realised in order to determine the best performing approaches to implementing the proposed set of primitive XML update operations that can be composed to express modifications at multiple levels within an XML document. This tests were geared to assess performances on the most widely researched means of storing data presented as XML : the relational database.

The experiment variables dealt with two kinds of workloads that processed several modes of data syntheticity as in fixed synthetic data, randomized synthetic data and real life data. Synthetic with three tuning parameters namely root subtree number, subtree depth and internal child complexity of the subtree nodes - summed up as scaling, depth and fanout respectively - were studied unvaryingly considering the huge number of possible configurations. The other synthetic data type was generated with randomised scaling, depth and fanout factors when the real life data was the XML organized publications portion of the DBLP bibliography. The workload modes listed :

- bulk workloads : where update operations were applied to every subtree element of the synthetic input document.
- random workloads : only a 10 randomly chosen subtrees were submitted to update operations.

The study group of the experiment were the update operations expressed as a combination of ***DELETE*** and ***INSERT*** statements sequence. Overall the fastest ***INSERT*** method was table-based ***INSERT*** when the fastest ***DELETE*** method was per-tuple trigger-based ***DELETE.*** All the benchmarked candidates are the following :

- per-tuple trigger, per-statement trigger, ASR, and cascading deletes.
- the tuple, table, and ASR based inserts.

This winners performed significantly better through the two synthetic data and also on the DBLP bibliography data.

The study frame was a java-written SQL queries issuer to an IBM B2 UDB 7.1 via JDBC on commodity hardware computer -2016 adjusted.

Conclusions and future work

An XML update language provides a general-purpose way to express changes to any data that can be represented as XML — whether the data is actually stored within an XML repository or a relational database with an XML view. The article have proposed a set of primitive XML update operations that can be composed to express modifications at multiple levels within an XML document and have incorporated these operations as a set of language extensions to XQuery. This proposal was put to test on the commonly researched topic of xml organised relational databases. Paths to more focused attention are proposed including investigating query techniques for typechecking and path matching of update targets.