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.