Efficient Structural Bulk Updates on the Pre/Dist/Size XML Encoding

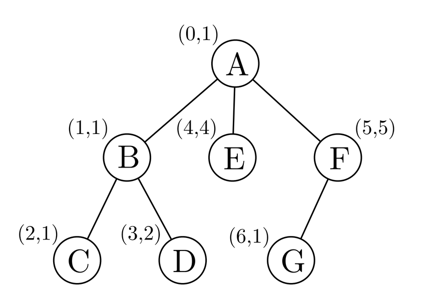
Lukas Kircher, Michael Grossniklaus, Christian Grün, and Marc H. Scholl
Department of Computer and Information Science
Database & Information Systems (DBIS) Group
University of Konstanz
P.O. Box 188, 78457 Konstanz, Germany
firstname.lastname@uni.kn

Motivation

XQuery is a functional programming language. It is developed with a strong focus on querying and manipulating XML data. XQuery 1.0 became a W3C recommendation in 2007. The XQuery Update Facility (XQUF) extension followed in 2011 and is widely adopted today.

BaseX is an open-source, native XML database and XPath/XQuery 3.1 processor. Its internal XML representation is the interval-based *Pre/Dist/Size* encoding schema. Very compact records of fixed length support efficient evaluation of all XPath axes.

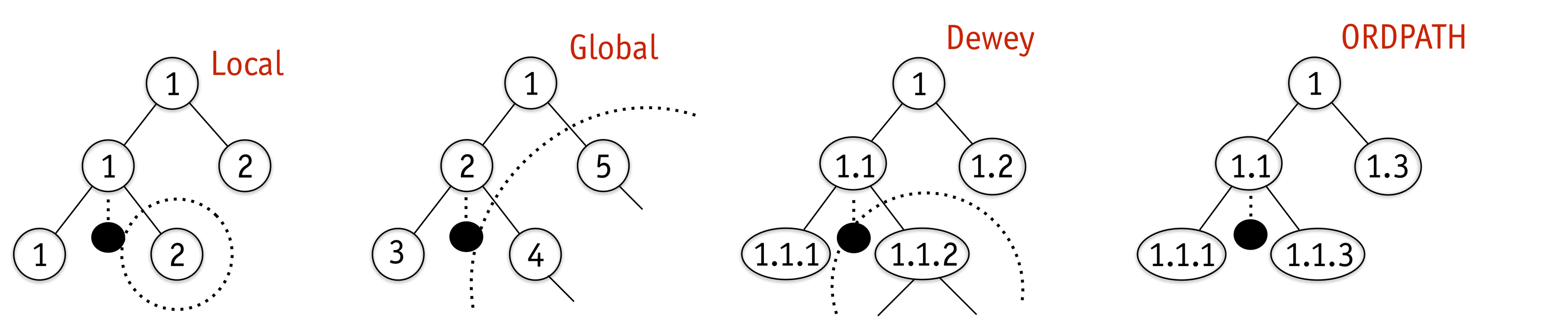
```
<A>
  <B>
    <C/><D/>
  </B>
  <E/>
  <F>
    <G/>
  </F>
</A>
```



pre	dist	size	val
0	1	7	A
1	1	3	B
2	1	1	C
3	2	1	D
4	4	1	E
5	5	2	F
6	1	1	G

- **Pre:** node position during pre-order traversal
- **Dist:** # nodes between self and parent
- **Size:** # nodes in subtree + 1

In contrast, prefix-based encodings have a high updatability. Dynamic labels require little (no) re-labeling after inserts or deletes. Yet, accessing nodes on disk requires a layer of indirection as records vary in length.



Each encoding has its advantages and disadvantages. But have we really reached the limits with *Pre/Dist/Size* with regards to updatability?

Pending Update List (PUL)

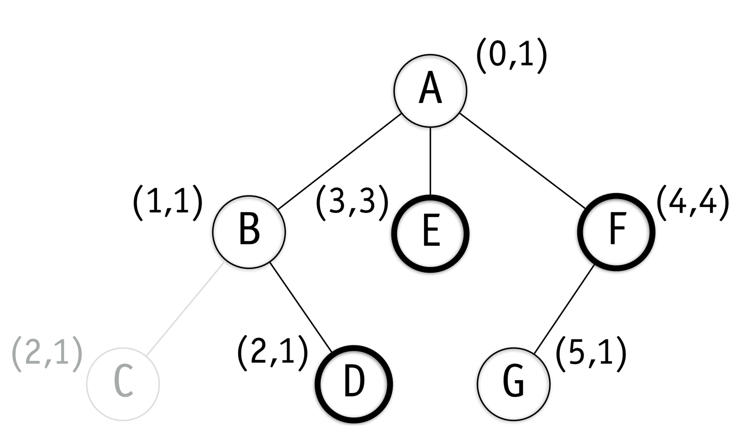
- 'deep' cache of XQUF atomic update primitives
- memory-intensive
- changes applied as last step of a transaction
- *"update effects only accessible after the transaction"*
- *stages:* 1. collect primitives 2. prepare application 3. apply updates

query snapshot

```
for $person in //person return
  insert node
    <id_confirmed>no</id_confirmed>
  into $person
```

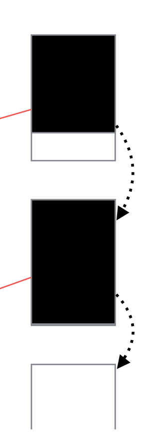
Pre/Dist/Size Updatability

While value updates are cheap, structural updates require re-labeling a potentially large amount of records. Implicitly stored *Pre* values of successor records are shifted efficiently. Only *Size* values of ancestor nodes are affected. However, a *Dist* value is updated if a node is inserted or deleted between an existing node and its parent. The costs depend on the update location and document structure.



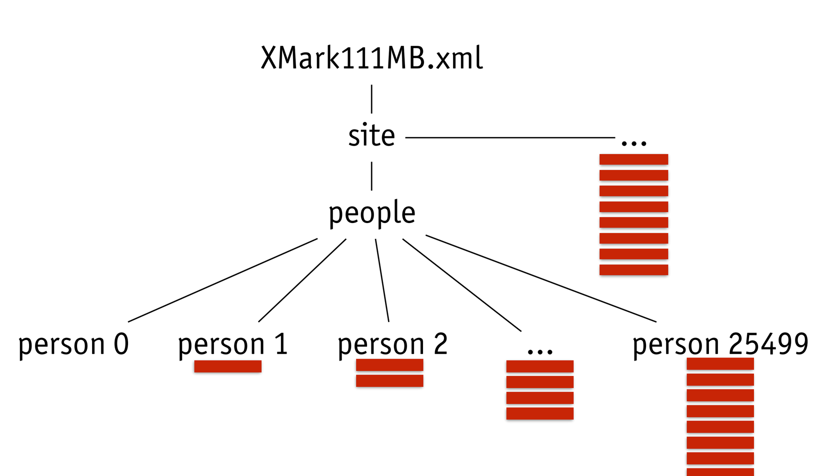
DEL

Pre	Dist	Size	Value
0	1	6	A
1	1	2	B
2	1	1	C
2	1	1	D
3	3	1	E
4	4	2	F
5	1	1	G



Bulk updates are especially expensive. Each structural update primitive affects a set of *Dist* values. These sets are heavily intersected.

```
for $person in //person return
  insert node
    <id_confirmed>no</id_confirmed>
  into $person
```



Possible Solution

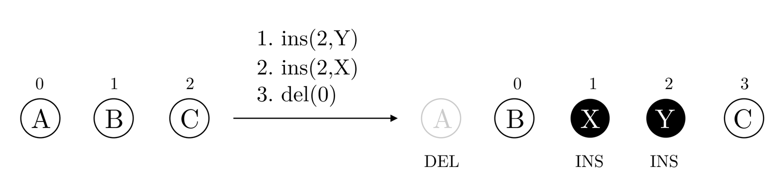
- leverage Pending Update List to pre-compute final distance values
- adjust each *Dist* value only once

Efficient Structural Bulk Updates

The *Atomic Update Cache* is an efficient data structure that complements the PUL.

AUC

- collects all update information of a snapshot in *document order*
- points to affected table records and *Dist* values
- maps *Pre* values before/after updates
- *is applied in reverse document order*
- —> enables us to delay and pre-calculate *Dist* updates

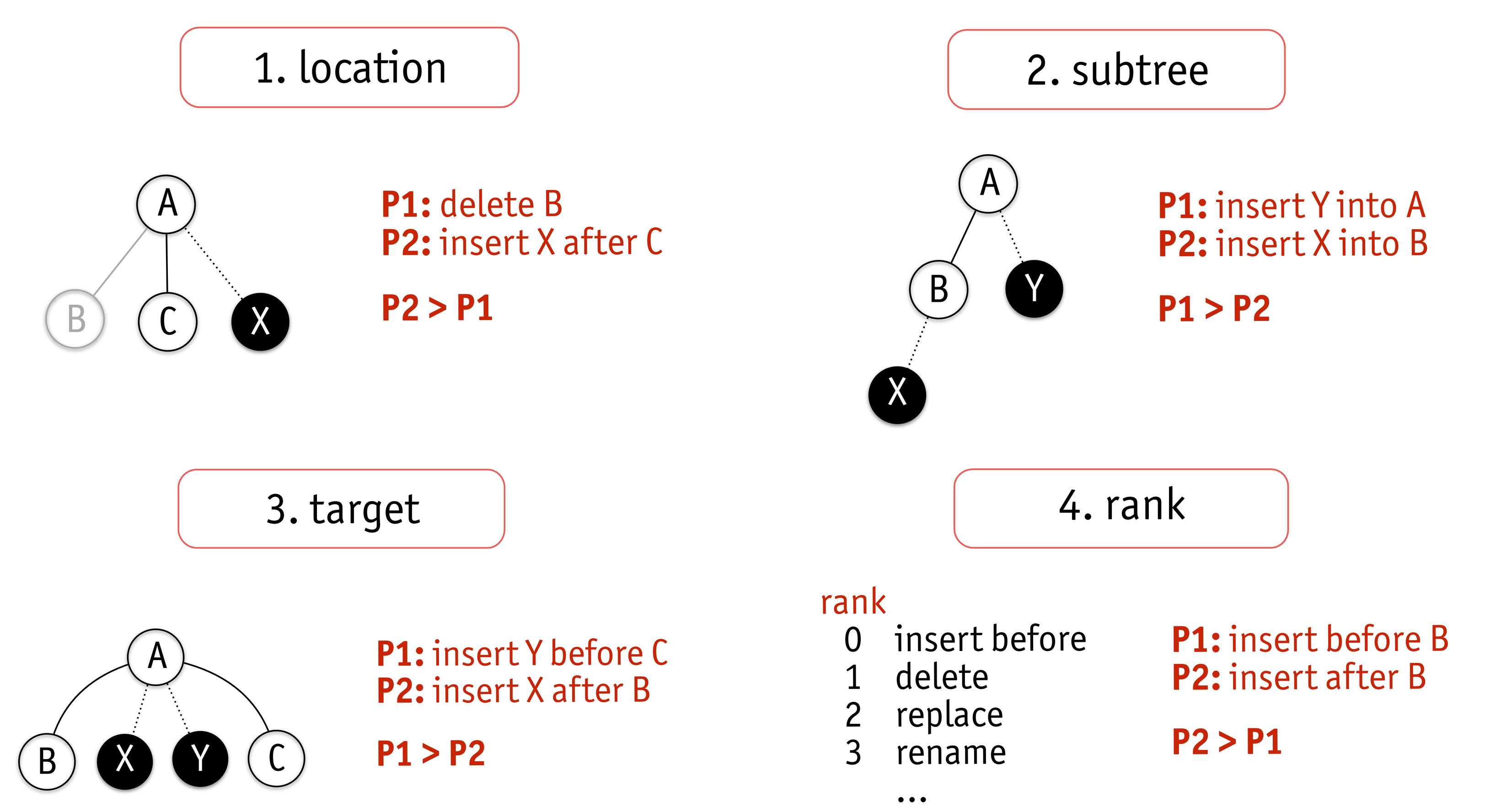


atomic	first affected tuple	shifts	accum. shifts
del(0)	1 (0)	−1	−1
ins(2,X)	2 (2)	1	0
ins(2,Y)	2 (3)	1	1

Bulk Update Stages

fill → ! constraints → apply updates → adjust *Dist* → text adjacency

Filling the AUC is non-trivial. How to map XQUF primitives to atomic updates without violating the document order? Each update primitive operates on a target node and affects a specific table location. Two primitives can be ordered by a four-step comparison:

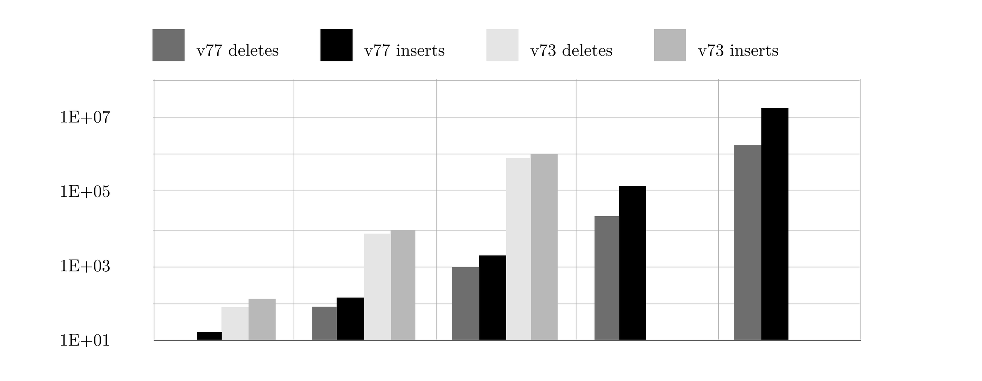


Results

2010 Apple iMac, Intel Core i3 3.2 GHz, 8 GB RAM,1 TB hard-disk drive, BaseX 7.3, 7.7 (AUC), OS X 10.8.4, Oracle Java 7.12, -Xmx6G, XMark data

Q1 *delete node //date*

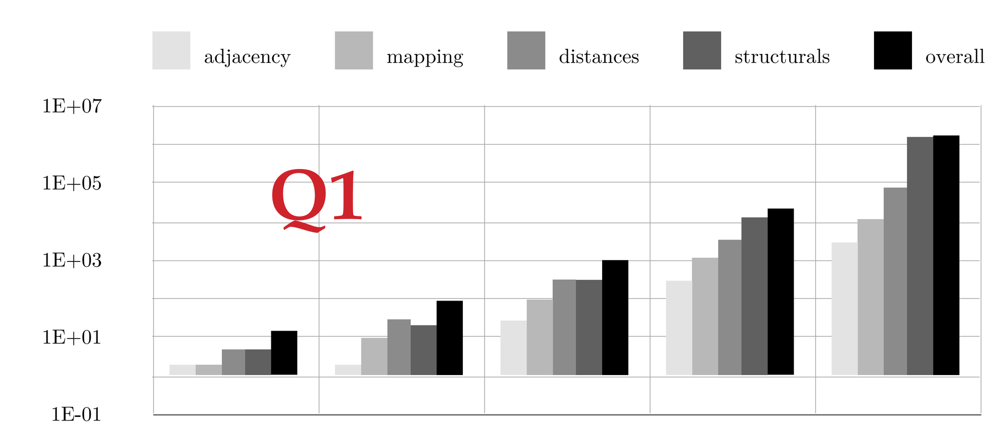
Q2 *for $d in //date return insert node <ndate>99</ndate> after $d*

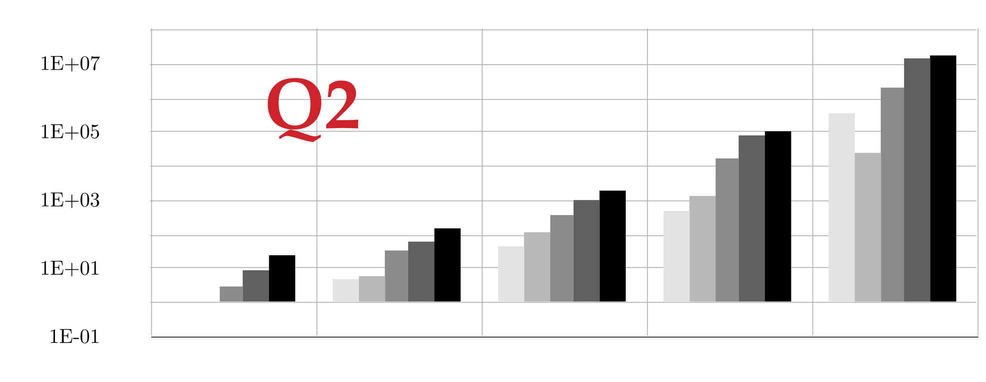


Observations

- AUC improves performance by magnitude
- much faster distance adjustment
- still quadratic complexity for bigger documents
- insertion/deletion of records is the new bottleneck (structurals)

Keeping records in document order is the new limiter





Conclusion

- Efficient Structural Bulk Updates exploit the XQUF conditions
- little overhead is added in the form of the AUC
- the tree structure is efficiently restored after updates
- *Pre/Dist/Size* reduces advantage of prefix labelling schemas (bulk updates)
- *Document order* is the new limiter

Universität Konstanz

BASE X