documents into tuples and storing them in a relational DBMS

- Benefit from the existing database technology by "shredding" XML

- Idea:

  - Management, Indexing, Queries, Query Optimization...

  - Databases are well-equipped to process relational data (storage

  - Growing amounts of data are readily available in the XML format

  - XML model is more convenient for some application domains (e.g.

  - XML structure is less rigid than the traditional relational format (semi-

  - XML vs. Relational Representation:

Motivation

September 2004
Dept. of Computer & Information Science
University of Konstanz

Svetlana Vinkovic

Managing XML Data

on Relational Databases
Basic Approaches

Two basic approaches have emerged:
- XPath Accelerator (Torsten Grust, 2002; Grust et al., 2004): a purely relational storage structure developed with a close eye on XPath semantics. The underlying mapping scheme is schema-oblivious and supports all XPath axes, rooted from any context node.
- Schema-based techniques usually store any tree without a close eye on the document's structure. The document's schema may be less flexible.

XPath

- #+**+
- $\times$ in fn:doc("bib.xml")/bib/books/book[author = "John Doe"]
- where @price >= 42
- return <expensive-book> { $x/title/text() } </expensive-book>

Motivation

- What are the obstacles?
  - The data model behind XML is the tree.
  - XML trees are navigated through following a path.
- Existing technology cannot directly be applied to XML data.
- How can we represent trees as relations?
- How can we benefit from index structures?
- How can we implement tree navigation?
- The W3C XQuery proposal poses additional challenges:
  - existing technology cannot directly be applied to XML data.
  - XML trees are navigated through following a path.
Xpath Expressions

Xpath Terminology

from left to right
(2) a sequence of location steps syntactically separated by /, evaluated
(1) a context node, which is the starting point of the traversal, and

Xpath expressions consist of

• from a to a leaf in the sub-tree rooted at a,

heuristic (h)
the length of the longest path

node o is the root of the tree.

element or attribute nodes.

leaf nodes of o, c, g, and if represent either

inner nodes o, d, c, g, and if represent

Xpath expressions operate on trees of

Xpath Expressions

document order.

N.B. Result of path expression is duplicate-free node sequence in

execution of a single step s on node sequence

expression (or path) is XPath's core construct.
XPath axes

Semantics of axes supported by XPath: (step 8/9):

- This set of nodes is based on the location of context node c in the document tree, a sequence of reachable nodes.
- Each step is built from two components (separated by '::').
- Given the context node c and step, compute the sequence of nodes reached by this step.
- The processor (c, s) primitive function lies at the heart of the XPath.

Typical XPath queries thus look like:

```
comment / a::a / a/a
```

- a::a is a path expression.
- The '::' just marks the start of a step.
- Given nodes (e.g., any specifically named element nodes or text nodes), only text nodes from this sequence is examined.
For each node \( \nu \), we receive:
- the startElement event before the events of any children of \( \nu \), and
- the endElement event after the events of any children of \( \nu \).

We say that:
- the startElement event occurs in pre-order,
- endElement events occur in post-order.

For each tree node \( \nu \), the two values to be stored in a relational table are:
- pre-order rank \( \text{pre}(\nu) \),
- post-order rank \( \text{post}(\nu) \).

**Idea:**
The sequence of SAX events contains all necessary information for representing a tree.

- Numbering according to this order is sufficient to answer all queries.
- The SAX events arrive during a sequential read of the XML document.
- This corresponds to a depth-first, left-to-right traversal of the XML tree.
XPath Steps

Range predicates:
- Recursion turns into simple
- \( \{ a | \pre(a) > \{ a | \pos(a) \} \lor \pre(a) < \{ a | \pos(a) \} \} \)
- In the \textit{pre/post} encoding:
  - we saw the end tag of \( a \). \textit{Before} the end tag of \( a \).
  - we saw the start tag of \( a \). \textit{Before} the start tag of \( a \).
  - Node \( a \) is a descendant of \( a \). \textit{If} The \textit{descendant axis}:
- The \textit{preceding axis}:
  - Node \( a \) is a predecessor of \( a \).
pre/post plane

- Regard pre and post values of nodes as points in the two-dimensional pre/post plane.

- XPath axes can be represented as windows in the plane:

- XPath Steps

The following axes:

- Contrary to preceding relationship:
  \( \text{pre}(v) \leq \text{post}(v) \)

- Contrary to descendant relationship:
  \( \text{pre}(v) < \text{pre}(w) \land \text{post}(v) > \text{post}(w) \)

- \( v \in \text{ancestor} \)

- \( v \in \text{following} \)
Evaluating XPath Location Steps

Retrieved
- Preorder Packets preserves document order on
  minimized
- Coverage and overlapping of leaves
  maximized
- Storage utilization in R-Tree Led Pages

Preorder Packets R-Trees
- and name least in parallel.
- 5-dimensional R-Tree evaluates XPath axis,
  - adopt well to node distribution.
  - R-Trees: incomplete space partitioning.
  - are points in a 5-dimensional space
  - Node evaluations (pre, post, par, off, tag)

XPath Evaluation Scheme

Empowering R-Trees

WHERE @ INSIDE window
WHERE @ query (d, a) acceld
FROM WHERE @ query
SELECT

XPath query on the pre/post plane.
- To evaluate an XPath expression, just repeatedly evaluate
- Other node properties can be mapped to additional dimensions.
- A combination of two R-Trees can be used for two dimensions.
- R-Trees as an n-dimensional index structure;
- Database technology can efficiently support n-dimensional data;
- Formally recursive queries turn into region queries on that plane;
- Tree-structured document is mapped into a 2-dimensional space.

XPath Evaluation Scheme
Further Optimizations

- Similar ideas apply to other axes.

Employing B+ Trees

- Notice that the pre value of the last descendant node \( v \) is \( \text{pre}(v) = \text{pre}(o) + \text{size}(o) \).

- The post value of the last descendant node \( v \) is \( \text{post}(v) \leq \text{post}(o) \land \text{pre}(o) + \text{level}(o) \).

- \( v \in \text{descendant} \)

- \( \text{level}(o) \) vs. \( \text{level}(v) \)

- \( \text{size}(o) \) vs. \( \text{size}(v) \)

- \( \text{post}(o) = \text{post}(v) = \text{post}(\text{ancestor}) + \text{size}(\text{descendant}) \)

- \( \text{pre}(o) = \text{pre}(v) = \text{pre}(\text{ancestor}) + \text{size}(\text{descendant}) \)

- Notice that the scan and post columns with a B+-index can do better.

- Index pre and post columns with a B+-index.

- A window query takes two index scans.

- Lots of false hits in either scan.

- Can we do better?
Further Optimizations

We thus extend our database table by a property Level:

- We may now even use either range, pre or post, to evaluate the descendant axis.

Further Optimizations
Research Group, University of Konstanz, December 2001.


ACM SIGMOD Conference on Management of Data, Pages 109-120.

Tosten Gust. Accelerating XPath location steps. In Proc. of the 21st Int'l

TDS'02, 20(1), March 2004.

Tosten Gust; Maurice von Reuten, Jason Teubner. Accelerating XPath

location steps in any RDBMS. ACM Transactions on Database Systems.

Tosten Gust, Maurice von Reuten, Jason Teubner. Accelerating XPath

location steps in any RDBMS. ACM Transactions on Database Systems.

Tosten Gust; Maurice von Reuten, Jason Teubner. Accelerating XPath

location steps in any RDBMS. ACM Transactions on Database Systems.


---

**Literature**

---

**Performance**