Abstract

The XPath accelerator ("pre/post numbering") has proven to be an efficient encoding to losslessly store XML data in relational databases. Conventional RDBMSs, however, remain ignorant of interesting properties of the encoded tree data, and thus no or poor use of these properties. At VLDB 2003 we devised a new join algorithm, staircase join, that encapsulates tree-specific knowledge and can turn SQL systems into highly efficient XPath processors.

This demonstration delivers the promise we have made at VLDB 2003: a notion of tree-awareness can be injected into a conventional disk-based RDBMS kernel in terms of staircase join. We feature a side-by-side comparison of both, an original and a staircase join-enhanced instance of PostgreSQL. The required changes to PostgreSQL were local. The achieved effect, however, is significant: the demonstration proves that this injection of tree awareness makes PostgreSQL a high-performance XPath processor that closely adheres to the XPath semantics.

Staircase Join: Pruning, Partitioning, and Skipping

- Behaves like any join.
- Allows for rewriting.
- Chosen during dynamic programming.

Execution Plan of an XPath Query.

Staircase Join in a Relational Database

- Works as a state automaton.
- Relies on existing index structures (B-tree).
- Allows for streaming/pipelining.

The descendant axis state automaton.

Performance Tests in PostgreSQL

Execution times and index-related buffer statistics in the tree-aware and the original PostgreSQL DBMS:

- Query: /descendant::t1/descendant::t2
- Query: /descendant::t1/ancestor::t2
- Query: /descendant::t1/preceding::t1
- Query: /descendant::t1/following::t1