Why Off-The-Shelf RDBMSs are Better at XPath Than You Might Expect

Jens Teubner · Torsten Grust · Jan Rittinger

http://www.pathfinder-xquery.org/
We do **not** want to clutter the RDBMS kernel with XPath specifics, e.g.,

- Multi-Predicate Merge Joins (MPMGJN),
- Holistic join algorithms (PathStack, TwigStack, etc.),
- Structural joins (Tree-Merge, Stack-Tree, staircase join, etc.).

**Instead:** Use *existing* functionality in *off-the-shelf* RDBMSs:

- (partitioned) **B-trees**,
- **aggregates.**
Recap: Tree Encodings

\[
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c} \\
\text{d} \\
\text{e} \\
\text{f} \\
\text{g} \\
\text{h} \\
\text{i} \\
\text{j}
\end{array}
\]

\[
\begin{array}{c|c}
\text{n} & \text{pre} \\
\hline
\text{a} & 0 \\
\text{b} & 1 \\
\text{c} & 2 \\
\text{d} & 3 \\
\text{e} & 4 \\
\text{f} & 5 \\
\text{g} & 6 \\
\end{array}
\]
Recap: Tree Encodings

Why Off-The-Shelf RDBMSs are Better at XPath Than You Might Expect

Jens Teubner, TUM
Recap: Tree Encodings

Why Off-The-Shelf RDBMSs are Better at XPath Than You Might Expect

Jens Teubner, TUM 3 / 12
Recap: Tree Encodings

<table>
<thead>
<tr>
<th>n</th>
<th>pre</th>
<th>size</th>
<th>level</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>9</td>
<td>1</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>c</td>
<td>2</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>e</td>
<td>4</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>f</td>
<td>5</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>g</td>
<td>6</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Why Off-The-Shelf RDBMSs are Better at XPath Than You Might Expect

Jens Teubner, TUM 3 / 12
Recap: Tree Encodings

Why Off-The-Shelf RDBMSs are Better at XPath Than You Might Expect
XPath on Encoded Tree Data

The XPath **descendant** axis turns into a **range predicate** on \( pre \).

✓ Efficiently supported by a **B-tree** index on column \( pre \).
The XPath **descendant** axis turns into a **range predicate** on `pre`.

✓ Efficiently supported by a **B-tree** index on column `pre`.

The **child** axis needs some more thought (e.g., `ctx/child::node()`):

```sql
SELECT DISTINCT d.*
FROM ctx c, document d
WHERE c.pre < d.pre AND d.pre <= c.pre + c.size
  AND d.level = c.level + 1
ORDER BY d.pre
```

---

Contrast to edge mapping (explicit parent/child edges):

```sql
SELECT DISTINCT d.*
FROM ctx c, document d
WHERE c.pre = d.parent
ORDER BY d.pre
```

**Why Off-The-Shelf RDBMSs are Better at XPath Than You Might Expect**

Jens Teubner, TUM
XPath on Encoded Tree Data

The XPath **descendant** axis turns into a **range predicate** on *pre*.

✔ Efficiently supported by a **B-tree** index on column *pre*.

The **child** axis needs some more thought (e.g., `ctx/child::node()`):

```sql
SELECT DISTINCT d.*
    FROM ctx c, document d
    WHERE c.pre < d.pre AND d.pre ≤ c.pre + c.size
        AND d.level = c.level + 1
ORDER BY d.pre
```

**no simple range**

(*level condition*)

→ **false hits**

Contrast to **edge mapping** (explicit parent/child edges):

```sql
SELECT DISTINCT d.*
    FROM ctx c, document d
    WHERE c.pre = d.parent
ORDER BY d.pre
```

**foreign key join**
XPath on Encoded Tree Data — Experiment

XMark: \(/\text{descendant::open\_auction/bidder/increase}\)

 execution time [ms]

11 MB

111 MB

1.1 GB

Why Off-The-Shelf RDBMSs are Better at XPath Than You Might Expect

Jens Teubner, TUM
XMark: /descendant::open_auction/bidder/increase

Why Off-The-Shelf RDBMSs are Better at XPath Than You Might Expect

Jens Teubner, TUM
To evaluate child, DB2 used a \(<level, pre>\) B-tree.

- \(level\) has a **low selectivity**.
- This effectively **partitions** the B-tree into \(h\) partitions (\(h\): height of the XML tree).
To evaluate child, DB2 used a $\langle \text{level}, \text{pre} \rangle$ B-tree.

- Level has a low selectivity.
- This effectively partitions the B-tree into $h$ partitions ($h$: height of the XML tree).

Why Off-The-Shelf RDBMSs are Better at XPath Than You Might Expect
To evaluate child, DB2 used a \( \langle \text{level}, \text{pre} \rangle \) B-tree.

- \textit{level} has a \textbf{low selectivity}.
- This effectively \textbf{partitions} the B-tree into \( h \) partitions (\( h \): height of the XML tree).

Scan for \( d \)'s children, start at \( \langle \text{level}(d) + 1, \text{pre}(d) \rangle \)

\[ \rightarrow \textbf{no false hits!} \]
More Partitioned B-Trees

Use partitioned B-trees depending on your query workload:

- XPath **name tests**:  
  → partitioned \( \langle \text{tag}, \text{pre} \rangle \) or \( \langle \text{tag}, \text{level}, \text{pre} \rangle \) index.
- XPath **kind tests** (e.g., text(), element(), *):
  → partitioned \( \langle \text{kind}, \text{pre} \rangle \) or \( \langle \text{kind}, \text{level}, \text{pre} \rangle \) index.
  → **Predicate pushdown** into the index.

Partitioned B-trees can implement **schema-awareness**:

- Record root-to-leaf path for each node in column *path*  
  (~PATH_ID field in SQL Server’s primary XML index).
  → partitioned \( \langle \text{path}, \text{pre} \rangle \) index.
**Staircase join:** prune context nodes that won’t contribute to the result.

- E.g., \((c, d)/\text{following::node()}\)
- Removing \(d\) from the context set does not affect query outcome (XPath: duplicate-free result).

→ **Prune** context set first.
**Staircase join: prune** context nodes that won’t contribute to the result.

- E.g., \((c, d)/\text{following::node()}\)
- Removing \(d\) from the context set does not affect query outcome (XPath: duplicate-free result).

→ **Prune** context set first.

In the \textit{pre/level} plane:
Context Pruning in SQL

Reduce context to node $v$ with $\text{minimum } \text{pre}(v) + \text{size}(v)$.

$\rightarrow$ Pruning turns into $\text{aggregation}$ on the relational back-end.

In SQL:

```
SELECT DISTINCT d.*
FROM ctx c, document d
WHERE d.pre > c.pre + c.size
ORDER BY d.pre
```

```
SELECT d.*
FROM document d
WHERE d.pre > (SELECT MIN (c.pre + c.size)
               FROM ctx c)
ORDER BY d.pre
```

Note that this can be done by $\text{purely algebraic rewrites}$.

$\rightarrow$ No XML/tree knowledge involved.

$\rightarrow$ Also $\text{non-XPath}$ queries may benefit from such rewrites.
Context Pruning on IBM DB2

Path: /descendant::city/following::zipcode

Without pruning: $8.1 \times 10^9$ duplicates to sort on 1.1 GB XML instance!
Relational vs. Native XML

Relational vs. DB2’s built-in native (pureXML®) XML storage.

Why Off-The-Shelf RDBMSs are Better at XPath Than You Might Expect

Jens Teubner, TUM
Conclusions

- **Off-the-shelf** RDBMSs provide everything we need for efficient XML processing:
  - **Partitioned B-trees** support non-recursive axes and others.
  - **Aggregation** implements the **pruning** idea of staircase join.

- **Relational** XPath evaluation can outperform **state-of-the-art native** XML processors.

- The **Pathfinder** XQuery compiler exploits these ideas in its upcoming **SQL code generator**.
  - **http://www.pathfinder-xquery.org/**
  - Demo session 4, tomorrow afternoon

Pathfinder is supported by the German Research Foundation **DFG**.