MonetDB/XQuery:
using relational technology
to query XML documents

Peter Boncz
Centrum voor Wiskunde en Informatica
• The Extensible Markup Language (XML) is the universal format for structured documents and data on the Web.

• Base specifications:
  – XML 1.0, W3C Recommendation Feb '98
  – Namespaces, W3C Recommendation Jan '99
<catalog>
  <book isbn="ISBN 1565114302">
    <title>No Such Thing as a Bad Day</title>
    <author>Hamilton Jordan</author>
    <publisher>Longstreet Press, Inc.</publisher>
    <price currency="USD">17.60</price>
    <review>
      <reviewer>Publisher</reviewer>: This book is the moving account of one man's successful battles against three cancers ... <title>No Such Thing as a Bad Day</title> is warmly recommended.
    </review>
  </book>
</catalog>
<catalog>
  <book isbn="ISBN 1565114302">
    <title>No Such Thing as a Bad Day</title>
    <author>Hamilton Jordan</author>
    <publisher>Longstreet Press, Inc.</publisher>
    <price currency="USD">17.60</price>
    <review>
      <reviewer>Publisher</reviewer>: This book is the moving account of one man's successful battles against three cancers ... <title>No Such Thing as a Bad Day</title> is warmly recommended.
    </review>
  </book>
<!­­ more books and specifications -->
</catalog>
XML Databases

- Managing
  - Large XML Documents
  - Many XML Documents

- Manage =
  - Query (/Transform) + Update
  - Multiple users, Complex Queries, ACID Properties, ..
XQuery 1.0, a W3C Recommendation

- Functional, strongly-typed query language
- **XQuery 1.0 =**
  - *XPath 2.0* for navigation, selection, extraction
- **+ A few more expressions**
  - For-Let-Where-Order By-Return (FLWOR)
  - XML construction
  - Operators on types

**+ User-defined functions & modules**

- Modularize large queries
- Process recursive data

**+ Strong typing**

- Checks values of required type (operator, function)
- Guarantees result value instance of output type
- Enforced statically or dynamically
XPath: XML Navigation

- Titles of all books published by Longstreet Press
  \$cat/catalog/book[publisher="Longstreet Press"]/title

  <title>No Such Thing As A Bad Day</title>

- Publications with Don Chamberlin as author or editor
  \$cat//*[author|editor = "Don Chamberlin"]

  <book><title>XQuery from the Experts</title>...</book>
  <spec><title>XQuery Formal Semantics</title>...</spec>
FLWR ("Flower") Expressions

**FOR ...** sequence expression

**LET...** variable definition

**WHERE...** condition

**RETURN...** result expression
FLWR ("Flower") Expressions

SQL

FOR ... sequence expression
LE... variable definition
WHERE... condition
RETURN... result expression
ORDERBY

SELECT... result expressions
FROM... tables
WHERE... condition
GROUP BY...
ORDER BY...
XQuery Systems: 2 Approaches

- Tree-based
  - Tree is basic data structure
    - Also on disk (if an XQuery DBMS)
  - Navigational Approach
    - Galax, X-Hive, BerkeleyDB XML
  - Tree Algebra Approach
    - TIMBER

- Relational
  - Data shredded in relational tables
  - XQuery translated into database query (eg SQL)
The Pathfinder Project

• Challenge / Goal:
  – Turn RDBMSs into efficient XQuery engines

• People:
  – Torsten Grust, Jens Teubner
    TUM Technical University Munich
  – Maurice van Keulen
    Technical University of Twente
MonetDB/XQuery

XQuery Client
- XQuery Parsing
  - Normalization
  - XML Schema Import
  - Core Generation
  - Core Simplification
  - Type Checking
  - Core Optimization
  - MIL Generation

MonetDB Server
- Loop-lifted Staircase Join
- XML Serialization

MonetDB Kernel
MonetDB: Applied CS Research at CWI

• a decade of “query-intensive” application experience

  • image retrieval: Peter Bosch ➔ ImageSpotter
  • audio/video retrieval: Alex van Ballegooij ➔ RAM
  • XML text retrieval: de Vries / Hiemstra ➔ TIJAH
  • XML databases: Albrecht Schmidt ➔ XMARK
    Grust / vKeulen ➔ Pathfinder
  • GIS: Wilco Quak ➔ MAGNUM
  • data warehousing / OLAP / data mining
    SPSS ➔ DataDistilleries
    Univ. Massachusetts ➔ PROXIMITY

Monet research group successfully spun off DataDistilleries (now SPSS)
Background

- **MonetDB/XQuery: a fast XQuery Processor Powered by A Relational Engine**

  P. Boncz, T. Grust, M. van Keulen, S. Manegold, J. Rittinger, J. Teubner

- **Open-Source Download**
  - Mozilla License
  - Project Homepage
    - http://monetdb.cwi.nl
  - Developers website:
    - http://sf.net/projects/monetdb

RoadMap

- 14-apr-04: initial Beta release MonetDB/SQL
- 30-sep-04: first official release MonetDB/SQL
- 1-mar-05: beta release of MonetDB/XQuery (i.e. Pathfinder)
- 24-feb-06: second release of MonetDB/XQuery
- XX-jun-06: second release of MonetDB/XQuery
Outline

• Relational XQuery
  – Path steps in the pre/post plane
  – Translating for-loops, and beyond

• Query Optimization
  – Order prevention
  – Join recognition
  – Loop-Lifted Staircase join

• Further Research
  – Distributed XML (→ P2P Querying)
  – XML Updates
  – Querying XML Annotations

• Conclusions
Tree Encoding: XPath Accellerator

Node-based relational encoding of XQuery's data model

\[ \text{pre} + \text{size} - \text{level} = \text{post} \]
Quadrants are the main XPath axes

- Ancestor
- Following
- Preceding
- Descendant
Example query:
/descendant::open_auction[./bidder]/annotation

```
SELECT DISTINCT a.pre
FROM doc r, doc oa, doc b, doc a
WHERE r.pre=0
  AND oa.pre > r.pre AND oa.post < r.post
  AND oa.name = "open_auction" AND oa.kind = "elem"
  AND b.pre > oa.pre AND b.post < oa.post
  AND b.level = oa.level + 1
  AND b.name = "bidder" AND b.kind < "elem"
  AND a.pre > oa.pre AND a.post < oa.post
  AND a.level = oa.level + 1
  AND a.name = "annotation" AND a.kind < "elem"
ORDER BY a.pre
```
Staircase Join: Pruning

Example:

(c1,c2,c3,c4)/ancestor:*

SELECT DISTINCT doc.pre
FROM c, doc
WHERE doc.pre < c.pre
  AND doc.post < c.post

Eliminate: c1, c3
Keep: c2, c4
Staircase Join: Pruning

Example:

\[(c_1, c_2, c_3, c_4)/\text{ancestor:*}\]

```
SELECT DISTINCT doc.pre
FROM c, doc
WHERE doc.pre < c.pre
  AND doc.post < c.post
```

Eliminate: \(c_1, c_3\)

Keep: \(c_2, c_4\)
Staircase Join: Partitioning

Example:

\[(c_1, c_2, c_3) / \text{ancestor:}*\]

```
SELECT DISTINCT doc.pre
FROM c, doc
WHERE doc.pre < c.pre
 AND doc.post < c.post
```

Single-pass algorithm that avoids generating duplicates
Staircase Join: Partitioning

Example:

(c1,c2,c3)/ancestor:*

SELECT DISTINCT doc.pre
FROM c, doc
WHERE doc.pre < c.pre
  AND doc.post < c.post

Single-pass algorithm that avoids generating duplicates
Staircase Join: Skipping

Example:

\[(c1,c2)/\text{descendant:}*\]

```
SELECT DISTINCT doc.pre
FROM c, doc
WHERE doc.pre > c.pre
AND doc.post > c.post
```

Avoid comparing large chunks of the document table.
Tree Encoding: XPath Accelerator

Node-based relational encoding of XQuery's data model

pre + size - level = post
Sequence Representation

(10, “x”, <a/>, 10) →

- sequence = table of items
- add pos column for maintaining order
- ignore polymorphism for the moment

<table>
<thead>
<tr>
<th>Pos</th>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>“X”</td>
</tr>
<tr>
<td>3</td>
<td>pre(a)</td>
</tr>
<tr>
<td>4</td>
<td>10</td>
</tr>
</tbody>
</table>
For-loops: the iter column

Nested for blocks

\[
\begin{align*}
S_0 : & \quad \text{for } v_0 \text{ in } (10, 20) \\
S_1 : & \quad \text{for } v_1 \text{ in } (100, 200) \\
S_2 : & \quad \text{return } v_0 + v_1
\end{align*}
\]

- Derive \(v_0, v_1\)

\[
\begin{array}{ccc}
\text{iter} & \text{pos} & \text{item} \\
1 & 1 & 10 \\
1 & 2 & 20 \\
\end{array}
\]
For-loops: the iter column

Nested for blocks

\[
\begin{align*}
S_0 & \left[ \text{for } \$v_0 \text{ in } (10, 20) \right. \\
S_1 & \left. \left[ \text{for } \$v_1 \text{ in } (100, 200) \right. \\
S_2 & \left. \text{return } \$v_0 + \$v_1 \right. \\
\end{align*}
\]

- Derive \$v_0, \$v_1

$\$v_0$ in $s_1$: $\begin{array}{ccc}
\text{iter} & \text{pos} & \text{item} \\
1 & 1 & 10 \\
2 & 1 & 20 \\
\end{array}$

$\$v_1$ in $s_2$: $\begin{array}{ccc}
\text{iter} & \text{pos} & \text{item} \\
1 & 1 & 100 \\
2 & 1 & 200 \\
3 & 1 & 100 \\
4 & 1 & 200 \\
\end{array}$
Loop-lifting

Nested for blocks

\[
\text{for } \$v_0 \text{ in } (10, 20) \\
\text{\hspace{1cm} for } \$v_1 \text{ in } (100, 200) \\
\text{\hspace{2cm} return } \$v_0 + \$v_1
\]

\[
\text{loop}(s_0) \hspace{1cm} \text{loop}(s_1) \hspace{1cm} \text{loop}(s_2)
\]

<table>
<thead>
<tr>
<th>iter</th>
<th>pos</th>
<th>item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>200</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>200</td>
</tr>
</tbody>
</table>

- Derive \$v_0, \$v_1

\[
\$v_0 \text{ in } s_1: \begin{array}{ccc}
\text{iter} & \text{pos} & \text{item} \\
1 & 1 & 10 \\
2 & 1 & 20
\end{array}
\]

\[
\$v_1 \text{ in } s_2: \begin{array}{ccc}
\text{iter} & \text{pos} & \text{item} \\
1 & 1 & 100 \\
2 & 1 & 200 \\
3 & 1 & 100 \\
4 & 1 & 200
\end{array}
\]

▷ Variable \$v_0 \text{ in scope } s_2?
Loop-lifting

Nested for blocks

\[ s_0 \quad s_1 \quad s_2 \]

\[ \text{for } \$v_0 \text{ in } (10, 20) \]
\[ \text{for } \$v_1 \text{ in } (100, 200) \]
\[ \text{return } \$v_0 + \$v_1 \]

- Relation \textit{map} captures the semantics of nested iteration:

\[
\begin{array}{c|c}
\text{inner} & \text{outer} \\
1 & 1 \\
2 & 1 \\
3 & 2 \\
4 & 2 \\
\end{array}
\]

▷ Representation of \$v_0 \text{ in } s_2:

\[
\pi_{\text{iter}, \text{inner}, \text{pos}, \text{item}}(\$v_0 \land_{\text{iter}=\text{outer}} \text{map})
\]

\[
\begin{array}{c|c|c}
\text{iter} & \text{pos} & \text{item} \\
1 & 1 & 10 \\
2 & 1 & 10 \\
3 & 1 & 20 \\
4 & 1 & 20 \\
\end{array}
\]
for $v_0$ in (10, 20)
    for $v_1$ in (100, 200)
        $s_2 [ return \; v_0 + v_1$
Mapping Rules

XQuery construct $\rightarrow$ relational algebra

See VLDB’04 / TDM’04 [Grust,Teubner]

- Sequence construction $\rightarrow$ union
- If-Then-[Else] $\rightarrow$ select, [union]
- For loop $\rightarrow$ map with cartesian product (all combinations)
- Calculations $\rightarrow$ projection expressions
- List-functions (e.g. fn:first) $\rightarrow$ select(pos=1)
- Element Construction $\rightarrow$ updates using descendant
- Path steps $\rightarrow$ selections on the pre/post plane
  - Staircase join [VLDB03]:
    - Single-pass for a *set* of context nodes
    - elaborate skipping!
XMark Query 2 (common subexpr)
XMark Benchmark

1MB XML

Galax (GLX)

X-Hive (XHV)

Berkeley DB/XML (BDB)

eXist (EXT)

1GB XML
Outline

- Relational XQuery
  - Path steps in the pre/post plane
  - Translating for-loops, and beyond

- Query Optimization
  - Order prevention
  - Join recognition
  - Loop-Lifted Staircase join

- Further Research
  - Distributed XML (→ P2P Querying)
  - XML Updates
  - Querying XML Annotations

- Conclusions
Outline

• Relational XQuery
  – Path steps in the pre/post plane
  – Translating for-loops, and beyond

• Query Optimization
  – Order prevention
  – Join recognition
  – Loop-Lifted Staircase join

• Further Research
  – Distributed XML (➡ P2P Querying)
  – XML Updates
  – Querying XML Annotations

• Conclusions
Order Prevention

XQuery is highly order-aware (document, sequence)

- introduces many relational sorts \( \Rightarrow \) expensive: \( N \log(N) \)
- Goal: reduce the amount of sort operations
- Means: annotating query plan nodes with order properties
  
  + propagation rules for such properties

![Bar chart showing normalized performance/speedup for queries Q1 to Q20. The chart compares non-order preserving and order preserving cases, with the y-axis representing normalized performance/speedup and the x-axis representing queries. The chart includes a label indicating [110MB].]
XQuery construct ➔ relational algebra

- Sequence construction ➔ union
- If-Then-[Else] ➔ select, [union]
- For loop ➔ map with cartesian product (all combinations)
- Calculations ➔ projection expressions
- List-functions (e.g. fn:first) ➔ select(pos=1)
- Element Construction ➔ updates using descendant
- Path steps ➔ selections on the pre/post plane
  - Staircase join [VLDB03]:
    - Single-pass for a *set* of context nodes
    - elaborate skipping!

See VLDB’04 / TDM’04 [Grust,Teubner]
Join Recognition

for $p$ in $auction/site/people/person$
    for $t$ in $auction/site/closed_auctions/closed_auction$
      where $t/buyer/@person = $p/@id$
      return $t$

- For loop ➔ map with cartesian product (all combinations)
- If `simple’ condition exist on two loop variables ➔ join
  - Only make a map with the matching combinations
Join Recognition

for $p$ in $\$auction/site/people/person
  for $t$ in $\auction/site/closed_auctions/closed_auction$
    where $t/buyer/@person = $p/@id
    return $t$

- For loop ➔ map with cartesian product (all combinations)
- If `simple' condition exist on two loop variables ➔ join
  - Only make a map with the matching combinations

Performed on the XCore tree
Recognize if-then expressions
➔ Fragile to syntactic variation

Current research:
➔ Detect joins as functional dependencies in the relational plan
XQuery construct $\rightarrow$ relational algebra

- Sequence construction $\rightarrow$ union
- If-Then-[Else] $\rightarrow$ select, [union]
- For loop $\rightarrow$ map with cartesian product (all combinations)
- Calculations $\rightarrow$ projection expressions
- List-functions (e.g. fn:first) $\rightarrow$ select(pos=1)
- Element Construction $\rightarrow$ updates using descendant
- Path steps $\rightarrow$ selections on the pre/post plane
  - Staircase join [VLDB03]:
    - Single-pass for a *set* of context nodes
    - elaborate skipping!
Loop-lifted staircase join

- Staircase join [VLDB03]:
  - Single-pass for a *set* of context nodes
    - Loop-lifting $\Rightarrow$ multiple iters $\Rightarrow$ multiple sets of context nodes
  - elaborate skipping!
- Loop-Lifted Staircase Join
  In a single pass: process multiple input context node lists
    - Use a stack
    - Exploit axis properties for pruning
Staircase join

document

List of context nodes
Loop-lifted staircase join

List of context nodes

Multiple lists of context nodes

Active stack
Loop-lifted staircase join

- Staircase join [VLDB03]:
  - Single-pass for a *set* of context nodes
    - Loop-lifting ⇒ multiple iters ⇒ multiple sets of context nodes
  - elaborate skipping!
- Loop-Lifted Staircase Join
  - In a single pass: process multiple input context node lists
    - Use a stack
    - Exploit axis properties for pruning

[Bar chart showing normalized performance/speedup for different queries (Q1 to Q20), comparing iterative child and descendant step, loop-lifted child step, and loop-lifted child and descendant step with a speedup of 100MB.]
Outline

• Relational XQuery
  – Path steps in the pre/post plane
  – Translating for-loops, and beyond

• Query Optimization
  – Order prevention
  – Join recognition
  – Loop-Lifted Staircase join

• Further Research
  – Distributed XML (➔ P2P Querying)
  – XML Updates
  – Querying XML Annotations

• Conclusions
Outline

• Relational XQuery
  – Path steps in the pre/post plane
  – Translating for-loops, and beyond

• Query Optimization
  – Order prevention
  – Join recognition
  – Loop-Lifted Staircase join

• Further Research
  – Distributed XML (➔ P2P Querying)
  – XML Updates
  – Querying XML Annotations

• Conclusions
XML Updates

- Highly needed XML Database Functionality
  - January 2006: first W3C language specification
  - June 2006: MonetDB/XQuery with Updates (and SOAP)

"Updating The Pre/Post Plane in MonetDB/XQuery"

P. Boncz, S. Manegold, J. Rittinger
Proc. XIME-P workshop, june 2005
http://www.cwi.nl/~boncz/pre_post.pdf
XQuery over the Web

• SOAP
  – RPC (and other communications) over HTTP
  – Basic protocol for web services
  – Messages are XML

• XQuery RPC
  – Call XQuery User-Defined-Functions as a web service
  – Call Web-services from an XQuery
  – Distributed XQuery Execution
    • XML DBMS group communication
    • Loop-lifted: calls in for-loops optimized to single messages

• Research Topics (AmbientDB):
  – Distributed XQuery Optimization
  – The role of P2P data structures (e.g. DHT)
  – Distributed Updates: transactional semantics & efficient implementation (trade-offs)
Querying XML Annotations

- XPath Standoff Extensions (http://www.cwi.nl/~boncz/standoff.pdf XIME-P submission)
  - make it easy to query for region overlap / containment / etc
  - fast using interval index & loop-lifted interval merge join
- Suprisingly wide usage scenarios
  - multimedia annotations (MultimediaN)
  - forensic data analysis (NFI)
  - Natural language processing (UvA)
Conclusion

• MonetDB/XQuery
  – major CWI open-source software product
  – Co-op with TU Twente and TU Munich
  – monetdb.cwi.nl / recent SIGMOD paper

• Techniques for Relational XQuery
  – relational node encoding
  – complete compilation scheme
  – query optimization
    • fastest & most scalable system on XMark
  – various avenues for future research