The DB research group at U Konstanz

Background & the Pathfinder project

Marc H. Scholl

Graduate Programme Kick-Off Meeting, St. Vigilio, Sep 2004
I just got our consultant's report. He's identified our biggest problem.

I recommend that we build a tracking database.

We can put it on the network.

Would you like to hear what the problem is first?

I hate to dwell on the negative.

We like databases.
The team members

- Prof. Dr. Marc H. Scholl
- PD Dr. Torsten Grust* §
- Alex Holupirek
- Sabine Mayer*
- Jan Rittinger, research student GK
- André Seifert
- Dr. Weiwei Sun°
- Jens Teubner*
- Svetlana Vinnik, associate member GK
- Rui Xi, member GK

* ... leaving soon
§ ... many subsequent slides „stolen“ from him
° ... guest from Fudan U, Shanghai, China, for 1 year starting Sep 1, 2004
Background & some previous work

- **DB models & languages (DBPLs)**
  - Relational $\Rightarrow$ nested relational
  - $\Rightarrow$ object-relational $\Rightarrow$ *<you-name-it>*

- **DBMS architecture & implement’n**
  - Query compilation (optimization, transformation, parallelization)
  - System architecture & tuning
  - Prototype systems (from scratch & on-top)
Relational DBMSs: Built-in Table Awareness

- RDBMS derive much of their efficiency from the awareness of specific table properties:

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td>“x”</td>
</tr>
<tr>
<td>2</td>
<td>min</td>
<td>“y”</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>“a”</td>
</tr>
<tr>
<td>4</td>
<td>max</td>
<td>“b”</td>
</tr>
<tr>
<td>5</td>
<td></td>
<td>“c”</td>
</tr>
</tbody>
</table>

Cardinality and tuple width

⚠ Absence of duplicates/gaps, sort order, min/max values

- Exploit table properties during query plan generation and execution
Ordered, Unranked Trees: Core of the XML Data Model

- The element nesting in a well-formed XML document induces its ordered, unranked skeleton tree:

```xml
<a>
  <b><c/></b>
  <d/>
  <e>
    <f>
      <g/><h/>
    </f>
    <i><j/></i>
  </e>
</a>
```
Ordered, Unranked Trees: Core of the XML Data Model

- The element nesting in a well-formed XML document induces its ordered, unranked skeleton tree:

```xml
<a>
  <b><c/></b>
  <d/>
  <e>
    <f>
      <g/><h/>
    </f>
    <i><j/></i>
  </e>
</a>
```

```
  a
 /|
 / \
 b  c
 |
 d
|
 e
 /|
 / |
 f  g  h  i

j
```
Ordered, Unranked Trees: Core of the XML Data Model

- The element nesting in a well-formed XML document induces its ordered, unranked skeleton tree:

```xml
<a>
  <b><c/></b>
  <d/>
  <e>
    <f>
      <g/><h/>
    </f>
    <i><j/></i>
  </e>
</a>

▶ The skeleton abstracts from XML node kinds (element, text, comment, processing instruction nodes)
A diversity of tree properties will be of interest to our relational XML processor.

1. The size of the subtree rooted in node a is 5.
Skeleton Tree Properties

- A diversity of tree properties will be of interest to our relational XML processor

1. The **size of the subtree** rooted in node a is 5
2. The leaf-to-root **paths** of nodes b, c **meet** in node d
A diversity of tree properties will be of interest to our relational XML processor.

1. The **size of the subtree** rooted in node a is 5.
2. The leaf-to-root **paths** of nodes b, c **meet** in node d.
3. The **subtrees** rooted in e and a are necessarily **disjoint**.
XPath: Context Nodes and Axis Steps

- XML query and processing languages (XQuery, XSLT) are based on a common data-bound sub-language: XPath

  ⊳ **Context node** f, **four XPath axes**:

  

  ![Diagram](image)

  - f/preceding = (b, c, d)
  - f/ancestor = (a, e)
  - f/descendant = (g, h)
  - f/following = (i, j)
For any context node, the XPath axes preceding, descendant, ancestor, following partition the skeleton tree:

\[
\{a \ldots j\} = \{f\} \cup \bigcup f/\alpha \\
\alpha \in \{\text{preceding, descendant, ancestor, following}\}
\]

- Axes-based tree encoding would contain each node (exactly once)
- Encoding makes XPath axes semantics explicit
Node-based Relational Encodings of XQuery’s Data Model

- Staircase join (entifier) evaluates XPath axes on pre/post plane
- Tuple = node (¬ tree-based)
- Any encoding reflecting node identity/document order suffices
Implementation efforts

- Hand-“optimized” SQL equivalents of XPath location steps (on DB2)
  - See Svetlana’s talk
- Built some basic support into relational DBMS engine (PostgreSQL)
  - See Sabine’s work
- Automatic translation into relational algebra + optimizations
  - See subsequent slides
- Implementation based upon MonetDB
  - See Jan’s talk
A Purely Relational XQuery Processing Stack

- A fully relational XQuery processor, developed bottom-up:

  SQL, relational algebra → RDBMS
A Purely Relational XQuery Processing Stack

- A fully relational XQuery processor, developed bottom-up:

  - XPath accelerator ✓
  - Tree Encoding
  - SQL, relational algebra
  - RDBMS
A Purely Relational XQuery Processing Stack

- A **fully relational** XQuery processor, developed bottom-up:

  - *staircase join ✓*
  - *XPath accelerator ✓*
  - SQL, relational algebra
  - **XPath Axes**
  - **Tree Encoding**
  - **RDBMS**
A Purely Relational XQuery Processing Stack

* A **fully relational** XQuery processor, developed bottom-up:

- derivation ✓
- staircase join ✓
- XPath accelerator ✓
- SQL, relational algebra
- Validation
- XPath Axes
- Tree Encoding
- RDBMS
A Purely Relational XQuery Processing Stack

- A **fully relational** XQuery processor, developed bottom-up:

  now →

  derivation ✓

  *staircase join* ✓

  *XPath accelerator* ✓

  SQL, relational algebra

  +---+   +---+   +---+   +---+   +---+
  | XQuery | Validation | XPath Axes | Tree Encoding | RDBMS |
  +---+   +---+   +---+   +---+   +---+
Source Language: XQuery Core

XQuery Core

- literals
- sequences ($e_1, e_2$)
- variables ($v$)
- let...return
- for...where...return
- for...[at $v$]...where...return
- if...then...else
- typeswitch...case...default
- element {...} {...}
- text {...}
- XPath ($e/\alpha$)
- function application

- document order ($e_1 << e_2$)
- node identity ($e_1$ is $e_2$)
- arithmetics (+,-,*,idiv)
- fn:doc()
- fn:root()
- fn:data()
- fn:distinct-doc-order()
- fn:count()
- fn:sum()
- fn:empty()
- fn:position()
- fn:last()

- Expression may nest as defined by W3C XQuery Working Draft
Target Language: Flat Relational Algebra

Relational Algebra

- $\pi$ column projection, renaming
- $\sigma$ row selection
- $\dot{\cup}, \setminus$ disjoint union, difference
- $\delta$ duplicate elimination
- $\bowtie$ equi-join
- $\times$ Cartesian product
- $\rho$ row numbering
- $\sqcup$ staircase join
- $\varepsilon, \tau$ element/text node construction
- $\otimes$ arithmetic/comparison/Boolean operator $*$

- No tree pattern matching or similar operators involved here
- This algebra is efficiently implementable on (top of) SQL hosts
Properties of Compiled Query Plans

- The target algebra as well as the compiled plans exhibit a number of nice properties:

  Algebra/plan properties
  - $\pi$ no need to eliminate duplicate tuples
  - $\cup$ all unions are disjoint
  - $\bowtie$ all joins are equi-joins
  - $\times$ one input is singleton (column attachment)
  - plans are DAGs with significant sharing

- Simple, “assembly style” operators with simple semantics
XQuery on SQL Hosts

XMark Query Q8

```xml
for $p in fn:doc("auction.xml")/site/people/person
return let $a := for $t in fn:doc("auction.xml")/site/
closed_auctions/closed_auction
  return if fn:data($t/buyer/person/text()) =
    fn:data($p/id/text())
    then $t else ()
return <item> { <person> { $p/name/text() } </person>,
  text { fn:count($a) } } </item>
```

- Compiles into DAG of 120 algebraic operators, significant sharing
  - Equivalent tree has \(\approx 2000\) nodes
- **NB:** No optimizations applied yet (neither XQuery nor algebraic)
Work in Flux

Optimizations

- Exploit **column properties**: unique, constant, dense
- **Order awareness** [ICDE 2004, SIGMOD 1996]
- Exploit **disjointness** of intermediate results
- "**Live node analysis**": evaluate \(\square\) over minimal tree fragments

New implementation

- Main-memory DBMS kernel **MonetDB** (CWI, Amsterdam)
- Target language is MIL, algebra over binary tables
- Ordered data model (\(\rho\) may largely become obsolete)
An Injection of Tree Awareness

- **PostgreSQL 7.3.3**
  - Enhanced database kernel (query planner, executor)

- **MonetDB** (CWI, Amsterdam)
  - Throughput $\approx 1 \text{ GB/s}$ of encoded XML, full use of cache lines
  - Highly predictable memory access pattern (preloads cache lines)

[VLDB 2003] [VLDB 2004]
Future work, for example

- Add retrieval-style functionality (inexact search, ranking, …)
  - Rui Xi (?)

- Add stream-processing (single-pass processing, “infinite” input, …)
  - Weiwei Sun (?)

- Applications
  - Other groups within the Graduate Programme (HR, MB, DK, …)
  - External partners
Done!

- Questions?
- Suggestions?