Materialized views for P2P XML warehousing

Ioana Manolescu\(^1\)  Spyros Zoupanos\(^1\)

\(^1\)GEMO/IASI group, INRIA Saclay – Île-de-France

July 2009
Dataring project meeting, Nantes
ViP2P at a glance

An XML data management system based on a DHT

- Declare tree pattern XML views over the network data
- Fill in the views with XML data
- Answer tree pattern queries using the existing views
  - View definition lookup
  - Query rewriting $\Rightarrow$ logical plan
  - Translation to a (distributed) physical plan
  - Execution of the physical plan
Architecture overview

Materialized views for P2P XML warehousing
The peers may store:
- documents
The peers may store:
- documents
- views
When $q$ arrives:
When $q$ arrives:
- view definition
- lookup
Architecture overview

When $q$ arrives:
- view definition
- rewriting
When $q$ arrives:
- view definition lookup
- rewriting
- execution of physical plan
When $d$ arrives:

- Search view definitions for which $v_i(d) \neq \emptyset$
- Compute $v_i(d)$
- Send results
When $d$ arrives:
- search view definitions for which $v_i(d) \neq \emptyset$
When $d$ arrives:

- search view definitions for which $v_i(d) \neq \emptyset$
- compute $v_i(d)$
When $d$ arrives:
- search view definitions for which $v_i(d) \neq \emptyset$
- compute $v_i(d)$
- send results
Tree pattern dialect $P$

$a_{cont}$
Tree pattern dialect $P$

$a_{cont} \quad a_{id,cont}$
Tree pattern dialect $P$

\[ a_{\text{cont}} \quad a_{\text{id,cont}} \quad a_{\text{id,val}} \]
Tree pattern dialect $P$

$a_{cont} \ a_{id, cont} \ a_{id, val}$

$a_{id}$

$\mid$

$b_{val}$
Tree pattern dialect $P$

\[
\begin{align*}
\text{a}_{\text{cont}} & \quad \text{a}_{\text{id},\text{cont}} & \quad \text{a}_{\text{id},\text{val}} \\
\text{a}_{\text{id}} & \\
\text{b}_{\text{val}} & \\
\text{a}_{\text{val}} & \\
\text{b}_{\text{id}} & \quad \text{d}_{\text{cont}} \\
\text{c}_{\text{cont}} & \quad \text{e}_{\text{id},\text{val}}
\end{align*}
\]

Materialized views for P2P XML warehousing
Pattern semantics and equivalence

\[
\begin{align*}
&\text{a}_{(1,12)} \\
&\text{b}_{(2,5)} \quad \text{f}_{(7,6)} \quad \text{b}_{(8,11)} \\
&\text{c}_{(3,4)} \quad \text{b}_{(9,9)} \quad \text{h}_{(12,10)} \\
&\text{d}_{(4,1)} \quad \text{e}_{(5,3)} \quad \text{g}_{(10,8)} \\
&\text{some}_{(6,2)} \quad \text{text}_{(11,7)}
\end{align*}
\]

\[
\begin{align*}
&\text{a}_{\text{val}} \\
&\text{b}_{\text{cont}} \\
&\text{f}_{\text{id}}
\end{align*}
\]
Pattern semantics and equivalence

\[
\begin{array}{c}
\text{some text} \\
\langle b \rangle < c > < d / > < e > \text{some} \rangle / e > \rangle / c > \rangle / b \rangle \\
\end{array}
\]
### Pattern semantics and equivalence

Consider the following pattern:

- \( a_{(1,12)} \)
  - \( b_{(2,5)} \)
    - \( f_{(7,6)} \)
    - \( b_{(8,11)} \)
  - \( c_{(3,4)} \)
  - \( b_{(9,9)} \)
  - \( h_{(12,10)} \)
- \( d_{(4,1)} \)
- \( e_{(5,3)} \)
- \( g_{(10,8)} \)
- some\( (6,2) \)
- text\( (11,7) \)

#### Table

<table>
<thead>
<tr>
<th></th>
<th>( a_{val} )</th>
<th>( b_{cont} )</th>
<th>( f_{id} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>some text</td>
<td>(&lt;b&gt;&lt;c&gt;&lt;d/&gt;&lt;/c&gt;&lt;e&gt;some&lt;/e&gt;&lt;c/&gt;&lt;/b&gt;)</td>
<td>( (7,6) )</td>
<td></td>
</tr>
<tr>
<td>some text</td>
<td>(&lt;b&gt;&lt;b&gt;&lt;g&gt;text&lt;/g&gt;&lt;b/&gt;&lt;/b&gt;&lt;/b&gt;)</td>
<td>( (7,6) )</td>
<td></td>
</tr>
</tbody>
</table>
Materialized views for P2P XML warehousing
Pattern semantics and equivalence

- Document $d$, pattern $p \Rightarrow p(d)$
- $p_1 \equiv p_2 \iff \forall d, p_1(d) = p_2(d)$

<table>
<thead>
<tr>
<th>$a_{val}$</th>
<th>$b_{cont}$</th>
<th>$f_{id}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>some text</td>
<td>$&lt;b&gt;&lt;c&gt;&lt;d/&gt;&lt;e&gt;some&lt;/e&gt;&lt;/c&gt;&lt;/b&gt;$</td>
<td>(7,6)</td>
</tr>
<tr>
<td>some text</td>
<td>$&lt;b&gt;&lt;b&gt;&lt;g&gt;text&lt;/g&gt;&lt;/b&gt;&lt;/b&gt;$</td>
<td>(7,6)</td>
</tr>
<tr>
<td>some text</td>
<td>$&lt;b&gt;&lt;g&gt;text&lt;/g&gt;&lt;/b&gt;$</td>
<td>(7,6)</td>
</tr>
</tbody>
</table>
Algebraic rewriting & operators

Let $q \in \mathcal{P}$ be a query and $\mathcal{V} = \{v_1, v_2, \ldots, v_k\}$ a set of views. A rewriting of $q$ using $\mathcal{V}$ is an algebraic expression

$$e(v_1, v_2, \ldots, v_k)$$

such that $e(D) = q(D)$ for any document set $D$

### Algebra operators

- $\text{scan}(v)$
- $\pi_{cols}(op)$
- $\text{sort}_{cols}(op)$
- $\text{nav}_{i, np}(op)$ evaluates $np$ over the $\text{cont}$ attribute $op.i$
- $\times$
- $\pi^0(op)$
- $\sigma_{\text{cond}}(op)$
Canonical expression:

We are interested in **canonical rewritings** only
Minimal canonical rewritings

Canonical expression:

We are interested in **minimal canonical rewritings** only
Rewriting example

\[ q \]

\[ a \]

\[ b \]

\[ d_{val=5} \]

\[ c_{cont} \]

\[ e_{val} \]

\[ v_1 \]

\[ v_2 \]

\[ v_3 \]

\[ a_{id} \]

\[ b_{id,cont} \]

\[ c_{id,cont} \]

\[ d_{val} \]
Rewriting example

\[ q \]

\[ \pi_{c.cont,e.val} \]

\[ \sigma_{a.id=a.id \land b.id \prec c.id \land a.id \prec b.id \land d.val=5} \]

\[ \text{nav}_{b.cont,e.val} \]

\[ v_1 \]

\[ v_2 \]

\[ v_3 \]

Materialized views for P2P XML warehousing
Finding query rewritings

Idea:
Compute covers of the query nodes with the view nodes.
Finding query rewritings

Idea:
Compute covers of the query nodes with the view nodes.

$q \overset{a_{id}}{\leftarrow} \overset{b_{id}}{\rightarrow} v_1 \overset{a_{id}}{\leftarrow} v_2 \overset{b_{id}}{\rightarrow}$
Finding query rewritings

Idea:
Compute covers of the query nodes with the view nodes.

$q$

$\text{id}$

$v$

$\text{id}_{cont}$
Finding query rewritings

Idea:
Compute covers of the query nodes with the view nodes.

\[ q \]

\[ v \]

- \( a_{id} \)
- \( b \)
- \( c \)
- \( d_{val} \)

- \( a_{id,cont} \)
- \( b \)
- \( c \)
- \( d_{val} \)
Finding query rewritings

**Idea:**
Compute covers of the query nodes with the view nodes.

```
q
  a
  b
  c_val

v_1
  a_id
  b_id
  c_val
  d_val

v_2
  a_id
  b_id
  c_val
  d_val
```
Finding query rewritings

Idea:
Compute covers of the query nodes with the view nodes.

No rewriting
Finding query rewritings

Idea:
Compute covers of the query nodes with the view nodes.

\[ q \quad \quad \quad \quad v_1 \quad \quad \quad \quad v_2 \]

\[ \begin{array}{ccc}
    a & & \quad a_{id} \\
    b & & \quad c_{id} \\
    c & & \quad d_{val} \\
    d_{val} & & \quad e_{val}
\end{array} \quad \quad \begin{array}{ccc}
    b & & \quad e_{val}
\end{array} \]
Finding query rewritings

Idea:
Compute covers of the query nodes with the view nodes.

No rewriting
Finding query rewritings

Idea:
Compute covers of the query nodes with the view nodes.

```
q
  a    v1
 b   c_val
 d_val

v2
  a_id
    b
    c_val
    d_val
```
Finding query rewritings

Idea:
Compute covers of the query nodes with the view nodes.

\[ q \rightarrow v_1 \rightarrow v_2 \]

No rewriting
Rewriting algorithms

SE (Subset Enumeration)
- For each new subset, check if a rewriting can be found
- Test minimality at the end

ISE (Increasing Subset Enumeration)
- Like SE but enumerates sets from the smallest to the largest
- Finds minimal rewritings first

SE and ISE
- Repeat work
- Try all possible subsets
Rewriting algorithms

Bottom-up algorithms: use smaller partial rewritings to build bigger ones

DPR (Dynamic Programming Rewriting)
- Dynamic programming style

DFR (Depth First Rewriting)
- Greedy based on the biggest query coverage
Bottom-up rewriting

\[
\begin{align*}
q \\
a \\
b \\
c_{\text{cont}} & \quad d_{[\text{val}=5]} \\
e_{\text{val}} \\
v_1 & \quad v_2 & \quad v_3 \\
& \quad a_{\text{id}} & \quad a_{\text{id}} \\
& \quad b_{\text{id,cont}} & \quad c_{\text{id,cont}} & \quad d_{\text{val}}
\end{align*}
\]
Bottom-up rewriting

$q$

\[
\begin{align*}
 & a \\
 & \downarrow \\
 & b \\
 & \downarrow \\
 & c_{cont} \\
 & \downarrow \\
 & v_1 \\
 & \downarrow \\
 & a_{id} \\
 & \downarrow \\
 & b_{id,\text{cont}}
\end{align*}
\]

\[
\begin{align*}
 & e_{val} \\
 & \downarrow \\
 & v_2 \\
 & \downarrow \\
 & c_{id,\text{cont}} \\
 & \downarrow \\
 & d_{val} \\
 & \downarrow \\
 & v_3 \\
 & \downarrow \\
 & a_{id} \\
 & \downarrow \\
 & \text{nav}_{b_{cont}, e_{val}}
\end{align*}
\]

\[
\begin{align*}
 & \bowtie_{a.id \prec b.id} \\
 & \downarrow \\
 & v_2 \\
 & \downarrow \\
 & v_1
\end{align*}
\]
Bottom-up rewriting

\[ q \]

\[ v_1 \]

\[ v_2 \]

\[ v_3 \]

\[ a_{id} \]

\[ b_{id,cont} \]

\[ c_{id,cont} \]

\[ d_{val} \]

\[ \sigma_{b.id < c.id} \]

\[ \bowtie_{a.id < b.id} \]

\[ \text{nav}_{b.cont,e.val} \]

\[ v_2 \]

\[ v_1 \]
Bottom-up rewriting

\[ q \]

\[ \begin{array}{c}
  a \\
  b \\
  c_{\text{cont}} \\
  e_{\text{val}} \\
\end{array} \quad \begin{array}{c}
  a_{\text{id}} \\
  b_{\text{id,cont}} \\
\end{array} \quad \begin{array}{c}
  d_{[\text{val}=5]} \\
  c_{\text{id,cont}} \\
  d_{\text{val}} \\
\end{array} \]

\[ \sigma_{b.\text{id} \prec c.\text{id}} \quad \sigma_{\text{val}=5} \]

\[ \bigtriangleup a.\text{id} = a.\text{id} \]

\[ \bigtriangledown a.\text{id} \prec b.\text{id} \]

\[ \text{nav}_{b.\text{cont},e.\text{val}} \]

\[ v_1 \quad v_2 \quad v_3 \]

Materialized views for P2P XML warehousing
Bottom-up rewriting

\[ q \]

\[ \pi_{c.cont, e.val} \]

\[ \bowtie a.id = a.id \]

\[ \sigma_{b.id < c.id} \]

\[ \sigma_{val = 5} \]

\[ \bowtie a.id < b.id \]

\[ v_3 \]

\[ nav_{b.cont, e.val} v_2 \]

\[ v_1 \]
Bottom-up rewriting

\[ q \]

\[ \pi_{c.cont,e.val} \]

\[ \sigma_{a.id=a.id \land b.id \prec c.id \land a.id \prec b.id \land d.val=5} \]

\[ \times \]

Materialized views for P2P XML warehousing
SE, ISE, DPR and DFR are correct and complete. They produce all minimal canonical rewritings of $q$ given $\mathcal{V}$.

- Which rewritings are "good"?
  - The one which leads to the best physical plan
  - We learn this too late!

- Heuristic: a good rewriting uses the **smallest number** of views.
  - DFR typically finds fast a solution which is reasonably good
  - ISE, DPR will need more time but return better quality results. They produce rewritings towards the end of the search
Peer $p$ has a view $v$, peer $p_d$ publishes a document $d$
$p$ indexes $v$ on the DHT by the labels of the view
$p_d$ looks up the labels and keywords of $d$
- View set $S_a$
  - All views $v$ such that $v(d) \neq \emptyset$ are in $S_a$
$p_d$ evaluates $v(d)$ for each $v \in S_a$
Query \( q \) asked at peer \( p \) ⇒ \( p \) needs to find useful views

4 different strategies

- **Label indexing** (LI):
  - Index \( v \) by each \( v \) node label
  - Look up by all node labels of \( q \)

- **Return label indexing** (RLI):
  - Index \( v \) by the labels of all \( v \) nodes which project some attributes
  - Same as for LI: use the labels of all \( q \) nodes as lookup keys
Indexing and lookup view definitions

- **Leaf path indexing (LPI):**
  - Let $LP(v)$ be the set of all the distinct root-to-leaf label paths of $v$.
  - Index $v$ using each element of $LP(v)$ as key
  - Look up details in the paper

- **Return Path Indexing (RPI):**
  - Let $RP(v)$ be the set of all rooted paths in $v$ which end in a node that returns some attribute. Index $v$ using each element of $LP(v)$ as key
  - Look up is the same as for LPI
ViP2P platform

- Fully implemented using Java 6
- Used Berkeley DB (version 3.3.75) to store view data
- Used FreePastry (version 2.1) as our DHT network
- Experiments carried on Grid5000 using **250 machines**
- **1000 ViP2P peers** were deployed
We used 1440 views related to but different from query $q$
We used 1440 views related to but different from query $q$. 

![Diagram showing view look up performance](chart.png)
For the experiments we indexed 2000 XMark documents and 500 views (70 related to the documents)
Performance of rewriting algorithms

- Graphs showing the performance of rewriting algorithms as a function of query size.
  - DFR First vs. DFR Total vs. DPR First vs. DPR Total vs. ISE First vs. ISE Total.
  - Time (ms) on the y-axis.
  - Query size on the x-axis.

Materialized views for P2P XML warehousing
Materialized views for P2P XML warehousing
Related work

Indexing documents in the DHT \[GWJD03, BC06, SHA05, AMP^{+}08\]

XPath query rewriting \[BOB^{+}04, XO05, CDO08, TY{"O}^{+}08\]
- XPath: wildcard *, union
- Rewritings: intersection, navigations, joins

Rewriting with structural constrains \[ABMP07\]
- Centralized setting
- Dataguide \[GW97\] constraints
ViP2P: data access support structures for DHT based XML data management

All the presented algorithms have been fully implemented in a functional Java based platform

Presented at DataX 2009 (no proceedings)

Extended version submitted for publication

Visit us at vip2p.saclay.inria.fr!
Thank you!


[CDO08] **Bogdan Cautis, Alin Deutsch, and Nicola Onose.**

Xpath rewriting using multiple views: Achieving completeness and efficiency.

[GW97] **Roy Goldman and Jennifer Widom.**

Dataguides: Enabling query formulation and optimization in semistructured databases.

[GWJD03] **L. Galanis, Y. Wang, S.R. Jeffery, and D.J. DeWitt.**

Locating data sources in large distributed systems.

[SHA05] **Gleb Skobeltsyn, Manfred Hauswirth, and Karl Aberer.**

Efficient processing of XPath queries with structured overlay networks.
In *OTM Conferences (2)*, 2005.

[TYÖ+08] **Nan Tang, Jeffrey Xu Yu, M. Tamer Özsu, Byron Choi, and Kam-Fai Wong.**
Multiple materialized view selection for xpath query rewriting.

[XO05] **W. Xu and M. Ozsoyoglu.**
Rewriting XPath queries using materialized views.
In *VLDB*, 2005.