Efficient XQuery Rewriting using Multiple Views

Ioana Manolescu ¹ Konstantinos Karanasos ¹
Vasilis Vassalos ² Spyros Zoupanos ¹

¹Leo team, INRIA Saclay and Univ. Paris-Sud XI, France ²AUEB, Greece

ICDE 2011, Hannover, Germany
April 15, 2011
XML view-based rewriting

• **XML**: the standard model for data on the Web
• **XQuery**: de-facto query language for XML
• **Materialized views** crucial in query optimization
XML view-based rewriting

- **XML**: the standard model for data on the Web
- **XQuery**: de-facto query language for XML
- **Materialized views** crucial in query optimization

- Equivalent XQuery rewriting using multiple XQuery views
Steps of view-based query evaluation

• Filter out useless views
• Find all equivalent query rewritings
  – the problem we consider
• Choose and evaluate the most appropriate rewriting
  – query optimizer, execution engine
  – implemented in ViP2P (http://vip2p.saclay.inria.fr)
Target rewrites:

- **Equivalent**: the same results as the query
- **Complete**: no need of the base documents
- **Multiple views**: joining them a problem on its own
- **Minimal**: no redundant view instances
- **Generic logical XML algebra**: easy to use by existing systems
Novelties of our approach

• **Query/view language**: powerful conjunctive XQuery dialect
  – for ... where ... return ...
  – bindings from multiple documents

• Rewriting **minimality** built in the search process

• Efficient query **rewriting algorithms**
  – algebra based pruning
  – exploit XDBMS implementation artifacts when available (IDs, structural IDs)
Query and view language

for $s$ in doc(”sculptures”)/sculpture,
  $t$ in $s/title$, $c$ in $s//country$, $y1$ in $s/year$,
  $p$ in doc(”museums”)/Sprengel/painting,
  $y2$ in $p/year$, $pr$ in $p//painter

where $c$=’Germany’ and $y1$=$y2

return 〈v 〈tval〉{string($t)}〈/cval〉 〈pid〉{id($p)}〈/pid〉
  〈prid〉{id($pr)}〈/prid〉 〈prcont〉{$pr}〈/prcont〉〈/v〉
Query and view language

for $s$ in doc("sculptures")//sculpture,
  $t$ in $s$/title, $c$ in $s$/country, $y1$ in $s$/year,
  $p$ in doc("museums")//Sprengel/painting,
  $y2$ in $p$/year, $pr$ in $p$/painter
where $c$='Germany' and $y1$=$y2$
return <v><tval>string($t$)</tval><cval><pid>id($p$)</pid>
  <prid>id($pr$)</prid><prcont>$pr$</prcont></cval></v>
Query/view data

- sculpture
  - title_val
  - country [Germany]
- year
- Sprengel
  - painting_ID
  - year
  - painter_ID cont
Query/view data

title.val  painting.ID  painter.ID  painter.cont

...    ...    ...    ...

Efficient XQuery Rewriting using Multiple Views
Outline

- Introduction

➢ Motivating example

- Tree pattern rewriting
- Joined pattern rewriting
- Experiments
- Conclusion
Motivating example

ICDE 2011
Motivating example
Motivating example
Motivating example

Navigation (compensation)

Axis adaptation
Motivating example

Navigation (compensation)
Axis adaptation
Upward joining

ICDE 2011
Motivating example
Motivating example

Navigation (compensation)
Axis adaptation
Upward joining
Downward joining
Value joins
Outline

• Introduction
• Motivating example
  ➢ **Tree pattern rewriting**
  • Joined pattern rewriting
  • Experiments
• Conclusion
LDQT rewritings

$q \parallel s_{id} \parallel c_{id} \parallel t_{id}

v_1 \parallel s_{id} \parallel c_{id} \parallel t_{id}

v_2 \parallel v_3 \parallel
LDQT rewrites
LDQT rewritings

\[ q \perp s_{id} \perp t_{id} \]

\[ v_1 \perp s_{id} \perp c_{id} \]

\[ v_3 \perp t_{id} \]

\[ e_1: \sigma_{s < c} \land s < t \]

\[ e_2: \sigma_{s < c} \land s < t \]

\[ e_3: \sigma_{s < t} \]

ICDE 2011
Every rewriting can be transformed to an LDQT rewriting
Rewriting algorithm

- **Choose** the useful (query embeddable) views
  - 1-view (partial) rewritings
- **Adaptation** of 1-view rewritings
  - navigation, axis adaptation, selection predicates
- **Join** partial rewritings to reach equivalent rewritings
- **Sound and complete** algorithm
Motivating example
Motivating example

\[
\begin{align*}
q' &\leadsto q' \text{ museums} & v_1 \leadsto v_1 \text{ museums} & \quad v_1 \text{ painting id} \\
&\quad q' \text{ Sprengel} & & \quad v_1 \text{ painting id} \\
&\quad q' \text{ painting} & & \quad v_1 \text{ info id, cont} \\
&\quad q' \text{ painter} & & \\
&\quad q' \text{ year} & & \\
&\quad q' \text{ info} & & \\
&\quad q' \text{ country} & & \\
\end{align*}
\]
Navigation (compensation)
Axis adaptation
Motivating example

Navigation (compensation)

Axis adaptation

Upward joining

Downward joining

Value joins
Motivating example
Query-driven join

• Given two partial rewritings for query \( q \), join them into a new (partial) rewriting for \( q \)
• Equi- and structural-joins (if structural IDs)
• Zipping: the paths above joining nodes
• Merging: the forests below joining nodes
• Extra challenges due to multiple returning nodes
Tree pattern joining

$q' \parallel \begin{array}{l}
museums \parallel \\
Sprengel \parallel \\
painting \parallel \\
painter \parallel \text{year} \\
info \parallel \\
country
\end{array}$

$v_1' \parallel 
\begin{array}{l}
museums \parallel \\
painting_{ID} \parallel \\
info_{ID, cont} \parallel \\
country_{val}
\end{array}$

$v_2'' \parallel 
\begin{array}{l}
Sprengel \parallel \\
painting_{ID} \parallel \\
year \parallel \text{painter}_{ID}
\end{array}$
Tree pattern joining

\[ q' \]

- museums
- Sprengel
- painting
- painter
- info
- country

\[ v_1' \]

- painting
  - info
  - country

\[ v_2'' \]

- Sprengel
- painting
  - info
  - country

\[ \sigma_{p1 = p2} \]

\[ X \]

\[ e_1 \]

\[ e_2 \]

ICDE 2011
Zipping

\[ q' \]

museums \[ \parallel \]
Sprengel \[ \parallel \]
painting

\[ v_{12} \] painting \[ \_ID \]

year painter \[ \_ID \]
info info \[ \_ID, \_cont \]
country country \[ \_val \]
Merging

\[ v_{12} \]

\[ q' \]

- museums
- Sprengel
- painting
- painter
- info
- country

- year
- painter ID
- info ID, cont
- country val
Merging

\[ q' \]

\[ \sigma_{p1 = p2 \land pr < in} \]

\[ \mathsf{e}_1 \]

\[ \mathsf{e}_2 \]
Rewriting strategies

• Strategy: order of partial rewriting joins
• Naïve dynamic programming (NDP)
  – rewrites of $k$ views only after all of $(k-1)$ views
• Query-driven dynamic programming (QDP)
  – joins inspired by query nodes and edges
• Query-driven depth-first (QDF)
  – greedily cover the biggest part of the query
• Strategies features:
  – Output only minimal rewrites
  – NDP and QDP find min-size rewrites
  – QDF may reach a solution more quickly
  – upper bound on the number of joins in an equivalent rewriting
Motivating example

Navigation (compensation)
Axis adaptation
Upward joining
Downward joining
Value joins

ICDE 2011
Motivating example

Navigation (compensation)
Axis adaptation
Upward joining
Downward joining
Value joins
Outline

• Introduction
• Motivating example
• Tree pattern rewriting
  ➢ **Joined pattern rewriting**
• Experiments
• Conclusion
Joined pattern rewriting

• Let $jq$ a joined query and $JV$ a joined pattern view set

<table>
<thead>
<tr>
<th>Joined pattern queries</th>
<th>Conjunctive queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>tree pattern</td>
<td>query atom</td>
</tr>
<tr>
<td>value join edge</td>
<td>join predicate (shared variable)</td>
</tr>
</tbody>
</table>

• **Bucket-style** algorithm:
  Rewrite each query tree pattern and combine the solutions.

• **Correct and complete** algorithm
Joined pattern rewriting
Joined pattern rewriting

\[ \text{q} \cdot t_1 \quad | \quad \text{q} \cdot t_2 \quad | \quad v_1 \quad | \quad v_2 \cdot t_1 \quad | \quad v_2 \cdot t_2 \]

- \text{museums}
- \text{Sprengel}
- \text{painting}
- \text{painter}
- \text{country}
- \text{year}
- \text{title}_{\text{val}}
- \text{country}_{\text{val}}
- \text{painting}_{\text{ID}}
- \text{info}_{\text{ID, cont}}
- \text{Sprengel}
- \text{painting}_{\text{ID}}
- \text{year}
- \text{painter}_{\text{ID}}
Joined pattern rewriting

q.t<sub>1</sub> || q.t<sub>2</sub> || v<sub>1</sub> || v<sub>2</sub>.t<sub>2</sub> || v<sub>2</sub>.t<sub>1</sub> || e.t<sub>1</sub> || e.t<sub>2</sub>

museums || sculpture
Sprengel || title<sub>val</sub>
 painting || country<sub>val</sub>
 painter || year
 info ||
country ||

σ<sub>p1 = p2</sub> || X
σ<sub>p < pr</sub> || nav

country<sub>val</sub> ||

ICDE 2011
Efficient XQuery Rewriting using Multiple Views
Joined pattern rewriting

\[ e.t_1 \quad \sigma_{p1 = p2} \]

\[ X \]

\[ \sigma_{p < pr} \quad \text{nav} \quad \parallel \quad \text{country val} \]

\[ v_2.t_2 \quad v_1 \]

\[ e.t_2 \]

\[ v_2.t_1 \]
Joined pattern rewriting

\[ e.t_1 \quad \sigma_{p_1 = p_2} \quad \sigma_{p < \text{pr}} \quad \text{nav} \quad \text{country}_{\text{val}} \]

\[ e.t_2 \]

\[ e.\text{t}_1 \quad e.\text{t}_2 \quad \sigma_{c_1.\text{val} = c_2.\text{val}} \]
Motivating example

**Navigation** (compensation)

Axis adaptation

Upward joining

Downward joining

Value joins

ICDE 2011
Motivating example

Navigation (compensation)
Axis adaptation
Upward joining
Downward joining
Value join

ICDE 2011
Outline

• Introduction
• Motivating example
• Tree pattern rewriting
• Joined pattern rewriting

➤ **Experiments**

• Conclusion
Experimental platform

- Fully implemented in **Java 6**
- **BerkeleyDB v3.3.75** for materialized views
- **Saxon v9.1** for navigation operator
- Execution engine of our **ViP2P** system
- **XMark** documents and queries
View-based query evaluation performance (100MB XMark doc)

Execution time (ms)

- TPQ1
- TPQ2
- TPQ3
- TPQ4
- JTPQ1
- JTPQ2
- JTPQ3

ICDE 2011
Efficient XQuery Rewriting using Multiple Views
Scalability of view-based query evaluation

![Graph showing scalability of view-based query evaluation]

- Execution time (ms)
- Document size (MB)

- TPQ1
- TPQ2
- TPQ3
- TPQ4
- JTPQ1
- JTPQ2
- JTPQ3
Rewriting strategies performance

![Bar chart showing rewriting time for different views and strategies.](chart.png)

- QDF first
- QDF all
- QDP first
- QDP all
- NDP first
- NDP all

ICDE 2011

Efficient XQuery Rewriting using Multiple Views
Rewriting scalability (QDF strategy)

![Graph showing the rewriting time for different query sizes. The graph compares flat queries, random queries, and linear queries, with the y-axis representing rewriting time in milliseconds and the x-axis representing query size (number of nodes).]
Outline

• Introduction
• Motivating example
• Tree pattern rewriting
• Joined pattern rewriting
• Experiments

➢ Conclusion
Conclusion

• XQuery view based rewriting
• Expressive query and view language
  – multiple returning nodes
  – value joins
• Equivalent, minimal and complete rewritings, expressed in a generic algebra
• Experimental evaluation
  – benefit of views
  – scalability of our approach
Related works

- N. Tang, J.X. Yu, M.T. Oszu, B. Choi and K.-F. Wong, “Multiple materialized view selection for XPath query rewriting,” in *ICDE, 2008*
Thank you!