Massively Multi-Query Join Processing in Publish/Subscribe Systems (SIGMOD'07)

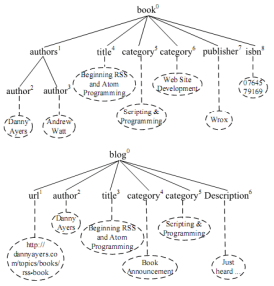
Ying Zheng
Instructor: Jun Yang
2008-03-18

Outline

- Motivation
- XSCL Query Language
- Two-Stage Query Processing
- Experiment
- Conclusion

Motivation

- Inter-document Queries in pub/sub systems
  - Different XML document joins based on values in their nodes, including attributes and text
  - E.g. Return a book announcement, followed by a blog article from one of its authors with the same title as the book.

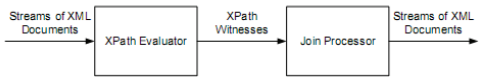


XSCL Query Language

- SELECT, FROM, PUBLISH
- XPath operator
- Variable binding construct
- Join operator (conjunctive predicates, window size)
  - JOIN
  - FOLLOWED BY
- Example

```
S//book->x1[.//author->x2][.//title->x3]
    FOLLOWED BY {x2=x5 AND x3=x6, T1}
S//blog->x4[.//author->x5][.//title->x6]
```

Two-Stage Query Processing

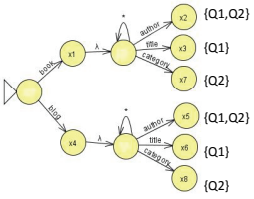


Query Example:

```
S//book->x1[.//author->x2][.//title->x3]
    FOLLOWED BY{x2=x5 AND x3=x6, T1}
S//blog->x4[.//author->x5][.//title->x6]
```

XPath Evaluation: YFilter

Q1	S//book->x1[.//author->x2][.//title->x3] FOLLOWED BY{x2=x5 AND x3=x6, T1} S//blog->x4[.//author->x5][.//title->x6]
Q2	S//book->x1[.//author->x2][.//category->x7] FOLLOWED BY{x2=x5 AND x7=x8, T2} S//blog->x4[.//author->x5][.//category->x8]



Xpath Evaluation: Generate DB

Query:

```
S//book->x1[.//author->x2][.//title->x3]
  FOLLOWED BY{x2=x5 AND x3=x6, T1}
S//blog->x4[.//author->x5][.//title->x6]
```

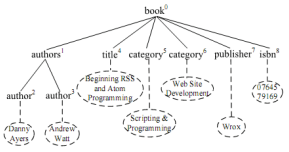


Table: R_{doc}

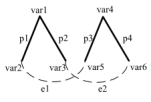
docid	node	strVal
d1	0	–
d1	2	Danny Ayers
d1	3	Andrew Watt
d1	4	Beginning RSS and Atom Programming
d1	5	Scripting & Programming
d1	6	Web Site Development

Table: R_{bin}

docid	var1	var2	node1	node2
d1	x1	x2	0	2
d1	x1	x2	0	3
d1	x1	x3	0	4
d1	x1	x7	0	5
d1	x1	x7	0	6

Query Template

Q1	S//book->x1[.//author->x2][.//title->x3] FOLLOWED BY{x2=x5 AND x3=x6, T1} S//blog->x4[.//author->x5][.//title->x6]
Q2	S//book->x1[.//author->x2][.//category->x7] FOLLOWED BY{x2=x5 AND x7=x8, T2} S//blog->x4[.//author->x5][.//category->x8]



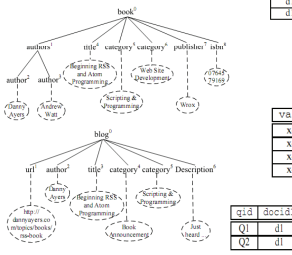
(a) R_T for Query Template $\mathcal{Q}$ in Figure 5

qid	var1	var2	var3	var4	var5	var6	w1
Q1	x1	x2	x3	x4	x5	x6	T1
Q2	x1	x2	x7	x4	x5	x8	T2

Join Processing & Optimization

Query:

```
S//book->x1[.//author->x2][.//title->x3]
  FOLLOWED BY{x2=x5 AND x3=x6, T1}
S//blog->x4[.//author->x5][.//title->x6]
```



(b) R_{doc} After Processing $d1$

docid	node	strVal
d1	0	–
d1	2	Danny Ayers
d1	3	Andrew Watt
d1	4	Beginning RSS and Atom Programming
d1	5	Scripting & Programming
d1	6	Web Site Development

(c) R_{bin} After Processing $d1$

docid	var1	var2	node1	node2
d1	x1	x2	0	2
d1	x1	x2	0	3

(d) R_{docW} of $d2$

node	strVal
0	–
2	Danny Ayers
	Programming
	ment
	mming

(e) R_{binW} of $d2$

var1	var2	node1	node2
x4	x5	0	2
x4	x6	0	3
x4	x8	0	4
x4	x8	0	5

(f) Content of R_{outT} After Processing $d2$

qid	docid1	node1	node2	node3	node4	node5	node6	w1
Q1	d1	0	2	4	0	2	3	T1
Q2	d1	0	2	5	0	2	5	T2

Experiment

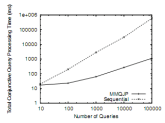


Figure 8: Performance on Simple Document Schema

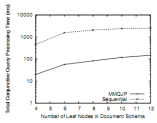


Figure 9: Performance on Simple Document Schema

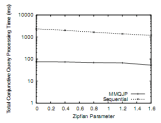


Figure 10: Performance on Simple Document Schema

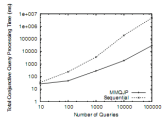


Figure 11: Performance on Complex Document Schema

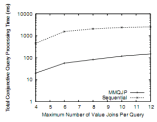


Figure 12: Performance on Complex Document Schema

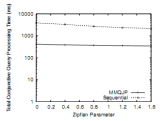


Figure 13: Performance on Complex Document Schema

Experiment

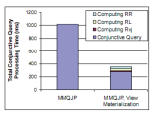


Figure 14: View Materialization on Simple Document Schema

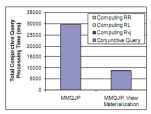


Figure 15: View Materialization on Complex Document Schema

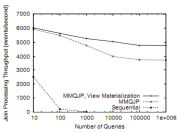


Figure 16: Performance on RSS Stream Processing

Conclusion

- Inter-document Queries in pub/sub systems
- XSCL Query Language
  - XML structure
  - Equivalence predicates for values on the leaves
- Two-stage Query Processing
  - XPath Evaluation
  - Join Processing
  - Optimization: View Materialization