Efficient Vectorial Operators for Processing XML Twig Queries

Guoliang Li, Jianhua Feng, Jianyong Wang, Feng Lin, and Lizhu Zhou
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, P. R. China
{liguoliang,fengjh,jianyong,dcszlz}@tsinghua.edu.cn; lin-f@mails.tsinghua.edu.cn

ABSTRACT

This paper proposes several vectorial operators for processing XML twig queries, which are easy to be performed and inherently efficient for both Ancestor-Descendant (A-D) and Parent-Child (P-C) relationships. We develop optimizations on the vectorial operators to improve the efficiency of answering twig queries in holistic. We propose an algorithm to answer GTP queries based on our vectorial operators.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Miscellaneous

General Terms

Algorithms, Performance, Languages

Keywords

Holistic Twig Join,Vectorial Operators,Subsequence Match

1. INTRODUCTION

Approaches based on structural index, numbering scheme and subsequence matching have been studied for processing XML twig queries, among which the most efficient one is holistic twig join based on numbering scheme. However, holistic twig join is suboptimal for those twig queries in which both Ancestor-Descendant (A-D) and Parent-Child (P-C) relationships are included, because it may involve huge even unbounded intermediate results. In addition, most of existing proposals are not efficient for the GTP queries [3, 4], because to answer GTP queries they have to compute the result sets of all the nodes although a part of them will not contribute to the answer, and hence they have to eliminate those redundancies involved in the result.

To address these problems, we present some vectorial operators for the P-C and A-D relationships to avoid redundant intermediate results and demonstrate how to answer twig queries using these vectorial operators efficiently. To accelerate the processing of twig queries, we propose several techniques to optimize these vectorial operators. Although $Twig^2Stack$[1] is proposed to process GTP queries, it is constrained by the fan-out of the XML document and thus leads to inefficiency. We discuss how to efficiently answer GTP queries according to our vectorial operators. To the best of our knowledge, this is the first paper that employs vectorial operators to process twig queries and GTP queries.

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

2. TJOPERATOR

We employ a sequence to represent an XML document and answer XML queries based on the sequence. For any node u, we construct list $\mathcal{I}_u$, which keeps the elements w.r.t. u and the positions of their occurrences in the corresponding sequence. To check whether element $e_a \in \mathcal{I}_a$ and element $e_d \in \mathcal{I}_d$ satisfy the A-D or P-C relationships, we transform $\mathcal{I}_a$ and $\mathcal{I}_d$ into bit-vectors and evaluate the A-D or P-C relationships on the bit-vectors.

2.1 Vectorial Operators

We begin by introducing three vectorial operators, λ_τ,λ_π and λ_ϖ, to transform any input list $\mathcal{I}_a$ into bit-vectors, $\mathcal{V}_{\tau_a}$,$\mathcal{V}_{\pi_a}$ and $\mathcal{V}_{\varpi_a}$, respectively as illustrated in Table 1. We propose three corresponding operators, τ, π, ϖ, which are similar to $\lambda_\tau, \lambda_\pi, \lambda_\varpi$ and operate on any bit-vector (e.g., an intermediate bit-vector) as shown in Table 1.

We propose two vectorial operators $\rho_{a[d]}$ and $\rho_{a/d}$ to evaluate the P-C relationship as shown in Table 1. The two operators can evaluate whether the elements in $\mathcal{I}_a$ and the elements in $\mathcal{I}_d$ satisfy a/d. In addition, we introduce three vectorial operators σ_a, $\rho_{a[//d]}$ and $\rho_{a//d}$, to evaluate the A-D relationship, as shown in Table 1. Equations (2.1-1)-(2.1-10) formally describe how to evaluate the P-C or A-D relationships, where $P[i]$ denotes the element at the i-th position in the sequence and $\mathcal{S}$ denotes the corresponding result set.

$$\mathcal{R}_{a[d]} = \rho_{a[d]}(\lambda_\tau(\mathcal{I}_a), \lambda_\pi(\mathcal{I}_d)) \quad (2.1\text{-}1)$$

$$\mathcal{R}_{a/d} = \rho_{a/d}(\lambda_\tau(\mathcal{I}_a), \lambda_\pi(\mathcal{I}_d)) \quad (2.1\text{-}2)$$

$$\mathcal{S}_{a[d]} = \{P[i] | \mathcal{R}_{a[d]}[i] = 1\} \quad (2.1\text{-}3)$$

$$\mathcal{S}_{a/d} = \{P[i] | \mathcal{R}_{a/d}[i] = 1\} \quad (2.1\text{-}4)$$

$$\mathcal{S}_{(a,/d)} = \{(P[i+1], P[i]) | \mathcal{R}_{a/d}[i] = 1\} \quad (2.1\text{-}5)$$

$$\mathcal{R}_{a//d} = \rho_{a//d}(\lambda_\varpi(\mathcal{I}_a), \lambda_\pi(\mathcal{I}_d)) \quad (2.1\text{-}6)$$

$$\mathcal{R}_{a[//d]} = \rho_{a[//d]}(\lambda_\pi(\mathcal{I}_a), \lambda_\pi(\mathcal{I}_d)) \quad (2.1\text{-}7)$$

$$\mathcal{S}_{a//d} = \{P[i] | \mathcal{R}_{a//d}[i] = 1\} \quad (2.1\text{-}8)$$

$$\mathcal{S}_{a[//d]} = \{P[i] | \mathcal{R}_{a[//d]}[i] = 1\} \quad (2.1\text{-}9)$$

$$\mathcal{S}_{(a,//d)} = \{(P[i], P[j]) | \forall i, j, \ \textit{if } \mathcal{R}_{a[//d]}[i]{=}1, \ P[i].\textit{leftmost} \le j < i, \mathcal{R}_{a//d}[j] = 1\}. \quad (2.1\text{-}10)$$

Table 1: XML Vectorial Operators for A-D and P-C relationships*

Operator	*Operand*	*Result*	*Definition*
$\lambda_\tau(\mathcal{I}_a)$	$\mathcal{I}_a$	$\mathcal{V}_{\tau_a}$	$\mathcal{V}_{\tau_a}[i]$=1 if $\exists j,k, 1\leq j\leq\lvert\mathcal{I}_a\rvert, 1\leq k\leq\lvert\mathcal{I}_a[j].occur\rvert$ and $i=\mathcal{I}_a[j].occur[k]$; otherwise, $\mathcal{V}_{\tau_a}[i]$=0.
$\lambda_\pi(\mathcal{I}_a)$	$\mathcal{I}_a$	$\mathcal{V}_{\pi_a}$	$\mathcal{V}_{\pi_a}[i]$=1 if $\exists j,k, 1\leq j\leq\lvert\mathcal{I}_a\rvert$, $k=\lvert\mathcal{I}_a[j].occur\rvert$ and $i=\mathcal{I}_a[j].occur[k]$; otherwise, $\mathcal{V}_{\pi_a}[i]$=0.
$\lambda_\varpi(\mathcal{I}_a)$	$\mathcal{I}_a$	$\mathcal{V}_{\varpi_a}$	$\mathcal{V}_{\varpi_a}[i]$=1 if $\exists j,k,t, 1\leq j\leq\lvert\mathcal{I}_a\rvert, k=\lvert\mathcal{I}_a[j].occur\rvert, t=\mathcal{I}_a[j].occur[k]$, $P[t].leftmost\leq i\leq t$; else $\mathcal{V}_{\varpi_a}[i]$=0.
$\tau(\mathcal{V}_a)$	$\mathcal{V}_a$	$\mathcal{V}_{\tau_a}$	$\mathcal{V}_{\tau_a}[i]$=1 if $\exists j, 1\leq j\leq\lvert\mathcal{V}_a\rvert$, $\mathcal{V}_a[j]$=1 and $P[i]=P[j]$; otherwise, $\mathcal{V}_{\tau_a}[i]$=0.
$\pi(\mathcal{V}_a)$	$\mathcal{V}_a$	$\mathcal{V}_{\pi_a}$	$\mathcal{V}_{\pi_a}[i]$=1 if $\exists j\leq i$, $\mathcal{V}_a[j]$=1 and $P[i]=P[j]$, but $\nexists k>i$, $P[i]=P[k]$; otherwise, $\mathcal{V}_{\pi_a}[i]$=0.
$\varpi(\mathcal{V}_a)$	$\mathcal{V}_a$	$\mathcal{V}_{\varpi_a}$	$\mathcal{V}_{\varpi_a}[i]$=1 if $\exists j, 1\leq j\leq\lvert\mathcal{V}_a\rvert$, $\pi(\mathcal{V}_a)[j]$=1 and $P[j].leftmost\leq i\leq j$; otherwise, $\mathcal{V}_{\varpi_a}[i]$=0.
$\rho_{a[d]}(\mathcal{V}_a,\mathcal{V}_d)$	$\mathcal{V}_a$,$\mathcal{V}_d$	$\mathcal{V}_{a[d]}$	$\mathcal{V}_{a[d]} = \pi(\ \tau(\mathcal{V}_a) \wedge (\pi(\mathcal{V}_d)>>1)\ )$
$\rho_{a/d}(\mathcal{V}_a,\mathcal{V}_d)$	$\mathcal{V}_a$,$\mathcal{V}_d$	$\mathcal{V}_{a/d}$	$\mathcal{V}_{a/d} = (\tau(\mathcal{V}_a)<<1) \wedge \pi(\mathcal{V}_d)$
$\sigma_a(\mathcal{V}_a,\mathcal{V}_d)$	$\mathcal{V}_a$,$\mathcal{V}_d$	$\mathcal{V}_{\sigma_a}$	$\mathcal{V}_{\sigma_a}[i]$=1 if $\pi(\mathcal{V}_a)[i]$ =1, $\exists j, P[i].leftmost\leq j<i$ and $\mathcal{V}_d[j]$=1; otherwise, $\mathcal{V}_{\sigma_a}[i]$=0.
$\rho_{a//d}(\mathcal{V}_a,\mathcal{V}_d)$	$\mathcal{V}_a$,$\mathcal{V}_d$	$\mathcal{V}_{a//d}$	$\mathcal{V}_{a//d} = \varpi(\mathcal{V}_a) \wedge \pi(\mathcal{V}_d)$
$\rho_{a[//d]}(\mathcal{V}_a,\mathcal{V}_d)$	$\mathcal{V}_a$,$\mathcal{V}_d$	$\mathcal{V}_{a[//d]}$	$\mathcal{V}_{a[//d]} = \pi(\ \sigma_a(\ \pi(\mathcal{V}_a),\ \varpi(\mathcal{V}_a) \wedge \pi(\mathcal{V}_d)\ )\ )$

2.2 Vectorial Operators for Twig Queries

We introduce how to answer twig queries according to our operators. Given a query $\mathcal{Q}$, let $\mathcal{R}_n$ denote the bit-vector result of node n w.r.t. the sub-query rooted at node n, while let $\mathcal{F}_n$ denote the bit-vector result of node n w.r.t. $\mathcal{Q}$. We first compute $\mathcal{R}_n$ from the leaf to the root and then compute $\mathcal{F}_n$ from the root to the leaf. Thus, we can get the result set of any query node according to Equations (2.2-1)-(2.2-5).

$$\mathcal{R}_n^c = \tau(\wedge_{c\in c\text{-}children(n)}\tau(\lambda_\tau(\mathcal{I}_n)\wedge(\mathcal{R}_c>>1))) \quad (2.2\text{-}1)$$

$$\mathcal{R}_n^d = \tau(\wedge_{d\in d\text{-}children(n)}\tau(\sigma_n(\lambda_\pi(\mathcal{I}_n),\lambda_\varpi(\mathcal{I}_n)\wedge\mathcal{R}_d))) \quad (2.2\text{-}2)$$

$$\mathcal{R}_n = \begin{cases} \lambda_\pi(\mathcal{I}_n) & \textit{if } n \textit{ is a leaf node} \\ \pi(\mathcal{R}_n^c) & \textit{if } n \textit{ only has c-child} \\ \pi(\mathcal{R}_n^d) & \textit{if } n \textit{ only has d-child} \\ \pi(\mathcal{R}_n^c\wedge\mathcal{R}_n^d) & \textit{otherwise} \end{cases} \quad (2.2\text{-}3)$$

$$\mathcal{F}_n = \begin{cases} \mathcal{R}_n & \textit{if } n \textit{ is the root node} \\ (\tau(\mathcal{F}_p)<<1)\wedge\mathcal{R}_n & \textit{if } n \textit{ is c-child of its parent } p \\ \varpi(\mathcal{F}_p)\wedge\mathcal{R}_n & \textit{if } n \textit{ is d-child of its parent } p \end{cases} \quad (2.2\text{-}4)$$

$$\mathcal{S}_n = \{P[i]\lvert\mathcal{F}_n[i]=1\} \quad (2.2\text{-}5)$$

2.3 Optimizations for Vectorial Operators

We introduce several techniques to optimize our vectorial operators. Let c_1,c_2,...,c_l be the c-children (P-C) of node n and d_1,d_2,...,d_k be the d-children (A-D), to optimize the computation of $\mathcal{R}_n^d$, we extend σ_a from a binary operator to a multiple operator σ_n:

$\mathcal{V}_{\sigma_n}=\sigma_n(\mathcal{V}_n,\mathcal{V}_{d_1},\mathcal{V}_{d_2},...,\mathcal{V}_{d_k})\models\mathcal{V}_{\sigma_n}[i]$=1, if $\pi(\mathcal{V}_n)[i]$=1,$\forall 1\leq j\leq k$, $\exists b_j$, $P[i].leftmost\leq b_j<i$ and $\mathcal{V}_{d_j}[b_j]$=1; otherwise, $\mathcal{V}_{\sigma_n}[i]$=0.

We introduce the following Equations (2.3-1)-(2.3-4) to optimize the construction of $\mathcal{R}_n^c$, $\mathcal{R}_n^d$, $\mathcal{R}_n$ and $\mathcal{F}_n$.

$$\mathcal{R}_n^c \equiv \tau(\tau(\tau(\tau(\lambda_\tau(\mathcal{I}_n)\wedge(\mathcal{R}_{c_1}>>1))\wedge(\mathcal{R}_{c_2}>>1))\wedge...)\wedge(\mathcal{R}_{c_l}>>1)) \quad (2.3\text{-}1)$$

$$\mathcal{R}_n^d \equiv \tau(\sigma_n(\lambda_\pi(\mathcal{I}_n),\lambda_\varpi(\mathcal{I}_n)\wedge\mathcal{R}_{d_1},...,\lambda_\varpi(\mathcal{I}_n)\wedge\mathcal{R}_{d_k})) \quad (2.3\text{-}2)$$

$$\pi(\mathcal{R}_n^c\wedge\mathcal{R}_n^d) \equiv \pi(\mathcal{R}_n^c)\wedge\pi(\mathcal{R}_n^d) \quad (2.3\text{-}3)$$

$$\pi(\mathcal{R}_n^d) \equiv \sigma_n(\lambda_\pi(\mathcal{I}_n),\lambda_\varpi(\mathcal{I}_n)\wedge\mathcal{R}_{d_1},...,\lambda_\varpi(\mathcal{I}_n)\wedge\mathcal{R}_{d_k}) \quad (2.3\text{-}4)$$

To answer twig queries in holistic and process GTP queries effectively, we propose an effective algorithm $TJOperator$ by employing the vectorial operators. $TJOperator$ directly computes the results and does not require a post-processing to eliminate the irrelevant elements. $TJOperator$ only maintains several bit-vectors and the number of these bit-vectors is no more than the number of result nodes, thus $TJOperator$ will not involve large intermediate results.

*$occur$ keeps the positions of e's occurrences in the sequence and $leftmost$ is the position of the occurrence of e's leftmost descendant.

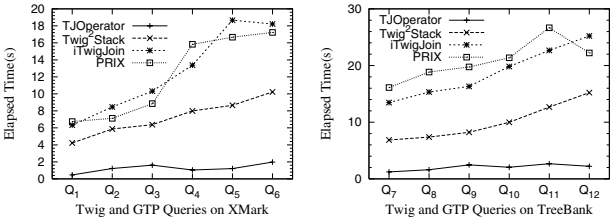

Figure 1: Experimental results

3. EXPERIMENTAL STUDY

We compared $TJOperator$ with state-of-the-art methods, PRIX [5], iTwigJoin [2] and Twig^{2}Stack [1]. All the algorithms were coded in C++ and all the experiments were conducted on a 2.4 GHz Pentium IV processor with 1GB RAM, running Microsoft Windows XP. We used the real-world dataset DBLP and the dataset TreeBank with deep recursive structures, and employed twelve queries for our experiments. Figure 1 shows the experimental results. We can observe that $TJOperator$ outperforms PRIX, iTwigJoin and Twig^{2}Stack on various queries and datasets significantly.

4. CONCLUSION

We propose several vectorial operators to evaluate the A-D and P-C relationships and present how to process twig queries in holistic by employing these operators. We demonstrate several effective techniques to optimize these operators for processing XML twig queries and GTP queries.

5. ACKNOWLEDGEMENT

This work is partly supported by the National Natural Science Foundation of China under Grant No.60573094, the National High Technology Development 863 Program of China under Grant No.2007AA01Z152 and 2006AA01A101, the National Grand Fundamental Research 973 Program of China under Grant No.2006CB303103.

6. REFERENCES

[1] S. Chen, H.-G. Li, J. Tatemura, W.-P. Hsiung, D. Agrawal, and K. S. Candan. Twig2stack: Bottom-up processing of generalized-tree-pattern queries over xml documents. In *VLDB*, 2006.

[2] T. Chen, J. Lu, and T. Ling. On boosting holism in xml twig pattern matching using structural indexing techniques. In *SIGMOD*, pages 455–466, 2005.

[3] Z. Chen, H. V. Jagadish, L. V. S. Lakshmanan, and S. Paparizos. From tree patterns to generalized tree patterns: On efficient evaluation of xquery. In *VLDB*, pages 237–248, 2003.

[4] G. Li, J. Feng, and L. Zhou. Efficient Holistic Twig Joins in Leaf-to-Root Combining with Root-to-Leaf Way. In *DASFAA*, 2007.

[5] P. Rao and B. Moon. Prix: Indexing and querying xml using prufer sequences. In *ICDE*, pages 288–299, 2004.