A Self-adaptive Scope Allocation Scheme for Labeling Dynamic XML Documents

Yun Shen[1], Ling Feng[2], Tao Shen[3], and Bing Wang[1]

[1] Department of Computer Science, University of Hull
Hull, HU6 7RX, United Kingdom
{Y.Shen, B.Wang}@dcs.hull.ac.uk
[2] Department of Computer Science, University of Twente
PO Box 217, 7500 Enschede, The Netherlands
{ling}@cs.utwente.nl
[3] School of Management, UESTC, 610054,P.R.China

Abstract. This paper proposes a self-adaptive scope allocation scheme for labeling dynamic XML documents. It is general, light-weight and can be built upon existing data retrieval mechanisms. Bayesian inference is used to compute the actual scope allocated for labeling a certain node based on both the prior information and the actual document. Through extensive experiments, we show that the proposed Bayesian allocation model can practically and significantly improve the performance of the conventional fixed scope allocation models.

1 Introduction

It is increasingly expected that XML [1] will become the de facto standard of the Web, ultimately replacing HTML. An XML document consists of data enclosed by user defined tags, and its nested tree structure is described by DTD. Figure. 1 shows an example of an XML document and figure. 2 illustrates the corresponding DTD.

To allow efficient querying of XML data, each node in the XML data trees is typically given a unique label, such that given the labels of two nodes, we can determine whether one node is an ancestor of the other. Till now, many indexing structures have been proposed to process structural queries efficiently based on certain coding schemes [9,13,4]. [9] "stitches" the binary twigs to obtain final results based on range labeling scheme. [4] improves [9] by using a stack-based algorithm to efficiently join root-to-leaf paths. [13] treats both document and query as sequences, and matches the query as a whole unit when querying.

However, to fully evolve XML into a universal data representation and exchange format, the capability of modifying XML document is indispensable. It thus arises an important aspect to XML indexing: how to support dynamic data insertion, deletion, and update with corresponding index structure.

Cohen et al. [6] firstly proposed a dynamic labeling scheme for XML documents to support updating operations. The children of a node v have the label concatenated with the string s attached to their incoming edge. Given s(1)=0, to obtain s(i+1), the binary number represented by s(i) is increased 1 and if the

F. Galindo et al. (Eds.): DEXA 2004, LNCS 3180, pp. 811–821, 2004.

```
<prices>
  <book>
    <title>Algorithms</title>
    <information>
      <source>bstore2.example.com</source>
      <price>31.99</price>
    </information>
  </book>
  <book>
    <title>Data on the Web</title>
    <information>
      <source>bstore2.example.com</source>
      <price>34.95</price>
    </information>
    <author>Serge Abiteboul</author>
    <author>Peter Buneman</author>
    <author>Dan Suciu</author>
  </book>
  <book>
    <title>TCP/IP Illustrated</title>
    <information>
      <source>bstore2.example.com</source>
      <price>65.95</price>
    </information>
    <author>W. Richard Stevens</author>
  </book>
</prices>
```

Fig. 1. Example of an XML Document

```
<!ELEMENT prices (book*)>
<!ELEMENT book (title?, information+, author*)>
<!ELEMENT information (source, price)>
<!ELEMENT author (#PCDATA)>
<!ELEMENT price (#PCDATA)>
<!ELEMENT source (#PCDATA)>
<!ELEMENT title (#PCDATA)>
```

Fig. 2. Example of DTD

representation of s(i)+1 consists of all ones, its length is double. The problem with this approach is that the size of labeling will grow fast with the increase of degree of the nodes for its the dependency on the fan-out of a node.

State-of-the-art research efforts [13,14,9] have been proposed to cope with the problem of allocating scope for dynamic XML documents. [9,14] considered to give some extra scope while allocating the labels. But as for how much to allocate, they did not address it. [13] considered to allocate scope for each node on the basis of the probability of its occurrences from statistical information. The deficiency is that the probability of allocating space for a certain node is fixed and is considered as a constant. To summarize, these methods are not self-adaptive to real document environment where the probability of a node's occurrence varies in difference XML documents.

In this paper, we improve the conventional scope allocation scheme using Bayesian inference technique. Combining the prior information (i.e DTD and statistical information) and the actual documents, better performance in scope allocation for dynamic XML documents is achieved. we propose Bayesian Allocation Model (BAM), a general, self-adaptive scope allocation model that addresses the above important challenges for dynamic XML data. It is general, light-weight and can be adapted to existing data retrieval mechanisms [5,4,8,16,7,13,9]. The scope allocated for each node depends on the probability it will be updated. Thus we can guarantee a better performance than the traditional allocation methods when updating.

Outline of the Paper. The rest of the paper is organized as follows. Our proposed method, Bayesian allocation model, is introduced and theoretically proved in section 2. The corresponding algorithms are shown in section 3. The performance is demonstrated in section 4 through a set of experiments. Finally, our work is concluded in Section 5.

2 Bayesian Allocation Model

Notations: Let u be a node in a DTD, which has t child nodes. Let $nodetype_i$ denote the type of the i*th* child node ($1 \leq i \leq t$), which occurs x_i times under the node u in corresponding XML document. $\sum_{i=1}^{t} x_i$ = z, where z equals to the total number of children under node u. Assume all sibling nodes of different $nodetype_i$ occur independently with probability θ_i. Let $\bar{\theta}_i$ denote estimators of θ_i, which can be obtained from semantic of our XML document or the statistics of a sample dataset. Let n denote the *range* scope allocated for the z nodes under node u. In the paper, n = c $\times$ z, where c denotes the *range enlarging* factor.

2.1 Scope Allocation Overview

The scope allocation scheme works as follows; 1) parse DTD to obtain prior information for each name type; then during the breadth first traversal of an XML document, embed the tree into a complete K-ary tree, 2) root node of an XML document is put in level 0 and labeled 1 in the complete K-ary tree; 3) for

each non-leaf node, calculate the number of children, denoted as z, and the types of its children, denoted as t; 4) allocate scope for each child node using Bayesian allocation model in l*th* level below its parent, satisfying $l \geq \lceil \log_k(c \times z) \rceil$, c denotes a range enlarging factor; 5) repeat 3) and 4) till breadth-first traversal finishes.

2.2 Bayesian Allocation Model

Self-adaptive Bayesian allocation model is proposed to allocate scope for dynamic XML documents on the basis of K-ary tree and Breadth-First Traversal. Pre-allocating scope for dynamic XML data is a natural solution. The core of Bayesian allocation model is on efficiently allocating scope for each node in actual dynamic XML documents in a self-adaptive manner.

Bayesian Allocation Model. The core of our work is on estimating probability θ_i. In ViST [13], Haixun Wang et al. calculate θ_i only from available DTD or statistics from sample set of XML documents, and consider θ_i as a constant, which is fixed without considering the actual documents. Our objective is self-adaptively allocating dynamic scope for each node according to its actual probability in each document, not just using a fixed constant probability, which is simply calculated from DTD or sample set of datasets.

Our proposed Bayesian allocation model considers θ_i as a random variable not a constant, and chooses a proper prior distribution for θ_i, which reflects the observer's beliefs about the proper value θ_i may take, and then updates these beliefs on the basis of the actual data observed. Thus estimators of θ_i can accord with the real world data in a self-adaptive manner. To summarize, the heuristics guiding the allocation model is that the more child nodes of $nodetype_i$ a node u has, the more likely for these child nodes being updated.

Given a node u with t children in a DTD. Each of them occurs x_i times, i = (1, ..., t), in the corresponding XML document. x_i may be zero in an actual XML document if a certain node is optional in DTD. Assume 1) all sibling nodes of different $nodetype_i$ occur independently with probability θ_i (i=1, ..., t) and 2) the probability of data insertion/deletion on these nodes occurs according to the same probability. Thus, given scope range n for the z nodes in an XML document, if we know θ_i (i = 1, ..., t) for the z nodes, a natural idea is that we allocate scope for each node type according to probability θ_i. For instance, if all the sibling nodes with $nodetype_i$ occur x_i times under a node u, and update probability is θ_i, then we allocate $\frac{n\theta_i}{x_i}$ for each node with $nodetype_i$.

In general our Bayesian allocation model is based on the following hypothesis below:

- Sibling nodes of different $nodetype_i$ occur independently with probability θ_i (i=1, 2, ..., t), where t denotes the node name types in the correspond DTD. And all data insertion/deletion/update on the nodes of different $nodetype_i$ occurs independently with the same probability θ_i.
- θ_i is a random variable ,we use $\pi(\theta_i)$ to denotes prior distribution of θ_i , which reflects our beliefs about what values parameters θ_i may take. We assume that θ_i is a $beta(\alpha_i, \beta_i)$ distribution

Whether or not using prior distribution in statistical inference is not only a problem of mathematical technic but a problem of philosophy [3] as well. Thus we do not discuss necessity of using prior distribution here, and use it by the way of Bayesian inference theory. However, how to choose prior distribution is another problem. We choose beta distribution as prior distribution because: 1) density curve of beta distribution is plane curve when $\alpha \gg 1, \beta > 1$, and $0 \leq$ value of beta distribution ≤ 1, matching the definition of probability. Thus we consider $\bar{\theta}_i$, prior information of θ_i, as mean value of beta distribution, which means that the probability of $node_i$ occurs around $\bar{\theta}_i$ is greater than in other zone, matching our hypothesis; 2) from lemma 1 we know posterior distribution $\theta_i|x_i$ is also a beta distribution which is convenient to compute the posterior estimators of θ_i. Using other prior like norm distribution will result in complicated monte carlo simulations and the computational complexity. In fact, using beta distribution as prior distribution of parameter θ_i in binomial distribution is very common in practice [3].

Theoretical Proof. In our model we consider sample information of θ_i as occurrence times x_i under a certain node u since we assume a node with $nodetype_i$ occurs with same probability θ_i in our hypotheses. From the Hypotheses x_i observes binomial distribution, denoted as $b(z, \theta_i)$. The updating procedure is performed using Bayesian theorem, which states that the posterior distribution $\theta_i|x_i$, representing our beliefs on the observed data, is proportional to the product of the prior distribution and the likelihood function.

The following two lemmas are proved for the correctness of our Bayesian allocation model:

Lemma 1 *Assume $\pi(\theta_i)$ is beta(α_i, β_i) ($\alpha_i \gg 1, \beta_i > 1$), and sample information variable $x_i \sim b(z, \theta_i)$ (binomial distribution) with parameter θ_i. Thus, the posterior distribution function $p(\theta_i|x_i)$ is also a beta(α_i^*, β_i^*) distribution, and beta(α_i, β_i) is called conjugate prior distribution. We have the following result:*

$$E(\theta_i|x_i) = \lambda_i E(x_i|\theta_i) + (1 - \lambda_i)\frac{x_i}{z}$$

where

$$E(x_i|\theta_i) = \frac{\alpha_i}{\alpha_i + \beta_i}$$

Proof: Given

$$\pi(\theta_i) \sim beta(\alpha_i, \beta_i)$$

and it's density function

$$\pi(\theta_i) = \frac{\Gamma(\alpha_i + \beta_i)}{\Gamma(\alpha_i)\Gamma(\beta_i)}\theta_i^{\alpha_i - 1}(1 - \theta_i)^{\beta_i - 1}$$

because x_i is binomial distribution, density function of x_i given parameter θ_i is:

$$f(x_i|\theta_i) = C_z^{x_i}\theta_i^{x_i}(1 - \theta_i)^{z - x_i}, where 0 \leq \theta_i \leq 1$$

Thus, according to Bayesian theorem posterior distribution of θ_i is:

$$p(\theta_i|x_i) = \frac{f(x_i|\theta_i)\pi(\theta_i)}{\int_0^1 f(x_i|\theta_i)\pi(\theta_i)d\theta_i}$$
$$= \frac{\theta_i^{\alpha_i+x_i-1}(1-\theta_i)^{z+\beta_i-x_i-1}}{\int_0^1 \theta_i^{\alpha_i+x_i-1}(1-\theta_i)^{z+\beta_i-x_i-1}d\theta_i}$$
$$= k_i\theta_i^{x_i+\alpha_i-1}(1-\theta_i)^{z+\beta_i-x_i-1})$$

where

$$k_i = \frac{\Gamma(\alpha_i+x_i)\Gamma(\beta_i+z-x_i)}{\Gamma(\alpha_i+\beta_i+z)}$$

hence:

$$p(\theta_i|x_i) \sim beta(\alpha_i+x_i, \beta_i+z-x_i)$$

and:

$$E(\theta_i|x_i) = \frac{\alpha_i+x_i}{\alpha_i+\beta_i+z} = \frac{\alpha_i}{\alpha_i+\beta_i}\lambda_i + (1-\lambda_i)\frac{x_i}{z}$$
$$= E(\theta_i)\lambda_i + (1-\lambda_i)\frac{x_i}{z}$$

where

$$\lambda_i = \frac{\alpha_i+\beta_i}{\alpha_i+\beta_i+z}$$

λ_i reflects on the importance balance between prior information and sample information.

Lemma 2 *Assume prior distribution $\pi(\theta_i)$, sample information x_i and square loss function $L(\delta_i, \theta_i)$ is given. The Bayesian estimators of θ_i, $\delta^\pi(x_i)$, is the expectation (or mean value) of posterior distribution $\pi(\theta_i|x_i)$ which is written as:*

$$\delta^\pi(x_i) = E(\theta_i|x_i)$$

Proof: The posterior risk of any decision function $\delta_i = \delta(x_i)$ given square loss function is :

$$E((\delta_i-\theta_i)^2|x_i) = \delta_i^2 - 2\delta_i E(\theta_i|x_i) + E(\theta_i^2|x_i)$$

obviously the function value is minimized *iff*:

$$\delta_i = E(\theta_i|x_i)$$

From Lemma 1 we know that

$$\delta_i = \frac{\alpha_i+x_i}{\alpha_i+\beta_i+z} = E(\theta_i)\lambda_i + (1-\lambda_i)\frac{x_i}{z}$$

In our model we assume that $E(\theta_i) = \bar{\theta_i}$ which means that the prior probability of node i occurs around $\bar{\theta_i}$ is greater than in other zone. And from lemma 1 we

get posterior expectation of θ_i .So we use $E(\theta_i|x_i)$ as estimators of θ_i according to Lemma 2.

Let $\lambda_i = 0.5$, which implies that importance of prior information is the same as that of sample information. Let $\theta_i^{'}$ denote estimators of θ_i given prior information $\bar{\theta_i}$. From Lemma 1 and Lemma 2, we have:

$$\theta_i^{'} = \frac{1}{2}\bar{\theta_i} + \frac{1}{2} * \frac{x_i}{z} \tag{1}$$

Eq(1) proves that the prior information and posterior information both contribute to the final probability a node will be updated, which is better than ViST allocation method which only utilizes prior information only.

3 Algorithms

We call the above process of allocating scope in a complete K-ary tree with Bayesian allocation model *Bayesianization*. After a brief description of how to compute the prior information from DTD, we present Algorithm BAYESIANIZATION and Algorithm BAYESINFERENCE which describe the labeling process that combines with the Bayesian theorem in detail. Figure. 3 gives an example of our proposed Bayesian allocation model. We show how to compute these p_i (i=1, 2, 3, ...) in the following section.

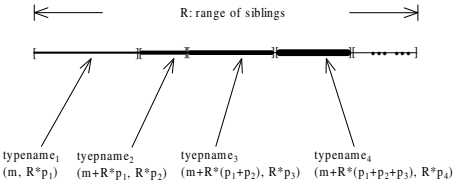

Fig. 3. Example of Bayesian Allocation Model

3.1 Prior Information

Prior information about the occurrence probability of all the children below a node u in DTD, denoted by $\bar{p}_{DTD_u}$, is defined as follows:

$$\bar{p}_{DTD_u} = (\bar{p}_{c_1}, \bar{p}_{c_2}, ..., \bar{p}_{c_t})$$

where t is the number of different child nodes u has. Each $\bar{p}_{c_i}$, i = 1, 2, ...,t, can be computed based DTD Table.1, which defines the proportion of the scope among the different cardinality.

For example, the value 0.08 at (?, +) in Table 1 specifies the proportion relationship 0.08:1 between child node type with "?" and child node type with "+". Also the proportion 1.25:1 between child node type with "+" and the one

with "*". Thus we have 0.08:1:0.8 having three child nodes with type "?", "+" and "*". Notice that the data in Table. 1 can be calculated from the statistic information of the sample XML documents.

Actually, table.1 reflects our belief on prior information. (a|b) is transformed into (a,b) to minimize the computational complexity. Consider, for example, given <!ELEMENT book (title?, information+, author*)>, the proportion among these three nodetypes is 0.08:1:0.8, to normalize, $\bar{p}_{title?} = \frac{0.08}{0.08+1+0.8} = 0.043$, $\bar{p}_{information+} = \frac{1}{0.08+1+0.8} = 0.532$ and $\bar{p}_{author*} = \frac{0.8}{0.08+1+0.8} = 0.425$. Therefore, $\bar{p}_{DTD_{book}} = (0.043, 0.532, 0.425)$.

Table 1. Prior information on the cardinality proportion

	*	+	?
*	1	0.8	10
+	1.25	1	12.5
?	0.1	0.08	1

3.2 Algorithms

As we can see in Algorithm BAYESIANIZATION and Algorithm BAYESINFERENCE, our specific Bayesian allocation model is light-weight and self-adaptive for each node in the XML document tree (Line 1 - 11 in Algo.BAYESINFERENCE). The time complexity of the algorithms is O(n), depending on the number of the nodes in a tree, and the space complexity is linear as well. It implies that our algorithm guarantees both time and space complexity efficiency while allocating self-adaptive scope for each node, which is not provided by the previous methods. The performance results are shown in section 4.

Algorithm BAYESIANIZATION
Input: T: Data tree of XML document; Queue: queue of nodes; p_{DTD}: the DTD prior distribution generated
1.
Output: BAM Allocated Document
2. Queue.insert(T.root)
3. **while** (!Queue.empty())
4. **do** u ← Queue.fetch()
5. list ← listofchildren(u)
6. z ← numofchildren(list)
7. t ← typeofchildren(list)
8. BAYESINFERENCE($\bar{p}_{DTD_u}$, z, t, list)
9. Queue.insert(list)

Algorithm BAYESINFERENCE
Input: $\bar{p}_{DTD_u}$: prior information of node u; z: number of child nodes; t: number of child node types, list: list of nodes

Output: BAM allocated scope of u
1. level = $\lceil \log_k(c \times z) \rceil$
2. n = k^{level}
3. **for** i ← 1 to t
4. **do** $p_i \leftarrow \frac{\bar{p}_{DTD_{u_i}} + x_i/z}{2}$
5. $subrange_i \leftarrow$ n × p_i
6. **for** j ← 1 to z in list
7. **do** $subrange_{node_j} \leftarrow subrange_i$ / t_i
8. $seqnum_{node_j} \leftarrow (\sum_1^{i-1} subrange_i + \sum_1^{j-1} subrange_{node_j})$

Consider, for example, given <!ELEMENT book (title?, information+, author *)>, we get $p_{DTD_{book}} = (0.043, 0.532, 0.425)$. If in an actual XML document, a node named "book" has 1 "title" child node, 2 "information" child nodes, however, 10 "author" child nodes. Suppose the range enlarging factor is 100, thus we allocate scope 13*100 = 1300 (n = c × z, section 2) for these 13 child nodes. Thus their actual probability should be <0.0599 = ((0.043 + $\frac{1}{13}$)/2), 0.4198 = ((0.532 + $\frac{2}{13}$)/2), 0.5971 = ((0.425 + $\frac{10}{13}$)/2) >, and the scope allocated for each node are 0.0599*1300 = 77 for "title" node, 0.4198 * 1300 = 545 for "information" nodes, and 0.5971 * 1300 = 776 for "author" nodes. We notice that the allocation scopes of these three nodes accord to their actual occurrence probability.

4 Performance Experiments

4.1 Experiments

The experiments were conducted on a PC with Pentium III CPU 1.2GHZ and 512 MB main memory. We implemented the proposed method Bayesian allocation model and the conventional fixed allocation scheme for comparison purpose. Our synthetic XML documents are generated using Sun XML instance generator [12] utilizing various DTDs, i.e. ACM Sigmod Record DTD [2], and public XML benchmark database XMARK [15].

Update Performance. We focus on studying the update performance in dynamic XML documents. We generate 200 XML documents with maximum depth 8 for experiments. Five different experiments are performed with different range enlarging factors. In the experiments, we respectively set the range enlarging factor $c = 50, 75, 100, 125$ and 150 for these documents in the set of experiments. We then implement an allocation scheme similar to ViST and our allocation scheme tailored to ViST. In ViST, it allocates scope for each type of child nodes directly from DTD without constructing trie-like tree. However, for comparison purpose, we first construct the trie-like tree physically, and then apply BAM to allocate scope for each node in the trie-like tree.

In the experiments, we randomly choose m nodes the datasets we generated. Suppose each $node_i$ (i = 1, ..., m) has t different node name types. Firstly we compute the prior information for each $node_i$ from the corresponding DTD, denoted as $\bar{p}_{DTD_{node_i}} = (\bar{p}_1, \bar{p}_2, ..., \bar{p}_t)$. Then we use t independent beta distribution to generate t random numbers, denoted as $(r_1, r_2, ..., r_t)$. We can

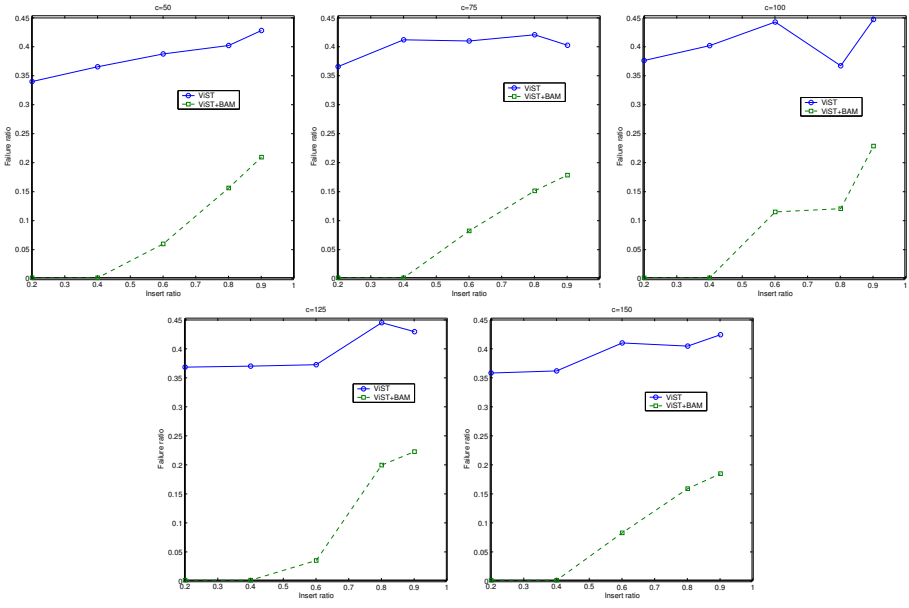

Fig. 4. Insert ratio vs. failure ratio

prove that $0 \leq r_1, r_2, ..., r_t < 1$. Thirdly we generate the insertion/deletion probability for each node type: $p_i = \frac{r_i}{s}$, where s $= \sum_{i=1}^{t} r_i$, which obeys our hypotheses and is fair to both ViST and ViST with BAM when the probability of insertion/deletion is concerned. Finally we random generate the position a node should be inserted/deleted.

We define that a "failure" occurs when a position has been allocated during inserting, and an "overflow" when pre-allocated space for a certain node is used up. During the experiments, we record the "failure"times. eq(2) and eq(3) are presented to clarify the experimental results shown in figure 4.

$$failure_{ratio} = \begin{cases} 1, & \text{when "overflow" occurs,} \\ \frac{times_{failure}}{times_{insert}}, & \text{otherwise.} \end{cases} \tag{2}$$

$$insert_{ratio} = \frac{times_{insert}}{Space_{free}} \tag{3}$$

We can see that BAM improves at least 49.34% comparing to the conventional ViST method, for the scope allocated for each node accords to the probability it would be updated, which further depends not only on the prior information, i.e. statistical information from sample datasets, but combining the actual probability of its occurrence as well. Especially, when the probability of inserting/deleting a node in an XML document is much greater than the average prior information, BAM performs much more better than ViST method.

5 Conclusion

In the paper, we propose a general self-adaptive labeling scheme for dynamic XML documents. We show that Bayesian allocation model can easily be adapted to the state-of-the-art data retrieval methods to provide support for dynamic XML documents. Through the experiments, we demonstrate that it can efficiently support updating for the dynamic XML documents, at least 49.34% better than the conventional methods, while not affecting their original query speed.

References

1. S.Abiteboul, P.Buneman, D.Suciu, Data on the web: from relations to semistructured data and XML, Morgan Kaufmann Publishers, San Francisco, California, 2000.
2. ACM Sigmod Record DTD, `http://www.dia.uniroma3.it/Araneus/Sigmod/`.
3. J.O. Berger, Statistical decision theory and bayesian analysis, Springer-Verlag, 1985.
4. N. Bruno, N. Koudas, D. Srivastava, Holistic twig joins: optimal XML pattern matching, in: Proc. ACM SIGMOD, 2002, pp. 310 - 321.
5. S.-Y. Chien, Z. Vagena, D. Zhang, V. Tsotras, C. Zaniolo, Effiecient structural joins on indexed XML documents, in: Proc 28th VLDB conference, 2002, pp. 263 - 274.
6. E. Cohen, H. Kaplan, T. Milo, Labeling dynamic XML trees, in: Proc 21st PODS conference, 2002, pp. 271 - 281.
7. T. Grust, Accelerating XPath location steps, in: Proc. ACM SIGMOD, 2002, pp. 109 - 120.
8. D. D. Kha, M. Yoshikawa, S. Uemura, An XML indexing structure with relative region coordinates, in: Proc. 17th ICDE, 2001, pp. 313 - 320.
9. Q.Li, B. Moon, Indexing and querying XML data for regular path expressions, in: Proc. 27th VLDB conference, 2001, pp. 361 - 370.
10. L.Mignet, D.Barbosa, P.Veltri, The XML web: a first study, in: Proc. 12th WWW, 2003, pp. 500–510.
11. Sleepycat Software, `http://www.sleepycat.com`. The Berkeley Database (Berkeley DB).
12. Sun XML instance generator, available at (2003): `http://wwws.sun.com/software/xml/developers/instancegenerator/`.
13. H.X. Wang, S. Park, W. Fan, P.S. Yu, ViST: a dynamic index method for querying XML data by tree structures, in: Proc. SIGMOD, 2003, pp. 110 - 121.
14. W. Wang, H. Jiang, H.J. Lu, J.X. Yu, PBiTree coding and efficient processing of containment joins, in: Proc. 19th ICDE Conference, 2003, pp. 391 - 402.
15. XMARK benchmark, `http://monetdb.cwi.nl/xml/index.html`.
16. C. Zhang, J. Naughton, D. Dewitt, P. Luo, G. Lohman, On supporting containment queries in relational database management systems, in: Proc. ACM SIGMOD, 2001, pp. 425 - 436.