Estimating the Selectivity of XML Path Expression with predicates by Histograms *

Yu Wang[1], Haixun Wang[2], Xiaofeng Meng[1], and Shan Wang[1]

[1] Information School, Renmin University of China, Beijing 100872, PRC
{wangyu6,xfmeng,swang}@mail.ruc.edu.cn
[2] IBM Thomas J. Watson Research Center Hawthorne, NY 10532
haixun@us.ibm.com

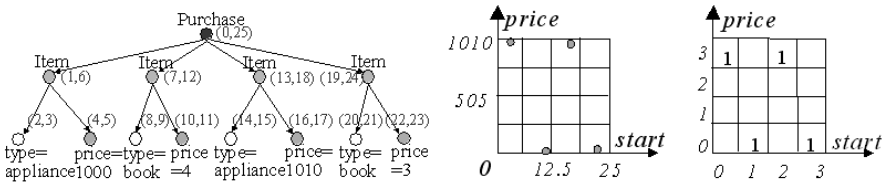
Abstract. Selectivity estimation of path expressions in querying XML data plays an important role in query optimization. A path expression may contain multiple branches with predicates, each of which having its impact on the selectivity of the entire query. In this paper, we propose a novel method based on 2-dimensional value histograms to estimate the selectivity of path expressions embedded with predicates. The value histograms capture the correlation between the structures and the values in the XML data. We define a set of operations on the value histograms as well as on the traditional histograms that capture nodes positional distribution. We then construct a cost tree based on such operations. The selectivity of any node (or branch) in a path expression can be estimated by executing the cost tree. Compared with previous methods (which ignore value distribution) our method offers much better estimation accuracy.

1 Introduction

As XML becomes the standard for data exchanging over the Internet, much research has been devoted to the efficient support of XML queries. We study XML query optimization based on selectivity estimation. If the branch with less selectivity is queried first, the intermediate result will have a relatively small size and the query efficiency can be improved. XML data is a hybrid of structures and values, some of which are highly correlated. Disregarding values' hierarchical distribution may affect the accuracy of selectivity estimation. For instance, assume we have an XML document as that of Fig. 1(a), and we want to estimate the selectivity of path expression `item[type='appliance'][price>500]`. For this particular example, the selectivity of `'appliance'` on item is 50% (2 of the 4 items are of type `appliance`), and that of `'price>500'` is also 50%, because there is a high correlation between item `price` and item `type`. Since no

* This research was partially supported by the grants from 863 High Technology Foundation of China under grant number 2002AA116030, the Natural Science Foundation of China(NSFC) under grant number 60073014, 60273018, the Key Project of Chinese Ministry of Education (No.03044) and the Excellent Young Teachers Program of M0EPRC (EYTP)

value distribution information is available, the best we can assume is that the values are independent. Thus, the selectivity of the path expression on item is estimated as 50% × 50%=25%, while the true selectivity is 50%. It is clear to see that the correlation among the data can be one of the key factors affecting the accuracy of the estimation, and we need some statistical structure to capture such correlation.



(a) Example data tree and its region codes **(b) Value/Code distribution graph** **(c) VH of *price***

Fig. 1. An example data and the VH.

State of the art selectivity estimation methods for XML queries can largely be divided into two categories: the graph-based [3,9] and the histogram-based [6,12,13]. Graph based methods use a summary data structure, such as the DataGuide, to represent the occurrence frequency of the nodes and the paths in the tree structure. However, it cannot capture the correlation between different paths or nodes under the same root, while such information is essential in estimating the selectivity of branching queries with predicates. Histogram-based methods [12, 13] provide information about nodes' hierarchical distribution. Information about predicate selectivity is made available by storing the number of value nodes that have a given attribute value in a simple, one dimensional value histogram. But such selectivity is not associated with nodes' hierarchical distribution, in other words, we do not know how such selectivity will affect other nodes.

To obtain an accurate selectivity estimation using state-of-the-art histograms [12,13], we have to build a position histogram for each particular predicate, so that the position distribution reflects the existence of the predicate. However, it is impossible to build histograms for each possible predicate in path expressions.

In this paper, we propose to use a novel histogram, the *value histogram (VH)*, to capture the correlation between the structure and the value. We define a set of operations on the *value histograms* as well as on the *code histograms (CH)* that keep track of the position distribution. We use these histograms to construct a cost tree. We can then compute the selectivity of any node in the path expression by evaluating the cost tree.

The rest of the paper is organized as follows. We introduce the *value histogram (VH)* in Section 2. In Section 3, we propose six basic operations that are used for selectivity estimation in Section 4. Section 5 shows the experiment result. We review related work in Section 6 and conclude in Section 7.

2 Value Histogram

In this section, we introduce the concept of *value histogram (VH)*. The 2-dimensional value histogram is devised to capture the distribution of predicate values across XML tree structures.

XML data are commonly modeled by tree structures. We encode each node n by a pair of integers *(start, end)*represents a region[14]. It is easy to tell how many descendants and ancestors n has by checking the regions containment. Fig. 1(a) shows an XML data tree whose nodes are labelled with region codes.

2.1 Value Histogram (VH)

The purpose of introducing *value histogram* is to associate the distribution of attribute values with the hierarchical positions of the nodes in the tree. We represent the relationship between value and position in a two-dimension space. Here, the x-axis corresponds to *start* (region code), and the y-axis corresponds to the *values* (for value nodes only). For each different node type, such as price (a continuous attribute), or type (a categorical attribute), we build a value/code distribution graph. Each price node in Fig. 1(a) maps to a unique point in the value/code distribution graph of Fig. 1(b). For example, point *(10, 4)* in Fig. 1(b) represents a node with *start=10*, and *value=4*, which is the second price node in Fig. 1(a).

To derive value histograms from the value/code distribution graphs, we discretize the region code *start* on the x-axis, and the continuous attribute values on the y-axis. More specifically, we discretize a two dimensional value/code distribution graph into g $\times$ m grids. Fig. 1(b) is an example of g=4, m=4. From a discretized value/code distribution graph, we derive the value histogram by counting the number of nodes inside each grid cell. Fig. 1 (c) is a 4 $\times$ 4 value histogram for the *price* nodes. We use *VH[i][j]*, ($0 \leq$ i¡g, $0 \leq$ j¡m), to denote the number of `price` nodes whose value and *start* code fall in grid cell *(i, j)* in the value/code distribution graph.

The value histogram embodies the relationship between the value and the hierarchical structure, and reflects the value correlation of the data. We can build a value histogram for each type of the value nodes. Thus, a predicate in a path expression (e.g., 200 $\leq$ price <500), together with a region constraint (e.g., descendants of a node whose region code is *(7,28)*), corresponds to a set of grid cells in the *value histogram*. We can sum up the counts *VH[i][j]* to get the distribution of the nodes that satisfy the predicate (see Section 3).

Note that in the value/code distribution graph and in the value histogram, we are only concerned with the *start* code but not the *end* code. This is because value nodes always appear as leaf nodes in the XML tree structure, and we only care about the distribution of its parent nodes and/or ancestor nodes. In order to count a node's ancestors and parents, we only need the *start* value of the region code.

Beside *value histograms*, we need another type of histogram, the *code histogram*, to summarize the Ancestor-Descendent or Parent-Child relationships among the nodes.

2.2 Code Histogram

Several code histogram models [12,13] have been proposed in previous works to estimate the size of structure join. We modify some statistics and use them for selectivity estimation.

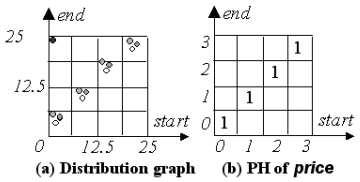


Fig. 2. Position Histogram.

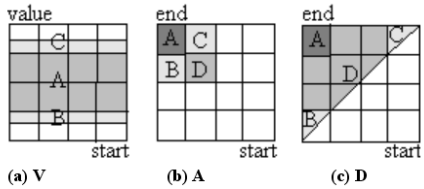


Fig. 3. The V, A, and D operations.

Position Histogram (PH): If we regard the region codes as coordinates in a 2 dimensional space, where values in the x-axis correspond to *start*, and values in the y-axis to *end*, the containment relationship between the nodes will be turned into the position relationship between the points. The points have the following property: (i) none of the points can fall into the area below the diagonal of the matrix, and (ii) given a point x, all the descendants of x appear to the right and lower part of point x, and all the ancestors of x to the left and upper part of point x [12,13]. Fig. 2(a) shows the distributions of nodes in Fig. 1(a). Fig. 2(b) shows a 4×4 position histogram for the price nodes.

PL Histogram (PLH): Instead of using a 2-dimensional mapping, we can also map region codes into intervals in a 1-dimensional space [12]. Assume the root node is encoded (min, max). The PL histogram is constructed by dividing the range of [min, max] into g grids and counting the nodes in each grid.

The statistics we need is in addition to those used in [12]. For each grid, we need to know i) *the number of intervals that start in the grid (CD)*, ii) *the number of the intervals contained in the grid (CA)*, iii) *the average length of the intervals*, and iv) *the selectivity of the intervals*. We use these statistics in histogram operations in Section 3.

In general, we build code histograms for each type of node in the data tree to keep track of the structural distribution of the XML data. Next, we show how the histograms can be used for selectivity estimation.

3 Basic Histogram Operations

We use histograms for selectivity estimation of path expressions with predicates. In this section, we define six basic operations on histograms: *V*, *S*, *D*, *A*, *PA*, and *PD*. The first two, *V* and *S*, are used to study value correlation, and the rest are for hierarchical correlation.

***V*: Value Selectivity Estimation** The V operation counts the number of the nodes within a given *value range* using the value histogram. The result of the operation, *SH[i] (0 ≤ i <g)*, is the number of nodes that satisfy the *value range* (in code grid i).

Let $Pmin \leq attr \leq Pmax$ be a predicate of interest. Assume the code/value space for attribute *attr* is divided into g × m grids, with Wx and Wy being the x-axis and y-axis grid width respectively. Then the number of nodes in each code grid that satisfy the predicates is estimated to be:

$$\begin{aligned} SH[i] = & \sum_{j=\lfloor P_{min}/W_y \rfloor + 1}^{\lfloor P_{max}/W_y \rfloor - 1} VH[i][j] \\ & + (\lfloor P_{min}/W_y \rfloor + 1 - P_{min}/W_y)\, VH[i]\, (\lfloor P_{min}/W_y \rfloor) \\ & + (P_{max}/W_y - \lfloor P_{max}/W_y \rfloor)\, VH[i]\, (\lfloor P_{max}/W_y \rfloor) \end{aligned} \quad (1)$$

There are three additive terms in the formula. The first adds up the grids fully included in the given value range of the predicate, and the other two adds up the grids partially included (assuming uniform distribution). They are shown by the shaded area in Fig. 3(a).

***S*: Value/Code Selectivity Estimation** The S operation (Eq 2) estimates the number of nodes in each position grid satisfying a predicate. It associates the value distribution (Eq 1) with the hierarchical distribution (position histogram). The result of the S operation is a two dimensional *code histogram (CH)*. The S operation assumes that the selectivity of a value on a node in a certain grid has uniform impact over the nodes in the same position grid. Our experimental results in Section 5 show that the assumption works well.

$$CH[i][j] = PH[i][j] \frac{SH[i]}{\sum_{j=0}^{g-1} PH[i][j]} \quad (2)$$

***D*: Descendant Selection** and ***A*: Ancestor Selection** Given a node, the D operation (A operation) returns the distribution of its descendants (ancestors) in a *code histogram (CH)*. The meaning of the two operations is illustrated by the position histograms in Fig. 3(b) and (c). For instance, to find the descendants for a node in region "A", we include the region of "D", and partially include the region of "B" and "C". Similar reasoning applies to the A operation. Due to a lack of space, we refer readers to [15] for a detailed description of the computation of D and A.

***PD*: Descendant Pruning** and ***PA*: Ancestor Pruning** Given a node, we might be only interested to know the number of its direct children instead of all of the descendants. However, such statistics is not immediately available from the position histogram. The PD and PA operations are used to exclude the descendants by estimation. Due to a lack of space, we refer readers to [15] for a detailed description of the computation of PD and PA.

4 Selectivity Estimating of Path

In this section, we demonstrate how to construct a cost tree using the operations discussed in Section 3 for selectivity estimation. The original value histograms

(*VHs*) and code histograms (*CHs*) are used as input, and the output is a code histogram that maintains the distribution information of the node satisfying the path.

4.1 Simple Predicate and Simple Path

To estimate the selectivity of a simple predicate, for example, `[type='appliance']`, we compute *SH* by applying the *V* operation (Eq 1) on the value histogram built for the value nodes. Then we perform the *S* operation on the *SH* and the code histogram for the `type` nodes to get a new code histogram *CH*, which includes in each grid only the nodes satisfying the predicate.

A simple path can have two basic types of components: `a/b` and `a//b`. For `a//b`, it is easy to estimate the distribution of `a` or `b` satisfying path `a//b`. We simply use the *A* or *D* operation.

It is not as straightforward to estimate the distribution of `b` satisfying the path `a/b`, because `b` nodes might be self-nested and we must exclude those `b` nodes that are descendants of `a` but not children of `a`. To achieve this goal, we first perform the descendent pruning operations *(PD)*. Then, we run *D* operation to get the estimation. Similarly, to estimate the distribution of `a` nodes satisfying path `a/b`, we perform the ancestor pruning operation *(PA)*.

4.2 Complex Path with Predicates

There are two factors affecting the selectivity of a node in a path: one coming from the ancestor path, the other from the descendant paths. We discuss the two factors respectively.

To estimate the selectivity of the node satisfying the ancestor path, we perform a series of D operations on the code histograms of the nodes in the ancestor path. The result of one operation is used as input to the next operation until we derive the code histogram of the target node in the path. If the relationship between two nodes is '/', we need additional *PD* operations as well. For example, assume we want to compute the selectivity of node `n3` in path expression `n1/n2//n3//n4[a>v]`. Because the relationship between `n1` and `n2` is '/', we perform a *PD* operation on the code histogram of node `n2` to prune those nodes that are descendants of `n1` but not children of `n1`. Then, we run a *D* operation on the result of the *PD* operation and the code histogram of node `n1`. Another *D* operation is needed to acquire the selectivity of node `n3`. Because the relationship between `n2` and `n3` is '//', the *PD* operation is skipped. To estimate the selectivity of the nodes satisfying the descendant path, we perform a series of *A* operations along the path in the reverse direction. If the relationship between the nodes is '/', we perform the *PA* operation on the parent nodes and the PD operation on the children nodes. In our example, the descendant path of `n3` is `//n4[a>v]`. We deal with the predicate first by performing a *V* operation and an *S* operation. Because the relationship between `n4` and `a` is '/', before the *A* operation, we perform a *PD* operation on the resulting code histogram of the

V operation and a PA operation on the code histogram of `n4`. The cost tree is shown in Fig. 4(a).

Any complex path can be decomposed into a set of simple predicates and simple paths. If the selectivity of a node is affected by many paths, we compute the selectivity of each path, and combine them together. For example, in path `n1[n2>v and n3>w]//n4`, `n1` is a switch node, which has three descendant paths: predicate `n2>v`, predicate `n3>w`, and a descendant path `//n4`. We deal with them one by one. First, we get the selectivity of `n1` from the predicates, and then use the intermediary results to compute the selectivity from the descendant path. The cost tree is shown in Fig. 4(b).

Sometimes the computation can be complicated. In our previous example, the path expression has 5 nodes, but the cost tree in Fig. 4(a) for selectivity estimation has 9 operations. However, if we know some nodes are not self-nested, we can omit some. For example, we can omit the PD and PA operations in Fig. 4(a), if we know (from the schema) that all `n3` nodes are descendants of `n2`, all `n2` nodes are children of `n1`, and all `a` nodes are children of `n4`. The simplified cost tree is shown in Fig. 4(c).

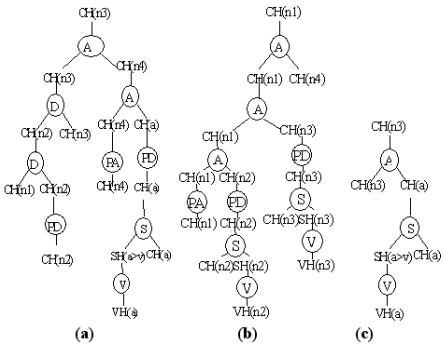


Fig. 4. complex Cost Trees.

5 Experiments

Data sets: We use two XML data sets in our experiments. One is the Xmark data set, whose sizes can range from 100KB to 100MB. The data schema contains 550 different types of nodes and 10 of them are value nodes. The other dataset, Bib, which is generated by XML SPY, has a simple structure, but the structure and the values in the data are highly correlated.

Storage Cost: The size of the statistical information we need is proportional to the number of nodes in the data schema as well as to the number of grids in the histogram. Since the schema is much smaller than the data, major savings can be realized in our approach. The storage cost of the *value histogram (VH)* is $g \times m \times n \times gridsize$, where g is the number of the code grids, m is the number of the value grids, and n is the number of value nodes. The storage cost of the *position histogram (PH)* is $g^2 \times$ n $\times$ gridsize. For example, an Xmark document with 550 nodes in schema, the total histograms size is 675KB. The VH and PH are sparse matrix that can be compressed. In Fig. 5, CVH represents the size of the compressed VH.

Estimation Cost: We use six typical queries with different selectivity. The query time is sensitive to the size of data sets, but the estimation time is not, instead, it is sensitive to the size of the schema, the size of the histograms, and the number of operations in the cost tree. We use a 2M data set to evaluate the

queries. Fig. 6 shows the average estimation time and query time. The estimation time for histograms with fewer than 50 grids is very short (less than 1 millisecond). With the increase of the grids, the estimation time of *PH* method increases faster than PLH method. When the grid is as fine as 100*100, estimation based on *PH* takes more time than query.

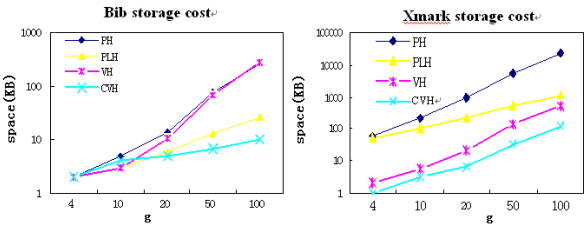


Fig. 5. Storage cost of statistical information.

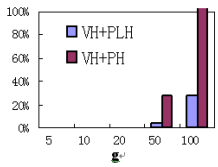


Fig. 6. Estimation Cost.

Estimation Accuracy: Fig. 7 shows the estimated accuracy for the above queries using *PLH+VH* and *PH+VH* methods. Note that the first method has very reasonable error margins (<10%) when the number of grids are greater than 20. For some queries, for example, queries Q1, Q2 and Q3, the *PLH+VH* method has virtually no error.

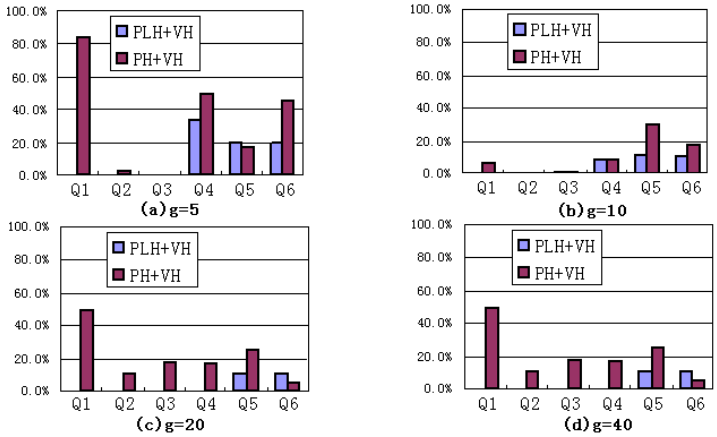


Fig. 7. Estimation Accuracy.

We also note in Table 1 that the uniform and independent distribution assumption results in large errors. An important factor that contributes to these errors is the correlation between the structure and the value in the data set. For example, query Q5 finds `close_aution` with `quantity=1` and `price` >100. There are 80% of the `quantity` nodes whose value is 1, and 80% of the `price` nodes whose value is great than 100. However, few close_aution nodes have both attributes. But with the uniform and independent distribution assumption, the total selectivity will be 60%. The errors will be magnified with longer paths and more predicates. The errors of the *PH+VH* method are caused by the grid. In

the sample data sets, the code region is wider than [0, 10000]. The width of each grid is as large as 100 even for 100*100 grids. The difference between the code of a parent code and its children may be very small (sometimes only 1). Thus, they very likely appear in the same grid. Estimation in the same grid is based on the uniform assumption, which leads to the error. This type of error is not evident in the *PLH+VH* method because *PLH* counts the ancestors and the descendants independently.

Table 1. Example queries list.

	Acutral	**Uniform**	**Queries**
Q1	60%	50%	//bib[No.>5]
Q2	40%	5%	//Book[price>100]
Q3	10%	1%	//Book[author='Mike']
Q4	80%	40%	//Paper[type=1][page>4]
Q5	5%	60%	//close_auction[quantity=1][price>100]
Q6	30%	50%	//Open_auction[increase>20]

6 Related Works

McHugh et al [9] used DataGuide to maintain statistics information. The system stores selectivity information for all paths in the database of length up to k, and uses it to infer selectivity estimates of longer path queries. Markov table is used to maintain statistics of all paths up to a certain length [1], and such information can be aggressively summarized. These techniques do not maintain correlations among paths, and consequently, they do not support accurate estimation of path expressions with embedded predicates.

Chen et al [3] used pruned suffix trees to summarize tree structures, and used a set hashing technology to capture the correlations among the various sub paths in the data. However, the technique only applies to specific twig patterns which contain only parent-child links and the predicate can only have the equality test. Another problem of the method is that their correlation computation is based on the same data type.

StatiX [6] combines schema information with histograms. The system gives each node an id, and ranges of ids of different types do not overlap. For each node type, histogram is used to store the count of children nodes. This method can estimate the selectivity of children nodes, but not the selectivity of the ancestor nodes satisfying some descendant paths.

Timber [13] combined the XML data encoding with a 2 dimension histogram method, using a position histogram to collect and estimate the sizes of containment join. The computation needs the position histogram of both the ancestor set and the descendant set. In a path expression, the predicate may be very complex. It is impossible to build position histograms for each set of predication. Furthermore, the algorithm can only estimate the selectivity of ancestor-descendant join. However, there can be many parent-child relations in path expression queries. Wang et al [12] provided a one-dimension histogram method for containment

join size estimation, which improves accuracy in some cases, but the computation limitation also exists.

7 Conclusion

In this paper, we propose to use a novel type of histogram to capture such correlations. Through a set of operations on the histograms, we are able to estimate the selectivity of any node in a complex path expression. Experimental results indicate that our method provides improved accuracy especially in cases where the distribution of the value or structure of the data exhibit certain a correlation. Our future work will extend the method to support XQuery.

References

1. Aboulnaga, A. R. Alameldeen, J. Naughton. Estimating the Selectivity of XML Path Expressions for Internet Scale Applications. In: Proc. of the 27th VLDB Conf., Italy,2001. 591-600.
2. Chan, P. Felber, M. Garofalakis, R. Rastogi. Efficient Filtering of XML Documents with Xpath Expressions. In Proc. of ICDE Conf., 2002. USA. 235-244.
3. Z. Chen, H. V. Jagadish, F. Korn, N. Koudas, S. Muthukrishnan, R. Ng, D. Srivastava. Counting Twig Matches in a Tree. In: Proc. of ICDE Conf, 2001. 595-604.
4. Choi, M. Fernandez, J. Simen . The XQuery Formal Semantics: A Foundation for Implementation and Optimization. Technical Report, http://www.cis.upenn.edu/kkchoi/ galas.pdf, May 31, 2002.
5. B. Cooper, N. Sample, M. J. Franklin, G. R. Hjaltason, M. Shadmon. A Fast Index for Semistructured Data. In Proc. of 27th VLDB Conference, Roma, 2001, 341-350.
6. J. Freire, R. Jayant, M. Ramanath, P. Roy, J. Simeon. StatiX: Making XML Count. In: Proc. of ACM SIGMOD 2002, USA. 181-191.
7. V.H. Jagadish, S. Al-Khalifa, L. Lakshmanan, A. Nierman, S. Paparizos, J. Patel, D. Srivastava, Y. Wu.. Timber: A native XML Database. Technical report, University of Michigan, 2002.
8. Q. Li, B. Moon. Indexing and Querying XML Data for Regular Path Expressions. In: Proc. of 27th VLDB,Roma,Italy, 2001. 341-350.
9. McHugh, J. Widom. Query Optimization for XML. In: Proc. of 25th VLDB Conference, UK, 1999.315-326.
10. T. Milo and D. Suciu. Index structures for path expression. In Proc. of 7th International Conference on Database Theory, 1999.277-295.
11. N. Polyzotis, M. Garofalakis. Statistical Synopses for Graph-Structured XML Databases, In: Proc. of ACM SIGMOD Conf., 2002.358-369.
12. W. Wang, H. Jiang, H. Lu, J.F.Xu. Containment Join Size Estimation: Models and Methods. In: Proc. of ACM SIGMOD 2003, USA. 145-156
13. Y. Wu, J. Patel, H. Jagadish. Estimating Answer Sizes for XML Queries. In: Proc. of EDBT 2002. 590-608
14. Zhang, J.F. Naughton, D.J. DeWitt, Q. Luo and G.M. Lohman. On Supporting Containment Queries in Relational Database Management Systems. In: Proc. of ACM SIGMOD Conf., 2001.425-436
15. Yu Wang, Xiaofeng Meng, Shan Wang. Estimating the selectivity of PWP by histograms. Technical Report. ftp://202.112.116.99/pub/paper