Towards Maintaining Path Selectivity Statistics in Distributed XML Stores

© Yury Soldak

Department of Computer Science, Saint-Petersburg State University,
University Prospekt 28, St.Petersburg, Russian Fed.
E-mail: soldak@hotbox.ru

Abstract

The main topics of this paper are XML-Store in the distributed case and the major problems of evaluation queries there. The significance of the research direction is grounded and the idea of using DataGuide-like strustures to store necessary statistical information in the distributed case is proposed. A notion of Distributed DataGuide (DDG) is introduced and the overview of various areas of its application is shown.

1 Introduction & Related Work

Developed for data exchange on the Web, XML becomes more and more popular. It is very likely that most of data on the Web can be reached in a form of XML documents in the nearest future. As for the Web, it can be characterized as fairly unpredictable network of heterogeneous data sources [5]. So, the topical problem of the present is the research of different aspects of XML-query evaluation on the Web, for example in the case of several remote servers which form a distributed store.

One of the major problems related to XML query evaluation is structural joins processing [7]. There were many research works focused on developing effective strategies to solve the problem recently. One of the main conclusions which researchers made is about the central role of an accurate estimation of the path expression selectivity for effective structural joining.

Two papers focused on problems related to estimating the selectivity of XML path expressions are the background of the current work. In the first one [1] two techniques were proposed for estimating the selectivity of simple path expressions over large-scale XML data: path trees and Markov tables. Both techniques summarize complex and large-scale data in a small amount of memory and use this summary for selectivity estimation. Concept of the path tree obtained from the paper is heavily used in the current work.

The second paper [4] introduces XPathLearner, a method for estimating selectivity of the most commonly used types of path expressions based on a feedback analysis. XPathLearner stores selectivity related statistics in a Markov table, but as considered further, this is not the best solution in the case of distributed XML store. Primary goal of the current paper is to define structure which will be (a) suited for the distributed case and (b) a convenient basis for developing XPathLearner-like solution.

Let's take a look at the (a) property. Both papers mentioned above study problem of gathering, updating and storing selectivity statistics in a global scope. In other words, there is no way to estimate the path expression selectivity for the particular server of the store. Therefore, the techniques lack for one of the most needful features for evaluating distributed queries (see 3.1).

The rest of the paper is organized as follows. In the section 2 we describe shortly a distributed data model used in the paper. Then the problems related to distributed query evaluation are discussed (section 3). After that, in the section 4, one can find definition of Distributed DataGuide concept, a structure introduced to make resolving of discussed problems easier. Attempt to outline solving methods using Distributed DataGuide was made in the section 5. Thoughts on some problems related to updating of Distributed DateGuide can be found in the section 6. And, finally, the section 7 contains conclusions.

2 Data Model

2.1 Two Presentation Layers

Two presentation layers can be determined for distributed XML store.

- a) system (lower) - set of documents each of which belongs to one of defined servers. Documents linked to each other for some kind.
- b) user (upper) - one big (master) document which consists of several parts. Every part resides at one of the servers.

Clearly the user layer is just a more handy representation of the system one. Thus the user layer can be introduced by a system architect with the purpose of understanding a distributed store structure more clear and easy to use (especially for query composition).

2.2 Difference to Previous Models

In the considered model (as opposed to [1, 5]) number of servers can't exceed some predefined value (at least pre-

Proceedings of the Spring Young Researcher's Colloquium on Database and Information Systems SYRCoDIS, St.-Petersburg, Russia, 2005

dictable). This restriction gives us the possibility to assume that the size of generated structure does not exceed reasonable size of memory even without using reducing techniques [1]. The question about adaptation of such techniques to the structure need to be studied in order to relax the restriction. There is no terms about the size of data stored at servers.

2.3 Link Defining Methods

One can use different methods to define relations between XML documents. The author is going to use the XLink technology [8] for modeling a distributed store at the experimental phase of the current research. One of XLink key features is usage of XPointer [10] as a tool for addressing elements of XML document. As we know, XPointer is a modest extension of XPath [9]. Therefore, it is possible to obtain a lot of information about the remote store structure directly from a link definition. And vice versa, if we know the selectivity of path expression, it is possible to estimate the selectivity of XLink link.

3 Distributed queries evaluation (Problem Definition)

On figures 1 and 2 one can see an example of the query to the distributed store in user and system versions respectively. The query will get an information about courses with the same name, but taking place at two different universities.

```
for
 $c1 in doc('m.xml')//univ[@id='SPbU']//course,
 $c2 in doc('m.xml')//univ[@id='MSU']//course
where $c1/name = $c2/name
return $c1
```

Figure 1: user presentation layer

```
for
 $c1 in doc('http://spbu.ru/db.xml')//course,
 $c2 in doc('http://msu.ru/db.xml')//course
where $c1/name = $c2/name
return $c1
```

Figure 2: system presentation layer

3.1 Selectivity of Path Expressions on Remote Servers

It is necessary to join two sequences $c1 and $c2 during evaluation of queries listed on figures 1 and 2. These sequences might be obtained in several different ways. For example, query evaluator can naively obtain all the course elements for the each university by sending simple queries to the corresponding servers, then locally join two (possibly) big sequences. Obviously, described strategy is not optimal especially in case of following: firstly, one of universities may have less courses and secondly, the cardinality of an intersection of two mentioned sequences may be quite little. More attractive strategy is described further. Evaluator selects a sequence with less elements and obtains its distinct values (query to one of servers), then sends a specially generated query (provides a remote evaluator with obtained distinct values and asks it to select corresponding elements from its local sequence) to the other server, finally returns an answer based on the result of last query to the user. To use this strategy we have to know selectivity of the path expressions for course elements. Moreover, it is important to know selectivity with regard to a server, not just abstract selectivity in global scope.

3.2 Implicit Initiators

The user query may be of two types: *low-level* or *high-level*. It depends on user's store awareness (in other words, on presentation layer of use). The low-level query (fig. 2) contains at least one explicit link to the remote server used in a "document" function call. We'll refer to such the constructions as *explicit initiators* (or simply *initiators*) further according to [6]. The high-level query (fig. 1) may not have any explicit initiator, but in most cases it contains set of *implicit initiators*. Implicit initiators can emerge when a path expression contains cross-server steps. So to evaluate such an expression subquery needs to be generated and evaluated independently.

For example take a look at fig. 1. It contains 2 implicit initiators which replaced with corresponding explicit ones at fig 2. It is not very hard to agree with following proposition: every implicit initiator have at least one explicit counterpart. Furthermore, several explicit initiators can emerge from the single implicit. This happens when step of path expression goes throw set of cross-document references (an associative arcs, as we'll define them later in 4.2).

Obviously, low-level query may have implicit initiators amount others. Hence, an additional step need to be introduced into query evaluation flow in any case to find out all the implicit initiators and replace them with corresponding explicit ones. Such a step is an integral part of every distributed query evaluator in our data model. To make the step we need the knowledge of a store architecture and fast cross-document links recognition. To support an evaluator with these things is one of the tasks of the current work.

Noticeable, histograms are not suitable for our aims now. It is not clear how to store information about initiators there. Additional analysis is need to be provided in order to answer the question.

3.3 Handling Cycles in a Store

Nature of the system layer can result in master document with cyclic associations. In fact the user may expect that master document has XML-like properties, a tree structure in particular. So such associations are harmful to the store and in most cases indicate store inconsistencies. We are in need of tool for effectively detection of cycles. On the other hand, when user layer is not introduced, cycles seems to be not so bad. Partly it is true, but only partly because query evaluators have to be able to detect such cycles in any case. In the next section we'll consider a structure which can help with the cycle detection problem.

4 Distributed Data Guide

4.1 DataGuide and Path Tree Definition

Conception of DataGuide (DG) was originally introduced in [3]. From that times till present DG is widely used as a base for indexes (for example [2]) and structures for statistical information representation [1].

DataGuide is defined as follows: every path of the document has exactly one path in the DG, and every path of the DG is a path of the document. It's worth to mention, the DG does not contain values of elements and attributes of the document.

It is obvious, the structure which meets definition above, needs noticeably less memory than the original XML document and thus is suitable to use in the query evaluation process.

Taking into consideration the tree nature of an XML document, we can introduce the Path Tree (PT) concept similarly to DG. PT is characterized by (a) it is a tree, (b) every node in PT holds information about number of the siblings with the same name which form this node [1].

4.2 New Arc Types

To describe an idea of the Distributed DataGuide we need some additional definitions.

It is clear, nodes of DG (and PT) are linked by parent-child arcs. We'll call them *simple* further. Besides this, arcs which specify ancestor-descendant relation are of interest too. Let's call them *generalized* and mark with "*" at figures. The generalized arcs are of interest in our case (see sec. 6 for details) because we always have only the partial information about the remote stores. Besides we can use generalized arcs for local store too when it is not important to know the exact path.

Both mentioned above arc types share a property – they define relations between nodes which belong to the same document. Arcs of the cross-document references appear in the distributed case. They can be not only cross-document, but cross-server too in case of documents reside at different servers. Such arcs will be called *associative* at the rest of the paper and marked with "˜" symbol. Assume that we have only simple XLink links in our store, so associative arcs are of one-way type. It's possible that in future we'll can throw out this limitation, but for now we assume it.

4.3 Distributed Data Guide Definition

Distributed DataGuide (DDG) is a set of PTs as defined above, but with one exception: arc in our PTs can be both simple and generalized. PTs are linked together with associative arcs, each of which has nodes from different documents as starting and ending points. See figure 3 for an example. There are 4 PTs and 4 associative arcs that can be found. As one can see in the figure, DDG is not always a tree, but, due to associative arcs, it is a directed graph.

5 Using Distributed Data Guide

Using DDG we can solve most of problems defined in section 3. For example, the selectivity estimation method for DDG is the same as for PT. In fact DDG is a directed graph, so to solve cycle detection problem well-known

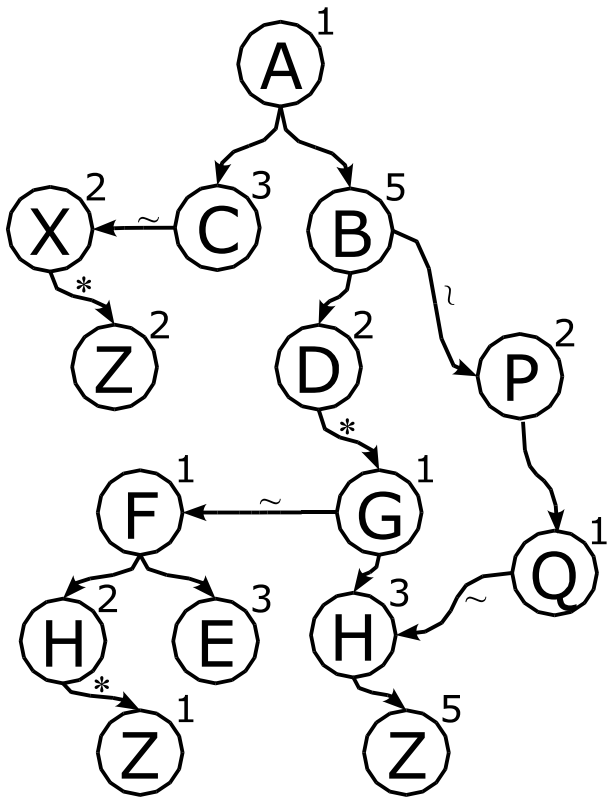

Figure 3: Example of DDG

algorithms can be used. Discovering implicit initiators is not so hard a challenge either, because the presence of implicit initiator in path expression can be iff at least one associative arc is in projection of this expression to DDG.

It is clear, that DDG may lack some needful features for certain tasks. But this is not a real problem because DDG can be easily extended to meet requirements. As example, for each ˜-arc an average response time of the remote server on the path expression containing this arc can be calculated and saved somewhere. The candidate to reasonable statistics for the DDG nodes is a frequency of modification of the cardinality value for every certain node and (average) delta of these alterations. The first extension can be useful for the estimation of query execution time and of course for choosing an optimal query evaluation strategy. Maintaining of additional statistics for nodes allow us to predict probability of selectivity changes. Improvements can be of a really different kinds, DDG is only the foundation for such extensions.

Moreover, very likely DDG can be useful in local case too. We think statistic about cross-document references can help there sometimes.

6 On Updating of Distributed Data Guide

As yet we described only DDG benefits, now it's time to discuss the construction problem of DDG-like structure. There two ways: the first one is when servers exchange with statistical information (local PTs) from time to time and the second one – when we can obtain statistic data about the remote storage only by analyzing answers from there (feedback way).

In the first case we can always have the complete pic-

ture about the structure of distributed storage on each server (or on a certain one). But this picture becomes out of date soon, and it can steal unallowable size of memory keeping a lot of unnecessary information (for example about the selectivity of path expressions, which never will be queried and so evaluated). Different variations of the techniques described in [1] can be used to solve the last problem.

To avoid these problems entirely we can go the second way (of information exchange). But it isn't easy to do that. As we mentioned above, technique based on Markov tables exists, namely XPathLearner, and described in [4]. There is no any research done for DDG-like structures in that direction yet. One of the major problems here is an ambiguity during update.

Assume that the new selectivity of the path expression "/A//Z" is equal to 10. How should we distribute this value among three possible variants (see fig. 3)? It's not clear yet, but, as discussed in previous section, average delta (or probability) of node cardinality modification, possibly, can help. Providing an answer to the question is the focus of author's further investigations.

7 Conclusions & Future Work

In the current paper one of the possible ways to store statistical information was introduced. We considered why this solution can be good enough, but no one experimental result was gained yet to prove suppositions. Experimental phase is the next aim of current research. Also, many starting points for further theoretical studies were outlined in the paper.

8 Acknowledgements

I would like to thank my scientific adviser Boris Novikov for his support and valuable comments.

References

[1] Ashraf Aboulnaga, Alaa R. Alameldeen, and Jeffrey F. Naughton. Estimating the Selectivity of XML Path Expressions for Internet Scale Applications. In *The VLDB Journal*, pages 591–600, 2001.

[2] A. Fomichev. XML Storing and Processing Techniques. In *SYRCoDIS*, pages 13–16. NIIMM, 2004.

[3] Roy Goldman and Jennifer Widom. DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases. In Matthias Jarke, Michael J. Carey, Klaus R. Dittrich, Frederick H. Lochovsky, Pericles Loucopoulos, and Manfred A. Jeusfeld, editors, *VLDB'97, Proceedings of 23rd International Conference on Very Large Data Bases*, pages 436–445. Morgan Kaufmann, 1997.

[4] L. Lim, M. Wang, S. Padmanabhan, J. S. Vitter, and R. Parr. XPathLearner: An On-line Self-Tuning Markov Histogram for XML Path Selectivity Estimation. In *VLDB*, pages 442–453, 2002.

[5] Marko Smiljanic, Henk M. Blanken, Maurice van Keulen, and Willem Jonker. Distributed XML Database Systems. Technical Report TR-CTIT-02-46, CTIT, University of Twente, The Netherlands, October 2002.

[6] Y. Soldak. Estimating influence of subquery granularity to evaluating a query to remote XML-store. In *SYRCoDIS*, pages 22–25. NIIMM, 2004.

[7] Y. Wu, J. M. Patel, and H. V. Jagadish. Structural Join Order Selection for XML Query Optimization. In *ICDE*, pages 443–454. IEEE Computer Society, 2003.

[8] XML Linking Language (XLink) Version 1.0, 27 June 2001. W3C Recommendation.

[9] XML Path Language (XPath) Version 1.0, 16 November 1999. W3C Recommendation.

[10] XML Pointer Language (XPointer), 25 March 2003. W3C Recommendation.