Analysis and design of approximate queries over XML documents using statistical techniques.

PhD student: Stefania Marrara advisor: Letizia Tanca

Politecnico di Milano, Dipartimento di Elettronica ed Informazione,
Piazza L. Da Vinci 32, I-20133 Milano, Italy
{marrara, tanca}@elet.polimi.it

Abstract

In the last few years several repositories for storing XML documents and languages for querying XML data have been studied and implemented. All the query languages proposed so far allow to obtain *exact* answers, but when applied to large XML repositories or warehouses, such precise queries may require high response times. To overcome this problem, in traditional relational warehouses fast *approximate* queries are supported, built on concise data statistics based on histograms or sampling techniques. We believe that the current trend of XML claims for the extension of such approaches also to query massive XML data-sets. In our work we propose a novel approach to *summarize an XML document collection* using concise data statistics (e.g., histograms), which allows *approximate* queries on such data using the XQuery standard language.

1 Introduction

In the last few years XML, initially proposed for representing, exchanging and publishing information on the Web, has spread in many applications: it is used as a solution for publishing legacy data, for storing data that cannot be represented with any other traditional data model, for guaranteeing interoperability among different applications, for integrating Web services, for mining Web data, and so on. All these applications put forth a strong demand both for repositories storing XML documents, and for XML query languages, explicitly conceived for retrieving and restructuring XML data. Several proposals address both issues, but all the languages actually used for XML data [BC00] allow to obtain only exact answers to queries. When applied to large XML repositories or warehouses, precise queries may require high response times. To overcome this problem, in traditional relational warehouses fast approximate queries are supported, based on concise data statistics built using histograms or sampling techniques. We believe that the current trend of XML claims for the extension of such approaches also to query massive XML data-sets. The basic idea for approximate answers is to store pre-computed summaries of the XML warehouse, also called *synopses* (concise data collections), and to query them instead of the original database, thus saving time and computational costs. The first part of this PhD Thesis proposes a methodology for the semi-automatic derivation of summarized documents for massive, heterogeneous data-sets, with the final aim of producing query transformation rules from queries on the original data-sets to queries on the summarized data-set. In the next future we intend to elaborate an XQuery based algebra to model XML queries and to individuate a measure of the error obtained when computing an approximate answer instead of an exact one. At the end, we intend to explore also other kinds of statistical summaries, such as different kinds of sampling, to individuate the techniques which offer the best performance, and to design a tool for the verification of the proposed methodology.

2 State of the art

Several works have been proposed for aggregate queries applied to traditional relational warehouses. The most important among these approaches are those of Hellerstein [HHW97], Poosala and Ganti [VV99a, VV99b], Gibbons and Matias [GM99] et al. Their works show the possibility to use means such as histograms, samples and wavelets as statistical summaries to make approximate queries involving a significantly reduced response time. The approach based on *histograms* gathers statistics which describe the data distribution. The *sample* technique calculates samples of the original data in order to decrease the number of tuples to be queried. A popular technique is the *uniform random sampling* (in which every item in the original data set has the same probability of being sampled) that mirrors the original data distribution. The

wavelets technique is based on a mathematical tool for the hierarchical decomposition of functions. Wavelets represent a function in terms of a coarse overall shape, plus details that range from coarse to fine. The first step to start the wavelet decomposition procedure is to choose the wavelet basis function. In traditional databases, Haar wavelets [BW94] are usually chosen because they are conceptually the simplest wavelet basis functions and they are the fastest to compute and the easiest to implement. The Haar wavelet decomposition is a set of coefficients representing the overall average of the original set of values, followed by the detail coefficients ordered by increasing resolution. If less detailed coefficients are stored (saving space and dimension of data), an approximation of the original data is obtained.

All these techniques have already been employed in the area of XML data, but only to the purpose of compressing them [CCPS01]: data can be divided into homogeneous classes to which specific semantic compressors apply. For example, lossless compressors apply to integers, real data, strings and IP addresses, while lossy compressors based on Haar wavelets allow to code sequences of values efficiently, keeping the possible error under control. The aim of our proposal is to apply these techniques to query large repositories of XML data, reducing the complexity derived from the need to manage very large amount of data. A first work about XML synopses is given by a novel approach to building and using statistical summaries of large XML data graphs for effective path-expression selectivity estimation. The graph-synopses model, termed XSketch and proposed in [PG02] by Garofalakis and Polizotis, exploits localized graph stability to accurately approximate the path and branching distribution in the data graph. To estimate the selectivity of complex path expressions over XSketch synopses they develop an estimation framework that relies on uniformity and independence assumptions to compensate for the lack of detailed distribution information. Their algorithm constructs an XSketch synopsis by successive refinements of the label-split graph, the coarsest summary of the XML data graph. The refinement operations act locally and attempt to capture important statistical correlations between data paths. The structural XSketch synopses are specific instantiations of a generic of generic graph-synopsis model that records some additional edge-label information to capture localized backward and forward stability conditions across synopsis nodes. Our approach has been presented in [CMT02, CMT03].

3 Representing summarized XML data

Following a common use, we represent XML documents as labelled trees $T = (V, E)$, where V is the node set comprising both nodes representing tags and nodes representing text content and attributes[1], and $E \subseteq V \times V$ represents elements and text containment arcs. The structure description of a document (i.e., its DTD) is also described as a labelled graph G_D using the same notation, while $\mathcal{G}_\mathcal{D}=\{G_D^j\}$ is the set of graphs representing the DTD's of the whole document collection.

Given a set $\mathcal{T}=\{T_i\}$ of XML documents, an *XML synopsis* is obtained as a transformation $XMLtransf$: $\{T_1, ..., T_h\} \in \mathcal{P}(\mathcal{T}) \rightarrow \{T'_1, ..., T'_k\} \in \mathcal{P}(\mathcal{T})$, $k \leq h$ (often it will be $k \ll h$), where $\{T_1, ..., T_h\}$ share the same DTD. Their summarization produces a (small) number of documents conforming to one or more new DTDs, which we call *synopsis DTDs* (DTD_{syn}). They can be obtained as a transformation $DTDtransf$: $\mathcal{G}_\mathcal{D} \rightarrow \mathcal{P}(\mathcal{G}_\mathcal{D})$ on the DTD of the initial set of documents.

3.1 Computation of the synopsis DTD

3.1.1 Preliminary analysis.

So far we have focused our work on the computation of the XML synopsis. An XML synopsis is composed by a small collection of documents that store histograms instead of the data composing the original data-set. Each histogram stores the statistical distribution of the values of a certain element divided into disjoint sets called *buckets*, each one characterized by a frequency value (i.e., the number of elements falling in each bucket), and a boundary value(i.e., the highest value of each set, the data of the element are supposed ordered), which are different for each histogram type. In our approach, the computation of (each) DTD_{syn} requires a preliminary phase aimed at analyzing the application aspects of the collection: the type of histogram for the current application and the set of elements to summarize must be chosen. According to the data to be stored in the synopsis a *DTD of the histogram* DTD_H ($\subset DTD_{syn}$) can be defined, dictating the form of the histogram data that collects the statistics of the XML elements. For example, the following DTD shows a possible DTD_H of a generic histogram where frequency and boundary values change for each bucket.

```
<!ELEMENT hist (bucket+)>
<!ELEMENT bucket (frequency, bv)>
<!ELEMENT frequency (PCDATA)>
<!ELEMENT bv (PCDATA)>
```

Set of elements to summarize. During the preliminary analysis step it is also necessary to identify the set $E_s \subseteq E$ of elements to summarize.

[1] Attributes will not be explicitly handled in this paper: however, if a literal semantics [RJJ99] for the representation of XML documents is adopted, they can be treated as a particular case of PCDATA elements.

We represent XML elements with their paths from the root of the document using a XPath [W3C99] notation, in order to distinguish elements with the same tag name but with different internal meaning (e.g., a person's home address is different from the address of the company the person belongs to).

Elements can be summarized in two different ways:

1. we can summarize each element content independently of the content of the other elements (*assumption of element independence*) or
2. we can summarize element values taking into account the values of other (related) elements (*assumption of dependent data*).

For example, we could summarize the content of the elements tagged "age" assuming that the age does not depend on any other value, or we could summarize them relating people's ages to the city where they live. Operationally, we will treat this second case by grouping the dependent elements w.r.t. the elements which they depend on and by calculating the statistics for the separate groupings.

Taking into account this distinction, each element $e \in E_s$ is formally represented as a pair ($pathe$, $\{pathg\}$), where $pathe$ is the path expression of the element to be summarized, and $\{pathg\}$ is a set of path expressions of the elements whereof $pathe$ is grouped in the summarization process. In case of element independence we have $\{pathg\} = \emptyset$. Usually, $pathe$ and $pathg$ will be leaf nodes, otherwise they will be considered as a shortcut for all their leaf descendant nodes (i.e., a non leaf element $pathe$ represents the set of all the paths $\{pathe\prime\}$, such that $pathe\prime$ is a leaf node and is a descendant of $pathe$). In the sequel we always consider the expanded version of E_s containing only leaf nodes.

Depending on E_s, different elements and/or structures may appear in DTD_{syn}: for example, in case of independence assumption, a single histogram will be created for all the values of each given element in the document collection; in case of summarization with grouping, for each element a set of histograms will be collected, one for each grouping value.

Running example. As an example consider the DTD shown in Fig. 1 describing a car store: cars are characterized by their model and color, and for each car the details of the customers who bought it are listed. The figure shows also an example of XML document fragment conforming to such DTD. In order to compute the synopsis DTD for this data collection, we need to choose the appropriate histogram family: since the XML data describe categories such as colors, car models and so on, the equi-width family is the most appropriate to our application.

As far as the set of elements to summarize is concerned, we suppose that the user be interested in storing a summarized collection about the colors, the city and the payment data of the different car models. Therefore, the set of elements E_s to summarize is: E_s={(`store/car/color`, {`store/car/model`}), (`store/car/customer/city`, {`store/car/model`}), (`store/car/customer/payment`, {`store/car/model`})}.

so that colors, cities and payments be grouped by car model.

```
<? xml version = "1.0" ?>
<!ELEMENT store (car+)>
<!ELEMENT car(model, color, customer*)>
<!ELEMENT model (PCDATA)>
<!ELEMENT color (PCDATA)>
<!ELEMENT customer (name, city, payment, card)>
<!ELEMENT city (PCDATA)>
<!ELEMENT name (PCDATA)>
<!ELEMENT payment (PCDATA)>
<!ELEMENT card (PCDATA) #implied>
```

```
<? xml version="1.0" encoding="UTF-8"?>
<store>
 <car>
  <model>Fiat Brava</model>
  <color>white</color>
  <customer>
   <name>James Smith</name>
   <city>New York</city>
   <payment>5.000</payment>
   <card>Visa</card>
  </customer>
  <customer>
   <name>Lena Brown</name>
    <city>Filadelfia</city>
    <payment>5.100</payment>
  </customer>
  ...
 </car>
 <car> ... </car>
 ...
</store>
```

Figure 1: DTD and example XML document

3.1.2 Computation of the synopsis DTD

After the analysis step, DTD_{syn} can be computed. The process produces a single DTD_{syn} containing all the histograms produced by the different groupings specified in E_s. Usually the number of grouping criteria could be too high to summarize all the data into a single document in a readable and effective way. A better solution is to store separately different documents containing all the histograms produced by a grouping criterion, i.e., to break E_s into several subsets $E_{si} \subseteq E_s$, where each E_{si} contains *compatible* or *nested* $\{pathg\}$. The process is applied to each E_{si} and generates a DTD_{syn} for each grouping criterion. Two grouping paths $Pathg' = \{pathg'_1, ...pathg'_k\}$ and

$Pathg'' = \{pathg''_1, ...pathg''_m\}$, with $k < m$, are compatible if $Pathg' \subseteq Pathg''$. Now we are ready to define synopsis compatibility: two synopsis subgraphs represented by $e' = (pathe', Pathg')$ and $e'' = (pathe'', Pathg'')$ are *compatible* if and only if $pathe''$ is a descendant of $pathe'$ and $Pathg'$ is compatible with $Pathg''$.

The elements which have been discarded belong neither to elements to be summarized nor to grouping elements. The only exception is represented by the elements that appear in paths of elements to be summarized or to paths of grouping elements: their original paths are kept also in DTD_{syn} because unchanged paths help query formulation.

3.2 Synopsis Creation

3.2.1 Preliminary analysis.

Also the computation of the document synopsis, $DATA_{syn}$, requires a preliminary step, since the actual summarization depends on the application data: for each element in E_s the *parameters* needed to compute the histogram (e.g., the number of buckets or the frequency value) must be defined.

Given DTD_{syn}, the original data collection, and the parameters of the histograms, the synopsis can be automatically created.

The parameters of the histograms to be computed for each element $e \in E_s$ are represented by a set of tuples $P = \{(e, \{f^j\}, n_b, \{bv^j\})\}$, where $1 < j < n_b$, f is the frequency of the element e in bucket j, n_b is the number of its buckets and $\{bv^j\}$ is the set of n_b boundary values of the buckets of the element.

If we consider the running example, the framework produces following synopsis DTD. Notice that the original path `store/car/model` (i.e., *pathg*) which is used to group colors, cities and payments has not been changed w.r.t. the original DTD and still contains PCDATA; instead, the contents of the elements in *pathe* (color, city and payment) have been replaced with the histogram element `hist`. The elements composing the histogram structure (buckets, frequency and bv) have also been added to DTD_{syn}.

```
<? xml version = 1.0 ?>
<!ELEMENT orders (car+)>
<!ELEMENT car (model, color, customer)>
<!ELEMENT model (PCDATA)>
<!ELEMENT color (hist)>
<!ELEMENT customer (city, payment)>
<!ELEMENT payment (hist)>
<!ELEMENT city (hist)>
<!ELEMENT hist (bucket+)>
<!ELEMENT bucket (frequency, bv)>
<!ELEMENT frequency (PCDATA)>
<!ELEMENT bv (PCDATA)>
```

The elements which have been discarded belong neither to elements to be summarized nor to grouping elements. The only exception in the running example is represented by the `customer` elements: these are kept because they belong to the path of `payment`. Its original path is kept also in DTD_{syn}: unchanged paths help query formulation.

3.2.2 Computation of the synopsis.

For each element in E_s statistics are computed and the corresponding data are inserted in the synopsis, according to the structure defined by DTD_{syn}. The following XML document shows the synopsis obtained by applying algorithm `compute_synopsis` to the example of the car store. The result is an XML document where the cars details are grouped according to E_{si}: cars details are grouped w.r.t the model, and each summarized node contains the histogram details of the colors, the cities and the payments of a given model.

```
<? xml version="1.0" encoding="UTF-8"?>
<store>
 <car>
  <model>Fiat Brava</model>
  <color>
   <hist>
    <bucket><freq>5</freq>
    <b_value>red</b_value></bucket>
    <bucket><freq>3</freq>
    <b_value>yellow</b_value></bucket>
        <!-- other color buckets
            are omitted for brevity --!>
   </hist>
  </color>
  <customer>
   <city>
    <hist><bucket><freq>30</freq>
    <b_value>New York</b_value></bucket>
      <!-- other buckets are
            omitted for brevity -->
    </hist>
   </city>
   <payment>
    <hist>
     <bucket><freq>0</freq>
     <b_value>4000</b_value></bucket>
     <!-- other buckets are
            omitted for brevity -->
     <bucket><freq>55</freq>
     <b_value>16000</b_value></bucket>
    </hist>
   </payment
  </customer>
 </car>
 <car>
  <model>Opel Tigra</model>
    <!-- other summaries are
                    omitted for brevity -->
 </car>
 <car>
```

```
  <model>Class A</model>
     <!-- other summaries
                 are omitted for brevity>
 </car>
</store>
```

4 Conclusions

In our work we have proposed an approach to construct a synopsis of a huge set of XML documents, in order to reduce query computation time to the price of some impreciseness. The structure of the synopsis is close enough to the structure of the original document collection in order to the query on the original data to be transformed into a query on the synopsis.

At the moment we are working to define a set of XQuery transformation rules in order to capture the more complex query cases in a uniform transformation algorithm. The main idea of this approach is to construct a synopsis with a structure as similar as possible to the original documents structure, in order to facilitate and automate the query transformation from the original data-set to the synopsis. The paper focuses on the construction of the synopsis DTD and of the summarized collection itself. The approach is going to be tested using a data set generated by the benchmark XBench [XB] freely available on the Internet. Among the number of benchmarks for XML databases recently proposed such as XMach-1, XMark, X007 etc., which allow to capture different application characteristics, we chose to use XBench because it is a family of benchmarks studied to be application independent. The XBench database generator can generate databases ranging from 10MB to 10GB in size. Moreover we plan to formalize the architecture of a demo prototype, to formalize and implement the automatic transformation of the query and its optimization using the XQuery, and to estimate the error resulting for query applied to summarized data. Then we intend to formally investigate the problem of summarization in XML. Starting from the summarization conditions exposed in [LS97] we will analyze the peculiarity of the XML structure and formulate a set of conditions with the aim to formally demonstrate the correctness of the results obtained by our approximate querying approach.

References

[BC00] Angela Bonifati and Stefano Ceri. Comparative analysis of five xml query languages. *SIGMOD Record*, 29(1):68–79, 2000.

[BW94] Jawerth B. and Sweldens W. An overview of wavelet based multiresolution analyses. *SIAM Rev.*, 36, nr.3:377–412, 1994.

[CCPS01] M. Cannataro, G. Carelli, A. Pugliese, and D. Saccá. Compressione semantica con perdita di documenti xml. In *Atti del IX Convegno Nazionale SEBD 2001*, 2001.

[CMT02] S. Comai, S. Marrara, and L. Tanca. Approximate aggregate queries on xml data. In *Atti del decimo convegno nazionale su sistemi evoluti per basi di dati - SEBD*, pages 222–233, 2002.

[CMT03] S. Comai, S. Marrara, and L. Tanca. Representing and querying summarized xml data. In *Proc. of DEXA 2003 Conf.*, page to appear, 2003.

[GM99] Phillip B. Gibbons and Yossi Matias. Synopsis data structures for massive data sets. *DIMACS: Series in Discrete Mathematics and Theoretical Computer Science: Special Issue on External Memory Algorithms and Visualization*, A, 1999.

[HHW97] J. M. Hellerstein, P. J. Haas, and H. J. Wang. Online aggregation. In *Proc. ACM SIGMOD Conference on the Management of Data*, pages 171–182, Tucson, AZ, 1997.

[LS97] Hans-Joachim Lenz and Arie Shoshani. Summarizability in olap and statistical data bases. In *Statistical and Scientific Database Management*, pages 132–143, 1997.

[PG02] N. Polyzotis and M. Garofalakis. Statistical synopses for graph-structured xml databases. In ACM, editor, *Proc. ACM SIGMOD Conference*, Madison,Wisconsin,USA, 2002.

[RJJ99] Goldman R., McHugh J., and Widom J. From semistructured data to xml: Migrating the lore data model and query language. In *Proc. WebDb*, pages 25–30, 1999.

[VV99a] Poosala V. and Ganti V. Fast approximate answers to aggregate quries on a data cube. In *International working conference on scientific and statistical database management*, 1999.

[VV99b] Poosala V. and Ganti V. Fast approximate query answering using precomputed statistics. *Proc. of IEEE Conf. on Data Engineering*, 1999.

[W3C99] W3C. Xml path language (xpath) version 1.0, 1999. http://www.w3.org/TR/xpath.

[XB] http://db.uwaterloo.ca/ ddbms/projects/ xbench/index.html.