A Flexible Infrastructure for Gathering XML Statistics and Estimating Query Cardinality

Juliana Freire [1] Maya Ramanath[2] Lingzhi Zhang[1]
[1]OGI/OHSU [2]Indian Institute of Science
{lzhang,juliana}@cse.ogi.edu {haritsa,maya}@dsl.serc.iisc.ernet.in

1. Introduction

A key component of XML data management systems is the *result size estimator*, which estimates the cardinalities of user queries. Estimated cardinalities are needed in a variety of tasks, including query optimization and cost-based storage design; and they can also be used to give users early feedback about the expected outcome of their queries.

In [2], we proposed StatiX. In contrast to previously proposed result estimators, which use specialized data structures and estimation algorithms, StatiX uses histograms to uniformly capture both the structural and value skew present in documents. It also leverages schema information to produce high-quality and concise statistical summaries. In particular, it exploits XML Schema transformations [1] to obtain statistics at different granularities.

The original version of StatiX was built as a proof of concept. With the goal of making the system publicly available, we have built *StatiX++*, a new and improved version of StatiX which extends the original system in significant ways. In this demonstration, we will show the key features of *StatiX++*.

2. *StatiX++*: Overview

StatiX++ implements the main components of the StatiX framework. As illustrated in Figure 1, the system takes as inputs an XML Schema and document, and outputs a StatiX summary. The input XML Schema is processed by the *Schema Transformer*, which *tunes* the granularity of the summary by adding and removing type definitions. The transformed schema is then passed on to the *Statistics Gatherer* which collects statistics for each type defined in the transformed schema, and generates the summary. Users may set, through command-line options or through a configuration file, a number of parameters for the *Statistics Gatherer*, including: the type of histograms (*e.g.,* equidepth, equiwidth); and the number of buckets to use for each type. In addition, *StatiX++* has an *estimator module* which takes as inputs a StatiX summary and a set of queries, and outputs the estimated cardinality for each query.

The system is implemented in Java, and uses Xerces-J as the validating parser. New features include: support for ambiguous schemas that result from some of the schema transformations (*e.g.,* union distribution); new techniques to further compress the statistical summaries; and the ability to create summaries of large documents (larger than 1GB) in a reasonable time. For more details about *StatiX++*, the reader is referred to [3].

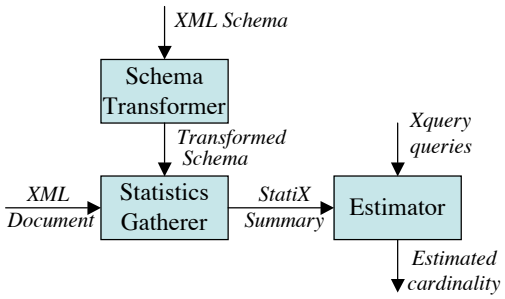

Figure 1. *StatiX++* Architecture

3. Demonstration

We will demonstrate the statistics collection process and result estimation for a variety of schemas and datasets such as IMDB, DBLP, and XMark. In particular, we will show how schemas can be *easily* transformed to collect statistics at different granularities; and how the statistics collection can be tuned through the various knobs available in the system (*e.g.,* different histogram sizes, different transformations), as well as the implications of these choices on the accuracy of cardinality estimation.

References

[1] P. Bohannon, J. Freire, P. Roy, and J. Siméon. From XML schema to relations: A cost-based approach to XML storage. In *Proc. of ICDE*, pages 64–75, 2002.

[2] J. Freire, J. Haritsa, M. Ramanath, P. Roy, and J. Siméon. StatiX: Making XML count. In *Proc. of SIGMOD*, pages 181–191, 2002.

[3] StatiX++. `http://www.cse.ogi.edu/~juliana/StatiX`.