Interface Design of Wrappers for XPath Queries on Non-XML Data

Kenji Watatani (310124)

School of Information Science,
Japan Advanced Institute of Science and Technology

February 10, 2005

Keywords: XML, XPath, non-XML data, wrapper, query processing.

As the cost of secondary storage drops, today there are huge amount of data in various formats saved in the personal computers and on the severs on the internet. It has arisen the demand for the tools to uniformly issue queries on those data in various formats in order to retrieve necessary data. There exist systems for issuing keyword queries on HTML, PDF, Microsoft Power Point, and so on. However, as the volume of data increases, it is getting hard to retrieve only necessary information from it, and it is recently regarded that it is important to be able to issue not only keyword queries but also structural queries which can specify the structure of the data to retrieve.

As an approach to this problem, it has been proposed to use XML as the standard data format for various data. If we represent personal address books, schedule data, e-mail boxes, and so on in XML format, we can uniformly issue queries on those data by using XML query languages, such as XPath, which is a simple standard structural query language for XML. For now, however, many application specific data formats, or standard data formats for images, movies and so on, are still widely used, and even in future, it is not likely that all the data would be in XML format because of various reasons, such as data size and processing efficiency.

Therefore, in this research, we propose a system where we can uniformly issue XPath queries on data in various formats. This system is composed

of a "common module" and "wrappers." There exists one wrapper for each data format, and wrappers provide a common interface to each data format. The common interface consists of a set of operations, and the common module evaluates given XPath queries by accessing each data through those operations.

Here, the design of the interface between the common module and the wrappers is very important. If the interface consists of very low-level operations, it is easier to develop wrappers, but on the other hand, in the evaluation of queries, it is more difficult to adopt various optimization depending on each data format. On the contrary, if the interface consists of very high-level operations, various format-dependent optimization can be embedded in wrappers, but the cost of developing wrappers is higher, which can be a serious obstacle to the support of many data formats.

Based on the observation above, in this research, we examine what operations are needed for efficient evaluation of XPath queries, and design a wrapper interface which achieves both relatively easy development of wrappers and efficient evaluation of queries. Then, we compare that interface and higher-level and lower-level interfaces in their query evaluation efficiency and programming costs in order to justify the design of our interface.

More specifically, we design the following four interfaces, from a high-level interface to a low-level one.

Interface 1: The common module do not process XPath at all. It just invokes wrappers corresponding to the target data format, and it passes given XPath queries to them. The wrappers process XPath queries, and return the result to the common module. In this approach, the programs of wrappers include the codes for the process common to all the data formats, i.e. the core part of XPath evaluations. Therefore, the cost of the development of wrappers is high.

Interface 2: The common module decomposes the given XPath query into a set of subexpressions in a subclass of XPath, which are called "simple paths." Each simple path is translated to an operation to move a file pointer (and sometimes read the data at the file pointer), and given to wrappers. Then, the wrappers execute that operation, return the result to the common module, and wait for the next instruction. In this interface,

one operation in the interface corresponds to multiple navigation steps on XML trees.

Interface 3: The common module decomposes the given XPath query into a set of "steps," which is a basic unit of XPath expressions. Each step is translated to an operation to move a file pointer (and sometimes read the data at the file pointer), and given to wrappers. Then, the wrappers execute that operation, return the result to the common module, and wait for the next instruction. In this interface, one operation in the interface corresponds to one navigation step on XML trees.

Interface 4: Wrappers do not include any processing that depends on the given query. They always process entire target data, and provide the common module with views as if the data are XML data. More specifically, in this research, we use SAX interface, which is a widely-used standard interface for accessing XML data, and a wrapper outputs a sequence of "SAX events" which correspond to the occurrences of opening/ending tags of elements, attributes, and so on. In this approach, wrappers always process whole target data, and therefore, query evaluation can be highly inefficient, e.g. when extracting only small data appearing at the beginning of huge data.

Among those interfaces, we have implemented 1, 3, 4. The experimental results show that the interface 3 achieves both more efficient query processing and lower memory usage. Thus, we have confirmed that the interface 3 is more appropriate than the interface 1 or 4.