Compact Data Structures for Querying XML

Mathias Neumüller

Department of Computer and Information Science
University of Strathclyde, Glasgow G1 1XH, Scotland, U.K.
mathias@cis.strath.ac.uk

Abstract. XML is of growing importance in a range of computer applications. In addition to being a document exchange format it is now commonly used for data storage and retrieval as well. While XML offers great potential to unite data exchange and storage, it is expensive to process data stored in textual format. The document object model (DOM) defines convenient means to access XML data, but many implementations struggle with performance limitations. The data model implied by the standard needs new data structures and adapted query algorithms to enable native XML databases to perform acceptably. This introduction describes a preliminary data structure developed with these objectives in mind. It also gives an overview of our future research directions and anticipated problems.

1 Introduction and Research Objectives

XML is of great importance in the domain of document management and is now emerging as an alternative to conventional database approaches, particularly in the case of less regularly structured data [1]. Further development of this capability requires that one must be able to query XML data sources with acceptable performance. Many database suppliers offer extensions to their RDBMSs that enable the data storage and retrieval in XML format, but the primary data model of these engines remains table-orientated. Emerging native XML databases (NXDs) [2] offer a tailor-made storage solution for XML, but their implementation technology is as yet immature compared to that of the established RDBMSs.

It is our belief that potential difficulties originate at the access level. While large volumes of XML data stored in textual form are hard to access, mappings to a relational database typically result in a large number of performance limiting joins [3]. Common in-memory representations suffer from excessive memory requirements while disk-based system suffer from poor response times. It is the goal of our research to combine elements of these different approaches to find a method suitable to handle large XML databases with acceptable performance.

2 Related Work

Significant research has already been carried out in the area of integrating XML data into relational [3] and object orientated databases [4]. Several mapping

schemata exist that allow storage of hierarchical XML into relational tables or object stores. Little research has been done in the area of compact representation of XML data despite of the fact, that this could help to move more of the processed data from external into internal storage. Similar approaches in the relational field have shown good results [5]. The semistructured, hierarchical nature of XML makes it harder to treat than two-dimensional, strongly typed relational data. Most implementations of the DOM [6] so far have concentrated on extensive functionality rather than on compact representations. Thus those implementations can only handle relatively small documents. PDOM [7], which is a persistent DOM implementation that supports queries using XQL[8], allows access to elements of larger documents. The data is held on disk and copied into a buffer memory for processing. Compression is used for disk storage but not in memory. Xindice [9], a native XML database, supports compression of individual elements, both in main memory and in secondary storage. The compression is limited to the tokenisation of element and attribute names and primarily used to increase the speed of querying. However, redundancy present in the data itself is not utilised, thus the compression achievable is moderate. Compression for document-centric XML is provided by some systems such as Tamino [10]. All of the designs mentioned are aimed at relatively small documents, though Xindice and Tamino allow documents to be grouped into collections and support queries across these collections.

3 Design

The overall task of finding a suitable combination of data structure and query algorithm is tackled in a bottom-up fashion. It is our believe, that a compact in-memory representation of semistructured data will offer opportunities for high-performance query algorithms. To date we already have developed a DOM compliant compact data structure based on dictionary substitution. The following sections describe some design decisions and early results from this work. It is described in more detail in [11]. The basic design combines two approaches to achieve a compact representation: dictionary substitution for the occurring strings and minimising of the number of objects required to represent the DOM tree. Although the created structure works with any well-formed XML document, *data-centric* XML files were assumed for optimal compression.

Dictionary substitution is a well understood, relatively simple compression mechanism [5]. Every occurring data word is stored in a dictionary and represented by a short binary token. Every occurrence of this word in the source document is replaced by its corresponding token. This compression technique is especially suitable for XML as its verbosity requires every piece of information to be expressed in textual form. In terms of our DOM, a data word is the name or the value of an attribute, an element name or any character data section. This means that the metadata contained in the element tags is compressed using the same compression technique as used for the data. Our system builds a separate dictionary for every domain, with the type of a node serving as primary domain.

The domains of character data, attribute names and values are additionally subdivided by the name of their containing or direct parent element, hence allowing the storage of related information in one dictionary and limiting the size of the individual dictionaries. A programming technique that minimises the number of objects held in memory to a minimum was used. The structure of documents is stored in an array of integers rather than as a tree of objects [11]. This approach is closely related to the flat, textual representation of XML. The structure is given by corresponding start and end tags. It is reasonably easy to translate this into a series of tokens as used for example for the binary WAP format [12]. The document structure can then be stored in a simple array of integers.

4 Implementation

The designed system, the *Dictionary substitution based DOM implementation* (DDOM), was implemented in Java. The object minimisation technique used saves memory but may also result in excessive processing power requirements to support the frequent generation of objects. To minimise this, internal methods do not use the methods provided by the DOM interface, which return *Node* objects. Instead they work directly on the compressed representation. Tree traversals are translated into short linear searches over the structure array.

5 Measurements

Initial measurements confirm that significant savings can be achieved. Figure 1 shows the measured memory requirements of a domain name server (DNS) database. The memory requirements of this real-world example lie 30–80 % below those of typical DOM implementations. Both Crimson and Xerces showed linear growth in memory consumption for files above a certain threshold. The memory consumption of the DDOM grew less than linear, but depended on the redundancy present in the data. However, data compressed using the current implementation still requires 1–2 orders of magnitude more memory than its gzip compressed textual representation, which was used as a practical measurement of the entropy.

6 Conclusions and Future Work

The prototype implementation demonstrates a significant space saving compared with standard DOM implementations. This could be improved using a better implementation of the dictionaries. The influence of the type of data needs to be analysed more closely, using a variety of possible sources. Currently the compression mechanism works at document level. To make this technique useful for NXDs, it will need to work on collection level [2] to allow queries across several documents. More work will be required to analyse the query performance of this and other solutions. Therefore the next step of our work aims to provide query

support for DDOM. We have currently two strategies to achieve this. One approach involves the integration of the compression technique used with an NXD. The alternative is to adapt the DDOM to work with an external query engine.

References

[1] Ceri, S., Fraternali, P., Paraboschi, S.: XML: Current developments and future challenges for the database community. In Zaniolo, C., Lockemann, P.C., et al., eds.: EDBT 2000. Volume 1777 of LNCS., Springer (2000) 3–17

[2] Staken, K.: Introduction to native XML databases. XML.com (2001) `http://www.xml.com/pub/a/2001/10/31/nativexml.html`.

[3] Shanmugasundaram, J., Tufte, K., et al.: Relational databases for querying XML documents: Limitations and opportunities. In Atkinson, M., Orlowska, M.E., et al., eds.: VLDB 1999, Morgan Kaufmann (1999) 302–314

[4] Renner, A.: XML data and object databases: The perfect couple? [13] 143–148

[5] Cockshot, W.P., McGregor, D., Wilson, J.: High-performance operations using a compressed database architecture. The Computer Journal **41** (1998) 283–296

[6] World Wide Web Consortium: Document Object Model (DOM) Level 1 Specification Version 1.0. W3C recommendation 1 october, 1998 edn. (1998) `http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001`.

[7] Huck, G., Macherius, I., Frankhauser, P.: PDOM: Lightweight persistency support for the document object model. In: Proceedings of the 1999 OOPSLA Workshop "Java and Databases: Persistence Options", Denver, CO, USA (1999)

[8] Robie, J., Lapp, J., Schach, D.: XML query language (XQL). Workshop proposal, W3C (1998) `http://www.w3c.org/TandS/QL/QL98/pp/xql.html`.

[9] Bradford, T.: dbXML XML database application server version 0.4. dbXML core technical specification, dbXML Group, L.L.C. (2000) "Xindice" since Dec 2001.

[10] Schöning, H.: Tamino – a DBMS designed for XML. [13] 149–154

[11] Neumüller, M.: Compression of XML data. MSc thesis, University of Strathclyde, Glasgow, Scotland, UK (2001)

[12] WAP Forum members IBM, Motorola and Phone.com: WAP binary XML content format. W3C Note (1999) `http://www.w3.org/TR/wbxml/`.

[13] IEEE Computer Society: Proceedings of the 17th International Conference on Data Engineering, April 2–6, 2001, Heidelberg, Germany. In: ICDE 2001, Heidelberg, Germany, IEEE Computer Society (2001)

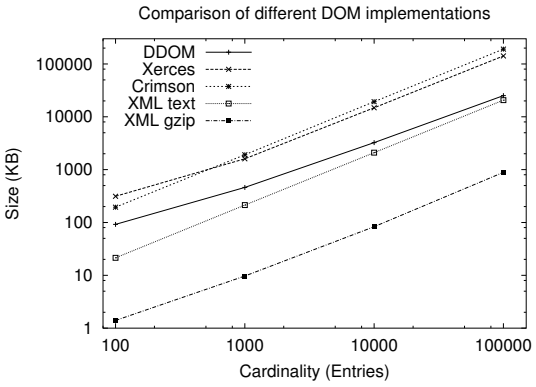

Fig. 1. Memory consumption of the different DOM implementations for the DNS database. Note that the textual XML files use 8 bit character encoding whereas all DOM implementations store characters using 16 bits according to the W3C recommendation. The text files were subsequently compressed using the Linux version of gzip 1.8.4 with maximum compression.