ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Information and Software Technology 46 (2004) 753–759

INFORMATION AND SOFTWARE TECHNOLOGY

www.elsevier.com/locate/infsof

The space efficiency of XML

Ramon Lawrence*[,1]

IDEA Lab, Department of Computer Science, University of Iowa, 201L MacLean Hall, Iowa City, IA 52242, USA

Received 8 February 2003; accepted 11 February 2004
Available online 12 March 2004

Abstract

XML is the future language for data exchange, and support for XML has been extensive. Although XML has numerous benefits including self-describing data, improved readability, and standardization, there are always tradeoffs in the introduction of new technologies that replace existing systems. The tradeoff of XML versus other data exchange languages is improved readability and descriptiveness versus space efficiency. There has been limited work on examining the space efficiency of XML. This paper compares XML to other data exchange formats. Experiments are performed to measure the overhead in XML files and determine the amount of space used for data, schema, and overhead in a typical XML document.
© 2004 Elsevier B.V. All rights reserved.

Keywords: XML; Relational database; Character-separated files; Data exchange; Space efficiency

1. Introduction

XML is rapidly replacing existing technologies as a medium for data exchange between systems. XML is a standardized, portable, human readable, flexible, and self-describing data format language. For many applications, it is a clear improvement over proprietary binary formats and non-standardized text files. XML is replacing these older technologies. However, like any new technology, XML has benefits and shortcomings with respect to the technologies that it is replacing. There has been very little attention paid to some of the tradeoffs associated with XML. This paper will focus on one tradeoff as it relates to the exchange of information between relational databases. The tradeoff of XML versus other technologies such as character-separated files is that the verbosity introduced by the descriptive tags of data elements improves readability and semantic exchange at the sacrifice of increased file size.

This paper reports on experiments comparing character-separated files and XML files for encoding information in relational databases, and experiments testing how space is utilized in XML documents. In Section 2, we motivate why the descriptiveness versus efficiency tradeoff is important for XML, and Section 3 provides some background information on XML encoding of relational databases. The major contribution of the work is in Section 4, where XML is compared with other file formats with respect to theoretical space efficiency of database encoding. To understand the efficiency of real-world XML documents, two separate statistics gathering programs are created. The first program, `DBstats`, uses JDBC to gather statistics on relational databases. The second program, `XMLstats`, uses a SAX-compliant parser to determine XML document statistics including tag name sizes and percentage of the document size devoted to XML overhead, schema information, and data. The experiment results provide interesting, and previously undocumented, statistics on XML usage. These results are then used to discuss the benefits and shortcomings of migrating to XML.

2. Motivation

Despite all its benefits, XML is not always the perfect data encoding language for all applications. The major limitation of XML is that it is very space inefficient, especially for regular data sets. There has been limited work on evaluating the efficiency of XML, even though there still remains applications and environments (such as wireless communications and scientific data sets) where space efficiency is still an important concern. To understand

* Tel.: +1-319-335-0561.
E-mail address: ramon-lawrence@uiowa.edu (R. Lawrence).
[1] http://www.cs.uiowa.edu/~rlawrenc

the efficiency of XML, one must determine the properties of element and attribute names used as tags. Investigating several real-world XML documents and analyzing their space efficiency provides information to potential XML users about the tradeoffs they may encounter when using XML for their application.

3. Background

One of the common uses of XML is to transmit data originally stored in relational databases. There has been previous work on publishing data as XML documents [8] where the focus is on how to efficiently perform the actual transformation by exploiting the relational database engine. Other work [5] involves making the translation between relational databases and XML more space efficient. Although a trivial transformation is always possible by mapping each tuple into an XML element and each tuple attribute into subelements, this is not always the most intuitive or space efficient choice. Nesting-based Translation (NeT) [5] improves on a flat translation by nesting some attributes of a single relation in the hierarchical XML structure. Nesting reduces data duplication if multivalued dependencies are present within a relation. An extension called CoT [5] allows nesting across relations by exploiting inclusion dependencies. Nesting results in smaller documents by reducing redundancy in the XML encoding and has the combined benefits of improving readability and space efficiency.

None of the current work is specifically focused on the space efficiency of the entire XML document. The missing piece is that space efficiency is directly impacted not only by the XML tag encoding and nesting, but also by the names chosen for tags. Thus, space efficiency dramatically decreases if tag names are long strings. Shortening the tag name length increases space efficiency, at the price of human readability. Understanding this tradeoff is critical as XML is designed to be both human and machine processable. Authors of XML documents should be aware of the impact of their naming decisions based on the amount of human interaction with the XML document. This research aims to provide such insights.

4. Space efficiency analysis of XML

In this section, we provide formulas for estimating the space efficiency of XML, character-separated (tab or comma) (CSV) files, and fixed-sized (FSV) files for encoding a single database relation. Estimating the encoding size of an entire database can be performed by applying the formulas repeatedly to each relation. We will assume that only attribute types that can be readily text encoded (string, int, float, double, date) will be translated, and others such as binary large objects (BLOBs) are transmitted using other means.

Format	*Overhead per Attribute*
CSV	1 (or 3)
FSV	S - D
XML (encode empty)	2*N+5
XML (no empty)	(2*N+5)*(1-E)

Fig. 1. Overhead per attribute for file formats.

4.1. Analysis

Overhead per attribute a, denoted as OV(a), is defined as the extra space required to delimit the attribute from others in the record (and file). Define the size in bytes of the data value stored in a as $D(a)$ or just D when a is understood. Let $S(a)$ or simply S denote the maximum schema size of a in bytes. Let $N(a)$ (or N) denote the size of the name for attribute a. Finally, let $E(a)$ or E denote the fraction of data instances of attribute a that are `NULL`. Given these definitions, it is possible to define formulas calculating the overhead per attribute in the various file formats (see Fig. 1).

Interestingly, only a CSV file has a constant overhead per attribute. This overhead is typically one character as a single comma or tab separates data values. Optionally, quotation marks can be used to denote strings which would increase the overhead to three characters. For FSV files, overhead is proportional to the amount of space required to store the data with respect to the maximum allocated size. Encoding a relational attribute in XML using an element has overhead based on the size of the attribute name.[2] We have displayed two possible formulas for XML files as some overhead savings can be achieved by not creating an element if the attribute value is `NULL`. The analysis assumes relational attributes are coded as XML elements instead of using XML attributes. The size of encoding a relation R in each of the formats is the sum of the formula values for each attribute $a \in R$ (assumes no nesting).

From these formulas, it is obvious that when encoding individual attributes, XML will always have more overhead than CSV files. However, it is less clear the efficiency of XML with respect to FSV files. For XML to be more efficient than FSV files, $N \leq (S - D - 5)/2$. We perform some experiments to determine the values of N, S, D, E in practice.

4.2. Experimental results

Two separate experiments are performed. The first experiment uses a Java program called `DBstats` that

[2] The constant 5 represents the five occurrences of the three tag characters (‘ > ’, ‘ < ’, ‘/’) required for the open and close tags.

	Schema Level		Data Level				
Database	**Name Size**	**Field Size**	**Name Size**	**Field Size**	**Data Size**	**% Usage**	**% Empty**
Northwind	8.79	19.27	8.96	15.24	6.86	45.00%	2.72%
Car	6.00	8.86	6.00	8.86	3.29	37.12%	0.00%
Energy	11.49	33.01	15.11	20.10	6.01	29.90%	4.10%
English	9.04	26.29	9.04	26.29	3.94	14.99%	29.48%
Flare	8.92	1.92	8.92	1.92	1.00	52.00%	0.00%
Census	7.49	10.00	4.00	10.00	6.55	65.54%	0.00%
Marriage	6.76	40.40	6.07	47.99	5.53	11.51%	21.09%
Medrefs	4.94	51.07	3.23	47.06	24.38	51.80%	4.46%
Pioneer	7.77	21.11	6.92	10.11	5.58	55.17%	1.88%
Average	7.91	23.55	7.58	20.84	7.01	40.34%	7.08%
Median	7.77	23.55	6.92	15.24	5.58	45.00%	2.72%

Fig. 2. Database statistics for sample databases.

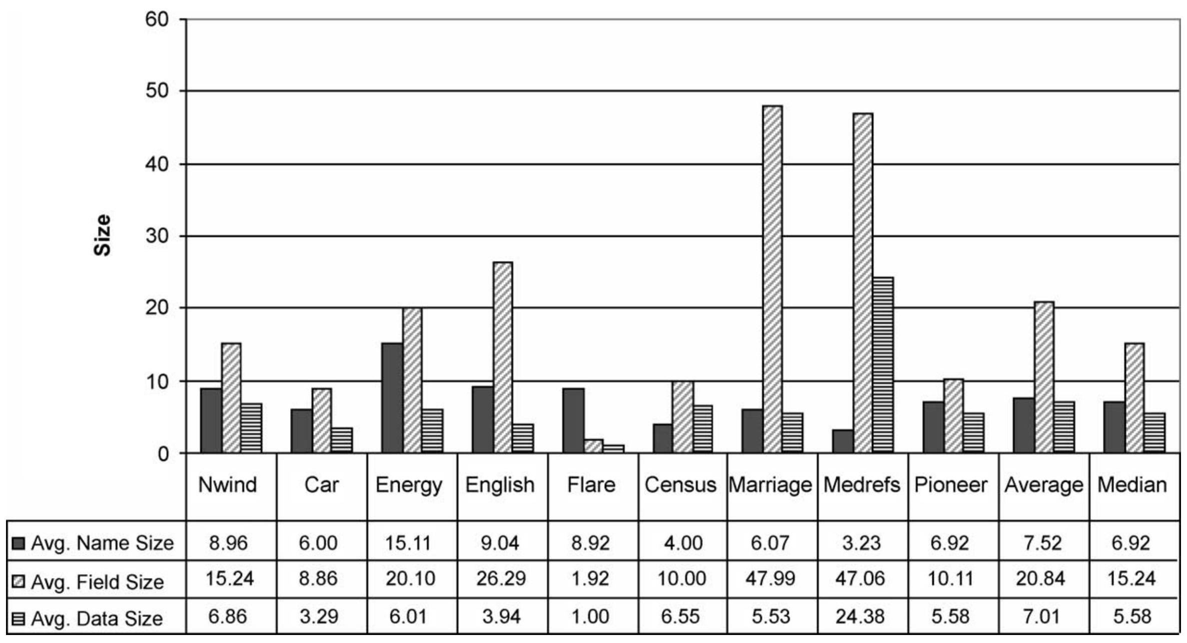
connects to sample databases with JDBC and extracts summary statistics. The statistics include average name size (N), schema field size (S), data size (D), and fraction of `NULL` fields (E). The sample databases include the Northwind database of Microsoft Access, some databases from the UCI KDD [4] and ML [1] repositories, and other databases publicly available on the WWW.

The results from this experiment are summarized in Fig. 2. Statistics are gathered at the schema (extensional) and data (intentional) levels. The usage column is the percentage of space used by the data with respect to the maximum size allocated in the schema definition. Although not shown due to space constraints, the usage percentage is relatively constant across all data types (int, varchar, etc.). The final column is the percentage of fields in the database that are `NULL`.

Interesting findings include that the average field name size (N) is about eight characters long (see Fig. 3), and that fields are typically only about 30–40% full. These statistics allow comparisons between the file formats in terms of overhead per attribute (see Fig. 4).

The experiment studies encoding a highly structured database into XML. It is not surprising that XML exhibits significant overhead as it was designed for more semi-structured data and less concerned with space efficiency. Further, the overhead calculations were biased to CSV and FSV because the data sets contained limited redundancy and no XML nesting was used. XML may outperform these formats when the data is less structured, contains longer text fields, and contains redundancy.

The second experiment examines the space characteristics of typical XML documents. The sample documents



	Nwind	Car	Energy	English	Flare	Census	Marriage	Medrefs	Pioneer	Average	Median
■ Avg. Name Size	8.96	6.00	15.11	9.04	8.92	4.00	6.07	3.23	6.92	7.52	6.92
▨ Avg. Field Size	15.24	8.86	20.10	26.29	1.92	10.00	47.99	47.06	10.11	20.84	15.24
▤ Avg. Data Size	6.86	3.29	6.01	3.94	1.00	6.55	5.53	24.38	5.58	7.01	5.58

Fig. 3. Average name, field, and data sizes.

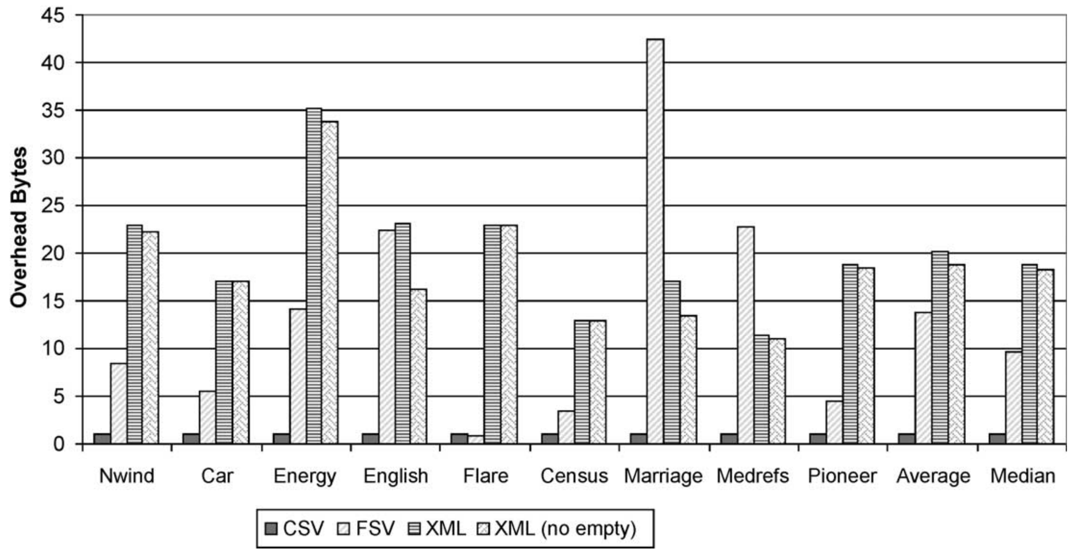


Fig. 4. Overhead per file type.

XML File	Name	Tag	Attr	%OH	%DT	%SC	%WS	Ratio	File (KB)	Time (ms)
auction.xml	6.33	8.88	6.67	7.69	71.96	18.66	1.69	0.37	113794	15582
allPlays.xml	5.03	7.53		11.75	61.39	23.65	3.22	0.58	7710	1672
hamlet.xml	5.03	7.53		11.87	61.03	23.87	3.23	0.59	282	380
profiles.xml	7.33	9.83		11.38	55.04	33.39	0.19	0.81	47	200
dblp.xml	5.85	8.35	3.35	13.73	53.18	30.32	2.77	0.83	137138	25426
sprotall.xml	5.09	7.63		19.81	38.63	33.68	7.88	1.15	161363	34930
sigrecord.xml	6.53	9.03	8.00	15.22	32.17	37.82	14.80	1.65	468	520
personal.xml	5.21	7.75	4.94	20.96	27.89	38.86	12.28	2.14	2	150
dept.xml	6.12	8.62		17.89	26.06	43.77	12.28	2.37	71	330
quotes.xml	6.58	9.08		21.50	21.69	56.58	0.23	3.60	39	270
nwind.xml	8.26	10.77		18.90	18.48	62.61	0.00	4.41	712	350
Average	6.12	8.64	5.74	15.52	42.50	36.66	5.32	1.68	38330	7255
Median	6.12	8.62	5.81	15.22	38.63	33.68	3.22	1.15	468	380

Fig. 5. XML file usage breakdown.

come from a wide-variety of sources with a sampling of both document-centric and data-centric documents. A complete list of the XML documents and their sources are shown in Appendix A. The experiment uses a Java program called `XMLstats` and the `Xerces` parser to extract XML document properties using SAX. `XMLstats` determines statistics on average tag name length (*N*), and a breakdown of the document size into the categories of data, schema, overhead, and whitespace. Data is defined as attribute and element values. Schema is the tag and attribute names. Overhead includes the characters used to denote tags such as '$>$', '$<$', '/', and quotes to denote attribute values. Whitespace is the portion of the document flagged by the parser as ignorable whitespace, end-of-line characters, and strings consisting of all spaces.[3] XML header information and processing instructions are not classified in any of the categories.

The experimental data is in Fig. 5. The first three columns denote the average size of tag names, the entire tag (including '$>$', '$<$', '/'), and attribute names (if present). The next four columns contain the document percent breakdown in terms of overhead (OH), data (DT), schema (SC), and whitespace (WS). An overhead ratio, defined as the overhead plus the schema size divided by the data size, is a useful metric. The program has good performance on a AMD 1.4 GHz machine as the maximum execution time was 35 s. In Fig. 6, the percentage of the file used for data, schema, overhead, and whitespace is displayed.

Note the distinction between document-centric and data-centric documents. Document-centric documents such as the Shakespeare plays consist of approximately 60% data, and the XML overhead is reasonable. For highly

[3] Not all whitespace is flagged by the Xerces parser (some end-of-line characters are often missed), so whitespace is slightly more than reported.

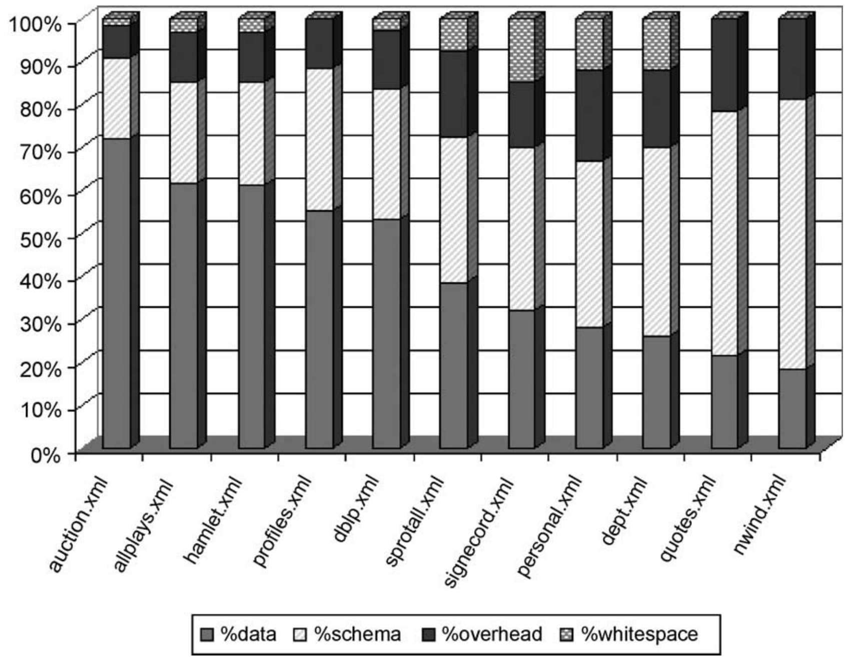


Fig. 6. XML file usage breakdown.

data-centric documents, like the simple XML encoding of the Northwind database, the data percentage is very low (from about 20 to 30%), and the overhead ratio is above 1. Structured documents such as auction.xml (from the XMark benchmark [7]), dblp.xml, and sigmodrecord.xml vary from 30 to 70% data content. Although the data itself is mostly structured, the amount of data content varies widely due to the presence of very large text fields. For example, the two publication-related XML documents contain large fields for keywords, abstract, and title. The data content of auction.xml is surprising, and results from large strings of random keywords and descriptions in the data set. Since this data is artificially generated, it is less representative of actual XML use than the other samples.

5. Discussion

Migration to XML has several benefits including standardization, validation, and commercial support. However, the space inefficiency and overhead cannot be overlooked for some applications. Below we summarize some key points uncovered.

XML is always less efficient than character-separated files for transferring single database tables, and is often less inefficient that even fixed-sized files. Schema repetition becomes a considerable overhead. Highly data-centric documents may have two to four times overhead with respect to data. This is especially important for large, regular data sets, such as scientific data. Conversions of such data sets to XML may not be practical or beneficial. Even by exploiting compression [9], the larger file size makes all operations on the XML document less efficient including the compression, querying, and display.

Due to the large amount of schema redundancy and overhead in XML documents, very high compression ratios are possible. For instance, compression of some of the sample documents in the experiments using the `gzip` utility resulted in compressions of: nwind.xml (90%), flare.xml (98%), allPlays.xml (72%), sprotall.xml (88%), and dblp.xml (81%). However, the compressed version of flare.xml was still 93% larger than the compressed version of the raw data file in CSV format. The uncompressed version was over 13 times larger. Although the effectiveness of compression mitigates some of the issues in the verbosity of XML encoding, compressed XML files are still larger than the compressed form of more space efficient formats. Further, the compression/decompression and translation to and from XML consume valuable processing resources. Conversion of the census data from database form (606 MB) into XML form, census.xml (3878 MB), took about an hour, and compressing the file took 195 s. The compressed file size was 146 MB. In comparison, converting the same data to CSV format took 18 min and the file was 362 MB (uncompressed) and 65 MB (compressed in 45 s). Compression by itself cannot eliminate all the negatives of the overhead in XML encoding because ultimately applications must process the XML in uncompressed form, which consumes valuable CPU/memory resources.

The overhead in constructing XML documents and their compression may be an issue with web servers that must compress dynamically generated content for transmission using HTTP 1.1. Studies [6] have confirmed the benefits of compressing HTML pages, and the benefits of XML compression are even larger due to the high, repetitive

Table A1
Data source list

Source name	Description	Author	Source location
Northwind		Microsoft	Comes standard with Microsoft Access
Car		M. Bohanec	UCI ML (http://www.ics.uci.edu/~mlearn/)
Energy	World energy	EIA	http://www.eia.doe.gov/emeu/world/main1.html
English	English verbs	R. Leeuwen	http://www.auboutdumonde.com/conjugeng.htm
Flare	Solar flares	G. Bradshaw	UCI ML (http://www.ics.uci.edu/~mlearn/)
Marriage	Marriage records	R. Neep	http://freereg.rootsweb.com/howto/general/download.html
Medrefs	Medical references	M. McGoodwin	http://www.mcgoodwin.net/pages/medrefs.html
Pioneer	Pioneer Wells	C. Asiala	http://www.geo.mtu.edu/svl/download/pioneer/
Census	1990 US census	C. Meek	UCI KD (http://kdd.ics.uci.edu/)

overhead when transmitting structured data using XML. XML encodings transferring large amounts of data should always be compressed because of the huge savings in space and transmission time. However, the point is that some data should never be converted into XML in the first place because of its inefficiencies.

XML becomes more efficient and practical if data is hierarchical in nature and can be nested. In some respects, these experiments validate the work on finding optimal DTDs that exploit nesting [5] for mapping relational databases to XML documents. Nesting has the potential to reduce overhead by eliminating tags. Optimal nesting of the data in the Northwind database results in the elimination of 43% of foreign key values and 12% of all data values. Thus, nesting does save space over flat translation, but not enough to compensate for the high overhead of XML. Determining the optimal nesting by data extraction is a very costly operation. Further, nesting amounts to producing a view on the data source. A view may make it easier to visualize the data hierarchically, but may also complicate the parsing and loading of the data if the purpose of the XML document is simply a transmission format between two relational databases. The major benefit of nesting is to reduce redundancy produced when encoding a query result in XML that spanned multiple tables. CSV and FSV files will contain significant redundancy as the query result is no longer normalized. Future work involves determining when the transition from CSV to XML is more space efficient based on the ability to nest data in XML to control redundancy.

There is an interesting tradeoff between overhead and the size of tag names. Between 20 and 60% of the document consists of tag names. This overhead can be reduced by shortening tag names, at the price of less readability. Currently, most XML tag names are approximately six characters long (not including namespaces) which is even shorter than average database field names (eight characters). Such name condensing makes the tag less understandable and partially defeats their purpose (so users can understand the data).

XML is human readable as a plain text file, but the growing use of DOM and SAX is making this less of a possibility. The XML encoding of Northwind eliminated almost all whitespace at the price of not being able to view it in any useful way in a regular text editor. Even though XML viewers resolve this issue by parsing the document into a navigation tree, not being able to read the file using a regular text editor is a negative. Further, it can be argued that it is easier to read CSV and FSV files because of their regular organizations instead of navigating a tree to find the appropriate data.

Alternate XML formats [3] such as JavaScript Object Notation (JSON) [2] have been proposed to make XML easier to read and edit with standard text editors. A drawback with these formats is that they are not standard XML, which mitigates the major benefit of XML. Further,

Table A2
XML source list

Source name	Description	Source location
auction.xml	XMark benchmark	http://monetdb.cwi.nl/xml/index.html
allPlays.xml	All Shakespeare plays	Jon Bosak (http://www.ibiblio.org/bosak/)
hamlet.xml	Hamlet	Jon Bosak (http://www.ibiblio.org/bosak/)
profiles.xml	Company profiles	UW Niagara data sets (http://www.cs.wisc.edu/niagara/data/)
dblp.xml	DBLP bibliography	Michael Ley (http://dblp.uni-trier.de/xml/)
sprotall.xml	Swiss protein DB	UW Niagara data sets (http://www.cs.wisc.edu/niagara/data/)
sigmodrecord.xml	SIGMOD record bibliography	UW Niagara data sets (http://www.cs.wisc.edu/niagara/data/)
personal.xml	Employee/supervisor	Unknown
dept.xml	University department	UW Niagara data sets (http://www.cs.wisc.edu/niagara/data/)
quotes.xml	Yahoo quotes in XML	UW Niagara Data Sets (http://www.cs.wisc.edu/niagara/data/)
nwind.xml	Northwind	Translated without nesting

although small space savings may be achieved, that benefit is secondary to the design goal of making the documents easier to read and edit. JSON is a lightweight data-interchange format that is based on a subset of the JavaScript programming language.

JSON is optimized for structured data transmission and does provide some space savings as closing XML tags are replaced with a single closing bracket.

For researchers and practitioners of XML, we thought it useful to summarize some of the important results found.

- Database field names are on average eight characters long.
- Database fields are on average 40% full with respect to their maximum size.
- The percentage of NULL fields is typically small (around 5%), although much higher percentages are possible if the database is not in fourth normal form.
- CSV files are almost always more efficient than XML for single tables. FSV and XML are roughly overhead equivalent, but differ widely depending on data set properties.
- CSV and FSV files are much more efficient for large, regular data sets, which probably should not be migrated to XML.
- XML tag names are on average six characters long.
- Document-centric XML documents on average consist of approximately 60% data, 25% schema, 10% overhead, and 5% whitespace.
- Data-centric XML documents on average consist of approximately 25% data, 50% schema, 20% overhead, and 5% whitespace.
- For transmission of normalized data, nesting XML on foreign keys reduces the overall document size by eliminating the encoding of many of the foreign keys and a small percentage of all data values.
- Compression of large XML documents should be performed due to the very high compression ratios resulting from the redundancy in XML encoding. Web servers should be configured to compress XML content for delivery using HTTP 1.1 as most browsers will automatically decompress the content.
- Compression cannot fully compensate for the inefficiencies of encoding regular data sets in XML because processing resources used for compression and decompression and XML generation and parsing must also be considered.

6. Conclusions

Although XML is the data exchange standard, there is an implicit tradeoff in XML between space efficiency and descriptiveness. Using tag names allows the data to be self-documenting, but the repetition of tag names can become a significant overhead. Although most applications are not affected by the additional overhead in XML documents, certain applications involving large, regular, data sets will incur a significant overhead and performance penalty by encoding in XML. This research has shown how the space efficiency of XML compares to character-separated and fixed-sized files for database table encoding. Future work involves determining the efficiency for general queries that may contain redundancy. The major contribution of the work is the first study of XML document properties including tag name sizes, and document characteristics with respect to data, schema, and overhead. These statistics and the accompanying formulas can be used by XML practitioners to determine the size of encoding their particular data sets into XML.

Appendix A

Tables A1 and A2

References

[1] C.L. Blake, C.J. Merz, UCI repository of machine learning databases, http://www.ics.uci.edu/~mlearn/MLRepository.html, 1998.

[2] D. Crockford, JavaScript Object Notation (JSON), http://www.crockford.com/JSON/index.html

[3] E. Dumbill, Exploring alternative syntaxes for XML, http://www-106.ibm.com/developerworks/xml/library/x-syntax.html

[4] S. Hettich, S. Bay, The UCI KDD Archive, http://kdd.ics.uci.edu, 1999.

[5] D. Lee, M. Mani, F. Chiu, W. Chu, NeT & CoT: translating relational schemas to XML schemas using semantic constraints, in: Proceedings of the 11th CIKM, 2002, pp. 282–291.

[6] S. Radhakrishnan, Speed Web delivery with HTTP compression, http://www-106.ibm.com/developerworks/web/library/wa-httpcomp/

[7] A. Schmidt, F. Waas, M. Kersten, M. Carey, I. Manolescu, R. Busse, XMark: a benchmark for XML data management, in: VLDB, 2002.

[8] J. Shanmugasundaram, E. Shekita, R. Barr, M. Carey, B. Lindsay, H. Pirahesh, Efficiently publishing relational data as XML documents, in: VLDB, 2000, pp. 65–76.

[9] P. Tolani, J. Haritsa, XGRIND: a query-friendly XML compressor, in: ICDE, 2002.