A NEW APPROACH TO DEVELOPING A DATA EXCHANGE LANGUAGE AS AN ALTERNATIVE TO XML

Erdinç UZUN
erdinc@trakya.edu.tr
Erdem UÇAR **Yılmaz KILIÇASLAN**
erdemu@trakya.edu.tr yilmazk@trakya.edu.tr
University of Trakya
Deparment of Computer Engineering

Abstract

Today XML is one of the most extensively adopted languages for data exchange. XML is rapidly replacing existing technologies as medium for data exchange between systems. As any newly emerging technology XML has certain tradeoffs compared to existing ones. Among its benefits are its increased readability, standardization and descriptiveness. As for its shortcomings, it fails to display a space efficient behavior (8). This study offers an initial and rough design of an alternative data exchange language that is intended to preserve the descriptiveness and readability of XML while circumventing its space deficient side.

Keywords: XML, Data Exchange Language, Space Efficiency

INTRODUCTION

XML is a standardized, readable, descriptive and flexible data exchange language. (3, 6, 7, 8) It is a language that can be quickly grasped by the programmer or researcher and easily embedded in many applications. However, due to some technical reasons (cf. below) it tends to cause considerable overhead in its files. It is clear that this will have an undesirable impact on the time period required for creating a file or accessing data. Furthermore, this will appear as a more serious problem with a network, communication devices that exchange data at a low rate or devices without much memory. As we will shortly see, there are various approaches to overcome this problem. A radical approach, however, could be to develop a new language for data exchange. It is this avenue that we will explore in this and our future studies.

ALTERNATIVE METHODS AND APPROACHES

In this section, we will briefly touch upon various studies that have been carried to increase the efficiency of XML. These studies involve:

- methods for compressing XML documents,
- methods for increasing the time efficiency of querying XML documents, and
- languages alternative to XML.

Transferring big files on the Internet or on any network takes a non-tolerable period of time. To deal with this problem, various compressing techniques have been used. In general terms, one can use a standard text compressor (e.g. gzip www.gzip.org) or an XML-specific data compressor to this effect. (7,9)

Another type of study concerns with developing a structure that can handle queries like those used in the database applications. Depending on the XML infrastructure various methods and algorithms, such as xpath processing algorithms (cf. 5), have been tried to increase the efficiency of such queries.

Among the data transfer languages that can be considered as an alternative to XML are Character or Comma Separated Values (CSV), Fixed-Sized Vectors (FSV), YAML Ain't Markup Language, JavaScript Object Notation (JSON) and

Some Modest Extensible Language (SMEL). Files formatted in either CSV or FSV are rather machine-readable. That is, they fail to satisfy the descriptiveness criterion. Particularly, the CSV and FSV file systems fall short of representing relational databases, which is one of the most crucial benefits of XML (1,8). YAML, JSON and SMEL, on the other hand, are descriptive languages that can handle relational databases, but, like XML, they too are not good candidates for attaining a desirable level of space efficiency. (2,4)

We intend to start a study that will hopefully end up with a language that embodies the advantages of both sides: it will be efficient in terms of data transfer and storage and it will be human readable. However, we are aware of the fact that these two aims can only be reconciled as a result of a tradeoff between the two.

ANALYSIS OF OUR APPROACH

The language we intend to develop comprises two files as XML does:

- Schema File and
- Data File

The schema file is the component where the fields in the data file and the relationships between them are defined. A definition is left- and right-delimited by the characters '<' and '>', respectively. In between these markers, the name of the table coming from the database and the names of the fields in this table are encoded. These pieces of information are preceded by the *table* and *field* keywords. It is also noteworthy that the ',' symbol is used to separate multiple fields.

The data file contains the values of the tables and relevant fields encoded in accordance with the definition specified in the schema file. A significant feature of the language is that it does not use the definitions again in the data file. Only the delimitation symbols, '<' and '>', are used once more. This is the most notable feature of our language distinguishing it from XML: The redundancy due to unnecessarily used tag names is avoided. The tradeoff paid at this point is that both files need to be looked at in order to get a complete understating of their content.

Let us now see an application with two records:

Our Language	XML

Schema File:

<table : Customers <field : CustomerID,CompanyName,ContactName,ContactTitle,Address,City,Region,PostalCode,Country,Phone,Fax> >	<?xml version="1.0" standalone="yes"?> <xs:schema id="NewDataSet" xmlns="" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:msdata="urn:schemas-microsoft-com:xml-msdata"> <xs:element name="NewDataSet" msdata:IsDataSet="true" msdata:Locale="tr-TR"> <xs:complexType> <xs:choice maxOccurs="unbounded"> <xs:element name="Customers"> <xs:complexType> <xs:sequence> <xs:element name="CustomerID" type="xs:string" minOccurs="0" />

	<xs:element name="CompanyName" type="xs:string" minOccurs="0" /> <xs:element name="ContactName" type="xs:string" minOccurs="0" /> <xs:element name="ContactTitle" type="xs:string" minOccurs="0" /> <xs:element name="Address" type="xs:string" minOccurs="0" /> <xs:element name="City" type="xs:string" minOccurs="0" /> <xs:element name="Region" type="xs:string" minOccurs="0" /> <xs:element name="PostalCode" type="xs:string" minOccurs="0" /> <xs:element name="Country" type="xs:string" minOccurs="0" /> <xs:element name="Phone" type="xs:string" minOccurs="0" /> <xs:element name="Fax" type="xs:string" minOccurs="0" /> </xs:sequence> </xs:complexType> </xs:element> </xs:choice> </xs:complexType> </xs:element> </xs:schema>

Data File

<Customers <ALFKI,Alfreds Futterkiste,Maria Anders,Sales Representative,Obere Str. 57,Berlin,,12209,Germany,030-0074321,030-0076545> <ANATR,Ana Trujillo Emparedados y helados,Ana Trujillo,Owner,Avda. de la Constitución 2222,México D.F.,,05021,Mexico,(5) 555-4729,(5) 555-3745> >	<?xml version="1.0" standalone="yes"?> <NewDataSet> <Customers> <CustomerID>ALFKI</CustomerID> <CompanyName>Alfreds Futterkiste</CompanyName> <ContactName>Maria Anders</ContactName> <ContactTitle>Sales Representative</ContactTitle> <Address>Obere Str. 57</Address> <City>Berlin</City> <PostalCode>12209</PostalCode> <Country>Germany</Country> <Phone>030-0074321</Phone> <Fax>030-0076545</Fax> </Customers> <Customers> <CustomerID>ANATR</CustomerID> <CompanyName>Ana Trujillo Emparedados y helados</CompanyName>

	<ContactName>Ana Trujillo</ContactName> <ContactTitle>Owner</ContactTitle> <Address>Avda. de la Constitución 2222</Address> <City>México D.F.</City> <PostalCode>05021</PostalCode> <Country>Mexico</Country> <Phone>(5) 555-4729</Phone> <Fax>(5) 555-3745</Fax> </Customers> </NewDataSet>

It is obvious that our language is much more preferable than XML in terms of space efficiency.

PERFORMANCE MEASUREMENTS

We have measured the time periods required to create files and the sizes of these files. When performing these measurements, we have to use .Net technologies. Resting on these measurements, we will compare some XML instructions found in .Net libraries with those of our language. We have developed our application using the C-Sharp programming language. For performance measurement, we have used the tables in the Nortwind database that come with the SQL server.

File Creation Rates – File Sizes

Nortwind Database (SQL Server)				
Table Name	XML Files		Our Language	
	Creating Time(ms)	Files Size* (KB)	Creating Time (ms)	File Size* (KB)
Customers	108	1,37+37,2	18	0,147+11,4
Orders	241	1,61+ 487	120	0,179+121
Order Details	163	0,938+373	74	0,076+53,8
Products	109	1,31+30,8	20	0,152+4,93

*Files 1. Şema + 2. Data

Let us now compare the data files in terms of tag and data usage:

Formulas for XML
Tag Usage = (Tag names + tag(</>)) / (Tag names + tag(</>) + Data)
Data Usage = (Data) / (Tag names + tag(</>) + Data)
Formulas for New Appoarch
Tag Usage = (tag(<,>)) / (tag(<,>) + Data)
Data Usage = (Data) / (tag(<,>) + Data)

Table Name	XML Files		Our Language	
	Tag	Data	Tag	Data
Customer	%59,85	%40,05	%9,77	%90,23
Orders	%59,39	%40,61	%10,18	%89,82
Order Detail	%74,59	%25,41	%25,45	%74,55
Products	%75,09	%24,91	%17,40	%82,60

CONCLUSION

As a result of our performance measurements, we have detected that the time for creating files is shorter with our new language than XML (approximately from 20% to 50%). Furthermore, we have observed the file size can be considerably shortened with new language (approximately 15%). Tags occupy almost 65% of XML data files. With our language, on the other hand, the ratio of tags appearing in the data files is much lower (about 15%).

To conclude, we have:

- shortened time periods required to create files;
- decreased file sizes;
- diminished the ratio of tag usage in data files and, thereby, increased the ratio of data usage.

Our ultimate aim is to develop a data exchange language as an alternative to XML. As future work, we plan:

- to compare the access time to XML and the access time to our language;
- to speed up the data search and query processes; and
- to complement our approach with methods for data compressing.

REFERENCES

[1] A. R. Schmidt, F. Waas, M. L. Kersten, M. J. Carey, I. Manolescu, and R. Busse. Xmark: A benchmark for xml data management. In VLDB, pages 974-985, 2002.

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

[3] 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.

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

[5] G. Gottlob, C. Koch, and R. Pichler. "XPath Query Evaluation: Improving Time and Space Effciency". In Proceedings of the 19th IEEE International Conference on Data Engineering (ICDE'03), Bangalore, India, Mar. 2003. to appear.

[6] 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.

[7] P. Tolani and J.R. Haritsa. XGRIND: A query-friendly XML compressor. In ICDE, 2002.

[8] R. Lawrence, The Space Efficiency of XML**,** Information and Software Technology Volume 46 Issue 11, September 2004, pages 753-759.

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