International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 2 Issue: 2 319 – 324

XML: Standardized W3C Tool for effective communication between B2B or B2CApplications

[1]J.Shyam Jegadeesh, [2]J.John Spencer, [3]P.Libin Jacob, [4]D.Shine Rajesh

[1,2,3,4]Bethlahem Institute of Engineering, Karungal, Tamil Nadu, India
shyamjegadeesh@gmail.com, john.spencer@hotmail.com, plibinjacob@gmail.com, shinerajesh@gmail.com

Abstract- XML is becoming a standard for communicating the information or data over the internet.The shift from SGML to XML has created new demands for managing the structured documents. Many XML documents will be transient representations for the purpose of data exchange between different types of applications, but there will also be a need for effective means to manage persistent XML data as a database. In this paper we explore the different types of XML documents and XML databases. The purpose of the paper is not to suggest this technology is covering all the features necessary for effective communication of data between applications. Instead the purpose is to initiate discussion of the requirements arising from document collections, to offer a context in which to evaluate current and future solutions, and to encourage the development of proper models and systems for XML database management. Our discussion addresses issues arising from data modeling, data definition, and data manipulation. In future, XML will become the standard for communicating data between business-to-business applications,inventory database access and sharing, integration of commercial transactions, and workflow

Keywords: *XML, Native XML Database, XML Enabled Database, SGML*

*****

1. INTRODUCTION

A common data format is needed to present their data or information, due to the rapid growth of internet. The proprietary data formats were sufficient for managing the information or data within the company or even for communicating data across a fixed number of partners; but this data model was not scalable for a big extent. The advantages of interoperability, represented by the Web, gave rise to the need for a highly standardized common data format for data exchange between the internet applications. Based upon the requirements, the programmers worked very hard and they invented the new technology such as of XML 1.0 [5] and XML 1.1.Organizations that manage complex or diverse structured content have discovered the power of XML for expressing their data in a standard manner. Related standards and tools allow their XML-encoded data to be processed and reused efficiently to satisfy a range of business objectives. Developers are becoming increasingly fluent in using XML, standards related to XML, and a range of platforms and tools to develop XML-based applications.

XML gave a significant boost to web-based and business-to-business (B2B) applications [6]. Data were stored in relational databases and XML was used as a medium to transport data between partners. This model quickly became the de facto standard for deploying applications that managed large volumes of data and either wanted to be able to communicate with other businesses or to expose their data on the webpages. XML is a data model suited for any combination of structured, unstructured and semi-structured data. XML data is easy to extend because new tags can be easily defined as needed. Also, XML documents can easily be transformed into "different looking" XML and even into other formats such as HTML. Furthermore, XML documents can easily be checked for compliance with a schema. All this has become possible through widely available tools and standards such as XML parsers, XSLT, and XML Schema[10]. Beyond XML for data exchange, enterprises are keeping large amounts of business critical data permanently in XML format. This has various reasons. Some businesses must retain XML documents in their original format for auditing and regulatory compliance. Typical examples are legal and financial documents as well as eForms, particularly in the government sector.Another reason for using XML as a permanent storage format is that XML can be a more suitable data model than a relational schema. This is not only true for content oriented applications, but also for certain data-oriented applications.

The now widespread adoption of XML brought on a new trend: storing data in XML format was now an area of new interest. The popularity of XML, a much less complex but still powerful enough subset of SGML [6], helped it gain ground in the application area of SGML itself, including the printing and publishing industries. Along with this shift in use, a new challenge started to become apparent. The XML data produced by these latter applications were no longer small to medium sized documents; however, XML-enabled databases (XED) [11] were not capable of supporting documents of this size, as decomposing and

recomposing huge documents meant a high number of join operations, leading to unacceptable performance, both in retrieving documents and in querying them.This technology gap was covered by native XML databases (NXDs) [11], that is database systems specifically developed for storing XML documents. These platforms include relational database management systems [11] that have continued to add more functionality for managing XML data and content. Even with this continued innovation and improvement, developers are finding that certain content and data processing requirements are best met with a native XML database technology [10]. The fundamental difference between XEDs and NXDs is that the NXDs adopt the XML data model for storing XML data. Much like hierarchical and object databases, they are able to preserve the hierarchy and ordering of nodes of XML documents in a much more efficient manner than XML-enabled databases, hence the tremendous performance improvement in handling large XML documents.

Usually, Relational databases have been offering support for storage, manipulation, search, and retrieval of XML data. This is usually based on storing XML documents in LOBs(data type in RDBMS) or mapping and shredding XML to a relational schema [11]. These solutions have inherent functional and performance constraints. Generally, LOB-based storage [11] allows for fast insert and retrieval of full documents but suffers from poor search and extract performance due to XML parsing at query execution time. This can be moderately improved if indexes are built at insert time. While this incurs XML parsing overhead, it may speed up queries that look for documents which match given search conditions. Yet, extraction of document fragments and sub-document level updates still require expensive XML parsing. Shredding XML to relational tables is expensive at insert time due to costly XML parsing [10] and multi-table inserts. But once XML is broken into relational scalar values, queries and updates in plain SQL promise higher performance.

The paper is organized as follows. The Classification of XML documents is described in Section 2. The anatomy of XML Database is described in Section 3. The Experimental results and discussions are described in Section 4. Finally, the conclusion is explained in Section 5.

2. CLASSIFICATION OF XML DOCUMENT

Before discussing the characteristics of XML database, we should know the details about data-centric and document-centric documents. Even though it is dated, this metaphor usually helps to identify the type of database that is more suitable for a particular application.

2.1Data-Centric Documents

Data-centric documents produced as an import or export format, which are primarily developed for machine consumption. These documents are used for communicating data between organizations or applications between varied platforms. So, XML is used as a common format is terms of convenience, and for reasons of interoperability. Data-centric documents have fairly consistent structure, fine-grained data and no mixed content. So, it can be easily processed by machines. Forexample, thefollowingmemo document isdata-centric:

```
<Books>
<Bookshop CustomerID = "123-1345">
        <Owner OwnerID="1345">
        <OwnerName>Mr.XavierAlwin</OwnerName>
                <address>Dubai,UAE</address>
</Owner>
<PurchaseDate>15/02/2013</PurchaseDate>
<Book>
        <BookName>Visual C++</BookName>
        <Quantity>10</Quantity>
        <UnitPrice>1000</UnitPrice>
</Book>
<Book>
        <BookName>Java</BookName>
        <Quantity>25</Quantity>
        <UnitPrice>550</UnitPrice>
</Book>
</Bookshop>
</Books>
```

Fig 1: A data centric XML document

The above XML document [5] is shown in Fig.1 is clearly data-centric: it has regular structure, fine-grained data and contains no mixed content. For example, in an e-commerce site, the web pages contain large amounts of data, such as the various detailsabout the product; and also the content of the webpages are highly structured and are common for all products. Such webpages, even when designed as a combination of HTML and CSS style-sheets [6], could be easily designed as data-centric XML documents together with XSL style-sheets.

2.2 Document-Centric Documents

Document-centric documents usually designed for human consumption, with examples ranging from books to hand-written XHTML documents. They do not need to have regular structure, have coarse-grained data (that is the smallest independent data unit) and have mixed content. For example, the following memo document is document-centric is shown in fig.2

```
<Product>
        <Name>Laptop</Name>
        <Developer>Dell Inc.</Developer>
        <Summary>Laptops and Desktops.</Summary>
        <Description>
        <Para>Alaptop is a portable personal computer
        with a clamshell form factor, suitable formobile
        use. They are also sometimes called notebook
        computers or notebooks.<Para>
```

```
<Para>You can:</Para>
<List>
<Item>
<Link URL="Order.html">Order </Link>
</Item>
<Item>
<Link URL="dell.org">Read more </Link>
</Item>
</Description>
</Product>
```

Fig 2: A document centric XML document.

This section explained the distinction between data-centric and document-centric documents, a distinction which serves as a good starting point in the world of XML databases [10].

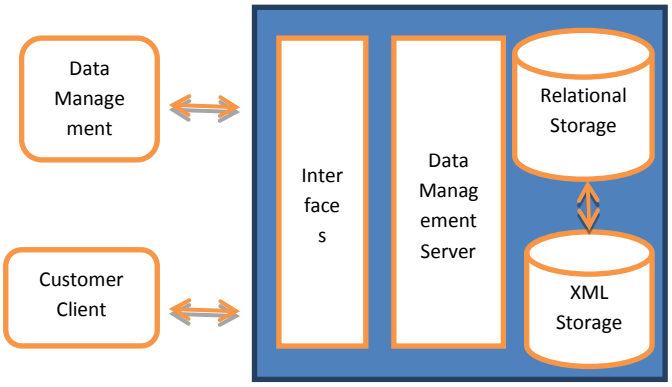

Fig.3 The block diagram of mapping relational schema to XML

It must be noted, however, that this distinction is a bit unrealistic in the sense that it is seldom easy to classify a document in just one of the two categories, since most documents have characteristics from both categories.

3. ANATOMY OF AN XML DATABASE

Systems for managing XML data are either specialized systems designed exclusively for XML or extended ("universal") databases, particularly designed to manage XML data amongst others. The first is referred to as XML enabled databases (XED) and the latterNative XML database (NXD).The XED [11] use the traditional databases in its entirety and only provide XML outputs via transformation of the results.The NXD [11]on theother hand will have a (logical) model for its fundamental unit, stores and retrieves documents according to this model. It can however employ underlying physical storage model based on a relational, hierarchical, or object-oriented database, or use a proprietary storage format such as indexed, compressed files.

3.1 XML-Enabled Databases

The basic use of XML-enabled systems is for publishing the existing relational data [11] as XML. The process of extracting data from a relational database and constructing an XML document is known as publishing or composition. The reverse process i.e. extracting data from an XML document and storing it in the relational database is known as shredding or decomposition. There are two important things to keep in mind before shredding or publishing XML documents. First, an XML-enabled database [10] does not contain XML, i.e. the XML document is completely external to the database. Second, the database schema matches the XML schema, that is, a different XML schema is needed for each database schema.XML-enabled databases generally include software for performing both publishing and shredding between relational data and XML documents.

3.1.1 Relational-to-XML schema mapping

When using an XML-enabled database, it is necessary to map the database schema to the XML schema (or vice versa). The mapping of relational schema to XML storage is depicted in Fig.3. Such mappings are many-to-many. There are two important types:

- Table-based mapping.
- Object-relational mapping.

Table-based mapping: When using a table-based mapping, the XML document must have the same structure as a relational database. That is, the data is grouped into "tuples" and tuples are grouped into "tables".

```
<Database>
<SalesOrders>
<SalesOrder>
<Number>123</Number>
<OrderDate>17-01-2013</OrderDate>
<CustomerNumber>456-783456</CustomerNumber>
</SalesOrder>
</SalesOrders>
<Items>
<Item>
<Number>140</Number>
<PartNumber>Aero-BC-47</PartNumber>
<Quantity>14</Quantity>
<Price>1060</Price>
<OrderNo>123</OrderNo>
</Item>
</Items>
</Database>
```

Object-Relational Mapping: An XML document is viewed as a set of serialized objects and is mapped to the relational database with an object- relational mapping. That is, objects are mapped to tables, properties are mapped to fields, and inter-object relationships are mapped to primary key / foreign key relationships.

```
<Database>
<SalesOrder>
```

<Number>123</Number>
<OrderDate>17-01-2013</OrderDate>
<CustomerNumber>456-783456</CustomerNumber>
<Item>
<Number>140</Number>
<PartNumber>Aero-BC-47</PartNumber>
<Quantity>14</Quantity>
<Price>1060</Price>
</Item>
</SalesOrder>
</Database>

3.2 NativeXMLDatabases

An XML database is native if it (a) defines a logical model for an XML document. It stores and retrieves documents based upon that model and (b) has an XML document as its fundamental unit of (logical) storage, while (c) the storage model itself is not constrained. The architecture of native XML databases fall into two broad categories: text-based and model-based.

3.2.1 Text-Based Native XML Databases

A text-based native XML database stores XML fragment as text. This might be a file in a file system, a BLOB (data type) in a relational database, or a proprietary text format. The text-based native XML databases [10] use indexes which permit the query engine to easily move the pointer to any point in XML document. This will provide a tremendous amount of execution speed while retrieving the entire documents or document fragments. This is because the database can perform a single index lookup, place the disk head once, and, by considering that the necessary XML fragment is stored in contiguous bytes on the disk, retrieve the entire XML document in a single read.

3.2.2 Model-Based Native XML Databases

The second category of native XML databases is model-based native XML databases. Instead of storing the XML document as text, they build an internal object model from the XML document and store this model. Based upon the protocol of database, the model is stored. Some databases store the model in a relational or object-oriented database. For example, storing the XML document in a relational database [11] might result in tables while other databases use a proprietary storage format optimized for their model.Model-based native XML databases built on other databases are likely to have performance similar to those databases for retrieving data. However, the design of the database, especially for native XML databases built on top of relational databases, has significant room for variation. On the other hand, most such databases optimize their storage models and retrieval software. Model-based native XML databases that use a proprietary storage format are likely to have performance similar to text-based native XML databases when retrieving data in the order in which it is stored. This is because most such databases use physical pointers between XML nodes, which should provide performance similar to retrieving text.

4. RESULTS AND DISCUSSIONS

The XML family of standards has become an important component to many content publishing and distribution systems for users of all types. XML has enjoyed popularity for single source publishing due to the benefits of reusable content. The below section describes the airline company of how databases that implement XML standards, deliver powerful benefits to content creation and delivery.

4.1 Querying XML Data

It is often interested to use and/or extend SQL statements [8] to retrieve XML data. One reason is that database users are familiar with SQL which makes it a good starting point for managing XML. Also, existing relational applications are frequently augmented with XML data. Since XML is now a regular SQL data type [6], full documents can be retrieved from an XML column with a simple select statement:

SelectpartNumber from sales where CustomerNumber LIKE "456%";

The database supports most of the new SQL/XML functions and predicates,including XMLQUERY[10], XMLEXISTS, XMLTABLE, XMLVALIDATE, XMLPARSE, and XMLCAST. XMLEXISTS method tests whether an XML document matches the given criteria or condition. It returns either true or false for every tuple. The XMLEXISTS [8]method evaluates an XPath or XQuery expression for each value of an XML column. It returns false, if the result of the XQuery expression is an empty sequence. Otherwise, it returns true.

The following sample query returns full sales documents as in the previous example, but with XMLEXISTS for additional filtering. Only those tuples are returned where the sales document contains an employee with employee number *456-783456*.

Select item, salesOrder from saless
whereCustomerNumer LIKE "4%" and xmlexists('$alesdoc/sales/customer[CustomerNumer= 456-783456]'
passing by ref s.salesdoc as "salesdoc")

Apart from document filtering, it is also desirable to extract and return partial XML documents such as subtrees or atomic attribute and element values. This is achieved with

the XMLQUERY function. The query in the next example selects the item for all CustomerNumber, and the XMLQUERY function extracts the employee names with CustomerNumber 456-783456.

Selectitem, xmlquery('for $s in $salesdoc/sales/customer
where $s/CustomerNumber = 456-783456
return $s/Customer
passing by ref s.salesdoc as "salesdoc" returning sequence)
fromsaless
whereCustomerNumber LIKE "4%";

4.2 An Airline Company

The major goal of an airline is to provide air travel service to an audience in a highly competitive market where costs and efficiency can make the difference between success and failure of a business. All over the world, the people access the information. So providing the correct information to the users is the challenge faced by many industries. Basically, airlines have managed their maintenance information in systems that produce books and documents. The current maturity of XML databases and its XML support make it possible to publish the document-centric XML. Therefore the information is highly integrated and easily reused. Data coming from various manufacturers, departments, sources and regulatory agencies can be quickly integrated and inserted into information objects that make up the diverse delivery formats. The airline industry has been using XML and SGML technology for many years for sharing information. Most airlines are looking to create a single repository of maintenance information components to facilitate effective reuse and repurposing. The end result is to improve accuracy and efficiency, while reducing costs and delivery schedules. The XML Data extracted from the document is depicted in Fig.4

Usually, the information objects fall into three categories:

- The content should automatically be replaced in all locations, when the same content appears in other location.
- In other cases, there are significant differences in the actual text in two related objects, so when information is changed in one place, the system must track that the information needs to be manually updated.
- Finally, in some information objects, sub-components can be automatically updated, while others need to be manually updated.

Airlines, like many other large corporations, are large operations with diverse user and authoring roles, and a heterogeneous mix of software, operating systems and data formats. The administrators tend to be highly concentrated in one location on a consistent set of workstations. Meanwhile data and application servers might be on other platforms that host applications and databases. Finally, consumers of this information are highly distributed and need to work in a mixture of paper documents and Web clients. The XML standards supported by XML Store enable data that can be captured, edited, and rendered throughout a heterogeneous environment such as this.

For these reasons airlines are interested in avoiding being locked into a proprietary format that would insist on more consistent environments for all phases of document and information production and consumption. Also, considering that most aircraft equipment, the airplanes themselves, the information used to support them must remain portable and outlive the environment in which it is processed today.According to one airline industry,administrator who manages these types of data, "There is no other technology that provides what XML standards do to meet these requirements."

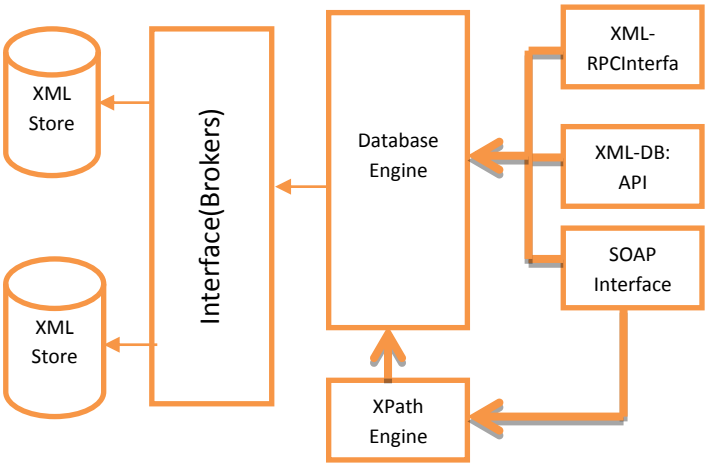

Fig 4 Block Diagram of Querying XML Data

4. CONCLUSION

Organizations intent on creating powerful and efficient environments for managing and accessing XML data have found that traditional relational databases are not designed to work with hierarchical XML content. A new breed of tools called XML databases not only understands the structure and syntax of XML content, but also utilizes the powerful standards developed by the W3C to process and access XML data. The W3C, developers of XML, have produced many related standards, including specifications that provide XML processing and database support. These include XML DTDs & Schemas, XSLT, DOM [4], XPath, XPointer, XLink, and XQuery. These specifications are well tested and deployed by organizations managing complex XML content. They enable a powerful platform for managing XML data that surpasses alternative traditional approaches for data management.

Information-intensive businesses such as an airline have built applications that demonstrate the power and efficiency of using an XML database to manage their XML information assets.The XML Databases provides a unified environment for storing, accessing, organizing, controlling, retrieving, and delivering any type of unstructured information within an enterprise. Usually, XML Store can be added on to the Content Server and will allow XML document storage and access via XQuery or SQL. The documents stored into XML Store will be part of the native XML repository, accessible through standard APIs and query language, and subject to the same policies and management as other documents. Customers can now get the security, compliance and archiving capabilities of the XML Database platform and the high performance provided by XML Store. But, XML schemas can have many nested and repeating elements such that the corresponding relational schema would consist of dozens or even hundreds of tables. Defining such a mapping from XML to a relational schema is a complicated task.

REFERENCES

[1]. Abiteboul, S., Cluet, S., and Milo, T. Querying and updating the file. in Proc. of the 19 Int. Conf. on Very Large Databases(1993), 73-84.

[2]. Adler, S., Berglund, A., Caruso, J., Deach, S., Grosso, P., Gutentag, E., Milowski, A., Parnell, S., Richman, J., and Zilles, S. (eds.). Extensible Stylesheet Language (XSL) Version 1.0, W3C Candidate Recommendation 21 November 2000. http://www.w3.org/TR/2000/CR-xsl-20001121/.

[3]. Arnold-Moore, T., Fuller, M., and Sacks-Davis, R. Approaches for structured document management. Markup Technologies (MT'1999).

[4]. Apparao, V., Byrne, S., Champion, M., Isaacs, S., Jacobs, I., Le Hors, A., Nicol, G., Robie, J., Sutor, R., Wilson, C., and Wood, L. (eds.). Document Object Model (DOM) Level 1 Specification Version 1.0, W3C Recommendation 1 October, 1998.http://www.w3.org/TR/1998/REC-DOM-Level-119981001/.

[5]. Bertino, E., Castano, S., Ferrari, E., and Mesiti, M. Controlled access and dissemination of XML documents. in Proc.2 International Workshop on Web Information and Data Management (1999), 22 – 27.

[6]. Cowan, J., and Tobin, R. (eds.). XML Information Set, W3C Proposed Recommendation 10 August 2001. http://www.w3.org/TR/2001/PR-xml-infoset-20010810.

[7]. M.F. Fernandez et al., "Publishing Relational Data in XML: The SilkRoute Approach," *IEEE Data Eng. Bull.*, vol. 24, no. 2, 2001, pp. 12-19

[8]. Balmin, A. et al.: *A Framework for Using Materialized XPath Views in XML Query Processing*, VLDB 2004, pages 60-71.

[9]. Beyer, K. et al.: *System RX: One Part Relational, One Part XML*, SIGMOD Conference, 2005.

[10]. Boag et al.: XQuery 1.0: An XML Query Language, February 2005, http://www.w3.org/TR/xquery

[11]. Bourret, R.: *XML Database Products*. http://www.rpbourret.com/xml/XMLDatabaseProds.htm