XML Technology
Khadija Abied Ali
Department of Computer/Faculty of Science
Sebha University
abied1991@gmail.com

Abstract— XML, the eXtensible Markup Language is an open and flexible standard created by the World Wide Web Consortium (W3C) in early 1998 for storing, publishing and exchanging any kind of information. It has been amazing to see how quickly the XML standard has been developed, and how quickly a large number of software vendors have adopted the standard.

The XML has been recently understood as a new approach to data modelling. A well formed XML document or a set of documents is an XML database and the associated DTD or schema specified in the language XML. Schema is its database schema. Implementation of a system enabling us to store and query XML documents efficiently (so called native XML databases) requires a development of new techniques and is of great importance in the world of information technologies today. In this work we briefly review XML; we include some basic knowledges that are required to understand the XML technology. We conclude that XML will be as important to the future of the Web as HTML has been to the foundation of the Web; XML is the future for all data transmission and data manipulation.

Index Term-- XML, XPath, XQuery, XML document, Parsing, DTD, XML schema, Application Programming Interfaces APIs, XML-enabled databases, Native XML databases NXDs.

1 INTRODUCTION
XML, the eXtensible Markup Language [1] is an open and flexible standard created by the World Wide Web Consortium (W3C) in early 1998 for storing, publishing and exchanging any kind of information. XML takes its major framework from the Standard Generalized Markup Language SGML [2]. XML is playing an increasingly important role in the exchange of a wide variety of data on the web and elsewhere. In this work we briefly review XML; we include some basic knowledges that are required to understand the XML technology.

This work is organized as follows: first in Section 2 we refer to the advantages of XML. In Section 3 we specify how XML can represent semi-structured data. The well-formedness and validity are two important concepts of XML document. We highlight these concepts in Section 4. In Section 5 we introduce briefly manipulating XML. The issue of storing XML documents in a database is the subject of Section 6. Finally, in Section 7 we present our conclusions.

2 ADVANTAGES OF XML
XML brings a number of powerful capabilities to information modeling:

- Heterogeneity, where each “record” can contain different data fields. There is a great advantage in being able to express information, as it exists, without restrictions.
- Extensibility, where new types of data can be added at will and do not need to be determined in advance.
- Flexibility, where data fields can vary in size and configuration from instance to instance; each data element can be as long or as short as necessary.
- Internationalization: XML supports the internationalization (Unicode).

We infer from the above that:

- Do not have to think too hard about the structure of the data before building a document.
- Do not have to define a schema.
- The documents are very flexible i.e. can add new things and change things in an ad hoc manner.

Despite of many advantages of XML, XML is not a replacement for HyperText Markup Language HTML [3]. XML and HTML are designed with different goals; XML was designed to focus on what data is, while HTML was designed to focus on how data looks. In other words, XML is about describing information, while HTML is about displaying information. Figure 1 represents an example of an HTML document.

![Fig. 1. An example of an HTML document](image-url)
3. HOW DOES XML LOOK?

XML is the ideal way to represent semistructured data; it is a text-based language, where markup tags like `<employee_number>` can clearly specify what kind of information is enclosed in a tag. An XML document consists mainly of elements and attributes. If an element is not empty, it begins with a start-tag (e.g. `<employee>`), and ends with an end-tag (e.g. `</employee>`). The text between the start-tag and end-tag is called the element’s content. An element with no content is said to be empty. Attributes occur inside start-tag after the element name. For instance, `<employee id="111">` is an employee element with the attribute id having the value 111.

Beside of elements and attributes, Comments, Processing Instructions, Entity References, CDATA Sections can be added to an XML document. Additional information on XML is possible to find in [4,5,6]. Consider the XML document in Figure 2. The tree structure of XML document in Figure 2 is represented in Figure 3.

We infer from Figure 3:

- An XML document has a single root node (address).
- The tree is a general ordered tree.
  - A parent node may have any number of children.
  - Child nodes are ordered, and may have siblings.
- Preorder traversals are usually used for getting information out of the tree.

4. THE WELL-FORMEDNESS AND VALIDITY

The well-formedness and validity are two important concepts of XML document. The former mainly deals with physical structure (e.g. matching and properly nested tags). In other words a well-formed document has a tree structure and obeys all the XML rules [8]. Here are some of these rules:

- Tags are enclosed in angle brackets.
- Documents must have a single root tag that begins the document.
- Tags come in pairs with start-tags and end-tags.
- Tags must be properly nested.
  - `<name><email>…</name></email>` is not allowed.
  - `<name><email>…</email><name>` is.

The later i.e validity focuses on the logical structure of elements. DTD [9] is the standard schema language for XML, that can be used to define the syntax and structure of XML documents. DTDs were developed first, so they are not as comprehensive as schema; XML Schema [10] is a much more comprehensive schema language for XML.

4.1 Parsing

Given a DTD or schema and its corresponding XML document, a parser can validate whether the document conforms to the desired structure and constraints. An XML parser is used to check that all the rules have been obeyed. Parsers are also available for free download over the Internet. One is Xerces, from the Apache open-source project. Java 1.4 also supports an open-source parser. Figure 4 shows the life cycle of XML document.
4.2 Sample DTD

Consider the XML document in Figure 5 which contains information about employees. The DTD of this XML document is shown in Figure 6.

The DTD in Figure 6 is interpreted as follows:

- `!ELEMENT employees` defines the employees element as having at least one element identified by employee. DTD allows to declare that the child element must occur one or more times (using +), zero or more times (using *), zero or one times (using ?), inside the parent element.

- `!ELEMENT employee` defines the employee element as having child elements name and salary.

- `ATTLIST id` defines the id attribute of the employee element. This attribute is required for each employee element.

- `!ELEMENT name` defines the name element to be of the type PCDATA. Consequently, the element contains character data.

- `!ELEMENT salary` defines the salary element to be of the type PCDATA.

5. MANIPULATING XML

In context of databases, XML is also a new database model serving as a powerful tool for approaching semistructured data. XML provides many of the database-like capabilities needed for the task, including schema languages (DTDs [9], XML Schema [10]), query languages (XPath [11], XQuery [12, 13, 14]), application programming interfaces (SAX [15], DOM [16], JDOM [17]).

In the following sections, we introduce briefly Xpath, XQuery, application programming interfaces (API), and other XML technologies.

5.1 Xpath( navigate through XML tree)

Besides the XML specification there are other standards that are associated with XML, e.g. XPath to identify elements in an XML documents. Xpath uses path expressions to navigate through the logical, hierarchical structure of an XML document, which is modelled as an ordered tree. A path expression consists of a series of "steps". Each step represents movement through a document along a specified "axis", and each step can apply one or more predicates to eliminate nodes that fail to satisfy a given condition. A path expression locates nodes within a tree. For example, the expression:
Fig. 5. An XML document containing information about employees

Fig. 6. DTD of the XML document in Fig. 5

employee/address/street, will select all “street” elements being children of “address” elements which have an ancestor element named “employee”. 

5.2 XQuery (the XML Query Language)

To obtain specified data from an XML database a number of special query languages have been developed, e.g. XML-QL, XQL, XQuery. XQuery[12, 13, 14] have been developed by the W3C XML Query Working Group. A subset of the XPath is a part of XQuery but more complex constructs are put into the language. 

XQuery can be implemented in many environments, such as in traditional databases[18], and XML repositories. The XML Query Working Group published XQuery 3.0: an XML Query Language, along with XQueryX, an XML representation for XQuery, both as W3C Recommendation, as well as the XQuery 3.0 use cases and requirements as final working group notes. XQuery extends the XPath language to provide efficient search and manipulation of information represented as trees from a variety of sources[19].

Data retrieval in XML

In context of databases, XML is also a new database model serving as a powerful tool for approaching semistructured data. As XML is a tool for data representation in the web, its main query language, i.e. XQuery, can serve as the bridge between XML repositories and the web.

XQuery is a functional language where each query is denoted by an expression. There are seven types of expressions in XQuery: path expression, element constructors, FLWR (For Let Where Return) expressions, expressions involving operators and functions, conditional expressions, quantified expressions and expressions that test or modify data types. Various expressions can be used together both sequentially and nested. Although XQuery provides the most expressive power, the powerful support of updates is missing.

An example

Consider the previous XML document describing employees in Figure 5. Suppose we want to retrieve the names of employees having salaries more than 7000. This query can be expressed in XQuery as follows:

```
for $emp in //employee
where $emp/salary > "7000"
return $emp/name
```

The output of the query will be as follows:

```
<name>Ahmed</name>
```

5.3 Application Programming Interfaces (APIs)

Application Programming Interfaces APIs are used to process an XML document by accessing its internal structure. Document Object Model DOM [16] is an API for tree-based XML parsing. It provides a standard set of programming interfaces for accessing, reading, writing, and modifying the nodes in the document tree; DOM represents a document tree fully in memory, where it can be accessed and manipulated by the user. A major advantage of DOM is that it provides a complete and editable view of the document’s contents. But, it requires lots of processing power and memory.
The Simple API for XML SAX [15] is a simple API for event-based XML parsing. SAX does not create a data structure; instead, it scans an input document and generates events, such as the start of an element, the end of an element, and so on. A major advantage of this approach is that SAX can result in reduced memory overhead. In the negative side, SAX does not allow random access manipulation of the document.

The Java Document Object Model JDOM [17] is a simple productive API for XML parsing. It incorporates the best features of DOM and SAX; it provides a full document view with random access, but it does not require the entire document to be in memory.

5.4 Another XML Technologies

There are a lot of technologies related to the XML. In the following lines we describe briefly some of them. SOAP [20] is the Simple Object Access Protocol used for invoke code offered by Web services over the Internet using XML and HTTP. The XML Linking (XLink) [21] allows elements to be inserted into XML documents in order to create and describe links between resources. The XML Pointer Language (XPointer) [22] defines an addressing scheme for individual parts of an XML document. These addresses can be used by any application that needs to identify parts of or locations in an XML document. A standard approach to formatting XML documents is using XSLT, the Extensible Stylesheet Language. It consists of XSL Transforms (XSLT), a language for transforming XML, and XSL Formatting Objects (XSL-FO), an XML vocabulary for specifying formatting semantics.

6. XML-ENABLED DATABASES VS NATIVE XML DATABASES NXDs

There are two different ways to store XML documents in a database: XML-enabled databases and Native XML Databases (NXDs). The former map the data to existing (relational) database systems. The latter are XML database systems whose inner data representation is XML-compliant [24]. NXDs are designed especially to store XML; NXDs preserve things like inner data representation is XML database systems. The latter are XML enabled databases and NXDs preserve things like inner data representation is XML. We have experimented with XPath and XQuery in the open source NXD (eXist) [25]. We have extended XQuery in “a logical way” to add time dimension to XML. The results has been published in [26]. Finally, we referred to the issue of storing XML.

We conclude that XML provides a flexible mechanism to represent complex data. XML can adequately handle rich data such as audio, video, nested data structures or complex documents.

REFERENCES

[2] Learning and using SGML. http://www.w3.org/MarkUp/SGML