17

XML: Managing Data Exchange

Words can have no single fixed meaning. Like wayward electrons, they can spin away from their initial orbit and enter a wider magnetic field. No one owns them or has a proprietary right to dictate how they will be used.

David Lehman, *End of the Word*, 1991

Learning objectives

Students completing this chapter will be able to

- define the purpose of XML;
- create a XML schema;
- code data in XML format;
- create an XML stylesheet;
- discuss data management options for XML documents.

Introduction

There are four central problems in data management: capture, storage, retrieval, and exchange. The focus for most of this book has been on storage (i.e., data modeling) and retrieval (i.e., SQL). Now it is time to consider capture and exchange. Capture has always been an important issue, and the guiding principle is to capture data once in the cheapest possible manner. Likewise, data exchange has long been an issue, but the Internet has elevated the importance of this issue. Electronic data interchange (EDI), the traditional standard of data exchange for large organizations, is giving way to XML, which is likely to become the data exchange standard for all organizations, irrespective of size.

EDI

EDI, which has been used for some 20 years, describes the electronic exchange of standard business documents between firms. A structured, standardized data format is used to

exchange common business documents (e.g., invoices and shipping orders) between trading partners. In contrast to the free form of e-mail messages, EDI supports the exchange of repetitive, routine business transactions. Standards mean that routine electronic transactions can be concise and precise. The main standard used in the United States and Canada is known as ANSI X.12, and the major international standard is EDIFACT. Firms following the same standard can electronically share data. Before EDI, many standard messages between partners were generated by computer, printed, and mailed to the other party, then data were manually entered into its computer. The main advantages of EDI are

- paper handling is reduced, saving time and money;
- data are exchanged in real time;
- data are keyed only once, resulting in fewer errors;
- activities are better coordinated between business partners through enhanced data sharing;
- money flows are accelerated and payments received sooner.

Despite these advantages, for most companies EDI use is low. A U.S. survey in the mid-1990s reported that almost 80 percent of the information flow between firms was on paper. Paper should be the exception, not the rule. Most EDI traffic has been handled by value-added networks (VANs) or private networks. VANs add communication services to those provided by common carriers (e.g., AT&T). However, these networks are too expensive for all but the largest 100,000 of the 6 million businesses in existence today in the United States. As a result, many businesses have not been able to participate in the benefits associated with EDI. However, the Internet enables smaller companies to take advantage of EDI.

Internet communication costs are typically less than with traditional EDI. In addition, the Internet is a global network potentially accessible by nearly every firm. Consequently, the Internet is displacing VANs as the electronic transport path between trading partners. The simplest approach is to use the Internet as a means of replacing a VAN by using a commercially available Internet EDI package. Another approach is to reexamine the technology of data exchange, since EDI was developed in the 1960s. A result of this rethinking is XML, but before considering XML, we need to learn about SGML, the parent of XML.

SGML

It is estimated that document management consumes up to 15 percent of a typical company's revenue, nearly 25 percent of its labor costs, and anywhere between 10 and 60 percent of an office worker's time. A U.S. Navy Cruiser, for example, has 26 tons of manuals in paper format on board. Each year, the U.S. Navy generates 15 million technical manuals and 237 million drawings which cost $4 billion to maintain. The standard generalized markup language (SGML) is designed to reduce the cost and increase the efficiency of document management.

A **markup language** embeds information about the text in the text. In Table 17-1, the markup tags indicate that the text contains CD liner notes. Note also that the titles and identifiers of the mentioned CDs are explicitly identified.

Table 17-1: Markup language

```
<cdliner>This uniquely creative collaboration between Miles Davis and
Gil Evans has already resulted in two extraordinary albums—
<cdtitle>Miles Ahead</cdtitle><cdid>CL 1041></cdid> and <cdti-
tle>Porgy and Bess</cdtitle><cdid>CL 1274</cdid>.
</cdliner>
```

SGML is an International Standard (ISO 8879) that defines the structure of documents. It is a vendor-independent language that supports cross system portability and publication for all media. Developed in 1986 to manage software documentation, SGML is widely accepted as the markup language for a number of information-intensive industries. As a meta-language, SGML is the mother of both HTML and XML. Thus, SGML can generate both these markup languages. SGML provides a stable platform for managing data when technology is rapidly changing. Because SGML is software- and hardware-neutral, businesses can choose *best of breed* tools to create, manage, retrieve, and disseminate data. This stability of SGML comes from its open systems approach.

SGML has three major advantages for data management:

- **Reuse.** Information can be created once and reused over and over. By storing critical documents in SGML, firms do not need to duplicate efforts when there are changes to documents. For example, a firm might store all its legal contracts in SGML.
- **Flexibility**. SGML documents can be published in any medium for a wide variety of audiences. Because SGML is content-oriented, presentation decisions are delayed until the output format is known. Thus, the same content could be printed, presented on the Web in HTML, or written to a CD as PDF.
- **Revision**. SGML enhances control over revision and enables version control. When stored in an SGML database, original data are archived alongside any changes. That means you know exactly what the original document contained and what changes were made.

Electronic publishing does not require SGML. CD-ROMs and Web sites can be created without SGML. However, the use of SGML preserves textual information independent of how and when it is presented. SGML protects a firm's investment in documentation for the long term. Because it is now possible to display documentation using multiple media (e.g., Web and CD-ROM), firms have become sensitized to the need to store documents in a single, independent manner that can then be converted for display by a particular media.

SGML's power is derived from its recording of both text and the meaning of that text. A short section of SGML demonstrates clearly the features and strength of SGML (see Table 17-2). The tags surrounding a chunk of text describe its meaning and thus support

presentation and retrieval. For example, the pair of tags <title> and </title> surrounding "XML: Managing Data Exchange" indicate that it is the chapter title.

Table 17-2: SGML code

```
<chapter>
<no>17</no>
<title>XML: Managing Data Exchange</title>
<section>
<quote><emph type = "2">Words can have no single fixed meaning. Like
wayward electrons, they can spin away from their initial orbit and
enter a wider magnetic field. No one owns them or has a proprietary
right to dictate how they will be used.</emph>
</quote>
</section>
</chapter>
```

Taking this piece of SGML, it is possible, using an appropriate stylesheet, to create a print version where the title of the chapter is displayed in Times, 16 point, bold, or a HTML version where the title is displayed in red, Georgia, 14 point, italics. Furthermore, the database in which this text is stored can be searched for any chapters that contain "exchange" in their title.

Now, consider the case where the text is stored as HTML (see Table 17-3). How do you, with complete certainty, identify the chapter title? Do you extract all text contained by <h1> and </h1> tags? You will then retrieve "16" as a possible chapter title. What happens if there is other text displayed using <h1> and </h1> tags? The problem with HTML is that it defines presentation and very little meaning. A similar problem exists for documents prepared with a word processor.

Table 17-3: HTML code

```
<html>
<body>
<h1><b>17 </b></h1>
<h1><b>XML: Managing Data Exchange</b></h1>
<p><i>Words can have no single fixed meaning. Like wayward electrons,
they can spin away from their initial orbit and enter a wider magnetic
field. No one owns them or has a proprietary right to dictate how they
will be used.</i>
</body>
</html>
```

By using embedded tags to record meaning, SGML makes a document platform independent and greatly improves the effectiveness of searching. However, before a firm can use SGML, it needs to determine what meaning is contained within its document. In other words, it needs to define a data model[1] and create a **document type definition** (DTD).

1. And you thought you had said goodbye to this beast :)

Document elements to be marked up in SGML are *mapped* to a DTD. The accuracy of an SGML mapping can then be validated using the DTD.

Fortunately, several industries have already developed DTDs for common documents. Today, DTDs have been designed for the aerospace, automotive, electronics manufacturing, publishing, software development, and telecommunications documentation sectors. For example, the publishing industry has defined ISO12083 for articles, books, and serials.[2]

There are some features of SGML that are considered to make implementation difficult, and that also limit the ability to create tools for information management and exchange. As a result, XML, a derivative of SGML, was developed.

XML

Extensible markup language (XML), a new language designed to make information self-describing, retains the core ideas of SGML. You can think of XML as SGML for electronic and mobile commerce. XML has the potential to extend the Internet beyond information delivery to many other kinds of human activity. Since the definition of XML was completed in early 1998 by the World Wide Web Consortium (W3C), this new standard has spread rapidly because it solves a critical data management problem. XML is more than a mere incremental improvement of HTML. It is a conceptual change because XML is a meta-language—a language to generate languages. XML will steadily replace HTML on many Web sites. The major differences between XML and HTML are captured in Table 17-4.

Table 17-4: XML vs. HTML

XML	HTML
Structured text	Formatted text
User-definable structure (extensible)	Predefined formats (not extensible)
Context-sensitive retrieval	Limited retrieval
Greater hypertext linking	Limited hypertext linking

HTML, an electronic-publishing language, describes how a Web browser should display text and images on a computer screen. It tells the browser nothing about the meaning of the data. For example, the browser does not know whether a piece of text represents a price, a product code, or a delivery date. Humans infer meaning from the context (e.g., Aug 8, 2002 is recognized as a date). Given the explosive growth of the Web, HTML clearly works well enough for exchanging data between computers and humans. It does not, however, work for exchanging data between computers, because computers are not smart enough to infer meaning from context.

Successful data exchange requires that the meaning of the exchanged data be readily determined by a computer. The XML solution is to embed tags in a file to describe the data (e.g., insert tags into an order that indicate attributes such as price, size, quantity, and color). A browser, or program for that matter, can then recognize this document as a cus-

2. See www.xmlxperts.com/12083.htm

tomer order. Consequently, it can do far more than just display the price. For example, it can convert all prices to another currency. More importantly, the data can be exchanged between computers.

XML consists of rules (see Table 17-5) that anyone can follow to create a markup language (e.g., a markup language for financial data). Hence, the eXtensible in the XML name indicates that the language can be easily extended to include new tags. In contrast, HTML is not extensible and its set of tags is fixed, which is one of the major reasons why HTML is easy to learn.The XML rules ensure that a parser, a type of computer program, can process any extension or addition of new tags.

Table 17-5: XML rules

- Elements must have both an opening and an closing tag.
- Elements must follow a strict hierarchy with only one root element.
- Elements must not overlap other elements.
- Element names must obey XML naming conventions.
- XML is case sensitive.

Consider the credit card company that wants to send your latest statement via the Internet so you can load it into your financial program. Since this is a common problem for credit card companies and financial software authors, these industry groups have combined to create Open Financial Exchange (OFX),[3] a language for the exchange of financial data across the Internet.

XML has a small number of rules. Tags almost always come in pairs, as in HTML. A pair of tags surrounds each piece of data (e.g., <price>89.12</price>) to indicate its meaning, whereas in HTML they indicate how the data are presented.Tag pairs can be nested inside one another to multiple levels, which effectively creates a tree or hierarchical structure. Because XML uses Unicode (see page 335), it enables exchange of information not only between different computer systems but also across language boundaries.

The differences between HTML and XML are captured in the following examples for each markup language. Note how HTML incorporates formatting instructions (i.e., the course code is bold), whereas XML describes the meaning of the data.

Table 17-6: Comparison of HTML and XML coding

HTML	XML
MIST7600 Data Management 3 credit hours	<course> <code>MIST7600</code> <title>Data Management</title> <credit>3</credit> </course>

3. www.ofx.net/

The introduction of XML will see a shift of processing from the server to the browser. At present, most processing has to be done by the server because that is where knowledge about the data is stored. The browser knows nothing about the data and therefore can only present but not process. However, when XML is implemented, the browser can take on processing that previously had to be handled by the server.

Imagine that you are selecting a shirt from a mail-order catalog. The merchant's Web server sends you data on 20 shirts (100k of text and images) with prices in U.S. dollars. If you want to see the prices in euros, the calculation will be done by the server and the full details for the 20 shirts retransmitted (i.e., another 100k is sent from the server to the browser). However, once XML is in place, all that needs to be sent from the server to the browser is the conversion rate of dollars to euros and a Java program to compute the conversion at the browser end (see Table 17-7). In most cases, less data will be transmitted between a server and browser when XML is in place. Consequently, widespread adoption of XML will reduce network traffic.

Table 17-7: Execution of HTML and XML code

HTML	XML
Retrieve shirt data with prices in $US Retrieve shirt data with prices in euros	Retrieve shirt data with prices in $US Retrieve conversion rate of $US to euro Retrieve Java program to convert currencies Compute prices in euros

XML will also make searching more efficient and effective. At present, search engines look for matching text strings, and consequently return many links that are completely irrelevant. For instance, if you are searching for details on the Nomad MP3 player, and specify "nomad" as the sought text string, you will get links to many items that are of no interest (e.g., *The Fabulous Nomads Surf Band*). Searching will be more precise when you can specify that you are looking for a product name that includes the text "nomad." The search engine can then confine its attention to text contained with the tags <productname> and </productname>, assuming these tags are the XML standard for representing product names.

The major expected gains from the introduction of XML are

- **store once and format many ways**—data stored in XML format can be extracted and reformatted for multiple presentation styles (e.g., printed report, CD-ROM);
- **hardware and software independence**—one format for all systems;
- **capture once and exchange many times**—data are captured as close to the source as possible and never again (i.e., no rekeying);
- **accelerated targeted searching**—searches are more precise and faster because they use XML tags;
- **less network congestion**—the processing load shifts from the server to the browser.

Classified ads XML standard

The NAA Classified Advertising Standards Task Force of approximately 40 classified advertisers, advertising publishers, aggregators, system users, suppliers, and technology experts is designing a standard for the electronic exchange of classified ads. The standard will define a common classified advertising data structure that technology providers can use to create better tools and systems for handling these data. With standardization, advertisers, ad aggregators, and publishers can simplify their workflows and more effectively provide data sought by consumers.

The task force has generated a document type description (DTD) to define XML tags and their proper usage in conjunction with this standard. The DTD has a set of elements, or fields, which describe a product being listed for sale. The use of keyword tagging and standardized ad data fields will dramatically improve searching speed and accuracy.

Source: www.naa.org/technology/clsstdtf/index.html

XML language design

XML lets developers design application-specific vocabularies. To create a new language, designers must agree on three things:

- the allowable tags;
- the rules for nesting tagged elements;
- which tagged elements can be processed.

The first two, the language's vocabulary and structure, are typically defined in an XML schema. When first introduced, XML adopted the Document Type Definition (DTD) of SGML to define markup tags. The more recently developed XML schema is now the recommended method for defining an XML document. Programmers use the XML schema to understand the meaning of tags so they can write software to process an XML file.

XML tags describe meaning, independent of the display medium. An XML stylesheet, another set of rules, defines how an XML file is automatically formatted for various devices. This set of rules is called an Extensible Stylesheet Language (XSL). Stylesheets allow data to be rendered in a variety of ways, such as Braille or audio for visually impaired persons.

XML Schema

An XML schema (called schema for brevity) is an XML file associated with an XML document that informs an application how to interpret markup tags and valid formats for tags. The advantage of a schema is that it leads to standardization. Consistently named and defined tags create conformity and support organizational efficiency. They avoid the confusion and loss of time when the meaning of data is not clear. Also, when validation information is built into a schema, some errors are detected before data are exchanged.

XML does not require the creation of a schema. If a document is well-formed, XML will interpret it correctly. A well-formed document follows XML syntax and has tags that are correctly nested. One of the strengths of XML is that browser writers have agreed to reject any XML file that is not well-formed. Incorrectly specified HTML works on some browsers and not others, so using XML should result in greater consistency across browsers.

A schema is a very strict specification, and any errors will be detected when parsing. A schema defines

- the names and contents of all elements that are permissible in a certain document;
- the structure of the document;
- how often an element may appear;
- the order in which the elements must appear;
- the type of data the element can contain.

DOM

The **document object model** (DOM) is the model underlying XML. It is based on a tree (i.e., it supports 1:1 and 1:m, but not m:m relationships). A document is modeled as a hierarchical collection of nodes that have parent/child relationships. The node is the primary object and can be of different types (such as document, element, attribute, text). Each document has a single document node which has no parent, and zero or more children that are element nodes. It is a good practice to create a visual model of the XML document and then convert this to a schema, which is XML's formal representation of the DOM.

At this point, an example is the best way to demonstrate XML, schema, and DOM concepts. We will use the familiar CD problem that was introduced in Chapter 3 (see page 83). In keeping with the style of this text, we define a minimal amount of XML to get you started, and then more features are added once you have mastered the basics.

CD library case

The CD library case gradually develops, over several chapters, a data model for recording details of a CD collection, culminating in the model on page 151. Unfortunately, we cannot quickly convert this final model to an XML document model since a DOM is based on a tree model. Thus, we must start afresh.

The model (see Figure 17-1), in this case, is based on the observation that a CD library has many CDs, and a CD has many tracks.

A model is then mapped into a schema using the following procedure.

- Each entity becomes a complex element type.
- Each data model attribute[4] becomes a simple element type.
- The 1:m relationship is recorded as a sequence.

4. XML also has the notion of an attribute, and so we need to be precise when talking about attributes.

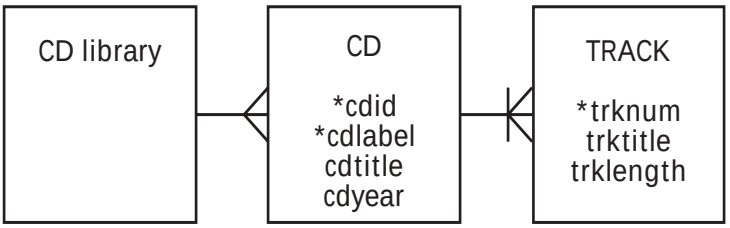

Figure 17-1. CD library tree data model

The schema for the CD library is shown in Table 17-8. For convenience of exposition, the source code lines have been numbered, but these numbers are not part of a schema.

Table 17-8: Schema for CD library (cdlib.xsd)

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xsd:schema xmlns:xsd='http://www.w3.org/2001/XMLSchema'>
 3 <!--CD library-->
 4   <xsd:element name="cdlibrary">
 5     <xsd:complexType>
 6       <xsd:sequence>
 7         <xsd:element name="cd" type="cdType" minOccurs="1"
 8           maxOccurs="unbounded"/>
 9       </xsd:sequence>
10     </xsd:complexType>
11   </xsd:element>
12 <!--CD-->
13   <xsd:complexType name="cdType">
14     <xsd:sequence>
15       <xsd:element name="cdid" type="xsd:string"/>
16       <xsd:element name="cdlabel" type="xsd:string"/>
17       <xsd:element name="cdtitle" type="xsd:string"/>
18       <xsd:element name="cdyear" type="xsd:integer"/>
19       <xsd:element name="track" type="trackType" minOccurs="1"
20          maxOccurs="unbounded"/>
21     </xsd:sequence>
22   </xsd:complexType>
23 <!--Track-->
24   <xsd:complexType name="trackType">
25     <xsd:sequence>
26       <xsd:element name="trknum" type="xsd:integer"/>
27       <xsd:element name="trktitle" type="xsd:string"/>
28       <xsd:element name="trklen" type="xsd:time"/>
29     </xsd:sequence>
30   </xsd:complexType>
31 </xsd:schema>
```

There are several things to observe about the schema.

- All XML documents begin with an XML declaration {1}.[5] The encoding statement (i.e., UTF-8 specifies that the 8-bit form of Unicode is used (see page 335)).
- The declaration for the XSD Schema namespace {2}.[6]
- Comments are placed inside the tag pair `'<!--'` and `'-->'` {3}.

5. In this chapter, numbers in {} refer to line numbers in the corresponding XML code.

- The CD library is defined {4-10} as a complex element type, which essentially means it has embedded elements, which are a sequence of CDs in this case.
- A sequence is a series of child elements embedded in a parent as illustrated by a CD library containing a sequence of CDs {7}, and a CD containing elements of CD identifier, label, and so forth {15-20}. The order of a sequence must be maintained by any XML document based on the schema.
- A sequence can have a specified range of elements. In this case, there must be at least one CD (minOccurs="1") but there is no upperbound (maxOccurs= "unbounded") on how many CDs there can be in the library {7}.
- An element that has a child (e.g., cdlibrary which is at the 1 end of a 1:m) or possesses attributes (e.g., track) is termed a complex element type.
- CD is a complex element type {13-20}, and has the name cdType {13}.
- The element CD is defined by specifying the name of the complex type (i.e., cdType) containing its specification {7}.
- Track is a complex type because it contains elements of track number, title, and length {24-30}. The name of this complex type is trackType {24}.
- Notice the reference within the definition of CD to the complex type trackType is used to specify the element track {19}.
- Simple types (e.g., cdid and cdyear) do not contain any elements, and thus the type of data they store must be defined. Thus, cdid is a text string and cdyear is an integer.

The purpose of a schema is to define the contents and structure of an XML file. It is also used to verify that an XML file has a valid structure and that all elements in the XML file are defined in the schema.

Some common datatypes are shown in Table 17-9. The meaning is obvious in most cases for those familiar with SQL, except for uriReference. A Uniform Reference Identifier (URI) is a generalization of the URL concept.[7]

Table 17-9: Some common datatypes

string
boolean
uriReference
decimal
float
integer
time
date

We can now use the recently defined CDlibrary schema to describe a small CD library containing the following CD.

6. A namespace is a collection of names of attributes, types, and elements. That's all you need to know for now.
7. See the Glossary for an extended definition.

Table 17-10: Data for a small CD library

Id	A2 1325		D136705	
Label	Atlantic		Verve	
Title	Pyramid		Ella Fitzgerald	
Year	1960		2000	
Track	Title	Length	Title	Length
1	Vendome	2:30	A tisket, a tasket	2:37
2	Pyramid	10:46	Vote for Mr. Rhythm	2:25
3			Betcha nickel	2:52

The XML for describing the CD library is displayed in Table 17-11. There are several things to observe:

- ❖ All XML documents begin with an XML declaration {1}.
- ❖ The declaration immediately following the XML declaration identifies the root element of the document (i.e., cdlibrary) and the schema[8] (i.e., cdlib.xsd) {2-3}.
- ❖ The definition of the first CD and its tracks {4-19}.
- ❖ The definition of the first track on the first CD {9-13}.

As you now realize, the definition of an XML document is relatively straightforward. It is a bit tedious with all the typing of tags to surround each data element. However, there are XML editors which relieve this tedium.[9]

Skill builder

1. Use Internet Explorer[10] to access this book's Web site, link to the XML section, and click on cdlib.xml. You will see how IE displays XML. Investigate what happens when you click minus (-) and then plus (+).
2. Save the XML code (File > Save As >) displayed by IE and paste it into a text editor or word processor. Notice that line 2 as displayed by IE might be different from that of the original XML file.
   Now, add details of the CD[11] displayed in Table 17-12 to this XML code, save it as cdlibv2.xml, and open it with IE.

XSL

As you now know from the prior exercise, the browser display of XML is not particularly usable. What is missing is a stylesheet that tells the browser how to display an XML file. **Extensible Stylesheet Language (XSL)** is a language for defining the rendering of an

8. IE6.0, the release available at the time of writing, does not validate an XML file against its schema.
9. See www.webreference.com/xml/column8/index.html for more information on XML editors.
10. Netscape Navigator does not display xml files correctly for all systems.
11. The CD actually has 16 tracks, but let's settle for entering five tracks.

Table 17-11: XML for describing a CD (cdlib.xml)

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <cdlibrary xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 3   xsi:noNamespaceSchemaLocation="cdlib.xsd">
 4     <cd>
 5         <cdid>A2 1325</cdid>
 6         <cdlabel>Atlantic</cdlabel>
 7         <cdtitle>Pyramid</cdtitle>
 8         <cdyear>1960</cdyear>
 9         <track>
10             <trknum>1</trknum>
11             <trktitle>Vendome</trktitle>
12             <trklen>00:02:30</trklen>
13         </track>
14         <track>
15             <trknum>2</trknum>
16             <trktitle>Pyramid</trktitle>
17             <trklen>00:10:46</trklen>
18         </track>
19     </cd>
20     <cd>
21         <cdid>D136705</cdid>
22         <cdlabel>Verve</cdlabel>
23         <cdtitle>Ella Fitzgerald</cdtitle>
24         <cdyear>2000</cdyear>
25         <track>
26             <trknum>1</trknum>
27             <trktitle>A tisket, a tasket</trktitle>
28             <trklen>00:02:37</trklen>
29         </track>
30         <track>
31             <trknum>2</trknum>
32             <trktitle>Vote for Mr. Rhythm</trktitle>
33             <trklen>00:02:25</trklen>
34         </track>
35         <track>
36             <trknum>3</trknum>
37             <trktitle>Betcha nickel</trktitle>
38             <trklen>00:02:52</trklen>
39         </track>
40     </cd>
41 </cdlibrary>
```

Table 17-12: CD data

Id	314 517 173-2
Label	Verve
Title	The essential Charlie Parker
Year	1992

Track	Title	Length
1	Now's the time	3:01
2	If I should lose you	2:46
3	Mango Mangue	2:53
4	Bloomdido	3:24
5	Star Eyes	3:28

XML file. An XSL document defines the rules for presenting an XML document's data. XSL is an application of XML, and an XSL file is also an XML file.

The power of XSL is demonstrated by applying the stylesheet shown in Table 17-13 to the XML displayed in Table 17-11 to produce Figure 17-2.

```
Complete List of Songs
Pyramid, Atlantic, 1960.5 [A2 1325]
1 Vendome 00:02:30
2 Pyramid 00:10:46

Ella Fitzgerald, Verve, 2000 [D136705]
1 A tisket, a tasket 00:02:37
2 Vote for Mr. Rhythm 00:02:25
3 Betcha nickel 00:02:52
```

Figure 17-2. Result of applying a stylesheet to CD data

To use a stylesheet with an XML file, you must add a line of code to point to the stylesheet file. In this case, you add the following:

```
<?xml-stylesheet type="text/xsl" href="cdlib.xsl" media="screen"?>
```

as the second line of cd.xml (i.e., it appears before <cdlibrary>). The added line of code points to cd.xsl as the stylesheet. This means that when the browser loads cdstyle.xml,[12] it uses the contents of cd.xsl to determine how to render the contents of cdstyle.xml.

We now need to examine the contents of cd.xsl so you can learn some basics of creating XSL commands. You will soon notice that all XSL commands are preceded by xsl:.

- Tell the browser it is processing an XML file {1}.
- Specify that the file is a stylesheet {2}.
- Specify a template, which identifies which elements should be processed and how they are processed. The match attribute {4} indicates the template applies to the source node. Process the template {11} defined in the file {15-45}. A stylesheet can specify multiple templates to produce different reports from the same XML input.
- Specify a template to be applied when the XSL processor encounters the <cdlibrary> node {15}.
- An outer loop for processing each CD {16-44}.
- Define the values to be reported for each CD (i.e., title, label, year, and id) {19, 21, 23, 25}. The respective XSL commands select the values. For example, `<xsl:value-of select="cdtitle" />` specifies selection of cdtitle.
- An inner loop for processing the tracks on a particular CD, because a CD has many tracks {29-41}.
- The track data are presented in tabular form using HTML table commands interspersed with XSL {28-42}.

12. The cd.xml file with the additional line of code to identify the stylesheet.

Table 17-13: Stylesheet for displaying an XML file of CD data (cdlib.xsl)

```
 1 <?xml version="1.0" encoding="UTF-8"?>
 2 <xsl:stylesheet version="1.0"
                xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
 3   <xsl:output encoding="UTF-8" indent="yes" method="html" version="1.0" />
 4   <xsl:template match="/">
 5   <html>
 6     <head>
 7       <title> Complete List of Songs </title>
 8     </head>
 9   <body>
10     <h1> Complete List of Songs </h1>
11     <xsl:apply-templates select="cdlibrary" />
12     </body>
13     </html>
14   </xsl:template>
15   <xsl:template match="cdlibrary">
16     <xsl:for-each select="cd">
17       <br/>
18       <font color="maroon">
19         <xsl:value-of select="cdtitle" />
20         ,
21         <xsl:value-of select="cdlabel" />
22         ,
23         <xsl:value-of select="cdyear" />
24         [
25         <xsl:value-of select="cdid" />
26         ] </font>
27       <br/>
28       <table>
29         <xsl:for-each select="track">
30           <tr>
31             <td align="left">
32               <xsl:value-of select="trknum" />
33             </td>
34             <td>
35               <xsl:value-of select="trktitle" />
36             </td>
37             <td align="center">
38               <xsl:value-of select="trklen" />
39             </td>
40           </tr>
41         </xsl:for-each>
42       </table>
43       <br/>
44     </xsl:for-each>
45   </xsl:template>
46 </xsl:stylesheet>
```

Skill builder

1. Use IE to access this book's Web site, link to the XML section, and download cdlib.xsl into a directory on your machine.
2. Edit a copy of cdlib.xml by inserting the following as the second line of cdlib.xml `<?xml-stylesheet type="text/xsl" href="cdlib.xsl" media="screen"?>.`
3. Save the file as cdlibv3.xml in the same directory as cdlib.xsl and open it with IE.
4. Make a similar change to cdlibv2.xml and open it with IE.

Converting XML

There are occasions when there is a need to convert an XML file.

- **transformation**—conversion from one XML vocabulary to another (e.g., between financial language FPML and finML);
- **manipulation**—reordering, filtering, or sorting parts of a document;
- **rendering in another language**—rendering the XML file using another format, such as Wireless Access Protocol (WAP).

You have already seen how XSL can be used to transform XML for rendering as HTML. The original XSL language has been split into three languages:

- XSLT for transformation and manipulation;
- XSLT for rendition;
- XPath for accessing the structure of an XML file.

For a data management course, this is as far as you need to go with learning about XSL. Just remember that you have only touched the surface. To become proficient in XML, you will need an entire course on the topic.

XML and databases

XML is more than a document processing technology. It is also a powerful tool for data management. For database developers, XML is likely to be used to facilitate middle tier data integration and schemas. Most of the current major DBMS producers are developing XML-centric extensions to their product lines.

Many XML documents are stored for the long term, because they are an important repository of organizational memory. A data exchange language, XML is a means of moving data between databases, which means a need for tools for exporting and importing XML.

XML documents can be stored in the same format as you would store a word processing or HTML file: you just place them in an appropriately named folder. File systems, however, have limitations that become particularly apparent when a large number of files need to be stored, as in the corporate setting.

What is needed is a database management system for storing, retrieving, and manipulating XML documents. Such a DBMS should

- be able to store a large number of documents;
- be able to store large documents;
- support access to portions of a document (e.g., the data for a single CD in a library of 20,000 CDs);
- enable concurrent access to a document but provide locking mechanisms to prevent the *lost update problem* (see page 519), which can happen when two or more people are editing the same portion of a document;

- ❖ keep track of different versions of a document;
- ❖ integrate data from other sources (e.g., insert the results of an SQL query formatted as an XML fragment into an XML document).

There are several possible solutions for XML document management: RDBMS, ODBMS, or an XML database.

RDBMS

An XML document could be stored within an RDBMS. Storing an intact XML document as a CLOB (see page 252) is a sensible strategy if the XML document contains static content that will only be updated by replacing the entire document. Examples include written text such as articles, advertisements, books, or legal contracts. These *document-centric* files[13] (e.g., articles and legal contracts) are retrieved and updated in their entirety.

For more dynamic *data-centric* XML files[14] (e.g., orders, price lists, airline schedules), the RDBMS must be extended to support the structure of the data so that portions of the document (e.g., elements) can be retrieved and updated. The object-relational extensions of most major databases provide the wherewithal to capture the structure of an XML document. For example, Oracle's XML SQL utility can store an XML document by mapping it to an object-relational format. The same utility can then retrieve the data as an XML document.

You can expect most RDBMS vendors to offer extensions for supporting storage and retrieval of both document- and data-centric XML files. Some of them are likely to release XML servers that are built on underlying relational technology.

Another issue is the conversion of existing relational data to XML. Products, such as DB2XML, are already emerging.

DB2XML

Some sources estimate that more than 75 percent of current Web pages are generated from database data. These data are typically converted into HTML by a server-side script, which you learned how to write in Chapter 16. An alternative is to convert relational data to XML, and with an appropriate stylesheet, let the browser generate the presentation. This approach has the advantage of reducing the complexity of applications by separating SQL and server-side scripting code. DB2XML[15] is an example of a tool for converting relational data to XML.

DB2XML has three main functions:

- ❖ transform the results of a query into an XML document;

13. Document-centric files have low structure (i.e., a few elements and mainly text).
14. Data-centric files have high structure (i.e., many elements, often repeated).
15. www.informatik.fh-wiesbaden.de/~turau/DB2XML/

- ❖ generate a schema for the XML;
- ❖ transform the XML using a stylesheet.

Skill builder

1. Use your browser to access www.informatik.fh-wiesbaden.de/~turau/DB2XML/demo/db2xmlxslservlet.html and execute one of the demonstrations. Inspect the stylesheet and XML document.
2. Which do you think is easier to write and maintain: PHP and server-side scripting or DB2XML and stylesheets?

A DBMS is designed to store data and XML is designed for data interchange. Thus, we need tools, such as DB2XML, for converting data extracted from a database into a format that can be exchanged among organizations. However, it is not clear at this point which path data presentation should follow: SQL and server-side scripting or XML and stylesheets. As tools are developed to support both of these paths, this uncertainty should be resolved.

ODBMS

ODBMS vendors can also use their technology for storing XML documents. However, at this point, most have done little more than store entire XML documents as objects. This is not surprising, as ODBMS vendors do not have the market size of RDBMS vendors and hence have less resources to invest in creating advanced XML handling features.

Some argue that the ODBMS model is a better candidate for managing XML documents than the relational model. As we learned in Chapter 13, despite some advantages, the ODBMS model has made little progress in replacing relational technology.

XML database

A third approach is to build a special-purpose XML database. Tamino is an example of such an approach, and you can expect others to emerge in the near future. Because this is such a new area, this textbook can do little more than make you aware of the technology and urge you to monitor the development of XML databases.

Skill builder

1. Visit the Web site for Tamino[16] and review its benefits and architecture.
2. Search the Tamino site or the computer press[17] to learn who has adopted Tamino.
3. What conclusions do you reach about the usefulness of an XML database?

16. www.softwareag.com/tamino platform/introduction.htm
17. e.g., computerworld.com/

XML in Government

The U.K.'s e-Government Interoperability Framework (e-GIF) sets out the government's technical policies and standards for achieving interoperability and information systems conformity across the public sector. It defines the essential prerequisite for linked, Web-enabled government, and is a cornerstone policy in the overall e-government strategy.

The main thrust is to adopt Internet and Web standards for all government systems. The e-GIF also adopts standards that are well supported in the market place. It is a pragmatic strategy that aims to reduce cost and risk for government systems whilst aligning them to the global Internet revolution.

The government policy is to use: XML and XML schemas for data integration; UML (unified modeling language), RDF (resource description framework), and XML for data modelling and description language; and XSL, DOM and XML for data presentation. XML products will be written so as to comply with the recommendations of the World Wide Web Consortium (W3C).

e-GIF is a key plank in the government's drive to get all its services online by 2005 and cut bureaucracy within the public sector. There are two main benefits the policy will bring: creating 24-hour one-stop government and banishing bureaucracy in government by moving the public sector away from traditional paper based ways of working by electronically integrating information across a range of government departments and organizations.

The government has launched the U.K. GovTalk initiative. This is a Cabinet Office led, joint government and industry forum for generating and agreeing on XML data schemas for use throughout the public sector. The GovTalk Group will manage the acceptance, publication, and any subsequent change requests for the schema. The initiative helps developers by providing information, best practice guidance, and toolkits for conversion of legacy data. It is intended to make adoption of the e-GIF policies and standards simple, attractive, and cost-effective.

Source: www.citu.gov.uk/egif/contents.htm

Conclusion

XML is a significant technological development, and its importance and value will be increasingly apparent. It is clear that its main role will be to facilitate the exchange of data between organizations and within those organizations that do not have integrated systems. XML will achieve much of what was promised by EDI in that it will significantly lower the cost of transactions by accelerating the transfer of information from paper to bits.

Whether XML will become dominant in data presentation and storage is undecided. There are acceptable alternatives for data presentation (e.g. server-side scripting and Java), and the superiority of XSL is not obvious. Similarly, relational technology can be successfully extended to store and process XML documents. Thus, the case for pure XML database technology is also not obvious.

New technologies inevitably generate a lot of excitement and speculation about the downfall of prior technology. XML certainly has a role in the management of organizational data. Although the size and extent of this role is unclear, it is certain that all data managers must have some knowledge of XML.

Mastery of XML is well beyond the scope of a single chapter. Indeed, it is a book-length topic and more than 100 books have already been written on this relatively recent technology. It is important to remember that the prime goal of XML is to support data interchange.

The tags on retail

The Association for Retail Technology Standards (ARTS), the standards arm of the National Retail Federation, has compiled a dictionary of XML tags. These tags are a prerequisite for exchanging business information on line in XML and are essential if B2B electronic commerce is to expand.

These tags are plain English words and not computer codes. They are used to define data in an XML file. For example, a brand name in an XML document would be flanked by the tags <brand name>. By having a standard data dictionary, XML messages can be transmitted without confusion in their interpretation.

The retail industry has not had an XML data dictionary until now. Currently, there are several in the works. The Uniform Code Council is compiling one as part of its UCCnet B2B exchange and data-synchronization initiative. The National Association of Convenience Stores (NACS) is developing a data dictionary for the convenience store and petroleum retailing industries.

There are two versions of the ARTS data dictionary. The standard, public-domain version is accessible at www.nrf-arts.org. It contains 2,298 tags for the retail industry with explanations for each tag.

Source: Anonymous. 2000. ARTS publishes XML retail data dictionary. *Chain Store Age* 76 (11):108.

Summary

Electronic data exchange has become more important with the widespread use of the Internet. EDI, which penetrated mainly the major organizations, is being replaced by XML.

SGML, a precursor of XML, defines the structure of documents. SGML's value derives from its reusability, flexibility, and support for revision. XML, a derivative of SGML, is designed to support electronic commerce and overcome some of the shortcomings of SGML. XML supports data exchange by making information self-describing. It is a metalanguage because it is a language to generate other languages (e.g., finML). It promises substantial gains for the management and distribution of data. The XML language consists of an XML schema, DOM, and XSL.

A schema defines the structure of a document and how an application should interpret XML markup tags. The DOM is a tree-based data model of an XML document. XSL is used to specify a stylesheet for displaying an XML document.

XML documents can be stored in either a RDBMS, ODBMS, or XML database. There are tools for converting relational data to XML format and vice versa. XML is a significant technological development that will facilitate the exchange of data between and within organizations.

Key terms and concepts

Connectors
Document object model (DOM)
Document type definition (DTD)
Electronic data interchange (EDI)
Extensible markup language (XML)
Extensible style language (XSL)
Hypertext markup language (HTML)
Markup language
Occurrence indicators
Standard generalized markup language (SGML)
XML database
XML schema

References

Anderson, R. 2000. *Professional XML*. Birmingham, UK; Chicago: Wrox Press.

Exercises

1. A business has a telephone directory that records the first and last name, telephone number, and e-mail address of everyone working in the firm. Departments are the main organizing unit of the firm, so the telephone directory is typically displayed in department order, and shows for each department the contact phone and fax numbers and e-mail address.
   a. Create a hierarchical data model for this problem.
   b. Define the schema.
   c. Create an XML file containing some directory data.
   d. Create an XSL file containing a stylesheet.
2. Create a schema for your university or college's course bulletin.
3. Create a schema for a credit card statement.
4. Take a page in a dictionary and do the following:
   a. Create a hierarchical data model for this problem.
   b. Define the schema.

c. Create an XML file containing some directory data.
d. Create an XSL file containing a stylesheet.

5. Search the Web to identify an XML standard for an industry. What do you observe about the standard? How detailed is it (e.g., how many elements)? Is industry adopting the standard?

Case: A conference support system

An ideal conference support system would record all conference data (e.g., accepted articles, timetable, attendees) in SGML (standard generalized markup language) in a conference information repository. Then, using the data in the repository, reports could be generated from multiple contexts (e.g., text on a PDA, synthesized voice on cell phone, and XML on a browser). The essence of the system is captured in Figure 17-3.

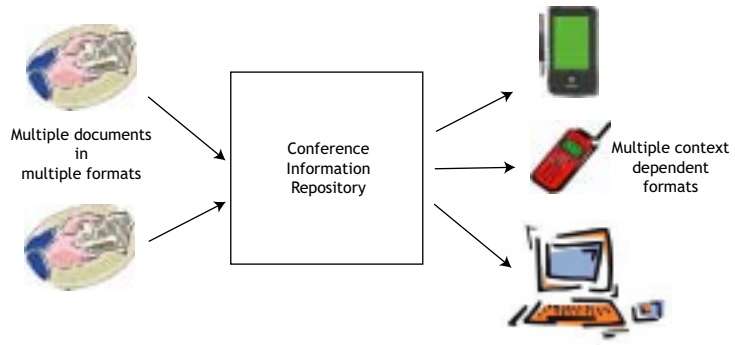

Figure 17-3. The conference support system

The major aspects of the data management problem are that the multiple documents of the conference information repository are prepared using a variety of word processing systems. Furthermore, some authors use a variety of stylesheets and some use none. Consequently, the current system makes it almost impossible to convert accepted articles into a standard content recording language, such as SGML or XML.

Answer the following questions:

1. What system might you implement for getting authors to create documents that can be readily converted to XML? Don't try to be too fancy as it is probably sufficient to capture heading levels as well as the body text.
2. What technology would you select for the conference information repository?
3. What technologies exist for converting XML for different devices (e.g., cell phones, PDAs)?
4. Design the XML schema.