Department of Computer Science and Engineering

CSV888 - Special Module in Database Systems

(2016)

XML (Reference Book- Silberschatz, et al. OR Ullman)

S. Bhalla

[I] Data Types and Data Modelling [2]

• **Structured Data**
The data is stored and processed as per a known scheme
- BibTex
- DBMSs

• **Semi-structured Data**
The data is stored and processed based on the markings
-Markup Data : Latex, HTML, XML, . . .
-XML (XML data + schema)

• **Unstructured Data**
The structure of data is not well defined
- text databases
- Newspaper repositories
- Government documents and file archives

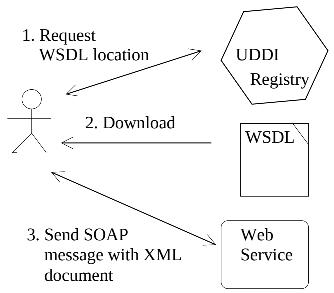
[I] Data Classification [3]

• Accessing programs have knowlegde about →
the structure of data being accessed $\boxed{\text{Structured Data}}$
• Examples : Databases, Bibliography databases, Bio-chemical databases

• Accessing programs have no knowlegde about →
the structure of data being accessed → depend on markup data
$\boxed{\text{SemiStructured Data}}$
• Examples : Web Browzers use HTML, Netscape

• Accessing programs have no knowlegde about →
the structure of data being accessed $\boxed{\text{Unstructured Data}}$
• Music, or Video files
• New Database Research on : Multimedia Databases

[I] Web Services Interaction Diagram [4]



[I] Web Services - Transactions [6]

- Server performs the requested transaction

$\longrightarrow$ sends the result as SOAP message to client

```
<Envelope>
   <Header>
      <transId>1234</transId>
   </Header>
   <Body>
      <AddResponse>
        <c>7</c>
      </AddResponse>
   </Body>
</Envelope>
```

[I] Semi-structured Data [8]

- Web $\longleftrightarrow$ Data

Information interchange, exchange requires $\longrightarrow$ document structure

Semi-structured Data

{ name: "Alan", tel: 2157786, email: "abc@wwexch.net" }

[I] Web Services - Transactions [5]

- Client $\longrightarrow$ sends a request SOAP message to server

```
<Envelope>
   <Header>
      <transId>1234</transId>
   </Header>
   <Body>
      <Add>
        <a>3</a>
        <b>4</b>
      </Add>
   </Body>
</Envelope>
```

[I] Semi-structured Data [7]

- 1. (a) Relational Data $\longrightarrow$ Database Scheme $\longrightarrow$ database

Structure of data $\longrightarrow$ Directory

- 1. (b) Semi-structured Data $\longrightarrow$ Self describing Schema

No separate description

- 2. Program $\longleftrightarrow$ [Data]

Typically - Program uses inbuilt type, scheme

$\longleftrightarrow$ There is no data independence

[I] Web Data - Labels [10]

• Duplicate labels

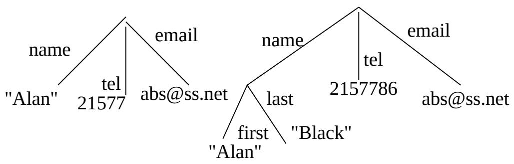
{ name: "Alan", tel: 2157786, tel: 3782535 }

• Many labels or missing labels

{ person:
{name: "Alan", tel: 2157786, email: "abc@wwexch.net" },
person:
{name: {first: "Sara", last:"Green"},
tel: 2136877, email: "sara@the.xyz.edu"},
person:
{name: "Fred", tel: 4257783, Height: 183 }
}

[I] Web Data - structure [9]

• Data elements have a document structure
- similar to documents



[I] Conversion of XML data [12]

• 1. Many ways to represent data in XML

+ easy to convert $\boxed{\text{relation}}$ ⟶ XML data

+ Conversion of XML documents ⟶ database (not simple in few cases)

• 2. XML data (may contain, web data, documents)

+ multiple attributes, missing attributes ...

• 3. Represented as a graph

1. edge labelled
2. rooted and directed acyclic graph (DAG)

with unique path from root (Tree)

[I] Semi-structured Data [11]

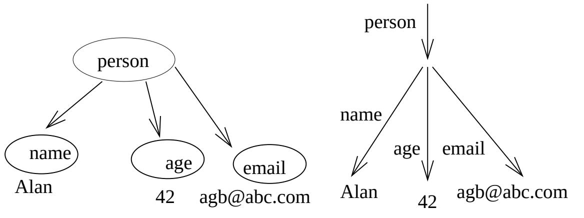
• Type definitions in C++ can not support frequent changes

• XML ⟶ Self Describing

+ Serialization for transfer
+ Byte stream
+ Waste of space ?
+ High interoperability
+ easy web transfer and interpretation
+ type can be dropped (if not in use)

[I] Representing XML Contents [14]

• 4. Search / Query on XML documents (database contents)
+ Graph Searching Algorithms,
a) Depth first search
b) Breadth first search



(a) Tree for XML data

(b) Semi−structured data expression (ssd−expression)

[I] XML: Extensible Markup Language [16]

• Defined by the WWW Consortium (W3C)
• Originally intended as a document markup language not a database language

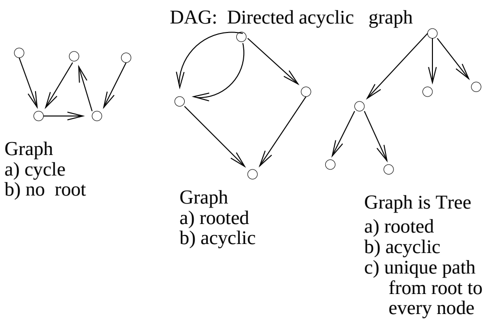
• Documents have tags giving extra information about sections of the document

E.g. <title> XML </title><slide> Introduction . . . </slide>

• Model : Tree Structured data

• Extensible :
Users can add new tags, and separately specify how the tag should be handled for display
• $\boxed{\text{XML Inflience}}$ HTML ⟶ XHTML

[I] Representing XML Contents [13]



• XML uses edge-labeled tree [Hierarchical Edges]

[I] Selected Data in XML form [15]

• $\boxed{\text{Two main approaches for web data}}$

- $\boxed{\text{1. native-XML}}$: storing XML documents in the same format

- $\boxed{\text{2. XML-embeded}}$: storing XML documents after reformatting as relational tables

• 1. native-XML
- $\boxed{\text{Free Software}}$: Software Berkeley DB XML IBM DB2 with Pure XML uses X-Query and HTTP, XML data formats

• 2. ORACLE [Oracle11g has XML support] XDK (XML Development Kit)
many tools in the tool kit
Transformed documents are stored in the relational Database

[I] A Comparision of XML and HTML [17]

XML	HTML	XHTML
Metalanguage	SGML-based	XML-based
describing and structuring data	formatting and displaying data	formatting and displaying data
Tags not predefined	Predefined tags	Predefined tags
Case-sensitive	Case-insensitive	Case-sensitive.
documents must be well formed	documents need not be well formed	documents must be well formed
elements require end tags	Some end tags are optional	elements require end tag

[I] Comparision (Contd.) [18]

XML	HTML	XHTML
Empty elements must be terminated, e.g., <img />	Empty elements are not terminated, e.g. <img >	Empty elements must be terminated, e.g., <img />
Attribute values must be quoted	Unquoted attribute values are allowed	Attribute values must be quoted
No attribute minimalization is allowed	The minimal form of an attribute is allowed	No attribute minimalization is allowed
Tags must be nested properly, without overlapping	Tags may be nested with overlapping	Tags must be nested properly, without overlapping

[I] Querying and Transforming XML Data [19]

• 1. Translation of information from one XML schema to another
• 2. Querying on XML data
• Above two are closely related $\longleftarrow$ handled by same tools

• Standard languages - XML querying/translation
+ **XPath**

- Simple language consisting of path expressions
+ **XSLT** : Extensible Stylesheet Language Transformations
- Simple language for translation (XML $\longleftrightarrow$ XML and XML $\longrightarrow$ HTML)

+ **XQuery**
- An XML query language with a rich set of features
• Many other languages have been proposed;
some of these $\longrightarrow$ basis for the Xquery standard + XML-QL, Quilt, XQL, ...

[I] Tree Model of XML Data [20]

• Query and transformation languages $\longleftarrow$ a tree model of XML data
• An XML document is modeled as a tree (nodes $\longrightarrow$ elements and attributes)

+ Element nodes $\longrightarrow$ children nodes - can be attributes or subelements

+ Text in an element: modeled as a text node child of the element
+ Children of a node: ordered as per the order in the XML document
+ Element and attribute nodes (except for the root node) have a single parent, which is an element node

+ The root node has a single child - the root element of the document

• We use the terminology of nodes, children, parent, siblings, ancestor, descendant, etc., using the above tree model of XML data.

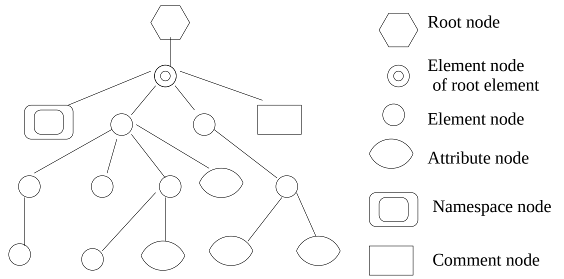
[I] XPath [22]

• XPath is used to address (select) parts of documents using path expressions

• A path expression is a sequence of steps separated by "/"
+ Think of file names in a directory hierarchy
• Result of path expression: set of values that along with their containing elements/attributes match the specified path
• E.g. /bank-2/customer/name evaluated on the bank-2 data

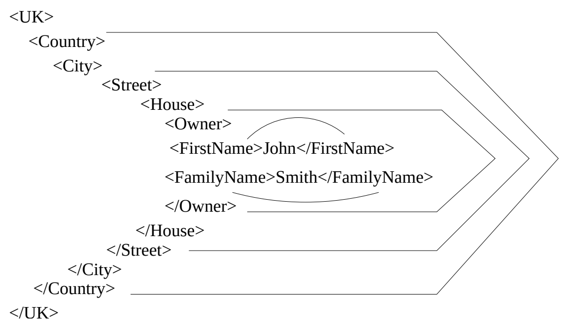
we saw earlier returns
< name > Joe </name >
< name > Mary </name >
• E.g. /bank-2/customer/name/text( )
returns the same names, but without the enclosing tags

[I] Xpath data model [24]



• An abstract data model - on which XML path language is based.

[I] Nesting XML elements [21]



[I] XPath (Cont.) [23]

• The initial "/" denotes root of the document (above the top-level tag)
• Path expressions are evaluated left to right
+ Each step operates on the set of instances produced by the previous step

• Selection predicates may follow any step in a path, in [ ]
+ E.g. /bank-2/account[balance > 400]
- returns account elements with a balance value greater than 400
- /bank-2/account[balance] returns account elements containing a balance subelement

• Attributes are accessed using "@"
+ E.g. /bank-2/account[balance > 400]/@account-number
- returns the account numbers of those accounts with balance > 400
+ IDREF attributes are not dereferenced automatically

[I] More XPath Features [26]

• Operator " | " used to implement union
+ E.g. /bank-2/account/id(@owner) | /bank-2/loan/id(@borrower)
- gives customers with either accounts or loans
- However, "|" cannot be nested inside other operators.

• "//" can be used to skip multiple levels of nodes
+ E.g. /bank-2//name
- finds any name element anywhere under the /bank-2 element, regardless of the element in which it is contained.

• A step in the path can go to:
parents, siblings, ancestors and descendants of the nodes generated by the previous step, not just to the children
+ "//", described above, is a short from for specifying "all descendants"
+ ".." specifies the parent.

[I] XSLT [28]

• A stylesheet stores formatting options for a document,
+ usually separately from document
+ Example HTML style sheet ⟶ font colors and sizes for headings, etc.

• The XML Stylesheet Language (XSL) originally designed
⟶ for generating HTML from XML

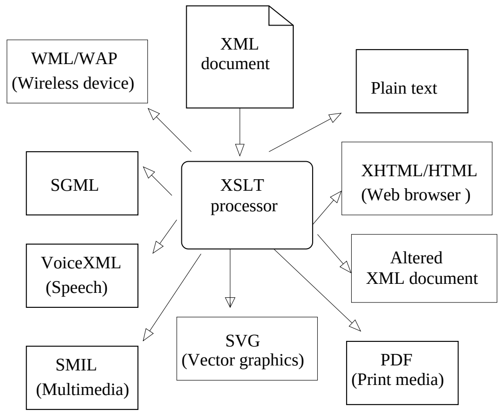
• XSLT is a general-purpose transformation language
+ Can translate XML to XML, and XML to HTML

• XSLT transformations are expressed using rules called templates
+ Templates combine selection using XPath
with construction of results

[I] Functions in XPath [25]

• A set of functions in XPATH function library
+ for converting and translating data
+ results ⟶: 1. Node set 2. Boolean 3. Number 4. String
+ Example: name() - returns a QUALIFIED NAME (name of a node)
• XPath provides several functions
+ The function count() at the end of a path
⟶ counts the number of elements in the set generated by the path
- E.g. /bank-2/account[customer/count() $>$ 2]
= Returns accounts with $>$ 2 customers
= Also, for testing position (1, 2, ..) of node w.r.t. siblings
• Boolean connectives **and, or**, function **not()** can be used in predicates
• IDREFs can be referenced using function id()
+ id() can also be applied to sets of references such as IDREFS
- even to strings containing multiple references separated by blanks
+ E.g. /bank-2/account/id(@owner)
⟶ all customers referred to from owners attribute of account elements.

[I] Use of XSLT for transformation [27]



[I] XSLT Templates [29]

- Example of XSLT template with match and select part

```
<xsl:template match="/bank-2/customer">
    <xsl:value-of select="customer-name"/>
</xsl:template>
<xsl:template match="."/>
```

- The match attribute of xsl:template specifies a pattern in XPath
- Elements in the XML document matching the pattern ⟶ processed by the actions within the xsl:template element

+ xsl:value-of selects (outputs) specified values (here, customer-name)

- If an element matches several templates, only one is used

⟶ depends on a complex priority scheme/user-defined priorities

+ We assume only one template matches any element

- Elements that do not match any template are output as is
- The <xsl:template match="."/> template matches all elements that do not match any other template,

⟶ used to ensure that their contents do not get mixed with output.

[I] Creating XML Output [30]

- Any non-tag text, and any tag that is not in the xsl namespace is output as is

- E.g. to wrap results in new XML elements.

```
<xsl:template match="/bank-2/customer">
<customer>
<xsl:value-of select="customer-name"/>
 </customer>
 </xsl;template>
 <xsl:template match="."/>
```

- Example output:

< customer > John < /customer >
< customer > Mary < /customer >

[I] Structural Recursion [31]

- Action of a template can be to recursively apply templates to the contents of a matched element Example

```
<xsl:template match="/bank">
   <customers>
       <xsl:template apply-templates/>
   </customers >
<xsl:template match="/customer">
     <customer>
        <xsl:value-of select="customer-name"/>
     </customer>
 </xsl:template>
 <xsl:template match="."/>
```

- Example output:

<customers>
< customer > John < /customer >
< customer > Mary < /customer >
< /customers>

[I] Joins in XSLT [32]

- XSLT keys allow elements to be looked up (indexed) by values of subelements or attributes

+ Keys must be declared (with a name) and, the key() function can then be used f or lookup. E.g.

* <xsl:key name= "acctno" match= "account" use= "account-number" / >
* <xsl:value-of select=key("acctno", "A-101")

- Keys permit (some) joins to be expressed in XSLT

```
<xsl:key name="acctno" match="account" use="account-number"/>
 <xsl:key name="custno" match="customer" use="customer-name"/>
 <xsl:template match="depositor".
       <cust-acct>
       <xsl:value-of select=key("custno", "customer-name")/>
       <xsl:value-of select=key("acctno", "account-number")/>
       </cust-acct>
  </xsl:template>
  <xsl:template match="."/>
```

[I] Sorting in XSLT [33]

• Using an xsl:sort directive inside a template causes all elements matching the template to be sorted

+ Sorting is done before applying other templates

• E.g.

```
<xsl:template match="/bank">
     <xsl:apply-templates select="customer">
     <xsl:sort select="customer-name"/>
     </xsl:apply-templates>
</xsl:template>
<xsl:template match="customer">
   <customer>
         <xsl:value-of select="customer-name"/>
         <xsl:value-of select="customer-street"/>
         <xsl:value-of select="customer-city"/>
    </customer>
<xsl:template><xsl:template match="."/>
```

[I] XQuery [34]

• XQuery is a general purpose query language for XML data

• Currently being standardized by the World Wide Web Consortium

+The textbook description is based on a March 2001 draft of the standard.

• Alpha version of XQuery engine available free from Microsoft

• XQuery is derived from the Quilt query language, which itself borrows from SQL, XQL and XML-QL

• XQuery uses a

for ... let ... where .. result ...

syntax

for $\leftrightarrow$ SQL from

where $\leftrightarrow$ SQL where

result $\leftrightarrow$ SQL select

let allows temporary variables $\leftrightarrow$ has no equivalent in SQL

[I] FLWR Syntax in XQuery [35]

• For clause uses XPath expressions, and variable in for clause ranges over values in the set returned by XPath

• Simple FLWR expression in XQuery

+ find all accounts with balance $>$ 400, with each result enclosed in an $< account - number >$ .. $< /account - number >$ tag

for $x in /bank-2/account

let $acctno := $x/@account-number

where $x/balance > 400$

return $< account - number >$ $acctno $< /account - number >$

• Let clause not really needed in this query, and selection can be done In XPath. Query can be written as:

for $x in /bank-2/account[balance>400]

return $< account - number >$ $X/@account-number

$< /account - number >$

[I] Path Expressions and Functions [36]

• Path expressions are used to bind variables in the for clause, but can also be used in other places

+ E.g. path expressions can be used in let clause, to bind variables to results of path expressions

• The function distinct( ) can be used to removed duplicates in path expression results

• The function document(name) returns root of named document

+ E.g. document("bank-2.xml")/bank-2/account

• Aggregate functions such as sum( ) and count( ) can be applied to path expression results

• XQuery does not support groupby, but the same effect can be got by nested queries, with nested FLWR expressions within a result clause

! More features

[I] Changing Nesting Structure [38]

- The following query converts data from the flat structure for bankinformation into the nested structure used in bank-1

```
<bank-1>
   for $c in /bank/customer
   return
     <customer>
      $c/*
      for $d in /bank/depositor[customer-name = $c/customer-name],
          $a in /bank/account[account-number=$d/account-number]
      return $a
    </customer>
</bank-1>
```

- $c/* denotes all the children of the node to which $c is bound, without the enclosing top-level tag
- Exercise for reader: write a nested query to find sum of account balances, grouped by branch.

[I] Sorting in XQuery [40]

- Sortby clause can be used at the end of any expression. E.g. to return customers sorted by name

for $c in /bank/customer
return $< customer >$ $c/* $< /customer >$ sortby(name)

- Can sort at multiple levels of nesting (sort by customer-name, and by account-number within each customer)

```
<bank-1>
for $c in /bank/customer
return
    <customer>
      $c/*
      for $d in /bank/depositor[customer-name=$c/customer-name],
          $a in /bank/account[account-number=$d/account-number]
      return <account> $a/* </account> sortby(account-number)
   </customer> sortby(customer-name)
 </bank-1>
```

[I] Joins [37]

- Joins are specified in a manner very similar to SQL

for $b in /bank/account,
$c in /bank/customer,
$d in /bank/depositor
where $a/account-number = $d/account-number
and $c/customer-name = $d/customer-name
return $<$ cust-acct $>$ $c $</$cust-acct $>$

- The same query can be expressed with the selections specified as XPath selections:

for $a in /bank/account
$c in /bank/customer
$d in /bank/depositor[
        account-number = $a/account-number and
        customer-name = $c/customer-name]
return $<$ cust-acct $>$ $c a<$/cust-acct $>$

[I] XQuery Path Expressions [39]

- $c/text() gives text content of an element without any subelements/tags
- XQuery path expressions support the "$->$" operator for dereferencing IDREFs

+ Equivalent to the id( ) function of XPath, but simpler to use
+ Can be applied to a set of IDREFs to get a set of results
+ June 2001 version of standard has changed "$->$" to "$=>$"

[I] Application Program Interface [42]

• There are two standard application program interfaces to XML data:
+ SAX (Simple API for XML)
* Based on parser model, user provides event handlers for parsing events
- E.g. start of element, end of element
- Not suitable for database applications
+ DOM (Document Object Model)
* XML data is parsed into a tree representation
* Variety of functions provided for traversing the DOM tree
* E.g.: Java DOM API provides Node class with methods

```
getParentNode( ), getFirstChild( ), getNextSibling( )
getAttribute( ), getData( ) (for text node)
getElementsByTagName( ), ...
```

• Also provides functions for updating DOM tree

[I] Storing XML in Relational Databases [44]

• Store as string
! E.g. store each top level element as a string field of a tuple in a database
+ Use a single relation to store all elements, or
+ Use a separate relation for each top-level element type
- E.g. account, customer, depositor
- Indexing:
>> Store values of subelements/attributes to be indexed, such as customer-name and account-number as extra fields of the relation, and build indices
>> Oracle 9 supports function indices which use the result of a function as the key value. Here, the function should return the value of the required subelement/attribute

[I] Functions and Other XQuery Features [41]

• User defined functions with the type system of XMLSchema

```
function balances(xsd:string $c) returns list(xsd:numeric) {
   for $d in /bank/depositor[customer-name = $c],
       $a in /bank/account[account-number=$d/account-number]
   return $a/balance
}
```

• Types are optional for function parameters and return values
• Universal and existential quantification in where clause predicates
+ some $e in path satisfies P
+ every $e in path satisfies P
• XQuery also supports If-then-else clauses

[I] Storage of XML Data [43]

• XML data can be stored in
+ Non-relational data stores
* Flat files
- Natural for storing XML
- But has all problems discussed in Chapter 1 (no concurrency, no recovery, ...)
* XML database
- Database built specifically for storing XML data, supporting DOM model and declarative querying
- Currently no commercial-grade systems
+ Relational databases
- Data must be translated into relational form
- Advantage: mature database systems
- Disadvantages: overhead of translating data and queries

[I] Storing XML in Relational Databases (Contd) [45]

+ Benefits:
! Can store any XML data even without DTD
! As long as there are many top-level elements in a document, strings are small compared to full document, allowing faster access to individual elements.
+ Drawback: Need to parse strings to access values inside the elements; parsing is slow.

[I] Storing XML as Relations (Cont.) [46]

• Tree representation: model XML data as tree and store using relations
nodes(id, type, label, value)
child (child-id, parent-id)
+ Each element/attribute is given a unique identifier
+ Type indicates element/attribute
+ Label specifies the tag name of the element/name of attribute
+ Value is the text value of the element/attribute
+ The relation child notes the parent-child relationships in the tree
- Can add an extra attribute to child to record ordering of children
+ Benefit: Can store any XML data, even without DTD
+ Drawbacks:
- Data is broken up into too many pieces, increasing space overheads
- Even simple queries require a large number of joins, which can be slow

[I] Storing XML in Relations (Cont.) [47]

• Map to relations
+ If DTD of document is known, can map data to relation
+ Bottom-level elements and attributes are mapped to attributes of relations
+ A relation is created for each element type
! An id attribute to store a unique id for each element
! all element attributes become relation attributes
! All subelements that occur only once become attributes
- For text-valued subelements, store the text as attribute value
- For complex subelements, store the id of the subelement
! Subelements that can occur multiple times represented in a separate table
- Similar to handling of multivalued attributes when converting ER diagrams to tables

[I] Storing XML in Relations (Cont.) [48]

+ Benefits:
! Efficient storage
! Can translate XML queries into SQL, execute efficiently, and then translate SQL results back to XML
+ Drawbacks: need to know DTD, translation overheads still present

[I] References - Books for reading [49]

• Database Systems Concepts, by A. Silberschatz, H. F. Korth, S. Sudarshan, McGraw Hill, 4th/ 5th/ 6th edition (2011)

•Database Systems: The Complete Book, H. Garcia-Molina, J. Ullman, and J. Widom. The second edition, 2008

[I] References - Main Web Resources [50]

The list below comprises only the major Web sites, several of which can serve as portals to further online resources.

http://www.w3.org/
The Web site of the World Wide Web Consortium (W3C) contains all the latest W3C standards, related to XML technologies and the Semantic Web.

http://www.xml.org/
The XML Industry Portal, hosted by OASIS, provides an independent resource for news and information about the industrial and commercial applications of XML.

[I] References - Main Web Resources [51]

http://www.oasis-open.org/cover/
The XML Cover Pages is a comprehensive Web-accessible reference collection.

http://www.xml.com/
Articles, tutorials, software and other XML-related information hosted by O'Reilly.

http://www.semanticweb.org/
The portal of the Semantic Web community.

http://www.garshol.priv.no/download/xmltools/
A comprehensive list of free XML tools and software by Lars Marius Garshol.

[I] References - Main Web Resources [52]

http://wdvl.com/Authoring/Languages/XML/
The XML section of the Web Developer's Virtual Library (WDVL), which contains links to major XML sites and specifications.

http://www.ucc.ie/xml/
A list of Frequently Asked Questions (FAQ) about XML.

http://msdn.microsoft.com/xml/
Microsoft Developer Network's XML developer center, which aggregates content and resources about XML.

[I] References - Main Web Resources [53]

http://www.w3schools.com/
Free tutorials on XML, XSL, XPath and other XML
technologies.

http://www.xmlbooks.com/
Charles F. Goldfarb's "All the XML Books in Print"
Web site.

http://www.oasis-open.org/
The Web site of OASIS (the Organization for the
Advancement of Structured Information
Standards).

[I] References - Main Web Resources [54]

http://xml.apache.org/
The Apache XML Project (part of the Apache Software Foundation).

http://java.sun.com/xml/
Java technology and XML.

http://metalab.unc.edu/xml/
Cafe con Leche's XML news and resources.

http://www.alphaworks.ibm.com/xml/
IBM's XML Web site for early adopter developers.

http://www.idealliance.org/XMLRoadmap/
WEB/TOC/xmlrotoc.htm
The "XML Road Map" a guide to XML standards.

[I] References - Main Web Resources [55]

http://www.ontoweb.org/
The Web site of a European Union-funded project
about ontology-based information exchange for
knowledge management and electronic commerce.

http://www.perfectxml.com/
A collection of information on different aspects of XML.

http://www.xml-acronym-demystifier.org/
A project that is intended to collect and published
information about the various acronyms prevalent

http://www.xmlfiles.com/
XML-related resources.

[I] References - Main Web Resources [56]

http://www.xmlhack.com/
A news Web site for XML developers.

http://www.xmlmag.com/
An online XML magazine.

http://www.xmlsoftware.com/
An index of XML-related software resources.

http://www.egroups.com/group/xml-dev/
An informal unmoderated list to support those who are
interested in the implementation and development of XML.

http://www.xml.org/xml-dev/
The XML developers' mailing list.