Introduction

Web Data Management and Distribution

Serge Abiteboul   Ioana Manolescu   Philippe Rigaux
Marie-Christine Rousset   Pierre Senellart

Web Data Management and Distribution
http://webdam.inria.fr/textbook

November 15, 2010

Outline

Web data handling

Web data = by far the largest information system ever seen, and a fantastic means of sharing information.

- Billions of textual documents, images, PDF, multimedia files, provided and updated by millions of institutions and individuals.

- An anarchical process which results in highly heterogeneous data organization, steadily growing and extending to meet new requirements.

- New usage and applications appear every day: yesterday P2P file sharing, today social networking, tomorrow ?

The challenge, under a data management perspective: master the size and extreme variability of Web information, and make it usable.

The role of XML

Web data management has been primarily based on HTML, which describes presentation.

- HTML is appropriate for humans: allows sophisticated output and interaction with textual documents and images;
- HTML falls short when it comes to software exploitation of data.

XML describes content, and promotes machine-to-machine communication and data exchange

- XML is a generic data format, apt to be specialized for a wide range of fields,
  ⇒ (X)HTML is a specialized XML dialect for data presentation
- XML makes easier data integration, since data from diferent sources now share a common format;
- XML comes equipped with many softwares, APIs and tools.

Perspective of the course

The course develops an XML perspective of the management of heterogeneous data (e.g., Web data) in a distributed environment. We introduce models, languages, architectures and techniques to fulfill the following goals:

- flexible data representation and retrieval
  XML is viewed as a new data model, both more powerful and more flexible than the relational one

- data exchange and integration
  XML data can be serialized and restructured for better exchange between systems, and integration of multiple data sources

- efficient distributed computing and storage
  XML can be the glue for high-level description of distributed repositories; this calls for efficient storage, indexing of management.

Outline

Semi-structured data model

A data model, based on graphs, for representing both regular and irregular data.

Basic ideas

Self-describing data. The content comes with its own description;
⇒ contrast with the relational model, where schema and content are represented separately.

Flexible typing. Data may be typed (i.e., "such nodes are integer values" or "this part of the graph complies to this description"); often no typing, or a very flexible one

Serialized form. The graph representation is associated to a serialized form, convenient for exchanges in an heterogeneous environment.

Self-describing data

Starting point: association lists, i.e., records of label-value pairs.

```
{name: "Alan", tel: 2157786,  email: "agb@abc.com"}
```

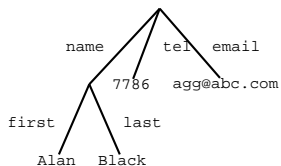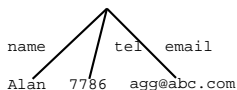Natural extension: values may themselves be other structures:

```
{name: {first: "Alan",  last: "Black"},
    tel: 2157786,
    email: "agb@abc.com"}
```

Further extension: allow duplicate labels.

```
    {name: "alan''", tel:  2157786, tel: 2498762 }
```
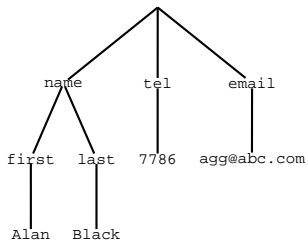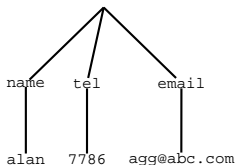
Tree-based representation

Data can be graphically represented as trees: label structure can be captured by tree edges, and values reside at leaves.

Tree-based representation: labels as nodes

Another choice is to represent both labels and values as vertices.

Remark

The XML data model adopts this latter representation.

Representation of regular data

The syntax makes it easy to describe sets of tuples as in:

```
{ person:  {name: "alan", phone: 3127786, email: "alan@abc.com"}
  person:  {name: "sara", phone: 2136877, email: "sara@xyz.edu"}
  person:  {name: "fred", phone: 7786312, email: "fd@ac.uk"} }
```

Remark

1. relational data can be represented
2. for regular data, the semi-structure representation is highly redundant.

Representation of irregular data

Many possible variations in the structure: missing values, duplicates, changes, etc.

```
{person: {name: "alan", phone: 3127786, email: "agg@abc.com"},
 person: &314
         {name: {first: "Sara", last: "Green"                 },
          phone: 2136877,
          email: "sara@math.xyz.edu",
          spouse: &443                                          },
 person: &443
         {name: "fred", Phone: 7786312,  Height: 183,
          spouse: &314                                          }}
```

Node identity

Nodes can be identified, and referred to by their identity. Cycles and objects models can be described as well.

XML in brief

XML is the World-Wide-Web Consortium (W3C) standard for Web data exchange.

- XML documents can be serialized in a normalized encoding (typically iso-8859-1, or utf-8), and safely transmitted on the Internet.
- XML is a generic format, which can be specialized in "dialects" for specific domain (e.g., XHTML, see further)
- The W3C promotes companion standards: DOM (object model), XSchema (typing), XPath (path expression), XSLT (restructuring), XQuery (query language), and many others.

Remark

1. XML is a simplified version of SGML, a long-term used language for technical documents.

2. HTML, up to version 4.0, is also a variant of SGML. The successor of HTML 4.0, is XHTML, an XML dialect.

XML documents

An XML document is a labeled, unranked, ordered tree:

Labeled means that some annotation, the label, is attached to each node.

Unranked means that there is no a priori bound on the number of children of a node.
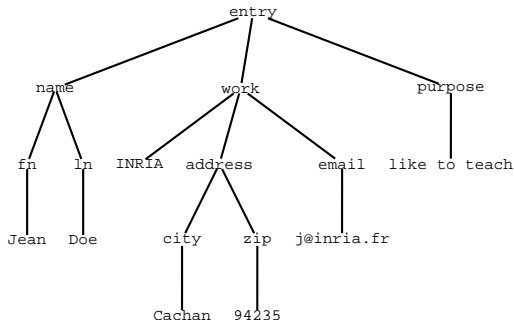
Ordered means that there is an order between the children of each node.

XML specifies nothing more than a syntax: no meaning is attached to the labels.

A dialect, on the other hand, associates a meaning to labels (e.g., title in XHTML).

XML documents are trees

Applications view an XML document as a labeled, unranked, ordered tree:

Remark

Some low-level softwares work on the serialized representation of XML documents, notably SAX (a parser and an API).

Serialized representation of XML document

Documents can be serialized, such as, for instance:

```
<entry><name><fn>Jean</fn><ln>Doe</ln></name>INRIA<adress><city>
Cachan</city><zip>94235</zip></adress><email>j@inria.fr</email>
</job><purpose>like to teach</purpose></entry>
```

or with some beautification as:

```
<entry>
    <name>
        <fn>Jean</fn>
        <ln>Doe</ln> </name>
    <work>
        INRIA
        <adress>
            <city>Cachan</city>
            <zip>94235</zip> </adress>
        <email>j@inria.fr</email> </work>
    <purpose>like to teach</purpose>
</entry>
```

XML describes structured content

Applications cannot interpret unstructured content:

```
The book ``Fundations of Databases'', written by Serge Abiteboul,
Rick Hull and Victor Vianu, published in 1995 by Addison-Wesley
```

XML provides a means to structure this content:

```
<bibliography>
  <book>
    <title> Foundations of Databases </title>
    <author> Abiteboul </author>
    <author> Hull </author>
    <author> Vianu </author>
    <publisher> Addison Wesley </publisher>
    <year> 1995 </year> </book>
  <book>...</book>
</bibliography>
```

Now, an application can access the XML tree, extract some parts,
rename the labels, reorganize the content into another structure, etc.

Applications associate semantics to XML docs

The description of a letter

Letter document

```
<letter>
  <header>
    <author>...</author>
    <date>...</date>
    <recipient>...</recipient>
    <cc>...<cc>
  </header>
<body>
    <text>...</text>
    <signature>...</signature>
</body>
</letter>
```

Applications associate semantics to XML docs (2)

Letter style sheet
  if *letter* then ...
  if *header* then ...
  if *author* then ...
  if *date* then ...
  if *recipient* then ...
  if *cc* then ...
  if *body* then ...
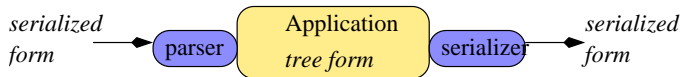 if *text* then ...
  if *signature* then ...

Some software then produces the actual letter to mail or email.

Outline

Serialized form, tree form

Typically, an application gets a document in *serialized form*, parse it in *tree form*, and *serializes* it back at the end.

- The serialized form is a textual, linear representation of the tree; it complies to a (sometimes complicated) syntax;
- There exist an object-oriented model for the tree form: the *Documet Object Model* (W3C).

Remark

We present here the most significant aspects of both the syntax and the DOM. Details can be found in the W3C documents.

The syntax for serialized document, in brief

Four examples of XML documents (separated by blank lines) are:

```
<document/>

<document> Hello World! </document>

<document>
  <salutation> Hello World! </salutation>
</document>

<?xml version="1.0" encoding="utf-8" ?>
<document>
  <salutation color="blue"> Hello World! </salutation>
</document>
```

Last example shows the *prologue*, useful for XML parsers (it gives in particular the document encoding).
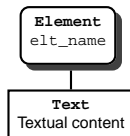
From serialized to tree form: text and elements

The basic components of an XML document are *element* and *text*.
Here is an *element*, whose content is a *text*.

```
<elt_name>
  Textual  content
</elt_name>
```

The tree form of the document, modeled in DOM: each node has a **type**, either **Document** or **Text**.
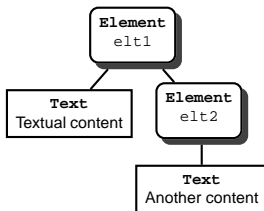
From serialized to tree form: nesting elements

The content of an element is

1. the part between the opening and ending tags (in serialized form),
2. the subtree rooted at the corresponding **Element** node (in DOM).

The content may range from atomic text, to any recursive combination of text and elements (and gadgets, e.g., comments).

Example of an element nested in another element.

```
<elt1>
  Textual  content
  <elt2>
    Another content
  </elt2>
</elt1>
```

Note the correspondence between opening/ending tags, and subtrees in the DOM representation.
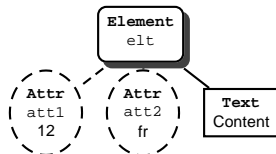
From serialized to tree form: attributes

Attributes are pairs of name/value attached to an element.

1. as part of the opening tag in the serialized form,
2. as special child nodes of the **Element** node (in DOM).

The content of an attribute is always atomic text (no nesting).

An element with two attributes.

```
<elt1 att1='12' att2='fr'>
  Textual  content
</elt1>
```

Unlike elements, attributes are *not* ordered, and there cannot be two attributes with the same name in an element.

From serialized to tree form: the document root

The first line of the serialized form must *always* be the *prologue* if a prologue is present:

```
<?xml version="1.0"encoding="utf-8"?>
```

and the document content must *always* be enclosed in a single opening/ending tag, called the *element root*.

A document with its pro-logue, and element root.

```
<?xml version="1.0"
    encoding="utf-8" ?>
<elt>
  Document content.
</elt>
```

Note: there may be other syntactic objects after the prologue (processing instructions).

In the DOM representation, the prologue appears as a **Document** node, called the *root node*.
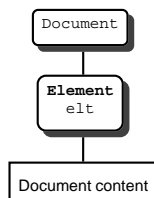
Summary: syntax and vocabulary

Serialized form

- A document begins with a prologue,
- It consists of a single upper-level tag,
- Each *opening tag* `<name>` has a corresponding *closing tag* `</name>`; everything between is either text or properly enclosed tag content.

Tree form

- A document is a tree with a *root node* (**Document** node in DOM),
- The root node has one and only one element child (**Element** node in DOM), called the *element root*)
- Each element node is the root of a *subtree* which represents its structured *content*

Remark

Other syntactic aspects, not detailed here, pertain to the physical organization of a document. See the lecture notes.

Summary: syntax and semantics

XML provides a syntax

The core of the syntax is the element name

Element names have no a priori semantics

Applications assign them a semantics

Entities and references

Entities are used for the physical organization of a document.

An entity is declared (in the document type), then referenced.

```
<?xml version="1.0" encoding="ISO-8859-1"?>

<!DOCTYPE a [
  <!ENTITY  myName "John Doe">
  <!ENTITY  mySignature SYSTEM "signature.xml">
]>

<a>
  My name is &myName;.

  &mySignature;
</a>
```

Predefined entities

Several symbols cannot be directly used in an XML document, since they would be misinterpreted by the parser.

They must be introduced as entity references.

Declaration	Reference	Symbol.
`<!ENTITY lt "<">`	`<`	`<`
`<!ENTITY gt ">">`	`>`	`>`
`<!ENTITY amp "&">`	`&`	`&`
`<!ENTITY apos "'">`	`'`	`'`
`<!ENTITY quot """>`	`"`	`"`

Comments and instructions

Comments can be put at any place in the serialized form.

```
<!--This is a comment -->
```

They appear as **Comment** nodes in the DOM tree (they are typically ignored by applications).

Processing instructions: specific commands, useful for some applications, simply ignored by others.

The following instruction requires the transformation of the document by an XSLT stylesheet

```
<?xml-stylesheet href="prog.xslt"type="text/xslt"?>
```

Literal sections

Usually, the content of an element is *parsed* to analyse its structure.
Problem: what if we do *not* want the content to be parsed?

```
<?xml version='1.0'?>
<progam>
if ((i < 5) && (j > 6))
   printf("error");
</program>
```

Solution: use entities for all special symbols; or prevent parsing with a *literal section*.

```
<?xml version='1.0'?>
<program>
<![CDATA[if ((i < 5) && (j > 6))
          printf("error");
]]>
</program>
```

Outline

To type or not to type

What kind of data: very regular one (as in relational databases), less regular (hypertext systems) - all kind of data from very structured to very unstructured.

What kind of typing (unlike in relational systems)

- Possibly irregular, partial, tolerant, flexible
- Possibly evolving
- Possibly very large and complex
- Ignored by some applications such as keyword search.

Typing is not compulsory.

Type declaration

XML documents *may* be typed, although they do not need to. The simplest (and oldest) typing mechanism is based on *Document Type Definitions* (DTD).

A DTD may be specified in the prologue with the keyword DOCTYPE using an ad hoc syntax.

A document with proper opening and closing of tags is said to be well-formed.

- `<a><b>...</b><c>...</c></a>` is well-formed.
- `<a><b>...</b><c>...</a></c>` is not.
- `<a><b>...</b></a><a>...</a>` is not.

A document that conforms to its DTD is said to be valid

Document Type Definition

An example of valid document (the root element matches the
DOCTYPE declaration).

```
<?xml version="1.0" standalone="yes" ?>
<!-- Example of a DTD -->
<!DOCTYPE email [
  <!ELEMENT  email ( header, body )>
  <!ELEMENT header ( from, to, cc? )>
  <!ELEMENT to (#PCDATA)>
  <!ELEMENT from (#PCDATA)>
  <!ELEMENT cc (#PCDATA)>
  <!ELEMENT body paragraph* >
  <!ELEMENT paragraph (#PCDATA)> ]>
<email>
  <header>
     <from> af@abc.com </from>
     <to> zd@ugh.com </to> </header>
  <body> </body> </email>
```

Document Type Definition

A DTD may also be specified externally using an URI.

```
<!DOCTYPE docname SYSTEM "DTD-URI"
[local-declarations]>
```

- docname is the name of the element root
- DTD-URI is the URI of the file that contains the DTD
- local-declarations are local declarations (mostly for entities.)

Interpreting labels: Namespaces

A particular label, e.g., *job*, may denote different notions in different contexts, e.g., a hiring agency or a computer ASP (application service provider).

The notion of namespace is used to distinguish them.

```
<doc xmlns:hire='http://a.hire.com/schema'
     xmlns:asp='http://b.asp.com/schema'  >
...
<hire:job> ... </hire:job> ...
<asp:job> ... </asp:job> ...
</doc>
```

DTD vs. XML schema

DTD: old style typing, still very used
XML schema: more modern, used e.g. in Web services
DTD:

```
<!ELEMENT note (to, from, heading, body)>
<!ELEMENT to (#PCDATA)>
<!ELEMENT from (#PCDATA)>
<!ELEMENT heading (#PCDATA)>
<!ELEMENT body (#PCDATA)>
```

DTD vs. XML schema

The same structure in XML schema (an XML dialect)

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="note">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="to" type="xs:string"
          minOccurs='1' maxOccurs='1'/>
        <xs:element name="from" type="xs:string"/>
        <xs:element name="heading" type="xs:string"/>
        <xs:element name="body" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Outline

Popular XML dialects

RSS is an XML dialect for describing content updates that is heavily used for blog entries, news headlines or podcasts.

WML, the Wireless Mark-up Language, is used in Web sites by wireless applications based on the Wireless Application Protocol (WAP).

MathML, the mathematical mark-up language, is an XML dialect for describing mathematical notation and capturing both its structure and content.

Xlink, the XML Linking Language, is an XML dialect for defining hyperlinks between XML documents. These links are expressed in XML and may be introduced inside XML documents.

SVG, Scalable Vector Graphics, is an XML dialect for describing two-dimensional vector graphics, both static and animated. With SVG, images may contain outbound hyperlinks in XLinks.

XML standards

SAX, the Simple API for XML, sees an XML document as a
sequence of tokens (its serialization).

DOM, the Document Object Model, is an object model for
representing (HTML and) XML document independently
of the programming language.

XPath, the XML Path Language, that we will study, is a language
for addressing portions of an XML document.

XQuery, that we will study, is a flexible query language for
extracting information from collections of XML documents.

XSLT, the Extensible Stylesheet Language Transformations, that
we will study, is a language for specifying the
transformation of XML documents into other XML
documents.

Web services provide interoperability between machines based on
Web protocols.

Zoom: DOM

Document Object Model - DOM

Document model that is tree-based

Used in APIs for different programming languages (e.g. Java)

Object-oriented access to the content:

- Root of the document
- First child of a node (with a particular label)
- Next one (with a particular label)
- Parent of a node
- Attribute of a node...

Zoom: XPATH

Language for expressing paths in an XML document

- Navigation: child, descendant, parent, ancestor
- Tests on the nature of the node
- More complex selection predicates

Means to specify portions of a document

Basic tool for other XML languages: Xlink, XSLT, Xquery

Zoom: XLINK

XML Linking Language

Advanced hypertext primitives

Allows inserting in XML documents descriptions of links to external Web resources

Simple monodirectional links ala (HREF) HTML

Mulridirectional links

XLink relies on XPath for addressing portions of XML documents

Remark
XML trees + XLINK $\Rightarrow$ graph

Zoom: XSLT

Transformation langage: "Perl for XML"

An XSLT style sheet includes a set of transformation rules:
pattern/template

Pattern: based on XPATH expressions; it specifies a structural context
in the tree

Template: specifies what should be produced

Principle: when a pattern is matched in the source document, the
corresponding templates produces some data

Zoom: XQuery

Query language: "SQL for XML"

Like SQL: select portions of the data and reconstruct a result

Query structure: FLW (pronounced "flower")

Exemple
```
FOR $p IN document("bib.xml")//publisher
LET $b := document("bib.xml")//book[publisher = $p]
WHERE count($b) > 100
RETURN $p
```

$p : scans the sequence of publishers
$b : scans the sequence of books for a publisher
WHERE filters out some publishers
RETURN constructs the result

Generic XML tools

API

Parsers and type chechers

GUI (Graphical User Interfaces)

Editors

XML diff

XML wiki

Etc.

Facilitate the development of applications specific to particular XML dialects.

Outline

1. Preliminaries

2. XML, a semi-structured data model

3. XML syntax

4. Typing

5. The XML World

6. Use cases

Exploiting XML content

Publishing: an XML document can easily be converted to another XML document (same content, but another structure)
$\Rightarrow$ Web publishing is the process of transforming XML documents to XHTML.

Integration: XML documents from many sources can be transformed in a common dialect, and constitute a collection.
$\Rightarrow$ Search engines, or portals, provide browsing and searching services on collections of XML documents.

Distributed Data Processing: many softwares can be adapted to consume/produce XML-represented data.
$\Rightarrow$ Web services provide remote services for XML data processing.
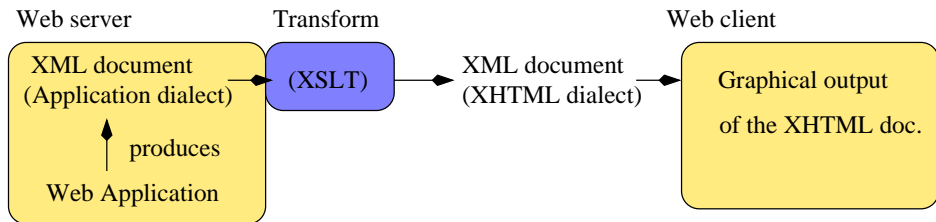
Genericity of softwares and APIs

XML comes with many tools that are generic: A software or API for XML documents applies to all the possible XML dialects.

Web Publishing: restructuring to XHTML

The Web application produces some XML content, structured in some application-dependent dialect, on the server.

In a second phase, the XML content is transformed in an XHTML document that can be visualized by humans.

The transformation is typically expressed in XSLT, and can be processed either on the server or on the client.

Web server   Transform   Web client

XML document (Application dialect) → (XSLT) → XML document (XHTML dialect) → Graphical output of the XHTML doc.

produces

Web Application

Web publishing: content + presentation instructions

The following document is an XHTML version of the bibliographic content presented above:

```
<h1> Bibliography </h1>
  <p> <i> Foundations of Databases </i>
         Abiteboul, Hull, Vianu
         <br/> Addison Wesley, 1995 </p>
  <p> <i> Data on the Web </i>
         Abiteoul, Buneman, Suciu
         <br/> Morgan Kaufmann, 1999 </p>
```

Now the labels belong to the XHTML dialect, and can be interpreted by a Web browser.
Test: biblio.html

Note that the "meaning" of labels is restricted to presentation purposes. It becomes complicated for a software to distinguish the name of authors.
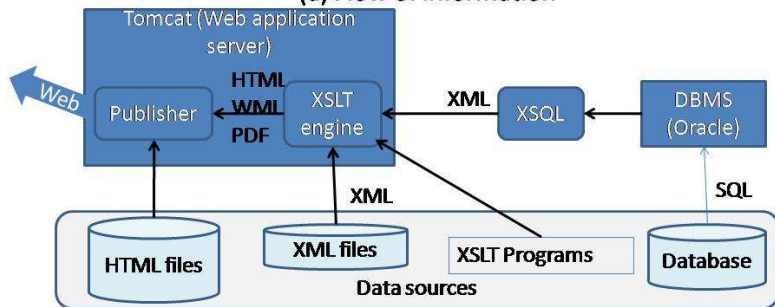
Web publishing

The same content may be published using different means:

- Web publishing: XML $\Rightarrow$ XHTML
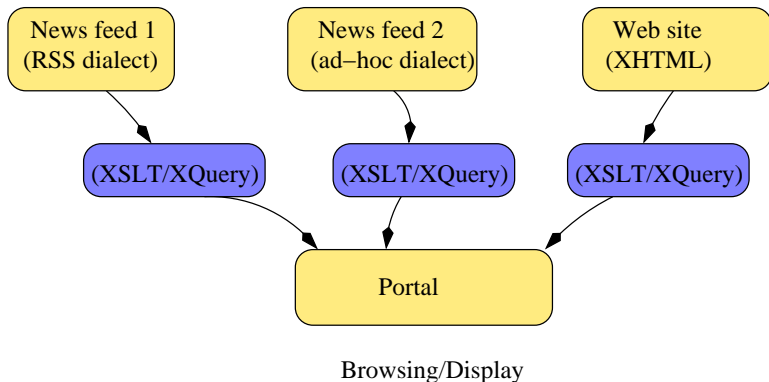- WAP (Wireless Application Protocol): XML $\Rightarrow$ WML

# Web publishing, the big picture
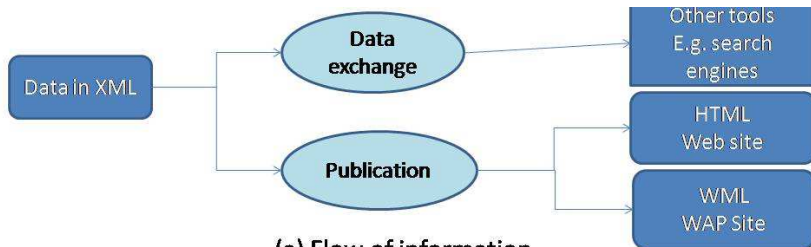
(a) Flow of information

Web Integration: gluing together heterogeneous sources

Browsing/Display

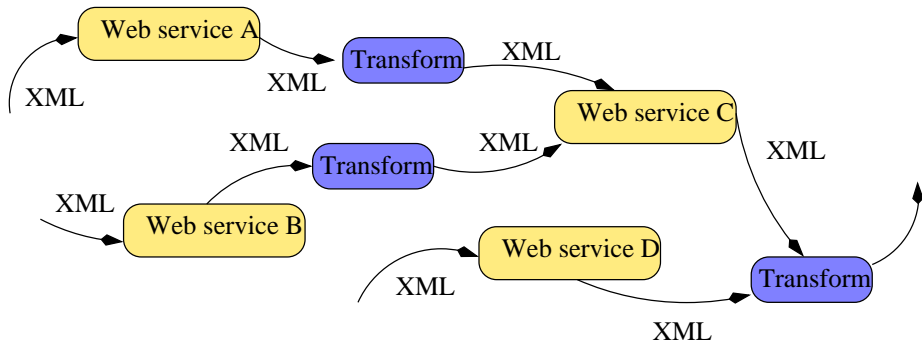The portal receives (possibly continuously) XML-structured content, each source using its own dialect.

Each feed provides some content, extracted with XSLT or XQuery, or any convenient XML processing tool (e.g., SAX).
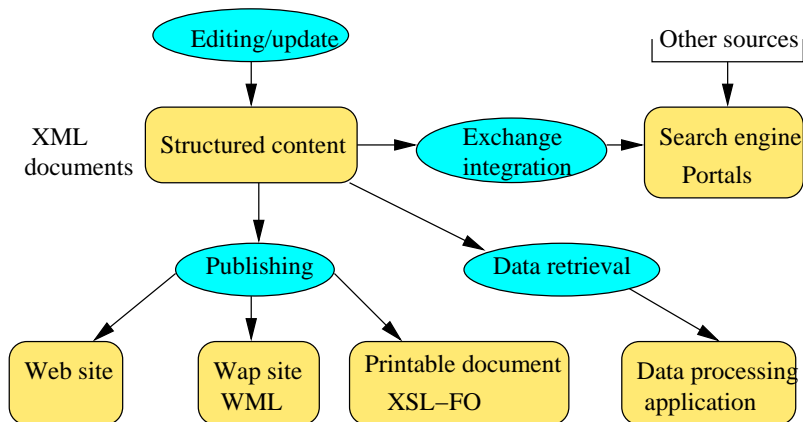
Data integration, a larger perspective
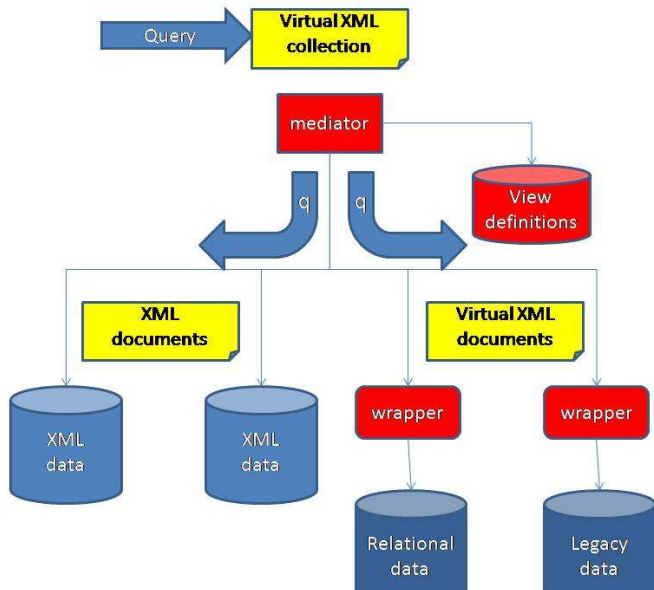
Distributed Data Management with XML

XML encoding is used to *exchange* information between applications.

A specific dialect, WDSL, is used to describe Web Services Interfaces.

Exploiting XML documents: the big picture

... and many other possible use of XML flexibility.

Bibliography

Document Object Model. w3.org/DOM.
World wide web consortium. w3.org/.
      Extensible Markup Language. w3.org/XML.
      XML Schema. w3.org/XML/Schema.
      XML Query (XQuery). w3.org/XML/Query.
      Extensible Stylesheet Language. w3.org/Style/XSL.

S. Abiteboul, P. Buneman, and D. Suciu. Data on the Web: From
Relations to Semistructured Data and XML. Morgan-Kaufman, New
York, 1999.
A. Michard, XML - langage et applications, Eyrolles, 2000.

The end for today

Merci