eXtensible Markup Language

(XML)

OUTLINE

- Introduction
- Xml Vs HTML
- Structure of XML
- Querying and Transformation in XML
- Storage of XML
- Application XML
- Conclusion

Introduction

- eXtensible Markup Language.
- e**X**tensible
- Markup
- A set of rules for encoding documents in a format that is both human readable & machine readable.
- Defined by **WWW** consortium(W3C) in 1996.
- XML tags are **not predefined**. You must define your own tags
- Bridge for Data exchange on the web.

- XML emerged from the confluence of two technologies:
- SGML
- HTML
- XML is not a replacement of HTML.
- What does the XML do??
- Used to structure and describe information.
- Used as a way to interchange data between desperate system.
- XML is a piece of larger system….
- MathML and WML language derived from XML.

Comparison

XML	HTML
eXtensible set of Tags.	Fixed set of Tags.
XML was designed to transport and store data, with focus on what data is….	HTML was designed to display data, with focus on how data looks….
Content Oriented	Presentation Oriented
In XML it is mandatory to close each and every tag.	In HTML it is not required.
XML is case sensitive.	HTML is case insensitive

Structure of XML

- **Declaration:**

  <xml version="1.0" encoding="UTF-16" standalone="yes"?>

- **Elements**: A pair of matching start and end tags and all text appears between them.

  <account> … <balance> …. </balance> </account>

- **Attributes:** provide additional information about the element.

- **Comments:** We can insert comments anywhere in an XML document.

  Start with **<!--** and end with **-->**

- **Entity References:** Start with **'&'** and end with **';'**

<xml version="1.0" encoding="UTF-16" standalone="yes"?>

<article>

<author>Navathe </author>

<title> Web in 10 years </title>

<text>

<abstract> In order to evolve …</abstract>

<section number='1' title="introduction">

<index> Web <index> provide the universal.

</section>

</text>

</article>

Example

- <xml version="1.0" encoding="UTF-16" standalone="yes"?>
- <article>
- <author>Navathe </author>
- <title> Web in 10 years </title>
- <text>
- <abstract> In order to evolve …</abstract>
- <section number='1' title="introduction">
- <index> Web <index> provide the universal.
- </section>
- </text>
- </article>

Freely definable tags

Start Tag

Attributes with Name and Value

End tag

XML Schema

- **Database** has **schema,** used to constrain what information can be stored in the database and the data types of the stored information..
- Schemas are very important for XML data exchange
  - Otherwise, a site cannot automatically interpret data received from another site.
- Two mechanisms for specifying XML schema
  - **Document Type Definition (DTD)**
  - **XML Schema**

Document Type Definition(DTD)

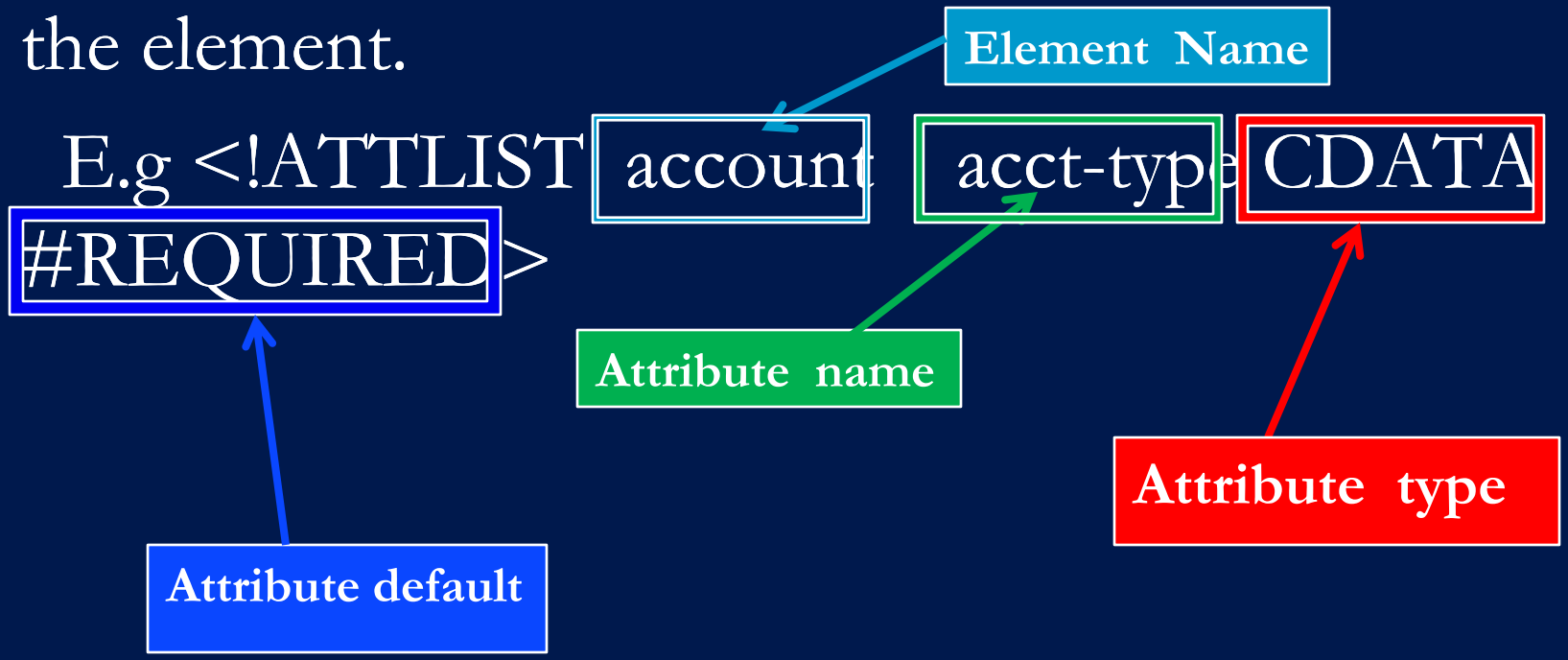
- DTD is a **set of rules** that allows us to specify our own set of elements , attributes and entities.
- It specifies which elements we can use and constraints on these elements…
- DTD does not constrain data types
  - All values represented as strings in XML
  - DTD enclosed in **<!DOCTYPE name[DTD declaration]>**
- DTD syntax
  - <!ELEMENT element (subelements-specification) >
  - <!ATTLIST element (attributes) >

Element Declaration in DTD

- Subelements can be specified as
  - names of elements, or
  - #PCDATA (parsed character data), i.e., character strings E.g.**<!ELEMENT LASTNAME(#PCDATA)>**
  - EMPTY (no subelements) or ANY (anything can be a subelement)
- Subelement specification may have regular expressions
  E.g. <!ELEMENT BOOKLIST(BOOK)*>
  - Notation:
    - "?" - 0 or one occurrences
    - "+" - 1 or more occurrences
    - "*" - 0 or more occurrences

Attribute declaration in DTD

- Attributes are declared with an ATTLIST declaration.
- Attributes of elements are declared outside of the element.
- E.g <!ATTLIST account acct-type CDATA #REQUIRED>



Example

- <!ELEMENT BOOKLIST(BOOK)*>
- <!ELEMENT BOOKLIST(BOOK)+>
- <!ELEMENT BOOK(AUTHOR,TITLE,PUBLISHED?)>
- <!ELEMENT AUTHOR(FIRSTNAME,LASTNAME)>
- <!ELEMENT FIRSTNAME (#PCDATA)>
- <!ELEMENT LASTNAME (#PCDATA)>
- <!ATTLIST BOOK genre(science|maths)#REQUIRED>
- <!ATTLIST BOOK

  format(paperback |hardcore)"Paperback">

XML Schema

- XML Schema is a more sophisticated schema language which addresses the drawbacks of DTDs. Supports
  - Typing of values
    - E.g. integer, string, etc
    - Also, constraints on min/max values
  - User-defined, complex types
  - Many more features, including
    - uniqueness and foreign key constraints, inheritance

Querying and Transforming XML Data

- Translation of information from one XML schema to another
- Standard XML querying/translation languages
  - XPath
    - Simple language consisting of path expressions
  - XSLT
    - Simple language designed for translation from XML to XML and XML to HTML
  - XQuery
    - An XML query language with a rich set of features

Xpath

- XPath is used to address (select) parts of documents using **path expressions.**
- A path expression is a sequence of steps separated by **"/"**
- E.g. /bank-2/customer/customer_name evaluated on the bank-2 data would return

<customer_name>Joe</customer_name>
<customer_name>Mary</customer_name>

- E.g. /bank-2/customer/ customer_name/text( )
  returns the same names, but without the enclosing tags

E.g. /bank-2/account[balance > 400]

E.g. /bank-2/account[balance >400]/@ID

Xpath Functions

The categories of function Xpath provides are:

- String--- finding the length of string ,
  1)string(obj) 2)string() 3)starts-with(str,str)
- Node-set --- Return information about node-sets.
- 1) last() 2) position()
- Boolean--- returns either true or false.
- Number--- enables you to add numbers , find nearest integer value and convert string to numbers.
- 1)number() 2)round(num) 3)ceiling(num)

XSLT

- XSLT is a general-purpose transformation language
  - Can translate XML to XML, and XML to HTML
- XSLT transformations are expressed using rules called **templates**
- Example of XSLT template with **match** and **select** part

E.g.

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

XQuery

- XQuery is a general purpose query language for XML data
- XQuery does not represent queries in XML.
- Organized into "FLWR" expression:

✓ **For** -gives a series of variables that range over the result of xpath expression.

✓ **Let** -allows complicated expressions to be assigned to variable names for simplicity of representation.

✓ **Where** -performs additional tests on the joined tuples from the for section

✓ **Return** -allows the construction of results in XML

- Simple FLWOR 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>
```

  - Items in the **return** clause are XML text unless enclosed in {}, in which case they are evaluated
- 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>
```

Storage of XML data

- XML data can be stored in
  - **Relational databases**
    - Data must be translated into relational form
    - String Representation
    - Tree Representation
    - Map to Relation
  - **Non-relational data stores**
    - **Flat files**
      - Natural for storing XML
    - **XML database**
      - Database built specifically for storing XML data, supporting DOM model and declarative querying

String Representation

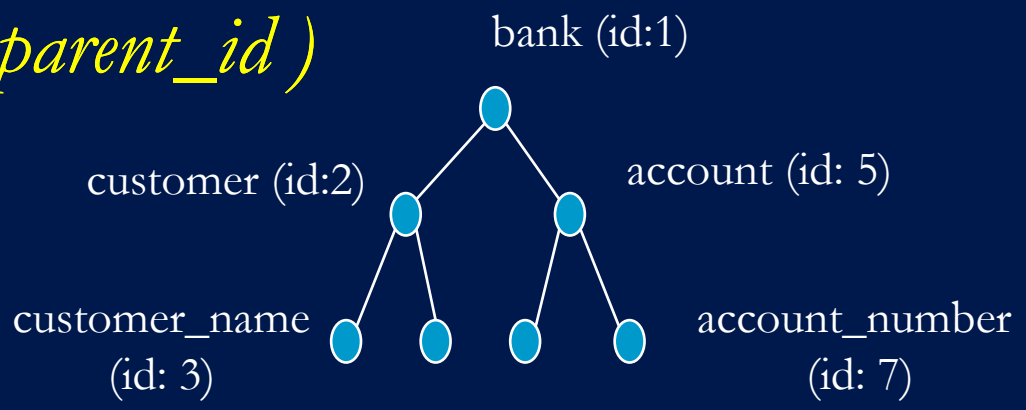
Store each top level element as a string field of a tuple in a relational database

- Use a single relation to store all elements, or
- Use a separate relation for each top-level element type
  - E.g. account, customer, depositor relations
    - Each with a string-valued attribute to store the element

- **Benefits**:
  - Can store any XML data even without DTD

Tree Representation

- **Tree representation:** **XML can be** modeled as tree and store using relations.
- *nodes(id, type, label, value)*
  child (child_id , parent_id)



- **Benefit:**

  Can store any XML data, even without DTD

- **Drawbacks:**
  - Data is broken up into too many pieces, increasing space overheads

Advantages of XML

- Domain specific vocabulary
- Platform independence
- Smart searches
- Granular updates
- User-selected view of data.

XML Applications

- **Two application of XML for communication**

Exchange of Data:

- **Business applications** such as banking and shipping
- **Scientific applications** such as chemistry and molecular biology.

Data Mediation:

- **Comparison shopping**

- **Web publishing**: XML allows you to create interactive pages, allows the customer to customize those pages, and makes creating e-commerce applications more intuitive.
- **Web searching and automating Web tasks**: XML defines the type of information contained in a document, making it easier to return useful results when searching the Web.
- **Pervasive computing**: XML provides portable and structured information types for display on pervasive (wireless) computing devices such as personal digital assistants (PDAs), cellular phones, and others.

2)XSL(EXtensible Stylesheet Language) :

CSS = HTML Style Sheets

- HTML uses predefined tags and the meaning of the tags are well understood.
- The <table> element in HTML defines a table - and a browser knows how to display it.
- Adding styles to HTML elements is simple. Telling a browser to display an element in a special font or color, is easy with CSS.

XSL = XML Style Sheets

- XML does not use predefined tags (we can use any tag-names we like), and the meaning of these tags are not well understood.

- A <table> element could mean an HTML table, a piece of furniture, or something else - and a browser does not know how to display it.

- XSL describes how the XML document should be displayed!

Object Model (DOM) :

- The DOM defines a standard for accessing documents like HTML and XML.
- The DOM defines the objects and properties of all document elements, and the methods (interface) to access them.
- Provides a set of function calls to manipulate XML and HTML documents using a programming lang such as Java.

THANK

YOU