CS315: Principles of Database Systems
XML

Arnab Bhattacharya
arnabb@cse.iitk.ac.in

Computer Science and Engineering,
Indian Institute of Technology, Kanpur
http://web.cse.iitk.ac.in/~cs315/

2nd semester, 2013-14

Tue, Fri 1530-1700 at CS101

Extensible Markup Language (XML)

- A language to markup, i.e., separate content from how it should be formatted or what it is about (metadata)
- Markups are specified by tags
  - Begin and end tags should appear in pairs

Extensible Markup Language (XML)

- A language to markup, i.e., separate content from how it should be formatted or what it is about (metadata)
- Markups are specified by tags
  - Begin and end tags should appear in pairs
- HTML markups only format
- XML (Extensible Markup Language) can markup data as well
- XML can use its *own* tags and there is no pre-defined list
- XML is a *document format*
- Originated from SGML (Standard Generalised Markup Language)

Example

```
<?xml version=``1.0'' encoding=``UTF-8''?>
<bookstore>
  <book category=``cooking''>
    <title lang=``en''>Everyday Mughlai</title>
    <author><fname>Mullah</fname><lname>Do Piaza</lname></author>
  </book>
  <book category=``web''>
    <title lang=``en''>XML</title>
    <author><fname>Web</fname><lname>Enthusiast</lname></author>
    <author><fname>Internet</fname><lname>Nerd</lname></author>
    <price>499</price>
  </book>
  <book category=``web'' cover=``paperback''>
    <title lang=``hi''>Computer Chalao</title>
    <author><fname>Sanganak</fname><lname>Vidyan</lname></author>
    <price>100</price>
  </book>
</bookstore>
```

Structure of XML

- XML can be considered as semi-structured data
- Sometimes also called self-describing data

Structure of XML

- XML can be considered as semi-structured data
- Sometimes also called self-describing data
- Elements are tags and must be properly nested
- Elements can have sub-elements

Structure of XML

- XML can be considered as semi-structured data
- Sometimes also called self-describing data
- Elements are tags and must be properly nested
- Elements can have sub-elements
- Attributes are descriptions of an element
- There can be only one value of an attribute
- Difference between an attribute and a sub-element can be arbitrary

Structure of XML

- XML can be considered as semi-structured data
- Sometimes also called self-describing data
- Elements are tags and must be properly nested
- Elements can have sub-elements
- Attributes are descriptions of an element
- There can be only one value of an attribute
- Difference between an attribute and a sub-element can be arbitrary
- Comments within <!- and ->

Structure of XML

- XML can be considered as semi-structured data
- Sometimes also called self-describing data
- Elements are tags and must be properly nested
- Elements can have sub-elements
- Attributes are descriptions of an element
- There can be only one value of an attribute
- Difference between an attribute and a sub-element can be arbitrary
- Comments within <!- and ->
- Sub-elements are unordered
  - XData encodes order

Structure of XML

- XML can be considered as semi-structured data
- Sometimes also called self-describing data
- Elements are tags and must be properly nested
- Elements can have sub-elements
- Attributes are descriptions of an element
- There can be only one value of an attribute
- Difference between an attribute and a sub-element can be arbitrary
- Comments within <!- and ->
- Sub-elements are unordered
  - XData encodes order
- A single *root* element

DTD

- XML can be described using document type declaration (DTD)
- Using DTD renders a specific *structure* to XML

```
<!DOCTYPE bookstore
[
  <!ELEMENT bookstore (book)*>
    <!ELEMENT book (title, author+, price?)>
      <!ELEMENT title (#PCDATA)>
        <!ATTLIST title lang (#PCDATA)>
      <!ELEMENT author (fname, lname)>
        <!ELEMENT fname (#PCDATA)>
        <!ELEMENT lname (#PCDATA)>
      <!ELEMENT price (#PCDATA)>
      <!ATTLIST book category (#PCDATA)>
      <!ATTLIST book cover (#PCDATA)>
]>
```

- * (any), + (> 0), ? (0 or 1) has regular expression meanings
- #PCDATA stands for parsed character data and are not interpreted
- DTD information should be included in the header

```
<!DOCTYPE bookstore SYSTEM ''bookstore.dtd''>
```

XML Schema

- XML Schema improves upon DTD
- Richer set of features
- XML file itself, and so, same parsing can be used
- Introduces namespace that separates XML documents from different places with the same element names
- XML Schema should be included in the header

```
<bookstore xmlns=''http://web.cse.iitk.ac.in/users/cs315/''
xmlns:xsi=''http://web.cse.iitk.ac.in/users/cs315/''
xsi:schemaLocation=''http://web.cse.iitk.ac.in/users/cs315/
    bookstore.xs''>
```

XML Schema example

```
<?xml version="1.0"?>
<xs:schema xmlns:xs=``http://web.cse.iitk.ac.in/users/cs315/''>
<xs:element name="bookstore">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="title" type="xs:string"/>
      <xs:element name="author">
      <xs:sequence>
        <xs:element name="fname" type="xs:string"/>
        <xs:element name="lname" type="xs:string"/>
       </xs:sequence>
  ...
      <xs:attribute name="category" type="xs:string"/>
  ...
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>
```

XPath

- Path expressions in XMl are specified using the XPath language
- /: immediate child
- //: descendant
- @: attribute value
- |: or
- *: any element

XPath

- Path expressions in XMl are specified using the XPath language
- /: immediate child
- //: descendant
- @: attribute value
- |: or
- *: any element
- `/bookstore//author/lname`

XPath

- Path expressions in XMl are specified using the XPath language
- /: immediate child
- //: descendant
- @: attribute value
- |: or
- *: any element
- `/bookstore//author/lname` selects all last names of all authors
- `/bookstore/book[price < 300]`

XPath

- Path expressions in XMl are specified using the XPath language
- /: immediate child
- //: descendant
- @: attribute value
- |: or
- *: any element
- `/bookstore//author/lname` selects all last names of all authors
- `/bookstore/book[price < 300]` selects all books with price less than 300
- `/bookstore/book[price < 300]/@cover`

XPath

- Path expressions in XMl are specified using the XPath language
- /: immediate child
- //: descendant
- @: attribute value
- |: or
- *: any element
- `/bookstore//author/lname` selects all last names of all authors
- `/bookstore/book[price < 300]` selects all books with price less than 300
- `/bookstore/book[price < 300]/@cover` selects covers of all books with price less than 300
- Can use aggregates such `count` and `sum`
- XPath is used by XQuery and XSLT (Extensible Style Sheet language for Transformations)

XQuery

- Query language for XML specified by W3C consortium
- Uses XPath
- FLWR or FLOWR ("flower") query expressions
- for: similar to `from` in SQL and also binds to variables
- let: similar to `from` in SQL and also assigns to variables
- where: similar to `where` in SQL
- return: similar to `select` in SQL

XQuery

- Query language for XML specified by W3C consortium
- Uses XPath
- FLWR or FLOWR ("flower") query expressions
- for: similar to `from` in SQL and also binds to variables
- let: similar to `from` in SQL and also assigns to variables
- where: similar to `where` in SQL
- return: similar to `select` in SQL
- order by: similar to `order by` in SQL
  - Default ordering is as in original XML document

XQuery

- Query language for XML specified by W3C consortium
- Uses XPath
- FLWR or FLOWR ("flower") query expressions
- for: similar to `from` in SQL and also binds to variables
- let: similar to `from` in SQL and also assigns to variables
- where: similar to `where` in SQL
- return: similar to `select` in SQL
- order by: similar to `order by` in SQL
  - Default ordering is as in original XML document
- Either for or let and return

XQuery

- Query language for XML specified by W3C consortium
- Uses XPath
- FLWR or FLOWR ("flower") query expressions
- for: similar to `from` in SQL and also binds to variables
- let: similar to `from` in SQL and also assigns to variables
- where: similar to `where` in SQL
- return: similar to `select` in SQL
- order by: similar to `order by` in SQL
  - Default ordering is as in original XML document
- Either for or let and return
- Queries can be nested

Difference between `for` and `let`

- for binds a variable to each element in the path expression
- let binds a variable to the entire collection of elements in the path expression

```
for $p in /bookstore//price
return <result> { $p } </result>
```

returns

```
<result><price>499</price></result>
<result><price>100</price></result>
```

while

```
let $p in /bookstore//price
return <result> { $p } </result>
```

returns

```
<result>
  <price>499</price>
  <price>100</price>
</result>
```

Tree pattern queries in XQuery

- book branches to price and author which has a child lname

```
for $b in //book, $p in $b/price, $a in $b/author, $l in $a/lname
return <result> {$l} </result>
```

Tree pattern queries in XQuery

- book branches to price and author which has a child lname

```
for $b in //book, $p in $b/price, $a in $b/author, $l in $a/lname
return <result> {$l} </result>
```

- Using nested queries

```
for $b in //book[price]
return
<result>
  for $a in $b/author
    return <name> {$a/lname} </name>
</result>
```

XML storage

- XML can be stored in multiple formats

XML storage

- XML can be stored in multiple formats
- Flat file

XML storage

- XML can be stored in multiple formats
- Flat file
- Object-oriented document object model (DOM)

XML storage

- XML can be stored in multiple formats
- Flat file
- Object-oriented document object model (DOM)
- Graph (i.e., tree)

XML storage

- XML can be stored in multiple formats
- Flat file
- Object-oriented document object model (DOM)
- Graph (i.e., tree)
- Relational

XML storage

- XML can be stored in multiple formats
- Flat file
- Object-oriented document object model (DOM)
- Graph (i.e., tree)
- Relational
- Native, as CLOB in relational

Graph databases

- Nodes as elements
- Edges encode relations

Graph databases

- Nodes as elements
- Edges encode relations
- Object exchange model (OEM)
- Two relations
- nodes(id,type,label,value)
  - Each element and attribute is given an *id*
  - *Type* denotes whether it is an element or an attribute
  - Name of the element or attribute is the *label*
  - *Value* contains the text value

Graph databases

- Nodes as elements
- Edges encode relations
- Object exchange model (OEM)
- Two relations
- nodes(id,type,label,value)
  - Each element and attribute is given an *id*
  - *Type* denotes whether it is an element or an attribute
  - Name of the element or attribute is the *label*
  - *Value* contains the text value
- child(child_id,parent_id) denotes the tree hierarchy

Graph databases

- Nodes as elements
- Edges encode relations
- Object exchange model (OEM)
- Two relations
- nodes(id,type,label,value)
  - Each element and attribute is given an *id*
  - *Type* denotes whether it is an element or an attribute
  - Name of the element or attribute is the *label*
  - *Value* contains the text value
- child(child_id,parent_id) denotes the tree hierarchy
- Uses basic edge scans to process queries
- Indexes using
  - vindex(l,o,pred): all children *o* with edge label *l* satisfying *pred*
  - lindex(o,l,p): all parents *p* of *o* with edge label *l*
  - bindex(x,l,y): all edges (x, y) with label *l*
  - pindex(path,o): all objects *o* reachable by path *path*

XML to relational

- Shredding: conversion of XML to relational
- Conversion of XQuery to XML
- Publishing: conversion of relational to XML

XML to relational

- Shredding: conversion of XML to relational
- Conversion of XQuery to XML
- Publishing: conversion of relational to XML
- Simple elements with at most value are columns
- Attributes are columns
- Complex elements are relations
- Identifiers may be added to encode foreign key relationships
- DTD can guide

XML to relational

- Shredding: conversion of XML to relational
- Conversion of XQuery to XML
- Publishing: conversion of relational to XML
- Simple elements with at most value are columns
- Attributes are columns
- Complex elements are relations
- Identifiers may be added to encode foreign key relationships
- DTD can guide
- Automatically adds ACID support

XML to relational

- Shredding: conversion of XML to relational
- Conversion of XQuery to XML
- Publishing: conversion of relational to XML
- Simple elements with at most value are columns
- Attributes are columns
- Complex elements are relations
- Identifiers may be added to encode foreign key relationships
- DTD can guide
- Automatically adds ACID support
- Recent SQL definition (called SQL/XML) allows extension to XML