XSLT

CPS 196.3
Introduction to Database Systems

XSLT

- W3C recommendation
- XML-to-XML rule-based transformation language
- An XSLT program is an XML document itself
- Used most frequently as a stylesheet language

![Diagram of XSLT process]

- XSLT program
  - Input XML
  - XSLT processor
  - Output XML

Actually, output does not need to be in XML in general

XSLT program

- An XSLT program is an XML document containing
  - Elements in the `<xsl:>` namespace
  - Elements in user namespace
- The result of evaluating an XSLT program on an
  input XML document =
  the XSLT document where each `<xsl:>` element
  has been replaced with the result of its evaluation
- Uses XPath as a sub-language

XSLT elements

- Element describing transformation rules
  - `<xsl:template>`
- Elements describing rule execution control
  - `<xsl:apply-templates>`
  - `<xsl:call-template>`
- Elements describing instructions
  - `<xsl:if>`, `<xsl:for-each>`, `<xsl:sort>`, etc.

XSLT example

- Find titles of books authored by "Abiteboul"

```xml
<?xml version="1.0"?>
<xsl:stylesheet
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    version="1.0">
  <xsl:template match="book[author='Abiteboul']">
    <booktitle>
      <xsl:value-of select="title"/>
    </booktitle>
  </xsl:template>
</xsl:stylesheet>
```

Not quite; we will see why later

<xsl:template>

- `<xsl:template match="match_expr">` is the basic XSLT
  construct describing a transformation rule
  - `match_expr` is an XPath-like expression specifying which
    nodes this rule applies to
- `<xsl:value-of select="xpath_expr"/>` converts the
  node-set returned by `xpath_expr` to a string
- `<booktitle>` and `<booktitle>` simply get copied to the
  output for each node match
Template in action

<xs:template match="book[author='Abiteboul']">
  <book>
    <booktitle>
      <xsl:value-of select="title"/>
    </booktitle>
  </book>
</xs:template>

Example XML fragment
<book ISBN="ISBN-10" price="80.00">
  <title>Foundations of Databases</title>
  <author>Abiteboul</author>
  <author>Hull</author>
  <author>Vianu</author>
  <publisher>Addison Wesley</publisher>
  <year>1995</year>
  <section>…</section>…
</book>
  <title>A First Course in Databases</title>
  <author>Ullman</author>
  <author>Widom</author>
  <publisher>Prentice-Hall</publisher>
  <year>2002</year>
  <section>…</section>…
</book>

Removing the extra output
- Add the following template:
  <xs:template match="text()|@*"/>
- This template matches all text and attributes
- XPath features
  - text() is a node test that matches any text node
  - @* matches any attribute
  - | means "or" in XPath
- Body of the rule is empty, so all text and attributes become empty string
  - This rule effectively filters out things not matched by the other rule

<xs:attribute>

- Again, find titles of books authored by "Abiteboul"; but make the output look like <book title="booktitle"/>
  - <xs:template match="book[author='Abiteboul']">
    <book>
      <booktitle>
        <xsl:value-of select="title"/>
      </booktitle>
    </book>
  </xs:template>

- A more general method
  - <xs:template match="book[author='Abiteboul']">
    <book>
      <attribute name="title">
        <xsl:value-of select="title"/>
      </attribute>
    </book>
  </xs:template>

  - <xs:attribute name="attr">
  adds an attributed named attr with value body to the parent element in the output

<xs:copy-of>

- Another slightly different example: return (entire) books authored by "Abiteboul"
  - <xml version="1.0">
  - <xs:stylesheet
    xmlns:xs="http://www.w3.org/1999/XSLT/Transform"
    version="1.0">
    <xs:template match="text()|@*"/>
    <xs:template match="book[author='Abiteboul']">
      <xsl:copy-of select="."/>
    </xs:template>
  </xs:stylesheet>

- <xs:copy-of select="xpath_expr"/> copies the entire contents (including tag structures) of the node-set returned by xpath_expr to the output

Formatting XML into HTML

- Example templates to
  - Render a book title in italics in HTML.
  - Render the authors as a comma-separated list
  - <xs:template match="book/title">
    <i><xsl:value-of select="."/></i>
  </xs:template>
  - <xs:template match="book/author[1]">
  </xs:template>
  - <xs:template match="book/author[position()>1]">
  
- (Continue on next slide)

- <xs:apply-templates>
  - Example: generate a table of contents
    - Display books in an HTML unordered list
    - For each book, first display its title, and then display its sections in an HTML ordered list
    - For each section, first display its title, and then display its subsections in an HTML ordered list
  - <xs:template match="title">
    <xs:value-of select="."/>
  </xs:template>
  - <xs:template match="section">
    <li>
      <xsl:apply-templates select="title"/>
      <ol><xsl:apply-templates select="section"/></ol>
    </li>
  </xs:template>

(Continue on next slide)
Example continued

```xml
<xsl:template match="book">
  <li>
    <xsl:apply-templates select="title"/>
    <ol>
      <xsl:apply-templates select="section"/>
    </ol>
  </li>
</xsl:template>
<xsl:template match="bibliography">
  <html>
    <head><title>Bibliography</title></head>
    <body>
      <ul>
        <xsl:apply-templates select="book"/>
      </ul>
    </body>
  </html>
</xsl:template>

 One problem remains
  ▪ Even if a book or a section has no sections, we will still generate an empty <ol></ol> element

Whitespace control

 Whitespace is everywhere in XML.
  ```xml
  <book ISBN="ISBN-10" price="80.00">
    <title>FoundationsofDatabases</title>
  </book>
  ```
  • “.ttt” goes into a text node
  • “tttt FOUNDATIONS of DATABASES.” goes into another text node
  Specify <xsl:strip-space elements="*"/> to remove text nodes (under any element) containing only whitespace
  To strip leading and trailing whitespace and replace any sequence of whitespace characters by a single space, specify <xsl:template match="text()">
    <xsl:value-of select="normalize-space()"/>
  </xsl:template>

XSLT summary

 Used often as a stylesheet language, but can be considered a query language too
  ▪ Very expressive, with full recursion
    • Cannot be replaced by XQuery
  ▪ Easily non-terminating, difficult to optimize
    • Cannot replace XQuery
   So many features, so little time!