XSLT

CPS 116

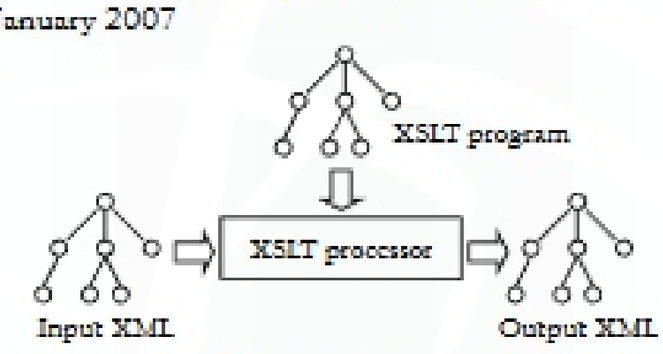
Introduction to Database Systems

Announcements (October 25)

- Homework #3 due in 1½ weeks
  - Start early!
- Project milestone #2 due in 2 weeks

XSLT

- XML-to-XML rule-based transformation language
  - Used most frequently as a stylesheet language
  - An XSLT program is an XML document itself
  - Current version is 2.0; W3C recommendation since January 2007



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
- Basic ideas
  - Templates specify how to transform matching input nodes
  - Structural recursion applies templates to input trees recursively
- 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.
- Elements generating output
  - `<xsl:value-of>`, `<xsl:attribute>`, `<xsl:copy-of>`, `<xsl:text>`, etc.

XSLT example

- Find titles of books authored by "Abiteboul"

```
<?xml version="1.0"?>                    Standard header of an XSLT document
<xsl:stylesheet
  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
  version="2.0">
<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="book[author='Abiteboul']">
  <booktitle>
    <xsl:value-of select="title"/>
  </booktitle>
</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*`"/>` evaluates *xpath_expr* within the context of the node matching the template, and converts the result sequence to a string
- `<booktitle>` and `</booktitle>` simply get copied to the output for each node match

Template in action

```
<xsl:template match="book[author='Abiteboul']">
  <booktitle>
    <xsl:value-of select="title"/>
  </booktitle>
</xsl: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>
<book ISBN="ISBN-20" price="55.00">
  <title>A First Course in Database</title>
  <author>Ullman</author>
  <author>Widom</author>
  <publisher>Prentice-Hall</publisher>
  <year>2002</year>
  <section>...</section>...
</book>
```

Template applies

```
<booktitle>
  Foundations of Databases
</booktitle>
```

Template does not apply; default behavior is to process the node recursively and print out all text nodes

```
A First Course in Database
Ullman
Widom
Prentice-Hall
2002
...
```

Removing the extra output

- Add the following template:

```
<xsl: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