Processing of XML Documents, XPath - XSLT - XQuery

Angelo Mario Del Grosso

CNR-ILC
http://ilc.cnr.it/
angelo.delgrosso@ilc.cnr.it

Selezione, Elaborazione e Presentazione di documenti XML-TEI mediante i linguaggi XPath, XSLT e XQuery

.

Istituto di Linguistica Computazionale "A. Zampolli",
15th November 2024

Outline

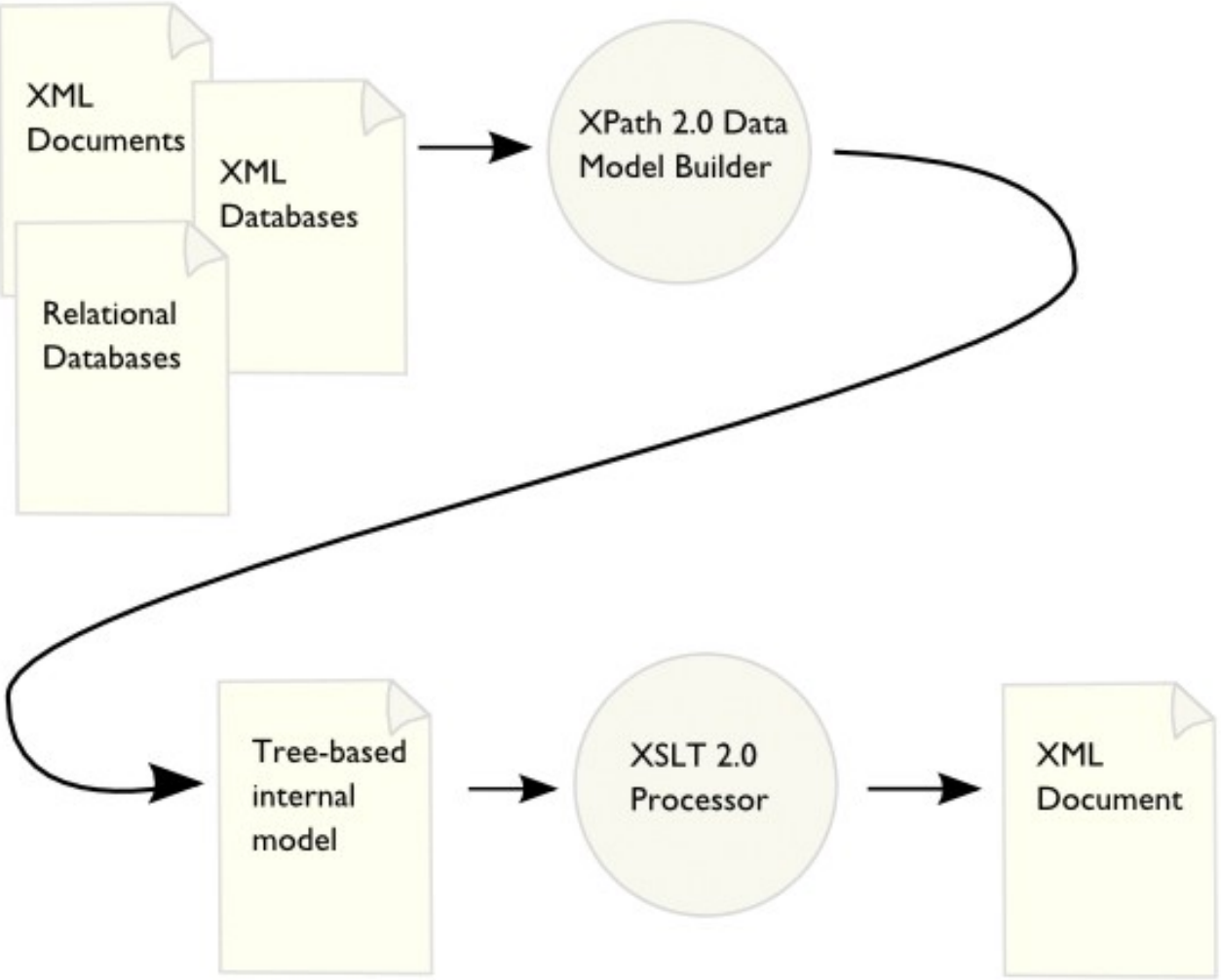
Progress status

Seminar aims

We talk about

Introduction to processing and visualizing XML documents via **XPATH** and **XSL** languages.

Seminar path



Progress status

Progress status

Progress status

Progress status

Progress status

XQuery

A family of languagies

XSL is a family of W3C recommendations for defining XML document transformation and presentation.

XSL

- **XSL Transformations (XSLT)**:
  a language for transforming XML
- **The XML Path Language (XPath)**:
  an expression language to refer to parts of an XML document;
- **XSL Formatting Objects (XSL-FO)**:
  an XML vocabulary for specifying formatting semantics.

XQuery

XQuery is a query language for XML to extract data.

Tree data model (XDM)

- The W3C specifications for XSLT, XQuery, and XPath **model an XML document as a tree**. This data model is known as **XDM**, and the nodes of an XDM tree are known as XDM nodes.
- XDM defines the information contained in the input to an XSLT processor as well as it defines all **permissible values of expressions** in the XSLT
- The **node-sets of XPath 1.0** are replaced in XPath 2.0 by **sequences of nodes**.

Tree data model (XDM)

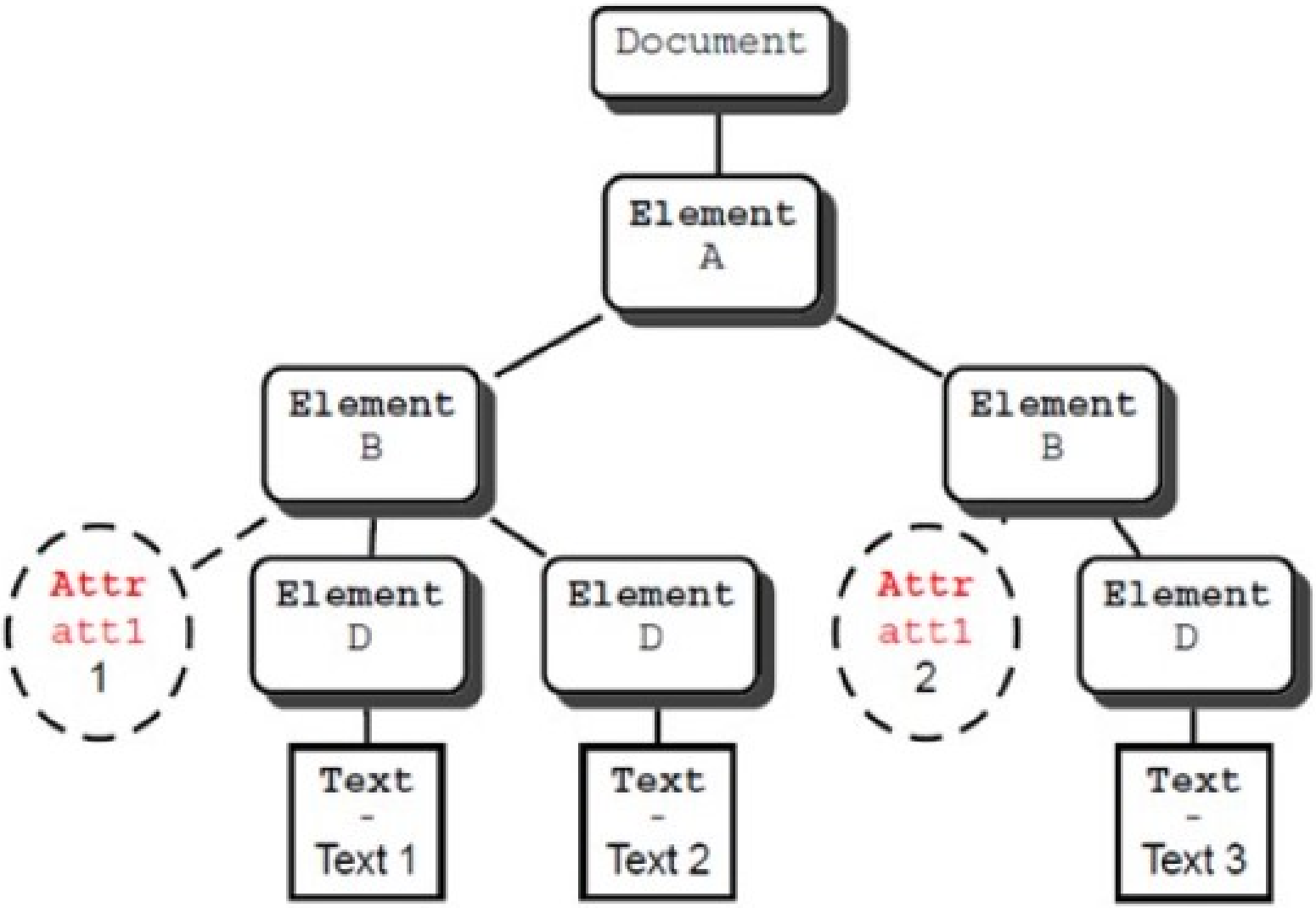
- The W3C specifications for XSLT, XQuery, and XPath **model an XML document as a tree**. This data model is known as **XDM**, and the nodes of an XDM tree are known as XDM nodes.
- XDM defines the information contained in the input to an XSLT processor as well as it defines all **permissible values of expressions** in the XSLT
- The **node-sets of XPath 1.0** are replaced in XPath 2.0 by **sequences of nodes**.

Tree data model (XDM)

- The W3C specifications for XSLT, XQuery, and XPath **model an XML document as a tree**. This data model is known as **XDM**, and the nodes of an XDM tree are known as XDM nodes.
- XDM defines the information contained in the input to an XSLT processor as well as it defines all **permissible values of expressions** in the XSLT
- The **node-sets of XPath 1.0** are replaced in XPath 2.0 by **sequences of nodes**.

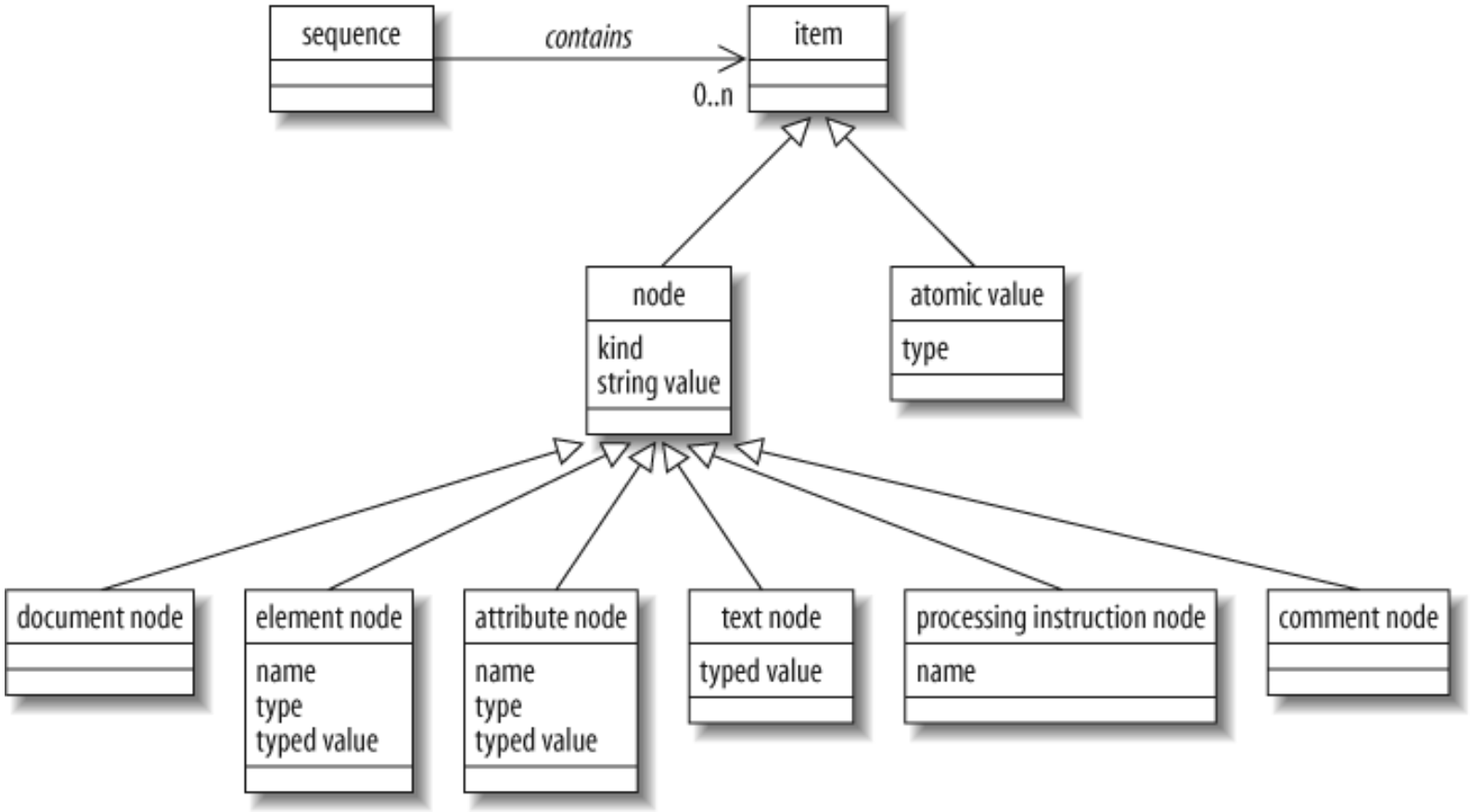
XML Trees

Hierarchical Ordered Nodes

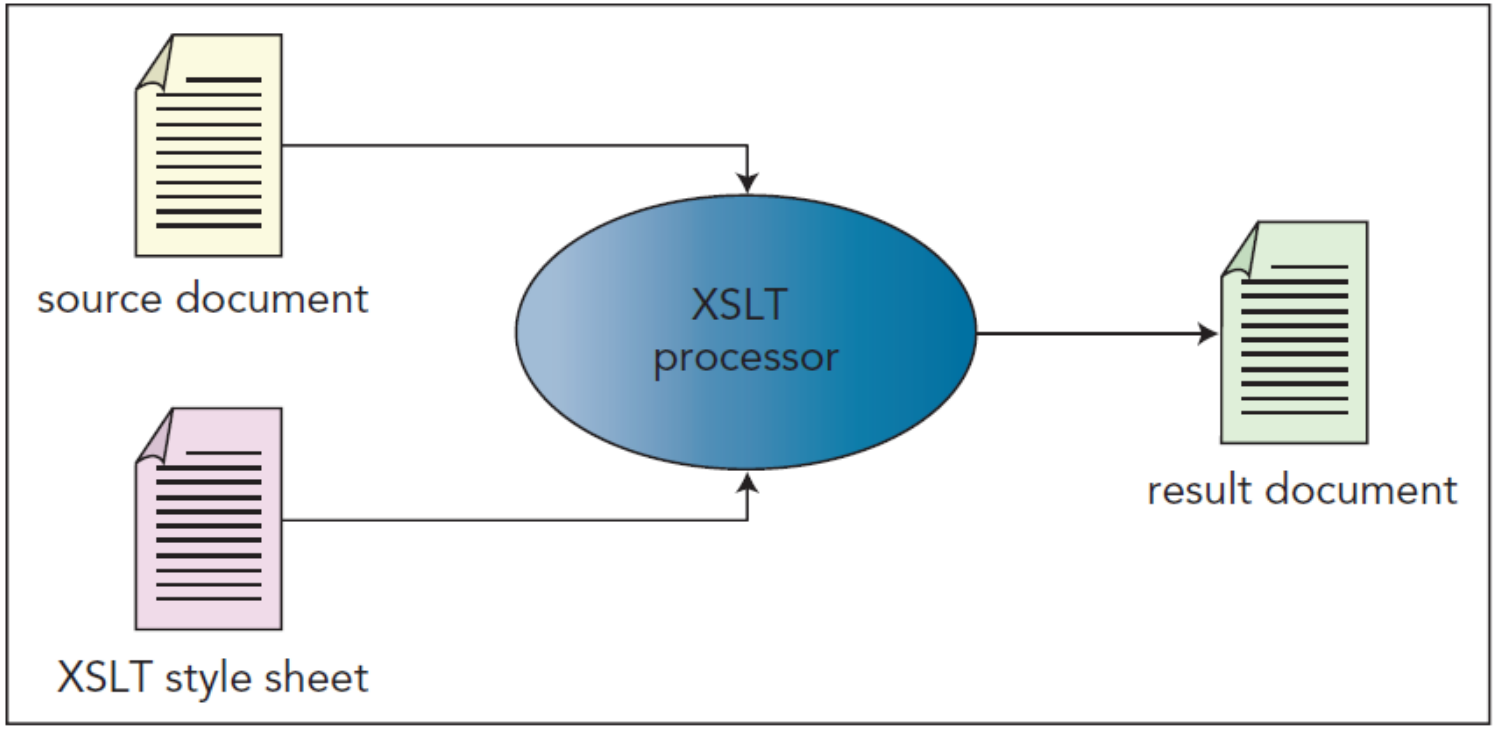


XPath Data Model

XDM



Fondamenti Extensible Stylesheet Language



XQuery

Intro

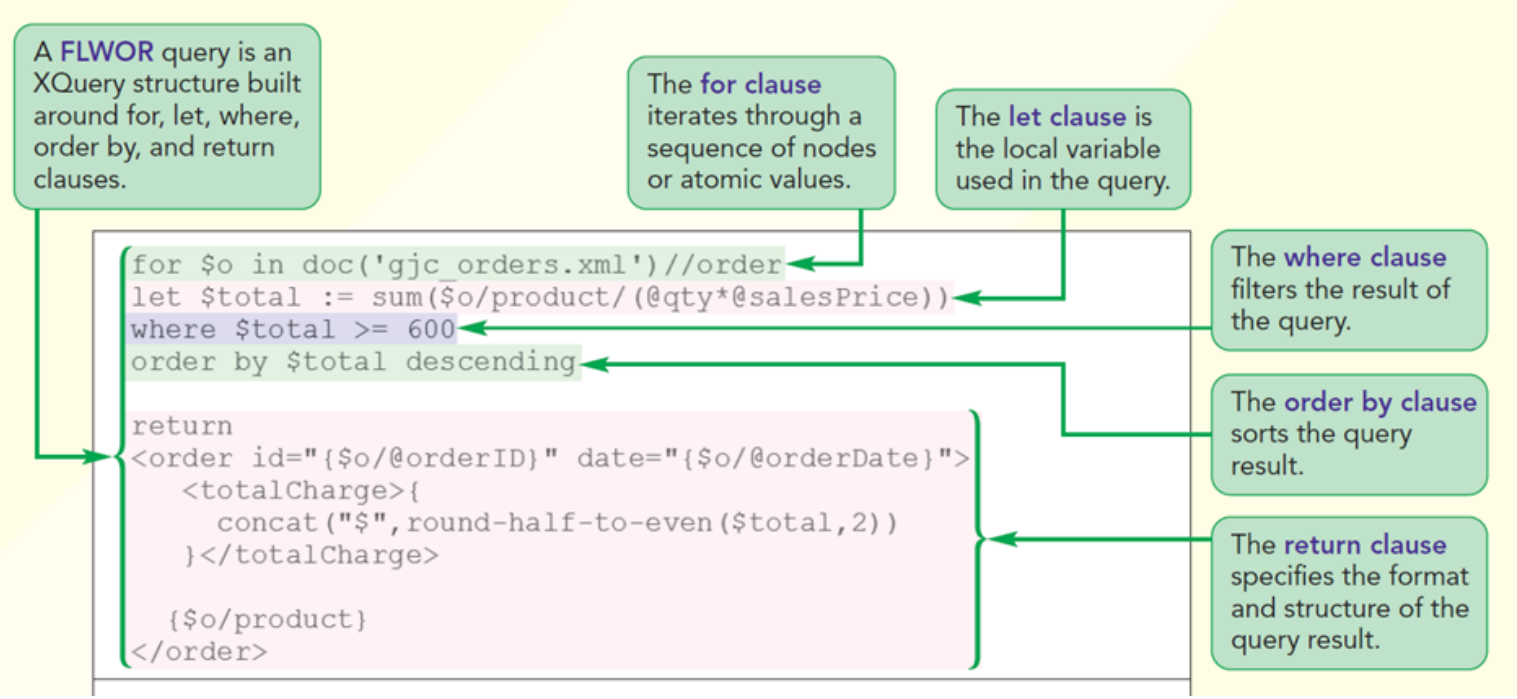
XQuery is a powerful **query language** designed to query and manipulate XML data

Compared to XSLT, XQuery is extremely **compact** and this aspect enhances the **readability** of the scripts

FLWOR Statement

XQuery is based on the **FLWOR** (*For*, *Let*, *Where*, *Order-by*, *Return*) construct. It is very compact and also students in traditional humanities can manage it

FLWOR Statement



FOR CLAUSE

- **Purpose**: Iterates over a sequence of items
- **Functionality**: The *For clause* binds a variable to each item in a sequence, allowing for iteration over large datasets, much like a loop in other programming languages. It can **iterate** over *nodes*, *values*, or *any sequence* generated by an XQuery expression

```
for $word in doc("aen.xml")//tei:w
    return $word
```

LET CLAUSE

Purpose: Binds values to variables

Functionality: The *Let clause* is used to assign a value to a variable. Unlike the For clause, Let does not iterate but simply assigns a value to be used later in the FLWOR statement. It is useful for storing intermediate results

```
let $count = count(doc("aen.xml")//tei:w)
```

WHERE CLAUSE

Purpose: Filters data based on conditions

Functionality: The *Where clause* restricts the results returned by the FLWOR expression, acting as a filter that allows only items meeting certain conditions to be processed in the subsequent clauses

```
where $word/@xml:lang = "grc"
```

ORDER BY CLAUSE

- **Purpose**: Sorts the results of the query
- **Functionality**: The *Order by clause* sorts the items returned by the query, based on *specified criteria*. This sorting can be done in ascending or descending order

```
order by $word/text() ascending
```

RETURN CLAUSE

Purpose: Specifies what is to be returned from the query

Functionality: The *Return clause* determines what will be included in the final output of the FLWOR statement. This can be a node, a constructed element, a value, or any combination of these

```
for $line in doc("aen.xml")//l
let $line_id := $line/@xml:id
let $text := string-join($line/w, ' ')
return
  <span class="line">
    <span class="id">{$line_id}</span>
    <span class="text">{$text}</span>
  </span>
```

FLWOR Nesting Statements

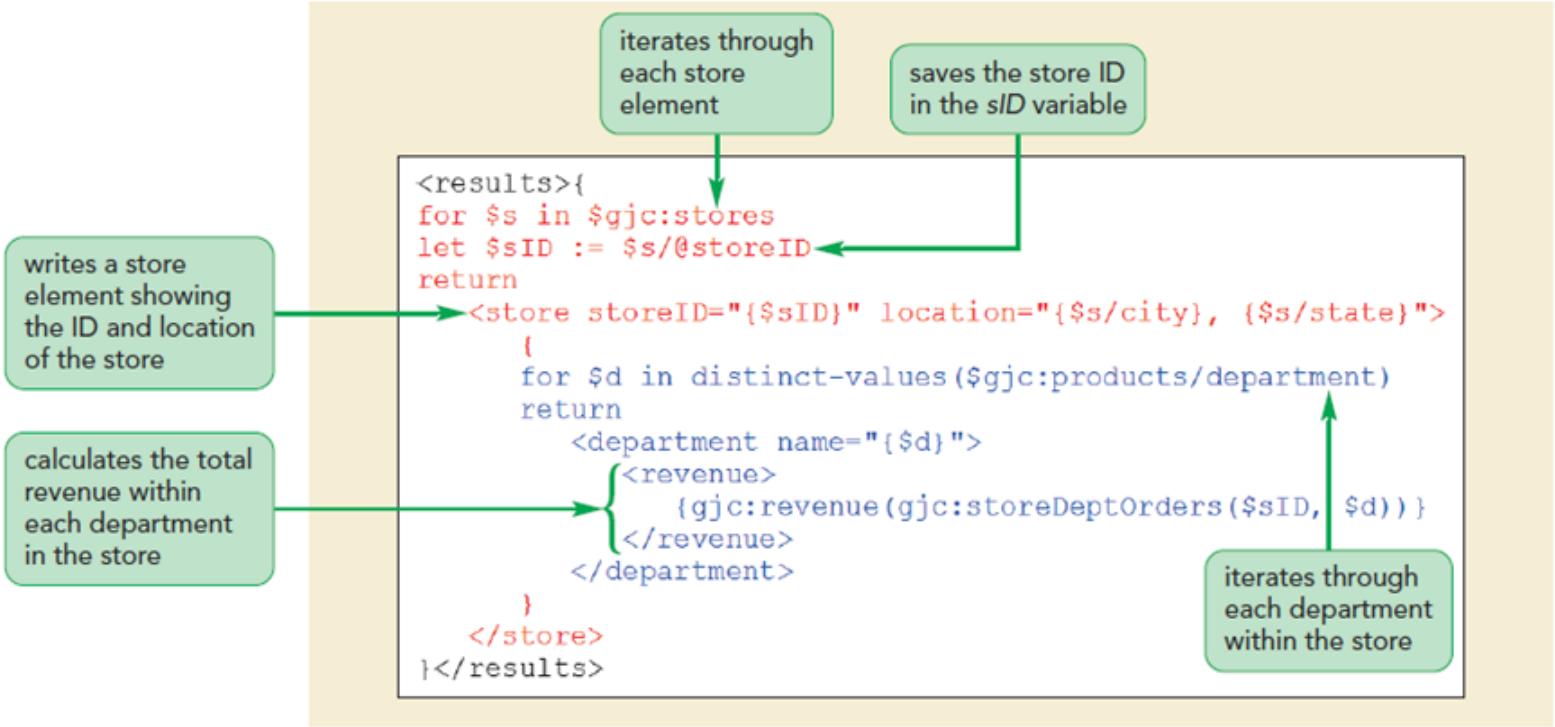


Immagine dal libro: New Perspectives on XML, Patrick Carey, Sasha Vodnik, 2014

XSLT Transformation in XQuery

XQuery can *perform XSLT* transformations, which are more familiar to the **XML-TEI community** than the xquery language

```
<!--transform the TEI lg in an XHTML div-->
{transform:transform($lg,doc("trans/fromTeiToHtml.xsl"),())}
```

```
<!-- process tei:lg -->
<xsl:template match="tei:lg">
    <xsl:element name="div">
        <!--<xsl:apply-templates select="tei:head"/>-->
        <xsl:for-each select="tei:l">
            <xsl:element name="span">
                <xsl:attribute name="class">latcit</xsl:attribute>
                <xsl:attribute name="id">
                    <xsl:value-of select="@n"/>
                </xsl:attribute>
                    <xsl:value-of select="@n"/>
                <xsl:text> </xsl:text>
            </xsl:element>
            <xsl:element name="span">
                <xsl:attribute name="class">verseText</xsl:attribute>
                <xsl:for-each select="tei:w">
                    <xsl:value-of select="text()"/>
                    <xsl:text> </xsl:text>
                </xsl:for-each>
            </xsl:element>
            <br/>
        </xsl:for-each>
    </xsl:element>
</xsl:template>
```

Progress status

XQuery in Action

Working with XQuery


```
<teiHeader>···
</teiHeader>
<text>
    <body>
        <div1 xml:id="d001" type="section" decls="#md" met="H">···
        </div1>
        <div1 xml:id="d002" type="section" decls="#md" met="H">···
        </div1>
        <div1 xml:id="d003" type="section" decls="#md" met="H">···
        </div1>
        <div1 xml:id="d004" type="section" decls="#md" met="H">···
        </div1>
        <div1 xml:id="d005" type="section" decls="#md" met="H">
            <head>
                <title>5</title>
                <abbr>5</abbr>
            </head>
            <p xml:id="d005l1" n="0|1">
                <w xml:id="d005w600">MENALCAS</w>
            </p>
            <l xml:id="d005l2" n="1">
                <w xml:id="d005w1">Cur</w>
                <w xml:id="d005w2">non,</w>
                <w xml:id="d005w3">Mopse,</w>
                <w xml:id="d005w4">boni</w>
```

XQuery in Action

Working with XQuery


```
<l xml:id="d001l2" n="1">
  <s>
    <w lemma="Tityre," pos="UPosTag=ADV">Tityre,</w>
    <w lemma="tu" pos="UPosTag=PRON|Case=Nom|Gender=Masc|Number=Sing">tu</w>
    <w lemma="patula" pos="UPosTag=NOUN|Case=Gen|Gender=Fem|Number=Sing">patulae</w>
    <w lemma="recubo" pos="UPosTag=VERB|Case=Nom|Gender=Masc|Number=Sing|Tense=Pres|VerbForm=Part">recubans</w>
    <w lemma="sub" pos="UPosTag=ADP">sub</w>
    <w lemma="tegmen" pos="UPosTag=NOUN|Case=Abl|Gender=Neut|Number=Sing">tegmine</w>
    <w lemma="fago" pos="UPosTag=VERB|Aspect=Perf|Mood=Ind|Number=Sing|Person=1|Tense=Past|VerbForm=Fin|Voice=Act">fagi</w>
  </s>
</l>
<l xml:id="d001l3" n="2">
  <s>
    <w lemma="siluester" pos="UPosTag=NOUN|Case=Acc|Gender=Fem|Number=Sing">siluestrem</w>
    <w lemma="teneo" pos="UPosTag=VERB|Aspect=Perf|Mood=Ind|Number=Sing|Person=1|Tense=Past|VerbForm=Fin|Voice=Act">tenui</w>
    <w lemma="musa" pos="UPosTag=NOUN|Case=Acc|Gender=Fem|Number=Sing">musam</w>
    <w lemma="meditaris" pos="UPosTag=ADJ|Case=Gen|Gender=Fem|Number=Sing">meditaris</w>
    <w lemma="auena:" pos="UPosTag=PUNCT">auena:</w>
  </s>
</l>
<l xml:id="d001l4" n="3">
  <s>
    <w lemma="nos" pos="UPosTag=PRON|Case=Nom|Gender=Masc|Number=Plur">nos</w>
    <w lemma="patria" pos="UPosTag=NOUN|Case=Gen|Gender=Fem|Number=Sing">patriae</w>
    <w lemma="finis" pos="UPosTag=NOUN|Case=Gen|Gender=Masc|Number=Sing">finis</w>
    <w lemma="et" pos="UPosTag=CCONJ">et</w>
    <w lemma="dulcis" pos="UPosTag=NOUN|Case=Abl|Gender=Fem|Number=Sing">dulcia</w>
    <w lemma="linquio" pos="UPosTag=VERB|Mood=Ind|Number=Plur|Person=1|Tense=Pres|VerbForm=Fin|Voice=Act">linquimus</w>
    <w lemma="aruauintus" pos="UPosTag=NOUN|Case=Abl|Gender=Masc|Number=Sing">arua.</w>
  </s>
</l>
```

MQDQ - Eclogae di Virgilio

XQuery in Action

Working with XQuery

esempio.xq

```
xquery version "3.1";
(: Esempio Workshop FeDHLab 2024 :)

declare namespace output = "http://www.w3.org/2010/xslt-xquery-serialization";
declare option output:method "text";
declare option output:encoding "UTF-8";
declare option output:media-type "text/plain";

let $c := 'num DIVs: ' || count(//div)
return $c

```

live-xq.txt

```
num DIVs: 7
```

TERMINALE PROBLEMI 7 OUTPUT PORTE CONSOLE DI DEBUG powershell

```
2024\tools> java -cp .\SaxonHE11-6J\* net.sf.saxon.Query -s:..\xml\IRT030.xml -q:..\xql\esem
pio.xq -o:..\out\live-xq.txt
PS C:\Users\angel\risorse\files\seminars\lezioni-seminari-fedhlab\fedhlab-xpath-xslt-xquery-
2024\tools>
```

```
java -cp .\SaxonHE11-6J\* net.sf.saxon.Query
```

Example one

```
4  declare namespace output = "http://www.w3.org/2010/xslt-xquery-serialization";
5  declare option output:method   "html";
6  declare option output:encoding "UTF-8";
7  declare option output:media-type "text/html";
8
9
10 let $doc := doc('./xml/switchboard.clarin.eu-step9.xml')
11 let $words := $doc//w[lower-case(@lemma)="musa"]
12 for $w in $words
13 return <p><b>{string($w/@lemma)} </b><i style="padding-left:1em"> {string($w)}</i></p>
14
```

Example two

```
4  declare namespace output = "http://www.w3.org/2010/xslt-xquery-serialization";
5  declare option output:method   "xml";
6  declare option output:encoding "UTF-8";
7  declare option output:media-type "text/xml";
8
9  let $doc := doc('./xml/switchboard.clarin.eu-step9.xml')
10 let $lemmata := distinct-values ($doc//w/@lemma)
11
12 return
13     <lemmata n="{count($lemmata)}">{
14             for $l in $lemmata
15             return
16              <lemma value="{$l}">{
17                 for $form in distinct-values($doc//w[@lemma=$l])
18                 return
19                     <form value="{data($form)}" />
20
21             }</lemma>
22     }
23     </lemmata>
```

Example three

```
4  declare namespace output = "http://www.w3.org/2010/xslt-xquery-serialization";
5  declare option output:method   "html";
6  declare option output:encoding "UTF-8";
7  declare option output:media-type "text/html";
8
9  let $doc := doc('./xml/switchboard.clarin.eu-step9.xml')
10 let $words := $doc//w
11 for $w in $words
12 group by $l := $w/@lemma
13 order by count($w) descending
14 return <ul>{
15     for $p in distinct-values($w)
16     return <li><b>{$p}</b><i style="padding-left:1em">{string($w[1]/@lemma)}</i> ({count($w)})</li>
17 }</ul>
```

Example four

```
4  declare namespace output = "http://www.w3.org/2010/xslt-xquery-serialization";
5  declare option output:method   "text";
6  declare option output:encoding "UTF-8";
7  declare option output:media-type "text/plain";
8
9  let $doc := doc('./xml/switchboard.clarin.eu-step9.xml')
10 for $word in $doc//w
11 let $lemma := $word/@lemma
12 where lower-case($lemma) = "miser"
13 let $eclo := data($word/ancestor::div1/descendant::head/descendant::title)
14 let $verso := data($word/ancestor::l/@n)
15 return concat(string($word), " in ", normalize-space($eclo), " ecloga ", $verso, " verso", "&#xa;")
```

Example five

```
4  declare namespace output = "http://www.w3.org/2010/xslt-xquery-serialization";
5  declare option output:method   "html";
6  declare option output:encoding "UTF-8";
7  declare option output:media-type "text/html";
8
9  let $doc := doc('./xml/switchboard.clarin.eu-step9.xml')
10 let $words := $doc//w[ft:query(@lemma,"miser~0.5")]
11 for $w at $idx in $words
12 return <li style="list-style-type:none">
13         <span style="font-weight:bold">{$idx}</span>
14         <span style="padding-left:1em">{string($w/@lemma)}</span>
15         <span style="font-style:italic; padding-left:1em">({string($w)})</span>
16       </li>
```

Example existdb and Lucene

```
4  declare namespace output = "http://www.w3.org/2010/xslt-xquery-serialization";
5  declare option output:method   "html";
6  declare option output:encoding "UTF-8";
7  declare option output:media-type "text/html";
8
9  let $doc := doc('./xml/switchboard.clarin.eu-step9.xml')
10 let $words := $doc//w[ft:query(@lemma,"miser~0.5")]
11 for $w at $idx in $words
12 return <li style="list-style-type:none">
13         <span style="font-weight:bold">{$idx}</span>
14         <span style="padding-left:1em">{string($w/@lemma)}</span>
15         <span style="font-style:italic; padding-left:1em">({string($w)})</span>
16       </li>
```

/db/apps/clarin-siena-2024/esempio5.xq

Adaptive Output ☑ Indent ☐ Live Preview ☑ Highlight Index Matches

```
1 <li style="list-style-type:none">
    <span style="font-weight:bold">1</span>
    <span style="padding-left:1em">miror</span>
    <span style="font-style:italic; padding-left:1em">(miror)</span>
  </li>
2 <li style="list-style-type:none">
    <span style="font-weight:bold">2</span>
    <span style="padding-left:1em">mis</span>
    <span style="font-style:italic; padding-left:1em">(mea)</span>
  </li>
```

Example XSLT transformation with XQuery

```
4   declare option output:encoding
    "UTF-8";
5   declare option output:media-type
    "text/xml";
6
7   let $result := transform(map
    {"stylesheet-node":doc('../xsl/
    add-license.xsl'),"source-node":doc
    ('../xml/DeVulgariEloquentia.xml'),
    "stylesheet-params": map {
8       QName('', 'user'): "ANGELO"
9   }})
10
11  return $result?output
```

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!--De vulgari eloquentia-->
3   <TEI xmlns="http://www.tei-c.org/ns/1.0">
4       <teiHeader>
5           <fileDesc>
6               <titleStmt>…
13              </titleStmt>
14              <editionStmt>
15                  <edition
                    n="CLARIN">CLARIN
                    EDITION</edition>
16                  <respStmt>
17                      <resp>Adapted by</
                        resp>
18                      <name>ANGELO</name>
19                  </respStmt>
```

TERMINALE PROBLEMI 5 OUTPUT PORTE CONSOLE DI DEBUG

```
xslt-xquery-2024\tools> java -cp .\SaxonHE11-6J\* net.sf.saxon.Query  -q:..\xql\
transform.xq -o:..\out\live-xq.xml
PS C:\Users\angel\risorse\files\seminars\lezioni-seminari-fedhlab\fedhlab-xpath-
xslt-xquery-2024\tools>
```

powe...
wsl
powe...

Progress status

Bibliography

deepen into XPath and XSLT

Some References

- XQuery and XPath Data Model 3.1
  https://www.w3.org/TR/xpath-datamodel-31/
- XSLT Recommendations
  https://www.w3.org/TR/xslt/
- XPath Recommendations
  https://www.w3.org/TR/xpath/
- Kay, M. (2011). XSLT 2.0 and XPath 2.0 Programmer's Reference. Wiley.
- Williams, I. (2009). Beginning XSLT and XPath: Transforming XML Documents and Data. Wiley.
- Walmsley, P. (2015) XQuery: Search Across a Variety of XML Data. O'Reilly.
- Saxonica documentation:
  https://www.saxonica.com/documentation11/documentation.xml