Esercitazione XQuery

Paolo Papotti

papotti@dia.uniroma3.it

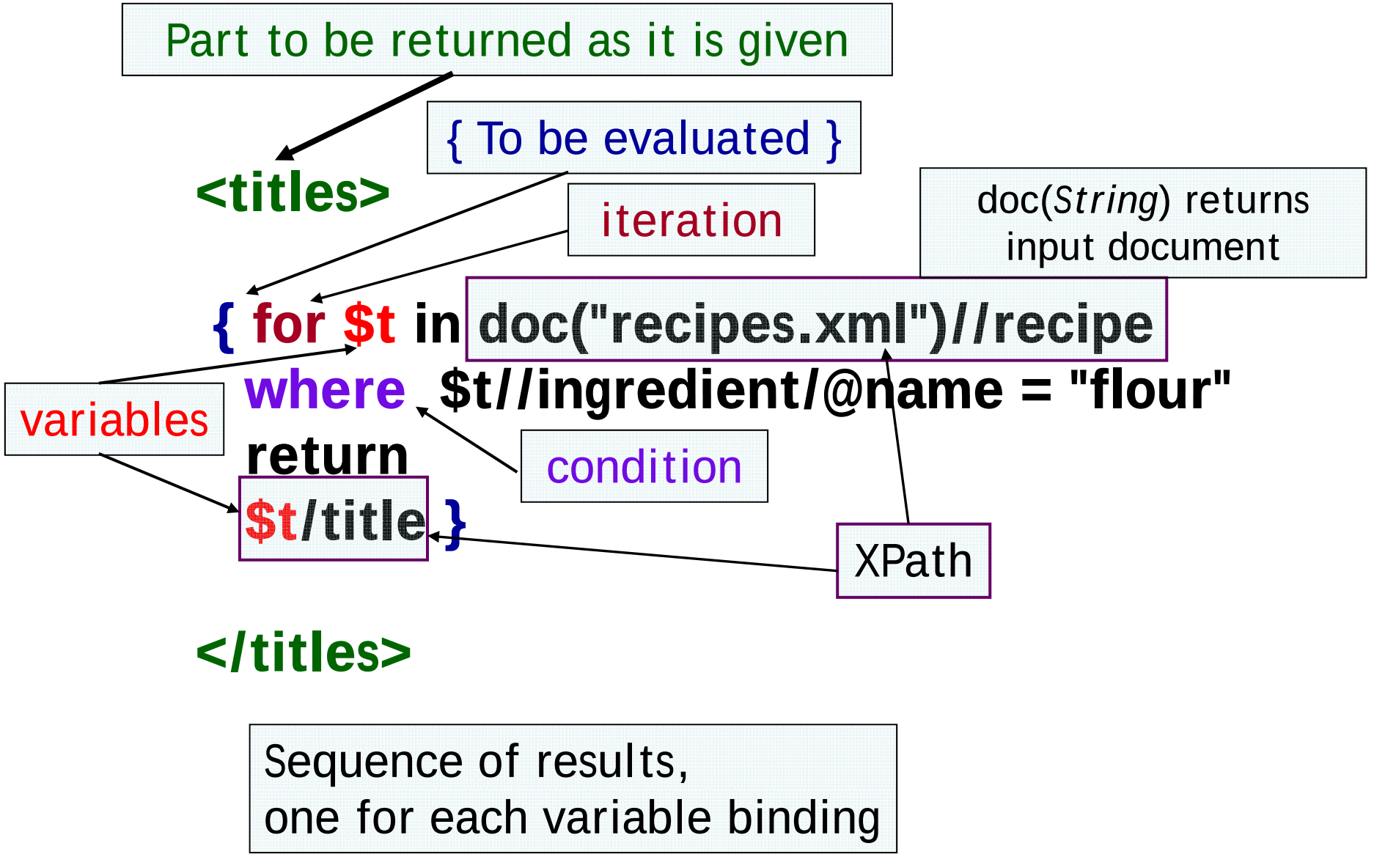
http://www.dia.uniroma3.it/~papotti/

Recipes

```
<!ELEMENT recipes    (recipe*)>
<!ELEMENT recipe     (title, ingredient+, preparation, comment?, nutrition)>
<!ELEMENT title      (#PCDATA)>
<!ELEMENT ingredient (ingredient*,preparation?)>
<!ATTLIST ingredient
       name   CDATA #REQUIRED
       amount CDATA #IMPLIED
       unit   CDATA #IMPLIED>
<!ELEMENT preparation (step+)>
<!ELEMENT step        (#PCDATA)>
<!ELEMENT nutrition   EMPTY>
<!ELEMENT comment (#PCDATA)>
<!ATTLIST nutrition
    calories CDATA #REQUIRED
      fat           CDATA #REQUIRED
      carbohydrates  CDATA #REQUIRED
      protein       CDATA #REQUIRED
      alcohol       CDATA #IMPLIED>
```

```
<recipes>
  <recipe>
    <title>Beef Parmesan</title>
    <ingredient name="beef" amount="1.5"
                unit="pound"/>
    <ingredient name="onion" amount="1"/>
    <ingredient name="green rings"
                amount="1"/>
    <preparation>
      <step>Boil pasta</step>
    </preparation>
    <comment>... </comment>
    <nutrition calories="1167" fat="23"
                carbohydrates="45" protein="32"/>
  </recipe>
...
```

Query Features



XPath and XQuery solutions - recipes

1. "The titles of all recipes, returned as strings."
   - //title/text()
   - for $t in doc("recipes.xml")//title/text() return $t

2. "The titles of all recipes that use flour."
   1. //recipe[ingredient/@name="flour"]/title **NO!**
   2. //recipe[.//ingredient/@name="flour"]/title
   3. //recipe[descendant::ingredient/@name="flour"]/title

```
<floury>{
   for $r in doc("recipes.xml")//recipe,
   $i in $r//ingredient
   where $i[@name="flour"]
   return <dish>{$r/title/text()}</dish>
}</floury>
```

```
<floury>
<dish>Ricotta Pie</dish>
<dish>Zuppa Inglese</dish>
<dish>Cailles en Sarcophages</dish>
<dish>Cailles en Sarcophages</dish>
<\floury>
```

```
<floury>{
      for $r in doc("recipes.xml")
      //recipe[.//ingredient[@name="flour"]]
      return <dish>{$r/title/text()}</dish>
}</floury>
```

```
<floury>
<dish>Ricotta Pie</dish>
<dish>Zuppa Inglese</dish>
<dish>Cailles en Sarcophages</dish>
<\floury>
```

XPath and XQuery solutions - recipes

Genera le <my-recipes> con i titoli come attributi e i relativi <my-ingredients> con i nomi come elementi testo

```
<my-recipes>
  { for $r in doc("recipes.xml")//recipe
    return
    <my-recipe title="{$r/title}">
      { for $i in $r//ingredient
          return
          <my-ingredient>{ string($i/@name) }</my-ingredient>
      }
    </my-recipe>
  }
</my-recipes>
```

```
<my-recipes>
  <my-recipe title="Beef Parmesan with Garlic Angel Hair Pasta">
    <my-ingredient>beef cube steak</my-ingredient>
    <my-ingredient>onion, sliced into thin rings</my-ingredient>
    <my-ingredient>green bell pepper, sliced in rings</my-ingredient>
    <my-ingredient>italian seasoned bread crumbs</my-ingredient>
    <my-ingredient>grated Parmesan cheese</my-ingredient>
    <my-ingredient>olive oil</my-ingredient>
    .....
</my-recipes>
```

XQuery solutions - recipes

Per ogni ingrediente, la lista delle ricette dove viene usato

```
let $d := doc("recipes.xml")//recipe
for $i in distinct-values($d//ingredient/@name)
return <ingredient name="{$i}">
       { for $r in $d
            where $r//ingredient[@name=$i]
            return $r/title
       }
</ingredient>
```

Remove duplicates

Join condition

```
<ingredient name="vanilla extract">
            <title>Ricotta Pie</title></ingredient>
<ingredient name="semisweet chocolate chips">
            <title>Ricotta Pie</title></ingredient>
<ingredient name="dough">
            <title>Ricotta Pie</title></ingredient>
<ingredient name="flour">
            <title>Ricotta Pie</title>
            <title>Zuppa Inglese</title>
            <title>Cailles en Sarcophages</title>
</ingredient> ...
```

Beware: XPath Attributes

- **doc("recipes.xml")//recipe[1]/ingredient[1]/@name**
  → attribute name {"beef cube steak"} (run-time error)

  a constructor for an attribute node

- **string(doc("recipes.xml")//recipe[1]/ingredient[1]/@name)**
  → "beef cube steak"

  a value of type string

XPath Attributes (cntd.)

- **<first-ingredient>**
  { string(doc("recipes.xml")//recipe[1]
  /ingredient[1]/@name) }
  </first-ingredient>

→ <first-ingredient>beef cube steak</first-ingredient>

an element with string content

XPath Attributes (cntd.)

- **<first-ingredient>**
  { doc("recipes.xml")//recipe[1]
  /ingredient[1]/@name }
  </first-ingredient>

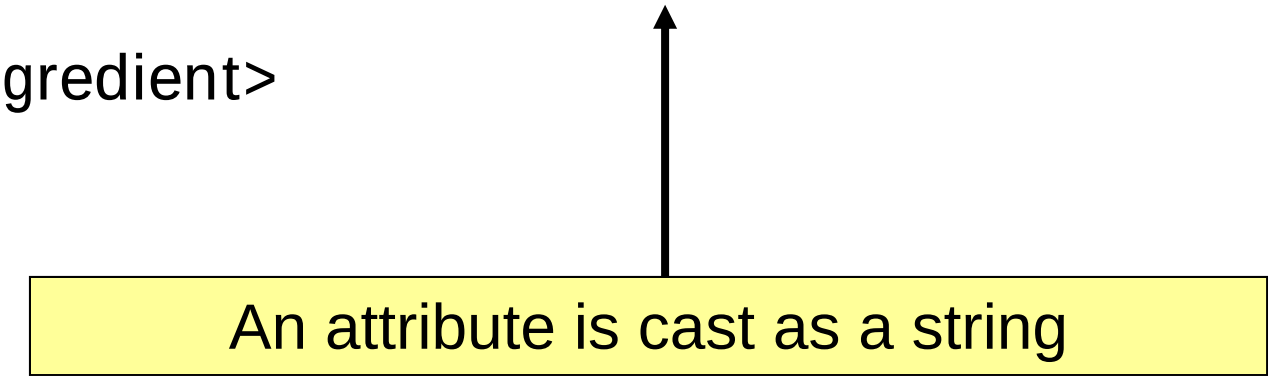
  → <first-ingredient name="beef cube steak"/>

an element with an attribute

XPath Attributes (cntd.)

- **<first-ingredient**
  oldName="{doc("recipes.xml")//recipe[1]
  /ingredient[1]/@name}">
  Beef
  </first-ingredient>

  → <first-ingredient oldName="beef cube steak">
  Beef
  </first-ingredient>



```
<bib>
   <book year="1994">
      <title>TCP/IP Illustrated</title>
      <author><last>Stevens</last><first>W.</first></author>
      <publisher>Addison-Wesley</publisher>
      <price> 65.95</price>
   </book>
   <book year="1992">
      <title>Advanced Programming in the Unix environment</title>
      <author><last>Stevens</last><first>W.</first></author>
      <publisher>Addison-Wesley</publisher>
      <price>65.95</price>
   </book>
   <book year="2000">
      <title>Data on the Web</title>
      <author><last>Abiteboul</last><first>Serge</first></author>
      <author><last>Buneman</last><first>Peter</first></author>
      <author><last>Suciu</last><first>Dan</first></author>
      <publisher>Morgan Kaufmann Publishers</publisher>
      <price>39.95</price>
   </book>
   <book year="1999">
      <title>The Economics of Technology and Content for Digital TV</title>
      <editor>
            <last>Gerbarg</last><first>Darcy</first>
             <affiliation>CITI</affiliation>
      </editor>
         <publisher>Kluwer Academic Publishers</publisher>
      <price>129.95</price>
   </book>
</bib>
```

bib.xml

1. List books published by Addison-Wesley after 1991, including their year and title
2. For each book, create a flat list of all the title-author pairs, with each pair enclosed in a "result" element.
3. For each book in the bibliography, list the title and authors, grouped inside a "result" element
4. For each author in the bibliography, list the author's full name and the titles of all books by that author, grouped in a "result" element.
5. For each book that has at least one author, list the title and first two authors, and an empty "et-al" element if the book has additional authors.

1. List books published by Addison-Wesley after 1991, including their year and title.

```
<bib>
 {
  for $b in doc("bib.xml")/bib/book
  where $b/publisher = "Addison-Wesley" and $b/@year > 1991
  return
    <book year="{ $b/@year }">
      { $b/title }
    </book>
 }
</bib>
```

2. For each book, create a flat list of all the title-author pairs, with each pair enclosed in a "result" element.

```
<results>
  {
    for $b in doc("bib.xml")/bib/book,
        $t in $b/title,
        $a in $b/author
    return
        <result>
            { $t }
            { $a }
        </result>
  }
</results>
```

3. For each book in the bibliography, list the title and authors, grouped inside a "result" element

```
<results>
  {
    for $b in doc("bib.xml")/bib/book
    let $t:=$b/title
    let $a:=$b/author
    return
        <result>
            { $t }
            { $a }
        </result>
  }
</results>
```

3. For each book in the bibliography, list the title and authors, grouped inside a "result" element

```
<results>
{
    for $b in doc("bib.xml")/bib/book
    return
        <result>
            { $b/title }
            { $b/author  }
        </result>
}
</results>
```

4. For each author in the bibliography, list the author's name and the titles of all books by that author, grouped in a "result" element.

```
<results>  {
  let $a := doc("bib.xml")//author
  for $last in distinct-values($a/last),
        $first in distinct-values($a[last=$last]/first)
  order by $last, $first
  return
   <result>
     <author> <last>{ $last }</last> <first>{ $first }</first></author>
    {  for $b in doc("bib.xml")/bib/book/author
       where $b/last = $last and $b/first=$first
       return $b/../title
     } </result>  }
</results>
```

5. For each book that has at least one author, list the title and first two authors, and an empty "et-al" element if the book has additional authors.

```
<bib>  {
    for $b in doc("bib.xml")//book
    where count($b/author) > 0
    return
        <book>
            { $b/title }
            { for $a in $b/author[position()<=2]
                return $a  }
            {   if (count($b/author) > 2)
                then <et-al/>
                else ()  }
        </book>
  } </bib>
```

6. List the titles and years of all books published by Addison-Wesley after 1991, in alphabetic order
7. Find books in which the name of some element ends with the string "or" and the same element contains the string "Suciu" somewhere in its content. For each such book, return the title and the qualifying element. *Hint: use string(), local-name(), contains(str1, str2) and ends-with(str1, str2) string functions*
8. For each book with an author, return the book with its title and authors. For each book with an editor, return a reference with the book title and the editor's affiliation.
9. Find pairs of books that have different titles but the same set of authors (possibly in a different order). *Hint: use the deep-equal($1, $2) function and <<.*

6. List the titles and years of all books published by Addison-Wesley after 1991, in alphabetic order

```
<bib>
 {
   for $b in doc("bib.xml")//book
   where $b/publisher = "Addison-Wesley" and
                 $b/@year > 1991
   order by $b/title
   return
      <book>
         { $b/@year }
         { $b/title }
      </book>
 }
</bib>
```

7. Find books in which the name of some element ends with the string "or" and the same element contains the string "Suciu" somewhere in its content. For each such book, return the title and the qualifying element.

```
for $b in doc("bib.xml")//book
let $e := $b/*[contains(string(.), "Suciu")
               and ends-with(local-name(.), "or")]
where exists($e)
return
    <book>
        { $b/title }
        { $e }
    </book>
```

converts a value or a node to a string

For an element or attribute, this is simply its name, stripped of any prefix it might have.

8. For each book with an author, return the book with its title and authors. For each book with an editor, return a reference with the book title and the editor's affiliation.

```
<bib>
  { for $b in doc("bib.xml")//book[author]
      return  <book>
                  { $b/title }
                  { $b/author }
              </book>
  }
  { for $b in doc("bib.xml")//book[editor]
      return  <reference>
                  { $b/title }
                  {$b/editor/affiliation}
              </reference>
  }
</bib>
```

9. Find pairs of books that have different titles but the same set of authors (possibly in a different order).

Hint: use the deep-equal($1, $2) function and <<

The deep-equal function returns true if the $s1 and $s2 sequences contain the same values, in the same order.

$a << $b returns true if $a precedes $b in document order

9. Find pairs of books that have different titles but the same set of authors (possibly in a different order).

```
<bib>
 { for $book1 in doc("bib.xml")//book,
       $book2 in doc("bib.xml")//book
   let $aut1 := for $a in $book1/author
                order by $a/last, $a/first
                return $a
   let $aut2 := for $a in $book2/author
                order by $a/last, $a/first
                return $a
   where $book1 << $book2 and $book1/title!=$book2/title
            and deep-equal($aut1, $aut2)
   return <book-pair>
             { $book1/title }
             { $book2/title }
          </book-pair>
 } </bib>
```

```
<book>
  <title>Data on the Web</title>
  <author>Serge Abiteboul</author>
  <author>Peter Buneman</author>
  <author>Dan Suciu</author>
  <section id="intro" difficulty="easy" >
    <title>Introduction</title>
    <p>Text ... </p>
    <section>
      <title>Audience</title> <p>Text ... </p>
    </section>
    <section>
      <title>Web Data and the Two Cultures</title> <p>Text ... </p>
      <figure height="400" width="400">
        <title>Traditional client/server architecture</title>
        <image source="csarch.gif"/>
      </figure>
      <p>Text ... </p>
    </section>
(...)
</section>
</book>
```

book.xml

10. Prepare a (nested) table of contents for Book1, listing all sections with titles. Preserve the original attributes of each <section> element, if any. *Make use of a local function.*
11. Prepare a (flat) figure list for Book1, listing all the figures and their titles. Preserve the original attributes of each <figure> element, if any.
12. How many sections are in Book1, and how many figures?
13. How many top-level sections are in Book1?
14. Make a flat list of the section elements in Book1. In place of its original attributes, each section element should have two attributes, containing the title of the section and the number of figures immediately contained in the section.
15. Make a nested list of section elements in Book1, preserving their original attributes and hierarchy. Inside each section element, include the title of the section and an element that includes the number of figures immediately contained in the section.

10. Prepare a (nested) table of contents for Book1, listing all sections with titles. Preserve the original attributes of each <section> element, if any.

```
declare function local:toc($book-or-section as element()) as
element()*
{
    for $section in $book-or-section/section
    return
      <section>
        { $section/@* ,
          $section/title ,
          local:toc($section) }
      </section>  };

<toc>
   {
     for $s in doc("book.xml")/book return local:toc($s)
   }
</toc>
```

11. Prepare a (flat) figure list for Book1, listing all the figures and their titles. Preserve the original attributes of each <figure> element, if any.

```
<figlist>
 {
   for $f in doc("book.xml")//figure
   return
       <figure>
          { $f/@* }
          { $f/title }
       </figure>
 }
</figlist>
```

12. How many sections are in Book1, and how many figures?

```
<section_count>
 {
   count(doc("book.xml")//section)
 }
</section_count>,

<figure_count>
 {
    count(doc("book.xml")//figure)
 }
</figure_count>
```

13. How many top-level sections are in Book1?

```
<top_section_count>
 {
   count(doc("book.xml")/book/section)
 }
</top_section_count>
```

14. Make a flat list of the section elements in Book1. In place of its original attributes, each section element should have two attributes, containing the title of the section and the number of figures immediately contained in the section.

```
<section_list>
  {
    for $s in doc("book.xml")//section
    let $f := $s/figure
    return
        <section title="{$s/title/text()}" figcount="{count($f)}"/>
  }
</section_list>
```

15. Make a nested list of section elements in Book1, preserving their original attributes and hierarchy. Inside each section element, include the title of the section and an element that includes the number of figures immediately contained in the section.

```
declare function local:summary($sec as element())  as
element()*
{for $section in $sec
  return <section>
              { $section/@* }
              { $section/title }
              <figcount>
                   { count($section/figure) }
              </figcount>
              { for $s1 in $section/section
                        return local:summary($s1)}
          </section> };
<toc>
  {for $s in doc("book.xml")/book/section
    return local:summary($s)  } </toc>
```