P.PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Funções e Transformações XQuery

Bruno Oliveira (bmo@estg.ipp.pt)

PEI Processamento Estruturado de Informação

Agenda

- Element constructors
- Built-in functions
- Expressões quantificadas
- User functions
- Introdução a XQuery

Manipulação e Transformação de Dados

- Ao executar uma expressão XQuery, os elementos são devolvidos exatamente como estão no documento original.
- No entanto, é possível adicionar novos elementos e atributos conforme necessário.
- Além disso, o XQuery permite incorporar múltiplas consultas numa única transformação, proporcionando maior flexibilidade na manipulação e estruturação de dados XML.

Manipulação e Transformação de Dados

Exemplo XML original:

```xml
<?xml version="1.0" encoding="UTF-8"?>
<livros>
    <livro>
        <titulo>XQuery</titulo>
        <autor>João Silva</autor>
        <preco>29.99</preco>
    </livro>
    <livro>
        <titulo>XML</titulo>
        <autor>Ana Sousa</autor>
        <preco>39.99</preco>
    </livro>
</livros>
```

Exemplo de XQuery com transformação:

```
let $livros := doc("livros.xml")/livros/livro
return
  <livrosTransformados>
    {
      for $livro in $livros
      return
        <livro novo="sim">
          { $livro/titulo }
          { $livro/autor }
          <precoComDesconto>{
            $livro/preco * 0.9 }
          </precoComDesconto>
        </livro>
    }
  </livrosTransformados>
```

Manipulação e Transformação de Dados

Resultado:

```
<?xml version="1.0" encoding="UTF-8"?>
<livrosTransformados>
    <livro novo="sim">
        <titulo>XQuery</titulo>
        <autor>João Silva</autor>
        <precoComDesconto>26.991</precoComDesconto>
    </livro>
    <livro novo="sim">
        <titulo>XML</titulo>
        <autor>Ana Souzs</autor>
        <precoComDesconto>35.991</precoComDesconto>
    </livro>
</livrosTransformados>
```

Manipulação e Transformação de Dados

Exemplo 2:

```
<html>
     <body>
         <h1>Bookstore</h1>
            <ul>
                {
                    for $x in doc("bookstore.xml")/bookstore/book
                    order by $x/title
                    return <li   class="{data($x/@category)}">{data($x/title)}</li>
                }
              </ul>
       </body>
</html>
```

Resultado Exemplo 2:

```
<!DOCTYPE HTML>
<html>
   <body>
      <h1>Bookstore</h1>
      <ul>
         <li class="COOKING">30 Dias para Mudar de Vida, Detox Paleo</li>
         <li class="COOKING">Everyday Italian</li>
         <li class="CHILDREN">Harry Potter</li>
         <li class="CHILDREN">I Wish You More</li>
         <li class="WEB">Learning XML</li>
         <li class="WEB">MongoDB: The Definitive Guide</li>
         <li class="ROMANCE">XML In A Nutshell</li>
         <li class="WEB">XQuery Kick Start</li>
      </ul>
   </body>
</html>
```

XQuery Nodes

- Com element constructors, podemos não só criar elementos mas também atributos, text nodes e instruções de processamento utilizando a mesma sintaxe XML;
- As chavetas: {} permitem criar um document node vazio;
- Podemos combinar os vários construtores para criar um documento completo, incluindo construtores para a inclusão de XSD ou até comentários.

XQuery Nodes

Exemplo 1:

```
let $livros := doc("livros.xml")/livros/livro
return
  element livrosTransformados {
    for $livro in $livros
    return
      element livro {
        attribute novo { "sim" },
        element titulo { $livro/titulo/text() },
        element autor { $livro/autor/text() },
        element precoComDesconto { $livro/preco * 0.9 }
      }
  }
```

XQuery Nodes

Exemplo 1:

```
let $livros := doc("livros.xml")/livros/livro
return
document{
      element livrosTransformados {
        for $livro in $livros
        return
          element livro {
            attribute novo { "sim" },
            element titulo { $livro/titulo/text() },
            element autor { $livro/autor/text() },
            element precoComDesconto { $livro/preco * 0.9
}
          }
      }
}
```

Padrão válido de XML: Ao criar um documento XML garante que o resultado é um documento XML completo e bem formado.

O element constructor element é usado para criar explicitamente cada elemento XML dentro da consulta.

attribute é usado para criar o atributo novo="sim" em cada elemento <livro>.

element titulo { $livro/titulo/text() } extrai o texto do título do XML original.

O cálculo do preço com desconto é realizado diretamente no constructor do elemento <precoComDesconto>.

XQuery Nodes

- Os computed elements e attribute construtors representam uma sintaxe alternativa que pode ser utilizada para o mesmo objetivo dos construtores “XML-style”;
- Computed elements são uma funcionalidade do XQuery que permite criar dinamicamente elementos XML cujos nomes ou conteúdos são calculados ou determinados em tempo de execução.

XQuery Nodes

- A sintaxe básica de um computed element em XQuery é: element { expression } { content }

```
let $tipo := "livro"
let $nomeElemento := if ($tipo = "livro") then "livro" else
”revista"
return
  element { $nomeElemento } {
    element titulo { "XQuery" },
    element autor { "João Silva" }
  }
```

XQuery Nodes

- Exemplo como namespaces e referenciação de schemas:

```
document{
    element book{
        namespace xsi {'http://www.w3.org/2001/XMLSchema-instance'},
        attribute xsi:noNamespaceSchemaLocation {'bookRules.xsd'},
        attribute year { 1977 },
        element author {
                element first { "Crockett" },
                element last { "Johnson" }
                },
                element publisher {"HarperCollins Juvenile Books"},
                element price { 14.95 }
    }
}
```

XQuery Nodes

- Resultado

```
<?xml version="1.0" encoding="UTF-8"?>
<book xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="bookRules.xsd"
      year="1977">
   <author>
      <first>Crockett</first>
      <last>Johnson</last>
   </author>
   <publisher>HarperCollins Juvenile Books</publisher>
   <price>14.95</price>
</book
```

Expressões quantificadas

- Este tipo de expressões são úteis quando temos uma lista de itens numa sequência e queremos testar se algum (any) ou todos (all) os items correspondem a uma determinada condição.
- Exemplo de uma consulta que retorna os livros do autor: J K. Rowling:

```
for $b in doc("bookstore.xml")//book
where some $a in $b//author satisfies $a="J K. Rowling"
return $b
```

Expressões quantificadas

- Sintaxe Some: `some $var in sequence satisfies condition`
- Sintaxe Every: `every $var in sequence satisfies condition`

Expressões quantificadas

- Exemplo some:

```
let $numeros := (1, 2, 3, 4, 5)
return some $n in $numeros satisfies $n > 4
```

- A sequência de números é (1, 2, 3, 4, 5).
- A expressão verifica se algum número da sequência é maior que 4.
- O resultado é true, pois o número 5 satisfaz a condição.

Expressões quantificadas

- Exemplo every:

```
let $numeros := (1, 2, 3, 4, 5)
return every $n in $numeros satisfies $n > 0
```

- A sequência de números é (1, 2, 3, 4, 5).
- A expressão verifica se todos os números da sequência são maiores que 0.
- O resultado é true, porque todos os números na sequência são maiores que 0.

XQuery/Quantified Expressions

- Exemplo com “some”
  - Verificar se existe pelo menos um livro da “bookstore” pertencente à categoria “WEB”:

```
some $category in doc("bookstore.xml")//book/@category
satisfies ($category = "WEB")
```

true

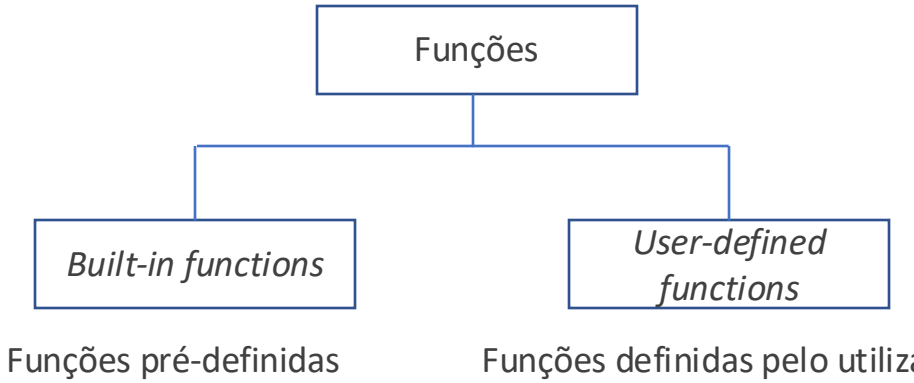
- Exemplo com “every”:
  - Verificar se todos os livros da “bookstore” pertencem à categoria “WEB”:

```
every $category in doc("bookstore.xml")//book/@category
satisfies ($category = "WEB")
```

false

Funções XQuery

- A linguagem XQuery permite a existência de dois tipos de funções:



- A XQuery utiliza as mesmas funções pré-definidas que o XPath. Estas funções podem ser usadas, por exemplo, para manipular strings ou para realizar cálculos matemáticos.
- Além destas funções pré-definidas, é também possível construírem-se funções personalizadas.

Funções XQuery

- A XQuery dispõe de um conjunto significativo de funções pré-definidas que podem ser consultas em http://www.xqueryfunctions.com/.
- Normalmente estas funções pertencem ao namespace fn.
- Existem também outras funções pré-definidas, que foram adicionadas em versões posteriores, e que necessitam da colocação do prefixo a que pertencem (prefixos math, map e array).
- Exemplos:
  - É equivalente escrever-se `count(doc("bookstore.xml")//book)` OU `fn:count(doc("bookstore.xml")//book)`
  - Para escrever e10, utiliza-se `math:exp(10)`

Funções XQuery

- Exemplo da função de cálculo de média:

```
let $b := doc("bookstore.xml")//book
let $avg := avg($b//price)
return $b[price > $avg]
```

- Verificar se uma sequência tem pelo menos um item (exists):

```
for $b in doc("bookstore.xml")//book
where exists($b/author)
return $b
```

Nota: A consulta XQuery não retorna um documento XML válido

Funções definidas pelo utilizador

- Por vezes, pode ser útil criar funções específicas que não fazem parte da lista de funções pré-definidas.
- Criar funções específicas permite tornar as consultas mais simples e possibilita a sua reutilização.
- Normalmente, as funções criam-se quando as expressões se tornam complexas e/ou repetitivas.

Funções definidas pelo utilizador

- Por exemplo, a seguinte consulta devolve uma lista de livros para um dado autor:

```
let $l:= "J K. Rowling"
for $b in doc("bookstore.xml")//book
where some $ba in $b//author satisfies ($ba=$l)
order by $b/title
return $b/title
```

Funções definidas pelo utilizador

A palavra local identifica a função como sendo local, ou seja, não pertence à biblioteca de funções do XQuery

Parâmetro (input) da função

```
declare function local:books-by-author($name) as element()*
{
   for $b in doc("bookstore.xml")//book
   where some $ba in $b//author satisfies ($ba=$name)
   order by $b/title
   return $b/title
};
let $n := local:books-by-author("J K. Rowling")
return $n
```

Retorno da função: um série de elementos. Por exemplo, se pretendermos retornar um elemento podemos colocar: *element()*

Invocação da função

Funções BaseX – Tipos de Dados

Tipo	Descrição
xs:date	refers to the built-in atomic schema type named xs:date
attribute()?	refers to an optional attribute node
element()	refers to any element node
element(po:shipto, po:address)	refers to an element node that has the name po:shipto and has the type annotation po:address (or a schema type derived from po:address)
element(*, po:address)	refers to an element node of any name that has the type annotation po:address (or a type derived from po:address)
element(customer)	refers to an element node named customer with any type annotation
node()*	refers to a sequence of zero or more nodes of any kind
item()+	refers to a sequence of one or more nodes or atomic values

BaseX

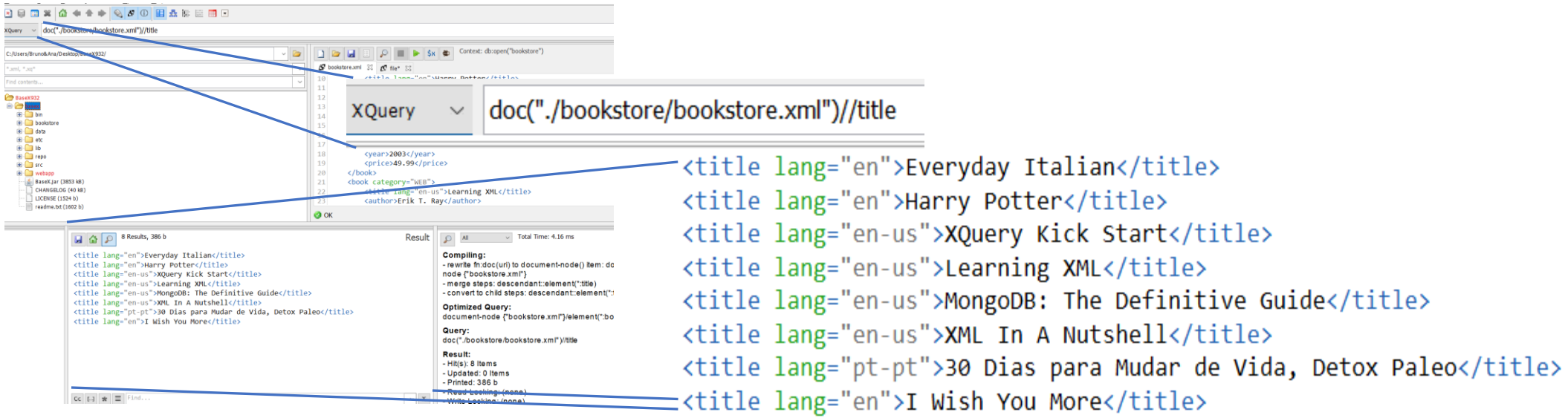
- O BaseX é um Editor e gestor de bases de dados XML;
- No BaseX, a base de dados de documentos é um conceito bastante “leve”;
- Suporta tecnologias XML e ainda várias versões do XQuery (incluindo a 4 que ainda se encontra em “draft”)

BaseX

- Existem vários pacotes de instalação, com diferentes componentes e aplicação;
- https://basex.org/download/ -> Core Package

BaseX

- É possível abrir um documento XML e executar consultas Xquery:



Referências

- Referências Web:
  - https://www.altova.com/training/xquery3/constructors
  - http://www.swennenhuis.nl/basexfordummies/BaseX_for_dummies.pdf
  - https://docs.basex.org/wiki/Main_Page
- Livro:
  - Anders M. and Michel S., An introduction to XML and Web Technologies, Addison-Wesley, 2006;
  - Blokdyk, G. (2018). Extensible Markup Language XML A Complete Guide. 5STARCooks.

P.PORTO

ESCOLA SUPERIOR DE TECNOLOGIA E GESTÃO

Funções e Transformações XQuery

Bruno Oliveira (bmo@estg.ipp.pt)

PEI Processamento Estruturado de Informação