XQuery

Carlos A. Heuser
UFRGS

XQuery

- Linguagem de consulta do padrão XML
- Construída com base em alguns **requisitos**:
  - Ser **declarativa**
  - Ser consistente com **XMLSchema**
  - Suportar **quantificadores** (existencial e universal)
  - Combinar conteúdo de **vários documentos**
  - Capaz de **criar** e **transformar** documentos XML

Precursoras de XQuery

- Várias linguagens de consulta foram propostas antes de XQuery:
  - XML-QL
  - YATL
  - Lorel
  - Quilt

Conceitos básicos de XQuery

- Uma expressão Xquery
  - Lê uma seqüência de fragmentos XML ou átomos e
  - Gera uma seqüência de fragmentos XML ou átomos.
- Exemplos de expressões:
  - Expressão de caminho (XPath 2.0)

Xquery versus XPath

- **XPath**
  - Destina-se a **referenciar partes** de documentos
  - **Não gera** um novo documento
- **XQuery** (como XSLT)
  - Destina-se a **construir novos documentos** (não necessariamente XML)

Expressão de caminho

- Exemplo sobre artigo.xml:

```
doc("artigo.xml")//autor
```

- Obtém todos elementos **autor** dentro do documento.
- O operador "**doc**" devolve o documento XML indicado.

Expressão de caminho

- Exemplo sobre artigo.xml:

```
doc("artigo.xml")//autor
```

- Resultado:

```
<autor><nome>Arnaud Sahuguet</nome><instituicao>University of
  Pennsylvania</instituicao><endereco/></autor>
<autor><nome>Serge Abiteboul</nome><endereco/></autor>
<autor><nome>Peter Buneman</nome><endereco/></autor>
<autor><nome>Dan Suciu</nome><instituicao>University of
  Pennsylvania</instituicao><instituicao>ATT - Bell
  labs</instituicao><endereco/></autor>
<autor><nome>Peter Buneman</nome><endereco/></autor>
<autor><nome>Benjamin Pierce</nome><endereco/></autor>
<autor><nome>Mary F. Fernandez</nome><endereco/></autor>
<autor><nome>Dan suciu</nome><endereco><rua>Rua das
  Flores</rua></endereco></autor>
```


Expressão de caminho

- Exemplo sobre artigo.xml:

```
doc("artigo.xml")//autor
```

- Resultado:

```
<autor><nome>Arnaud Sahuguet</nome><instituicao>University of
  Pennsylvania</instituicao><endereco/></autor>
<autor><nome>Serge Abiteboul</nome><endereco/></autor>
<autor><nome>Peter Buneman</nome><endereco/></autor>
<autor><nome>Dan Suciu</nome><instituicao>University of
  Pennsylvania</instituicao><instituicao>ATT - Bell
  labs</instituicao><endereco/></autor>
<autor><nome>Peter Buneman</nome><endereco/></autor>
<autor><nome>Benjamin Pierce</no
<autor><nome>Mary F. Fernandez</
<autor><nome>Dan suciu</nome><en
  Flores</rua></endereco></autor
```

Observar que o resultado não necessariamente é um documento XML

Aqui: seqüência de elementos

Construtores de XML

- Elementos de XML pode ser construídos através de **construtores XML**:

```
<employee empid="12345">
     <name>John Doe</name>
     <job>XML specialist</job>
    <deptno>187</deptno>
    <salary>125000</salary>
</employee>
```

- Resulta em um documento com este conteúdo.

Expressão FOR

- A cláusula "for" liga uma variável a que fragmento resultante de uma expressão XPath

```
for $a  in doc('artigo.xml')//autor
return
<autor>
     <nome>{$a/nome/text()}</nome>
    {$a/endereco}
</autor>
```


Expressão FOR

- A cláusula "for return" executa uma expressão XQuery para cada fragmento resultante de uma expressão XPath

```
for $a  in doc('artigo.xml')//autor
return
<autor>
     <nome>{$a/nome/text()}</
    {$a/endereco}
</autor>
```

Variável definida pela expressão FOR

Percorre o resultado da expressão XPath na cláusula IN

Expressão FOR

- A cláusula "for return" executa uma expressão XQuery para cada fragmento resultante de uma expressão XPath

```
for $a  in doc('artigo.xml')//autor
return
<autor>
     <nome>{$a/nome/text()}</nome>
    {$a/endereco}
</autor>
```

RETURN contém a expressão avaliada para cada valor da variável $a

Resultado da consulta

```
<autor><nome>Arnaud Sahuguet</nome><endereco/></autor>
<autor><nome>Serge Abiteboul</nome><endereco/></autor>
<autor><nome>Peter Buneman</nome><endereco/></autor>
<autor><nome>Dan Suciu</nome><endereco/></autor>
<autor><nome>Peter Buneman</nome><endereco/></autor>
<autor><nome>Benjamin Pierce</nome><endereco/></autor>
<autor><nome>Mary F.
  Fernandez</nome><endereco/></autor>
<autor><nome>Dan suciu</nome><endereco><rua>Rua das
  Flores</rua></endereco></autor>
```


Misturando construtores de XML com expressões XQuery

- Outro exemplo:
- Resultado da consulta anterior é "**empacotado**" em um elemento XML
- Resultado da consulta agora é um documento XML bem formado

```
<autores>{
   for $a  in doc('artigo.xml')//autor
   return
   <autor>
        <nome>{$a/nome/text()}</nome>
      {$a/endereco}
    </autor>
   }</autores>
```


Misturando construtores de XML com expressões XQuery

- Outro exemplo:
- Resultado da consulta anterior é "empacotado" em um elemento XML
- Resultado da consulta agora é um documento XML bem formado

```
<autores>{
   for $a  in
   return
     <autor>
        <nome>{
        {$a/endereco}
     </autor>
   }</autores>
```

Observar o uso das chaves

Trecho dentro das chaves é uma expressão XQuery

Resultado

```
<autores>
  <autor><nome>Arnaud Sahuguet</nome><endereco/></autor>
  <autor><nome>Serge Abiteboul</nome><endereco/></autor>
  <autor>
    <nome>Peter Buneman</nome><endereco/></autor>
  <autor><nome>Dan Suciu</nome><endereco/></autor>
  <autor><nome>Peter Buneman</nome><endereco/></autor>
  <autor><nome>Benjamin Pierce</nome><endereco/></autor>
  <autor><nome>Mary F. Fernandez</nome><endereco/></autor>
  <autor><nome>Dan suciu</nome><endereco><rua>Rua das
  Flores</rua></endereco></autor>
</autores>
```

Usando várias variáveis

- A cláusula **FOR** permite a atribuição de várias variáveis

```
<autores>
  {
     for $a in doc('artigo.xml')//autor,
         $n in $a/nome,
         $i in $a/instituicao
     return
        <autor>{$n} {$i}</autor>
 }
 </autores>
```


Resultado

```
<autores>
  <autor>
       <nome>Arnaud Sahuguet</nome>
       <instituicao>University of Pennsylvania</instituicao>
  </autor>
  <autor>
       <nome>Dan Suciu</nome>
       <instituicao>University of Pennsylvania</instituicao>
  </autor>
  <autor>
       <nome>Dan Suciu</nome>
       <instituicao>ATT - Bell labs</instituicao>
  </autor>
</autores>
```


Cláusula WHERE

- Os valores (*bindings*) das variáveis definidas no **FOR**, pode ser filtrado através de uma cláusula **WHERE**

```
<bib>
{
 for $b in
  doc("artigo.xml")/artigo/bibliografia/referencia
 where $b/@id < 'b9' and $b/ano = 1999
 return
   <book year="{ $b/ano }">
    { $b/obra }
   </book>
}
</bib>
```


Resultado

```
<bib>
  <book year="1999">
       <obra>Data on the Web: From Relations to
  Semistructured Data and XML</obra>
  </book>
  <book year="1999">
       <obra>Union Types for Semistructured Data</obra>
  </book>
</bib>
```

Cláusula LET

- Cláusula **LET** atribui um conjunto de valores a uma variável
- Cláusula **FOR** atribuir cada um dos valores de um conjunto a uma variável
- No caso do exemplo abaixo, a cláusula **RETURN** é executada uma única vez para o valor das variáveis de **LET**

```
let $a  := doc('artigo.xml')//autor
return
<autores>{$a/nome/text()}</autores>
```


Resultado

```
<autores>Arnaud SahuguetSerge AbiteboulPeter BunemanDan
  SuciuPeter BunemanBenjamin PierceMary F. FernandezDan
  suciu</autores>
```


Cláusulas FOR aninhadas

- Cláusula FOR podem ser aninhadas, para construir coleções de elementos filhos de um elemento:

```
<bib>
{
 for $b in doc("artigo.xml")/artigo/bibliografia/referencia
 where $b/@id < 'b9' and $b/ano = 1999
 return
   <book year="{ $b/ano }">
    { $b/obra }
    <authors>
          {
           for $a in $b/autor
           return
              <author>
                   {$a/nome/text()}
              </author>
          }
    </authors>
   </book>
}
</bib>
```


Resposta

```
<bib>
   <book year="1999">
         <obra>Data on the Web: From Relations to Semistructured Data
   and XML</obra>
         <authors>
                  <author>Serge Abiteboul</author>
                  <author>Peter Buneman</author>
                  <author>Dan Suciu</author>
         </authors>
   </book>
   <book year="1999">
         <obra>Union Types for Semistructured Data</obra>
         <authors>
                  <author>Peter Buneman</author>
                  <author>Benjamin Pierce</author>
         </authors>
   </book>
</bib>
```

Referenciando diferentes documentos

- Diferentes documentos podem ser referenciados na consulta.
- Exemplo: junção de dados de dois documentos

```
<autores>
{
 for $aut in
  doc("artigo.xml")/artigo/bibliografia/referencia/autor
 for $pess in doc("pessoas.xml")/pessoas/pessoa
 where $aut/nome = $pess/nome
 return
   <author>
        <name>{ $aut/nome/text() }</name>
        <address>{ $pess/instituicao/text() }</address>
  </author>
}
</autores>
```


Resposta

```
<autores>
   <author>
          <name>Serge Abiteboul</name>
          <address>INRIA</address>
   </author>
   <author>
          <name>Peter Buneman</name>
          <address>Univ. of Edinburgh</address>
   </author>
   <author>
          <name>Dan Suciu</name>
          <address>UPenn</address>
   </author>
   <author>
          <name>Peter Buneman</name>
          <address>Univ. of Edinburgh</address>
   </author>
   <author>
          <name>Benjamin Pierce</name>
          <address>Stanford</address>
   </author>
   <author>
          <name>Mary F. Fernandez</name>
          <address>INRIA</address>
   </author>
</autores>
```


Outra sintaxe para a mesma consulta

```
<autores>
{
 for $aut in
  doc("artigo.xml")/artigo/bibliografia/referencia/autor,
      $pess in
  doc("pessoas.xml")/pessoas/pessoa[$aut/nome = nome]
 return
   <author>
        <name>{ $aut/nome/text() }</name>
        <address>{ $pess/instituicao/text() }</address>
  </author>
}
</autores>
```


Ordenação do resultado

- A cláusula ORDER permite classificar o resultado

```
<autores>
{
 for $aut in distinct-
  values(doc("artigo.xml")/artigo/bibliografia/referencia/aut
  or/nome)
 for $pess in doc("pessoas.xml")/pessoas/pessoa
 where $aut = $pess/nome/text()
 order by $pess/instituicao/text()
 return
   <author>
        <name>{ $aut }</name>
        <address>{ $pess/instituicao/text() }</address>
  </author>
}
</autores>
```

Resposta

```
<autores>
   <author>
         <name>Serge Abiteboul</name>
         <address>INRIA</address>
   </author>
   <author>
         <name>Mary F. Fernandez</name>
         <address>INRIA</address>
   </author>
   <author>
         <name>Benjamin Pierce</name>
         <address>Stanford</address>
   </author>
   <author>
         <name>Dan Suciu</name>
         <address>UPenn</address>
   </author>
   <author>
         <name>Peter Buneman</name>
         <address>Univ. of Edinburgh</address>
   </author>
</autores>
```


Uso de funções XPath (COUNT,POSITION…)

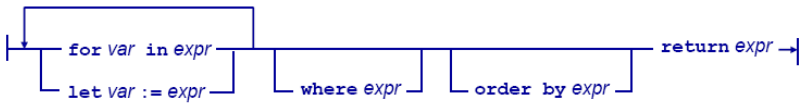
```
<bib>
 {
   for $b in doc("artigo.xml")/artigo/bibliografia/referencia
   where count($b/autor) > 1
   return
       <book>
           { $b/obra }
           {
               for $a in $b/autor[position()<=2]
               return $a
           }
           {
               if (count($b/autor) > 2)
                then <et-al/>
                else ()
           }
       </book>
 }
</bib>
```


Resposta

```
<bib>
   <book>
         <obra>Data on the Web: From Relations to Semistructured Data
   and XML</obra>
         <autor><nome>Serge Abiteboul</nome><endereco/></autor>
         <autor><nome>Peter Buneman</nome><endereco/></autor>
         <et-al/>
   </book>
   <book>
         <obra>Union Types for Semistructured Data</obra>
         <autor><nome>Peter Buneman</nome><endereco/></autor>
         <autor><nome>Benjamin Pierce</nome><endereco/></autor>
   </book>
   <book>
         <obra>Optimizing regular path expressions using graph
   Schemas</obra>
         <autor><nome>Mary F. Fernandez</nome><endereco/></autor>
         <autor><nome>Dan suciu</nome>
                <endereco><rua>Rua das Flores</rua></endereco></autor>
   </book>
</bib>
```


Juntando tudo: expressões FLOWR

- Sintaxe de uma expressão "**FLOWR**":



Expressão FLOWR

- Obter um documento XML que contém nomes de autores das referencias bibliográficas de 1999 e suas instituições em ordem de nome de instituição.

```
<authors>{
for $a in doc("artigo.xml")
            /artigo/bibliografia/referencia/autor
let $i:=doc("pessoas.xml")
         /pessoas/pessoa[nome=$a/nome]/instituicao
where $a/../ano = 1999
order by $i
return
  <author>
       <name>{$a/nome/text()}</name>
       <inst>{ $i/text() }</inst>
   </author>
}</authors>
```



```
<authors>
   <author>  <name>Serge Abiteboul</name>
             <inst>INRIA</inst>
   </author>
   <author>  <name>Benjamin Pierce</name>
             <inst>Stanford</inst>
   </author>
   <author>  <name>Dan Suciu</name>
             <inst>UPenn</inst>
   </author>
   <author>  <name>Peter Buneman</name>
             <inst>Univ. of Edinburgh</inst>
   </author>
   <author>  <name>Peter Buneman</name>
             <inst>Univ. of Edinburgh</inst>
   </author>
</authors>
```


Função distinct-values()

- Devolve um conjunto de valores atômicos sem duplicatas.

```
xquery version "1.0";
<instituicoes>
{for $i in  distinct-values(doc("pessoas.xml")//instituicao)
return
   <instituicao>
      {$i}
      <pessoas>
          {for $n in doc("pessoas.xml")
                        /pessoas/pessoa[instituicao=$i]/nome/text()
           return
               <pessoa>{$n}</pessoa>
          }
      </pessoas>
   </instituicao>}
</instituicoes>
```


Resposta

```
<instituicoes>
   <instituicao>INRIA<pessoas>
                <pessoa>Serge Abiteboul</pessoa>
                <pessoa>Mary F. Fernandez</pessoa>
         </pessoas>
   </instituicao>
   <instituicao>Univ. of Edinburgh<pessoas>
                <pessoa>Peter Buneman</pessoa>
         </pessoas>
   </instituicao>
   <instituicao>UPenn<pessoas>
                <pessoa>Dan Suciu</pessoa>
         </pessoas>
   </instituicao>
   <instituicao>Stanford<pessoas>
                <pessoa>Benjamin Pierce</pessoa>
         </pessoas>
   </instituicao>
</instituicoes>
```