XPath e XQuery

Carina F. Dorneles
dorneles@inf.ufsc.br - UFSC

Navegação/acesso aos dados em XML

- Encontrar os elementos através de **caminhos** que indiquem o contexto de tais elementos

- Para chegar a um elemento:
  - Como em URL:
    - Uso de caminho **absoluto**
      - Especificar toda a hierarquia de elementos de uma árvore XML, desde a raiz
    - Uso de caminho **relativo**
      - Especificar, em qualquer ponto do caminho, elementos relativos ao elemento contexto

XPath

- XPath usa expressões
  - *Strings* com símbolos significativos;

"/", "*", ".", "..", "//"

  - Exemplo:
    - Descer até o elemento titulo que é filho direto de um elemento livro:

livro/titulo ← String com símbolo significativo "/"

Caminhos Absolutos e Relativos

- Exemplo:

```
<curriculo>
    <dados>
       <nome> </nome>
       <cpf> </cpf>
       <fone> </fone>
    </dados>
    <experiencia>
       <profissional>
          <experiencias>
              <ano>1996</ano>
              <cargo>Analista</cargo>
           <experiencias>
       </profissional>
       <academica> ...
       </academica>
    </experiencia>

</curriculo>
```

Expressão XPath:

/curriculo/dados/nome/../../experiencia

Absoluto – desde a raiz Relativo – a "nome"

Filtros

- Usados para chegar a certos elementos, dada uma condição
- Sempre usar a condiçao dentro de [...]
- Exemplo: Um filtro pode ser usado para selecionar um elemento que está em uma posição determinada
  - Selecionar o primeiro parágrafo do capítulo

```
/livro/capitulo/paragrafo[1]
```

- Usa "[", e "]", para manipular o predicado. Os resultados do teste são um valor booleano, e a seleção só ocorre quando o valor é true.

Testes de elementos

- Selecionar um elemento, indicando o filho
  - Selecionar um elemento nota se ele contém diretamente um elemento tituto

    `nota[titulo]`

- Testar o valor de um elemento:
  - Selecionar a nota cujo titulo seja “Nota inicial”

    `nota[titulo=”Nota inicial”]`

    

    O que está dentro do colchetes não é retornado, serve apenas para teste
  - Selecionar o titulo com valor “Nota inicial”

    `nota[titulo=”Nota inicial”]/titulo`

Testes de atributos

- O símbolo @ é usado para representar um atributo e precede o nome do atributo
  - Selecionar o atributo autor do elemento livro:

    ```
    livro/@autor
    ```

  - Seleciona todo parágrafo com o valor do atributo tipo igual a 'secreto':

    ```
    para[@tipo='secreto']
    ```

Comparações

- Comparar dois números

```
paragrafo[position()=3]
```

- Comparar expressões booleanas e strings

```
paragrafo[titulo=”primeiro paragrafo"]
```

- Selecionar todos paragraf, mas não o último

```
paragrafo[position()!=last()]
```

- Outras comparações:

```
paragrafo[position()>2]
paragrafo[position()>=3]
paragrafo[position()>2 and position()<last()]
paragrafo[position()=2 or position() = 4]
```

Tratamento de Strings

- Função contains() retorna true se a string contém o texto dado
  - Usando "text()", testa somente o texto do elemento
    - Selecionar titulo que contenha a palavra "relacional"

```
titulo[contains(text(), "relacional")]

<titulo>Modelo relacional</titulo>
```

  - Usando ".", testa o elemento secao e seus subelementos
    - Selecionar secao que contenha a palavra "relacional"

```
secao[contains(., "relacional")]

<secao>Esta secao apresenta...
    <paragrafo>O modelo relacional ...</paragrafo>
    <paragrafo>Como já mencionado, ...</paragrafo>
</secao>
```

Resumo das funções, XPath 1.0

Tabela baseada nas funções apresentadas na página da W3C

- Funções para *nodos (elementos)*

Nome	***Sintaxe***	***Descrição***
`count()`	`count(node-set) = number`	Retorna o número de nodos de um node-set
`id()`	`id(value) = node-set`	Seleciona elementos pelo seu ID único
`last()`	`last() = number`	Retorna o número da posição do ultimo nodo em uma lista de nodos processados
`local-name()`	`local-name(node)= string`	Retorna a parte local de um nodo. Um nodo geralmente consiste de um prefixo, uma vírgula e seguida de um nome local
`name()`	`name(node) = string`	Retorna o nome de um nodo
`namespace-uri()`	`namespace-uri(node) = uri`	Retorna a URI da *namespace* de um nodo específico
`position()`	`position() = number`	Retorna a posição em uma lista de nodos do nodo que está sendo processado

Resumo das funções, XPath 1.0

Tabela baseada nas funções apresentadas na página da W3C

- Funções para *string*

Nome	***Sintaxe e Exemplo***	***Descrição***

`concat()`	`string=concat(val1, val2, ..)`
	Exemplo:
	`concat('The',' ','XML')`
	`Resultado: 'The XML'`
		Retorna a concatenação de todos os seus argumentos

`contains()`	`bool=contains(val,substr)`
	Exemplo:
	`contains('XML','X')`
	`Resultado: true`
		Retorna true se a segunda string está contida na primeira

`normalize-space()`	`string=normalize-space(string)`
	Exemplo:
	`normalize-space(' The XML ')`
	`Resultado: 'The XML'`
		Normaliza os espaços em broncos para um só

`starts-with()`	`bool=starts-with(string,substr)`
	Exemplo:
	`starts-with('XML','X')`
	`Resultado: true`
		Retorna true se a primeira string inicia com a segunda

`string()`	`string(value)`
	Exemplo:
	`string(314)`
	`Resultado: '314'`
		Converte o valor do argumento para string

Resumo das funções, XPath 1.0

Tabela baseada nas funções apresentadas na página da W3C

- Funções para *string*

Nome	***Sintaxe e Exemplo***	***Descrição***

`string-length()`	`number=string-length(string)`
	`Exemplo:`
	`string-length('Beatles')`
	`Resultado: 7`
		Retorna o número de caracteres em uma string

`substring()`	`string=substring(string,start,length)`
	`Exemplo:`
	`substring('Beatles',1,4)`
	`Resultado: 'Beat'`
		Retorna a parted a string indicada nos argumentos

`substring-after()`	`string=substring-after(string,substr)`
	`Exemplo:`
	`substring-after('12/10','/')`
	`Resultado: '10'`
		Retorna a parte da string que está depois do argumento substr

`substring-before()`	`string=substring-before(string,substr)`
	`Exemplo:`
	`substring-before('12/10','/')`
	`Resultado: '12'`
		Retorna a parted a string que está antes do argumento substr

`translate()`	`string=translate(value,string1,string2)`
	`Exemplo:`
	`translate('12:30','30','45')`
	`Resultado: '12:45'`
	`translate('12:30','03','54')`
	`Resultado: '12:45'`
	`translate('12:30','0123','abcd')`
	`Resultado: 'bc:da'`
		Executa reposição character a character.

Resumo das funções, XPath 1.0

Tabela baseada nas funções apresentadas na página da W3C

- Funções para *numéricos*

Nome	***Sintaxe e Exemplo***	***Descrição***

`ceiling()`	`ceiling(number) = number`
	`Exemplo:`
	`ceiling(3.14)`
	`Resultado: 4`
		Retorna o menor inteiro que não pe menor do que o argumento

`floor()`	`floor(number) = number`
	`Exemplo:`
	`floor(3.14)`
	`Resultado: 3`
		Retorna o maior inteiro que não é maior do que o argumento

`number()`	`number(value) = number`
	`Exemplo:`
	`number('100')`
	`Resultado: 100`
		Converte o valor do argumento para um numérico

`round()`	`round(number)= integer`
	`Exemplo:`
	`round(3.14)`
	`Resultado: 3`
		Arredonda o argumento ao inteiro mais próximo

`sum()`	`sum(nodeset)=number`
	`Exemplo:`
	`sum(/cd/price)`
		Retorna o valor total de um conjunto numérico de valores em um node-set

Resumo das funções, XPath 1.0

Tabela baseada nas funções apresentadas na página da W3C

- Funções *booleanas*

Nome	***Sintaxe e Exemplo***	***Descrição***
`boolean()`	`bool=boolean(value)`	Converte o argumento e retorna true ou false

`false()`	`false()`
	
	`Exemplo:`
	`number(false())`
	`Resultado: 0`
		Retorna false

`lang()`	`bool=lang(language)`	Retorna true se a linguagem do argumento casa com a linguagem do elemento xsl:lang

`not()`	`bool=not(condition)`
	
	`Exemplo:`
	`not(false())`
		Retorna true se a condição de argumento for falsa, e falsa se a condição for verdadeira

`true()`	`true()`
	
	`Exemplo:`
	`number(true())`
	`Resultado: 1`
		Retorna true

Exercício

- Usando o documento XML fornecido, e o processsador Xppath Visualiser, crie expressões XPath:

1. Encontre autores que possuam "Ana" no nome.

2. Recupere a versão do artigo

3. Selecionar nome do(s) autor(res) da bibliografia cujo titulo da obra é "**Union Types for Semistructured Data**".

XQuery

Introdução

- Baseada em Quilt
  (esta é baseada em XML-QL)
- http://www.w3.org/TR/xquery/2/2001

- XQuery usa
  - XPath
  - Tipo de dados do XML Schema

- XQuery não é sintaxe XML
  - Uma versão em sintaxe XML é chamada XQueryX

Introdução

- Uma consulta XQuery é uma expressão que:
  - Lê um documento XML, ou valores atômicos
  - Retorna um documento XML, ou valores atômicos

Introdução

- XQuery está para XML

Assim como

- SQL está para BD relacional

Principais formas de expressões

- Expressões de caminho (XPath)
- Construtores de elementos
- Expressões FLWOR ("*flower*")
- Expressões de lista
- Expressões condicionais
- Expressões quantificados
- Expressões de tipos de dados

Principais formas de expressões

- Expressões de caminho (XPath)
- Construtores de elementos
- Expressões FLWOR ("*flower*")
- Expressões de lista
- Expressões condicionais
- Expressões quantificados
- Expressões de tipos de dados

Expressões de caminho (XPath)

- A forma mais simples de expressão de caminho é uma expressão XPath

Expressões de caminho (XPath)

- A forma mais simples de expressão de caminho é uma expressão XPath
- Exemplo:

doc("receitas.xml")//receita[nome="Ricotta Pie"]//ingrediente[@quant]

Expressões de caminho (XPath)

- A forma mais simples de expressão de caminho é uma expressão XPath
- Exemplo:

doc("receitas.xml")//receita[nome="Ricotta Pie"]//ingrediente[@quant]

- O resultado é uma lista de fragmentos XML, cada um tendo como raiz o elemento ingrediente
- A ordem de saída dos fragmentos respeita a ordem do documento

Expressões de caminho (XPath)

doc(“receitas.xml")//receita[nome="Ricotta Pie"]//ingrediente[@quant]

- O contexto inicial da expressão de caminho é dado por

  doc("recipes.xml")

Construtores de elementos

- Uma expressão XQuery pode construir um novo elemento XML, que não existia no documento consultado

XQuery:

```
for $x in doc("books.xml")//book
return
  <livro>
    <titulo>{$x/title/text()}</titulo>
    {$x/author}
  </livro>
```

Construtores de elementos

- Exemplo

Doc. XML consultado:

```
<book>
    <title>XML and Web</title>
    <author>John</author>
</book>
```

Doc. XML resultante:

```
<livro>
    <titulo>XML and Web</titulo>
    <author>John</author>
</livro>
```

XQuery:

```
for $x in doc("books.xml")//book
return
  <livro>
    <titulo>{$x/title/text()}</titulo>
    {$x/author}
  </livro>
```

Exercicios

- Usando o documento XML fornecido, resolva as seguintes consultas em XQuery, usando XMLSpy.

1. Retornar autor do artigo cuja instituição seja UP. A estrutura de saída deve ser a seguinte:

```
<autor_artigo>
  <nome_autor>Maria Ana</nome_autor>
  <instituicao_autor>UP</instituicao_autor>
</autor_artigo>
```

Exercicios

2. Retornar referencia cujo titulo da obra comece com "Data on the Web". Neste caso devem ser geradas duas saídas:

A)

```
<referencia>
  <titulo>Data on the Web: From Relations to Semistructured Data and XML</titulo>
  <autor>Serge Abiteboul Peter Buneman Dan Suciu</autor>
  <ano>1999</ano>
</referencia>
```

B)

```
<referencia>
  <titulo>Data on the Web: From Relations to Semistructured Data and XML</titulo>
- <autor>
  <nome>Serge Abiteboul</nome>
  <nome>Peter Buneman</nome>
  <nome>Dan Suciu</nome>
  </autor>
  <ano>1999</ano>
</referencia>
```

Exercicios

3. Retornar o primeiro autor do artigo, com a seguinte estrutura:

<primeiro_autor>**Arnaud Sahuguet , University of Pennsylvania**</primeiro_autor>

4. Retornar o segundo autor da referencia citada na bibliografia, cujo titulo é "Union Types for Semistructured Data", com a seguinte estrutura:

<autor>Benjamin Pierce</autor>

XQuery - exemplos

Encontrar todos livros publicados após 1995:

```
for $x in doc("bib.xml")/bib/livro
where $x/ano > 1995
return $x/titulo
```

- Resultado:

<titulo> abc </titulo>

<titulo> def </titulo>

<titulo> ghi </titulo>

XQuery - exemplos

Listar os departamentos, o número de empregados e a média salarial

```
for $d in doc("depts.xml")//deptno  ($d é cada depto)
let $e := doc("funcs.xml")//funcionario[deptno = $d] ($d é a lista de func)
where count($e) >1
return

<Departamento>
   { $d}
      <qtdFuncionarios>{count($e)}</qtdFuncionarios>
      <mediaSalario>{avg($e/salary)}</mediaSalario>
</Departamento>
```

For e Let

- FOR $x in expr
  - liga a variável $x a cada elemento na lista de expr
  - for $i in 1 to 3 return element x {$i}

```
<x>1</x>
<x>2</x>
<x>3</x>
```

- LET $x := expr
  - liga $x a toda lista expr
  - let $i:=1 to 3 return element x {$i}

```
<x>1 2 3</x>
```

FOR v.s. LET

```
FOR $x IN doc("bib.xml")/bib/livro

RETURN <resultado> $x </resultado>
```

Retorna:

```
<resultado> <livro>...</livro></resultado>
<resultado> <livro>...</livro></resultado>
<resultado> <livro>...</livro></resultado>
<resultado> <livro>...</livro></resultado>
```

```
LET $x := doc("bib.xml")/bib/livro

RETURN <resultado> $x </resultado>
```

Retorna:

```
<resultado>
   <livro>...</livro>
   <livro>...</livro>
   <livro>...</livro>
   <livro>...</livro>
</resultado>
```

Exercícios

- Resolva as seguintes consultas usando ***for*** e ***let***

1. Listar títulos das obras e quantidade de autores
2. Retorne o titulo do artigo e a média dos valores do ano das obras das referencias