Engenharia de Dados e Conhecimento

XQuery Update Facility

W3C Working Group 24 January 2017

http://www.w3.org/TR/xquery-update-30/

XQUERY UPDATE FACILITY 3.0

What is XQUF?

universidade de aveiro

- The XQuery Update Facility 3.0 provides facilities to perform any or all of the following operations on an XML Document Model instance:
  - Insertion of a node. (*insert*)
  - Deletion of a node. (*delete*)
  - Modification of a node by changing some of its properties while preserving its node identity. (*rename*, *replace*)
  - Creation of a modified copy of a node with a new node identity. (*transform*)

XQUF – Rename

- Rename Syntax

- rename node <*N*> as <*name-expr*>
  - *N* – node (element or attribute) to rename
  - *name-expr* – must evaluate as a value of type xsd:QName

XQUF – Rename

universidade de aveiro

- Rename elements

```
let $d := doc('books')
for $a in $d//title
return rename node $a as 'SUBJECT'
```

Source:

```
<book category="COOKING">
   <title>Everyday</title>
   <author>Laurentiis</author>
...
</book>
```

Result:

```
<book category="COOKING">
   <SUBJECT>Everyday</SUBJECT>
   <author>Laurentiis</author>
...
</book>
```

XQUF – Rename

universidade de aveiro

- Rename attributes

```
let $d := doc('books')
for $a in $d//@*
return rename node $a as upper-case(name($a))
```

Source:

```
</bookstore>
  <book category="WEB">
    ...
    <year>2003</year>
    <price>39.95</price>
    <isbn>4</isbn>
  </book>
```

Result:

```
</bookstore>
  <book CATEGORY="WEB">
    ...
    <year>2003</year>
    <price>39.95</price>
    <isbn>4</isbn>
  </book>
```

XQUF – Delete

universidade de aveiro

- Delete Syntax

- delete node <*N*>
  - *N* – node (element or attribute) to delete
- delete nodes <*Ns*>
  - *Ns* – nodes (elements or attributes) to delete

XQUF – Delete

universidade de aveiro

- Delete elements

```
let $d := doc('books')
for $e in $d//price
return delete node $e
```

Source:

```
<book category="COOKING">
    ...
    <year>2005</year>
    <price>30.00</price>
    <isbn>1</isbn>
  </book>
```

Result:

```
<book category="COOKING">
    ...
    <year>2005</year>
    <isbn>1</isbn>
  </book>
```

XQUF – Delete

universidade de aveiro

- Delete attributes

```
let $d := doc('books')
for $a in $d//@lang
return delete node $a
```

Source:

```
<book category="CHILDREN">
 <title lang="en">Harry Potter</title>
 <year>2005</year>
</book>
```

Result:

```
<book category="CHILDREN">
 <title>Harry Potter</title>
 <year>2005</year>
</book>
```

XQUF – Delete

universidade de aveiro

- Delete selected nodes

```
let $d := doc('books')
for $e in $d//price[contains(../title, 'Potter')]
return delete node $e
```

Source:

```
<book category="CHILDREN">
 <title>Harry Potter</title>
 <author>J K. Rowling</author>
 <year>2005</year>
 <price>29.99</price>
 <isbn>2</isbn>
</book>
```

Result:

```
<book category="CHILDREN">
 <title>Harry Potter</title>
 <author>J K. Rowling</author>
 <year>2005</year>
 <isbn>2</isbn>
</book>
```

XQUF – Delete

universidade de aveiro

- Delete selected nodes text

```
let $d := doc('books')
for $e in $d//price[contains(../title, 'Potter')]
return delete node $e/text()
```

Source:

```
<book category="CHILDREN">
 <title>Harry Potter</title>
 <author>J K. Rowling</author>
 <year>2005</year>
 <price>29.99</price>
 <isbn>2</isbn>
</book>
```

Result:

```
<book category="CHILDREN">
 <title>Harry Potter</title>
 <author>J K. Rowling</author>
 <year>2005</year>
 <price />
 <isbn>2</isbn>
</book>
```

XQUF – Insert Node

universidade de aveiro

- Insert Syntax
- insert node <*A*> into <*B*>
  - node A becomes a new child of node B
- insert node <*A*> as first into <*B*>
  - node A becomes the first child of node B
- insert node <*A*> as last into <*B*>
  - node A becomes the last child of node B
- insert node <*A*> before <*B*>
  - node A becomes the first preceding sibling of node B
- insert node <*A*> after <*B*>
  - node A becomes the first following sibling of node B

XQUF – Insert

universidade de aveiro

- Insert elements

```
let $bs := doc('books')//book
for $i in 1 to count($bs)
return insert node element {'number'} {$i} as first into $bs[$i]
```

Source:

```
<book category="CHILDREN">
 <title>Harry Potter</title>
 ...
</book>
<book category="COOKING">
 <title>French Cusine</title>
 ...
</book>
```

Result:

```
<book category="CHILDREN">
 <number>1</number>
 <title>Harry Potter</title>
 ...
</book>
<book category="COOKING">
 <number>2</number>
 <title>French Cusine</title>
 ...
</book>
```

XQUF – Insert

universidade de aveiro

- Insert constant elements

```
let $bs := doc('books')//book
for $b in $bs
return insert node <publisher>Star Publishing</publisher>
before $b/year
```

Source:

```
<book category="CHILDREN">
 <title>Harry Potter</title>
 <author>J K. Rowling</author>
<year>2005</year>
 <price>29.99</price>
 <isbn>2</isbn>
</book>
```

Result:

```
<book category="CHILDREN">
 <title>Harry Potter</title>
 <author>J K. Rowling</author>
 <publisher>Star
Publishing</publisher>
 <year>2005</year>
 <price>29.99</price>
 <isbn>2</isbn>
</book>
```

XQUF – Insert Nodes

universidade de aveiro

- Insert Nodes Syntax
- insert nodes <*As*> into <*B*>
  - nodes As becomes a new child of node B
- insert nodes <*As*> as first into <*B*>
  - nodes As becomes the first child of node B
- insert nodes <*As*> as last into <*B*>
  - nodes As becomes the last child of node B
- insert nodes <*As*> before <*B*>
  - nodes As becomes the first preceding sibling of node B
- insert nodes <*As*> after <*B*>
  - nodes As becomes the first following sibling of node B

XQUF – Insert Nodes

- Insert multiple elements

```
let $bs := doc('books')//book
for $b in $bs
return insert nodes (
<publisher>Star Pubs</publisher>,
<library>LoC</library>
) before $b/year
```

Source:

```
<book category="CHILDREN">
 <title>Harry Potter</title>
 <year>2005</year>
 <price>29.99</price>
</book>
```

Result:

```
<book category="CHILDREN">
 <title>Harry Potter</title>
 <publisher>Star Pubs</publisher>
 <library>LoC</library>
 <year>2005</year>
 <price>29.99</price>
</book>
```

XQUF – Insert Nodes

universidade de aveiro

- Insert multiple attributes

```
let $bs := doc('books')//book
for $i in 1 to count($bs)
return insert nodes (
 attribute {'num'} {$i},
 attribute {'odd'} {$i mod 2 != 0}
) into $bs[$i]
```

Source:

```
<book cat...>
 ...
 <year>2005</year>ce>
</book>
<book cat...>
 ...
 <year>2001</year>ce>
</book>
```

Result:

```
<book num="1" odd="true" cat...>
 ...
 <year>2005</year>ce>
</book>
<book num="2" odd="false" cat...>
 ...
 <year>2001</year>ce>
</book>
```

XQUF – Multiple Operations

universidade de aveiro

- Multiple Operations
- It’s possible to run several primitive operations on one single instruction

- Example:
  - A kind of Replace = Delete + Insert

XQUF – Multiple Operations

universidade de aveiro

- Multiple Operations

```
let $bs := doc('books')//book
for $b in $bs
return
  let $name := name($b/@category)
  let $value := data($b/@category)
  return (
    delete node $b/@category,
    insert node element {$name} {$value} as first into $b
  )
```

XQUF – Multiple Operations

- Multiple Operations

Source:

```
<book category="Children">
 ...
</book>
<book category="Technology">
 ...
</book>
```

Result:

```
<book>
 <category>Children</category>
 ...
</book>
<book>
 <category>Technology</category>
 ...
</book>
```

XQUF – Replace Node

universidade de aveiro

- Replace Node Syntax

- replace node <*A*> with <*B*>
  - *A* – node (element or attribute) to be replaced
  - *B* – node (element or attribute) which will replace

XQUF – Replace Node

universidade de aveiro

- Replace Text Node

```
let $bs := doc('books')
for $y in $bs//book/year
where $y='2005'
return
  replace node $y/text() with '1995'
```

XQUF – Multiple Operations

universidade de aveiro

- Replace Text Node

Source:

```
<book>
  ...
  <year>2005</year>
</book>
<book>
  ...
  <year>2005</year>
</book>
```

Result:

```
<book>
...
<year>1995</year>
</book>
<book>
...
<year>1995</year>
</book>
```

XQUF – Replace Node

universidade de aveiro

- Replace Entire Node

```
let $bs := doc('books')
for $ib in $bs//book/isbn
where $ib < '3'
return
  replace node $ib with <DOI>{$ib/text()}</DOI>
```

XQUF – Multiple Operations

universidade de aveiro

- Replace Entire Node

Source:

```
<book>
 ...
 <isbn>1</isbn>
</book>
<book>
 ...
 <isbn>2</isbn>
</book>
```

Result:

```
<book>
...
 <DOI>1</DOI>
</book>
<book>
...
 <DOI>2</DOI>
</book>
```

XQUF – Issue

- Does the following XQuery code create an infinite loop?

```
let $doc := collection('books')
for $t in $doc//book/title
return
  insert node <title>{$t/text()}</title> after $t
```

XQUF – Issue

universidade de aveiro

- Infinite Loop??? NO!!!

Source:

```
<book>
 <title>Italian Cuisine</title>
...
</book>
<book>
 <title>Potter</title>
...
</book>
```

Result:

```
<book>
 <title>Italian Cuisine</title>
 <title>Italian Cuisine</title>
...
</book>
<book>
 <title>Potter</title>
 <title>Potter</title>
...
</book>
```

XQUF – XQuery Updates

universidade de aveiro

- XQuery updates do not apply during execution.
- The query will just return a pending update list.
- XQuery with Updates is a declarative language, so updates are applied at the end all together.
- That’s why last code doesn’t create an infinite loop.

XQUF – Pending Updates

universidade de aveiro

- <u>Accumulate Pending Updates</u>
- It's a list where XQuery engine saves all update operations for later execution.
- At the end of the query execution, the Pending Updates are applied all at once, and the XML is updated in atomic way.

XQUF – Transformation

universidade de aveiro

- Transform
- All the update primitives change the XML – they are Updating Expressions.
- Transform does not.
  - It operates on a copy of the XML nodes and updates that copy.
  - Transform is a Non-Updating Expression.
  - The updated copy can then be displayed or used to another purpose.

XQUF – Transform

universidade de aveiro

- Transformation Syntax
- copy $var1 := node1, $var2 := node2, $var3 := node3
- modify *update-expressions*
- return *expression*

XQUF – Transform

universidade de aveiro

- Transform a copy

```
copy $doc := collection('books')
modify (
for $b in $doc//book
return delete node $b/title
)
return $doc
```

- This code deletes all titles in the xml data set and returns the transformation.
- But, it doesn’t modify the xml source.