CS4221/CS5421

Tutorial 8: XQuery and XSLT

Mark Meng Huasong

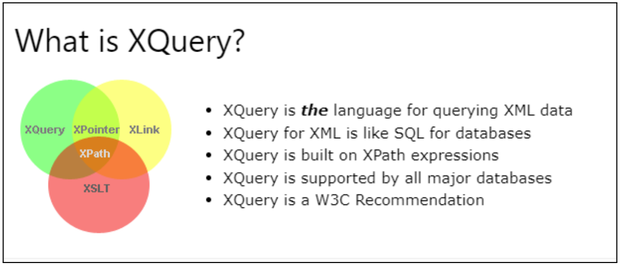
School of Computing
National University of Singapore

Week 10, 2022 Spring

An overview

We have covered XML (data insertion), DTD and XPath in the last week.

This week's tutorial covers XQuery and XSLT (eXtensible Stylesheet Language Transformations).



What is XQuery?

- XQuery is ***the*** language for querying XML data
- XQuery for XML is like SQL for databases
- XQuery is built on XPath expressions
- XQuery is supported by all major databases
- XQuery is a W3C Recommendation

Figure: Picture cited from W3 School
`https://www.w3schools.com/xml/xquery_intro.asp`

Question 1.1 (FLWOR Expression)

A breif intro:

(F) **for** + (L) **let** + (W) **where** + (O) **order by** + (R) **return** = FLWOR

```
for $loop_var in doc("library.xml")/child::library/...
  let $var := $loop_var/child::.../text()
  let $var_2 := count($loop_var/child::...)
  ...
  where $var [eq, ne, lt, le, gt, be, =, !=, <, >, <=, >=] {const}
  ...
  order by $var [ascending, descending]
  return
    {XML_expression_with_$var}
```

XQuery Syntax for Where Clause

For the difference of general comparisons (e.g., =, <=, and etc) and value comparisons (e.g., eq, le, and etc), you may go to W3 School tutorial to read more details.
`https://www.w3schools.com/xml/xquery_syntax.asp`

Question 1.1 Cont.

Find the albums where the number of songs is greater than or equal to 3. Return the album title and count sorted by the count with the highest count appearing first. Your query should be written in the form of a FLWOR expression.

For reference, your XML output should look somewhat similar to this:

```xml
<albums>
  <album>
    <title>No War</title>
    <count>4</count>
  </album>
  <album>
    <title>Bua Hati</title>
    <count>3</count>
  </album>
  <album>
    <title>Separuh Jiwaku Pergi</title>
    <count>3</count>
  </album>
</albums>
```

Question 1.1 Cont.

Solution:

```
<albums>{
  for $album in doc("library.xml")/child::library/child::album
    let $title := $album/child::title/text()
    let $count := count($album/child::songs/child::song)
    where $count ge 3
    order by $count descending
    return
      <album>
        <title>{$title}</title>
        <count>{$count}</count>
      </album>
}</albums>
```

Question 1.1 Cont.

There is an alternative of writing XML expression in element-wise manner

Native XML approach:

```
<album>
  for $album in doc("library.xml")/child::library/child::album
  ...
  return
    <album>
      <title>{$title}</title>
      <count>{$count}</count>
    </album>
}</albums>
```

Element-wise approach (out-of-scope and not recommended).:

```
<album>
  for $album in doc("library.xml")/child::library/child::album
  ...
  return
    element album{
      element title {$title},
      element count {$count}
    }
}</albums>
```

Question 1.2 (FLWOR Expression with Local Functions)

Show the title and duration of the songs made by Indonesian artists sorted by duration in descending order.

You may find declaring a local helper function as well as the built-in `tokenize($input, $pattern)` function useful. For reference, your output should look something like this:

```
<songs>
  <song>
    <title>Miliki Diriku</title>
    <duration>10:35</duration>
  </song>
  <song>
    <title>Belajarlah Untuk Cinta</title>
    <duration>10:23</duration>
  </song>
  <song>
    <title>Bua Hati</title>
    <duration>5:35</duration>
  </song>
...
```

```
...
  <song>
    <title>Timang-Timang</title>
    <duration>5:13</duration>
  </song>
  <song>
    <title>Separuh Jiwaku Pergi</title>
    <duration>5:00</duration>
  </song>
  <song>
    <title>Hujanpun Menangis</title>
    <duration>4:17</duration>
  </song>
</songs>
```

Question 1.2 Cont.

The definition of a local function is like:

```
declare function local:function_name($param_var as param_type) as return_type {
  let $local_var := ...
  return ...
};
```

Reference of XQuery Types: https://docs.oracle.com/cd/E13214_01/wli/docs92/xref/xqtypecon.html.
(e.g., xs:sting, xs:integer, xs:float, xs:double, xs:boolean, and etc)

Solution: Start by declaring a function to get the duration as the number of seconds with type integer:

```
declare function local:duration-in-sec($duration-str as xs:string) as xs:int {
  let $min-secs := tokenize($duration-str, ':')
  return 60 * xs:int($min-secs[1]) + xs:int($min-secs[2])
};
```

Question 1.2 Cont.

Let's write the XQuery statement in FLWOR format.

Option 1: with **let**-binding

```
<songs>{
  for $song in doc("library.xml")/child::library/child::album[child::artists/
        child::artist/
    child::country = 'Indonesia']/child::songs/child::song
    let $duration := local:duration-in-sec($song/child::duration)
    order by $duration descending
    return $song
}</songs>
```

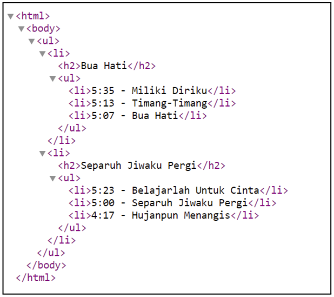
Option 2: without **let**-binding

```
<songs>{
  for $song in doc("library.xml")/child::library/child::album[child::artists/
        child::artist/
    child::country = 'Indonesia']/child::songs/child::song
    order by local:duration-in-sec($song/child::duration) descending
    return $song
}</songs>
```

Question 1.3 (Nested FLWOR)

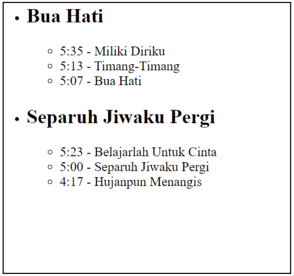
Show the list of titles of the albums made by Indonesian artists sorted by title. For each album show the songs sorted by duration in descending order.

For reference, your HTML output when rendered should look somewhat similar to this:



```
▼<html>
  ▼<body>
    ▼<ul>
      ▼<li>
          <h2>Bua Hati</h2>
        ▼<ul>
            <li>5:35 - Miliki Diriku</li>
            <li>5:13 - Timang-Timang</li>
            <li>5:07 - Bua Hati</li>
          </ul>
        </li>
      ▼<li>
          <h2>Separuh Jiwaku Pergi</h2>
        ▼<ul>
            <li>5:23 - Belajarlah Untuk Cinta</li>
            <li>5:00 - Separuh Jiwaku Pergi</li>
            <li>4:17 - Hujanpun Menangis</li>
          </ul>
        </li>
      </ul>
    </body>
  </html>
```

(HTML code view)



- **Bua Hati**
  - 5:35 - Miliki Diriku
  - 5:13 - Timang-Timang
  - 5:07 - Bua Hati
- **Separuh Jiwaku Pergi**
  - 5:23 - Belajarlah Untuk Cinta
  - 5:00 - Separuh Jiwaku Pergi
  - 4:17 - Hujanpun Menangis

(Direct output (as shown in eXide))

Question 1.3 Cont.

First of all, let's construct a draft.

```
<html>
  <body>
    <ul>{
      <!-- We only want albums made by Indonesian artists -->
      for $album in doc("library.xml")//album[artist/country = 'Indonesia']
        let $album-title := $album/title/text()
        order by $album-title
        return
          <li>
          <!-- For each album we want to output all songs titles and duration,
                sorted by their duration in descending order -->
          <h2>{$album-title}</h2>
          <ul>{
            for $song in $album//song
            order by local:duration-in-sec($song/duration) descending
            return
            <li>{$song/duration/text()} - {$song/title/text()}</li>
          }</ul>
          </li>
    }</ul>
  </body>
</html>
```

Question 1.3 Cont.

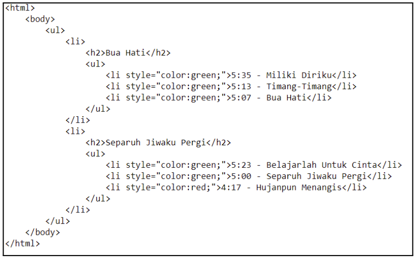
Solution: (Always use complete writing with "**child::**" in the exam/test.)

```
declare function local:duration-in-sec($duration-str as xs:string) as xs:int {
    let $min-secs := tokenize($duration-str, ':')
    return 60 * xs:int($min-secs[1]) + xs:int($min-secs[2])
};
<html>
  <body>
    <ul>{
      for $album in doc("library.xml")/child::library/child::album[child::artists/
           child::artist/child::country = 'Indonesia']
        let $album-title := $album/child::title/text()
        order by $album-title
        return
          <li>
            <h2>{$album-title}</h2>
            <ul>{
              for $song in $album/child::songs/child::song
                order by local:duration-in-sec($song/child::duration) descending
              return
                <li>{$song/child::duration/text()} - $song/child::title/text()}</li>
            }</ul>
          </li>
    }</ul>
  </body>
</html>
```

Question 1.4 (XQuery with Styled Output)

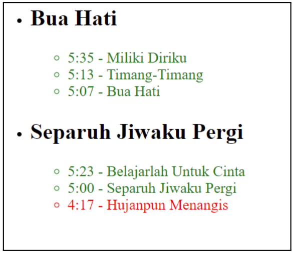
Color the songs with a duration greater than or equal to ~~600~~ **300** seconds green and otherwise red.

Note that "||" is used in XQuery for string concatenation. For reference, your HTML output when rendered should look somewhat similar to this:



```
<html>
    <body>
        <ul>
            <li>
                <h2>Bua Hati</h2>
                <ul>
                    <li style="color:green;">5:35 - Miliki Diriku</li>
                    <li style="color:green;">5:13 - Timang-Timang</li>
                    <li style="color:green;">5:07 - Bua Hati</li>
                </ul>
            </li>
            <li>
                <h2>Separuh Jiwaku Pergi</h2>
                <ul>
                    <li style="color:green;">5:23 - Belajarlah Untuk Cinta</li>
                    <li style="color:green;">5:00 - Separuh Jiwaku Pergi</li>
                    <li style="color:red;">4:17 - Hujanpun Menangis</li>
                </ul>
            </li>
        </ul>
    </body>
</html>
```

(HTML code view)



- **Bua Hati**
  - 5:35 - Miliki Diriku
  - 5:13 - Timang-Timang
  - 5:07 - Bua Hati
- **Separuh Jiwaku Pergi**
  - 5:23 - Belajarlah Untuk Cinta
  - 5:00 - Separuh Jiwaku Pergi
  - 4:17 - Hujanpun Menangis

(Direct output)

Solution shown in the next slide.

Question 1.4 Cont.

```
<!-- We keep the function local:duration-in-sec unchanged --><!-- omitted -->
<html>
  <body>
    <ul>{
      for $album in doc("library.xml")/child::library/child::album[child::artists/
           child::artist/child::country = 'Indonesia']
        let $album-title := $album/child::title/text()
        order by $album-title
        return
        <li>
        <h2>{$album-title}</h2>
        <ul>{
          for $song in $album/child::songs/child::song
            let $duration := local:duration-in-sec($song/child::duration)
            let $li-color := (if ($duration >= 300) then 'green' else 'red')
            order by $duration descending
            return
              <li style="color:{$li-color};">
                {$song/child::duration/text()} - {$song/child::title/text()}
              </li>
        }</ul>
        </li>
    }</ul>
  </body>
</html>
```

Question 1.4 Cont.

You can also get rid of let clause ($li_color) by defining color in the return:

```
...
return
  <li style="color:{(if␣($duration␣>=␣300)␣then␣'green'␣else␣'red')};">
    {$song/child::duration/text()} - {$song/child::title/text()}
  </li>
```

The element-wise approach alternative can be written as:

```
...
return
  element li {
    attribute style {
      if ($duration >= 600) then 'color:green;' else 'color:red;'
    },
    $song/child::duration/text() || '␣-␣' || $song/child::title/text()
  }
```

Question 1.5

Find the albums where the total listening duration (the sum of all the album's songs' duration) is greater than 1100 seconds. The albums should be sorted by title. In the XML output, you should return the album elements that satisfy the constraint with the title and an extra attribute duration. Your main code should be formed as a FLWOR expression. You may use functions you have declared in previous questions.

For reference, your XML output should look somewhat similar to this:

```
<albums>
  <album>
    <title>Bua Hati</title>
    <duration>1283</duration>
  </album>
  <album>
    <title>Separuh Jiwaku Pergi</title>
    <duration>1180</duration>
  </album>
</albums>
```

Question 1.5 Cont.

Solution:

```
<!-- We keep the function local:duration-in-sec unchanged -->
<albums>{
  for $album in doc("library.xml")/child::library/child::album
    let $title := $album/child::title/text()
    let $durations := $album/child::songs/child::song/child::duration
    let $total-duration := sum(for $duration in $durations return local:duration-in-
         sec($duration))
    where $total-duration gt 1100
    order by $title
    return
      <album>
        <title>{$title}</title>
        <duration>{$total-duration}</duration>
      </album>
}</albums>
```

Question 1.6

Explain a key difference between the for loop used in XQuery compared to an imperative language such as Python.

Solution:
XQuery is a functional language. Each iteration is executed in parallel and no communication between the threads are allowed. This is in contrast to imperative languages where the iterations are executed sequentially and may i.e. increment a counter or likewise. See XQuery/FLWOR Expression for more info.

To read more...

The XQuery Tutorial on W3 School:
`https://www.w3schools.com/xml/xquery_intro.asp`

Question (XSL) 2.1

Write an XSLT stylesheet that displays in a table or a list of an HTML document the name and released year of the albums with genre "Pop" of the XML document.

Prefer XSLT template matching with "**value-of**" and "**apply-template**" to imperative control structures (e.g. "**for-each**").

Question (XSL) 2.1 Cont.

Brief:

XSLT is another technique to make query.

XSL is still a (special) XML file with its own syntax.

XSL can be used to **navigate**, **query** and **transform** the data in the XML library.

XSLT describes the **transformation process** of a given XSL over the existing data.

Question (XSL) 2.1 Cont.

Solution: (Save the code below into an **.xsl** file)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0">
<!-- We generally always start with the root -->
<xsl:template match="/">
  <html>
    <body>
      <ul>
        <xsl:apply-templates select="child::library/child::album[child::genres/
             child::genre='Pop']"/>
      </ul>
    </body>
  </html>
</xsl:template>

<xsl:template match="album">
  <li>
    Album <xsl:value-of select="title"/> (<xsl:value-of select="year"/>)
  </li>
</xsl:template>

</xsl:stylesheet>
```

Question (Transformation) 2.2

Perform XLST transformation through eXist-db using the XLST file you just created.

Solution:

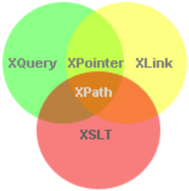
```
transform:transform(doc("library.xml"), doc("library.xsl"), ())
```

We don't need parameters in XSLT for this module.

To read more...

The XSLT Tutorial on W3 School:
https://www.w3schools.com/xml/xsl_intro.asp

To wrap-up: the big picture of XPath, XQuery and XSLT



Can you implement the Question 2 in XQuery?

```
<html>
  <body>
  <ul>{
    for $album in doc("library.xml")//album[genres/genre = '
         Pop']
    let $album-title := $album/title/text()
    let $album-year := $album/year/text()
    return
      <li> Album {$album-title} ({$album-year}) </li>
  }</ul>
  </body>
</html>
```

In fact, you can find the XQuery approach is simpler.

More exercise

Optional

Zemmy post a thread on the Luminus Forum for extra exercise on XQuery/XSLT. Feel free to have a try.

For any further question, please feel free to email me:

huasong.meng@u.nus.edu