8. xPath, xQuery & BaseX

www.LearnDB.com

Dr. Imed Bouchrika
Dept of Mathematics & Computer Science
University of Souk-Ahras
imed@imed.ws

Part I

xPath

XML : Reminder

- XML is a textual structured format for :
  - Storing data
  - Representing data
  - Communicating data.
- XML is based on opening and closing ***tags*** to enclose ***data content***.
- XML documents should comply with the rules defined within DTD or XSD,

XPATH : what

- xPath is a language used to search or explore parts of XML documents using Path Expressions.
- XPath is a major element in xQuery & XSLT
- XPath is a W3C recommendation

XPATH : Basics

- To learn xPath, we will address the following:
  - Accessing XML elements
  - Accessing Attributes
  - Adding Conditions
  - Using xPath Functions.

XPATH : XML Example

```xml
<?xml version="1.0"?>
<library>
    <book id="3">
         <title language="en">Feature Extraction</title>
        <author gender="male">Mark Nixon</author>
    </book>
    <book id="4">
        <title language="en">Java, A Beginner's Guide</title>
       <author gender="male">Herbert Schildt</author>
    </book>
    <dvd id="6">
         <title language="en">No Angel</title>
        <artist gender="male">Dido</artist>
         <genre year="1999">Pop</genre>
    </dvd>
    <dvd id="7">
         <title language="en">Old Yellow Moon</title>
        <artist gender="female">Emmylou Harris</artist>
        <genre year="2013">Country</genre>
    </dvd>
</library>
```

XPATH : Elements

- **/library/book/title**

  Select all **title** elements which are —UNDER—>**book**—UNDER—>**library**

```
<title language="en">Feature Extraction</title>
<title language="en">Java, A Beginner's Guide</title>
```

- First / → **Root Element.**

XPATH : Elements

- **//title**

  Select **all title** elements at **<u>any</u>** level or place within the document

```
<title language="en">Feature Extraction</title>
<title language="en">Java, A Beginner's Guide</title>
<title language="en">No Angel</title>
<title language="en">Old Yellow Moon</title>
```

XPATH : Elements

- **//author | //artist :**
  Bar → Concatenation of results
  Select all author elements **AND** all artist elements

```
<author gender="male">Mark Nixon</author>
<author gender="male">Herbert Schildt</author>
<artist gender="male">Dido</artist>
<artist gender="female">Emmylou Harris</artist>
```

XPATH : Elements

- **/library/book/child::*** **:**
  Select all **children** elements UNDER current element (**book**)

```
<title language="en">Feature Extraction</title>
<author gender="male">Mark Nixon</author>
<title language="en">Java, A Beginner's Guide</title>
<author gender="male">Herbert Schildt</author>
```

- ***** usually means everything or All

- Other keywords for xPath that can be used to access the tree of elements: **ancestor,attribute,..**

XPATH : Attributes

- **/library/book/title/@language**
  select the **attribute *language*** under the elements /library/book/title

```
en
en
```

- **/library/book/title/attribute::language**
  select the **attribute *language*** under the elements /library/book/title

```
en
en
```

XPATH : Attributes

- **/library/book/title/@***
  select **ALL** attributes under the elements /library/book/title.

```
en
en
```

XPATH : Conditions

- In short, we place conditions for xPath expression within
  [condition]

- . (one dot) means current node.

- .. (dot and dot) means go UP one level.

XPATH : Conditions

- **/library/book/title[../author='Mark Nixon']**
  List all elements of : /library/book/title.
  - **The condition** is that the author element which is found **one level up** from the current node (title)

    → should be equal to “Mark Nixon”.

```
<title language="en">Feature Extraction</title>
```

XPATH : Conditions

- **/library/book[@id<4]/title[../author='Mark Nixon']**
  List all elements of : /library/book/title.
  - Two conditions:
    - **[at book node]** Checks if the attribute named id is less than 4.
    - **[at element node]** The author element which is found **one level up** from the current node (title) should be equal to “Mark Nixon”.

```
<title language="en">Feature Extraction</title>
```

XPATH : Conditions

Operators	Explanation
+, - , *	Math Operations
div	Math Operation for Division
mod	Modulus operator
or	Logical or
and	Logical and
=	Equal
!=	Not Equal

XPATH : Functions

- **text( ) :**
  is used the access the text content of an element.

- Example:
  /library/book/title/text()

```
Feature Extraction
Java, A Beginner's Guide
```

XPATH : Functions

- **position( ) :**
  returns a number representing the position of this node in the sequence of nodes.

- Example:
  /library/book[position()=2]/title/text()

- List title **content** for the second book.

```
Java, A Beginner's Guide
```

XPATH : Functions

- **starts-with(s1,s2)**
  returns true if s1 starts with s2.

- Example:
  /library/dvd/artist[starts-with(text(),'D')]

- List element artists whose name starts with “**D**”

```
<artist gender="male">Dido</artist>
```

XPATH : Functions

- **contains(s1,s2)**
  returns true if s1 contains s2

- Example:
  //title[contains(text(),'A')]

- List elements whose titles contains “A”

```
<title language="en">Java, A Beginner's Guide</title>
<title language="en">No Angel</title>
```

XPATH : Functions

- **count(ABC) :**
  returns the number of elements named ***ABC*** under the current node.

- Example:
  //*[count(*)=3]/title

- List element titles where the parent has only three sub-elements.

```
<title language="en">No Angel</title>
<title language="en">Old Yellow Moon</title>
```

XPATH : Functions

- Other functions include :
  - string-length(ABC)
  - substring(string, start, length?)
  - not(...)
  - .....

Part II

xQuery

xQuery : what

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

xQuery : Basics

FOR ...

LET ...

WHERE...

ORDER BY...

RETURN...

→ FLWOR (Flower)

xQuery : XML Example

```
<?xml version="1.0"?>
<library>
    <book id="3">
         <title language="en">Feature Extraction</title>
        <author gender="male">Mark Nixon</author>
    </book>
    <book id="4">
        <title language="en">Java, A Beginner's Guide</title>
       <author gender="male">Herbert Schildt</author>
    </book>
    <dvd id="6">
         <title language="en">No Angel</title>
        <artist gender="male">Dido</artist>
         <genre year="1999">Pop</genre>
    </dvd>
    <dvd id="7">
         <title language="en">Old Yellow Moon</title>
        <artist gender="female">Emmylou Harris</artist>
        <genre year="2013">Country</genre>
    </dvd>
</library>
```

xQuery : Example

- Find all DVD titles published after 2005:

```
FOR $x IN document("bib.xml")//dvd
WHERE $x/genre/@year > 2005
RETURN $x/title/text()
```

```
Old Yellow Moon
```

Xquery : Nested For

- For each author, list all books they published:

```
FOR $a IN
            distinct(document("bib.xml“)//book/author)
RETURN  <result>
                $a,
                FOR $t IN //book[author=$a]/title
                RETURN $t
             </result>
```

- distinct = a function that eliminates duplicates

Xquery : Nested For

```
<result>
   <author gender="male">Mark Nixon</author>
   <title language="en">Feature Extraction</title>
</result>

<result>
   <author gender="male">Herbert Schildt</author>
   <title language="en">Java, A Beginner's Guide</title>
</result>
```

Xquery : For vs. Let

- FOR $x **in** expr
  - binds $x to each element in the list expr
- LET $x **:=** expr
  - binds $x to the entire list expr

XQuery

- Find all authors who have over 100 books

```
<result>
        FOR $p IN distinct(document("bib.xml")//author)
        LET $b := document("bib.xml")//book[author = $p]
        WHERE count($b) > 100
        RETURN $p
</result>
```

- count = a (aggregate) function that returns the number of elements

FOR v.s. LET

```
FOR $x IN document("bib.xml")//book
RETURN <result> $x </result>
```

Returns:

```
<result> <book>...</book></result>
<result> <book>...</book></result>
<result> <book>...</book></result>
...
```

```
LET $x := document("bib.xml")//book
RETURN <result> $x </result>
```

Returns:

```
<result> <book>...</book>
         <book>...</book>
         <book>...</book>
         ...
</result>
```

If-Then-Else

```
FOR $h IN //dvd

RETURN <result>

                  $h/title,

                  IF $h/@genre = “pop"

                          THEN $h/artist

                  ELSE $h/title

            </result>
```

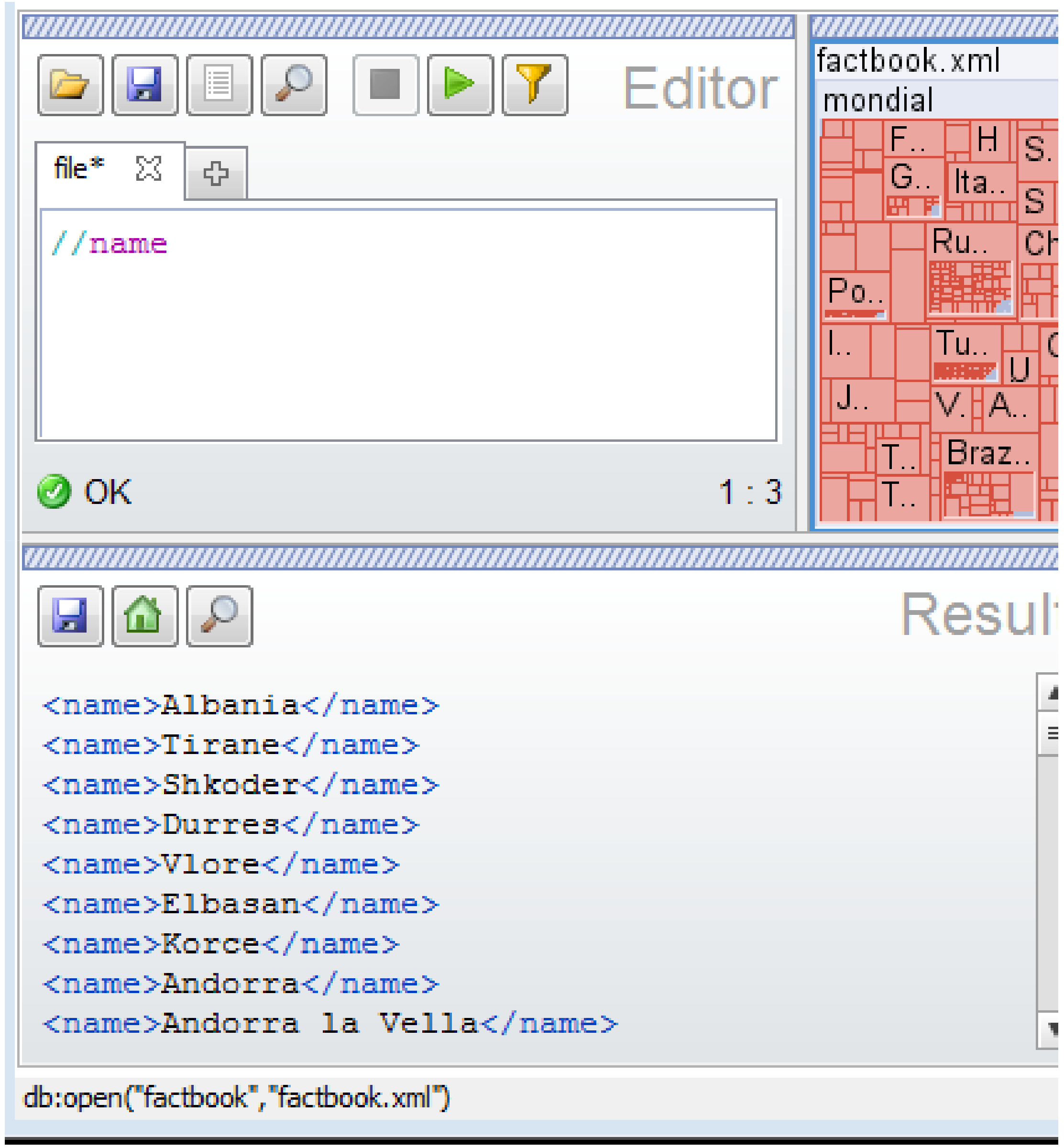
Part III

BaseX

BaseX : what

- ***BaseX*** is a native and light-weight XML database management system and XQuery processor.
- BaseX is specialized in storing, querying, and visualizing large XML documents.
- The XML DBMS is platform-independent
- BaseX is distributed under a free software license.
- Download from : http://www.basex.org

BaseX : Evaluating xPath

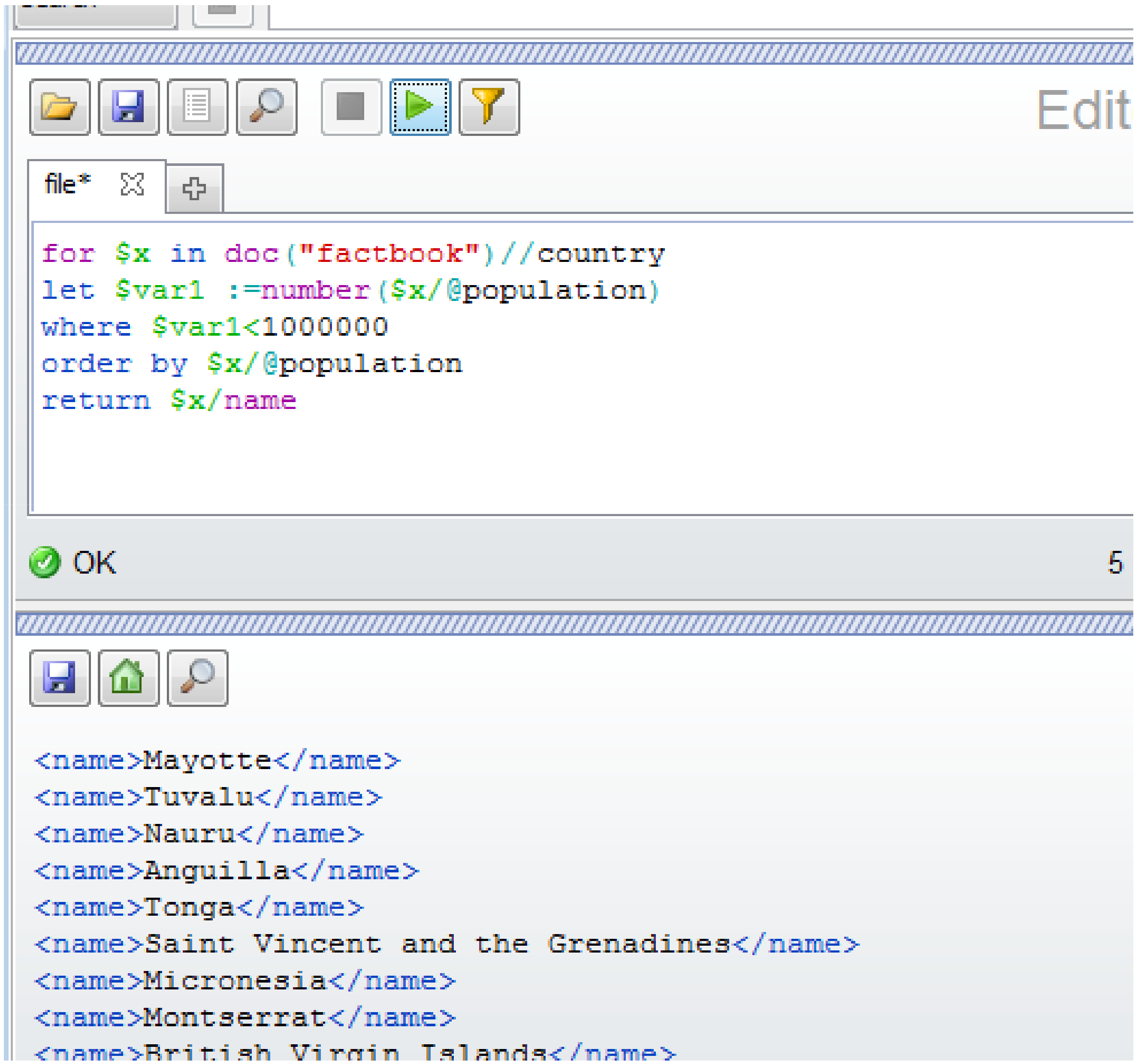


```
//name
```

```
<name>Albania</name>
<name>Tirane</name>
<name>Shkoder</name>
<name>Durres</name>
<name>Vlore</name>
<name>Elbasan</name>
<name>Korce</name>
<name>Andorra</name>
<name>Andorra la Vella</name>
```

```
db:open("factbook","factbook.xml")
```

BaseX : Evaluating xQuery



```
for $x in doc("factbook")//country
let $var1 :=number($x/@population)
where $var1<1000000
order by $x/@population
return $x/name
```

```
<name>Mayotte</name>
<name>Tuvalu</name>
<name>Nauru</name>
<name>Anguilla</name>
<name>Tonga</name>
<name>Saint Vincent and the Grenadines</name>
<name>Micronesia</name>
<name>Montserrat</name>
<name>British Virgin Islands</name>
```

For you to search !

- **SVG**
- **XUpdate**
- **eXist**
- **GROUP BY for xQuery**
- **SORTBY for xQuery**