Lenguaje de consultas XPath y XQuery

Preparación

NOTA: Nos colocamos en el directorio donde hemos instalado eXistDB

- Para iniciar el servicio: `> java -jar start.jar jetty`

Esta instrucción pone en marcha en servicio y habilita el acceso web de administración en: <IPdelaMáquina>:8080

- Para iniciar el cliente de consola: `> bin/client.sh -s -u admin -P austria`

Primeros pasos con eXists

- Documentación cliente de consola
- Para obtener ayuda online: escribir “help”
- Para acceder al panel de control (“Dashboard”) escribir en un navegador

```
http://<IPmáquina>:8080
```

Qué son XPath y XQuery

- Tanto XPath com XQuery son estándares para acceder y obtener datos desde documentos XML.
- Estos lenguajes tienen en cuenta que la información en los documentos está semiestructurada o jerarquizada como árbol.
- **XPath** → Lenguaje de rutas XML, se utiliza para navegar dentro de la estructura jerárquica de un XML
- **XQuery** → es a XML lo mismo que SQL es a las bbdd relacionales, es decir, un lenguaje de consulta diseñado para consultar documentos XML. XQuery contiene a XPath, toda expresión de consulta en XPath es válida en XQuery, pero XQuery permite mucho más

Expresiones XPath

- XPath es un lenguaje que permite seleccionar nodos de un documento XML y calcular valores a partir de su contenido. Existen varias versiones de XPath aprobadas por W3C aunque la versión más utilizada sigue siendo la 1.
- La forma en que XPath selecciona partes del documento XML se basa en la representación arbórea que se genera del documento.
- A la hora de recorrer un árbol XML podemos encontrarnos con los siguientes tipos de nodos:
  - **nodo raíz** → raíz del árbol, se representa por /
  - **nodos elemento** → cualquier elemento del árbol, son las etiquetas del árbol
  - **nodos texto** → los caracteres entre etiquetas
  - **nodos atributo** → propiedades añadidas a los nodos elementos, se representan con @
  - **nodos comentario** → etiquetas de comentario
  - **nodos espacio de nombres** → contienen espacio de nombres
  - **nodos instrucción de proceso** → instrucciones de proceso, van entre <?.....?>

Ejemplo

```
<?xml version =”1.0” encoding = “ISO-8859-1”?>
<universidad>
<espacio xmlns=”http://www.misitio.com”
      xmlns:prueba=”http://www.misitio.com/pruebas” />
<!-- DEPARTAMENTO -- >
<departamento telefono=”112233” tipo=”A”>
      <codigo>IFC1</codigo>
      <nombre>Informática</nombre>
</departamento>
….
</universidad>
```

Instrucción de proceso

Espacio de nombres

Tendríamos:

- Elementos: <universida>, <departamento>,<codigo>,<nombre>
- Texto: IFC1, Informática
- Atributo: telefono=”112233”, tipo=”A”
- Comentario: DEPARTAMENTO

Problemas: XPath_1

1. Clica en el dashboard de eXistDB el apartado “Collections”. Crea una nueva colección de documentos que llamarás “nueva”
2. Añade a la colección nueva todos los documentos contenidos en el archivo “ColecciónPruebas”
3. Clica en la app eXide. Desplázate por los directorios de la izquierda hasta localizar la colección “nueva”
4. Clica en algún documento para visualizarlo
5. Crea en la pestaña new XQuery una primera consulta:

```
doc(‘db/nueva/departamentos.xml’)/departamentos
```

Primeras consultas XPath

6. Comprueba el resultado de las siguientes consultas
    a. /departamentos → devuelve todos los datos de departamentos (Esta sería la misma consulta del ejercicio anterior pero sin indicar toda la ruta)
    b. /departamentos/DEP_ROW → devuelve todas las etiquetas de cada DEP_ROW
    c. /departamentos/DEP_ROW/DNOMBRE → devuelve nombres de departamentos entre etiquetas
    d. /departamentos/DEP_ROW/DNOMBRE/text() → Lo mismo que antes pero sin etiquetas
    e. //LOC/text() → localidades

NOTA: / se usa para dar rutas absolutas. Si el descriptor comienza con // se supone que la ruta descrita puede comenzar en cualquier parte

7. Ahora averigua el resultado de las siguientes consultas (utilizaremos el documento ‘db/nueva/empleados.xml’)
   a. /EMPLEADOS/EMP_ROW[DEPT_NO=10]
   b. /EMPLEADOS/EMP_ROW/APELLIDO|/EMPLEADOS/EMP_ROW/DEPT_NO
   c. /EMPLEADOS/EMP_ROW [DEPT_NO=10]/APELLIDO/text()
   d. /EMPLEADOS/EMP_ROW [not(OFICIO=’ANALISTA’)]
   e. /EMPLEADOS/EMP_ROW[SALARIO>1300 and DEPT_NO=20]/APELLIDO
   f. /EMPLEADOS/EMP_ROW[1]
8. Investiga en la web las siguientes funciones de XPath y pon algún ejemplo utilizando los documentos departamentos.xml y empleados.xml
   a. last()
   b. position()
   c. count()
   d. sum(),div(),mod()
   e. max(), min(),avg()
   f. concat(cadena1, cadena2,...)
   g. starts-with (cadena1, cadena2)
   h. contains(cad1,cad2)
   i. string-length(argumento)

9. Resuelve las siguientes consultas:
    a. Devuelve el apellido del penúltimo empleado (NOTA: utilizar last())
    b. Obtén los elementos del empleado que ocupa la posición 3 (position())
    c. Cuenta el número de empleados del departamento 10
    d. Obtén la suma de SALARIO de los empleados del DEPT_NO =20
    e. Obtén el salario máximo, el mínimo de los empleados con OFICIO=ANALISTA
    f. Obtén la media de salario en el DEPT_NO=10
    g. Devuelve la concatenación de apellido, oficio y salario
    h. Obtén los elementos de los empleados cuyo apellido empieza por ‘A’
    i. Devuelve los oficios que contienen la sílaba ‘OR’
    j. Obtén los datos de los empleados cuyo apellido tiene menos de 4 caracteres

10. Resuelve las siguientes consultas referentes al documento productos.xml. Este documento contiene los datos de los productos de una distribuidora de componentes informáticos. La estructura del documento es:

```
<produc>
  <cod_prod>xxx</cod_prod>
  <denominacion>xxxx</denominacion>
  <precio>xxx</precio>
  <stock_actual>xxx</stock_actual</stock_actual>
  <stock_minimo>xxxx</stock_minimo>
  <cod_zona>xxx</cod_zona>
</produc>
```

a. Obtén la denominación y precio de todos los productos
b. Obtén los productos que sean “Placa base”
c. Obtén los productos cuyo precio sea mayor que 60€ y de la zona 20
d. Obtén el número de los productos que sean memorias y de la zona 10
e. Obtén la media de los precios de los micros
f. Obtén los datos de los productos cuyo stock mínimo sea mayor que el stock actual (NOTA: usa función number())
g. Obtén el producto más caro
h. Obtén el producto más barato de la zona 20

Consultas XQuery

- Una consulta XQuery es una expresión que lee datos de uno o más documentos en XML y devuelve como resultado otra secuencia de datos en XML.
- XQuery contiene a XPath. Es decir, toda expresión de consulta en XPath es válida y devuelve el mismo resultado en XQuery.
- XQuery nos va a permitir:
  - Seleccionar información basada en un criterio específico
  - Buscar información en un documento o conjunto de documentos
  - Unir datos desde múltiples documentos
  - Transformar y reestructurar datos XML en otro vocabulario o estructura
  - ...

- En las consultas XQuery podemos utilizar las siguientes funciones para referirnos a colecciones y documentos dentro de la bbdd:
  - **collection(“/ruta”)** → indicamos el camino para referirnos a una colección
  - **doc(“/ruta/documento.xml”)** → indicamos el camino de un documento de una colección
- En XQuery las consultas se pueden construir utilizando expresiones FLWOR que corresponde a las siglas de **For, Let, Where, Order y Return**.
- La sintaxis general es:

```
for <variable> in <expresión XPath>
let <variables vinculadas>
where <condición XPath>
order by <expresión>
return <expresión de salida>
```

- **For** → se usa para seleccionar nodos y almacenarlos en una variable, similar a la cláusula FROM de SQL. Dentro del for escribimos una expresión XPath que seleccionará a los nodos. Si se especifica más de una variable en el for se actúa como producto cartesiano. Las variables comienzan con $
- Las consultas XQuery deben llevar obligatoriamente una orden **Return**, donde indicaremos lo que queremos que nos devuelva la consulta.
- Ejemplo comparativo XQuery - XPath

XQuery	XPath
for $emp in /EMPLEADOS/EMP_ROW return $emp	/EMPLEADOS/EMP_ROW
for $emp in /EMPLEADOS/EMP_ROW return $emp/APELLIDO	/EMPLEADOS/EMP_ROW/APELLIDO

- **Let** → permite que se asignen valores resultantes de expresiones XPath a variables para simplificar la representación. Se pueden poner varias líneas let una por cada variable, o separar las variables por comas.
- En el siguiente ejemplo, se crean dos variables $nom y $ofi. La salida sale ordenada por OFICIO, y se crea una etiqueta <APE_OFI> que incluye el nombre y oficio concatenado.

```
for $emp in /EMPLEADOS/EMP_ROW
let $nom:=$emp/APELLIDO, $ofi:=$emp/OFICIO
order by $emp/OFICIO
return <APE_OFI> {concat($nom, ‘ ‘ , $ofi)} </APE_OFI>
```

- **Where** → para filtrar elementos
- **Order by** → ordena los datos según un criterio dado
- **Return** → construye el resultado de la consulta en XML. Se pueden añadir etiquetas XML a la salida. Si lo hacemos, los datos a visualizar los encerramos entre llaves {}. Además, en el return se puede añadir condicionales usando **if-then-else**
- El siguiente ejemplo devuelve los departamentos de tipo A encerrados en una etiqueta:

```
for $dep in /universidad/departamento
return if ( $dep/@tipo='A')
        then <tipoA> {data ( $dep/nombre)} </tipoA>
```

- Utilizaremos la función **data()** para extraer el contenido en texto de los elementos.

Problemas XQuery_1

1. Prueba las siguientes expresiones en eXide y averigua qué devuelven:

```
for $emp in /EMPLEADOS/EMP_ROW
order by $emp/APELLIDO
return if ($emp/OFICIO='DIRECTOR')
then <DIRECTOR>{$emp/APELLIDO/text()}</DIRECTOR>
else <EMPLE> {data($emp/APELLIDO)} </EMPLE>
```

```
for $prof in /universidad/departamento[@tipo='A']/empleado
let $profe:=$prof/nombre, $puesto:=$prof/puesto
where $puesto='Profesor'
return $profe
```

```
for $dep in /universidad/departamento
return if ($dep/@tipo='A')
then <tipoA>{data($dep/nombre)}</tipoA>
else <tipoB>{data($dep/nombre)}</tipoB>
```

```
for $dep in /universidad/departamento
let $nom:=$dep/empleado
return <depart>{data($dep/nombre)}
        <emple>{count($nom)} </emple>
</depart>
```

```
for $dep in /universidad/departamento
let $emp:=$dep/empleado
let $sal:=$dep/empleado/@salario
return
        <depart>{data($dep/nombre)}
                <emple>{count($emp)}</emple>
                <medsal>{avg($sal)}</medsal>
        </depart>
```

```
for $dep in /universidad/departamento
let $emp:=$dep/empleado
let $sal:=$dep/empleado/@salario
let $maxi:=max($dep/empleado/@salario)
let $emplmax:=$dep/empleado[@salario=$maxi]
return
        <depart>{data($dep/nombre)}
                <emple>{count($emp)}</emple>
                <medsal>{avg($sal)}</medsal>
                <salmax>{$maxi}</salmax>
                <emplemax>{$emplmax/nombre/text()} - {data($emplmax/@salario)}</emplemax>
        </depart>
```

Operadores en XQuery

- Matemáticos: +, - , * , div, idiv (división entera), mod
- Comparación: <, > , =, !=, <=, >=, not()
- Redondeo: floor(), ceiling(), round()
- Funciones de agrupación: count(), min(), max(), avg(), sum()
- Funciones de cadena: concat(), string-length(), starts-with(), ends-with(), substring(), upper-case(), lower-case(), string()
- Uso general:
  - distinct-values() → extrae los valores de una secuencia de nodos y crea una nueva secuencia con valores únicos, eliminando los nodos duplicados
  - empty() → devuelve cierto cuando la expresión entre paréntesis está vacía.
  - exists() → devuelve cierto cuando una secuencia contiene, al menos, un elemento
- Los comentarios en XQuery van encerrados entre caras sonrientes (: blabla :)

Problemas XQuery_2

1. Resuelve las siguientes consultas utilizando el documento EMPLEADOS.xml
   a. Obtén los nombres de oficio que empiezan por P
   b. Obtén los nombres de oficio y el número de los empleados de cada oficio. Utiliza distinct-values
   c. Obtén el número de empleados que tiene cada departamento y la media de salario redondeada
2. Utilizando el documento productos.xml, resuelve con XQuery:
   a. Obtén por cada zona el número de productos que tiene
   b. Obtén la denominación de los productos entre las etiquetas <zona10></zona10> si son del código de zona 10, <zona20></zona20> si son del código de zona 20, etc.
   c. Obtén por cada zona la denominación del o de los productos más caros.
   d. Obtén la denominación de los productos contenida entre las etiquetas <placa></placa> para los productos en cuya denominación aparece la palabra Placa Base, <memoria></memoria>, para los que contienen la palabra Memoria <micro></micro>, para los que contienen la palabra Micro y <otros></otros> para el resto de productos