

UT05. 01_XML. TRATAMIENTO Y RECUPERACIÓN DE DATOS. XQUERY

Curso: 2DAW.

Lenguajes de Marcas y sistemas de gestión de información

Introducción

Introducción

- **XQuery**
 - Lenguaje de consulta que permite extraer y procesar información almacenada en formato XML, habitualmente en bases de datos nativas XML o en tablas y campos de tipo XML en bases de datos relacionales.
 - Parecidos razonables
 - **SQL** (Standard Query Language) en algunas de las cláusulas empleadas (where, order by).
 - **XPath**, con el que comparte modelo de datos y soporta las mismas funciones y operadores.
 - Se podría considerar a XQuery como un **superconjunto de XPath**, ya que toda expresión XPath es una expresión XQuery válida.

Introducción

- XQuery
 - Visión introductoria.
 - BaseX.
 - Documento **factbook.xml**

Elementos del lenguaje

- Lenguaje de expresiones
 - Todo es una expresión que se evalúa a un valor.
 - if (3 < 4) then “Verdadero” else “Falso” ➔ Verdadero
- Tipos de datos
 - Los tipos de datos primitivos (predefinidos) o atómicos (no compuestos) son los mismos que los de los esquemas XML.
 - Numéricos: enteros y reales.
 - Booleanos.
 - Cadenas de texto.
 - Fechas, horas y duraciones.
 - Tipos relacionados con XML, como QName.
 - Nodos XML: nodo raíz, elemento, atributo, texto, comentario, instrucción de procesamiento y espacio de nombres.

Elementos del lenguaje

- Secuencias
 - Listas de valores simples (atómicos).

Se declara una variable \$s1
y se le asigna una secuencia
de 3 elementos

\$s2 es la concatenación de la
primera consigo misma, de manera
que
contiene 6 elementos

\$s3 contiene un
único elemento.

\$s4 es vacía, no
contiene elementos

```
let $s1:= (2, 4, 6)
let $s2:= ($s1, $s1)
let $s3:= 10
let $s4:= ()
return (count ($s1), count($s2), count ($s3), count($s4))
```

Devuelve el tamaño de cada una de las secuencias, que será:
3 6 1 0

Elementos del lenguaje

- Con la ruta XPath `//city/name/text()` se obtienen una serie de nombres de ciudades (ubicadas en cualquier lugar del árbol XML), que se guardan en la variable `$city`, de ellas nos quedamos con las que empiecen por Q, se ordenan y se muestran.

```
for $city in doc ('mondial') //city/name/text()  
where starts-with($city, 'Q')  
order by $city  
return data($city)
```

Lectura de archivos

- **doc (*documento-XML*)**.- lee el documento XML y devuelve el nodo raíz o los elementos que se indiquen mediante una expresión XPath.

```
for $city in doc ('mondial') //city/name/text()  
where starts-with($city, 'Q')  
order by $city  
return data($city)
```


Cláusulas FLWOR

- **For-Let-Where-Order by-Return**
- **for**
 - Elementos se van a seleccionar (habitualmente desde un documento XML de partida).
 - Con cláusula **at**, permite numerar los elementos que se van procesando.
- **let**
 - Declara variables a las que se les asignan valores.
 - Opciones
 - Una cláusula **let** y variables separadas por comas
 - Una cláusula **let** por variable

Cláusulas FLWOR

- **where**
 - Permite introducir condiciones que deben de cumplir los elementos seleccionados por la cláusula for.
- **order by**
 - Permite ordenar los resultados de la consulta
- **return**
 - Devuelve los resultados

```
for $x in (1 to 5)
return <numero>{$x}</numero>
```

```
<numero>1</numero>
<numero>2</numero>
<numero>3</numero>
<numero>4</numero>
<numero>5</numero>
```

```
for $x at $i in
doc("clasicos.xml")/clasicos/clasico/titulo
return <libro>{$i}. {data($x)}</libro>
```

```
<libro>1. El señor de las moscas</libro>
<libro>2. El guardián entre el centeno</libro>
```

```
for $x at $i in doc("formacionProfesional")
/formacionProfesional/modulos/modulo/nombre
return <m>{$i}. {data($x)}</m>
```

```
<m>1. Sistemas operativos monopuesto</m>
<m>2. Lenguajes de marcas y sistemas de gestión de información</m>
<m>3. Servicios de red e internet</m>
<m>4. Programación</m>
```

Cláusulas FLWOR

- Se declaran dos variables, **\$alfa**, cuyos valores son 1 y 3, y **\$beta** cuyos valores son 2 y 4. Se genera una salida cuyo elemento raíz es **<datos>** y que combina todos los valores de **\$alfa** y **\$beta** en pares de valores. Esta expresión se rodea del elemento **<datos>**, lo que permite que la respuesta aparezca dentro del elemento **<datos>**.

```
<datos> {  
  for $alfa in (1,3), $beta in (2,4)  
  return <dato><alfa>{$alfa}</alfa>  
    <beta>{$beta}</beta></dato>  
} </datos>
```

```
<datos>  
  <dato>  
    <alfa>1</alfa>  
    <beta>2 </beta>  
  </dato>  
  <dato>  
    <alfa>1</alfa>  
    <beta>4</beta>  
  </dato>  
  <dato>  
    <alfa>3</alfa>  
    <beta>2</beta>  
  </dato>  
  <dato>  
    <alfa>3</alfa>  
    <beta>4</beta>  
  </dato>  
</datos>
```

Cláusulas FLWOR

Declaración con una cláusula let

```
let $x := 7, $y := 3 return 10*$x+$y
```

Declaración con una cláusula let por variable

```
let $x := 7 let $y := 3 return 10*$x+$y
```

Cláusulas FLWOR

Declaración de la variable
\$continente que contiene
todos los elementos
/mondial/continent

```
for $continente in doc('factbook')/mondial/continent
for $pais in doc('factbook')/mondial/country
where $continente/@id = $pais/encompassed/@continent
and $continente/@name="Europe"
return data($pais/@name)
```

Variable
\$pais que
contiene
todos los
elementos
/mondial/cou
ntry.

Se hace sólo para
los continentes cuyo
nombre es Europa.

El valor del atributo
@id de \$continente
tiene que ser igual
que el atributo
@continent de \$pais

Se devuelve el
nombre de los
países ubicados
en Europa.

Cláusulas FLWOR

- Transformar atributos en elementos con la función `data()`

Resultado dentro de un elemento **<continentes>**.

Variable **\$continentes** con todos los elementos `/mondial/continent`.

Ordena por **nombre**

```
<continentes>
{
  for $continentes in doc('factbook')/mondial/continent
  let $c:= $continentes/@name/data()
  order by $c
  return <continente>{$c}</continente>
}
</continentes>
```

Variable **\$c** con el valor del atributo **name** de cada elemento **continent**.

De cada elemento **<continent>** se devuelve el valor de su variable **name** dentro de un elemento **<continente>** que se crea en este momento.

Principales operadores y funciones de XQuery

- **Matemáticos:** `+`, `-`, `*`, `div`, `(*)`, `idiv(*)`, `mod`
- **Comparación:** `=`, `!=`, `<`, `>`, `<=`, `>=`, `not()`
- **Secuencia:** `union (|)`, `intersect`, `except`
- **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()`, `empty()`, `exists()`

Otras cláusulas

- **declare function**
 - Permite declarar funciones
- **if ... else**
 - permite declarar comportamientos condicionales
- Comentarios
(: **Esto es un comentario XQuery:**)

Funciones en XQuery

- Funciones predefinidas.
 - En muchos casos estas funciones coinciden con las existentes en **XPath**.
 - El **URI** del espacio de nombres de las funciones **XQuery** es el mismo que el de las funciones **XPath**.
<http://www.w3.org/2005/02/xpath-functions>.
 - El **prefijo** de estas funciones es **fn:** (habitualmente se omite).
 - **De texto:** `uppercase`, `substring`, `contains`, `starts-with`, `replace`, `normalize-space` ...
 - **Numéricas:** `max`, `abs`, `avg`, `sum`, `floor`...
 - **De fechas:** `current-date`, `current-time`, `day-from-date`, `hours-from-time`...
 - **De nodos XML:** `root`, `data`...

Funciones en XQuery

- Devolver el número de elementos **<members>** por elemento **<organization>**. La respuesta será una serie de números.

```
for $organizaciones in doc ('factbook')  
/mondial/organization  
return count($organizaciones/members)
```

- Devolver la media de los valores obtenidos en el ejemplo anterior ➔ **avg()**

```
avg (  
  for $organizaciones in doc('factbook')  
  /mondial/organization  
  return count ($organizaciones/members)  
)
```

Funciones en XQuery

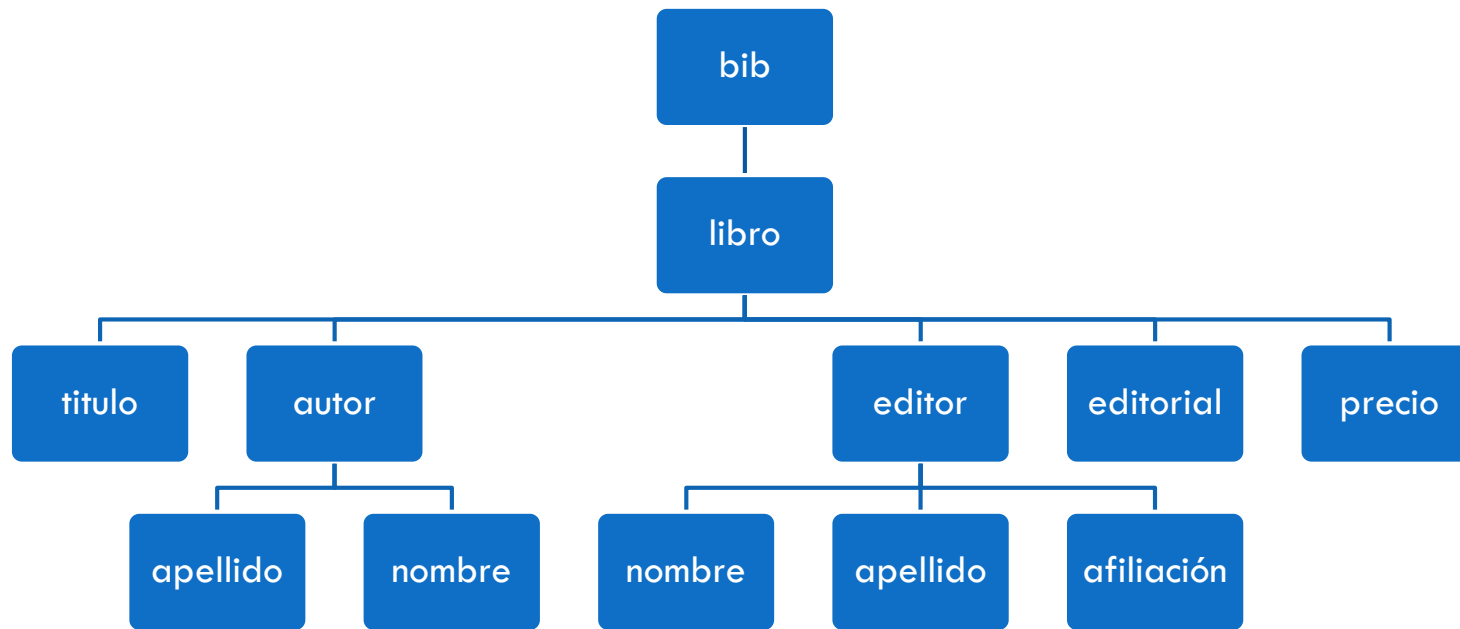
- Redondear a dos decimales lo anterior

```
round-half-to-even (  
  avg (  
    for $organizaciones in doc('factbook')  
    /mondial/organization  
    return count ($organizaciones/members)  
  )  
)
```

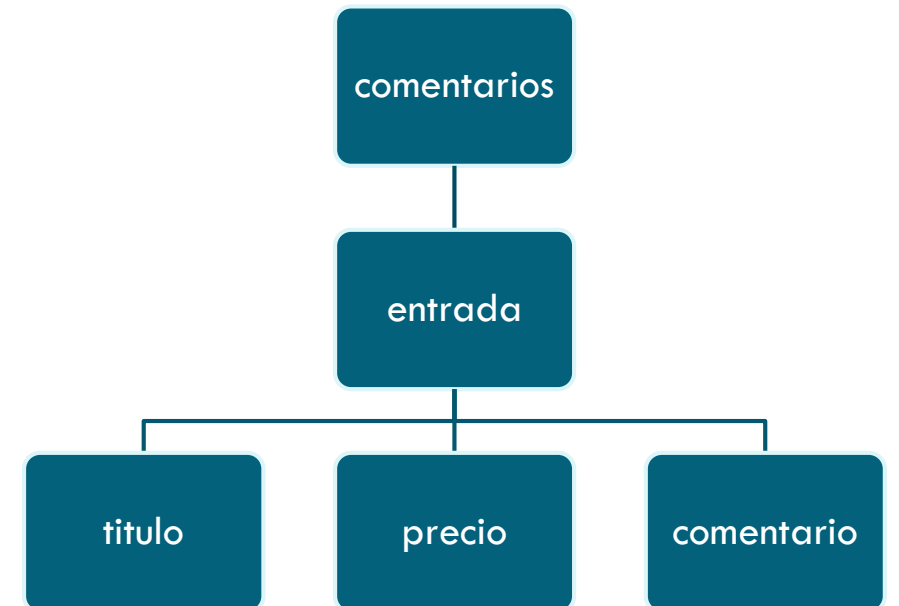
```
for $agua in doc('etc/factbook.xml') /mondial/sea | /mondia]/lake'  
let $nombre:= normalize-space($agua/@name), $tipo:= $agua/name()  
order by $nombre  
return_<agua tipo='{ $tipo}'>{ $nombre}</agua>
```

Ejercicios

libros.xml



comentarios.xml



Ejercicios I

- **1.** Devuelve los títulos de los libros que tengan más de dos autores ordenados por su título.
- **2.** Devuelve los títulos de los libros del año 2000.

Ejercicios II

- **3. Diferencia entre for y let:** Mostrar los títulos de todos los libros almacenados en el archivo “**libros.xml**”, primero con una cláusula **for** y, a continuación, con una cláusula **let** y vamos a detallar qué diferencia hay en la información obtenida. **return**
<titulos>{....}</titulos>

```
<titulos>
  <titulo>TCP/IP Illustrated</titulo>
  <titulo>Advanced Programming in Unix environment</titulo>
  <titulo>Data on the Web</titulo>
  <titulo>Economics of Technology for Digital TV</titulo>
</titulos>
```

```
<titulos>
  <titulo>TCP/IP Illustrated</titulo>
</titulos>
<titulos>
  <titulo>Advanced Programming in Unix environment</titulo>
</titulos>
<titulos>
  <titulo>Data on the Web</titulo>
</titulos>
<titulos>
  <titulo>Economics of Technology for Digital TV</titulo>
</titulos>
```


Ejercicios III

for vincula una variable con cada nodo que encuentre en la colección de datos, creando una tupla por cada título.

let vincula una variable con todo el resultado de una expresión, creando una única tupla con todos los títulos.

- **4.** Devuelve el título de cada uno de los libros de archivo **"libros.xml"** junto con el número de autores de cada libro.

```
<libro>
  <titulo>TCP/IP Illustrated</titulo>
  <autor>1</autor>
</libro>
<libro>
  <titulo>Advanced Programming in Unix environment</titulo>
  <autor>1</autor>
</libro>
<libro>
  <titulo>Data on the Web</titulo>
  <autor>3</autor>
</libro>
<libro>
  <titulo>Economics of Technology for Digital TV</titulo>
  <autor>0</autor>
</libro>
```

- Otra opción

```
<libros>
  <titulo>TCP/IP Illustrated</titulo>
  <autores>1</autores>
</libros>
<libros>
  <titulo>Advanced Programming in Unix environment</titulo>
  <autores>1</autores>
</libros>
<libros>
  <titulo>Data on the Web</titulo>
  <autores>3</autores>
</libros>
<libros>
  <titulo>Economics of Technology for Digital TV</titulo>
  <autores>0</autores>
</libros>
```

Ejercicios IV

Si en la consulta aparece más de una cláusula **for** o más de una variable en una cláusula **for**, el resultado es el producto cartesiano de dichas variables.

5. Devuelve los títulos de todos los libros contenidos en el archivo **libros.xml** y todos los comentarios de cada libro contenidos en el archivo **comentarios.xml**.

La cláusula **where** de una consulta permite filtrar las tuplas que aparecerán en el resultado y una expresión condicional permite crear una u otra estructura de nodos en el resultado que dependa de los valores de las tuplas filtradas.

```
<libros>
  <titulo>TCP/IP Illustrated</titulo>
  <comentarios>
    <comentario>Uno de los mejores libros de TCP/IP</comentario>
  </comentarios>
</libros>
<libros>
  <titulo>Advanced Programming in Unix environment</titulo>
  <comentarios>
    <comentario>Un libro claro y detallado de programación en UNIX.</
comentario>
  </comentarios>
</libros>
<libros>
  <titulo>Data on the Web</titulo>
  <comentarios>
    <comentario>Un libro muy bueno sobre bases de datos.</comentario>
  </comentarios>
</libros>
```

Con **if-then-else--**, la cláusula **else()** es obligatoria.

6. Devuelve los títulos de todos los libros almacenados en el archivo **libros.xml** y sus dos primeros autores. El formato de salida es el siguiente: elemento libro formado por elemento título y elemento autor (apellidos, coma, nombre)

```
<libro>
  <titulo>TCP/IP Illustrated</titulo>
  <autor>Stevens, W.</autor>
</libro>
<libro>
  <titulo>Advanced Programming in Unix environment</titulo>
  <autor>Stevens, W.</autor>
</libro>
<libro>
  <titulo>Data on the Web</titulo>
  <autor>Abiteboul, Serge</autor>
  <autor>Buneman, Peter</autor>
  <autor>et al.</autor>
</libro>
<libro>
  <titulo>Economics of Technology for Digital TV</titulo>
</libro>
```

Ejercicios V

XQuery soporta dos cuantificadores llamados **some** y **every**, de tal manera que nos permite definir consultas que devuelva algún elemento que satisfaga (**satisfies**) la condición (**some**) o consultas que devuelvan los elementos en los que todos sus nodos satisfagan la condición (**every**)

[some | every] variable in <ruta> satisfies(condición)

Cuando un cuantificador universal se aplica sobre un nodo vacío, siempre devuelve cierto

- **7.** Devuelve los títulos de los libros en los que al menos uno de sus autores es W. Stevens.



```
<titulo>TCP/IP Illustrated</titulo>  
<titulo>Advanced Programming in Unix environment</titulo>
```

Result

Ejercicios VI

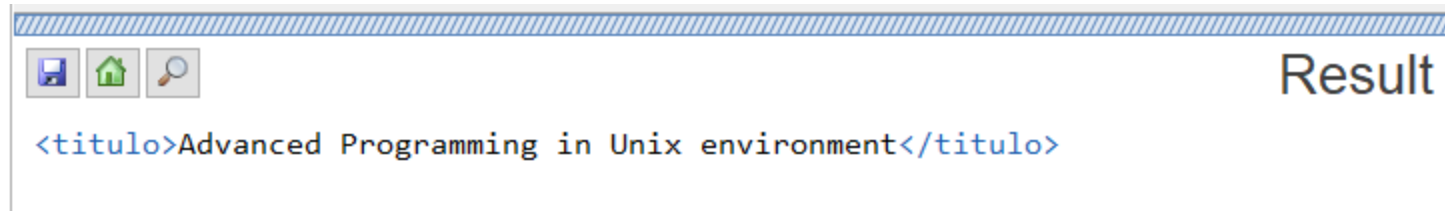
- 8. Devuelve **todos** los títulos de los libros en los que todos los autores de cada libro es W. Stevens.



Result

```
<titulo>TCP/IP Illustrated</titulo>  
<titulo>Advanced Programming in Unix environment</titulo>  
<titulo>Economics of Technology for Digital TV</titulo>
```

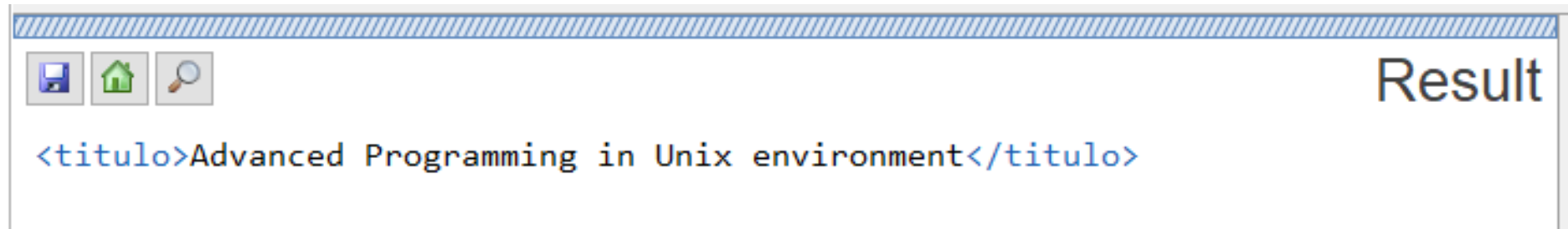
- **9.** Devuelve los títulos de los libros que mencionen “Unix” y “Programming” en el mismo título. Si el libro tiene más de un título solo es necesario que aparezca en, al menos, uno de ellos.



Result

```
<titulo>Advanced Programming in Unix environment</titulo>
```


- **10.** Devuelve el título de todos los libros que mencionen “Programming” en cada uno de los títulos de los libros almacenados en “libros.xml”.



The screenshot shows a web application interface. At the top, there is a blue header bar. Below the header, on the left, there are three icons: a blue folder icon, a green house icon, and a magnifying glass icon. On the right side of the header, the word "Result" is displayed in a large, bold, black font. Below the header, the main content area displays an XML snippet: `<titulo>Advanced Programming in Unix environment</titulo>`. The text is rendered in a monospaced font, with the opening and closing tags in blue and the text in black.

Ejercicios VII

- distinct-values()** extrae los valores de una secuencia de nodos y crea una nueva secuencia con valores únicos, eliminando los nodos duplicados.
- **11.** Obtener una lista ordenada de apellidos de todos los autores y editores.



```
<apellido>Abiteboul</apellido>  
<apellido>Buneman</apellido>  
<apellido>Gerbarg</apellido>  
<apellido>Stevens</apellido>  
<apellido>Suciu</apellido>
```

except permite excluir del resultado un elemento

- **12.** Obtener un nodo libro con todos sus nodos hijos salvo el nodo **<precio>**.



```
<libro año="1994">
  <titulo>TCP/IP Illustrated</titulo>
  <autor>
    <apellido>Stevens</apellido>
    <nombre>W.</nombre>
  </autor>
  <editorial>Addison-Wesley</editorial>
</libro>
```

- **13.** Devuelve todos los apellidos distintos de los autores.



```
<apellido>Stevens</apellido>  
<apellido>Abiteboul</apellido>  
<apellido>Buneman</apellido>  
<apellido>Suciu</apellido>
```

La función **empty()** devuelve cierto cuando la expresión entre paréntesis está vacía.

- **14.** Devuelve todos los nodos libro que tengan al menos un nodo autor.

```
<libro año="1994">
  <titulo>TCP/IP Illustrated</titulo>
  <autor>
    <apellido>Stevens</apellido>
    <nombre>W.</nombre>
  </autor>
  <editorial>Addison-Wesley</editorial>
  <precio>65.95</precio>
</libro>
<libro año="1992">
  <titulo>Advanced Programming in Unix environment</titulo>
  <autor>
    <apellido>Stevens</apellido>
    <nombre>W.</nombre>
  </autor>
  <editorial>Addison-Wesley</editorial>
  <precio>65.95</precio>
</libro>
<libro año="2000">
  <titulo>Data on the Web</titulo>
  <autor>
    <apellido>Abiteboul</apellido>
    <nombre>Serge</nombre>
  </autor>
  <autor>
    <apellido>Buneman</apellido>
    <nombre>Peter</nombre>
  </autor>
  <autor>
    <apellido>Suciu</apellido>
    <nombre>Dan</nombre>
  </autor>
  <editorial>Morgan Kaufmann editorials</editorial>
  <precio>39.95</precio>
</libro>
```

Ejercicios VIII

La función opuesta a **empty()** es **exists()**, la cual devuelve cierto cuando una secuencia contiene, al menos, un elemento.

- **15.** Reescribir la consulta anterior usando la función **exists()**.



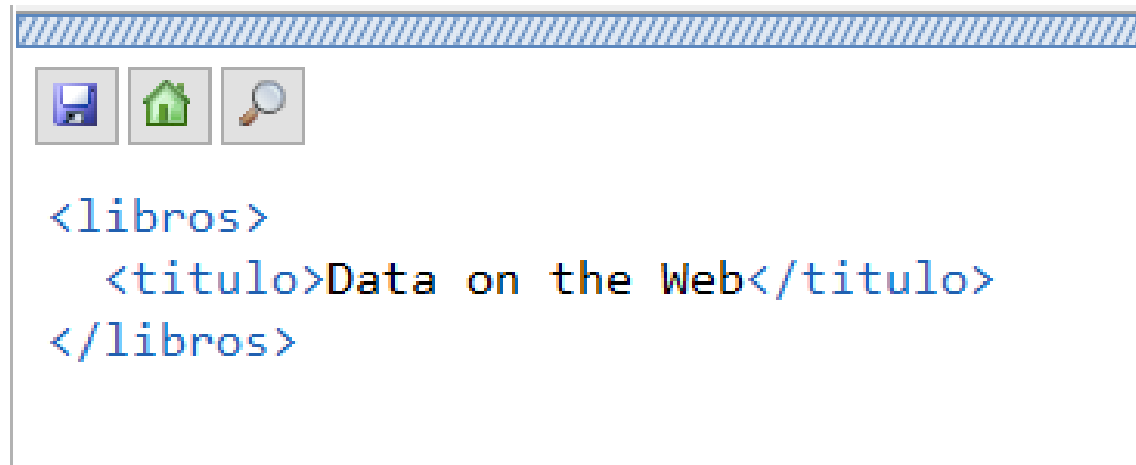
```
<libro año="1994">
  <titulo>TCP/IP Illustrated</titulo>
  <autor>
    <apellido>Stevens</apellido>
    <nombre>W.</nombre>
  </autor>
  <editorial>Addison-Wesley</editorial>
  <precio>65.95</precio>
</libro>
<libro año="1992">
  <titulo>Advanced Programming in Unix environment</titulo>
  <autor>
    <apellido>Stevens</apellido>
    <nombre>W.</nombre>
  </autor>
  <editorial>Addison-Wesley</editorial>
  <precio>65.95</precio>
</libro>
<libro año="2000">
  <titulo>Data on the Web</titulo>
  <autor>
    <apellido>Abiteboul</apellido>
    <nombre>Serge</nombre>
  </autor>
  <autor>
    <apellido>Buneman</apellido>
    <nombre>Peter</nombre>
  </autor>
  <autor>
    <apellido>Suciu</apellido>
    <nombre>Dan</nombre>
  </autor>
  <editorial>Morgan Kaufmann editorials</editorial>
  <precio>39.95</precio>
</libro>
```

- **16.** Consulta que obtenga el nombre y el año de todos los libros publicados por Addison-Wesley después de 1991.

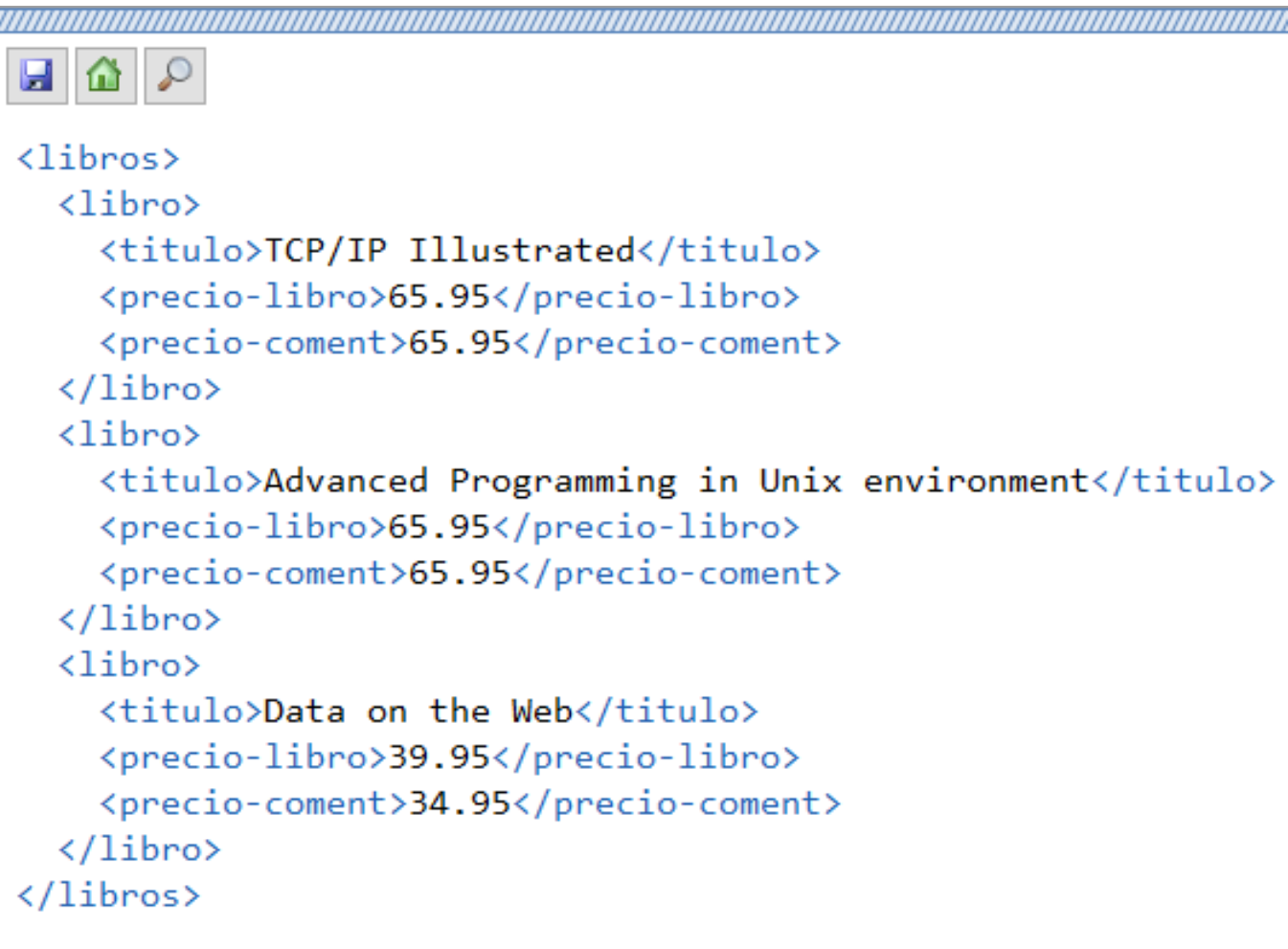


```
<libros>
  <libro año="1994">
    <titulo>TCP/IP Illustrated</titulo>
  </libro>
  <libro año="1992">
    <titulo>Advanced Programming in Unix environment</titulo>
  </libro>
</libros>
```

- **17.** Escribir una consulta que obtenga el título de los libros cuyo precio esté por debajo de 50.00€.



- **18.** Escribir una consulta que, por cada libro almacenado en el archivo **libros.xml** devuelva el título del libro, el precio con que consta dicho libro en el archivo **libros.xml** y el precio con que consta ese libro en el archivo **comentarios.xml**

A screenshot of a code editor window with a blue header bar. The header bar contains three icons: a floppy disk (save), a house (home), and a magnifying glass (search). The editor displays XML code for a list of books. The code is as follows:

```
<libros>
  <libro>
    <titulo>TCP/IP Illustrated</titulo>
    <precio-libro>65.95</precio-libro>
    <precio-coment>65.95</precio-coment>
  </libro>
  <libro>
    <titulo>Advanced Programming in Unix environment</titulo>
    <precio-libro>65.95</precio-libro>
    <precio-coment>65.95</precio-coment>
  </libro>
  <libro>
    <titulo>Data on the Web</titulo>
    <precio-libro>39.95</precio-libro>
    <precio-coment>34.95</precio-coment>
  </libro>
</libros>
```

Ejercicios IX


- **19.** Escribir una consulta que, por cada libro con autores, devuelva el título del libro y sus autores. Si el libro no tiene autores pero sí editor, la consulta devolverá el título del libro y la afiliación del editor.



```
<bib>
  <libro>
    <titulo>TCP/IP Illustrated</titulo>
    <autor>
      <apellido>Stevens</apellido>
      <nombre>W.</nombre>
    </autor>
  </libro>
  <libro>
    <titulo>Advanced Programming in Unix environment</titulo>
    <autor>
      <apellido>Stevens</apellido>
      <nombre>W.</nombre>
    </autor>
  </libro>
  <libro>
    <titulo>Data on the Web</titulo>
    <autor>
      <apellido>Abiteboul</apellido>
      <nombre>Serge</nombre>
    </autor>
    <autor>
      <apellido>Buneman</apellido>
      <nombre>Peter</nombre>
    </autor>
    <autor>
      <apellido>Suciu</apellido>
      <nombre>Dan</nombre>
    </autor>
  </libro>
  <referencia>
    <titulo>Economics of Technology for Digital TV</titulo>
    <afiliacion>CITI</afiliacion>
  </referencia>
</bib>
```

Ejercicios X

- función **deep-equal()** encargada de comparar secuencias de nodos: dos consultas son iguales si todos los nodos de la primera secuencia aparecen en la segunda secuencia en la misma posición que en la primera secuencia.
- **20.** Mostrar los títulos que sean distintos pero tengan el mismo autor o grupo de autores. Hay que tener en cuenta que el orden de aparición de los autores puede variar de un libro a otro.



```
<bib>
  <libro-par>
    <titulo>TCP/IP Illustrated</titulo>
    <titulo>Advanced Programming in Unix environment</titulo>
  </libro-par>
</bib>
```

Operadores de comparación de orden de nodos comparan pares de nodos según sus posiciones en un documento.

- A continuación se exponen las comparaciones que se realizan en función del orden en el documento:
- <<: **Operando 1** preceden **operando 2** en el orden del documento.
- >>: **Operando 1** siga **operando 2** en el orden del documento.

-
- **21.** También es posible utilizar XQuery para transformar datos XML en otros formatos, como HTML, convirtiéndose XQuery en una alternativa más sencilla y rápida de usar que XSLT. Ejemplo: consulta que crea una tabla HTML con los títulos de todos los libros contenidos en el archivo **libros.xml**.