

Tema 6. Anexo - XQuery

Actualizado al 07/02/2023

6.1.	Introducción	2
6.2.	Expresiones	2
6.2.1.	Expresiones FLWOR (FLWOR expressions).....	2
6.2.2.	Expresión “Let”	3
6.3.	Funciones	3
6.3.1.	data()	3
6.3.2.	Funciones con cadenas.....	4
6.3.2.1.	concat().....	4
6.3.2.2.	substring()	4
6.3.2.3.	upper-case()	4
6.3.3.	Estructura de control condicional	5
6.3.4.	Funciones “de agregado”	5
6.4.	Combinar resultados con HTML.....	6

6.1. Introducción

XQuery incorpora todo lo visto en XPath y, además, algunas posibilidades que éste último no posee. En este anexo al tema vamos a ver algunas características propias de XQuery, que nos ayudarán en el acceso a datos almacenados en ficheros o bases de datos XML.

Para ello, nos ayudaremos de BaseX. Hay que remitirse al capítulo anterior para aprender a instalar y registrar una base de datos XML en este sistema gestor.

6.2. Expresiones

6.2.1. Expresiones FLWOR (FLWOR expressions)

FLWOR es acrónimo de For, Let, Where, Order by, Return. Una expresión FLWOR representa a un conjunto de datos que se itera, filtrándolos y ordenándolos para terminar devolviendo valores.

Veamos un ejemplo. Si queremos obtener los nombres de los regantes cuyo identificador es menor que 3 podríamos usar la sentencia:

```
1 db:open("xmlregantes")/datos/regantes[id<3]/nombre
```

O bien la siguiente expresión FLWOR:

```
1 for $reg in db:open("xmlregantes")/datos/regantes
2 where $reg/id < 3
3 return $reg/nombre
```

Donde \$reg es una variable declarada (let) dentro de un bucle (for) que recorre datos XML que pueden ser filtrados (where) para posteriormente ser devueltos (return).

Puede que parezca que hemos escrito mucho más que antes innecesariamente, pero lo cierto es que ahora se nos abren más posibilidades. De entrada, tenemos que:

- Podemos usar la cláusula de ordenación (order by).
- Tenemos más control sobre el aspecto de los resultados devueltos.

Veremos que disponemos también de otras nuevas facilidades y posibilidades.

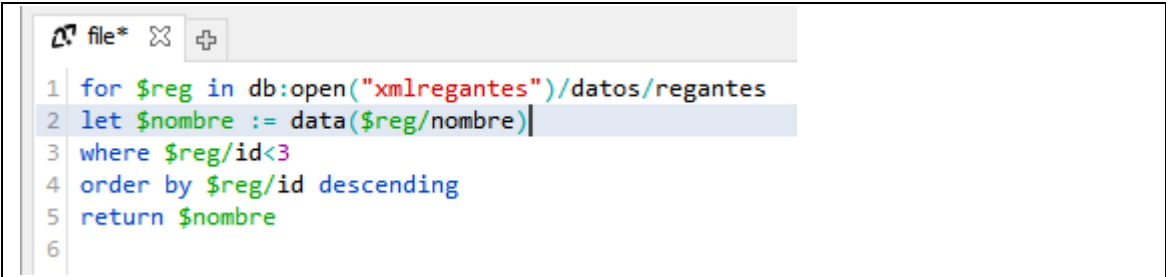
Por ejemplo, partiendo de la consulta anterior, ahora se ordenan los datos por identificador de regante descendientemente, mostrando únicamente sus nombres (sin las etiquetas):

```
1 for $reg in db:open("xmlregantes")/datos/regantes
2 where $reg/id < 3
3 order by $reg/id descending
4 return data($reg/nombre)
```

6.2.2. Expresión “Let”

En el ejemplo anterior el “let” ya estaba implícito en el “for”. Esto es, al plantear el bucle “for” automáticamente se asignan valores a “\$reg”, aunque se supone que esa asignación es justamente lo que realiza “let”.

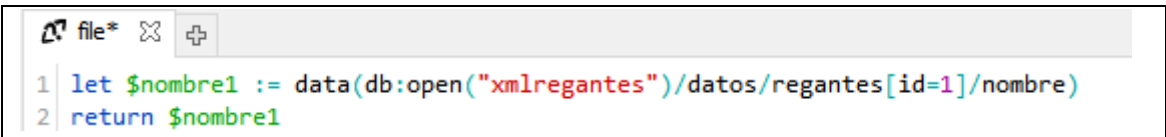
No obstante, podemos usar la asignación con “let” en cualquier otro momento, según nuestra conveniencia. Vamos a retocar un poco nuestro ejemplo para almacenar en una variable “\$nombre” el nombre a mostrar (con *return*):



```
1 for $reg in db:open("xmlregantes")/datos/regantes
2 let $nombre := data($reg/nombre)
3 where $reg/id<3
4 order by $reg/id descending
5 return $nombre
6
```

La línea 2 en realidad podría haberse escrito tras el “where” o tras el “order by”. No hay limitación a este respecto.

Tampoco hay que perder de vista que “let” puede usarse fuera del ámbito de un “for”. En el siguiente ejemplo se consulta el nombre del regante con id 1:



```
1 let $nombre1 := data(db:open("xmlregantes")/datos/regantes[id=1]/nombre)
2 return $nombre1
```

6.3. Funciones

6.3.1. data()

Según hemos visto en los ejemplos anteriores, el cometido de *data()* es extraer estrictamente el valor de la expresión que se pasa como argumento: Esta función puede aplicarse:

- A elementos, descartando las etiquetas de apertura y clausura que rodean al contenido. Este es el caso del ejemplo anterior.
- A atributos, descartando su nombre, el signo “=” y los calificadores de texto (comillas).

Para ver un ejemplo de uso de `data()` sobre un atributo, veamos esta sencilla forma de listar los valores del atributo `tabla` de los elementos raíces etiquetados como “datos”. A la izquierda el código XQuery y a su derecha el resultado obtenido:

```

1 for $reg in db:open("xmlregantes")/datos
2 return data($reg/@tabla)
3

```

Resultados:

- parcelas
- regantes
- votos

Nótese que en XQuery también hay que referirse a los atributos precediendo su nombre con el carácter de la arroba “@”.

6.3.2. Funciones con cadenas

6.3.2.1. `concat()`

Concatena dos o más textos, por ejemplo:

```

1 for $reg in db:open('xmlregantes')/datos/parcelas
2 let $datos_parcela := concat($reg/id, ' - ', $reg/poligono, ' ', $reg/parcela)
3 return $datos_parcela

```

Resultados:

- 1.- 137.143
- 2.- 137.144
- 3.- 137.145
- 4.- 137.146
- 5.- 137.147
- 6.- 137.143

6.3.2.2. `substring()`

Corta una subcadena del valor que se pasa como primer argumento, desde la posición indicada como segundo argumento (1 es el primer carácter) e incluyendo tantos caracteres como se pase en el tercer argumento:

```

1 for $reg in db:open('xmlregantes')/datos/regantes
2 let $nombre_abreviado := substring($reg/nombre,1,3)
3 return $nombre_abreviado

```

Resultados:

- Mar
- Seb
- Ama

6.3.2.3. `upper-case()`

Convierte a mayúsculas el texto pasado como único argumento. El ejemplo anterior, anidando la función “substring” dentro de “upper-case”, quedaría:

```

1 for $reg in db:open('xmlregantes')/datos/regantes
2 let $nombre_MAY := upper-case(substring($reg/nombre,1,3))
3 return $nombre_MAY

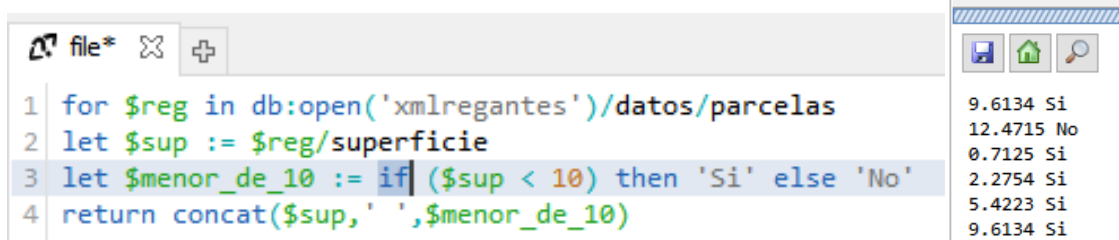
```

Resultados:

- MAR
- SEB
- AMA

6.3.3. Estructura de control condicional

Es posible usar el condicional “if” para evaluar una expresión booleana y devolver un valor u otro dependiendo de si es verdadera o falsa. Podemos ver un ejemplo en el que se listan las superficies de las fincas y un texto “Si” o “No” según dicha superficie sea inferior a 10Ha o no:



```
1 for $reg in db:open('xmlregantes')/datos/parcelas
2 let $sup := $reg/superficie
3 let $menor_de_10 := if ($sup < 10) then 'Si' else 'No'
4 return concat($sup, ' ', $menor_de_10)
```

9.6134	Si
12.4715	No
0.7125	Si
2.2754	Si
5.4223	Si
9.6134	Si

6.3.4. Funciones “de agregado”

He llamado así al apartado por analogía con dichas funciones de SQL. Me refiero a las funciones count(), min(), max(), avg() y sum().

Ya hemos visto que muchas de las expresiones pueden devolver múltiples valores. Estas funciones tienen la capacidad de resumir en un único valor todos los originales devueltos. La forma de calcular ese resumen depende de cada una.

Dos de ellas sólo tienen sentido para valores numéricos: *avg()*, que calcula la media aritmética de los valores devueltos y *sum()*, que calcula el sumatorio de los mismos.

Vamos a ver un ejemplo en el que sumamos la superficie de todas las parcelas:



```
1 let $superficie_total := sum(db:open("xmlregantes")/datos/parcelas/superficie)
2 return $superficie_total
```

6.4. Combinar resultados con HTML

Una posibilidad ofrecida por los intérpretes de XQuery es la de combinar los resultados devueltos con otras etiquetas. Por ejemplo, podríamos formar una lista HTML de ítems con los resultados anteriores, ejecutando esta consulta:

```
1 <ul>
2 {
3   for $reg in db:open("xmlregantes")/datos/regantes
4   order by $reg/id
5   return <li>{data($reg/nombre)}</li>
6 }
7 </ul>
```

Nótese el uso de "{" y "}" como delimitadores de expresión que ha de reemplazarse por su valor evaluado. Encontramos dos emparejamientos:

- Uno interior, flanqueando la función data(), para que el valor devuelto se una a .
- Una exterior, para que los valores devueltos por el "for" constituyan la lista.