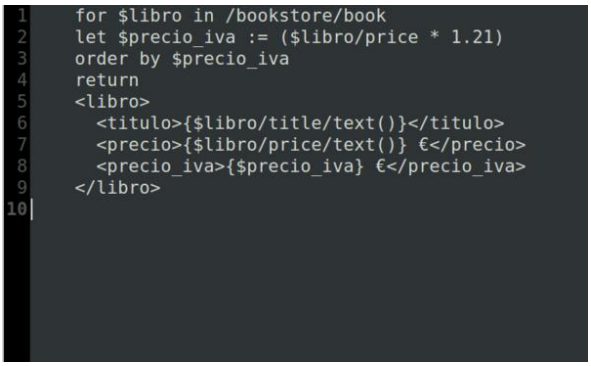
QUERY:

TEORÍA Y EJERCICIO RESUELTO Y EXPLICADO



```
for $libro in /bookstore/book
let $precio_iva := ($libro/price * 1.21)
order by $precio_iva
return
<libro>
  <titulo>{$libro/title/text()}</titulo>
  <precio>{$libro/price/text()} €</precio>
  <precio_iva>{$precio_iva} €</precio_iva>
</libro>
```

Qué es el lenguaje XQuery

XQuery es un **lenguaje de consulta** que permite extraer información de bases de datos o documentos XML. Se puede decir que XQuery es a XML lo mismo que SQL a las bases de datos relacionales.

XQuery se basa en el lenguaje XPath para el acceso a los nodos XML, pudiendo utilizar todos sus operadores y funciones.

EJERCICIO 1

Para todos los **ejemplos** siguiente vamos a tomar como referencia el siguiente documento XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<bib>
  <book id="1">
    <title>TCP/IP Illustrated</title>
    <author>Stevens</author>
    <publisher>Addison-Wesley</publisher>
    <year>2002</year>
  </book>
  <book id="2">
    <title>Advanced Programming in the Unix Environment</title>
    <author>Stevens</author>
    <publisher>Addison-Wesley</publisher>
    <year>2004</year>
  </book>
  <book id="3">
    <title>Data on the Web</title>
    <author>Abiteboul</author>
    <author>Buneman</author>
    <author>Suciu</author>
    <year>2006</year>
  </book>
</bib>
```

Software y herramientas online

Para probar XQuery podemos utilizar el **software** libre BaseX o cualquier otro que comentamos aquí.

Consultas FLWOR

Las consultas XQuery se componen de cinco cláusulas, que debido a sus iniciales se las conoce como FLWOR. Definimos cada una de ellas:

- **FOR**: Indica qué nodos se van a seleccionar desde la base de datos XML o desde un documento XML.
- **LET**: Permite declarar variables a las que se le asignan valores.
- **WHERE**: Permite introducir condiciones que deben cumplir los nodos seleccionados por la cláusula "for".
- **ORDER BY**: Permite ordenar los nodos antes de su visualización.
- **RETURN**: Devuelve los resultados. Es la única cláusula obligatoria.

Cláusula "for" y "return"

Con la cláusula "for" recuperaremos una serie de nodos mediante una consulta XPath y los introduciremos en una variable para poder utilizarla en la cláusula "return". Hay que señalar que la cláusula "return" se ejecutará una vez por cada nodo que devuelva la cláusula "for".

```
for $book in /bib/book
return $book/title
```

En este caso el nodo "title" se imprime junto con las etiquetas:

```
<title>TCP/IP Illustrated</title>
<title>Advanced Programming in the Unix Environment</title>
<title>Data on the Web</title>
```

Como no hemos indicado ningún documento tras "in" la consulta se lanzará contra la base de datos que tengamos abierta en nuestro programa. El resto de ejemplos del manual se realizarán de esta manera, pero si quieremos lanzar la consulta contra un documento XML que no es una base de datos podemos hacerlo usando "doc":

```
for $book in doc("bib.xml")/bib/book
return $book/title
```

Si queremos imprimir nuestras propias etiquetas en la cláusula "return", tendremos que encerrar la variable entre llaves { }:

```
for $book in /bib/book
return <titulo>{$book/title/text()}</titulo>
```

Obteniendo como resultado:

```
<titulo>TCP/IP Illustrated</titulo>
<titulo>Advanced Programming in the Unix Environment</titulo>
<titulo>Data on the Web</titulo>
```

Podemos utilizar "at" dentro de la cláusula "for" para obtener una variable con la numeración de los nodos que se van a recorrer:

```
for $book at $i in /bib/book
return <titulo>({$i}) {$book/title/text()}</titulo>
```

Lo hemos utilizado para incluirlo dentro de la etiqueta "titulo":

```
<titulo>(1) TCP/IP Illustrated</titulo>
<titulo>(2) Advanced Programming in the Unix Environment</titulo>
<titulo>(3) Data on the Web</titulo>
```

Si quisiéramos englobar todas las etiquetas anteriores en una superior, tendríamos que encerrar la consulta completa entre llaves { } como vemos en este ejemplo:

```
<biblioteca>
{
  for $book in /bib/book
  return <titulo>{$book/title/text()}</titulo>
}
</biblioteca>
```

Obteniendo la salida:

```
<biblioteca>
  <titulo>TCP/IP Illustrated</titulo>
  <titulo>Advanced Programming in the Unix Environment</titulo>
  <titulo>Data on the Web</titulo>
</biblioteca>
```

Cláusula "let"

La cláusula "let" nos va a permitir crear variables con cierto contenido. La diferencia con "for" es que ésta sólo se ejecutaría una sola vez con la cláusula "return". La cláusula "let" asigna las variables mediante los caracteres ":=". Si el ejemplo anterior lo realizáramos con "let":

```
let $book := /bib/book
return <titulo>{$book/title}</titulo>
```

Podemos observar como la etiqueta "titulo" sólo aparece una vez, es decir, no se repite para cada nodo como en el caso de la cláusula "for".

```
<titulo>
  <title>TCP/IP Illustrated</title>
  <title>Advanced Programming in the Unix Environment</title>
  <title>Data on the Web</title>
</titulo>
```

La clasula "let" nos va a permitir utilizar funciones de agrupación, como calcular la media, la suma, contar, etc. Estas son las mismas funciones que las que se utilizan en el leguaje XPath y que podéis repasar aquí. Podemos por ejemplo buscar el año más alto que exista mediante la función "max" para ver el último libro que se ha escrito:

```
let $book := /bib/book
return <last_year>{max($book/year)}</last_year>
```

Y la salida sería:

```
<last_year>2006</last_year>
```

Cláusula "for" y "let"

Podemos combinar las cláusulas "for" y "let". De esta manera conseguimos que la clasula "let" se ejecute una vez por cada nodo, al igual que hace la clasula "return". Por ejemplo, si queremos contar el número de autores que tiene cada libro podemos utilizar la siguiente consulta:

```
for $book in /bib/book
let $autores := count($book/author)
return
  <libro>
    <titulo>{$book/title/text()}</titulo>
    <autores>{$autores}</autores>
  </libro>
```

Lo que conseguimos es que para cada nodo que pasa por la cláusula "for" utilicemos "let" para incluir en la variable "$autores" la cuenta de nodos "author" de dicho libro que tenemos en la variable "$book", consiguiendo el siguiente resultado:

```
<libro>
  <titulo>TCP/IP Illustrated</titulo>
  <autores>1</autores>
</libro>
<libro>
  <titulo>Advanced Programming in the Unix Environment</titulo>
  <autores>1</autores>
```

```
</libro>
<libro>
  <titulo>Data on the Web</titulo>
  <autores>3</autores>
</libro>
```

Aunque este mismo caso también lo podríamos realizar sin utilizar la clásula "let":

```
for $book in /bib/book
return
  <libro>
    <titulo>{$book/title/text()}</titulo>
    <autores>{count($book/author)}</autores>
  </libro>
```

Cláusula "where"

Con la cláusula "where" podemos filtrar los nodos que se seleccionan en la cláusula "for", para ello también podemos utilizar los mismos operadores y funciones que en el leguaje XPath y que podéis repasar aquí. MUY IMPORTANTE, la cláusula "where" NO filtraría los nodos si los estamos obteniedo con "let". Por ejemplo podemos buscar los títulos de un determinado autor:

```
for $book in /bib/book
where $book/author = "Stevens"
return $book/title
```

Y el resultado sería:

```
<title>TCP/IP Illustrated</title>
<title>Advanced Programming in the Unix Environment</title>
```

La misma consulta anterior se podría realizar de igual manera filtrando los nodos en la consulta XPath sin tener que utilizar la cláusula "where":

```
for $book in /bib/book[author = "Stevens"]
return $book/title
```

Un ejemplo más con la cláusula "where" utilizando una función:

```
for $book in /bib/book
where starts-with($book/author, "S")
return $book/title
```

Cláusula "order by"

Con la cláusula "order by" podemos ordenar los nodos antes de que empiece a ejecutarse la cláusula "return", ya que como sabemos, la salida será la misma que el orden que tengan los nodos en el documento o base de datos XML:

```
for $book in /bib/book
order by $book/title
return $book/title
```

Obteniendo la salida:

```
<title>Advanced Programming in the Unix Environment</title>
<title>Data on the Web</title>
<title>TCP/IP Illustrated</title>
```

Otros ejemplos de funciones y operadores

La función "distinct-values" nos permite dentro de la cláusula "for" o "let" seleccionar sólo los nodos que tengan valores diferentes. Pero hay que tener en cuenta que los devuelve sin sus etiquetas, como si usáramos "/text()":

```
for $author in distinct-values(/bib/book/author)
where $author
return $author
```

El operador "except" nos permite eliminar nodos de la salida de la consulta, pero para ello es obligatorio utilizar "/*" en el nodo donde vayamos a utilizarlo como vemos en el ejemplo. En este caso también se devuelve el libro sin etiquetas, de ahí que le hayamos incluido nosotros nuestras propias etiquetas en el "return":

```
for $book in /bib/book
return <libro>{$book/* except $book/year except $book/author}</libro>
```