Unidad 6 ALMACENAMIENTO DE INFORMACIÓN

Xquery.

De manera rápida podemos definir XQuery con un símil en el que XQuery es a XML lo mismo que SQL es a las bases de datos relacionales.
XQuery es un lenguaje de consulta diseñado para escribir consultas sobre colecciones de datos expresadas en XML. Abarca desde archivos XML hasta bases de datos relacionales con funciones de conversión de registros a XML. Su principal función es extraer información de un conjunto de datos organizados como un árbol nário de etiquetas XML. En este sentido XQuery es independiente del origen de los datos.
XQuery es un lenguaje funcional, lo que significa que, en vez de ejecutar una lista de comandos como un lenguaje procedimental clásico, cada consulta es una expresión que es evaluada y devuelve un resultado, al igual que en SQL. Diversas expresiones pueden combinarse de una manera muy flexible con otras expresiones para crear nuevas expresiones más complejas y de mayor potencia semántica. XQuery está llamado a ser el futuro estándar de consultas sobre documentos XML.

Consultas en XQuery.

Una consulta en Xquery es una expresión que lee una secuencia de datos en XML y devuelve como resultado otra secuencia de datos en XML.
Un detalle importante es que, a diferencia de lo que sucede en SQL, en Xquery las expresiones y los valores que devuelven son dependientes del contexto. Por ejemplo los nodos que aparecerán en el resultado dependen de los namespaces, de la posición donde aparezca la etiqueta raíz del nodo (dentro de otra, por ejemplo), etc.

En XQuery las consultas pueden estar compuestas por cláusulas de hasta cinco tipos distintos. Las consultas siguen la norma FLWOR (leído como flower), siendo FLWOR las siglas de For, Let, Where, Order y Return.

XQuery emplea **funciones** para extraer datos de los documentos XML. Por ejemplo, l a función doc() se usa para abrir un fichero XML: doc("books.xml")

Expresiones FLWOR

Documento: “biblio.xml”

```
<bookstore>

<book category="COOKING">
  <title lang="en">Everyday Italian</title>
  <author>Giada De Laurentiis</author>
  <year>2005</year>
  <price>30.00</price>
</book>

<book category="CHILDREN">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>

<book category="WEB">
  <title lang="en">XQuery Kick Start</title>
  <author>James McGovern</author>
  <author>Per Bothner</author>
  <author>Kurt Cagle</author>
  <author>James Linn</author>
  <author>Vaidyanathan Nagarajan</author>
  <year>2003</year>
  <price>49.99</price>
</book>

<book category="WEB">
  <title lang="en">Learning XML</title>
  <author>Erik T. Ray</author>
  <year>2003</year>
  <price>39.95</price>
</book>

</bookstore>
```

For
Vincula una o más variables a expresiones escritas en Xpath, creando un flujo de tuplas en el que cada tupla está vinculada a una de las variable.

```
for $x in doc("biblio.xml")/bookstore/book
return $x/title
```

Resultado:

Devuelve los títulos de los libros.

```
for $x at $i in doc("books.xml")/bookstore/book/title
return <book>{$i}. {data($x)}</book>
```

La palabra reservada **at** se usa para almacenar el número de iteración.

Resultado:

```
<book>1. Everyday Italian</book>
<book>2. Harry Potter</book>
<book>3. XQuery Kick Start</book>
<book>4. Learning XML</book>
```

Let
Vincula una variable al resultado completo de una expresión añadiendo esos vínculos a las tuplas generadas por una cláusula for o, si no existe ninguna cláusula for, creando una única tupla que contenga esos vínculos.

```
let $x := doc("biblio.xml")/bookstore/book
return $x/title
```

Where
Filtra las tuplas eliminando todos los valores que no cumplan las condiciones dadas.

```
for $x in doc("biblio.xml")/bookstore/book
where $x/price>30 and $x/price<100
return <titulos>{data($x/title)}</titulos>
```

Order by..... (ascending- descending)
Ordena las tuplas según el criterio dado.

```
for $x in doc("biblio.xml")/bookstore/book
order by  $x/title
return $x/title
```

Return
Construye el resultado de la consulta para una tupla dada, después de haber sido filtrada por la cláusula where y ordenada por la cláusula order by.

```
let $x := doc("biblio.xml")/bookstore/book
return $x/title
```

En XQuery, cuando usamos el térmico tupla, nos estamos refiriendo a cada uno de los valores que toma una variable.

Expresiones condicionales.

Además de la cláusula where, XQuery también soporta expresiones condicionales del tipo “if-then-else” con la misma semántica que en los lenguajes de programación más habituales (C, Java, Delphi, etc..). Por ejemplo, la siguiente consulta
devuelve los títulos de todos los libros clasificados por categorías.

```
for $x in doc("books.xml")/bookstore/book
return if ($x/@category="CHILDREN")
then <child>{data($x/title)}</child>
else <adult>{data($x/title)}</adult>
```

La diferencia de la mayoría de los lenguajes, la cláusula else es obligatoria y debe aparecer siempre en la expresión condicional. El motivo de esto es que toda expresión en XQuery debe devolver un valor. Si no existe ningún valor a devolver al no cumplirse la cláusula if, devolvemos una secuencia vacía con ‘else ()’.

Operadores y funciones principales.

El conjunto de funciones y operadores soportado por XQuery 1.0 es el mismo conjunto de funciones y operadores utilizado en XPath 2.0 y XSLT 2.0.
XQuery soporta operadores y funciones matemáticas, de cadenas, para el tratamiento de expresiones regulares, comparaciones de fechas y horas, manipulación de nodos XML, manipulación de secuencias, comprobación y conversión de tipos y lógica booleana. Además permite definir funciones propias y funciones dependientes del entorno de ejecución del motor XQuery. Los operadores y funciones más importantes son:

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(), startswith(), ends-with(), substring(), upper-case(), lower-case(), string()

Uso general: distinct-values(), empty(), exits()

(*) La división se indica con el operador 'div' ya que el símbolo '/' es necesario para indicar caminos. El operador 'idiv' es para divisiones con enteros en las que se ignora el resto.

Cuantificadores existenciales:

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

Por ejemplo, seleccionar los títulos de los libros en los que al menos uno de sus autores es "James Linn".

```
for $b in doc("libros.xml")//lbook
where some $a in $b/author
satisfies ($a="James Linn")
return $b/titulo
```

XQuery e HTML

La siguiente expresión FLWOR de XQuery seleccionará todos los elementos title bajo elementos book que estén bajo elementos bookstore y devolverá los títulos ordenados alfabeticamente:

```
for $x in doc("books.xml")/bookstore/book/title
order by $x
return $x
```

Si quisiéramos representarlos mediante una lista en HTML podríamos añadir las etiquetas <ul> y <li> tal e como se muestra en el siguiente ejemplo:

```
<ul>
{
for $x in doc("books.xml")/bookstore/book/title
order by $x
return <li>{$x}</li>
}
</ul>
```

El resultado será el siguiente:

```
<ul>
<li><title lang="en">Everyday Italian</title></li>
<li><title lang="en">Harry Potter</title></li>
<li><title lang="en">Learning XML</title></li>
<li><title lang="en">XQuery Kick Start</title></li>
</ul>
```