XQUERY

1. Introducción

XQuery, también conocido como XML Query, es un lenguaje creado para buscar y extraer elementos y atributos de documentos XML. La mejor forma de entender este lenguaje es diciendo que XQuery es para XML lo que SQL es para las bases de datos. 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 n-ario de etiquetas XML. En este sentido XQuery es independiente del origen de los datos.

Una consulta XQuery podría resolver, por ejemplo, la siguiente pregunta: "Seleccionar todos los libros con un precio menor a 20 euros de la colección de libros almacenada en un documento XML llamado catalogo.xml".

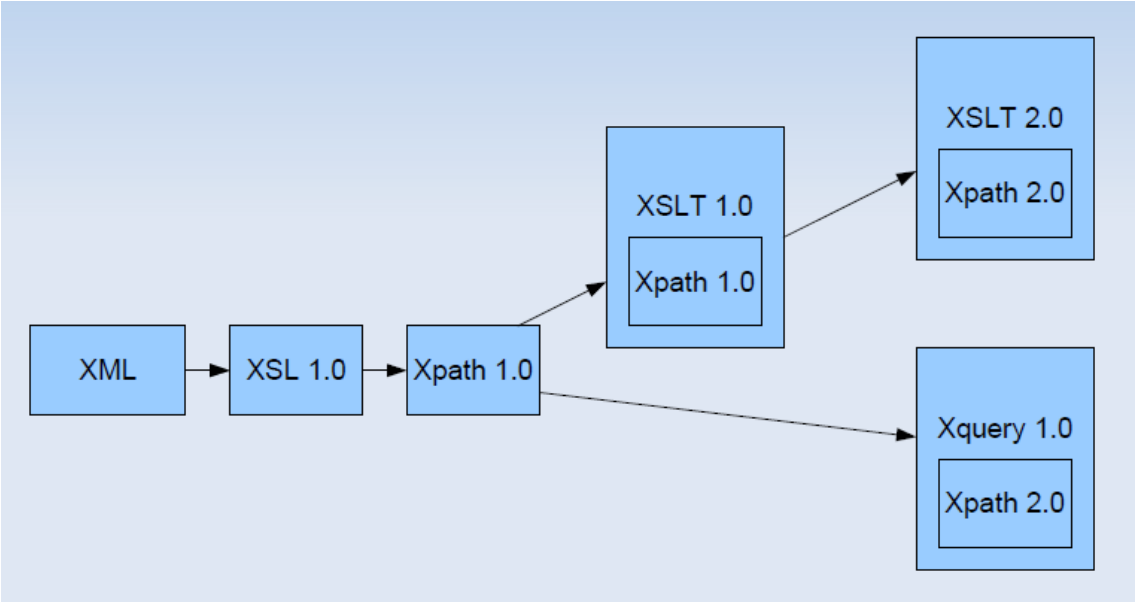
XQuery es un lenguaje funcional, lo que significa que en vez de ejecutar una lista de comandos como un lenguaje procedimental clásico, cada consulta es una expresión que es evaluada y devuelve un resultado, al igual que en SQL. Diversas expresiones pueden combinarse de una manera muy flexible con otras expresiones para crear nuevas expresiones más complejas y de mayor potencia semántica.

XQuery pretende ser el futuro estándar de consultas sobre documentos XML. Actualmente XQuery es un conjunto de borradores en el que trabaja el grupo W3C, por lo que es compatible con XML, Namespaces, XSLT, XPath y XML Schema. Sin embargo, a pesar de no tener una redacción definitiva ya existen o están en proceso numerosas implementaciones de motores y herramientas que lo soportan.

XQuery hace uso de XPath, un lenguaje utilizado para seleccionar partes de XML. De hecho, XQuery 1.0 y XPath 2.0 comparten el mismo modelo de datos y soportan las mismas funciones y operadores.

Principales funciones:

- SQL de XML
- Transformación XML
- Transformación a otros documentos HTML, PDF, etc.



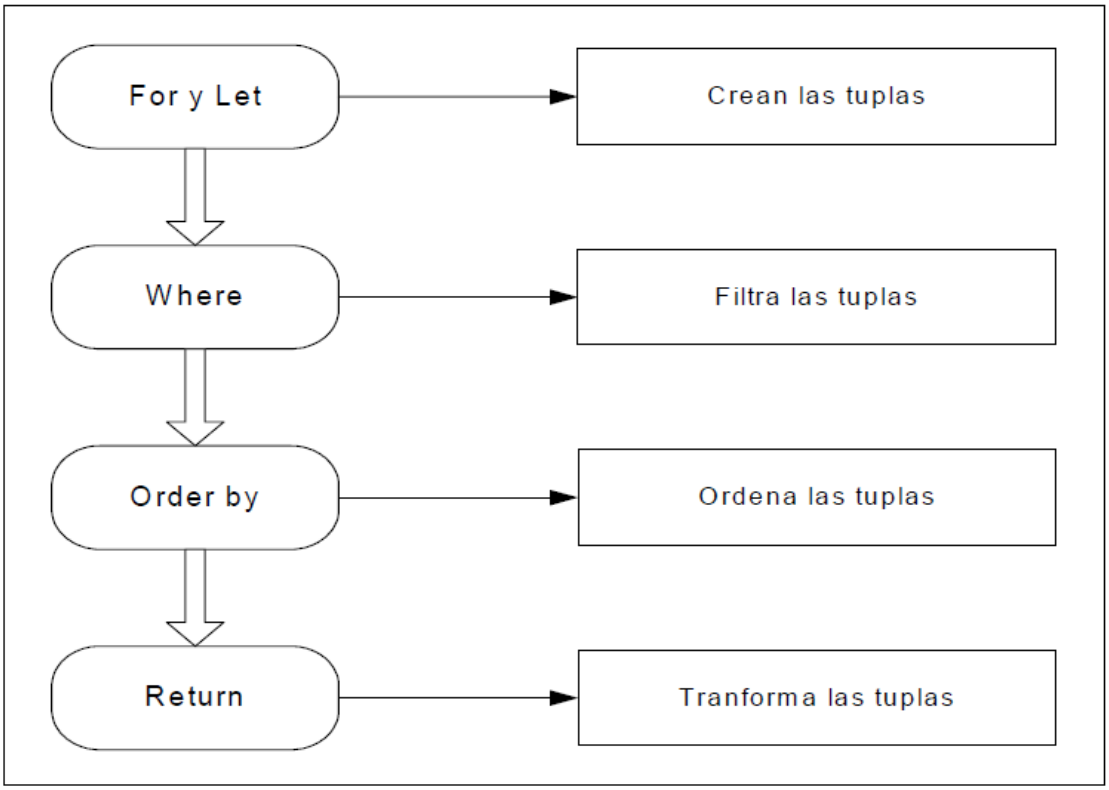
2. Consultas en XQuery.

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.

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 variables.
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.
Where	Filtra las tuplas eliminando todos los valores que no cumplan las condiciones dadas.
Order by	Ordena las tuplas según el criterio dado.
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.



- **FOR y LET**

Partimos del siguiente documento course.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<course>
	<lecture date="04/03/2004">
		<title>Introduction</title>
	</lecture>
	<lecture date="11/03/2004">
		<title>What is Telecooperation? Concepts</title>
	</lecture>
	<lecture date="18/03/2004">
		<title>Introduction XML</title>
	</lecture>
</course>
```

Usando **FOR**, cada nodo recuperado por la expresión, es vinculado a una única tupla.

```
for $b in doc("course.xml")//lecture/title
return <titulo>{$b}</titulo>
```

```
<titulo>
	<title>Introduction</title>
</titulo>
<titulo>
	<title>What is Telecooperation? Concepts</title>
</titulo>
<titulo>
	<title>Introduction XML</title>
</titulo>
```

Usando **LET** todos los nodos recuperados estarán en una única tupla.

```
let $b:= doc("course.xml")//lecture/title
return <titulo>{$b}</titulo>
```

```
<titulo>
       <title>Introduction</title>
       <title>What is Telecooperation? Concepts</title>
       <title>Introduction XML</title>
</titulo>
```

- **WHERE**

  - Filtra las tuplas producidas por las cláusulas let y for.
  - Contiene una expresión que es evaluada para cada tupla. Si su evaluación es false esa tupla es descartada

```
for $b in //lecture/title
where $b/../@date>"04/03/2004"
return
<titulos>{$b}</titulos>
```

```
<titulos>
       <title>What is Telecooperation? Concepts</title>
</titulos>
<titulos>
       <title>Introduction XML</title>
</titulos>
```

- **ORDER**

Indica, para un conjunto de información, un orden de salida. Por defecto organiza de manera ascendente según el tipo de datos que se esté evaluando.

Indica para un conjunto de información un orden de salida y va seguido de for.

- Por defecto de manera ascendente, si no es indica lo contrario, acorde con el tipo de datos que se esté evaluando.
- Ordenar según tipo de datos.
- Clausula "as" y el tipo "xs:integer".

```
for $b in //lecture/title
order by $b/../@date
return
<titulos>{$b}</titulos>
```

- **RETURN**

Formatea la información de salida. Esto nos ofrece muchas posibilidades, ya que podemos convertir la información recuperada a HTML, PDF o cualquier otro tipo.

3. EXPRESIONES CONDICIONALES: IF-THEN-ELSE

XQuery también admite expresiones condicionales IF-THEN-ELSE. Esta cláusula es útil para dar un formato diferente a la salida dependiendo de la información de esta, es decir, podemos cambiar la estructura de los nodos en los que recuperamos la información según nos convenga por su contenido.
Si es usa IF, es obligatorio el ELSE, pero puede ser únicamente ELSE ().

```
for $b in //lecture/title
order by $b/../@date
return
if ($b/../@date="04/03/2004")
then
<coincidencia>{$b}</coincidencia>
else
<titulos>{$b}</titulos>
```

```
<coincidencia>
<title>Introduction</title>
</coincidencia>
<titulos>
<title>What is Telecooperation? Concepts</title>
</titulos>
<titulos>
<title>Introduction XML</title>
</titulos>
```

4. CUANTIFICADORES EXISTENCIALES: EVERY Y SOME

EVERY: recupera aquellas tuplas en las que todos los nodos cumplan la condición.
SOME: recupera aquellas tuplas en las que algún nodo cumpla la condición.

5. OPERADORES Y FUNCIONES MATEMÁTICAS

Al igual que otros lenguajes, XQuery tiene una serie de funciones y operadores aritméticos para la recuperación de información

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()

6. COMENTARIOS

Los comentarios en XQuery, a diferencia de XML, van entre caras sonrientes " (: :) ".

7. EJEMPLOS

Partiendo de los siguientes documentos libros.xml y comentarios.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<bib>
      <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>Advan Programming for 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>
      <libro año="1999">
            <titulo> Economics of Technology for Digital TV</titulo>
            <editor>
                  <apellido>Gerbarg</apellido>
                  <nombre>Darcy</nombre>
                  <afiliacion>CITI</afiliacion>
            </editor>
            <editorial>Kluwer Academic editorials</editorial>
            <precio>129.95</precio>
</libro>
</bib>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<comentarios>
      <entrada>
            <titulo>Data on the Web</titulo>
            <precio>34.95</precio>
            <comentario>Un libro muy bueno sobre bases de
datos.</comentario>
      </entrada>
      <entrada>
            <titulo>Advanced Programming in the Unix
environment</titulo>
            <precio>65.95</precio>
            <comentario>Un libro claro y detallado de programación en
UNIX.</comentario>
      </entrada>
      <entrada>
            <titulo>TCP/IP Illustrated</titulo>
            <precio>65.95</precio>
            <comentario>Uno de los mejores libros de
TCP/IP</comentario>
      </entrada>
</comentarios>
```

Títulos de los libros que tengan más de dos autores ordenados por su título.

```
for $b in //libro
let $c := $b//autor
where count($c) > 2
order by $b/titulo
return $b/ titulo
```

Títulos de los libros del año 2.000.

```
for $b in //libro
where $b/@año = "2000"
return $b/titulo
```

Diferencias entre for y let

```
for $d in /bib/libro/titulo
return
<titulos>{ $d }</titulos>

let $d := /bib/libro/titulo
return
<titulos>{ $d }</titulos>
```

Título de cada uno de los libros de archivo "libros.xml" junto con el número de autores de cada libro.

```
for $b in //libro
let $c := $b/autor
return
<libro>{ $b/titulo, <autores>{ count($c)
}</autores>}</libro>
```

Títulos de todos los libros contenidos en el archivo “libros.xml” y todos los comentarios de cada libro contenidos en el archivo “comentarios.xml”.

```
for $t in //titulo, $e in //entrada
where $t = $e/titulo
return <comentario>{ $t, $e/comentario }</comentario>
```

Títulos de todos los libros almacenados en el archivo "libros.xml" y sus dos primeros autores. En el caso de que existan más de dos autores para un libro, se añade un tercer autor "et al.".

```
for $b in //libro
return
<libro>
{ $b/titulo }
{
for $a at $i in $b/autor
where $i <= 2
return <autor>{string($a/nombre), ", "
,string($a/apellido)}</autor>
}
{
if (count($b/autor) > 2)
then <autor>et al.</autor>
else ()
}
</libro>
```

Títulos de los libros en los que al menos uno de sus autores es W. Stevens. (SOME)

```
for $b in doc("libros.xml")//libro
where some $a in $b/autor
satisfies ($a/last="Stevens" and $a/first="W.")
return $b/titulo
```

Títulos de los libros en los que todos los autores de cada libro es W. Stevens.

```
for $b in //libro
where every $a in $b/autor
satisfies ($a/apellido="Stevens" and $a/nombre="W.")
return $b/titulo
```

El último título devuelto como resultado de la consulta es un libro que no tiene autores. Cuando un cuantificador universal se aplica sobre un nodo vacío, siempre devuelve cierto, como se ve en el ejemplo anterior.

Lista ordenada de apellidos de todos los autores y editores con operador de unión.

```
for $l in distinct-values(//(autor | editor)/apellido)
order by $l
return <apellidos>{ $l }</apellidos>
```

Consulta que usa el operador sustracción para obtener un nodo libro con todos sus nodos hijos salvo el nodo <precio>.

```
for $b in //libro
where $b/titulo = "TCP/IP Illustrated"
return
<libro>
{ $b/@* }
{ $b/* except $b/precio }
</libro>
```

La función empty() devuelve cierto cuando la expresión entre paréntesis está vacía. Por ejemplo, la siguiente consulta devuelve todos los nodos libro que tengan al menos un nodo autor.

```
for $b in //libro
where not(empty($b/autor))
return $b
```

Lo mismo se podría obtener de la siguiente manera

```
for $b in //libro
where exists($b/autor)
return $b
```