Unidad 10.

XML. Almacenamiento y recuperación de información.

RA6. Gestiona información en formato XML analizando y utilizando tecnologías de almacenamiento y lenguajes de consulta.

Criterios de evaluación:

a) Se han identificado los principales métodos de almacenamiento de la información usada en documentos XML.
b) Se han identificado los inconvenientes de almacenar información en formato XML.
c) Se han establecido tecnologías eficientes de almacenamiento de información en función de sus características.
d) Se han utilizado sistemas gestores de bases de datos relacionales en el almacenamiento de información en formato XML.
e) Se han utilizado técnicas específicas para crear documentos XML a partir de información almacenada en bases de datos relacionales.
f) Se han identificado las características de los sistemas gestores de bases de datos nativas XML.
g) Se han instalado y analizado sistemas gestores de bases de datos nativas XML.
h) Se han utilizado técnicas para gestionar la información almacenada en bases de datos nativas XML.
i) Se han identificado lenguajes y herramientas para el tratamiento y almacenamiento de información y su inclusión en documentos XML.

Contenido:

Bases de datos XML.
Xquery
Actividades.

Bases de datos XML.

Una base de datos XML constituye un sistema software que da persistencia a datos almacenados en formato XML. Estos datos pueden ser interrogados, exportados y serializados. Las bases de datos XML están generalmente asociadas con las bases de datos documentales.

Existen dos grandes clases de bases de datos XML:

- XML habilitado: Este tipo de base de datos puede mapear XML en estructuras tradicionales de bases de datos (como las relacionales), aceptando XML como entrada y formateando en XML la salida, o más recientemente soportando tipos XML nativos en la propia base de datos. Esto implica que la base de datos procesa el XML internamente.
- XML nativo (NXD): El modelo interno de estas bases de datos usa documentos XML como la unidad elemental de almacenamiento, los cuales no han de almacenarse necesariamente en formato de texto.

XQuery.

XQuery es un lenguaje de consultas para información almacenada en ficheros XML y bases de datos nativas XML. Similar a lo que es SQL para base de datos relacionales. Cuando hablamos de base de datos XML nos estamos refiriendo a una colección de documentos XML, por lo general organizados y estructurados siguiendo el estándar.

XQuery comparte el miso modelo de datos de XPath y admite las mismas funciones y operadores.

XQuery es una recomendación de W3C y compatibles con otros estándares XML: XSQLT, XPath, Esquemas, …

XQuery se puede utilizar para:

- Recuperar información de documentos XML, especialmente para usar en servicios web.
- Generar informes.
- Aplicar transformaciones a datos XML. Alternativa a XSLT.

Motores XQuery.

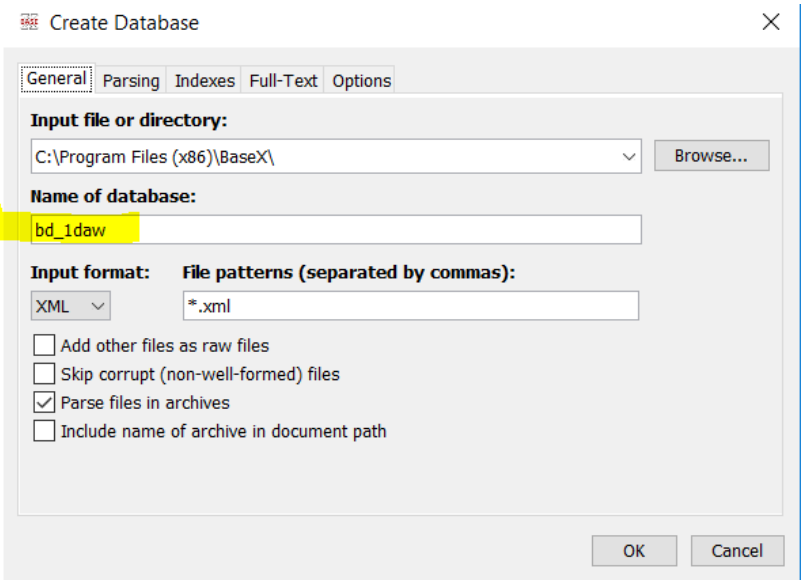
- Qexo.
- Saxon.
- Qizx/open.

Otras herramientas relacionadas con XQuery.

- Xquark Bridge.
- BumbleBee.

Actividad

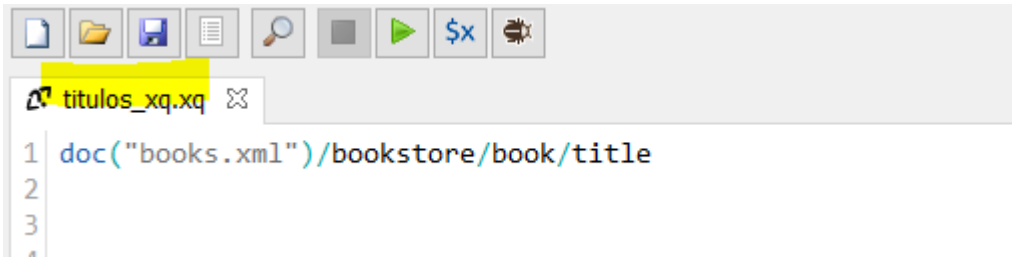
- Descargamos e instalamos una base de datos XML
- Creamos una base de datos.



- Añadimos el siguiente fichero XML

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
```

- Seleccionamos los títulos de los libros.



- Resultado obtenido.

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

Sintaxis básica.

Una consulta XQuery es una expresión que lee una secuencia de datos en XML, y devuelve como resultado otra secuencia de datos, por lo general XML.

Toda expresión XPath es una consulta XQuery válida.

XQuery distingue entre mayúsculas y minúsculas.

Los elementos, atributos y variables de XQuery deben ser nombres XML válidos.

Un valor de cadena XQuery puede delimitarse con comillas simples o dobles.

Un identificador de variable XQuery comienza con el carácter **$.** Por ejemplo, ***$var***

Los comentarios de XQuery están delimitados por **(: y :).** Por ejemplo, ***(: Comentario de XQuery :)***

En una expresión XQuery, los caracteres {} delimitan expresiones que son evaluadas para crear un nuevo documento.

XQuery admite expresiones condicionales similares a las estructuras de control selectivas (if-then-else) de los lenguajes de programación.

XQuery comparte los mismos tipos de datos que el esquema XML 1.0 (XSD).

XQuery utiliza el mismo conjunto de funciones y operadores de XPath y XSL.

XQuery usa predicados para limitar los datos extraídos de documentos XML.

Las consultas XQuery pueden estar formadas por hasta cinco tipos de cláusulas diferentes, cumpliendo la norma FLWOR, que veremos más adelante. Al menos ha de existir una cláusula FOR o una LET, el resto, si existen, han de aparecer en el orden establecido en el nombre de la norma.

Una consulta XQuery está formada por dos partes:

- Prólogo. En el se declaran los espacios de nombres, funciones, variables, etc.
- Expresión. Consulta XQuery

Ejemplo: Seleccionar los títulos de los libros que tienen un precio menor de 30.

```
doc("books.xml")/bookstore/book[price<30]/title
```

La función ***doc*** () se utiliza para abrir el archivo "books.xml":

FLWOR

Acrónimo de "For, Let, Where, Order by, Return".

For. Selecciona una secuencia de nodos.
Let. Une una secuencia a una variable.
Where. Condición para filtrar la salida.
Order by. Ordena los resultados.
Return. Salida. Es evaluada para cada nodo.

Ejemplo. Título de los libros con precio superior a 30$

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
return $x/title
```

Resultado:

```
<title lang="en">XQuery Kick Start</title>
<title lang="en">Learning XML</title>
```

Ordenamos por título.

```
for $x in doc("books.xml")/bookstore/book
where $x/price>30
order by $x/title
return $x/title
```

Resultado:

```
<title lang="en">Learning XML</title>
<title lang="en">XQuery Kick Start</title>
```

FLWOR permite transformar el resultado obtenido en la consulta generar diferentes formatos: XHTML, HTML, SVG, …

Ejemplo. Generar una lista HTML con todos los títulos.

Ahora queremos enumerar todos los títulos de libros en nuestra librería en una lista HTML. Añadimos etiquetas <ul> y <li> a la expresión FLWOR:

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

Resultado:

```
<ul>
  <li>Everyday Italian</li>
  <li>Harry Potter</li>
  <li>Learning XML</li>
  <li>XQuery Kick Start</li>
</ul>
```

Actividad: ¿Qué resultado obtendríamos tras ejecutar la siguiente consulta?

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

Cláusula ***for.***

Permite especificar el número de iteraciones del bucle utilizando la palabra reservada ***to***.

Ejemplo:

```
for $x in (1 to 5)
return $x
```

Resultado:

```
1
2
3
4
5
```

La palabra clave ***at*** se utiliza para cargar en una variable el número de ciclo en el que se encuentra el bucle.

Ejemplo:

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

Resultado:

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

La cláusula ***for*** permite varias expresiones separadas por comas.

Ejemplo:

```
for $x in (10,20), $y in (100,200)
return <test>x={$x} and y={$y}</test>
```

Resultado:

```
<test>x=10 and y=100</test>
<test>x=10 and y=200</test>
<test>x=20 and y=100</test>
<test>x=20 and y=200</test>
```

Cláusula ***let.***

Se utiliza para asignar valores a variables. No da lugar a iteración.

Ejemplo:

```
let $x := (1 to 5)
return <test>{$x}</test>
```

Resultado:

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

Cláusula ***where***.

Establece criterios de selección en el resultado.

Ejemplo:

```
for $x in doc("books.xml")/bookstore/book
where  $x/@category='CHILDREN'
return $x/title
```

Resultado:

```
<title lang="en">Harry Potter</title>
```

Cláusula ***order by.***

Se utiliza para especificar orden de clasificación del resultado.

Ejemplo:

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

Resultado:

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

Cláusula ***return***

Se utiliza para especificar el resultado devuelto por la consulta.

Ejemplo:

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

Resultado:

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

Expresiones condicionales

if-then-else es la estructura de control condicional utilizada por XQuery.

Ejemplo:

```
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>
```

Resultado:

```
<adult>Everyday Italian</adult>
<adult>Harry Potter</adult>
<adult>XQuery Kick Start</adult>
<adult>Learning XML</adult>
```

En las expresiones condicionales además de los operadores relacionales clásicos (=, !=,<,<=,>,>=) pueden utilizarse eq, ne, lt, le, gt, ge. Estos sólo pueden ser utilizados cuando la expresión devuelve un único valor.

Ejemplos:

$bookstore//book/@q > 10. Devuelve verdadero si alguno de los atributos q tiene un valor mayor que 10.

$bookstore//book/@q gt 10. Devuelve verdadero si solo hay un atributo q devuelto por la expresión, y su valor es mayor que 10. Si se devuelve más de un atributo se produce un error.

Funciones Xquery.

Xquery soporta un conjunto importante de funcione: Matemáticas, de cadenas, para el tratamiento de expresiones regulares, para manejo 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.

Algunas de las funciones más importantes son:

- Funciones numéricas
  - floor(). Devuelve el valor numérico inferior más próximo al dado.
  - Ceiling(). Devuelve el valor numérico superior más próximo al dado.
  - Round(). Redondea el valor dado al más próximo.
  - Count(). Determina el número de ítems en una colección.
  - min(), max(). Devuelven respectivamente el mínimo y el máximo de los valores de los nodos dados.
  - avg(). Calcula el valor medio de los valores dados.
  - sum(). Calcula la suma total de una cantidad de ítems dados.

- Funciones de cadenas de texto
  - concat(). Devuelve una cadena construida por la unión de dos cadenas dadas.
  - string-length(). Devuelve la cantidad de caracteres que forman una cadena.
  - startswith(), ends-with(). Determinan si una cadena dada comienza o termina, respectivamente, con otra cadena dada.
  - upper-case(), lower-case(). Devuelve la cadena dada en mayúsculas o minúsculas respectivamente.

- Funciones de uso general
  - empty(). Devuelve “true” cuando la secuencia dada no contiene ningún elemento.
  - exits(). Devuelve “true” cuando una secuencia contiene, al menos, un elemento.
  - distinct-values(). Extrae los valores de una secuencia de nodos y crea una nueva secuencia con valores únicos, eliminando los nodos duplicados.

- Cuantificadores existenciales:
  - some,every. Permiten definir consultas que devuelven algún, o todos los elementos, que verifiquen la condición dada.

Además del conjunto de funciones compartido con XPath 2.0 y XSLT 2.0, XQuery permite la definición de funciones de usuario.

```
declare function prefix:function_name($parameter as datatype)
as returnDatatype
{
 ...function code here...
};
```

Ejemplo: Precio de los libros incluyendo el iva.

```
declare function local:precTotal($p as xs:decimal)
as xs:decimal
{
let $importe:= $p * 1.21
return ($importe)
};

for $x in doc("books.xml")/bookstore//price
return local:precTotal($x)
```

Resultado:

```
36.3
36.2879
60.4879
48.3395
```

Actividades.

1 Documnto XML.

```
<?xml version="1.0" encoding="UTF-8"?>
<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>
```

Consultas XQuery.

1. Mostrar los títulos de los libros con la etiqueta "titulo".
2. Mostrar los libros cuyo precio sea menor o igual a 30. Primero incluyendo la condición en la cláusula "where" y luego en la ruta del XPath.
3. Mostrar sólo el título de los libros cuyo precio sea menor o igual a 30.
4. Mostrar sólo el título sin atributos de los libros cuyo precio sea menor o igual a 30.
5. Mostrar el título y el autor de los libros del año 2005, y etiquetar cada uno de ellos con "lib2005".
6. Mostrar los años de publicación, primero con "for" y luego con "let" para comprobar la diferencia entre ellos. Etiquetar la salida con "publicacion".
7. Mostrar los libros ordenados primero por "category" y luego por "title" en una sola consulta.
8. Mostrar cuántos libros hay, y etiquetarlo con "total".
9. Mostrar los títulos de los libros y al final una etiqueta con el número total de libros.
10. Mostrar el precio mínimo y máximo de los libros.
11. Mostrar el título del libro, su precio y su precio con el IVA incluido, cada uno con su propia etiqueta. Ordénalos por precio con IVA.
12. Mostrar la suma total de los precios de los libros con la etiqueta "total".
13. Mostrar cada uno de los precios de los libros, y al final una nueva etiqueta con la suma de los precios.

14. Mostrar el título y el número de autores que tiene cada título en etiquetas diferentes.
15. Mostrar en la misma etiqueta el título y entre paréntesis el número de autores que tiene ese título.
16. Mostrar los libros escritos en años que terminen en "3".
17. Mostrar los libros cuya categoría empiece por "C".
18. Mostrar los libros que tengan una "X" mayúscula o minúscula en el título ordenados de manera descendente.
19. Mostrar el título y el número de caracteres que tiene cada título, cada uno con su propia etiqueta.
20. Mostrar todos los años en los que se ha publicado un libro eliminando los repetidos. Etiquétalos con "año".
21. Mostrar todos los autores eliminando los que se repiten y ordenados por el número de caracteres que tiene cada autor.
22. Mostrar los títulos en una tabla de HTML.

2 Documento XML

```
<?xml version="1.0" encoding="UTF-8"?>
<bailes>
  <baile id="1">
    <nombre>Tango</nombre>
    <precio cuota="mensual" moneda="euro">27</precio>
    <plazas>20</plazas>
    <comienzo>1/1/2011</comienzo>
    <fin>1/12/2011</fin>
    <profesor>Roberto Garcia</profesor>
    <sala>1</sala>
  </baile>
  <baile id="2">
    <nombre>Cha-cha-cha</nombre>
    <precio cuota="trimestral" moneda="euro">80</precio>
    <plazas>18</plazas>
    <comienzo>1/2/2011</comienzo>
    <fin>31/7/2011</fin>
    <profesor>Miriam Gutierrez</profesor>
    <sala>1</sala>
  </baile>
  <baile id="3">
    <nombre>Rock</nombre>
    <precio cuota="mensual" moneda="euro">30</precio>
    <plazas>15</plazas>
    <comienzo>1/3/2011</comienzo>
    <fin>1/12/2011</fin>
    <profesor>Laura Mendiola</profesor>
    <sala>1</sala>
  </baile>
  <baile id="4">
    <nombre>Merengue</nombre>
    <precio cuota="trimestral" moneda="dolares">75</precio>
    <plazas>12</plazas>
    <comienzo>1/1/2011</comienzo>
    <fin>1/12/2011</fin>
    <profesor>Jesus Lozano</profesor>
    <sala>2</sala>
  </baile>
  <baile id="5">
    <nombre>Salsa</nombre>
    <precio cuota="mensual" moneda="euro">32</precio>
    <plazas>10</plazas>
    <comienzo>1/5/2011</comienzo>
    <fin>1/12/2011</fin>
    <profesor>Jesus Lozano</profesor>
    <sala>2</sala>
  </baile>
  <baile id="6">
```

```
    <nombre>Pasodoble</nombre>
    <precio cuota="anual" moneda="euro">320</precio>
    <plazas>8</plazas>
    <comienzo>1/1/2011</comienzo>
    <fin>31/12/2011</fin>
    <profesor>Miriam Gutierrez</profesor>
    <sala>1</sala>
  </baile>
</bailes>
```

Consultas XQuery.

1. Mostrar cada uno de los nombres de los bailes con la etiqueta "losbailes".
2. Mostrar los nombres de los bailes seguidos con el número de plazas entre paréntesis, ambos dentro de la misma etiqueta "losbailes".
3. Mostrar los nombres de los bailes cuyo precio sea mayor de 30.
4. Mostrar los nombres de los bailes cuyo precio sea mayor de 30 y la moneda "euro".
5. Mostrar los nombres y la fecha de comienzo de los bailes que comiencen el mes de enero (utiliza para buscarlo la cadena de texto "/1/").
6. Mostrar los nombres de los profesores y la sala en la que dan clase, ordénalos por sala.
7. Mostrar los nombres de los profesores eliminando los repetidos y acampañar cada nombre con todas las salas en la que da clase, ordénalos por nombre.
8. Mostrar la media de los precios de todos los bailes.
9. Mostrar la suma de los precios de los bailes de la sala 1.
10. Mostrar cuántas plazas en total oferta el profesor "Jesus Lozano".
11. Mostrar el dinero que ganaría la profesora "Laura Mendiola" si se completaran todas las plazas de su baile, sabiendo que sólo tiene un baile.
12. Mostrar el dinero que ganaría el profesor "Jesus Lozano" si se completaran todas las plazas de su baile, pero mostrando el beneficio de cada baile por separado.
13. Mostrar el dinero que ganaría la profesora "Laura" (no conocemos su apellido) si se completaran todas las plazas de su baile.
14. Mostrar el nombre del baile, su precio y el precio con un descuento del 15% para familias numerosas. Ordenar por el nombre del baile
15. Mostrar todos los datos de cada baile excepto la fecha de comienzo y de fin.
16. Mostrar en una tabla de HTML los nombres de los bailes y su profesor, cada uno en una fila.