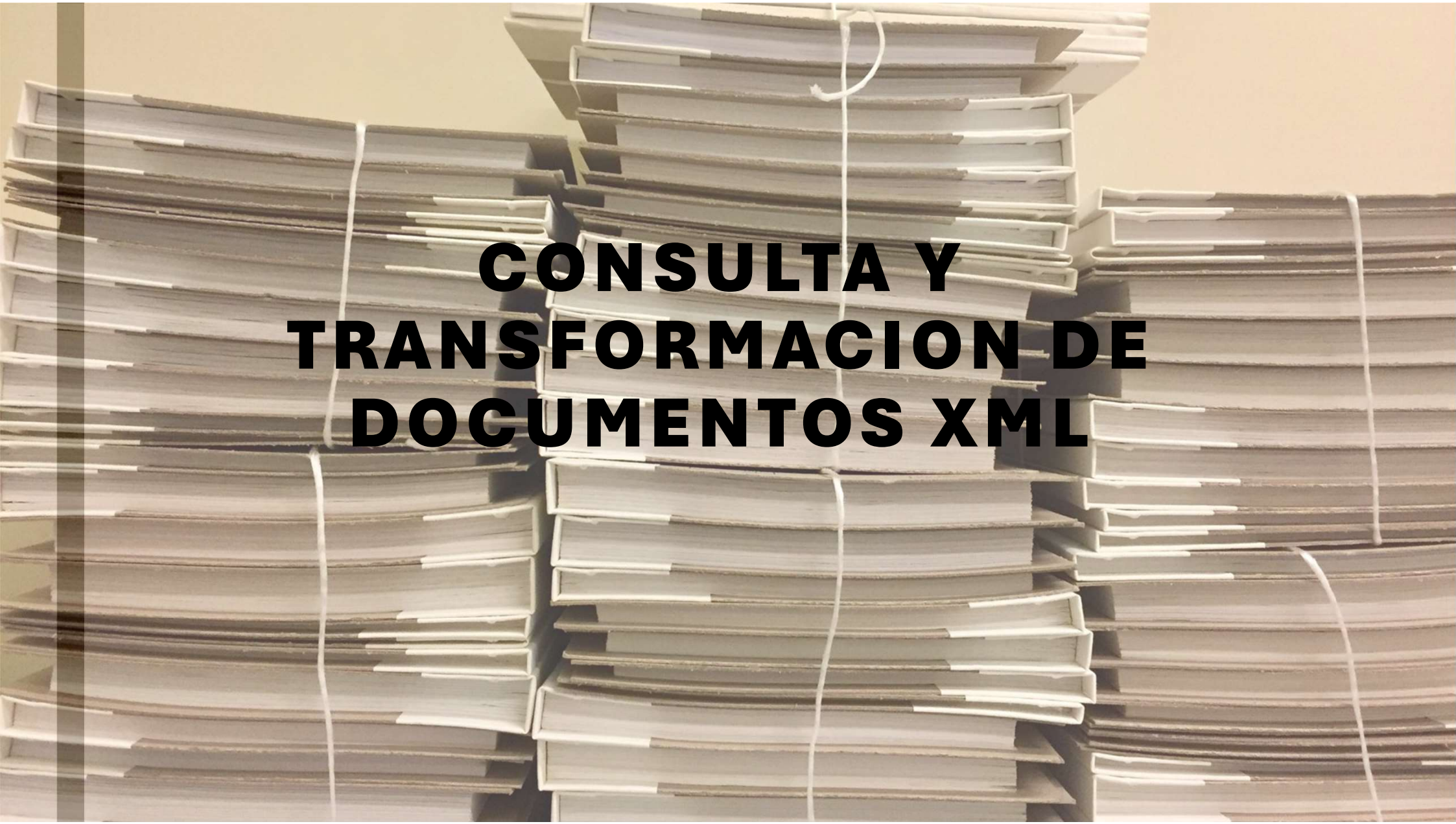


CONSULTA Y TRANSFORMACION DE DOCUMENTOS XML

1.- Sistemas de almacenamiento de ficheros XML

Los datos contenidos en un documento XML no siempre se analizan en el momento. Normalmente se quiere guardar la información en un sistema de almacenamiento persistente para poder recuperarla y procesarla luego.

Cuando se usa el formato XML solo como “vehículo” para portar la información no se hace necesario almacenar los ficheros porque no hay necesidad de persistencia de esos datos. El XML está cargado en la memoria del programa. Lo que sí hay que hacer en estos casos es convertir de XML a otros formatos (ej. desde una aplicación se genera un xml que pasa a otra aplicación que convierte ese XML para obtener los datos)

En los casos en los que hay que almacenar los documentos XML se puede hacer:

- ✓ Almacenamiento en ficheros, es el mecanismo más simple. Ej. fichero de configuración de arranque de un router.
- ✓ En una BBDD relacional, almacenando los ficheros en tablas
- ✓ En BBDD Nativas: son bases de datos orientadas al almacenamiento de documentos XML

La decisión del almacenamiento de los documentos XML va a depender del uso que se tenga que hacer de ellos, cada sistema tiene sus ventajas e inconvenientes

Solución	Dificultad	Proporciona independencia de otras tecnologías	Permite operaciones directas
Ficheros	Baja	Sí	No
Bases de datos	Media	Opcionalmente	Opcionalmente
Bases de datos XML	Media	No	Sí

En una base de datos **relacional**:

- Tablas.
- Registros/tuplas.
- Atributos.
- Relaciones.
- ...

En una base de datos **XML**:

- Elementos.
- Atributos.
- Comentarios.
- Espacios de nombres.
- ...

El almacenamiento en bbdd relacionales

Ventajas:

- Podemos gestionar grandes volúmenes de información de manera más sencilla que con ficheros.
- Las BB.DD. relacionales están implantadas en la mayoría de las empresas, no hay que cambiar el software.

Desventajas:

- Engorroso: con SQL solo podemos extraer el documento completo. No podemos extraer información concreta del XML.
- Las BB.DD. relacionales son mucho más complejas que XML.

El almacenamiento en bbdd Nativas

Ventajas:

- Acceso a la información contenida sin necesidad de leer todo el documento.
- Más simples que las relacionales

Desventajas:

- Hay que aprender nuevas técnicas de acceso, como el lenguaje Xpath o Xquery (“análogos” a SQL).
- Implica implementar nuevo software en nuestra empresa

1.1.- Almacenamiento de documentos XML en fichero

Se trata del mecanismo más sencillo, rápido y simple, pero es el que ofrece menos posibilidades a la hora de acceder a los datos que contiene, ya que la manipulación de la información que contiene hay que hacerla después de cargar ese XML en la memoria del programa que lo procesa.

Al tratarse de un fichero de texto, cualquier programa de programación puede acceder a ellos para leer el contenido como cadenas de caracteres de forma relativamente sencilla y una vez cargado procesar su información. Para ello se utilizan los analizadores de XML disponibles en la mayoría de los lenguajes de programación.

Los analizadores XML son programas o librerías diseñados para leer y procesar documentos XML. Su función principal es analizar la estructura del XML y extraer información de él, permitiendo a las aplicaciones acceder y manipular los datos contenidos en el XML. Hay diferentes tipos de analizadores XML, incluyendo analizadores como DOM y SAX

- **Analizadores DOM (Modelo de Objeto de Documento):**
  - Estos analizadores construyen una representación en memoria del documento XML en forma de un árbol de objetos, donde cada elemento, atributo y nodo de texto se convierte en un objeto en la memoria del programa.
  - Esto permite acceder a cualquier parte del documento XML de manera fácil y eficiente, ya que todo el documento está disponible en memoria.
  - Sin embargo, el DOM puede consumir mucha memoria, especialmente para documentos XML grandes, ya que carga todo el documento.

- **Analizadores SAX (Simple API for XML):**
  - Los analizadores SAX analizan el documento XML de manera secuencial y emiten eventos a medida que encuentran elementos, atributos y texto en el documento.
  - El usuario debe escribir un manejador de eventos que responda a estos eventos según sea necesario.
  - SAX es eficiente en términos de uso de memoria, ya que no carga el documento XML completo en memoria y es útil para procesar documentos XML grandes de manera eficiente.

Comparativa DOM y SAX

Característica	SAX	DOM
Tipo de API	Tipo «push», en «streaming»	Árbol en memoria
Facilidad de uso	Media	Alta
Admite XPath	No	Sí
Consumo de recursos	Bajo	Variable, en función del documento
Solo permite lectura hacia delante	Sí	No
Permite leer XML	Sí	Sí
Permite construir XML	No	Sí
Creación, lectura, modificación y borrado de elementos	No	Sí

1.2.- Almacenamiento de documentos XML en BBDD relacionales

En general, las BBDD relacionales no están diseñadas para este propósito, pero disponen de mecanismos suficientes para almacenar y recuperar documentos de cualquier tipo, incluidos los XML, además, algunos sistemas de gestión de bases de datos avanzadas proporcionan características específicas para su manejo.

Una solución en todas las BBDD convencionales es almacenar el documento XML en un campo de tipo texto, de esta manera se consigue la persistencia de la información del XML de forma similar al almacenamiento en fichero, añadiendo la facilidad que tienen las bases de datos para el manejo de grandes volúmenes de información.

En las BBDD que soportan tipos de datos específicos para almacenar documentos XML permiten el almacenamiento estructurado, por ejemplo, Oracle Database ofrece el tipo de datos XMLType para almacenar y consultar documentos XML de manera eficiente.

Puede ser útil en entornos donde ya se utiliza una base de datos relacional como sistema de almacenamiento de datos principal pero algunas operaciones sobre datos XML pueden ser menos eficientes en una base de datos relacional que en bases de datos XML nativas.

1.3.- Almacenamiento de documentos XML en BBDD Nativas

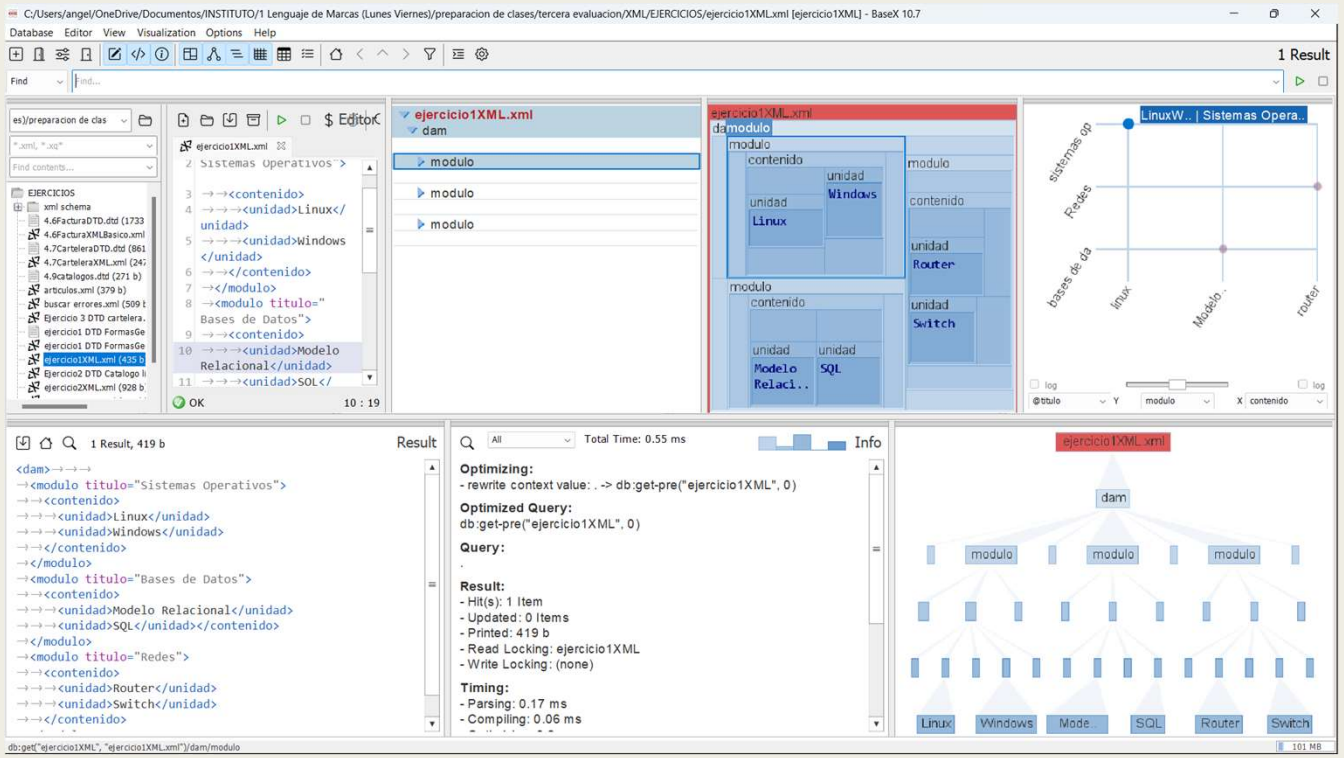
Es una opción diseñada específicamente para manejar y almacenar datos en formato XML de manera eficiente.

Las bases de datos nativas XML están optimizadas para trabajar directamente con documentos XML y proporcionan características específicas para el almacenamiento, consulta y manipulación de sus datos.

Estas bases de datos suelen ofrecer un lenguaje de consulta específico para XML (Xpath y XQuery)que permite realizar consultas complejas sobre los datos XML almacenados de manera eficiente y permite realizar transformaciones XSLT directamente en la base de datos

Nombre	Tipo de licencia
BaseX	Software libre – BSD
eXist-db	Software libre – LGPL
MarkLogic	Comercial
Qualcomm Qizx	Comercial

BaseX es uno de los sistemas gestores de BBDD nativas más reconocido. La instalación se realiza fácilmente descargándola desde https://basex.org
Una vez instalado y creando una base de datos indicando el fichero XML deseado se puede ver y operar con él. Ofrece múltiples opciones, así como múltiples ventanas de visualización de la estructura del xml.



2.- Lenguajes de consulta y documentación XML

El lenguaje XML se concibió para el intercambio de información en un formato estandarizado, esto es posible debido a su estructura jerárquica y al cumplimiento de las reglas establecidas.
Una de las operaciones más complejas es la búsqueda de datos en un documento XML, herramientas como los analizadores DOM y SAX permiten realizar estas búsquedas, pero pueden resultar algo complejas y poco eficientes.
Los lenguajes de consulta y manipulación de documentos XML son herramientas diseñadas para interactuar con documentos XML, permitiendo realizar operaciones como **recuperar datos**, realizar **transformaciones**, **filtrar información** y **modificar** la estructura del documento. Estos lenguajes proporcionan una forma estándar y expresiva de trabajar con documentos XML, facilitando su procesamiento y análisis partiendo de la estructura establecida en el XML.

Estos lenguajes proporcionan herramientas poderosas y expresivas para trabajar con documentos XML:

- **XPath**: proporciona una sintaxis sencilla y poderosa para expresar consultas
- **XQuery**: permite realizar consultas más complejas que XPath, incluyendo operaciones de unión, filtrado y agregación de datos XML.
- **XSLT** (Extensible Stylesheet Language Transformations): se utiliza para transformar documentos XML en otros formatos como HTML o texto, XSLT también puede utilizarse para realizar consultas y manipulaciones simples en documentos XML. Permite escribir reglas de transformación para seleccionar, filtrar y reorganizar elementos XML.

2.1.- XPath

Es un lenguaje de manipulación de documentos XML que permite seleccionar nodos o conjuntos de nodos de un documento XML navegando por su estructura.

Se utiliza como soporte para otras tecnologías (XQuery, XPointer, XSLT o XSL-FO) y es un estándar del W3C

Proporciona una sintaxis sencilla y poderosa para expresar consultas sobre la estructura y el contenido de un documento XML

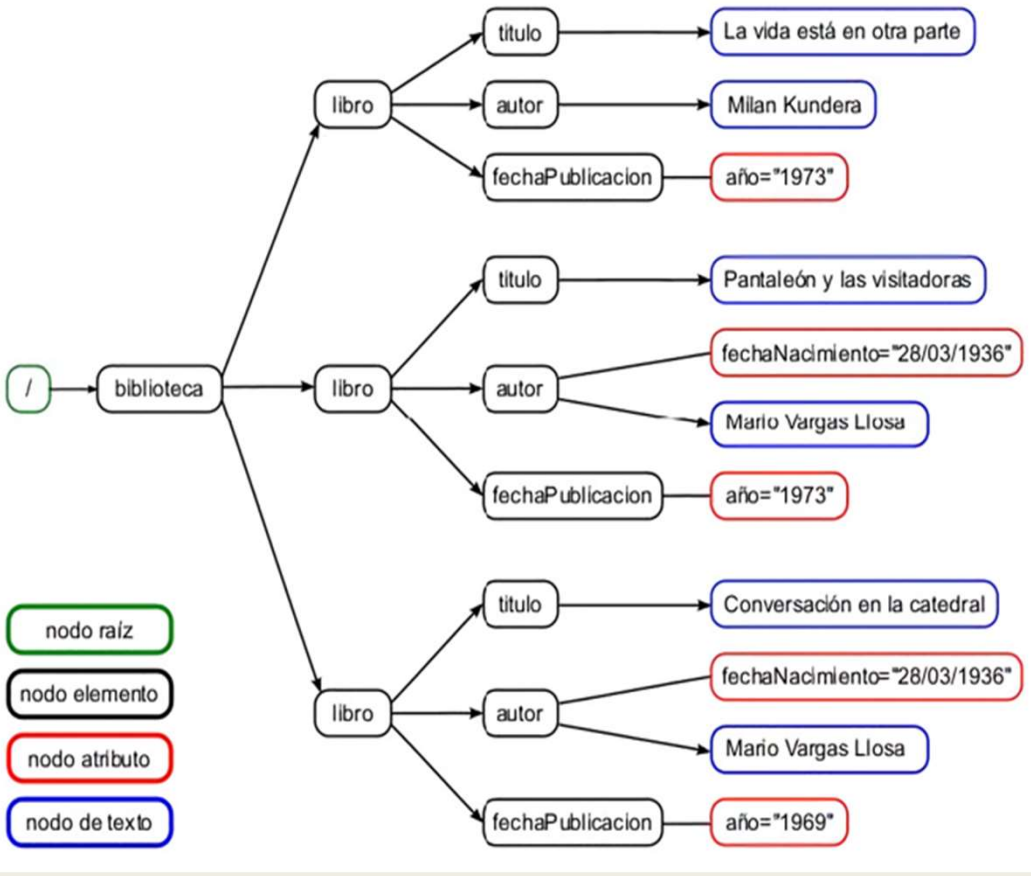
Existen varias versiones de XPath aprobadas por el W3C, la última es la 3.1 presentada el 2017 aunque la versión que más se utiliza sigue siendo la versión 1 presentada en 1999.

Recomendación del w3c sobre XPath https://www.w3.org/TR/xpath

Un XML tiene una estructura árbol que es una estructura parecida a la de directorios de cualquier sistema operativo.

Podemos representar un XML como un árbol dirigido

```
<?xml version="1.0" encoding="UTF-8"?>
<biblioteca>
    <libro>
        <titulo>La vida está en otra parte</titulo>
        <autor>Milan Kundera</autor>
        <fechaPublicacion año="1973"/>
    </libro>
    <libro>
        <titulo>Pantaleón y las visitadoras</titulo>
        <autor fechaNacimiento="28/03/1936">Mario
        Vargas Llosa</autor>
        <fechaPublicacion año="1973"/>
    </libro>
    <libro>
        <titulo>Conversación en la catedral</titulo>
        <autor fechaNacimiento="28/03/1936">Mario
        Vargas Llosa</autor>
        <fechaPublicacion año="1969"/>
    </libro>
</biblioteca>
```



Xpath se utiliza escribiendo expresiones, es una notación similar a la usada en el establecimiento de rutas en los sistemas de ficheros. Existen siete tipos de nodos:

- **Raiz**: representa el documento xml completo, se denota comúnmente con una barra diagonal ("/") seguida del nombre del elemento raíz. En el ejemplo <biblioteca>, el nodo raíz se representaría como /biblioteca
- **Elemento**: Estos nodos corresponden a las etiquetas de apertura y cierre en el XML. Por ejemplo, <libro> y <autor> son nodos de elemento.
- **Atributo**: Representan los atributos asociados a los elementos en el documento XML. Por ejemplo, año="1969" en <fechaPublicacion año="1969">
- **Texto**: Representan el texto contenido dentro de los elementos. Por ejemplo, en <titulo>La vida está en otra parte</titulo>
- **Comentario**: Representan comentarios y otras instrucciones de procesamiento
- **Instrucción de procesamiento**: encapsula instrucciones de procesamiento del xml
- **Espacio de nombres**: Representan los espacios de nombres definidos en el documento XML.

(la declaración DOCTYPE no se considera como nodo)
Los elementos se indican por su nombre.
Los atributos, anteponiendo @ a su nombre

Una expresión Xpath es una cadena de texto que representa un recorrido en el árbol del documento. Las expresiones más simples se parecen a las rutas de los archivos en el explorador de ficheros.

Expresión	Resultado
`/biblioteca/libro/autor`	<autor>Milan Kundera</autor> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
`/autor`	No devuelve nada porque <autor> no es hijo del nodo raíz
`/biblioteca/autor`	No devuelve nada porque <autor> no es hijo de <biblioteca>.
`/biblioteca/libro/autor/@fecha Nacimiento`	fechaNacimiento="28/03/1936" fechaNacimiento="28/03/1936"

La parte de una expresión Xpath que establece la ruta, es lo que se denomina el **eje** o **axis.** El eje nos permite seleccionar un subconjunto de nodos del documento y corresponde a recorridos (rutas) en el arbol

❑ Operadores

Para escribir el eje utilizamos operadores:

Operador	Descripción
/	Permite establecer la ruta hasta el nodo que queramos.
//	Selecciona cualquier nodo que se encuentre en algún subnivel.
..	Selecciona el nodo padre.
|	Permite varios recorridos
*	Todos los elementos debajo del nodo

Eje descendiente //

Expresión	Resultado
`//autor`	<autor>Milan Kundera</autor> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
`//@fechaNacimiento`	fechaNacimiento="28/03/1936" fechaNacimiento="28/03/1936"
`/biblioteca//titulo`	<titulo>La vida está en otra parte</titulo> <titulo>Pantaleón y las visitadoras</titulo> <titulo>Conversación en la catedral</titulo>

Eje padre **/..**

Expresión	Resultado
`/biblioteca/libro/autor/@fechaNacimiento/..`	<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
`//@fechaNacimiento/../..`	<libro> <titulo>Pantaleón y las visitadoras</titulo> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor> <fechaPublicacion año="1973"/> </libro> <libro> <titulo>Conversación en la catedral</titulo> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor> <fechaPublicacion año="1969"/> </libro>

Eje varios recorridos |

Expresión	Resultado
`//autor | //titulo`	<titulo>La vida está en otra parte</titulo> <autor>Milan Kundera</autor> <titulo>Pantaleón y las visitadoras</titulo> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor> <titulo>Conversación en la catedral</titulo> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
`//autor | //titulo | //@año`	<titulo>La vida está en otra parte</titulo> <autor>Milan Kundera</autor> año="1973" <titulo>Pantaleón y las visitadoras</titulo> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor> año="1973" <titulo>Conversación en la catedral</titulo> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor> año="1969"

❑ Predicados

Los predicados en XPath son condiciones adicionales que se pueden agregar a una expresión XPath para filtrar y seleccionar nodos específicos. Se utilizan para **restringir** el conjunto de nodos que se seleccionan, permitiendo así realizar consultas más específicas y detalladas sobre la estructura y el contenido del documento XML.

Se agregan a una expresión XPath utilizando corchetes **[ ]**, y contienen una condición que debe cumplirse para que un nodo sea seleccionado

Expresión	Resultado
`//autor[@fechaNacimiento]`	<autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor> <autor fechaNacimiento="28/03/1936">Mario Vargas Llosa</autor>
`//libro[1]`	<libro> <titulo>La vida está en otra parte</titulo> <autor>Milan Kundera</autor> <fechaPublicacion año="1973"/> </libro>

❑ Funciones

Xpath permite utilizar funciones tanto en los predicados como en los ejes.
Ejemplos:

```
//titulo/string()
//title/text()
//numero-trabajadores[data()>16]
```

- **boolean()**. Convierte a booleano. Al aplicarlo sobre un conjunto de nodos devuelve true si esta vacio.
- **position()**. Devuelve la posición de un nodo en su contexto.
- **last()**. Devuelve la ultima posición.
- **count()**. Devuelve el números de nodos e un conjunto de nodos.
- **not()**. Devuelve el contrario de un booleano dado.

Listado de funciones en https://developer.mozilla.org/es/docs/Web/XPath/Functions

En las condiciones se pueden utilizar los operadores siguientes:

- operadores lógicos: and, or, not().
- operadores aritméticos: +, -, *, div, mod.
- operadores de comparación: =, !=, <, >, <=, >=.

2.2.- XQUERY

Es un lenguaje diseñado para escribir consultas sobre colecciones de datos expresadas en XML, es decir, para extraer determinados elementos y atributos de documentos XML.
Se puede decir que XQuery es para XML como SQL para las BBDD relacionales, o sea, un lenguaje de consulta de datos.

Características de XQuery:

- ✓ Lenguaje expresivo y declarativo: permite a los usuarios especificar **qué** datos quieren recuperar o transformar, **en lugar de cómo** hacerlo (se indica lo que se busca, no la forma).
- ✓ Independiente del protocolo de acceso y del origen de los datos.
- ✓ Consultas y resultados respeta el modelo de datos XML.
- ✓ Incluye soporte para los namespaces.
- ✓ Soporta XML-Schemas y DTDs
- ✓ Soporta tipos simples (enteros, cadenas...) y tipos complejos.
- ✓ Puede combinar información de múltiples fuentes en una consulta.
- ✓ Independiente de la sintaxis. Pueden existir diferentes formas de expresar una misma consulta.
- ✓ Estándar W3C: lo que significa que su sintaxis y semántica están bien definidas y son consistentes en diferentes implementaciones y plataformas

Entre otras cosas XQuery puede utilizarse para:

- ✓ Recuperar información a partir de conjuntos de datos XML. Se utiliza para realizar consultas complejas en documentos XML. Puede seleccionar nodos específicos, filtrar datos, realizar cálculos y aplicar funciones sobre datos XML para recuperar la información necesaria

- ✓ Realizar transformaciones de datos en XML a otro tipo de representaciones, como HTML o PDF. Se utiliza para transformar la estructura y el contenido de los documentos XML.

- ✓ Se puede utilizar en entornos web y de desarrollo de aplicaciones para generar contenido dinámico basado en datos XML. Esto incluye la generación de páginas web dinámicas, la creación de feeds de noticias, la construcción de interfaces de usuario interactivas y la personalización de contenido en tiempo real en función de datos XML

- ✓ Puede utilizarse para extraer y procesar información específica de los documentos XML con el fin de generar informes, tableros de control y análisis de datos.

❑ Aspectos básicos del uso de XQuery

Para los ejemplos, vamos a utilizar el siguiente fichero “libros.xml” que lo puedes descargar desde el aula virtual

```
<?xml version="1.0" encoding="ISO-8859-1"?>
	<libreria>
		<libro categoria="COOKING">
			<titulo leng="en">Everyday Italian</titulo>
			<autor>Giada De Laurentiis</autor>
			<anyo>2005</anyo>
			<precio>30.00</precio>
		</libro>
		<libro categoria="CHILDREN">
			<titulo leng="en">Harry Potter</titulo>
			<autor>J K. Rowling</autor>
			<anyo>2005</anyo>
			<precio>29.99</precio>
		</libro>
		<libro categoria="WEB">
			<titulo leng="en">XQuery Kick Start</titulo>
			<autor>James McGovern</autor>
			<autor>Per Bothner</autor>
			<autor>Kurt Cagle</autor>
			<autor>James Linn</autor>
			<autor>Vaidyanathan Nagarajan</autor>
			<anyo>2003</anyo>
			<precio>49.99</precio>
		</libro>
		<libro categoria="WEB">
			<titulo leng="en">Learning XML</titulo>
			<autor>Erik T. Ray</autor>
			<anyo>2003</anyo>
			<precio>39.95</precio>
		</libro>
	</libreria>
```

- ✓ XQuery usa funciones para extraer los datos de los documentos XML. La función **doc()** se emplea para poder abrir el fichero y devuelve el documento completo. Por ejemplo, doc("libros.xml") devuelve el documento “libros.xml”.

- ✓ Usa expresiones “path” (rutas) para navegar a través de los diferentes nodos de un documento XML. Por ejemplo, la siguiente expresión se emplea para seleccionar todos los títulos (elementos título) del fichero "libros.xml": doc("libros.xml")/libreria/libro/titulo

- ✓ Usa predicados para limitar (filtrar) el número de datos que se extraen de un documento XML. El siguiente predicado permite seleccionar todos los elementos “libro” bajo el elemento “librería” que tienen un “precio” inferior a 30: doc("libros.xml")/libreria/libro[precio<30]

- ✓ XQuery y Namespaces: si el documento XML sobre el que va a realizar una consulta utiliza espacios de nombres, las consultas XQuery deben también utilizarlos. Para ello, es necesario declarar en la consulta los espacios de nombres y sus prefijos, mediante la instrucción:

  declare namespace prefijo=“espacio-de-nombres”;

- ✓ *Los comentarios en XQuery, a diferencia de XML, van encerrados entre caras sonrientes :), tal y como se muestra a continuación.*

  (: Esto es un comentario XQuery :)

Al trabajar con XQuery hay que tener en cuenta las siguientes reglas básicas:

- ✓ XQuery es sensible a mayúsculas y minúsculas.
- ✓ Cualquier elemento, atributo o variable de XQuery tiene que ser un identificador válido en XML.
- ✓ Un valor string o cadena de XQuery puede estar contenido tanto en comillas simples como dobles.
- ✓ Una variable XQuery se define con un símbolo $ seguido por un nombre. Por ejemplo, $libreria.

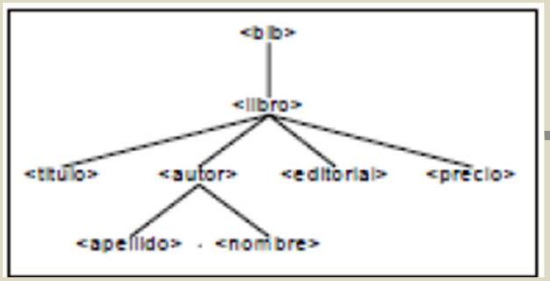
❑ Consultas XQuery: expresiones FLWOR

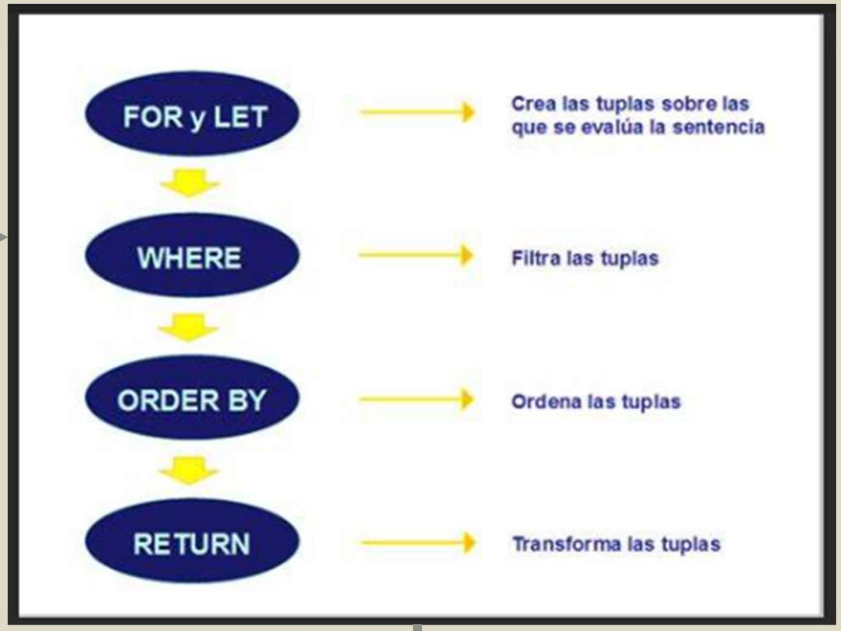
FLWOR (For Let Where Order Return) es una de las expresiones más potentes y típicas de XQuery. Se basa en ligar valores a variables con las cláusulas for y let y emplear esas variables para crear nuevos resultados.

Una expresión FLWOR:

- Comienza con una o más cláusulas for o let en cualquier orden (al menos una de ellas)
- Seguidas por una cláusula where opcional
- Puede aparecer una cláusula order by opcional
- Finaliza con una cláusula return obligatoria

Datos XML





Resultado XML

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.

Tupla: cada uno de los valores que adopta la variable

Consulta

```
for $x in doc("libros.xml")/libreria/libro
where $x/precio>30
order by $x/titulo
return $x/titulo
```

La tuplas en este caso son cada uno de los libros que hay en el documento (4)

Resultado

```
<titulo leng="en">Learning XML</titulo>
<titulo leng="en">XQuery Start</titulo>
```

❑ Clausula for

La clausula “for” enlaza una variable con cada item que se devuelve por la expresión “in”. Esta cláusula produce una iteración. Puede haber múltiples cláusulas “for” en una misma expresión.

- Para conseguir una iteración un determinado número de veces se utiliza la palabra reservada “to”. Por ejemplo:

Consulta

```
for $x in (1 to 3) return <libro>{$x}</libro>
```

Resultado

```
<libro>1</libro>
<libro>2</libro>
<libro>3</libro>
```

- La palabra reservada **“at”** permite contar la iteración:

Consulta

```
for $x at $i in /libreria/libro/titulo
return <libro>{$i}. {data($x)}</libro>
```

Resultado

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

- Se permite más de una expresión en una cláusula for. Para ello, se separan los valores mediante comas:

Consulta

```
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>
```

❑ Clausula let

La cláusula “let” permite hacer asignaciones de variables, esto permite no tener que repetir la misma expresión varias veces. La cláusula let no provoca una iteración.

Consulta

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

Resultado

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

❑ Clausulas where y order by

- La cláusula “where” se usa para especificar uno o más criterios para el resultado:

Consulta

```
for $x in /libreria/libro
where $x/precio>30 and $x/precio<100
return $x/titulo
```

Resultado

```
<titulo leng="en">XQuery Kick Start</titulo>
<titulo leng="en">Learning XML</titulo>
```

- La cláusula “order by” especifica el criterio de ordenación del resultado. Por defecto se ordena por orden ascendente. Si se quiere ordenar por orden descendente se debe incluir la palabra “descending” al final.

Consulta

```
for $x in /libreria/libro
order by $x/@categoria, $x/titulo
return $x/titulo
```

Resultado

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

❑ Clausula return

Esta cláusula indica lo que se quiere devolver

Consulta

```
for $x in /libreria/libro return $x/titulo
```

Resultado

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

Se puede devolver más de un valor poniendo los distintos valores entre paréntesis y separados por comas (,).

Consulta

```
for $x in /libreria/libro
return ($x/titulo, $x/autor, string($x/@categoria))
```

Resultado

```
<titulo leng="en">Everyday Italian</titulo>
<autor>Giada De Laurentiis</autor>
COOKING
<titulo leng="en">Harry Potter</titulo>
<autor>J K. Rowling</autor>
CHILDREN
<titulo leng="en">XQuery Kick Start</titulo>
<autor>James McGovern</autor>
<autor>Per Bothner</autor>
<autor>Kurt Cagle</autor>
<autor>James Linn</autor>
<autor>Vaidyanathan Nagarajan</autor>
WEB
<titulo leng="en">Learning XML</titulo>
<autor>Erik T. Ray</autor>
WEB
```

También se puede devolver con un texto combinando marcas y datos obtenidos en la consulta.

Consulta

```
for $x in /libreria/libro
return <h1>{$x/titulo}</h1>
```

Resultado

```
<h1><titulo leng="en">Everyday Italian</titulo></h1>
<h1><titulo leng="en">Harry Potter</titulo></h1>
<h1><titulo leng="en">XQuery Kick Start</titulo></h1>
<h1><titulo leng="en">Learning XML</titulo></h1>
```

No es posible devolver un nodo atributo en XQuery. Para devolver el valor de un atributo hay que utilizar la función string o data.

Consulta

```
for $x in /libreria/libro
return ($x/titulo, string($x/@categoria))
```

Resultado

```
<titulo leng="en">Everyday Italian</titulo>
COOKING
<titulo leng="en">Harry Potter</titulo>
CHILDREN
<titulo leng="en">XQuery Kick Start</titulo>
WEB
<titulo leng="en">Learning XML</titulo>
WEB
```

❑ Archivos XQuery

Las consultas XQuery se pueden almacenar en archivos, que normalmente llevan la extensión “xq”. Estos ficheros comienzan con la expresión *xquery version "1.0";* y en ellos se pueden combinar elementos de marcado, datos carácter y expresiones FLOWR (cada una encerrada entre llaves “{ }”).

```
xquery version "1.0";
<html>
    <body>
        <table border="1">
        {
            for $x in doc("libros.xml")/libreria/libro
            return <tr><td>{string($x/titulo)}</td></tr>
        }
        </table>
    </body>
</html>
```

Este fichero “ejemploXQuery.xq” obtiene la lista de todos los títulos de los libros en forma de tabla html:

```
<html>
    <body>
        <table border="1">
            <tr><td>Everyday Italian</td></tr>
            <tr><td>Harry Potter</td></tr>
            <tr><td>XQuery Kick Start</td></tr>
            <tr><td>Learning XML</td></tr>
        </table>
    </body>
</html>
```

❑ Operadores y funciones XQuery

XQuery 1.0, XPath 2.0 y XSLT 2.0 comparten el mismo conjunto de operadores y funciones.
https://www.w3.org/TR/xpath-functions/

Hay operadores y funciones de todo tipo: matemáticas, de cadenas, de comparación de fechas y horas, de manipulación de nodos XML, etc. Además de la biblioteca de funciones predefinidas, XQuery permite la creación de funciones definidas por el usuario.

Una llamada a una función puede aparecer en el mismo lugar que las expresiones. El prefijo usado por defecto en el espacio de nombres de las funciones es “fn:”. Por ejemplo, ***fn:string()***. Sin embargo, en el caso del nombre de las funciones, no es necesario usar el prefijo cuando son llamadas.
Algunas funciones predefinidas:
.

Funciones generales

- doc("URI"): devuelve el documento completo (el nodo documento)
- empty(item, item, ...): devuelve "true" si el valor del argumento ES una secuencia vacía. De lo contrario devuelve "false".
- exists(item1, item2, ...): es la función opuesa a "empty". Devuelve "true" si el valor del argumento NO ES una secuencia vacía. De lo contrario devuelve "false".
- distinct-values(): extrae los valores de una secuencia de nodos y crea una nueva secuencia con valores únicos, eliminando los nodos duplicados.
- string(): convierte el argumento (que será un valor atómico o un nodo) a string. Si el argumento es un nodo elemento, devuelve los datos carácter del contenido de elemento y de todos sus descendientes concatenados. Si es un nodo atributo, devuelve el valor del atributo como string. Si es un valor atómico, devuelve ese valor convertido a string.
- data(): devuelve el valor atómico del argumento. Si estos son nodos, devuelve sus datos carácter

Funciones agregadas

sum, avg, count, max, min, etc., que operan sobre una secuencia de números y devuelven un resultado numérico.

Funciones de cadena

string-length, substring, upper-case, lower-case(), concat(), etc.

❑ Modificación de XML con XQuery

Con Xquery también podemos utilizar funciones y expresiones que nos permiten modificar le fichero XML, entre ellas:

- ✓ **insert node** para insertar un nodo dentro de un documento XML.

  insert node <nodo_a_insertar> into <ubicación>

  La ubicación puede ser con la sentencia para insertar al principio de la base de datos ***as first into*** O para insertar al final ***as last into***

- ✓ **replace** se utiliza para reemplazar nodos o valores así como para reemplazar contenido de texto o atributos

- ✓ **delete node** se utiliza para eliminar nodos específicos,elementos, atributos, texto.

- ✓ **rename node** se utiliza para cambiar el nombre de un nodo, de un elemento, atributo, proceso de instrucción, etc

- ✓ ....