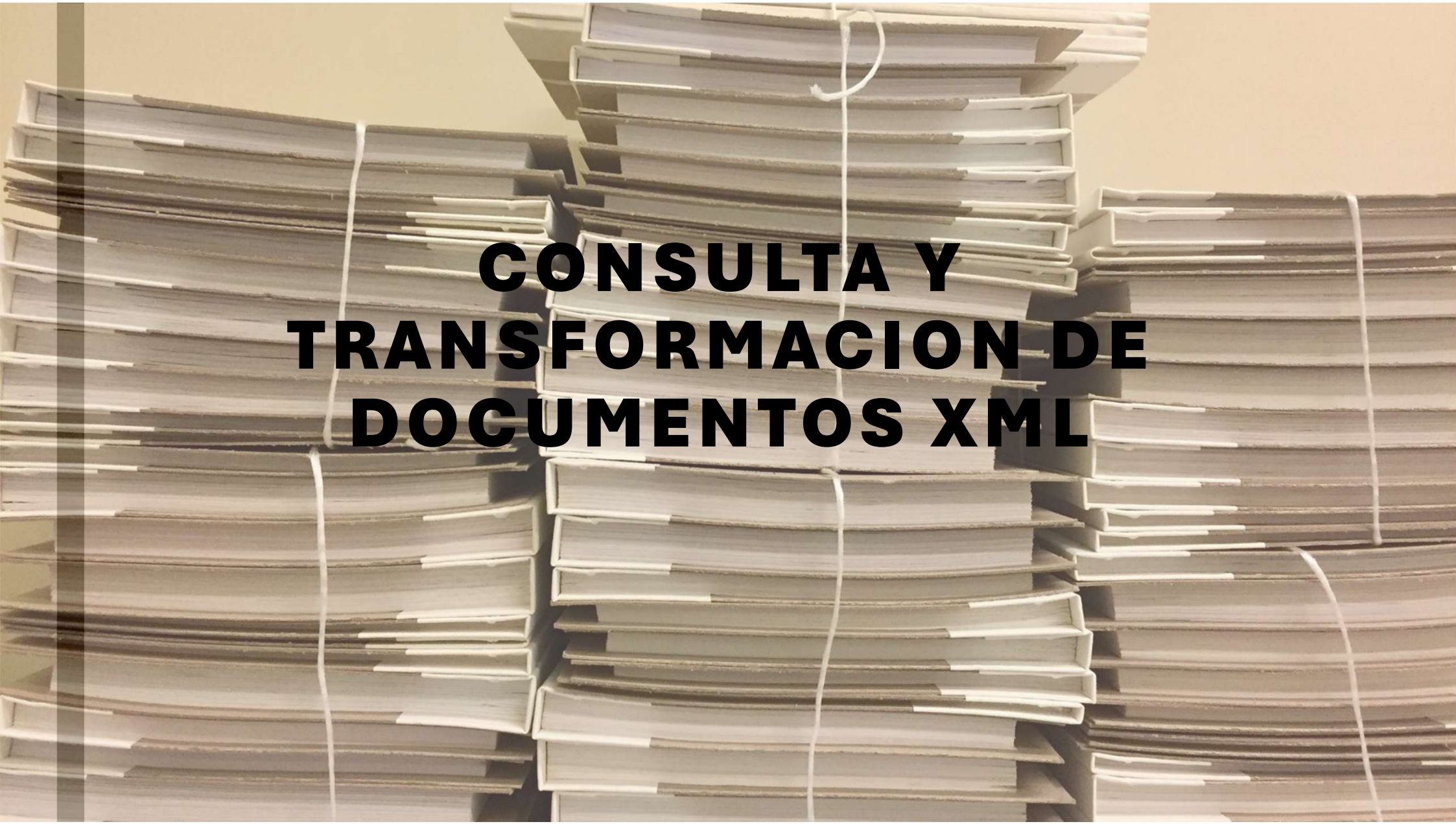


CONSULTA Y TRANSFORMACION DE DOCUMENTOS XML

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

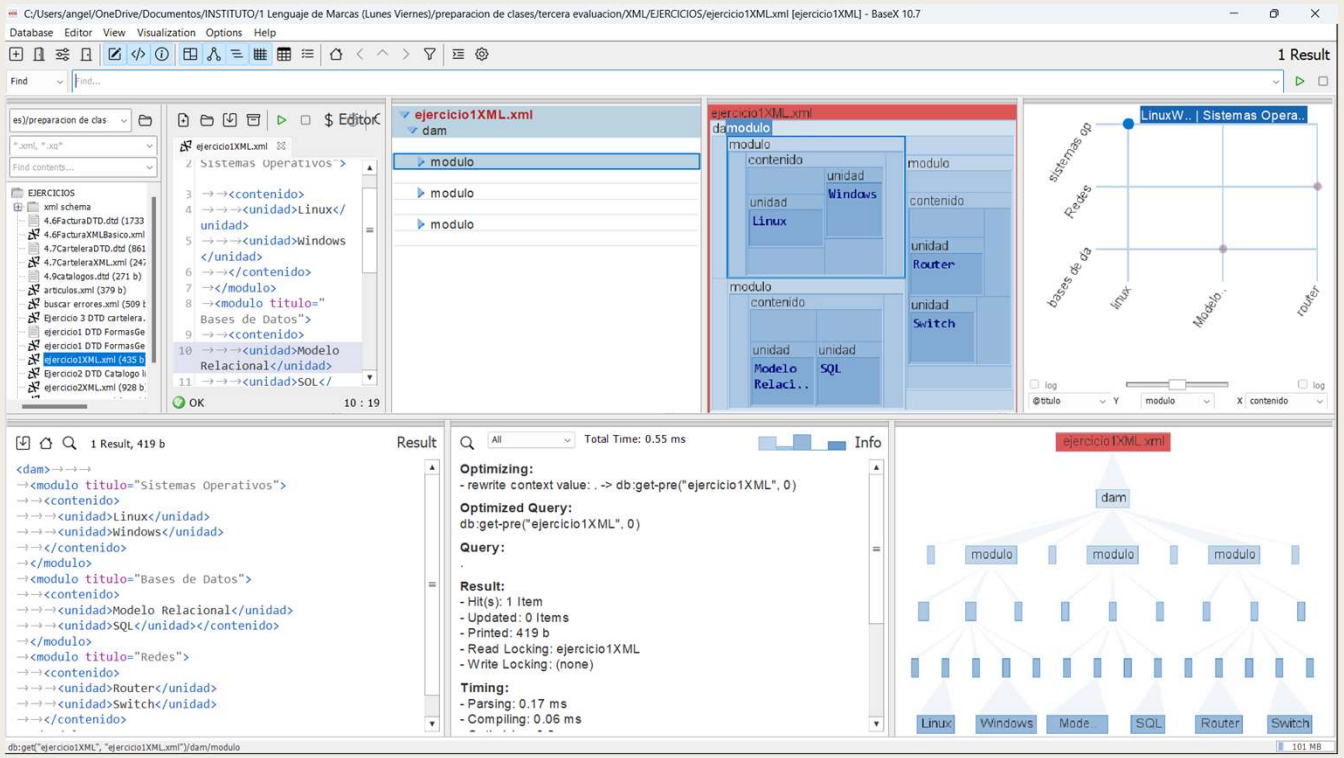
Es una opción diseñada específicamente para manejar y almacenar datos en formato XML de manera eficiente.

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

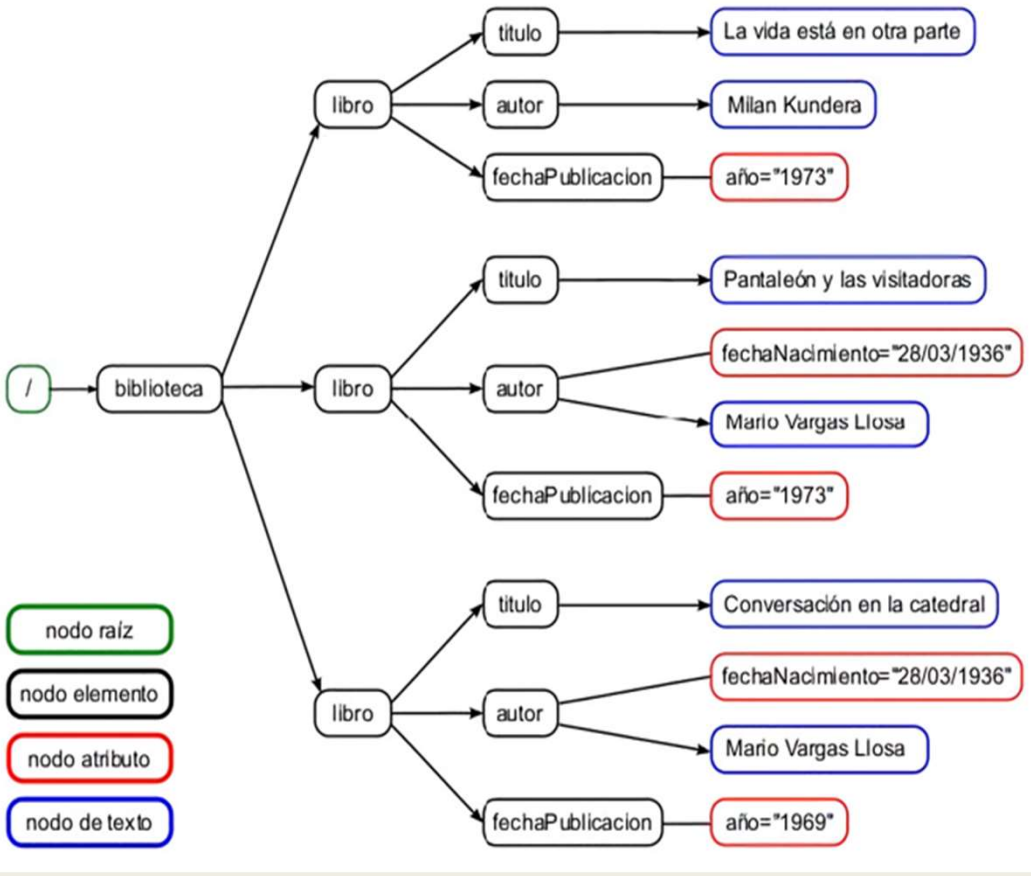
Existen varias versiones de XPath aprobadas por el W3C, la última es la 3.1 presentada el 2017 aunque la versión que más se utiliza sigue siendo la versión 1 presentada en 1999.

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.

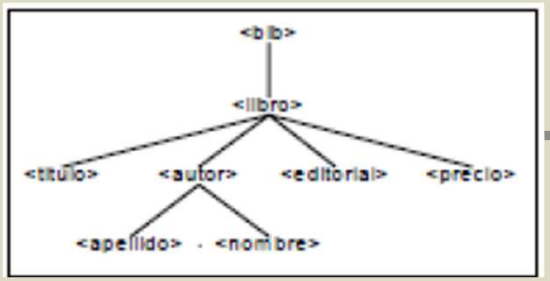
❑ Consultas XQuery: expresiones FLWOR

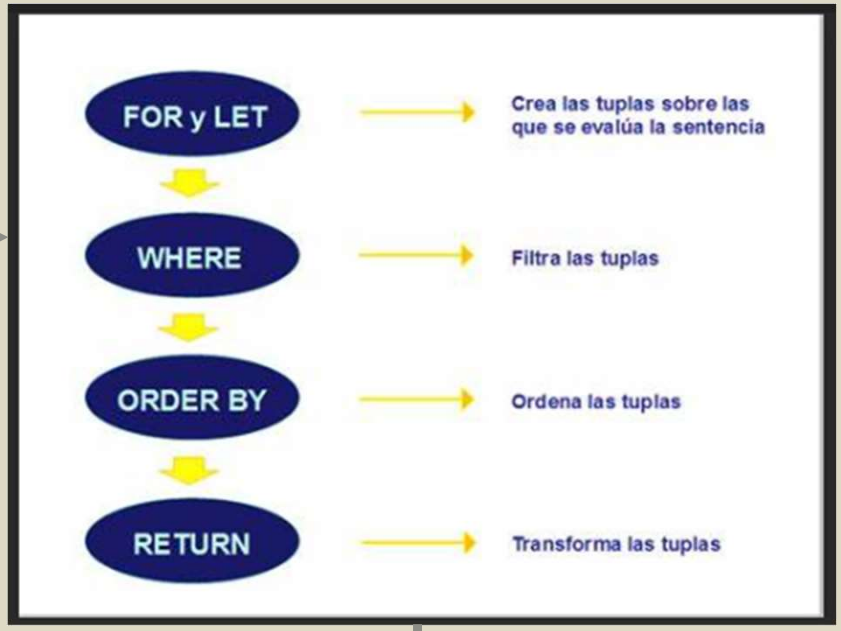
FLWOR (For Let Where Order Return) es una de las expresiones más potentes y típicas de XQuery. Se basa en ligar valores a variables con las cláusulas for y let y emplear esas variables para crear nuevos resultados.

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

- ✓ ....

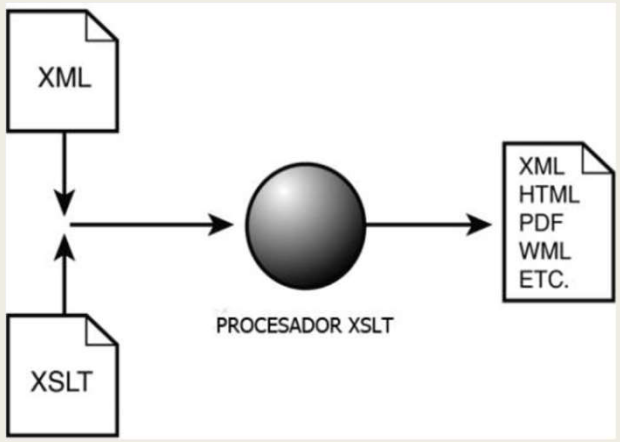
2.3.- XSLT

¿Qué es XSLT?

- eXtensibleStylesheetLanguage–Transformations: Lenguaje para transformación de documentos XML
- Es un lenguaje que permite reestructurar el documento XML original según el formato deseado: XML, HTML, texto, PDF, etc.
- Lenguaje declarativo: son sucesivas declaraciones de reglas o plantillas (“templates”)

¿Cómo funciona?

En el proceso de transformación, XSLT utiliza XPath para definir las partes del documento de origen que debe coincidir con una o varias plantillas predefinidas. Cuando se encuentra una coincidencia, XSLT transformará la parte correspondiente del documento de origen en el documento de resultado



CSS vs XSLT

- A un documento XML se le pueden aplicar hojas de estilo CSS (formato), pero esto NO PERMITE:
  - Cambiar el orden de los elementos al visualizarlos
  - Realizar operaciones numéricas
  - Combinar múltiples elementos

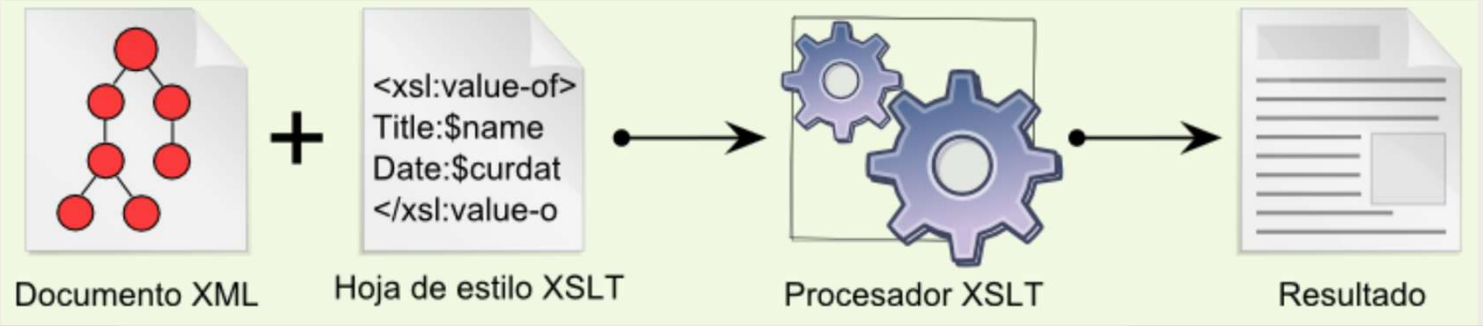
  Aplicar una hoja de estilos CSS (formato) a un documento XML:

  ```
  <?xml-stylesheet type= “text/css” href= “hoja.css” ?>
  ```

- XSLT permite hacer todo esto, determinando además:
  - Qué contenido se muestra
  - En qué orden
  - Resultados de cálculos, etc…

  Aplicar una transformación XSLT a un documento XML:

  ```
  <?xml-stylesheet type= “text/xsl” href= “hoja.xsl” ?>
  ```



- El documento XML es el documento inicial a partir del cual se va a generar el resultado.

- La hoja de estilo XSLT es el documento que contiene el código fuente del programa, es decir, las reglas de transformación que se van a aplicar al documento inicial.

- El procesador XSLT es el programa que aplica al documento inicial las reglas de transformación incluidas en la hoja de estilo XSLT y genera el documento final.

- El resultado de la ejecución del programa es un nuevo documento (que puede ser un documento XML u otro formato deseado, generalmente HTML).

Ejemplos de procesadores:

- https://xslttest.appspot.com/
- https://freeformatter.com/xsl-transformer.html

¿Cómo crear una hoja de estilos XSLT?

Las hojas de estilo XSLT se construyen sobre estructuras denominadas plantillas (templates). Una plantilla indica qué es lo que hay que buscar en el árbol de origen y qué es lo que hay que devolver en el árbol de resultado.
Un documento XSLT está escrito en XML

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

Un documento XSLT se engloba entre las etiquetas <xsl:stylesheet … > y </xsl:stylesheet> y entre estas etiquetas hay otro conjunto de etiquetas **xsl:template** (reglas de plantilla) que permiten definir las **transformaciones** que se van **a aplicar** sobre cada elemento del documento original.

```
<?xml versión=”1.0” encoding=”ISO-8859-1” ?>
<xsl:stylesheet  xmlns:xsl=“http://www.w3.org/1999/XSL/Transform” version=”1.0” >
…
<!-- Reglas de transformación-->
…
</xsl:stylesheet>
```

El espacio de nombres xmlns:xsl = "http://www.w3.org/1999/XSL/Transform" apunta al espacio de nombres oficial W3C XSLT. Si se utiliza este espacio de nombres, también debe incluir como atributo la versión = "1.0".

Para especificar el formato del documento de salida que deseamos se utiliza la declaración **<xsl:output>** Esta declaración no es obligatoria, si no se declara, se utiliza un método de salida predeterminado (XML). Sin embargo, se recomienda su uso para personalizar la salida según sea necesario.

```
<?xml versión=”1.0” encoding=”ISO-8859-1” ?>
<xsl:stylesheet  xmlns:xsl=“http://www.w3.org/1999/XSL/Transform” version=”1.0” >
<xsl:output method="valor"  encoding="valor“ doctype-system="valor“ doctype-public="valor“
indent="yes|no“..... />
<!-- Reglas de transformación-->
...
</xsl:stylesheet>
```

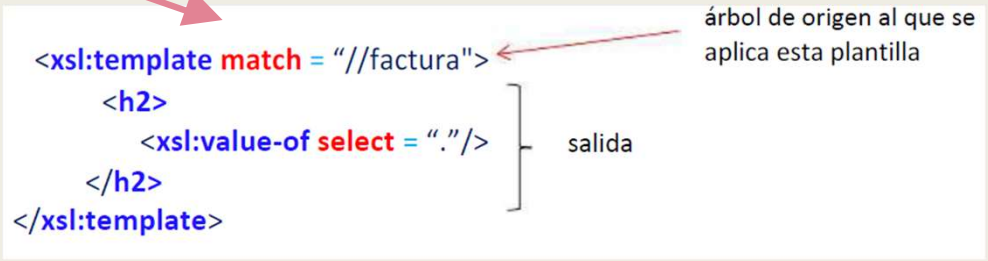
Con el atributo **method** se indica el formato deseado:

- ✓ xml: predeterminado, para generar un documento XML bien formado.
- ✓ html: para generar un documento HTML.
- ✓ text: para generar texto sin formato.
- ✓ json: para generar un documento JSON.
- ✓ adaptive: Este método es compatible con varios tipos de salida, y la decisión sobre el tipo de salida se toma en función del contenido de la plantilla.
- ✓ other: Permite especificar un método de salida personalizado.

Reglas de transformación

El elemento **plantilla:** es el elemento utilizado en la hoja de estilos para realizar las transformaciones

```
<xsl:stylesheet  xmlns:xsl="http://www.w3.org/1999/XSL/Transform" version="1.0" >
...
<!-- Reglas de transformación-->
...
</xsl:stylesheet>
```



La regla se aplicará a todas las instancias del elemento factura que se encuentren en el documento xml al que se va a aplicar la transformación. Esto se indica mediante el atributo match que acompaña al elemento xsl:template

Entre las etiquetas de inicio y de fin del elemento xsl:template se escribe la transformación que se debe realizar, es decir, qué texto y qué marcas se escribirán en el documento resultado de la transformación cada vez que se encuentre una instancia del elemento factura en el documento origen.

Con <xsl:value-of...>, se recupera y escribe en el documento resultado el valor del elemento que está siendo procesado.

xsl:apply-templates: se usa dentro de la plantilla para llamar a otras plantillas, se puede considerar como una especie de "subplantilla" ya que permite aplicar plantillas específicas de forma dinámica a diferentes partes del documento XML de origen.
Aplica, en el lugar donde aparece, la plantilla definida para el elemento indicado en el atributo select.


```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
        <html>
        <body>
                <h2>Mi coleccion de Cds</h2>
                <xsl:apply-templates/>
        </body>
         </html>
</xsl:template>
<xsl:template match="cd">
        <p>
                <xsl:apply-templates select="titulo"/>
                <xsl:apply-templates select="artista"/>
        </p>
</xsl:template>

<xsl:template match="titulo">
        Titulo: <span style="color:#ff0000">
        <xsl:value-of select="."/></span>
        <br />
</xsl:template>
<xsl:template match="artista">
        Artista: <span style="color:#00ff00">
        <xsl:value-of select="."/></span>
        <br />
</xsl:template>
</xsl:stylesheet>
```

Elementos básicos

Estos elementos XSLT nos permiten seleccionar y efectuar operaciones sobre los elementos del documento XML. Los más importantes son los siguientes:

xsl:value-of
xsl:for-each
xsl:sort
xsl:if
xsl:choose
xsl:text
xsl:element
xsl:attribute
xsl:variable

❑xsl:value-of

<xsl:value-of> se utiliza para extraer el valor de un nodo seleccionado.
El elemento <xsl:value-of> puede utilizarse para extraer el valor de un elemento XML y agregarlo a la corriente de salida de la transformación:

```
<xsl:value-of  select="expresión_xpath" disable-output-escaping="yes|no"  separator="separador"/>
```

select: expresión XPath utilizada para seleccionar el nodo del que se extraerá el valor. Determina qué valor se va a extraer y procesar.

disable-output-escaping: controla si se desactiva o no el escape de salida. Puede ser útil cuando se necesita incluir caracteres especiales en el resultado. (hace que en la salida muestren o no los caracteres “&” y “<“ en lugar de “&” o “<”)

separator: especifica el separador que se utilizará si se seleccionan varios nodos con la expresión XPath. Por defecto, los valores de los nodos seleccionados se concatenan sin ningún separador, este atributo permite especificar un carácter o cadena para separar los valores.

XML

```
<libros>
   <libro>
      <titulo>El gran Gatsby</titulo>
      <autor>F. Scott Fitzgerald</autor>
      <genero>Ficción</genero>
   </libro>
   <libro>
      <titulo>1984 &amp;amp; 2001</titulo>
      <autor>George Orwell &amp;amp; Arthur C. Clarke</autor>
      <genero>Ciencia ficción</genero>
   </libro>
</libros>
```

SALIDA

El gran Gatsby, 1984 & 2001

XSLT

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:output method="text"/>
   <xsl:template match="/">
      <!-- Utilizando xsl:value-of con disable-output-escaping="yes" -->
      <xsl:value-of select="libros/libro/titulo" disable-output-escaping="yes" separator=", "/>
   </xsl:template>
</xsl:stylesheet>
```

❑xsl:for-each

<xsl:for-each> Se utiliza para recorrer los elementos de un documento, y realizar una serie de operaciones con los mismos. Permite iterar sobre un conjunto de nodos seleccionados y aplicar la plantilla a cada uno de ellos

```
<xsl:for-each select="expresión_xpath">
    <!-- Contenido: Instrucciones XSLT para procesar cada nodo en la selección -->
</xsl:for-each>
```

select: expresión XPath utilizada para seleccionar el conjunto de nodos sobre los cuales se va a iterar.

contenido: Dentro del bloque <xsl:for-each> se puede incluir cualquier instrucción XSLT que se necesite ejecutar para cada nodo en la selección. Esto puede incluir <xsl:value-of>, <xsl:apply-templates>, o cualquier otra instrucción XSLT necesaria para procesar los nodos seleccionados.

XML

```
<colores>
   <color>Rojo</color>
   <color>Verde</color>
   <color>Azul</color>
</colores>
```

XSLT

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <html>
         <body>
            <h1>Lista de colores</h1>
            <ul>
               <!-- Iterar sobre cada color -->
               <xsl:for-each select="colores/color">
                  <li><xsl:value-of select="."/></li>
               </xsl:for-each>
            </ul>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

SALIDA

```
<html>
   <body>
      <h1>Lista de colores</h1>
      <ul>
         <li>Rojo</li>
         <li>Verde</li>
         <li>Azul</li>
      </ul>
   </body>
</html>
```

❑xsl:sort

<xsl:sort> Permite especificar el criterio de ordenación basado en los valores de los nodos o en otros criterios definidos. Se utiliza para ordenar elementos o atributos.

```
<xsl:sort select="expresión_xpath“ order="ascending|descending" data-type="text|number|date|..."
   lang="código_de_idioma" collation="URI_de_colación"/>
```

select: expresión XPath utilizada para seleccionar el valor que se utilizará para ordenar los nodos.

order: Indica si los nodos se ordenarán en orden ascendente (order="ascending") o descendente (order="descending")

data-type: especifica el tipo de datos de los valores a ordenar, como text, number, date, etc.

lang: opcional, se puede especificar el código de idioma (por ejemplo, "en" para inglés) para la ordenación.

collation: opcional, se puede especificar una URI que define reglas específicas para la ordenación, especialmente en casos donde las reglas de ordenación estándar no son suficientes.

XML

```
<numbers>
   <number>5</number>
   <number>2</number>
   <number>8</number>
   <number>3</number>
   <number>1</number>
</numbers>
```

XSLT

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <html>
         <body>
            <h1>Números ordenados</h1>
            <ul>
               <!-- Iterar sobre cada número, ordenándolos de forma ascendente -->
               <xsl:for-each select="numbers/number">
                  <xsl:sort select="." order="ascending" data-type="number"/>
                  <li><xsl:value-of select="."/></li>
               </xsl:for-each>
            </ul>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

SALIDA

```
<html>
   <body>
      <h1>Números ordenados</h1>
      <ul>
         <li>1</li>
         <li>2</li>
         <li>3</li>
         <li>5</li>
         <li>8</li>
      </ul>
   </body>
</html>
```

❑xsl:if

<xsl:if> Aplica una serie de instrucciones si se cumple la condición establecida en el atributo test. Permite ejecutar ciertas acciones dependiendo de si una condición dada es verdadera o falsa.

```
<xsl:if test="expresión_xpath">
    <!-- Contenido: Instrucciones XSLT para ejecutar si la condición es verdadera -->
</xsl:if>
```

test: expresión XPath que se evaluará para determinar si la condición es verdadera o falsa. Si la expresión XPath devuelve un resultado que se considera "verdadero" entonces el contenido dentro del <xsl:if> se ejecutará. Si la expresión XPath devuelve un resultado que se considera "falso", entonces el contenido dentro del <xsl:if> se omitirá.

contenido: instrucciones XSLT que se ejecutarán si la condición especificada en el atributo test es verdadera

Operador de igualdad (=):	“=”
Operador de desigualdad (≠):	“!=”
Operador menor que (<):	“<”
Operador mayor que (>):	“>”
Operador menor o igual que (<=):	“&ile;”
Operador mayor o igual que (>=):	“&ige;”

and	operación binaria inclusiva
or	operación binaria exclusiva
not()	operación unaria de negación

XML

```
<datos>
   <temperatura>25</temperatura>
</datos>
```

SALIDA

```
<html>
   <body>
      <p>¡Hace calor hoy!</p>
   </body>
</html>
```

XSLT

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <html>
         <body>
            <xsl:if test="datos/temperatura &gt;= 20">
               <p>¡Hace calor hoy!</p>
            </xsl:if>
            <xsl:if test="datos/temperatura &lt; 20">
               <p>Hace un clima agradable hoy.</p>
            </xsl:if>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

❑xsl:choose

<xsl:choose> permite elegir entre diferentes posibilidades. Es una estructura condicional múltiple. Similar al case o switch de otros lenguajes. Se utiliza cuando hay múltiples opciones posibles y se desea seleccionar una de ellas dependiendo de ciertas condiciones

```
<xsl:choose>
   <xsl:when test="expresión_xpath_1">
      <!-- Contenido: Instrucciones XSLT para ejecutar si la condición 1 es verdadera -->
   </xsl:when>
   <xsl:when test="expresión_xpath_2">
      <!-- Contenido: Instrucciones XSLT para ejecutar si la condición 2 es verdadera -->
   </xsl:when>
   <!-- Agregar más <xsl:when> según sea necesario -->
   <xsl:otherwise>
      <!-- Contenido: Instrucciones XSLT para ejecutar si ninguna de las condiciones anteriores es verdadera -->
   </xsl:otherwise>
</xsl:choose>
```

Se compone de uno o más operadores xsl:when, seguidos al final de un único operador xsl:otherwise

<xsl:choose>:elemento principal que inicia la estructura de selección condicional.

<xsl:when>: para especificar una condición a ser evaluada. Si la condición es verdadera, el contenido de ese <xsl:when> se ejecutará.

test: la expresión XPath que se evaluará para determinar si la condición es verdadera.

<xsl:otherwise>: se utiliza como una opción de "último recurso" que se ejecutará si ninguna de las condiciones especificadas en los elementos <xsl:when> es verdadera. No tiene un atributo test.

XML

```
<clima>
   <temperatura>15</temperatura>
   <descripcion>Nublado</descripcion>
</clima>
```

SALIDA

```
<html>
   <body>
      <h2>Estado del clima:</h2>
      <p>El clima es templado.</p>
   </body>
</html>
```

XSLT

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <html>
         <body>
            <h2>Estado del clima:</h2>
            <xsl:choose>
               <xsl:when test="clima/temperatura &gt;= 25">
                  <p>Hace calor hoy.</p>
               </xsl:when>
               <xsl:when test="clima/temperatura &lt; 25 and clima/temperatura &gt;= 15">
                  <p>El clima es templado.</p>
               </xsl:when>
               <xsl:otherwise>
                  <p>Hace frío hoy.</p>
               </xsl:otherwise>
            </xsl:choose>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

❑xsl:text

<xsl:text> permite escribir texto de manera literal en el árbol de salida. Esto es útil cuando se desea añadir texto fijo o espacios en blanco en la salida generada por la transformación XSLT.

```
<xsl:text>texto literal aquí</xsl:text>
```

XML

```
<productos>
   <producto>
      <nombre>Camisa</nombre>
      <precio>25</precio>
   </producto>
   <producto>
      <nombre>Pantalón</nombre>
      <precio>40</precio>
   </producto>
</productos>
```

XSLT

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <html>
         <body>
            <h2>Listado de Productos:</h2>
            <xsl:for-each select="productos/producto">
               <p>
                  Producto: <xsl:value-of select="nombre"/>
                  <xsl:text>, Precio: $</xsl:text>
                  <xsl:value-of select="precio"/>
               </p>
            </xsl:for-each>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

SALIDA

```
<html>
   <body>
      <h2>Listado de Productos:</h2>
      <p>Producto: Camisa, Precio: $25</p>
      <p>Producto: Pantalón, Precio: $40</p>
   </body>
</html>
```

❑xsl:element

<xsl:element> permite crear un nuevo elemento en el árbol resultado. Es útil cuando se necesita generar elementos cuyos nombres o valores pueden variar dependiendo de los datos de entrada o la lógica de transformación

```
<xsl:element name="nombre_del_elemento">
    <!-- Contenido: Instrucciones XSLT para generar el contenido del elemento -->
</xsl:element>
```

name: Especifica el nombre del elemento que se creará dinámicamente. Puede ser un valor fijo o una expresión XPath que se evalúa dinámicamente durante la transformación.

contenido: dentro de las etiquetas <xsl:element>, se puede incluir cualquier instrucción XSLT que tenga que formar parte del contenido del elemento que se está creando.

XML

```
<productos>
   <producto>
      <nombre>Camisa</nombre>
      <precio>25</precio>
   </producto>
   <producto>
      <nombre>Pantalón</nombre>
      <precio>40</precio>
   </producto>
</productos>
```

XSLT

```
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <productos>
         <xsl:apply-templates select="productos/producto"/>
      </productos>
   </xsl:template>
      <xsl:template match="producto">
      <xsl:element name="{nombre}">
         <precio><xsl:value-of select="precio"/></precio>
      </xsl:element>
   </xsl:template>
</xsl:stylesheet>
```

SALIDA

```
<productos>
   <Camisa>
      <precio>25</precio>
   </Camisa>
   <Pantalón>
      <precio>40</precio>
   </Pantalón>
</productos>
```

❑xsl:attribute

<xsl:attribute> permite crear un nuevo atributo en el árbol resultado. Es útil cuando se necesita generar elementos con atributos cuyos nombres o valores pueden variar dependiendo de los datos de entrada o la lógica de transformación.

```
<xsl:attribute name="nombre_del_atributo">
    <!-- Valor del atributo: expresión XSLT o texto literal -->
</xsl:attribute>
```

name: Especifica el nombre del atributo que se añadirá al elemento XML. Puede ser un valor fijo o una expresión XPath que se evalúa dinámicamente durante la transformación

contenido: dentro de las etiquetas <xsl:attribute>, se puede incluir cualquier instrucción XSLT que tenga que formar parte del contenido del elemento que se está creando.

XML

```
<productos>
   <producto>
      <nombre>Camisa</nombre>
      <precio>25</precio>
   </producto>
   <producto>
      <nombre>Pantalón</nombre>
      <precio>40</precio>
   </producto>
</productos>
```

XSLT

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <productos>
         <xsl:apply-templates select="productos/producto"/>
      </productos>
   </xsl:template>
   <xsl:template match="producto">
      <producto>
         <xsl:attribute name="nombre"><xsl:value-of select="nombre"/></xsl:attribute>
         <xsl:attribute name="precio"><xsl:value-of select="precio"/></xsl:attribute>
      </producto>
   </xsl:template>
</xsl:stylesheet>
```

SALIDA

```
<productos>
   <producto nombre="Camisa" precio="25"/>
   <producto nombre="Pantalón" precio="40"/>
</productos>
```

❑xsl:variable

<xsl:variable> para declarar y asignar valores a variables. Estas variables pueden contener valores que se pueden utilizar en varias partes de la hoja de estilos XSLT

```
<xsl:variable name="nombre_de_la_variable" select="expresión"/>
```

name: Especifica el nombre de la variable. Debe ser único dentro del ámbito en el que se declara la variable.

select: Especifica la expresión XPath que se evaluará para asignar un valor a la variable. El resultado de esta expresión se asigna a la variable.

XML

```
<productos>
   <producto>
      <nombre>Camisa</nombre>
      <precio>25</precio>
   </producto>
   <producto>
      <nombre>Pantalón</nombre>
      <precio>40</precio>
   </producto>
</productos>
```

XSLT

```
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
   <xsl:template match="/">
      <html>
         <body>
            <h2>Lista de Productos:</h2>
            <xsl:variable name="precio_total">
               <xsl:value-of select="sum(productos/producto/precio)"/>
            </xsl:variable>
            <p>Precio total de los productos: <xsl:value-of select="$precio_total"/></p>
         </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

SALIDA

```
<html>
   <body>
      <h2>Lista de Productos:</h2>
      <p>Precio total de los productos: 65</p>
   </body>
</html>
```

Ejemplos: Sobre el fichero **catalogoscd.xml** del aula virtual.

Ejemplo 1: Visualizar el titulo y el artista de todos mis Cds

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
                    <table border="1">
<caption><h2>Título y artista de todos mis cds</h2></caption>
      <tr bgcolor="#9acd32">
            <th>Titulo</th>
            <th>Artista</th>
      </tr>
      <xsl:for-each select="catalogo/cds/cd">
      <tr>
            <td><xsl:value-of select="titulo"/></td>
            <td><xsl:value-of select="artista"/></td>
      </tr>
      </xsl:for-each>
 </table>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Transformation Result:

Close

Título y artista de todos mis cds

Titulo	Artista
Empire Burlesque	Bob Dylan
Esconde tu corazon	Bonnie Tyler
Greatest Hits	Dolly Parton
Aun tengo el blues	Gary Moore
Eros	Eros Ramazzotti
Aguas tranquilas	Bee Gees
Un poco más	Dr.Hook
Maggie May	Rod Stewart
Romanza	Andrea Bocelli
Cuando un hombre ama a una mujer	Percy Sledge
Angel negro	Savage Rose
Convirtiendose en piedra	Many
Por los Buenos tiempos	Kenny Rogers
Big Willie style	Will Smith
Tupelo Honey	Van Morrison
Soulsville	Jorn Hoel
Lo mejor de Cat Stevens	Cat Stevens
¡Alto!	Sam Brown
Puente de los espias	T`Pau
Bailarin Privado	Tina Turner
En medio de la noche	Kim Larsen
Concierto de Gala de Pavarotti	Luciano Pavarotti
El muelle de la bahia	Otis Redding
Libro de fotos	Simply Red
Rojo	The Communards
Libera mi corazon	Joe Cocker

Ejemplo 2: visualizar los títulos, artista y años de los cds cuyo año es menor que 1988

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
        <body>
            <table border="1">
<caption><h2>Título, artista y año de los cds cuyo año es menor que 1988</h2></caption>
                <tr bgcolor="#9acd32">
                <th>Titulo</th>
                <th>Artista</th>
                    <th>año</th>
                </tr>
                <xsl:for-each select="catalogo/cds/cd[anio &lt; 1988]">
                <tr>
                <td><xsl:value-of select="titulo"/></td>
                <td><xsl:value-of select="artista"/></td>
                    <td><xsl:value-of select="anio"/></td>
                </tr>
                </xsl:for-each>
            </table>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

Transformation Result:

Close

Título, artista y año de los cds cuyo año es menor que 1988

Titulo	Artista	año
Empire Burlesque	Bob Dylan	1985
Greatest Hits	Dolly Parton	1982
Un poco más	Dr.Hook	1976
Cuando un hombre ama a una mujer	Percy Sledge	1966
Tupelo Honey	Van Morrison	1971
Puente de los espias	T`Pau	1987
Bailarin Privado	Tina Turner	1984
En medio de la noche	Kim Larsen	1983
El muelle de la bahia	Otis Redding	1968
Libro de fotos	Simply Red	1985
Rojo	The Communards	1987
Libera mi corazon	Joe Cocker	1987

Ejemplo 3: visualizar el título los caracteres 2 a 5 del titulo de los cds que hay entre 1987 y 1996

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
        <body>
            <table border="1">
<caption><h2>Título y  caracteres 2 a 5 del título de los cds cuyo año está comprendido entre
1987 y 1996</h2></caption>
                <tr bgcolor="#9acd32">
                    <th>Titulo</th>
                    <th>caracteres 2 a 5 del título</th>
                </tr>
            <xsl:for-each select="catalogo/cds/cd[anio &gt; 1986 and anio &lt;1997]">
                <tr>
                    <td><xsl:value-of select="titulo"/></td>
                    <td><xsl:value-of select="substring(titulo,2,4)"/></td>
                </tr>
            </xsl:for-each>
            </table>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

Transformation Result:

Close

Título y caracteres 2 a 5 del título de los cds cuyo año está comprendido entre 1987 y 1996

Titulo	caracteres 2 a 5 del título
Esconde tu corazon	scon
Aun tengo el blues	un t
Maggie May	aggi
Romanza	oman
Angel negro	ngel
Convirtiendose en piedra	onvi
Por los Buenos tiempos	or l
Soulsville	ouls
Lo mejor de Cat Stevens	o me
¡Alto!	Alto
Puente de los espias	uent
Concierto de Gala de Pavarotti	onci
Rojo	ojo
Libera mi corazon	iber

Ejemplo 4: visualizar los cds que hay entre 1987 y 1996 y debajo visualizar el número de cds que hay

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
      <table border="1">
      <caption><h2>Título, artista y año de los cds cuyo año está comprendido entre 1987 y 1996</h2></caption>
            <tr bgcolor="#9acd32">
                  <th>Titulo</th>
                  <th>Artista</th>
                  <th>año</th>
            </tr>
            <xsl:for-each select="catalogo/cds/cd[anio &gt; 1986 and anio &lt;1997]">
                  <tr>
                        <td><xsl:value-of select="titulo"/></td>
                        <td><xsl:value-of select="artista"/></td>
                        <td><xsl:value-of select="anio"/></td>
                  </tr>
            </xsl:for-each>
                  <tr>
                        <th colspan="2">Número de cds entre 1987 y 1996</th>
                        <th><xsl:value-of select="count(//cd[anio &lt; 1997]) - count(//cd[anio &lt; 1986])" /></th>
                  </tr>
      </table>
 </body>
 </html>
</xsl:template>
</xsl:stylesheet>
```

Transformation Result:

Close

Título, artista y año de los cds cuyo año está comprendido entre 1987 y 1996

Titulo	Artista	año
Esconde tu corazon	Bonnie Tyler	1988
Aun tengo el blues	Gary Moore	1990
Maggie May	Rod Stewart	1990
Romanza	Andrea Bocelli	1996
Angel negro	Savage Rose	1995
Convirtiendose en piedra	Many	1993
Por los Buenos tiempos	Kenny Rogers	1995
Soulsville	Jorn Hoel	1996
Lo mejor de Cat Stevens	Cat Stevens	1990
¡Alto!	Sam Brown	1988
Puente de los espias	T`Pau	1987
Concierto de Gala de Pavarotti	Luciano Pavarotti	1991
Rojo	The Communards	1987
Libera mi corazon	Joe Cocker	1987
Número de cds entre 1987 y 1996		**14**

Ejemplo 5: Visualizar el titulo y artista de todos mis Cds ordenados alfabéticamente por artista

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
        <body>
        <table border="1">
                <caption><h2>Título, y artista de todos mis cds ordenados por artista</h2></caption>
                <tr bgcolor="#9acd32">
                <th>Titulo</th>
                <th>Artista</th>
                </tr>
                <xsl:for-each select="catalogo/cds/cd">
                <xsl:sort select="artista"/>
                <tr>
                <td><xsl:value-of select="titulo"/></td>
                <td><xsl:value-of select="artista"/></td>
                </tr>
                </xsl:for-each>
            </table>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

Transformation Result:

Close

Título, y artista de todos mis cds ordenados por artista

Titulo	Artista
Romanza	Andrea Bocelli
Aguas tranquilas	Bee Gees
Empire Burlesque	Bob Dylan
Esconde tu corazon	Bonnie Tyler
Lo mejor de Cat Stevens	Cat Stevens
Greatest Hits	Dolly Parton
Un poco más	Dr.Hook
Eros	Eros Ramazzotti
Aun tengo el blues	Gary Moore
Libera mi corazon	Joe Cocker
Soulsville	Jorn Hoel
Por los Buenos tiempos	Kenny Rogers
En medio de la noche	Kim Larsen
Concierto de Gala de Pavarotti	Luciano Pavarotti
Convirtiendose en piedra	Many
El muelle de la bahia	Otis Redding
Cuando un hombre ama a una mujer	Percy Sledge
Maggie May	Rod Stewart
¡Alto!	Sam Brown
Angel negro	Savage Rose
Libro de fotos	Simply Red
Puente de los espias	T`Pau
Rojo	The Communards
Bailarin Privado	Tina Turner
Tupelo Honey	Van Morrison
Big Willie style	Will Smith

Ejemplo 6: Visualizar el titulo artista y precio de los Cds con un precio superior o igual a 10 unidades de la empresa Columbia

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
        <table border="1">
        <caption><h2>Título, empresa, artista y precio de mis cds con precio mayor  a 10 de Columbia</h2></caption>
                <tr bgcolor="#9acd32">
                        <th>Titulo</th>
                        <th>Empresa</th>
                         <th>Artista</th>
                         <th>Precio</th>
                </tr>
                <xsl:for-each select="catalogo/cds/cd">
                        <xsl:if test="precio &gt; 10">
                                <xsl:if test="empresa='Columbia'">
                                        <tr>
                                                <td><xsl:value-of select="titulo"/></td>
                                                <td><xsl:value-of select="empresa"/></td>
                                                <td><xsl:value-of select="artista"/></td>
                                                <td><xsl:value-of select="precio"/></td>
                                        </tr>
                                </xsl:if>
                        </xsl:if>
                </xsl:for-each>
        </table>
  </body>
  </html>
</xsl:template>
</xsl:stylesheet>
```

Transformation Result:

Close

Título, empresa, artista y precio de mis cds con precio mayor a 10 de Columbia

Titulo	Empresa	Artista	Precio
Empire Burlesque	Columbia	Bob Dylan	10.90

Ejemplo 7: Visualizar el titulo, artista y precio de los Cds y a aquellos cuyo precio es mayor o igual que 10 unidades sombrear la celda del artista en rosa.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
        <html>
        <body>
        <h2>Mi coleccion de Cds</h2>
        <table border="1">
                <tr bgcolor="#9acd32">
                        <th>Titulo</th>
                        <th>Artista</th>
                        <th>Precio</th>
                </tr>
                <xsl:for-each select="catalogo/cds/cd">
                <tr>
                <td><xsl:value-of select="titulo"/></td>
                        <xsl:choose>
                        <xsl:when test="precio>= 10">
                                        <td bgcolor="#ff00ff">
                                <xsl:value-of select="artista"/></td>
                                </xsl:when>
                        <xsl:otherwise>
                                <td><xsl:value-of select="artista"/></td>
                        </xsl:otherwise>
                        </xsl:choose>
                        <td><xsl:value-of select="precio"/></td>
                </tr>
                </xsl:for-each>
        </table>
        </body>
        </html>
</xsl:template>
</xsl:stylesheet>
```

Transformation Result:

Close

Mi coleccion de Cds

Titulo	Artista	Precio
Empire Burlesque	Bob Dylan	10.90
Esconde tu corazon	Bonnie Tyler	9.90
Greatest Hits	Dolly Parton	9.90
Aun tengo el blues	Gary Moore	10.20
Eros	Eros Ramazzotti	9.90
Aguas tranquilas	Bee Gees	10.90
Un poco más	Dr.Hook	8.10
Maggie May	Rod Stewart	8.50
Romanza	Andrea Bocelli	10.80
Cuando un hombre ama a una mujer	Percy Sledge	8.70
Angel negro	Savage Rose	10.90
Convirtiendose en piedra	Many	10.20
Por los Buenos tiempos	Kenny Rogers	8.70
Big Willie style	Will Smith	9.90
Tupelo Honey	Van Morrison	8.20
Soulsville	Jorn Hoel	7.90
Lo mejor de Cat Stevens	Cat Stevens	8.90
¡Alto!	Sam Brown	8.90
Puente de los espias	T`Pau	7.90
Bailarin Privado	Tina Turner	8.90
En medio de la noche	Kim Larsen	7.80
Concierto de Gala de Pavarotti	Luciano Pavarotti	9.90
El muelle de la bahia	Otis Redding	7.90
Libro de fotos	Simply Red	7.20
Rojo	The Communards	7.80
Libera mi corazon	Joe Cocker	8.20

Ejemplo 8: Visualizar los nombres de las canciones de los cds cuyo título empieza por C

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
<body>
        <h2>Nombre de las canciones de los Cds cuyo titulo empeza por C</h2>
                <ul>
                <xsl:variable name="cod" select="//cds/cd/@cod_cd"/>
                <xsl:for-each select="//cds/cd">
                        <xsl:if test="starts-with(titulo,'C')">
                                        <xsl:if test="@cod_cd=$cod">
                                                <li><b>nombre del Cd:</b><xsl:value-of select="titulo"/></li>
                                                <li><b>nombre de las canciones:</b></li>
                                                <ul>
                                                <xsl:for-each select="canciones/cancion">
                                                        <li><xsl:value-of  select="titulo"/></li>
                                                        <br></br>
                                                </xsl:for-each>
                                        </ul>
                                        </xsl:if>
                                </xsl:if>
                </xsl:for-each>
          </ul>
</body>
</html>
</xsl:template>
</xsl:stylesheet>
```

Transformation Result:

Close

Nombre de las canciones de los Cds cuyo titulo empeza por C

- **nombre del Cd:**Cuando un hombre ama a una mujer
- **nombre de las canciones:**
  - Cuando un hombre ama a una mujer
  - Mi adorable
  - Ponme un poco de amor
  - Ámame todo el camino
  - Cuando Ella Me Toca (Nada Más Importa)
  - Estás echando agua sobre un hombre que se está ahogando
  - Ladrón en la noche
  - Me engañaste
  - El amor hace girar al mundo
  - Éxito
  - Ámame como lo dices en serio
- **nombre del Cd:**Convirtiendose en piedra
- **nombre de las canciones:**
  - Convirtiéndose en piedra
  - Fuera de mi camino
  - Sentimientos en mi corazón
  - Ángel
  - Colores en mi mente
  - Ruleta sexual
  - Mantengámonos juntos
  - Salva mi alma
  - Quiero cantar
  - Ningun lugar a donde ir
  - Marywanna
- **nombre del Cd:**Concierto de Gala de Pavarotti
- **nombre de las canciones:**
  - Oh Noche Santa
  - Piedad, Signore
  - Panis Angelicus
  - Qual Giglio Cándido
  - AVE María