

Unidad 7

Gestión y almacenamiento de información en formatos XML

LENGUAJES DE MARCAS Y SISTEMAS DE GESTIÓN DE INFORMACIÓN

ÍNDICE

1. Introducción
2. XML para el almacenamiento de la información
 1. Formas de trabajar con XML para el almacenamiento de la información
3. Lenguajes de consulta y manipulación
4. XQuery
 1. Usos
 2. Terminología XQuery
 3. Reglas básicas de sintaxis
5. Primeros pasos
6. Consultas
 1. Carga del documento en memoria
 2. Selección de nodos básica
 3. Selección de nodos avanzada
 4. Selección del texto de los nodos
 5. Selección de nodos usando IF
 6. Añadiendo elementos y atributos
 7. Consultas en varios ficheros
7. Funciones
8. Referencias
9. Bibliografía

ÍNDICE

1. **Introducción**
2. XML para el almacenamiento de la información
 1. Formas de trabajar con XML para el almacenamiento de la información
3. Lenguajes de consulta y manipulación
4. XQuery
 1. Usos
 2. Terminología XQuery
 3. Reglas básicas de sintaxis
5. Primeros pasos
6. Consultas
 1. Carga del documento en memoria
 2. Selección de nodos básica
 3. Selección de nodos avanzada
 4. Selección del texto de los nodos
 5. Selección de nodos usando IF
 6. Añadiendo elementos y atributos
 7. Consultas en varios ficheros
7. Funciones
8. Referencias
9. Bibliografía

1. Introducción

- ✓ Actualmente, XML se ha convertido en una herramienta de uso cotidiano en los entornos de tratamiento de información y en los entornos de programación web.
- ✓ Este estándar es un metalenguaje que puede ser utilizado para describir la estructura lógica y el contenido de una gran variedad de documentos, y puede ser adaptado para ser utilizado en una gran cantidad de aplicaciones.
- ✓ XML tiene la capacidad de ser universal y extensible, y gracias a ello, se abre un rango ilimitado de usos, desde procesadores de texto, páginas web, comercio electrónico, hasta las más complejas soluciones de almacenamiento en bases de datos.

1. Introducción

- ✓ Sin embargo, a medida que se emplea en un mayor número de proyectos donde la cantidad de datos almacenados aumenta, se comprueba que **las herramientas tradicionales** para manipular dichos datos **no son prácticas**.
- ✓ El principal problema a la hora de procesar colecciones de datos en XML a bajo nivel es la necesidad de escribir una gran cantidad de código para realizar dicho procesamiento. Este código consiste, básicamente, en recorrer un árbol (si usamos el DOM) o detallar una lista de acciones según la etiqueta que se encuentre (si usamos SAX).
- ✓ En este tema se presenta qué tecnologías de alto nivel existen en la actualidad para poder procesar de forma efectiva esas colecciones de datos almacenadas en ficheros XML.

ÍNDICE

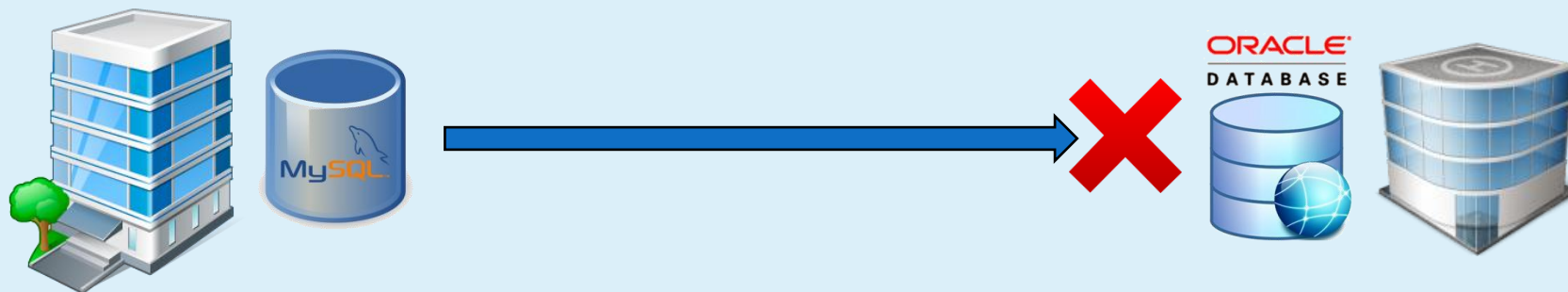
1. Introducción
2. **XML para el almacenamiento de la información**
 1. Formas de trabajar con XML para el almacenamiento de la información
3. Lenguajes de consulta y manipulación
4. XQuery
 1. Usos
 2. Terminología XQuery
 3. Reglas básicas de sintaxis
5. Primeros pasos
6. Consultas
 1. Carga del documento en memoria
 2. Selección de nodos básica
 3. Selección de nodos avanzada
 4. Selección del texto de los nodos
 5. Selección de nodos usando IF
 6. Añadiendo elementos y atributos
 7. Consultas en varios ficheros
7. Funciones
8. Referencias
9. Bibliografía

2. XML para el almacenamiento de la información

- ✓ La gran mayoría de las bases de datos que hay hoy en día están basadas en:
 - a) el modelo relacional
 - b) el modelo orientado a objetos.
- ✓ Dependiendo del problema a abordar el uso de estas bases de datos es el más correcto, pero existen otros casos donde no es así. Por ejemplo, con la llegada de la era internet, la compartición de la información resulta crucial para mejorar las relaciones comerciales, pero con el uso de estas bases de datos surgen una serie de problemas.

2. XML para el almacenamiento de la información

- ✓ Por ejemplo, una empresa X necesita enviarle a la empresa Y información en relación a sus transacciones comerciales. El envío de esta información beneficia a X y también a Y.
- ✓ El problema es que X utiliza unos modelos de datos distintos a los de Y. Además, tampoco es fácil ponerse de acuerdo con el sistema gestor de bases de datos a utilizar. Por tanto, es necesario buscar algo que permita que ambos sistemas, siendo distintos, puedan comunicarse entre sí.



2. XML para el almacenamiento de la información

- ✓ La solución es **XML**.
- ✓ XML permite definir de manera rápida e intuitiva una representación de la información que ambas empresas desean compartir.
- ✓ La empresa X usará su sistema gestor de bases de datos para exportar sus datos a XML y se los remitirá a Y. La empresa Y recibirá el fichero XML y será capaz de manejar dichos datos gracias a las ventajas que ofrece XML.
- ✓ Por tanto, para las empresas que deben trabajar con datos que provienen de sistemas incompatibles, XML puede servir como una tecnología común para el transporte de los mismos en torno a un formato neutral.
- ✓ Además, XML puede manejar toda clases de datos, incluyendo texto, imágenes y sonido lo que le permite manejar cualquier caso especial de modelado de estructuras complejas.

2. XML para el almacenamiento de la información

2.1. Formas de trabajar con XML para el almacenamiento de la información

- ✓ En el ejemplo propuesto XML se utiliza como puente entre los dos SGBD porque ambos sistemas son capaces de crear/interpretar el contenido de dicho XML e integrarlo en su modelo de datos. Pero internamente no trabajan con dicho XML en sí. Esto permite poner los datos de XML en un área de trabajo donde la búsqueda, el análisis, la actualización y la salida pueden realizarse en un ambiente más manejable, más sistemático y conocido. Esta es la forma más habitual de trabajar, pero no es la única.
- ✓ La otra forma de trabajar consiste en usar bases datos que almacenan internamente al XML en su formato nativo y trabajan sobre él.

2. XML para el almacenamiento de la información

2.1. Formas de trabajar con XML para el almacenamiento de la información

- ✓ Por tanto hay que distinguir entre dos tipos de bases de datos.
 - a) Si el XML no se almacena internamente como XML, llamaremos a eso una **"base de datos que soporta XML"** (XML-enabled). Algunos ejemplos de bases de datos que soportan XML:
 - IBM DB2 (pureXML)
 - Microsoft SQL Server
 - Oracle Database
 - PostgreSQL
 - b) Si el XML se almacena internamente como XML, la llamaremos una **"base de datos de XML nativa"** (Native XML). Algunos ejemplos de bases de datos XML nativas:
 - eXcelon XIS lite
 - BaseX
 - eXist
 - dbXML
 - Altova XMLSpy

2. XML para el almacenamiento de la información

2.1. Formas de trabajar con XML para el almacenamiento de la información

- ✓ Los puristas afirman que solamente las bases de datos que almacenan internamente al XML en su formato nativo merecen llamarse "base de datos de XML".
- ✓ Otros en cambio afirman que si un producto puede almacenar y recuperar XML en él y es una base de datos, entonces es una base de datos XML, sin importar cómo se almacenan los datos.
- ✓ Hoy en día hay cierta negativa a moverse a una base de datos nativa de XML, ya que algunas características (especialmente la escalabilidad) no se han probado. Es por ello, que las bases de datos relacionales y las bases de datos orientadas a objetos con soporte para XML son las opciones más usadas actualmente. Las empresas no desean arriesgarse a migrar hacia una base de datos XML nativa, si no tienen la necesidad de hacerlo.

ÍNDICE

1. Introducción
2. XML para el almacenamiento de la información
 1. Formas de trabajar con XML para el almacenamiento de la información
3. **Lenguajes de consulta y manipulación**
4. XQuery
 1. Usos
 2. Terminología XQuery
 3. Reglas básicas de sintaxis
5. Primeros pasos
6. Consultas
 1. Carga del documento en memoria
 2. Selección de nodos básica
 3. Selección de nodos avanzada
 4. Selección del texto de los nodos
 5. Selección de nodos usando IF
 6. Añadiendo elementos y atributos
 7. Consultas en varios ficheros
7. Funciones
8. Referencias
9. Bibliografía

3. Lenguajes de consulta y manipulación

- ✓ Siempre que se trabaja con un SGBD se debe emplear un lenguaje que permita crear la base de datos y sus tablas, y que permita consultar y manipular los datos almacenados.
- ✓ Cuando se trabaja con base de datos relacionales, el lenguaje de consulta estructurado más popular es **SQL**. Este lenguaje permite, de forma declarativa, acceder a las bases de datos relacionales y operar con ellas. SQL es el lenguaje que en la actualidad se considera estándar de facto pues la inmensa mayoría de los SGBD lo implementan.
- ✓ Cuando se va a trabajar con documentos XML usando tanto "bases de datos XML nativas" como "bases de datos que soportan XML", se necesita también un lenguaje que permita consultar y manipular los datos almacenados. Este lenguaje se llama **XQuery**.

3. Lenguajes de consulta y manipulación

- ✓ XQuery es un lenguaje de consulta similar a SQL que permite recorrer los documentos XML de manera que se pueda extraer y manipular la información contenida en el mismo.
- ✓ Es un lenguaje muy sencillo que no requiere de conocimientos de programación avanzados y es compatible con muchas de las tecnologías que estandarizada W3C.
- ✓ XQuery trabaja sobre el **árbol XML** que se genera a partir de cualquier fichero basado en XML. Para ello hace uso de **XPath**.

ÍNDICE

1. Introducción
2. XML para el almacenamiento de la información
 1. Formas de trabajar con XML para el almacenamiento de la información
3. Lenguajes de consulta y manipulación
4. **XQuery**
 1. Usos
 2. Terminología XQuery
 3. Reglas básicas de sintaxis
5. Primeros pasos
6. Consultas
 1. Carga del documento en memoria
 2. Selección de nodos básica
 3. Selección de nodos avanzada
 4. Selección del texto de los nodos
 5. Selección de nodos usando IF
 6. Añadiendo elementos y atributos
 7. Consultas en varios ficheros
7. Funciones
8. Referencias
9. Bibliografía

4. XQuery

- ✓ XQuery es un lenguaje de consulta diseñado para colecciones de datos XML que proporciona los medios para extraer y manipular directamente la información de documentos XML.
- ✓ XQuery utiliza **expresiones XPath** para acceder a determinadas partes del documento XML y expresiones conocidas como **expresiones FLWOR**.
 - Las expresiones XPath funcionan en base a una serie de sentencias que permiten identificar qué parte del documento XML se quiere acceder o recorrer.
 - Las expresiones FLWOR toman su nombre de los 5 tipos de sentencias de las que pueden estar compuestas: FOR, LET, WHERE, ORDER BY y RETURN.
- ✓ También incluye la posibilidad de construir nuevos documentos XML (o HTML) a partir de los resultados de la consulta.

4. XQuery

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

XQuery Example

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

4. XQuery

XQuery is a W3C Recommendation

XQuery is compatible with several W3C standards, such as XML, Namespaces, XSLT, XPath, and XML Schema.

XQuery 1.0 became a W3C Recommendation January 23, 2007.

<https://www.w3.org/TR/xquery/all/>

4. XQuery

4.1. Usos

- ✓ XQuery se emplea para:
 - Extraer información de una base de datos para usarla en un Servicio Web.
 - Generar un resumen de la información almacenada en una base de datos XML.
 - Realizar búsquedas textuales en la web y compilar los resultados de la misma.
 - Seleccionar y transformar datos de XML a XHTML de forma que se puedan publicar en la Web.
 - Obtener datos desde diferentes fuentes con vistas a ser integradas por la aplicación.
 - Dividir un documento XML que representa una serie de múltiples transacciones en varios documentos XML, uno por cada transacción.

4. XQuery

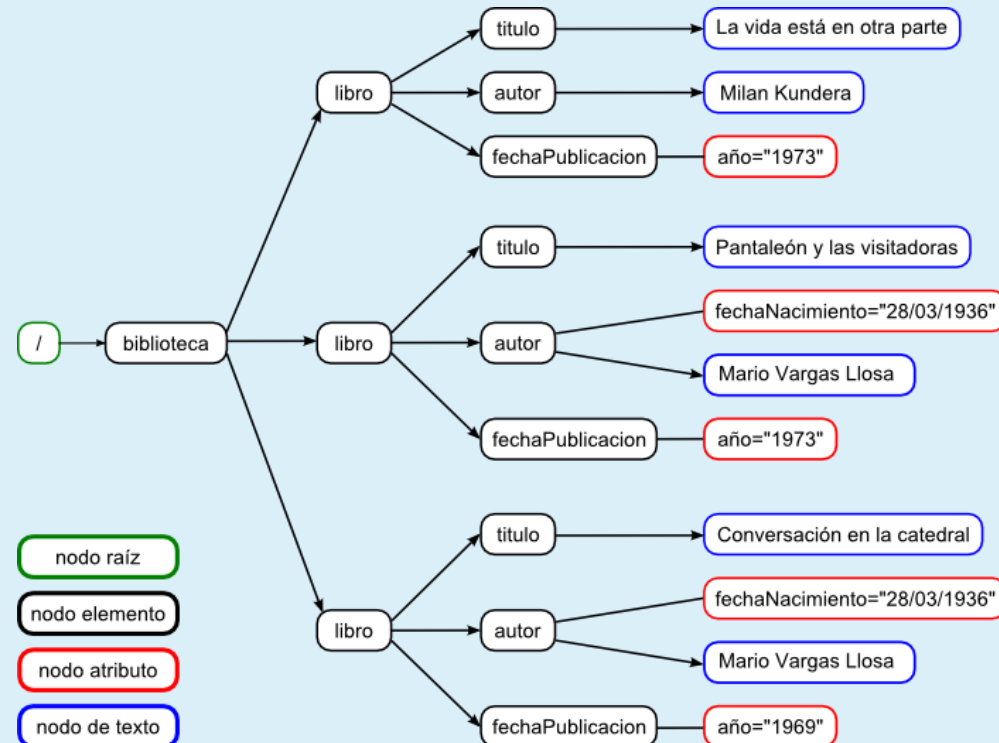
4.2. Terminología XQuery

✓ Dado que XQuery usa XPath, los conceptos manejados por esta tecnología ya los conocemos:

- Nodos o items.
- Valores atómicos

- Nodo raíz
- Nodo elemento
- Nodo atributo
- Nodo texto

- Padre
- Hijos
- Hermanos
- Ancestros



4. XQuery

4.3. Reglas básicas de sintaxis

✓ Reglas básicas:

XQuery Basic Syntax Rules

Some basic syntax rules:

- XQuery is case-sensitive
- XQuery elements, attributes, and variables must be valid XML names
- An XQuery string value can be in single or double quotes
- An XQuery variable is defined with a \$ followed by a name, e.g. \$bookstore
- XQuery comments are delimited by (: and :), e.g. (: XQuery Comment :)

```
1 xquery version "1.0";  
2 (: Comentarios en XQuery :)
```

4. XQuery

4.3. Reglas básicas de sintaxis

✓ 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 XQuery los espacios de nombres y sus prefijos usando la instrucción 'declare':

declare namespace prefijo="espacio-de-nombres".

Por ejemplo, para declarar el espacio de nombres "http://www.ejemplos.com" y asignarle el prefijo "eje" se debe hacer así:

declare namespace eje="http://www.ejemplos.com"

El nombre del espacio de nombres tiene que coincidir con el espacio de nombres del documento XML sobre el que se realiza la consulta. El prefijo puede ser diferente.

4. XQuery

4.3. Reglas básicas de sintaxis

✓ Comparaciones en XQuery:

XQuery Comparisons

In XQuery there are two ways of comparing values.

1. General comparisons: =, !=, <, <=, >, >=
2. Value comparisons: eq, ne, lt, le, gt, ge

The difference between the two comparison methods are shown below.

The following expression returns true if any q attributes have a value greater than 10:

```
$bookstore//book/@q > 10
```

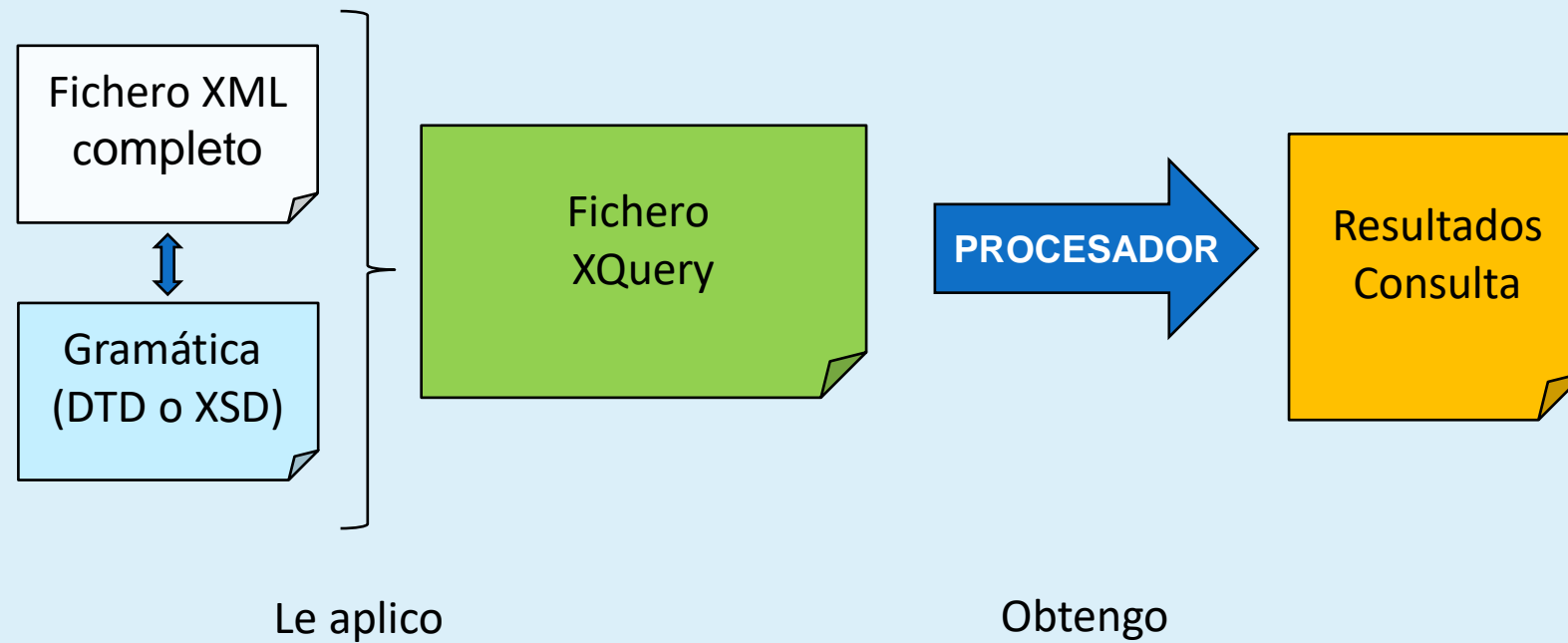
The following expression returns true if there is only one q attribute returned by the expression, and its value is greater than 10. If more than one q is returned, an error occurs:

```
$bookstore//book/@q gt 10
```


ÍNDICE

1. Introducción
2. XML para el almacenamiento de la información
 1. Formas de trabajar con XML para el almacenamiento de la información
3. Lenguajes de consulta y manipulación
4. XQuery
 1. Usos
 2. Terminología XQuery
 3. Reglas básicas de sintaxis
5. **Primeros pasos**
6. Consultas
 1. Carga del documento en memoria
 2. Selección de nodos básica
 3. Selección de nodos avanzada
 4. Selección del texto de los nodos
 5. Selección de nodos usando IF
 6. Añadiendo elementos y atributos
 7. Consultas en varios ficheros
7. Funciones
8. Referencias
9. Bibliografía

5. Primeros pasos



5. Primeros pasos

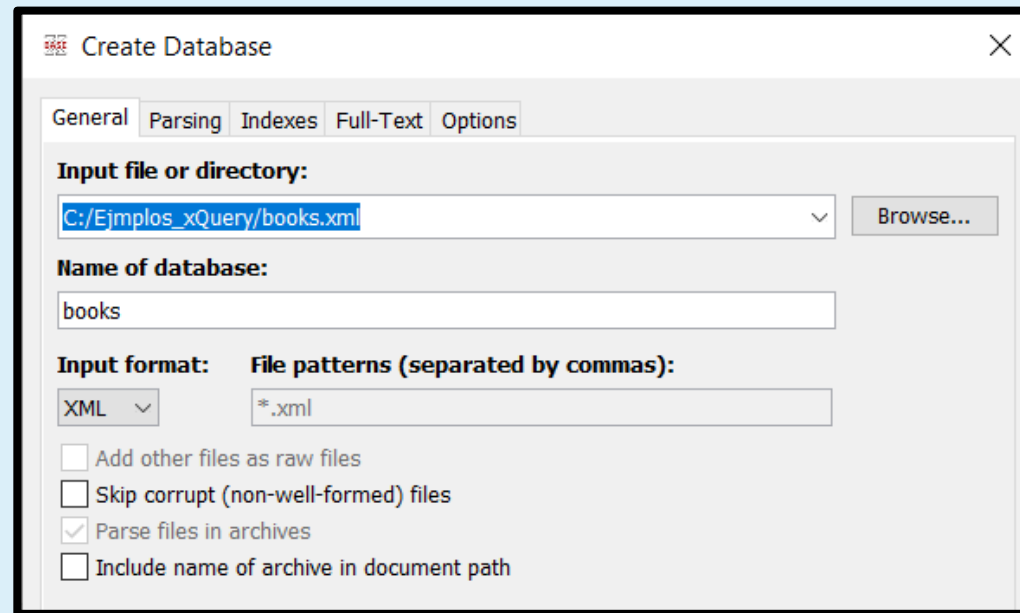
Dado el siguiente fichero XML con nombre '**books.xml**':

<https://www.w3schools.com/xml/books.xml>

```
3  <bookstore>
4  <book category="COOKING">
5    <title lang="en">Everyday Italian</title>
6    <author>Giada De Laurentiis</author>
7    <year>2005</year>
8    <price>30.00</price>
9  </book>
10 <book category="CHILDREN">
11   <title lang="en">Harry Potter</title>
12   <author>J K. Rowling</author>
13   <year>2005</year>
14   <price>29.99</price>
15 </book>
16 <book category="WEB">
17   <title lang="en">XQuery Kick Start</title>
18   <author>James McGovern</author>
19   <author>Per Bothner</author>
20   <author>Kurt Cagle</author>
21   <author>James Linn</author>
22   <author>Vaidyanathan Nagarajan</author>
23   <year>2003</year>
24   <price>49.99</price>
25 </book>
```

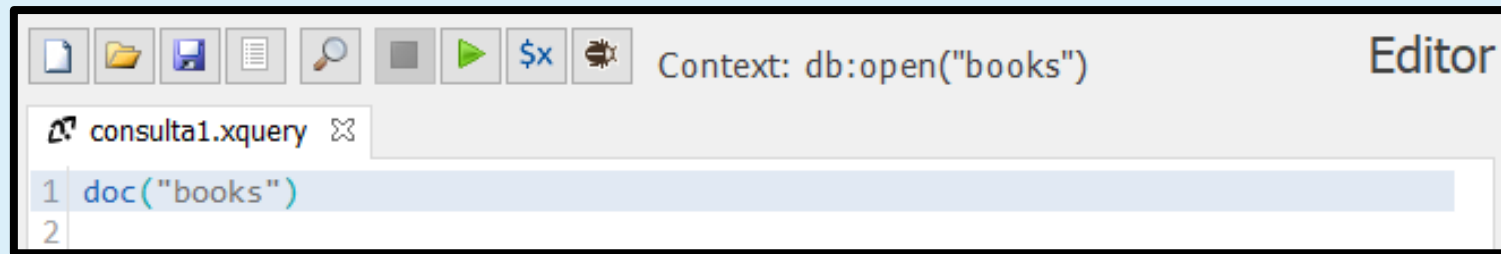
5. Primeros pasos

- ✓ Vamos a crear un fichero XQuery usando la herramienta BaseX
- ✓ En primer lugar, indicamos de dónde tenemos que obtener los datos:



5. Primeros pasos

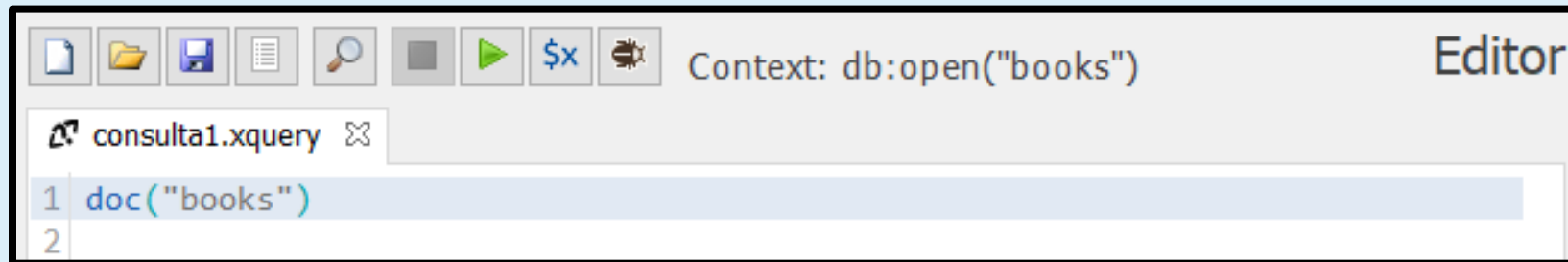
- ✓ Escribimos la consulta en el Editor.
 - Para extraer datos de documentos XML, XQuery utiliza la función ***doc()***



- ✓ Guardamos el fichero con nombre "consulta1.xquery".

5. Primeros pasos

- ✓ Ejecutamos la consulta:



5. Primeros pasos

✓ Obtenemos:



```
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J. K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">XQuery Kick Start</title>
    <author>James McGovern</author>
    <author>Per Bothner</author>
    <author>Kurt Cagle</author>
    <author>James Linn</author>
    <author>Vaidyanathan Nagarajan</author>
    <year>2003</year>
    <price>49.99</price>
  </book>
  <book category="web" cover="paperback">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

5. Primeros pasos

- ✓ ¿Qué ocurre en la consulta 2?

```
1 doc("books")/bookstore/book/title
```


5. Primeros pasos

- ✓ ¿Qué ocurre en la consulta 2?

```
1 doc("books")/bookstore/book/title
```





Result

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

5. Primeros pasos

- ✓ ¿Qué ocurre en la consulta 3?

```
1 doc("books")/bookstore/book[price<30]
```

5. Primeros pasos

- ✓ ¿Qué ocurre en la consulta 3?

```
1 doc("books")/bookstore/book[price<30]
```





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

Result

5. Primeros pasos

- ✓ ¿Qué ocurre en la consulta 4?

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





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

Result

ÍNDICE

1. Introducción
2. XML para el almacenamiento de la información
 1. Formas de trabajar con XML para el almacenamiento de la información
3. Lenguajes de consulta y manipulación
4. XQuery
 1. Usos
 2. Terminología XQuery
 3. Reglas básicas de sintaxis
5. Primeros pasos
6. Consultas
 1. Carga del documento en memoria
 2. Selección de nodos básica
 3. Selección de nodos avanzada
 4. Selección del texto de los nodos
 5. Selección de nodos usando IF
 6. Añadiendo elementos y atributos
 7. Consultas en varios ficheros
7. Funciones
8. Referencias
9. Bibliografía

6. Consultas

Dado el siguiente fichero XML:

<https://www.w3schools.com/xml/books.xml>

```
3  <bookstore>
4  <book category="COOKING">
5    <title lang="en">Everyday Italian</title>
6    <author>Giada De Laurentiis</author>
7    <year>2005</year>
8    <price>30.00</price>
9  </book>
10 <book category="CHILDREN">
11   <title lang="en">Harry Potter</title>
12   <author>J K. Rowling</author>
13   <year>2005</year>
14   <price>29.99</price>
15 </book>
16 <book category="WEB">
17   <title lang="en">XQuery Kick Start</title>
18   <author>James McGovern</author>
19   <author>Per Bothner</author>
20   <author>Kurt Cagle</author>
21   <author>James Linn</author>
22   <author>Vaidyanathan Nagarajan</author>
23   <year>2003</year>
24   <price>49.99</price>
25 </book>
```

6. Consultas

Dado el siguiente fichero XML:

<https://www.w3schools.com/xml/books.xml>

Puesto que este ejemplo es del w3schools, se va a obviar la codificación en los ficheros XQuery usados en los siguientes ejemplos.

```
3  <bookstore>
4  <book category="COOKING">
5    <title lang="en">Everyday Italian</title>
6    <author>Giada De Laurentiis</author>
7    <year>2005</year>
8    <price>30.00</price>
9  </book>
10 <book category="CHILDREN">
11   <title lang="en">Harry Potter</title>
12   <author>J K. Rowling</author>
13   <year>2005</year>
14   <price>29.99</price>
15 </book>
16 <book category="WEB">
17   <title lang="en">XQuery Kick Start</title>
18   <author>James McGovern</author>
19   <author>Per Bothner</author>
20   <author>Kurt Cagle</author>
21   <author>James Linn</author>
22   <author>Vaidyanathan Nagarajan</author>
23   <year>2003</year>
24   <price>49.99</price>
25 </book>
```

6. Consultas

6.1. Carga del documento en memoria

```
doc("books.xml")
```


6. Consultas

6.2. Selección de nodos básica

- ✓ Devuelven nodos completos. No devuelven el contenido del nodo.

```
doc("books.xml")/bookstore/book/title
```

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

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

6. Consultas

6.3. Selección de nodos avanzada

✓ Primer ejemplo:

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

- La cláusula "for" enlaza una variable con cada ítem que se devuelve por la expresión "in".
- Esta cláusula produce una iteración.
- Puede haber múltiples cláusulas "for" en una misma expresión.


6. Consultas

6.3. Selección de nodos avanzada

✓ Primer ejemplo:

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





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


Result

¿A qué es equivalente?

6. Consultas

6.3. Selección de nodos avanzada

✓ Segundo ejemplo:



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

FLWOR is an acronym for "For, Let, Where, Order by, Return".

The **for** clause selects all book elements under the bookstore element into a variable called \$x.

The **where** clause selects only book elements with a price element with a value greater than 30.

The **order by** clause defines the sort-order. Will be sort by the title element.

The **return** clause specifies what should be returned. Here it returns the title elements.

6. Consultas

6.3. Selección de nodos avanzada

✓ Segundo ejemplo:

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

Importante: No podemos devolver varios elementos de distinto tipo, es decir, no podemos devolver el "titulo", el "año" o el "precio" a la vez.

Aunque como veremos más adelante, si se integran dentro de una etiqueta principal que los contenga, sí se puede.

6. Consultas

6.3. Selección de nodos avanzada

✓ Tercer ejemplo:

```
1 for $x at $cont in doc("books.xml")/bookstore/book
2 return $cont
```

○ ¿Qué muestra?

Nota: Esto tiene otras utilidades que veremos más adelante.

6. Consultas

6.3. Selección de nodos avanzada

✓ Cuarto ejemplo:

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

- La cláusula "let" permite hacer asignaciones de variables, permitiendo evitar el tener que repetir la misma expresión varias veces.
- La cláusula "let" no provoca una iteración.
- ¿Qué muestra?

6. Consultas

6.3. Selección de nodos avanzada

✓ Cuarto ejemplo:

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





Result

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


6. Consultas

6.3. Selección de nodos avanzada

- ✓ Cuarto ejemplo:
 - **FOR vs LET**

```
for $x in (1 to 5)
return <test>{$x}</test>
```

Result:

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

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

Result:

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

6. Consultas

6.3. Selección de nodos avanzada

✓ Quinto ejemplo:

```
1 for $x in doc("books.xml")/bookstore/book
2 where $x/price>30 and $x/price<100
3 order by $x/@category, $x/title descending
4 return $x/title
```

- Múltiples criterios en ***where*** y ***order by***

6. Consultas

6.4. Selección del texto de los nodos

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





XQuery Kick Start
Learning XML

Result

- **Nota:** Existen otras funciones de XPath, como "string" o "text", para obtener el valor.

6. Consultas

6.5. Selección de nodos usando IF

```
1 for $x in doc("books.xml")/bookstore/book
2 return if ($x/@category="CHILDREN")
3 then $x/price
4 else $x/year
```

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

6. Consultas

6.6. Añadiendo elementos y atributos

- ✓ Es posible crear consultas XQuery que devuelvan los resultados entre los elementos y atributo que definamos. Estos elementos y atributos pueden ser definidos por nosotros o pueden ser elementos XHTML. Es decir, podemos obtener resultados similares a los obtenidos con XSLT.
- ✓ Para ello debemos diferenciar:
 - El código literal que se debe escribir en la salida tal cual lo escribamos.
 - El código que debe ser procesado como consulta XQuery.
- ✓ Esto se hace usando llaves **{ }**

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Primer ejemplo:

```
1      xquery version "1.0";
2      <html>
3      <head>
4          <title>Ejemplo</title>
5      </head>
6      <body>
7          <h1>Bookstore</h1>
8          <ul>
9              {
10                 for $x in doc("books.xml")/bookstore/book
11                 order by $x/title
12                 return <li>{data($x/title)}. Categoria: {data($x/@category)}</li>
13             }
14          </ul>
15      </body>
16      </html>
```

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Primer ejemplo:

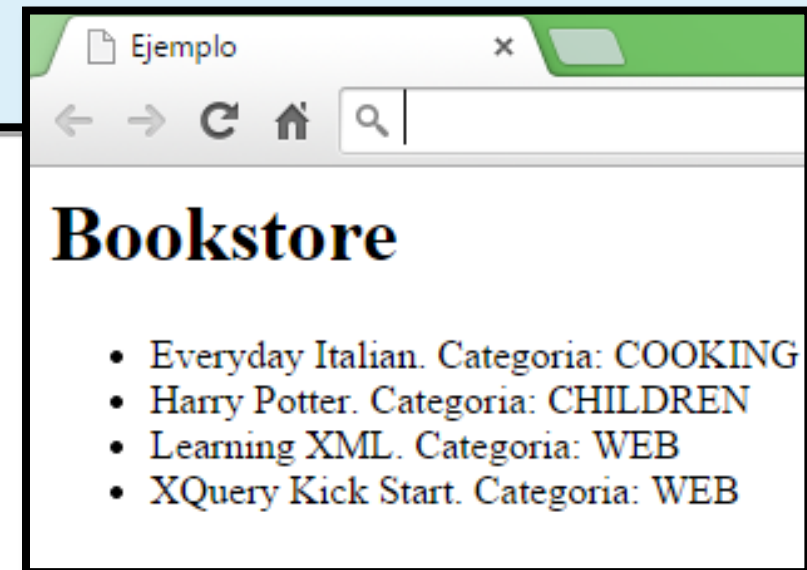
```
1 xquery version "1.0";
2 <html>
3 <head>
4   <title>Ejemplo</title>
5 </head>
6 <body>
7   <h1>Bookstore</h1>
8   <ul>
9     {
10      for $x in doc("books.xml")/bookstore/book
11      order by $x/title
12      return <li>{data($x/title)}. Categoria: {data($x/@category)}</li>
13    }
14  </ul>
15 </body>
16 </html>
```

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Primer ejemplo:

```
1  xquery version "1.0";
2  <html>
3  <head>
4    <title>Ejemplo</title>
5  </head>
6  <body>
7    <h1>Bookstore</h1>
8    <ul>
9    {
10   for $x in doc("books.xml")/bookstore/book
11   order by $x/title
12   return <li>{data($x/title)}. Categoria: {data($x/@category)}</li>
13   }
14  </ul>
15 </body>
16 </html>
```



6. Consultas

6.6. Añadiendo elementos y atributos

✓ Segundo ejemplo:

```
1      xquery version "1.0";
2      <html>
3      <head>
4          <title>Ejemplo</title>
5      </head>
6      <body>
7          <h1>Bookstore</h1>
8          <ul>
9              {
10                 for $x in doc("books.xml")/bookstore/book
11                 order by $x/title
12                 return <li class="{data($x/@category)}">{data($x/title)}</li>
13             }
14          </ul>
15      </body>
16      </html>
```

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Segundo ejemplo:

```
1  xquery version "1.0";
2  <html>
3  <head>
4    <title>Ejemplo</title>
5  </head>
6  <body>
7    <h1>Bookstore</h1>
8    <ul>
9      {
10     for $x in doc("books.xml")/bookstore/book
11     order by $x/title
12     return <li class="{data($x/@category)}">{data($x/title)}</li>
13   }
14 </ul>
15 </body>
16 </html>
```

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Tercer ejemplo:

```
1      xquery version "1.0";
2      <minibook>
3      {
4          for $x in doc("books.xml")/bookstore/book
5          return <book><titulo>{data($x/title)}</titulo>{$x/year}</book>
6      }
7      </minibook>
```

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Tercer ejemplo:

```
1  xquery version "1.0";
2  <minibook>
3  {
4    for $x in doc("books.xml")/bookstore/book
5    return <book><titulo>{data($x/title)}</titulo>
6  }
7  </minibook>
```

```
<?xml version="1.0"?>
- <minibook>
  - <book>
    <titulo>Everyday Italian</titulo>
    <year>2005</year>
  </book>
  - <book>
    <titulo>Harry Potter</titulo>
    <year>2005</year>
  </book>
  - <book>
    <titulo>XQuery Kick Start</titulo>
    <year>2003</year>
  </book>
  - <book>
    <titulo>Learning XML</titulo>
    <year>2003</year>
  </book>
</minibook>
```

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Cuarto ejemplo:

```
1      xquery version "1.0";
2      <minibook>
3      {
4          for $x in doc("books.xml")/bookstore/book
5          return if ($x/@category="CHILDREN")
6          then <book><titulo>{data($x/title)}</titulo>{$x/year}<child/></book>
7          else <book><titulo>{data($x/title)}</titulo>{$x/year}<adult/></book>
8      }
9      </minibook>
```

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Cuarto ejemplo:

```
1  xquery version "1.0";
2  <minibook>
3  {
4    for $x in doc("books.xml")/bookstore/book
5    return if ($x/@category="CHILDREN")
6    then <book><titulo>{data($x/title)}</titulo>{$x/year}<ch
7    else <book><titulo>{data($x/title)}</titulo>{$x/year}<ad
8  }
9  </minibook>
```

```
<?xml version="1.0"?>
- <minibook>
  - <book>
    <titulo>Everyday Italian</titulo>
    <year>2005</year>
    <adult/>
  </book>
  - <book>
    <titulo>Harry Potter</titulo>
    <year>2005</year>
    <child/>
  </book>
  - <book>
    <titulo>XQuery Kick Start</titulo>
    <year>2003</year>
    <adult/>
  </book>
  - <book>
    <titulo>Learning XML</titulo>
    <year>2003</year>
    <adult/>
  </book>
</minibook>
```

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Quinto ejemplo:

```
1  xquery version "1.0";
2  <minibook>
3  {
4    for $x in doc("books.xml")/bookstore/book
5    let $c := $x/author
6    return if (count($c)>1)
7    then <book><titulo>{data($x/title)}</titulo><autores>{data($c)}</autores></book>
8    else <book><titulo>{data($x/title)}</titulo><autor>{data($c)}</autor></book>
9  }
10 </minibook>
```

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Quinto ejemplo:

```
1 xquery version "1.0";
2 <minibook>
3
4   <?xml version="1.0"?>
5   - <minibook>
6     - <book>
7       <titulo>Everyday Italian</titulo>
8       <autor>Giada De Laurentiis</autor>
9     </book>
10    - <book>
      <titulo>Harry Potter</titulo>
      <autor>J K. Rowling</autor>
    </book>
    - <book>
      <titulo>XQuery Kick Start</titulo>
      <autores>James McGovernPer BothnerKurt CagleJames LinnVaidyanathan Nagarajan</autores>
    </book>
    - <book>
      <titulo>Learning XML</titulo>
      <autor>Erik T. Ray</autor>
    </book>
  </minibook>
```


6. Consultas

6.6. Añadiendo elementos y atributos

✓ Sexto ejemplo:

```
1  xquery version "1.0";
2  <minibook>
3  {
4    for $x in doc("books.xml")/bookstore/book
5    return if (count($x/author)>1)
6    then <book><titulo>{data($x/title)}</titulo><autores>{data($x/author)}</autores></book>
7    else <book><titulo>{data($x/title)}</titulo><autor>{data($x/author)}</autor></book>
8  }
9  </minibook>
```

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Sexto ejemplo:

```
1 xquery version "1.0";
2 <minibook>
3
4   <?xml version="1.0"?>
5   - <minibook>
6     - <book>
7       <titulo>Everyday Italian</titulo>
8       <autor>Giada De Laurentiis</autor>
9     </book>
10    - <book>
11      <titulo>Harry Potter</titulo>
12      <autor>J K. Rowling</autor>
13    </book>
14    - <book>
15      <titulo>XQuery Kick Start</titulo>
16      <autores>James McGovernPer BothnerKurt CagleJames LinnVaidyanathan Nagarajan</autores>
17    </book>
18    - <book>
19      <titulo>Learning XML</titulo>
20      <autor>Erik T. Ray</autor>
21    </book>
22  </minibook>
```

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Séptimo ejemplo:

```
1      xquery version "1.0";  
2      <minibook>  
3      {  
4          for $x at $i in doc("books.xml")/bookstore/book  
5          return <book num="{ $i }">{$x/title}{$x/price}</book>  
6      }  
7      </minibook>
```

6. Consultas

6.6. Añadiendo elementos y atributos

✓ Séptimo ejemplo:

```
1      xquery version "1.0";
2      <minibook>
3      {
4          for $x at $i in doc("books.xml")/bo
5          return <book num="{ $i }">{ $x/title }
6      }
7      </minibook>
```

```
<?xml version="1.0"?>
- <minibook>
  - <book num="1">
    <title lang="en">Everyday Italian</title>
    <price>30.00</price>
  </book>
  - <book num="2">
    <title lang="en">Harry Potter</title>
    <price>29.99</price>
  </book>
  - <book num="3">
    <title lang="en">XQuery Kick Start</title>
    <price>49.99</price>
  </book>
  - <book num="4">
    <title lang="en">Learning XML</title>
    <price>39.95</price>
  </book>
</minibook>
```

6. Consultas

6.7. Consultas en varios ficheros

- ✓ Suponemos que en el fichero "books.xml", cada libro tiene un atributo "codigo" que identifica al libro de forma unívoca:

```
<bookstore>
  <book category="COOKING" codigo='1'>
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
```

- ✓ Además, tenemos el fichero "librosAlmacen.xml" con el siguiente contenido:

```
2 <almacen>
3   <comprados>
4     <codigo>1</codigo>
5     <codigo>2</codigo>
6   </comprados>
7   <pendientes>
8     <codigo>3</codigo>
9     <codigo>4</codigo>
10  </pendientes>
11 </almacen>
```

6. Consultas

6.7. Consultas en varios ficheros

- ✓ Podemos seleccionar nodos de ambos ficheros en una misma consulta XQuery:

```
1 xquery version "1.0";  
2 for $book in doc("books.xml")/bookstore/book  
3 for $comp in doc("librosAlmacen.xml")/almacen/comprados  
4 where $book/@codigo=$comp/codigo  
5 return $book/title
```

ÍNDICE

1. Introducción
2. XML para el almacenamiento de la información
 1. Formas de trabajar con XML para el almacenamiento de la información
3. Lenguajes de consulta y manipulación
4. XQuery
 1. Usos
 2. Terminología XQuery
 3. Reglas básicas de sintaxis
5. Primeros pasos
6. Consultas
 1. Carga del documento en memoria
 2. Selección de nodos básica
 3. Selección de nodos avanzada
 4. Selección del texto de los nodos
 5. Selección de nodos usando IF
 6. Añadiendo elementos y atributos
 7. Consultas en varios ficheros
7. **Funciones**
8. Referencias
9. Bibliografía

7. Funciones

✓ Funciones predefinidas:

XQuery Functions

XQuery includes over 100 built-in functions. There are functions for string values, numeric values, date and time comparison, node and QName manipulation, sequence manipulation, Boolean values, and more. You can also define your own functions in XQuery.

XQuery Built-in Functions

The URI of the XQuery function namespace is:
<http://www.w3.org/2005/02/xpath-functions>

The default prefix for the function namespace is fn:.

Tip: Functions are often called with the fn: prefix, such as fn:string(). However, since fn: is the default prefix of the namespace, the function names do not need to be prefixed when called.

The reference of all the built-in XQuery 1.0 functions is located in our XPath tutorial.

7. Funciones

✓ Funciones predefinidas:

Example 1: In an element

```
<name>{upper-case($booktitle)}</name>
```

Example 2: In the predicate of a path expression

```
doc("books.xml")/bookstore/book[substring(title,1,5)='Harry']
```

Example 3: In a let clause

```
let $name := (substring($booktitle,1,4))
```

7. Funciones

✓ Funciones definidas por el usuario:

Syntax

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

Notes on user-defined functions:

- Use the declare function keyword
- The name of the function must be prefixed
- The data type of the parameters are mostly the same as the data types defined in XML Schema
- The body of the function must be surrounded by curly braces

7. Funciones

- ✓ **Funciones definidas por el usuario:**
 - Ejemplo

```
declare function local:minPrice($p as xs:decimal?,$d as xs:decimal?)
as xs:decimal?
{
  let $disc := ($p * $d) div 100
  return ($p - $disc)
};
```

Below is an example of how to call the function above:

```
<minPrice>{local:minPrice($book/price,$book/discount)}</minPrice>
```

ÍNDICE

1. Introducción
2. XML para el almacenamiento de la información
 1. Formas de trabajar con XML para el almacenamiento de la información
3. Lenguajes de consulta y manipulación
4. XQuery
 1. Usos
 2. Terminología XQuery
 3. Reglas básicas de sintaxis
5. Primeros pasos
6. Consultas
 1. Carga del documento en memoria
 2. Selección de nodos básica
 3. Selección de nodos avanzada
 4. Selección del texto de los nodos
 5. Selección de nodos usando IF
 6. Añadiendo elementos y atributos
 7. Consultas en varios ficheros
7. Funciones
8. **Referencias**
9. Bibliografía

8. Referencias

- ✓ Recuerda que XQuery se construye sobre XPath y por tanto en este punto, lo que vamos a presentar son funciones y operadores de XPath que junto con XQuery crean consultas muy potentes.
 - a) Operadores: https://www.w3schools.com/xml/xpath_operators.asp
 - b) Funciones: https://www.w3schools.com/xml/xsl_functions.asp

8. Referencias

a) Operadores: https://www.w3schools.com/xml/xpath_operators.asp

Operator	Description	Example
	Computes two node-sets	//book //cd
+	Addition	6 + 4
-	Subtraction	6 - 4
*	Multiplication	6 * 4
div	Division	8 div 4
=	Equal	price=9.80
!=	Not equal	price!=9.80
<	Less than	price<9.80
<=	Less than or equal to	price<=9.80
>	Greater than	price>9.80
>=	Greater than or equal to	price>=9.80
or	or	price=9.80 or price=9.70
and	and	price>9.00 and price<9.90
mod	Modulus (division remainder)	5 mod 2

8. Referencias

b) Funciones: https://www.w3schools.com/xml/xsl_functions.asp

Functions Reference

- [Accessor](#)
- [Error and Trace](#)
- [Numeric](#)
- [String](#)
- [AnyURI](#)
- [Boolean](#)
- [Duration/Date/Time](#)
- [QName](#)
- [Node](#)
- [Sequence](#)
- [Context](#)



The default prefix for the function namespace is fn:
The URI of the function namespace is: <http://www.w3.org/2005/xpath-functions>

ÍNDICE

1. Introducción
2. XML para el almacenamiento de la información
 1. Formas de trabajar con XML para el almacenamiento de la información
3. Lenguajes de consulta y manipulación
4. XQuery
 1. Usos
 2. Terminología XQuery
 3. Reglas básicas de sintaxis
5. Primeros pasos
6. Consultas
 1. Carga del documento en memoria
 2. Selección de nodos básica
 3. Selección de nodos avanzada
 4. Selección del texto de los nodos
 5. Selección de nodos usando IF
 6. Añadiendo elementos y atributos
 7. Consultas en varios ficheros
7. Funciones
8. Referencias
9. **Bibliografía**

9. Bibliografía

- ✓ **Lenguajes de Marcas y Sistemas de Gestión de Información**
Sánchez, F. J. y otros. Editorial: RA-MA.
- ✓ **w3schools.com: XML Tutorial:** <https://www.w3schools.com/xml/>
- ✓ **w3schools - XQuery Tutorial:** https://www.w3schools.com/xml/xquery_intro.asp
- ✓ **w3c: Extensible Markup Language (XML):** <https://www.w3.org/XML/>
- ✓ **w3c: XQuery 1.0: An XML Query Language (Second Edition):**
<https://www.w3.org/TR/xquery/all/>
- ✓ **JorgeSanchez.net: LMSGI:** <http://jorgesanchez.net/#web/lmsgi.html>
- ✓ **XQuery – Lenguajes y Sistemas Informáticos (Universidad de Sevilla):**
<https://www.lsi.us.es/docs/informes/LSI-2005-02.pdf>
- ✓ **Revista Científica Visión de Futuro: XML y Bases de Datos:**
https://revistacientifica.fce.unam.edu.ar/index.php?option=com_content&view=article&id=1:xml-y-bases-de-datos&catid=3:notas&Itemid=3
- ✓ **Wikipedia, la enciclopedia libre – Varios Artículos:** <https://es.wikipedia.org/>