

UNIDAD 7

ALMACENAMIENTO DE INFORMACIÓN EN XML

UNIDAD 9. Almacenamiento de información en XML

OBJETIVOS

- Conocer qué es SQL y cómo XQuery permite realizar las mismas funciones pero con BD XML nativas.
- Convertir una BD relacional a un conjunto de documentos XML equivalentes.

UNIDAD 9. Almacenamiento de información en XML

ÍNDICE

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.1 Bases de datos relacionales

9.1.2 Transformación a XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1 Herramienta BaseX

9.3 XQUERY

9.4 CONSULTAS

9.4.1 De FLWOR a HTML

9.5 ACTUALIZACIÓN

9.5.1 Inserción

9.5.2 Modificación

9.5.3 Borrado

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

- ✓ Gran parte de las BD que hay hoy en día están basadas en un **modelo entidad-relación**.
- ✓ Algunos SGBD tienen modelos híbridos que permiten añadir extensiones al modelo relacional y avanzar a un modelo orientado a objetos de manera poco traumática y transparente.
- ✓ Los SGBD actuales proporcionan en algunos casos extensiones que permiten trabajar con los modelos y representaciones definidas en documentos XML.

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

- ✓ Si el objetivo es utilizar XML desde el principio, se debería utilizar algunos de los siguientes gestores de bases de datos XML nativos:
 - eXcelon XIS Lite.
 - TEXTML.
 - dbXML.
 - eXist.
- ✓ Los dos primeros son comerciales y las dos últimas son OpenSource.

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

- ✓ Cuando se habla de BD XML nativas, se ha de dejar claro que existen dos maneras de almacenar información dentro de ellas:
 - **Usando un modelo centrado en el almacenamiento de los datos:** exactamente igual que las BD Relacionales (se guardan tuplas). Permite seguir utilizando los modelos relacionales dentro de BD XML.
 - **Usando un modelo centrado en el documento:** no hay campos, ni datos, tal y como se conocen en las BD Relacionales. Se guardan documentos XML. Permite almacenar documentación de diferentes modelos dentro de la BD.

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

- ✓ Dependiendo de los objetivos de almacenamiento que se plantee, quizás se ajuste más un modelo que otro.
- ✓ Dado que el modelo relacional es el más utilizado hoy en día, se indicarán una serie de pasos, para que usando XML, se pueda utilizar el mismo modelo de datos.

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.1 Bases de Datos Relacionales

- ✓ Cuando se utiliza un modelo centrado en el almacenamiento de los datos, la referencia son los las BD relacionales.
- ✓ Todo se basa en un conjunto de tablas bidimensionales que permiten almacenar los datos del mundo real.
- ✓ Para facilitar la comprensión, imaginemos que se quiere representar un conjunto de libros de una biblioteca.

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.1 Bases de Datos Relacionales

- ✓ Para conseguir almacenar toda la información relativa a cada libro, necesitaremos una tabla que almacene atributos como “Título”, “Autor”, “Editorial”, “Edición”, “ISBN”, y “NumPaginas”.
- ✓ Esta tabla se llamaría “Libros”.
- ✓ Si se quiere añadir un libro, se generaría una nueva entrada en la tabla (fila o tupla), en la que se contemplarán los atributos de ese libro.

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.1 Bases de Datos Relacionales

Tabla Libros

Cod_Libro	Título	Autor	Editorial	Edición	ISBN	Num. Páginas
1	Don Quijote	Miguel de Cervantes Saavedra	Juan de la Cuesta	3	9788466745840	176
2	La Celestina	Fernando de Rojas	Maxtor	1	9788471664938	320
3	Leyendas	Gustavo Adolfo Bécquer	Cátedra	21	9788437620244	416

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.1 Bases de Datos Relacionales

- ✓ Antes de continuar añadiendo libros, podríamos darnos cuenta que existen una serie de atributos compartidos entre muchos libros.
- ✓ Sería el caso de “Autor” (un autor puede escribir muchos libros) o “Editorial” (una editorial podría publicar muchos libros de muchos autores distintos), etc.
- ✓ Si por cada libro tuviéramos que rellenar esa misma información, la tabla “Libros” tendría muchísima información redundante.
- ✓ ¿Cómo se resuelve esto?

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.1 Bases de Datos Relacionales

- ✓ Sin duda, si hay mucha información redundante, toda esa información debe salir de la tabla “Libros” y ubicarse en una nueva tabla.
- ✓ Para el caso del atributo “Autor”, se podría crear una nueva tabla llamada “Autores” en la que almacenaríamos la información relativa a ellos más un código que le represente de manera única (“CodigoAutor, Nombre, Apellidos, FechaNacimiento, etc.)

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.1 Bases de Datos Relacionales

- ✓ Con ese código de autor, cada vez que se añada un nuevo libro de un autor ya añadido, únicamente haremos referencia en el campo “Autor” del nuevo libro al código que referencia a dicho autor (en la tabla “Autores”).
- ✓ Esta manera de indireccionar información a otras tablas se denomina **relación**.
- ✓ De la misma manera se haría con el campo “Editorial”.

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.1 Bases de Datos Relacionales

Tabla Libros

Cod_Libro	Título	Cod_Autor	Editorial	Edición	ISBN	Num. Pág
1	Don Quijote	1	Juan de la Cuesta	3	9788466745840	176
2	La Celestina	2	Maxtor	1	9788471664938	320
3	Leyendas	3	Cátedra	21	9788437620244	416

Tabla Autores

Cod_Autor	Nombre	Apellidos	Fecha Nacimiento
1	Miguel	de Cervantes Saavedra	29/9/1547
2	Fernando	de Rojas	01/01/1470
3	Gustavo	Adolfo Bécquer	17/2/1836

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.2 Transformación a XML

- ✓ En principio, la adaptación del modelo de datos de una BD relacional a XML es relativamente sencillo de realizar.
- ✓ Una vez obtenido el modelo relacional, como el explicado en el punto anterior, se podría realizar una transformación de tablas a un documento XML simplemente creando una DTD y creando un documento XML bien formado.
- ✓ A continuación se detallarán los pasos a realizar para una tabla de libro y de autores como la anterior:

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.2 Transformación a XML

1. Cada tabla del modelo relacional será un elemento dentro del DTD:

```
<!DOCTYPE libros[  
    <!ELEMENT libros (libro)*>  
]>
```

En este caso, se tiene una tabla llamada “libros” que almacena tuplas de tipo “libro” (cero o más libros).

2. Cada tupla de la tabla se llama “libro”. Es necesario indicar qué campos componen a la tupla:

```
<!ELEMENT libro (cod_libro, titulo, editorial, edicion, isbn,  
    numpaginas, autores)>
```


UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.2 Transformación a XML

3. Cada columna de la tabla deberá establecerse como un tipo de dato almacenable (char, integer, etc.):

```
<!ELEMENT cod_libro (#PCDATA)>  
<!ELEMENT titulo (#PCDATA)>  
<!ELEMENT editorial (#PCDATA)>  
<!ELEMENT edicion (#PCDATA)>  
<!ELEMENT isbn(#PCDATA)>  
<!ELEMENT numpaginas (#PCDATA)>
```

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.2 Transformación a XML

4. Si existe una columna compleja (como puede ser la de “Autores”), se debe crear un nuevo elemento similar al del paso 2. En este caso, un libro puede tener uno o más autores (por eso el “+”). También se definen los tipos de datos a almacenar:

```
<!ELEMENT autores (autor)+>
```

```
<!ELEMENT autor (cod_autor, nombre, apellidos, fechanacimiento)>
```

```
<!ELEMENT cod_autor (#PCDATA)>
```

```
<!ELEMENT nombre (#PCDATA)>
```

```
<!ELEMENT apellidos (#PCDATA)>
```

```
<!ELEMENT fechanacimiento (#PCDATA)>
```

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.2 Transformación a XML

5. La DTD que resulta de todo esto será similar a ésta:

```
<!DOCTYPE libros[  
  <!ELEMENT libros (libro)*>  
    <!ELEMENT libro (cod_libro, titulo, editorial, edicion, numpaginas,  
      autores)>  
    <!ELEMENT cod_libro (#PCDATA)>  
    <!ELEMENT titulo (#PCDATA)>  
    <!ELEMENT editorial (#PCDATA)>  
    <!ELEMENT edicion (#PCDATA)>  
    <!ELEMENT numpaginas (#PCDATA)>
```

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.2 Transformación a XML

```
<!ELEMENT autores (autor)+>
  <!ELEMENT autor (cod_autor, nombre, apellidos, fechanacimiento)>
  <!ELEMENT cod_autor (#PCDATA)>
  <!ELEMENT nombre (#PCDATA)>
  <!ELEMENT apellidos (#PCDATA)>
  <!ELEMENT fechanacimiento (#PCDATA)>
/>
```

Un documento XML ajustado a la anterior DTD puede ser el siguiente:

UNIDAD 9. Almacenamiento de información en XML

9.1 UTILIZACIÓN DE XML PARA EL ALMACENAMIENTO DE LA INFORMACIÓN

9.1.2 Transformación a XML

```
<?xml version="1.0"?>
<libros>
  <libro>
    <cod_libro>1</cod_libro>
    <titulo>Don Quijote de la Mancha </titulo>
    <editorial>Juan de la Cuesta</editorial>
    <edicion>3</edicion>
    <numpaginas>176</numpaginas>
  <autores>
    <autor>
      <cod_autor>1</cod_autor>
      <nombre>Miguel</nombre>
      <apellidos>de Cervantes Saavedra</apellidos>
      <fechanacimiento>29/09/1547</fechanacimiento>
    </autor>
  </autores>
</libro>
</libros>
```

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

- ✓ Tras instalar un SGBD, los usuarios se comunicarán con él mediante algún tipo de lenguaje que permita manipular los datos almacenados.
- ✓ El lenguaje de consulta estructurado más popular es SQL (*Structure Query Language*).
- ✓ Este lenguaje permite, de forma declarativa, acceder a las BD relacionales y operar con ellas.
- ✓ Operaciones típicas son:
 - Creación y borrado de tablas.
 - Inserción, modificación y borrado de tuplas.
 - Ejecución de búsquedas mediante consultas.

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

- ✓ SQL es el lenguaje que en la actualidad se considera estándar de facto, pues la inmensa mayoría de los SGBD lo implementan.
- ✓ Existen numerosas revisiones del lenguaje pero la mayoría de los SGBD parten del estándar establecido en 1992.



UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

- ✓ Si, en definitiva, se va a trabajar con documentos XML para almacenar en su interior información mediante un modelo relacional, se necesitará algún lenguaje que permita extraer y manipular información de igual manera que SQL con las BD relacionales.
- ✓ Este lenguaje se llama **XQuery**.
- ✓ Xquery es, por tanto, un lenguaje 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.

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

- ✓ Cuando se va a analizar un documento XML, se crea un árbol de nodos de él mismo.
- ✓ Ese árbol tiene un elemento raíz y una serie de hijos.
- ✓ Los hijos del nodo raíz pueden tener más hijos.
- ✓ Si repetimos ese proceso llegará un momento en el que el último nodo no tiene ningún hijo, lo que se denomina nodo hoja.
- ✓ Establecido ese árbol de nodos, se recorre esa representación del documento XML buscando la información que se quiere (ya sea algún nodo concreto, algún atributo de un nodo concreto, etc).

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

- ✓ ¿Qué tipos de nodos se puede encontrar en ese recorrido?
- ✓ Básicamente los siguientes:
 - **Nodo raíz o “/”**. Es el primer nodo del documento XML. En el ejemplo de la biblioteca de libros explicado anteriormente, sería el elemento “Libros”.
 - **Nodo elemento**. Cualquier elemento de un documento XML es un nodo elemento en el árbol. El nodo raíz es un caso especial de Nodo elemento (no tiene padre). Cada nodo elemento posee un padre y puede o no poseer hijos. En el ejemplo anterior, sería “libro”.

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

- **Nodo texto.** Cualquier elemento del documento que no esté marcado con una etiqueta del DTD del documento XML. Es el contenido almacenado.
- **Nodo atributo.** Un nodo elemento puede tener etiquetas que complementen la información de ese elemento. Eso sería un nodo atributo.

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

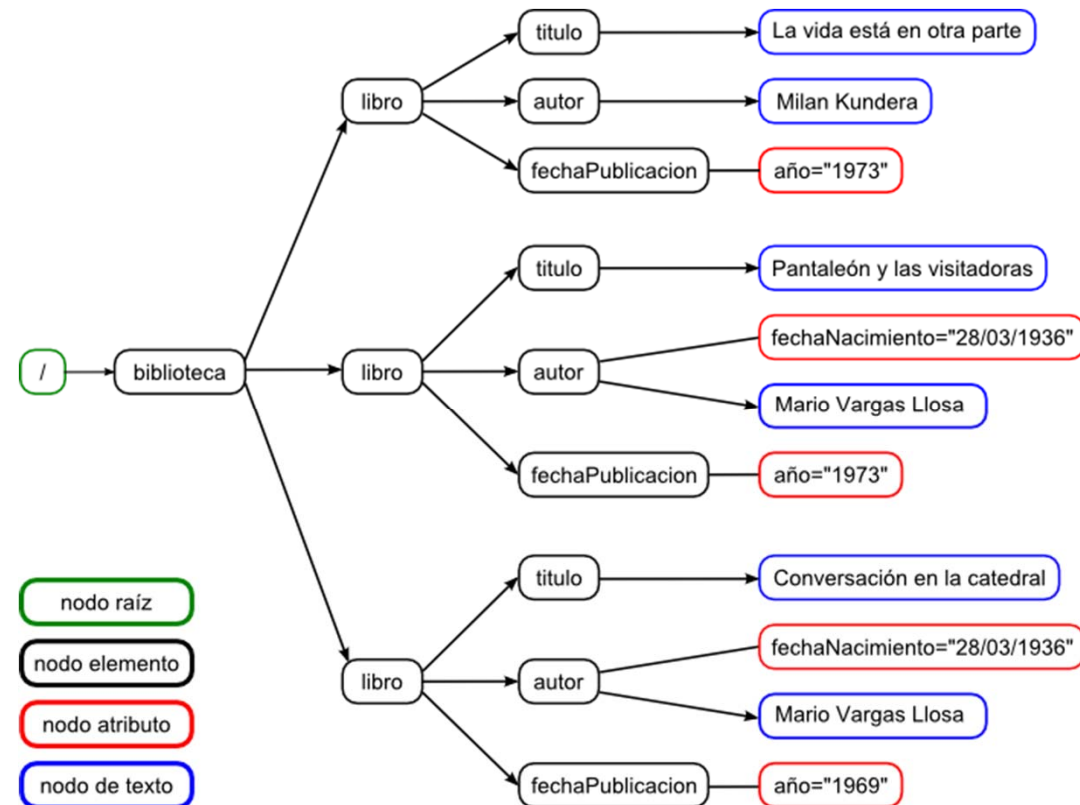
- ✓ Por ejemplo, el documento XML siguiente:

```
<?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>
```

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

✓ Se puede representar mediante el siguiente grafo:



UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

- ✓ La extracción de la información durante el recorrido del árbol será tan simple como la detección de los nodos a buscar y el procesamiento de la información que se quiere extraer de ese nodo concreto.
- ✓ Esto que parece tan complicado, se realiza fácilmente con una tecnología denominada **XPath** (*XML Path*).
- ✓ XPath es la herramienta que utiliza XQuery para procesar el árbol de nodos de un documento XML.
- ✓ Funciona en base a una serie de expresiones que permiten identificar qué parte del documento XML se quiere acceder o recorrer.

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

- ✓ La gran ventaja es que al ser una herramienta bastante genérica, XPath no solo es el motor de acceso en documentos XML para XQuery sino que es la base para otras tecnologías como Xpointer, Xlink o XSLT.
- ✓ Por tanto XQuery usa a XPath para hacer su trabajo.
- ✓ Pero lo realmente interesante es cómo hacer consultas con XQuery.

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

- ✓ **Basex** es un sistema de gestión de bases de datos XML y procesador XQuery, desarrollado como un proyecto comunitario en GitHub.
- ✓ Se especializa en almacenar, consultar y visualizar grandes documentos XML y colecciones.
- ✓ BaseX es independiente de la plataforma y se distribuye bajo una licencia de software libre permisiva.

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

- ✓ Proporcionan soporte para lenguajes de consulta estandarizados como XPath y XQuery.
- ✓ BaseX sigue las especificaciones de la World Wide Web Consortium (W3C), así como las actualizaciones oficiales y extensiones de texto completo.
- ✓ La interfaz gráfica de usuario permite a los usuarios buscar de forma interactiva, explorar y analizar sus datos y evaluar expresiones XPath/XQuery en tiempo real.

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

- ✓ Las tecnologías utilizadas son:
 - Lenguaje de consulta XPath.
 - XQuery 3.1
 - Formatos de datos soportados: XML , HTML , JSON, CSV, texto y datos binarios.
 - Interfaz gráfica de usuario que incluye varias visualizaciones como mapa, vista de tabla, vista de árbol y gráfico de dispersión.

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

- ✓ Está realizado en Java y es multiplataforma.
- ✓ Está disponible en varios idiomas como inglés, alemán, japonés, francés, italiano, entre otros. Ver la lista completa en el siguiente enlace: <http://docs.basex.org/wiki/Translations>
- ✓ Actualmente, está disponible la versión 9.3.2 de BaseX.

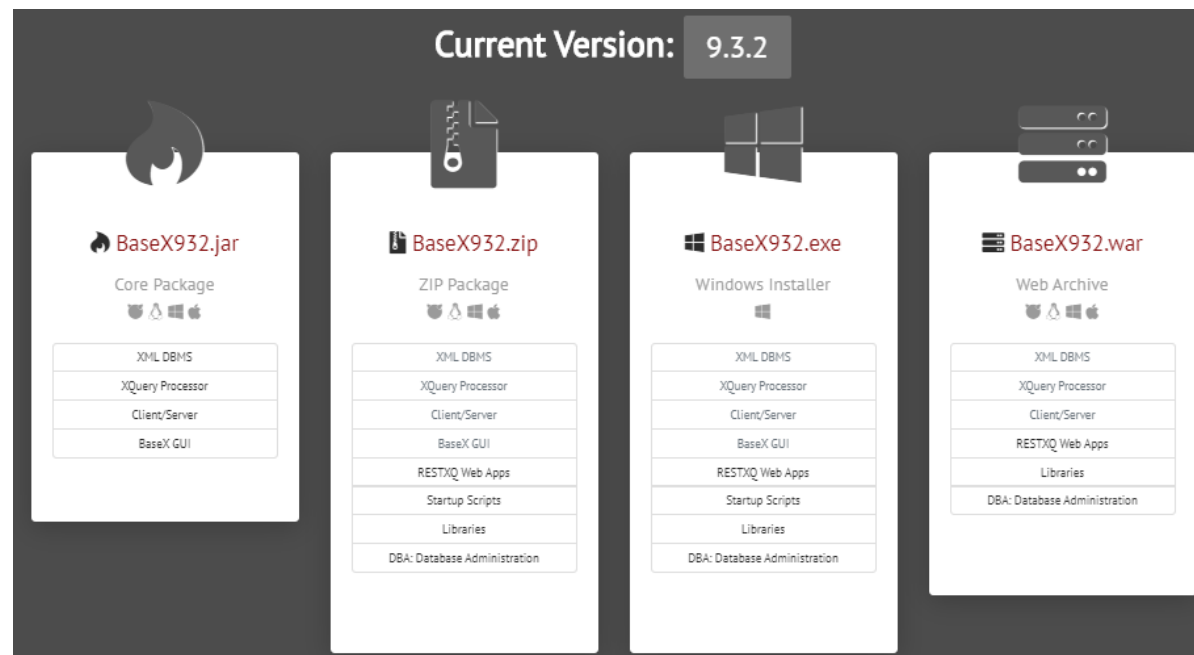
UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

- ✓ Descargar la última versión desde la página oficial e instalarla:

<http://basex.org/products/download/all-downloads/>

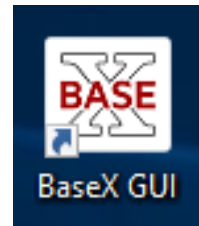


UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

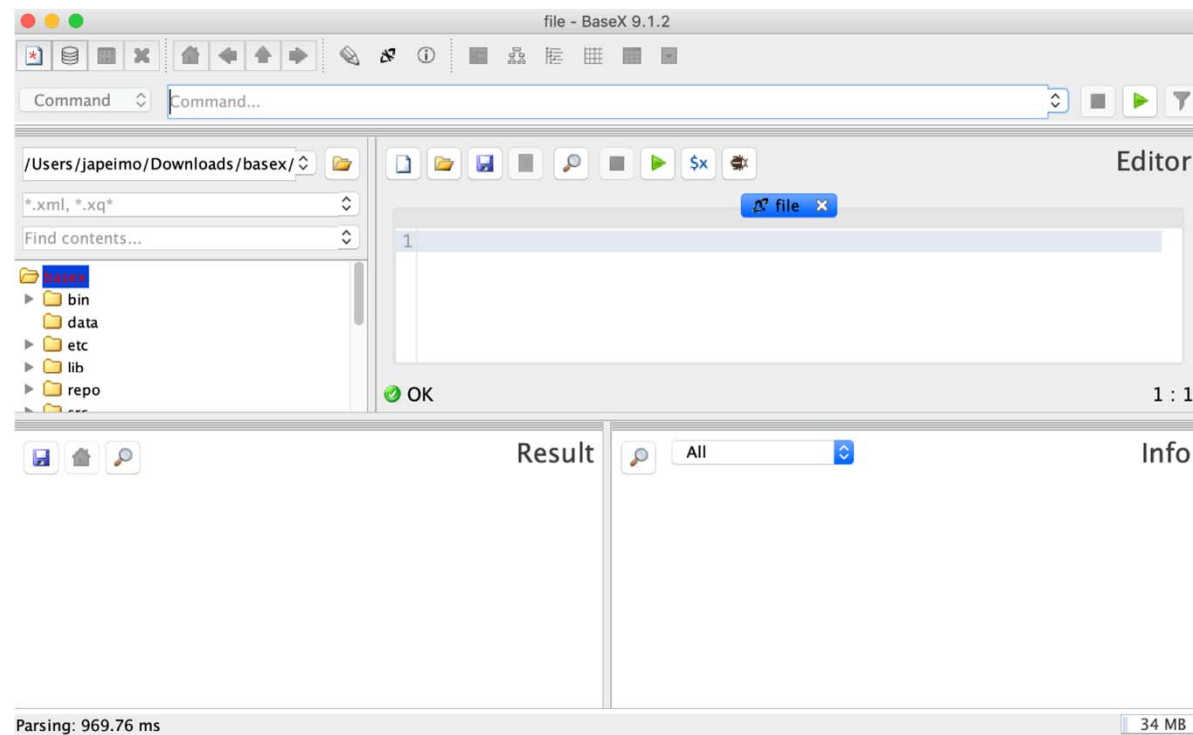
- ✓ Ejecutar la aplicación mediante su acceso directo:



UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX



UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

✓ En la herramienta podemos diferenciar las siguientes zonas:

- Barra de herramientas



- Consultas

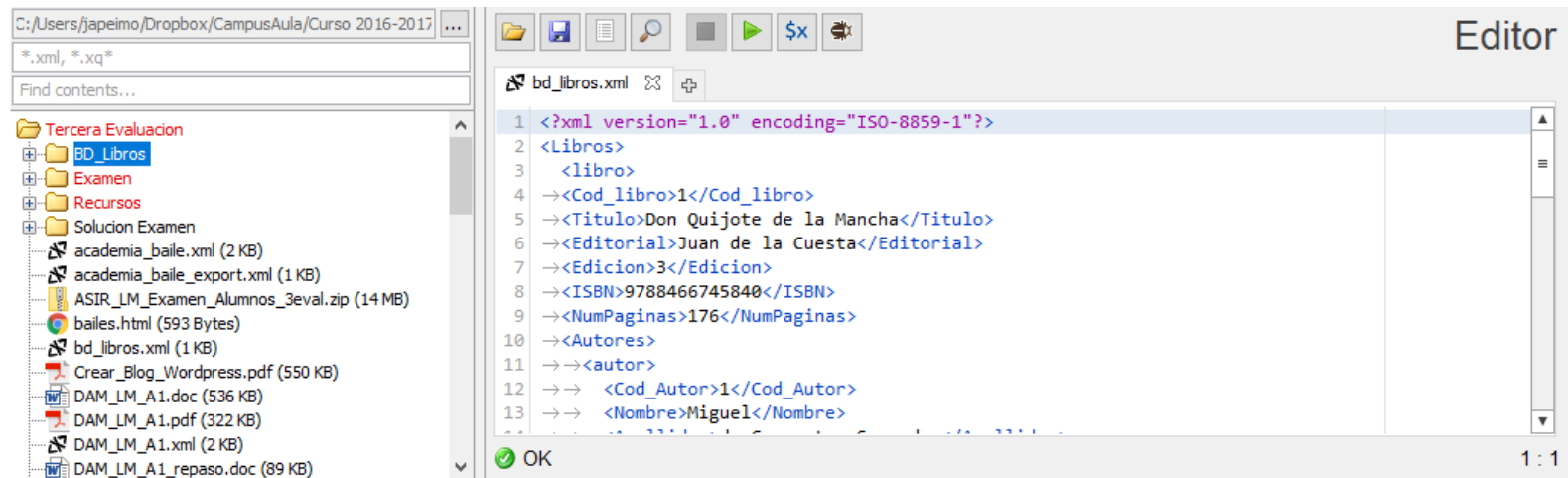


UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

- Editor



UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

- Resultado



The image shows the BaseX graphical user interface. At the top, there are three icons: a document, a home icon, and a magnifying glass. Below these icons, the XML query result is displayed in a text area. To the right of the text area, there is a vertical scrollbar and a label 'Result' at the top. The XML content is as follows:


```
<Libros>
  <libro>
    <Cod_libro>1</Cod_libro>
    <Titulo>Don Quijote de la Mancha</Titulo>
    <Editorial>Juan de la Cuesta</Editorial>
    <Edicion>3</Edicion>
    <ISBN>9788466745840</ISBN>
    <NumPaginas>176</NumPaginas>
    <Autores>
      <autor>
        <Cod_Autor>1</Cod_Autor>
        <Nombre>Miguel</Nombre>
        <Apellidos>de Cervantes Saavedra</Apellidos>
        <FechaNacimiento>29/09/1547</FechaNacimiento>
      </autor>
    </Autores>
  </libro>
</Libros>
```


UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

- Información

 Total Time: 32.99 ms

 Query Info

Compiling:

- pre-evaluating doc("bdlibros/bd_libros.xml")

Optimized Query:

document-node {"bd_libros.xml"}

Query:

doc("bdlibros/bd_libros.xml")

Result:

- Hit(s): 1 Item
- Updated: 0 Items
- Printed: 1349 Bytes
- Read Locking: bdlibros
- Write Locking: (none)

Timing:

- Parsing: 0.17 ms
- Compiling: 26.96 ms

UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

Comenzando a trabajar con BaseX

1. Crear una base de datos: *Database -> New*

- Input file or directory. Crear un nuevo directorio por cada base de datos nueva que creemos. El directorio lo crearemos en el directorio base “C:/Program Files (x86)/BaseX”
- Name of database. Dar un nombre a la nueva base de datos.

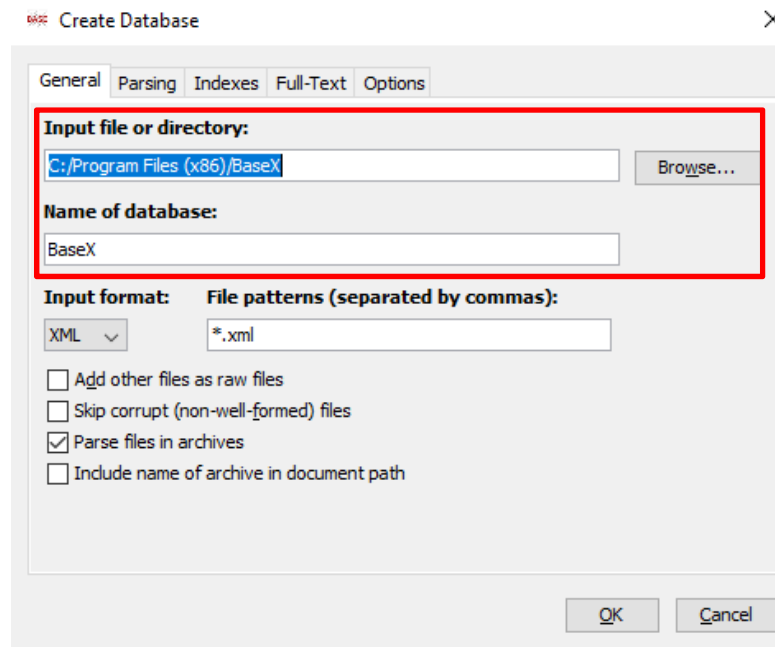
UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

Comenzando a trabajar con BaseX

1. Crear una base de datos:



UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

Comenzando a trabajar con BaseX

2. Copiar los archivos XML a la carpeta recién creada.
3. Abrir el archivo desde el *Editor*



UNIDAD 9. Almacenamiento de información en XML

9.2 LENGUAJES DE CONSULTA Y MANIPULACIÓN

9.2.1. Herramienta BaseX

Comenzando a trabajar con BaseX

4. Realizar la primera consulta escribiendo en la zona de consulta:
`doc("nombre_archivo.xml")` y ejecutar.



UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

Expresiones XPath

- ✓ Si se han realizado todos los pasos anteriores, ya tenemos disponible la base de datos para hacer consultas.
- ✓ En primer lugar, se va a utilizar una función de XQuery llamada **"doc(<nombreDocumento.xml>)"**, que permite extraer datos del documento indicado.
- ✓ Un ejemplo muy simple es que la consulta nos devuelva todo el documento XML almacenado.
- ✓ Esto se haría de la siguiente forma:
doc("bd_libros.xml")

UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

Expresiones XPath

- ✓ Habiendo escrito esta función en la pestaña **XQuery**, pulsamos el botón **Run Query** y veremos en la parte inferior (**Query Info**), si se ha ejecutado correctamente y el tiempo empleado para ello.
- ✓ Pero lo realmente interesante es el resultado de la consulta, que ha sido devuelta en la parte izquierda de la aplicación.

UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

Expresiones XPath

The screenshot displays the XQuery Editor and Result window. The Editor shows an XQuery expression that selects the root element 'biblioteca' and its children 'libro' and 'autor'. The Result window shows the output of the query, which is a list of XML elements representing the books and authors.

```
<?xml version="1.0" encoding="UTF-8"?>
<DOCTYPE biblioteca>
<libro cod_libro="1" titulo="Don Quijote de la Mancha" editorial="Juan de la Cuesta" edicion="3" isbn="9788466745848" numpaginas="176" autores="Cervantes Saavedra"/>
<libro cod_libro="2" titulo="La Celestina" editorial="Mator" edicion="1" isbn="9788471664938" numpaginas="320" autores="Mator"/>
<autor cod_autor="1" nombre="Miguel" apellidos="de Cervantes Saavedra" fecha_nac="29/09/1547"/>
```

The Result window shows the output of the query, which is a list of XML elements representing the books and authors. The output is displayed in a tree view, showing the hierarchy of the XML document.

XPath básica: muestra todo el documento XML

UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

Expresiones XPath

- ✓ Si en vez de devolver todo el documento, se quiere acceder únicamente a un conjunto de nodos bien identificados, añadiríamos el camino en el árbol para acceder a ellos, por ejemplo, los libros:

doc("bd_libros.xml")/libros/libro

```
<libro>
  <cod_libro>1</cod_libro>
  <titulo>Don Quijote de la Mancha</titulo>
  <editorial>Juan de la Cuesta</editorial>
  <edicion>3</edicion>
  <isbn>9788466745840</isbn>
  <numpaginas>176</numpaginas>
  <autores>
    <autor>
      <cod_autor>1</cod_autor>
      <nombre>Miguel</nombre>
      <apellidos>de Cervantes Saavedra</apellidos>
      <fecha_nac>29/09/1547</fecha_nac>
    </autor>
  </autores>
</libro>
<libro>
  <cod_libro>2</cod_libro>
  <titulo>La Celestina</titulo>
  <editorial>Maxtor</editorial>
  <edicion>1</edicion>
  <isbn>9788471664938</isbn>
  <numpaginas>320</numpaginas>
  <autores>
    <autor>
      <cod_autor>2</cod_autor>
      <nombre>Fernando</nombre>
      <apellidos>de Rojas</apellidos>
      <fecha_nac>01/01/1470</fecha_nac>
    </autor>
  </autores>
</libro>
<libro>
  <cod_libro>3</cod_libro>
  <titulo>Leyendas</titulo>
  <editorial>Cátedra</editorial>
  <edicion>21</edicion>
  <isbn>9788437620244</isbn>
  <numpaginas>416</numpaginas>
  <autores>
    <autor>
      <cod_autor>3</cod_autor>
      <nombre>Gustavo</nombre>
      <apellidos>Adolfo Bécquer</apellidos>
      <fecha_nac>17/02/1836</fecha_nac>
    </autor>
  </autores>
</libro>
```

UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

Expresiones XPath

- ✓ Como se puede observar, aparecen todos los libros que había almacenado en el documento XML inicial.
- ✓ Al igual que con las expresiones XSLT es posible que se quiera un solo conjunto de nodos dependiendo de un patrón de búsqueda.
- ✓ Si queremos todos los libros que tengan menos de 300 hojas, la consulta sería:

doc("bd_libros.xml")/libros/libro[numpaginas<300]

UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

Expresiones XPath

```
<libro>
  <cod_libro>1</cod_libro>
  <titulo>Don Quijote de la Mancha</titulo>
  <editorial>Juan de la Cuesta</editorial>
  <edicion>3</edicion>
  <isbn>9788466745840</isbn>
  <numpaginas>176</numpaginas>
  <autores>
    <autor>
      <cod_autor>1</cod_autor>
      <nombre>Miguel</nombre>
      <apellidos>de Cervantes Saavedra</apellidos>
      <fecha_nac>29/09/1547</fecha_nac>
    </autor>
  </autores>
</libro>
```

Resultado consulta XPath: libros que tengan menos de 300 hojas

UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

Expresiones XPath

- ✓ ¿Y si solo queremos los autores de los libros que tienen menos de 300 páginas?

La consulta sería:

doc("bd_libros.xml")/libros/libro[numpaginas<300]/autores

```
<autores>
  <autor>
    <cod_autor>1</cod_autor>
    <nombre>Miguel</nombre>
    <apellidos>de Cervantes Saavedra</apellidos>
    <fecha_nac>29/09/1547</fecha_nac>
  </autor>
</autores>
```

UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

Expresiones Xpath. FUNCIONES

- ✓ **text():** accede al contenido del elemento.

doc("bd_libros.xml")/libros/libro/titulo/text()

- ✓ **position():** devuelve la posición del nodo en la secuencia de nodos.

doc("bd_libros.xml")/libros/libro[position()=2]/titulo/text()

- ✓ **starts-with():** filtra por el comienzo de una letra.

doc("bd_libros.xml")/libros/libro/titulo[starts-with(text(),'D')]

- ✓ **contains():** filtra por contenido

doc("bd_libros.xml")/libros/libro/titulo [contains(text(),'Quijote')]

UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

Expresiones FLWOR

- ✓ Los ejemplos anteriores son la manera más sencilla de realizar búsquedas y selecciones de nodos concretos en un documento XML.
- ✓ Pero existe otra manera, mucho más potente, para realizar este trabajo.
- ✓ Es lo que se denomina expresiones **FLWOR** (*For, Let, Where, Order by, Return*).

UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

Expresiones FLWOR

- ✓ Una expresión FLWOR equivalente a la última consulta realizada, podría ser:

```
for $libro in doc("bd_libros.xml")/libros/libro  
where $libro/numpaginas<300  
return $libro/autores
```


UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

Expresiones FLWOR

- ✓ A continuación, se explicará para qué sirven cada una de las cláusulas FLWOR:
 - **for:** esta sentencia permite seleccionar los nodos que se quieren consultar, guardándose en la variable (\$variable)
 - **let:** esta clausula es opcional. Establece una nueva variable sobre el mismo u otro documento XML. Permite simplificar las expresiones posteriores y tener un código mucho más legible.
 - **where:** clausula que permite establecer una condición sobre la variable indicada en “for” y “let”.

UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

Expresiones FLWOR

- **order by:** clausula que define el orden de presentación de los resultados (ascending/descending).
 - **return:** permite devolver un valor concreto de los resultados obtenidos de las anteriores clausulas (uno por nodo).
-
- ✓ La utilización de expresiones FLWOR resulta similar a las consultas realizadas en SQL en las bases de datos relacionales.
 - ✓ Para más información:
https://www.w3schools.com/xml/xquery_flwor.asp

UNIDAD 9. Almacenamiento de información en XML

9.3 XQUERY

https://www.w3schools.com/xml/xquery_flwor.asp

Ejemplo FLWOR

```
1 for $x in doc("ejemplo3schools/bookstore.xml")/bookstore/book
2 where $x/price>30
3 return $x/title
```



2 Results, 81 b

Result

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

UNIDAD 9. Almacenamiento de información en XML

9.4 CONSULTAS

9.4.1 De FLWOR a HTML

- ✓ Queremos crear una consulta XQUERY que tras ejecutarla nos devuelva los resultados en formato HTML.
- ✓ Podemos unir dentro de XQuery etiquetas HTML y expresiones o cláusulas FLWOR.
- ✓ La única limitación es que cuando se fusionan en una consulta, tenemos que indicar al motor de consultas qué parte es la que tiene que procesar como consulta.
- ✓ Para ello, indicaremos entre llaves (“{”, “}”), que parte es FLWOR.

UNIDAD 9. Almacenamiento de información en XML

9.4 CONSULTAS

9.4.1 De FLWOR a HTML

- ✓ La mejor manera de verlo es a través de un ejemplo (se marcará en rojo la parte de la consulta FLWOR).
- ✓ **Ejemplo**
Queremos crear una consulta XQUERY cuyo resultado sea una tabla HTML que nos muestre el nombre del baile, el profesor que lo imparte y el número de plazas ofertadas.

UNIDAD 9. Almacenamiento de información en XML

9.4 CONSULTAS

9.4.1 De FLWOR a HTML

Ejemplo

```
<html>
<body>
  <h1> Bailes ofertados </h1>
  <table border="1">
    <tr>
      <th>Nombre baile</th>
      <th>Nombre profesor</th>
      <th>Plazas ofertadas</th>
    </tr>
```

UNIDAD 9. Almacenamiento de información en XML

9.4 CONSULTAS

9.4.1 De FLWOR a HTML

Ejemplo (cont.)

```
{  
for $baile in doc("academia_baile.xml")/bailes/baile  
let $nombre:=$baile/nombre  
let $profesor:=$baile/profesor  
let $plazas:=$baile/plazas  
return  
  <tr>  
    <td>{data($nombre)}</td>  
    <td>{data($profesor)}</td>  
    <td>{data($plazas)}</td>  
  </tr>  
}  
</table> </body> </html>
```

UNIDAD 9. Almacenamiento de información en XML

9.4 CONSULTAS

9.4.1 De FLWOR a HTML

Ejemplo (cont.)

Bailes ofertados

Nombre baile	Nombre profesor	Plazas ofertadas
Tango	Roberto Garcia	20
Cha-cha-cha	Miriam Gutiérrez	18
Rock	Laura Mendiola	15
Merengue	Jesús Lozano	12
Salsa	Jesús Lozano	10
Pasodoble	Miriam Gutiérrez	8

UNIDAD 9. Almacenamiento de información en XML

9.5 ACTUALIZACIÓN

- ✓ Hasta el momento, solo se ha podido realizar consultas sobre los contenidos ya almacenados en la base de datos XML.
- ✓ Quizás se quiera insertar, reemplazar, renombrar, cambiar y/o borrar entradas dentro de la base de datos.

UNIDAD 9. Almacenamiento de información en XML

9.5 ACTUALIZACIÓN

9.5.1 Inserción

Ejemplo

- ✓ Se quiere insertar un nuevo baile en la base de datos. Los datos son:
 - Nombre: Foxtrot
 - Precio: 22\$
 - Pago: mensual
 - Plazas:12
 - Comienzo: 01/01/2012
 - Fin: 31/07/2012
 - Profesor: Freddy Astaire
 - Sala: 3

UNIDAD 9. Almacenamiento de información en XML

9.5 ACTUALIZACIÓN

9.5.1 Inserción

Ejemplo

```
insert node
<baile id="7">
  <nombre>Foxtrot</nombre>
  <precio cuota="mensual" moneda="$">22</precio>
  <plazas>12</plazas>
  <comienzo> 01/01/2012</comienzo>
  <fin>31/07/2012</fin>
  <profesor>Freddy Astaire</profesor>
  <sala>3</sala>
</baile>
before doc("academia_baile.xml")/bailes/baile[1]
```

UNIDAD 9. Almacenamiento de información en XML

9.5 ACTUALIZACIÓN

9.5.1 Inserción

- ✓ Este código inserta el nodo indicado en la base de datos “academia_baile.xml”.
- ✓ El nodo se insertará antes del primer nodo de la base de datos (por la clausula “*before*”).
- ✓ Si se quiere insertar después, solo cambiaría “*before*” por “*after*”.
- ✓ Ambas utilizan como referencia al primer nodo (“[1]”).
- ✓ Una sentencia equivalente para insertar al principio de la base de datos, sin tener que referenciar a nodo alguno de la base de datos.

UNIDAD 9. Almacenamiento de información en XML

9.5 ACTUALIZACIÓN

9.5.1 Inserción

- ✓ Sería sustituyendo la última línea por esta otra:
as first into doc("academia_baile.xml")/bailes
- ✓ Si por el contrario se desea insertar al final:
as last into doc("academia_baile.xml")/bailes

9.5 ACTUALIZACIÓN

9.5.2 Modificación

Ejemplo

- ✓ En la inserción anterior se cometieron dos errores:
 1. El nombre correcto era “Ángel Correllada”
 2. El nº de plazas era 14
- ✓ Si en la inserción anterior se realizó antes del primer nodo de la BD, entonces el elemento insertado ocupará la primera plaza.
- ✓ Se cambiarán esos datos de dos maneras distintas:
 - Mediante la modificación del valor del nodo.
 - Mediante el reemplazo del nodo completo.
- ✓ La modificación se realizará de la siguiente manera:

UNIDAD 9. Almacenamiento de información en XML

9.5 ACTUALIZACIÓN

9.5.2 Modificación

Ejemplo (cont.)

```
replace value of node  
doc("academia_baile.xml")/bailes/baile[1]/profesor  
with "Angel Correllada"  
,  
replace node  
doc("academia_baile.xml")/bailes/baile[1]/plazas  
with <plazas>14</plazas>
```

9.5 ACTUALIZACIÓN

9.5.2 Modificación

Ejemplo (cont.)

- ✓ Como puede verse, las dos modificaciones se han realizado a la vez pero separándolas por una coma “,”.
- ✓ En el caso de no saber en qué posición de la BD se ha realizado la inserción (en principio debe dar igual la posición mientras estén insertados los datos), se podría tener un problema de actualizar erróneamente la tupla incorrecta.
- ✓ Para evitar este problema, lo más común es realizar la modificación mediante la utilización del identificador del baile (en este caso id=7)

UNIDAD 9. Almacenamiento de información en XML

9.5 ACTUALIZACIÓN

9.5.2 Modificación

Ejemplo (cont.)

```
replace value of node  
doc("academia_baile.xml")/bailes/baile[@id=7]/profesor  
with "Angel Correllada"  
,  
replace node  
doc("academia_baile.xml")/bailes/baile[@id=7]/plazas  
with <plazas>14</plazas>
```

UNIDAD 9. Almacenamiento de información en XML

9.5 ACTUALIZACIÓN

9.5.3 Borrado

Ejemplo

- ✓ Después de la inserción y la modificación del nuevo baile en la BD, finalmente parece que no se va a desarrollar ese curso.
- ✓ Se pide que finalmente se borre esa tupla en la BD (id=7).

```
delete node doc("academia_baile.xml")/bailes/baile[@id=7]
```