Ampliación de Bases de Datos
Año 2014/2015
Grado en Ing. Informática
Fdi - UCM

Hoja de ejercicios Tema 4

Modelo semiestructurado
Bases de datos XML

▷ **1.** Consideramos el siguiente documento `catalogo.xml` que representa un depósito de preguntas tipo test.

```
<catalogo-preguntas>
  <pregunta id="p1">
    <enunciado>¿En qué año se libró la batalla de Waterloo?</enunciado>
    <opcion num="2">1769</opcion>
    <opcion num="1">1845</opcion>
    <opcion num="3" correcta="sí">1815</opcion>
    <opcion num="4">1830</opcion>
  </pregunta>
  ...
</catalogo-preguntas>
```

Cada pregunta tiene un enunciado y una lista de opciones, de las cuales sólo una es correcta. El atributo `correcta` puede tomar los valores `sí` o `no`. El atributo `num` de cada opción especifica el orden en el que deben aparecer las opciones cuando se muestren en un cuestionario. Este orden no tiene por qué coincidir con el orden en el que aparecen las opciones en `catalogo.xml`.

(a). Especifica la DTD asociada a este documento.

(b). Realiza consultas en *XQuery* para obtener los siguientes resultados:

- Enunciado de la pregunta cuyo identificador es `p3`.
- Número de preguntas del catálogo.
- Preguntas con más de tres opciones.
- Preguntas que, debido a algún error en la elaboración del documento, aparezcan con más de una opción correcta.
- Preguntas que tengan una opción con el texto *"Ninguna de las anteriores"*.
- Preguntas que tengan una opción con el texto *"Ninguna de las anteriores"*, y que dicha opción no sea la última de la pregunta, es decir, que exista otra opción dentro de la misma pregunta que tenga un valor de `num` mayor que el de la opción *"Ninguna de las anteriores"*.

▷ **2.** Partiendo del depósito de preguntas del ejercicio anterior, queremos seleccionar algunas de las preguntas para introducirlas en un examen. Los exámenes se especifican mediante un documento `examen.xml` con el siguiente formato:

```
<examen>
  <ejercicio num="2" id="p6" puntuacion="0.5"/>
  <ejercicio num="1" id="p1" puntuacion="1.5"/>
  ...
</examen>
```

De nuevo, `num` determina el orden en el que aparecerán las preguntas en el examen, que no coincide necesariamente con el orden de las preguntas en el documento.

(a). Especifica la DTD asociada a este documento.

(b). Escribe una consulta en *XQuery* para obtener la puntuación máxima obtenible por el alumno que realice el examen.

(c). Realiza una consulta *XQuery* que a partir del documento `examen.xml` devuelva otro documento con el enunciado, opciones y puntuación de cada pregunta. El documento devuelto deberá tener el siguiente aspecto:

```
<examen>
  <ejercicio num="1" puntuacion="0.5">
    <enunciado>¿En qué año se libró la batalla de Waterloo?</enunciado>
    <opcion>1845</opcion>
    <opcion>1769</opcion>
    <opcion>1815</opcion>
    <opcion>1830</opcion>
  </ejercicio>
  <ejercicio num="2" puntuacion="1.5">
    ...
  </ejercicio>
  ...
</examen>
```

El orden de las opciones en el resultado viene determinado por el atributo `num` de las mismas dentro del documento `catalogo.xml`. Nótese que no debe incluirse este atributo, ni el atributo `correcta` en el resultado.

(d). Supongamos que un estudiante que realiza un determinado examen, y que sus respuestas están contenidas en un documento `respuestas.xml` con el siguiente formato:

```
<respuestas>
  <respuesta ejNum="1" opcionNum="3"/>
  <respuesta ejNum="2" opcionNum="1"/>
  ...
</respuestas>
```

Realiza una consulta *XQuery* que obtenga la calificación total del alumno a partir de la información contenida en los documentos `respuestas.xml`, `examen.xml` y `catalogo.xml`.

▷ **3.** Queremos representar la información de una serie de eventos en un calendario mediante un documento `calendario.xml`. Cada evento del calendario tiene un identificador único, un título, una descripción, una fecha de inicio, una fecha de finalización, y, posiblemente, una serie de uno o más participantes. El documento tiene el siguiente aspecto:

```
<calendario>
  <evento id="e01">
    <inicio>2013-04-15 16:00</inicio>
    <fin>2013-04-15 18:00</fin>
    <titulo>Clase de Laboratorio de ABD</titulo>
    <descripción>Entrega de Práctica 1</descripción>
    <participantes>
      <participante>Manuel Montenegro</participante>
      <participante>Yolanda García</participante>
      ...
    </participantes>
  </evento>
  ...
</calendario>
```

Tal como se muestra en este ejemplo, podemos suponer que el formato de las fechas es de la forma `YYYY-MM-DD hh:mm`. Se pide realizar consultas en *XQuery* que muestren los siguientes resultados:

- Número de eventos cuya fecha de inicio es 2013.
- Eventos transcurridos total o parcialmente en el 2013, en orden creciente de fecha de inicio.
- Eventos transcurridos a partir de las 16:00 con más de cinco participantes.
- Nombres de personas que participan en más de 3 eventos.
- Eventos que se solapan en el tiempo.
- Eventos que se solapan en el tiempo y tienen algún participante en común.

▷ **4. [Examen junio 2013]**
En este ejercicio realizaremos consultas sobre la base de datos XML de *Gafapastify*, un servicio de música que dispone de un catálogo de canciones que pueden ser escuchadas por sus usuarios vía *streaming*. El sistema almacena un historial de canciones escuchadas por los usuarios. De este modo un usuario puede echar un vistazo a las canciones que han sido reproducidas por él mismo, y también acceder al historial otros usuarios. No obstante, un usuario puede decidir (por motivos de privacidad) bloquear una determinada canción. Cuando una canción está bloqueada el usuario puede seguir escuchándola pero, en este caso, en el historial aparecerá la entrada correspondiente con un atributo especial `'bloqueado'` que impedirá que la entrada se muestre cuando otros usuarios consulten el historial.
En el documento XML mostrado a continuación (que llamaremos `bd.xml`) se muestra parte de la información almacenada en los servidores de la base de datos. Ésta contiene, en primer lugar, información sobre los distintos artistas. Para cada uno de ellos se almacena un identificador único, un nombre y

una descripción, siendo esta última opcional. Tras la lista de elementos `<artista>` tenemos una lista de canciones. Cada canción contiene un identificador, un título y una secuencia de uno o más artistas que participan en dicha canción. Cada uno de estos últimos se representa mediante un elemento `<participa>`, que contiene un atributo `id` con la referencia al artista correspondiente.

```
<bd-musica>
  <artista id="a01">
    <nombre>Muse</nombre>
    <descripcion>
      Banda inglesa de Rock formada en 1994 originaria de Teignmouth, Devon.
    </descripcion>
  </artista>
    <artista id="a02">
    <nombre>Lady gaga</nombre>
    <descripcion>
      Cantante, compositora, productora, bailarina, activista y diseñadora
      de moda estadounidense.
    </descripcion>
  </artista>
  ...
  <cancion id="c01">
    <titulo>Telephone</titulo>
    <participa id="a02"/>
    <participa id="a03"/>
  </cancion>
  <cancion id="c02">
    <titulo>Map of the problematique</titulo>
    <participa id="a01"/>
  </cancion>
  ...
  <reproduccion usuario="pepe43@gmail.com" cancion="c02" bloqueado="sí"/>
  <reproduccion usuario="otro@sip.ucm.es" cancion="c02"/>
  <reproduccion usuario="pepe43@gmail.com" cancion="c01"/>
  ...
</bd-musica>
```

(a). Escribe la DTD asociada a este documento

(b). Escribe una consulta en *XQuery* que devuelva los títulos de las canciones en las que participen más de dos artistas.

(c). Escribe una consulta en *XQuery* que devuelva los títulos de las canciones que hayan sido reproducidas más de un millón de veces, independientemente de si han sido bloqueadas o no por los usuarios que las escuchan.

(d). Decimos que una canción es un *placer culpable* si ha sido bloqueada del historial más de cien veces, bien sea por el mismo usuario, o bien por usuarios distintos. Escribe una consulta *XQuery* que

devuelva un listado con las canciones de la base de datos que sean placeres culpables. El listado estará ordenado de manera descendente según el número de bloqueos. Para cada entrada en esta lista se debe mostrar el título de la canción y los nombres de los artistas que participan en la misma. Cada resultado debe tener el siguiente formato:

```
<placer-culpable titulo="Telephone">
  <artista>Lady gaga</artista>
  <artista>Beyoncé</artista>
</placer-culpable>
```

Indicación: Recuerda que en las guardas de las expresiones *XPath* es posible introducir varias condiciones unidas mediante los operadores `and` y `or`. Por ejemplo, para obtener las canciones cuyo nombre es "`Only you`" y su identificador es "`c17`", podemos utilizar la expresión:

```
doc("bd.xml")/bd-musica/cancion[@id = "c17" and nombre = "Only you"]
```

▷ **5. [Examen septiembre 2013]**

Consideramos una base de datos XML destinada a la organización de un congreso. En particular, se almacena información sobre investigadores, artículos de investigación, y evaluaciones sobre dichos artículos. La DTD de este sistema viene dada a continuación:

```
<!ELEMENT congreso (investigador*, articulo*, evaluacion*)>
<!ELEMENT investigador (nombre, apellidos, correo-e)>
<!ELEMENT nombre (#PCDATA)>
<!ELEMENT apellidos (#PCDATA)>
<!ELEMENT correo-e (#PCDATA)>
<!ELEMENT articulo (autor+, titulo)>
<!ELEMENT autor EMPTY>
<!ELEMENT titulo (#PCDATA)>
<!ELEMENT evaluacion (nota, descripcion)>
<!ELEMENT nota (#PCDATA)>
<!ELEMENT descripcion (#PCDATA)>
<!ATTLIST investigador id ID #REQUIRED>
<!ATTLIST articulo id ID #REQUIRED>
<!ATTLIST autor id IDREF #REQUIRED>
<!ATTLIST evaluacion articulo IDREF #REQUIRED>
<!ATTLIST evaluacion investigador IDREF #REQUIRED>
```

El elemento raíz de la base de datos es `<congreso>`, que a su vez agrupa una lista de cero o más investigadores (`<investigador>`), una lista de cero o más artículos (`<articulo>`), y una lista de cero o más evaluaciones (`<evaluacion>`). Para cada investigador se almacena su nombre, apellidos, y dirección de correo electrónico. Para cada artículo se almacena una lista de autores (referencias a investigadores de la BD) y un título. Por último, cada evaluación contiene una nota (un número entero comprendido entre -3 y 3), y un informe detallado de la evaluación (`<descripcion>`). Además, una evaluación hace referencia al artículo evaluado, y al investigador que ha realizado la evaluación.

(a). Escribe una consulta en *XQuery* que devuelva los nombres y apellidos de los investigadores de la BD que no son autores de ninguna evaluación, cada uno de ellos con el siguiente formato:

```
<investigador-vago nombre="..." apellidos="..." />
```

(b). Decimos que una evaluación es *chanchullera* si el autor de dicha evaluación coincide con alguno de los autores del artículo que se evalúa. Escribe una consulta en *XQuery* que encuentre las evaluaciones chanchulleras existentes en la base de datos mostrando, para cada una de ellas, el título del artículo evaluado y los apellidos del autor de la evaluación con el siguiente formato:

```
<chanchullera titulo="..." evaluador="..." />
```

(c). Escribe una consulta en *XQuery* que muestre un listado de artículos ordenados por la nota media de las evaluaciones que versan sobre cada uno de ellos. Para cada uno de los artículos se mostrará su título, y la nota media obtenida en las evaluaciones sobre dicho artículo, con el siguiente formato:

```
<entrada titulo="..." nota-media="..." />
```

Indicación: La función avg devuelve la media de los elementos de una lista. Puedes suponer que paracada artículo existe, al menos, una evaluación que trata sobre él.

▷ **6.** Supongamos que almacenamos la información relativa a una red social en un documento XML, como el que se muestra a continuación:

```
<red-social>
  <usuario id="u01">
    <nick>Federico</nick>
    <edad>31</edad>
    <publicacion dia="20" mes="04" año="2003">
      <texto>En la playa con Federica.</texto>
      <voto id="u02"/>
      <voto id="u03"/>
    </publicacion>
    <publicacion dia="20" mes="04" año="2003">
      <texto>Hoy a dormir todo el día.</texto>
      <voto id="u03"/>
    </publicacion>
  </usuario>
  <usuario id="u02">
  ...
  </usuario>
  ...
</red-social>
```

Para cada usuario se almacena un identificador único, su *nick*, su edad y un conjunto de cero o más publicaciones. Cada una de estas últimas contiene una fecha (expresada mediante tres atributos) y un texto con el contenido de la publicación. Además, una publicación puede recibir varios votos de los usuarios. Cada voto se incluye dentro de la propia publicación, y contiene el identificador del usuario

que ha emitido dicho voto. Ten en cuenta que una publicación puede recibir votos de distintos usuarios, y un usuario puede votar a varias publicaciones.

(a). Escribe la DTD asociada a este documento.

(b). Especifica consultas en *XQuery* que obtengan los siguientes resultados:

- *Nicks* de usuarios cuya edad sea mayor o igual que 20.
- *Nicks* de usuarios que hayan realizado más de dos publicaciones en 2012.
- *Nicks* de usuarios que hayan emitido un voto a alguna publicación propia.
- *Nicks* de usuarios que hayan votado a más de cinco publicaciones, aunque sean propias. Por simplicidad, puedes suponer que un usuario no puede votar más de una vez a una misma publicación.

▷ **7. [Examen junio 2014]**

Partimos de un documento `dict.xml` como el de la Figura 1. Este documento almacena palabras de un diccionario multilingüe junto con sus significados. El diccionario se compone de elementos <palabra>, seguidos de elementos <definicion>. Una definición no es más que una cadena de texto. Por otra parte, cada palabra contiene un nombre, y uno o más significados. Cada significado hace referencia a una de las definiciones mencionadas anteriormente. Además, cada palabra contiene un atributo `idioma`, que indica el idioma al que pertenece, y un atributo que indica la categoría gramatical de la palabra (verbo, sustantivo, etc).

(a). Especifica la DTD asociada a este documento.

(b). Escribe una consulta en *XQuery* que muestre el nombre y la categoría gramatical de las palabras *monosémicas* (es decir, que tienen un único significado) pertenecientes al idioma español (ES). El formato del resultado debe ser como el del siguiente ejemplo:

```
<monosemica nombre="abdomen" categoria="sustantivo"/>
```

(c). Escribe una consulta *XQuery* que devuelva, para cada palabra española (ES) del diccionario, su nombre y una lista con sus definiciones. Cada palabra debe devolverse con el siguiente formato:

```
<div>
  <b>renunciar</b>
  <ol>
    <li>Hacer dejación voluntaria, dimisión o apartamiento de algo
        que se tiene, o se puede tener.</li>
    <li>Desistir de algún empeño o proyecto.</li>
    <li>En algunos juegos, no entrar.</li>
  </ol>
</div>
```

(d). Supongamos que una variable externa `$n`, de tipo `xs:string`, contiene el nombre de una palabra del diccionario, no necesariamente en español. Decimos que una palabra es una traducción de `$n` si contiene algún significado que coincide con los significados asociados a la palabra con nombre

```
<diccionario>
    <palabra idioma="ES" categoria="verbo">
        <nombre>renunciar</nombre>
        <significado id="c01"/>
        <significado id="c02"/>
        <significado id="c03"/>
    </palabra>

    <palabra idioma="ES" categoria="verbo">
        <nombre>cesar</nombre>
        <significado id="c01"/>
    </palabra>

    <palabra idioma="ES" categoria="verbo">
        <nombre>pasar</nombre>
        <significado id="c03"/>
    </palabra>

    <palabra idioma="EN" categoria="verbo">
        <nombre>give up</nombre>
        <significado id="c02"/>
        <significado id="c03"/>
    </palabra>
    ...
    <definicion id="c01">Hacer dejación voluntaria, dimisión o apartamiento de algo
            que se tiene, o se puede tener.</definicion>
    <definicion id="c02">Desistir de algún empeño o proyecto.</definicion>
    <definicion id="c03">En algunos juegos, no entrar.</definicion>
    ...
</diccionario>
```

Figura 1: Fichero `dict.xml` con información sobre las palabras de un diccionario.

$n, y pertenece a un idioma distinto al de la palabra correspondiente a $n. Escribe una consulta en *XQuery* que devuelva las traducciones de $n indicando, para cada una de ellas, el idioma al que pertenece. Por ejemplo:

```
<traduccion nombre="give up" idioma="EN"/>
```

▷ **8. [Examen septiembre 2014]**

Consideramos un sistema de encuestas cuya información está contenida en un fichero `encuestas.xml` como el de la Figura 2. El sistema se compone de una secuencia de preguntas, una secuencia de usuarios del sistema y una secuencia de votos. Cualquiera de estas tres secuencias puede ser vacía. Cada pregunta se representa mediante un elemento <pregunta> que contiene un identificador, un enunciado y una lista de una o más posibles respuestas (opciones). Cada opción viene acompañada por un número de opción, representado en el atributo num. Ten en cuenta que el orden en el que aparecen las opciones de cada pregunta dentro del documento XML no tiene por qué coincidir con el orden de los

```
<encuestas>
    <pregunta id="p1">
        <enunciado>¿Estás a favor de la nueva reforma de la LPI?</enunciado>
        <opcion num="2">Sí</opcion>
        <opcion num="1">No</opcion>
    </pregunta>
    <pregunta id="p2">
        <enunciado>¿Cual es la peor película de la primera mitad de 2014?</enunciado>
        <opcion num="4">RoboCop</opcion>
        <opcion num="2">Godzilla</opcion>
        <opcion num="1">Noé</opcion>
        <opcion num="3">Trascendence</opcion>
    </pregunta>
    ...
    <usuario id="u1">
        <nombre>Adolfa Gumersindez</nombre>
        <edad>31</edad>
    </usuario>
    ...
    <voto usuario="u1" pregunta="p2" opcion="2"/>
    <voto usuario="u1" pregunta="p1" opcion="1"/>
    ...
</encuestas>
```

Figura 2: Fichero `encuestas.xml`

atributos `num`, como se observa en las dos preguntas mostradas en la Figura 2. Para cada usuario se almacena un identificador, su nombre y su edad, ambos obligatorios. Por último, los elementos <voto> contienen las respuestas que han dado los usuarios a las preguntas del sistema. Por ejemplo, el primer elemento <voto> de la Figura 2 muestra que el usuario con identificador u1 (*Adolfa Gumersindez*) ha seleccionado la opción 2 (*Godzilla*) en la pregunta con identificador p2 (*¿Cual es la peor película de la primera mitad de 2014?*). Suponemos que cada usuario sólo puede votar una vez en cada pregunta.

(a). Especifica la DTD asociada a este documento.

(b). Escribe una consulta en *XQuery* que muestre la media de las edades de los usuarios del sistema. Utiliza, para ello, la función `avg`.

(c). Escribe una consulta en *XQuery* que devuelva los enunciados de aquellas preguntas que no hayan recibido ningún voto del usuario con identificador u1.

(d). Supongamos que una variable externa `$idPregunta`, de tipo `xs:string`, contiene el identificador de una pregunta del documento. Por ejemplo:

```
declare variable $idPregunta as xs:string := "p2";
```

Escribe una consulta en *XQuery* que devuelva el enunciado de la pregunta con identificador `$idPregunta` y devuelva, para cada opción, el porcentaje de votos obtenidos con respecto al número total de votos recibidos por la pregunta. Cada uno de estos porcentajes debe aparecer dentro de un elemento <respuesta> con el siguiente formato:

```
<respuesta num=" número de opción" porcentaje=" porcentaje %"/>
```

Por ejemplo, si el documento `encuestas.xml` contiene los siguientes votos sobre `p2`,

```
<voto usuario="u1" pregunta="p2" opcion="2"/>
<voto usuario="u2" pregunta="p2" opcion="1"/>
<voto usuario="u5" pregunta="p2" opcion="1"/>
<voto usuario="u9" pregunta="p2" opcion="4"/>
```

la consulta ha de devolver la siguiente información:

```
<resultado>
  <enunciado>¿Cual es la peor película de la primera mitad de 2014?</enunciado>
  <respuesta num="1" porcentaje="50%"/>
  <respuesta num="2" porcentaje="25%"/>
  <respuesta num="3" porcentaje="0%"/>
  <respuesta num="4" porcentaje="25%"/>
</resultado>
```

Indicación: Utiliza la función `div` para realizar divisiones. Por ejemplo, `2 div 4 ⇝ 0.5`.

▷ **9.**

Disponemos de un documento `series.xml` con información de series, episodios, y usuarios que votan a series. En la Figura 3 puedes encontrar un fragmento del código XML de esta base de datos, que está compuesta de una lista (posiblemente vacía) de series y una lista (posiblemente vacía) de usuarios. Para cada serie se almacena un identificador, un título y una lista de una o más temporadas (cada una de ellas numerada). Una temporada contiene uno o más episodios. Para cada episodio (también numerado) se almacena su duración (en minutos), un título, una sinopsis. Por otro lado, cada usuario puede asignar votos a las series. Para ello tiene que indicar el identificador de la serie a votar y asignarle una puntuación entre 0 y 10. Un usuario puede votar varias veces una misma serie.

(a). Especifica la DTD asociada a este documento.

(b). Escribe una consulta *XQuery* que devuelva el título y la duración del episodio 1 de la temporada 2 de la serie cuyo identificador es s1. El resultado debe ser devuelto con el siguiente formato:

```
<episodio-buscado titulo="What Lies Ahead" duracion="68"/>
```

(c). Decimos que una serie es *longeva* si tiene diez temporadas o más. Escribe una consulta *XQuery* que obtenga las series longevas de la base de datos. Para cada una de ellas se debe devolver su título y el número de episodios que contiene, con el siguiente formato:

```
<longeva titulo="The Simpsons" numero_episodios="552"/>
```

(d). Decimos que un usuario es un *troll* si más de la mitad de sus votos emitidos tienen un cero como nota. Escribe una consulta que devuelva toda la información de cada uno de los usuarios troll, tal y como aparece en la BD original.

(e). Un *calvario* es una serie en la que la duración media de todos sus capítulos es superior a 70 minutos y la calificación media de todos los votos que ha recibido es inferior a 3. Escribe una consulta que devuelva la lista de todos los calvarios de la base de datos con el siguiente formato:

```
<calvario titulo="Ana y los 7" duracion_media="71" calificacion_media="1.9"/>
```

Indicación: La función avg devuelve la media de los valores contenidos en la secuencia pasada como parámetro.

```
<red-series>
    <serie id="s1">
        <titulo>The Walking Dead</titulo>
        <temporada num="1">
            <episodio num="1" duracion="67">
                <titulo>Days Gone Bye</titulo>
                <sinopsis>
                Rick busca a su familia tras despertar de un coma en
                un mundo gobernado por los muertos vivientes. Morgan y
                Duane Jones, dos de los pocos sobrevivientes que quedan
                en pie ...
                </sinopsis>
            </episodio>
            <episodio num="2" duracion="45">
                ...
            </episodio>
            ...
        </temporada>
        <temporada num="2">
            <episodio num="1" duracion="68">
                <titulo>What Lies Ahead</titulo>
                ...
            </episodio>
            ...
        </temporada>
    </serie>
    <serie id="s2">
        ...
    </serie>
    <usuario nick="imdb01">
        <voto serie="s1" nota="9"/>
        <voto serie="s6" nota="1"/>
        ...
    </usuario>
    <usuario nick="cqtd01">
        ...
    </usuario>
</red-series>
```

Figura 3: Base de datos XML sobre series y episodios