Tutorial: XML, DTD, and XPath

Your company, Apasaja Private Limited, is commissioned by an online music database company to design and prototype XML version of a database application for the company's service. You are provided with a sample XML document.
Download the following file from Luminus "`Luminus Files > Cases > Music Library`".

`library.xml`.

The following is an excerpt of the XML document "`library.xml`".

```
<?xml version="1.0" encoding="UTF-8"?>
<library>
    <album>
        <title>Bua Hati</title>
        <artists>
            <artist>
                <name>Anang Ashanty</name>
                <country>Indonesia</country>
            </artist>
            <artist>
                <name>Kris Dayanti</name>
                <country>Indonesia</country>
            </artist>
        </artists>
        <songs>
            <song>
                <title>Timang-Timang</title>
                <duration>5:13</duration>
            </song>
            <song>
                <title>Miliki Diriku</title>
                <duration>5:35</duration>
            </song>
            <song>
                <title>Bua Hati</title>
                <duration>5:07</duration>
            </song>
        </songs>
        <genres>
            <genre>Pop</genre>
            <genre>World</genre>
        </genres>
        <year>1998</year>
    </album>
    ...
</library>
```

Your answers to the questions should be general enough to work for other similar documents (namely other documents following the same design logic).

XML

1. Add two more albums from CBS records (check `www.discogs.com/label/19936-CBS-Records`, for instance). Update the XML document and make sure that it is well-formed using eXist-db.

Solution:

```
<album>
    <title></title>
    <artists>
        <artist></artist>
    </artists>
    <songs>
        <song>
            <title></title>
            <duration></duration>
        </song>
    </songs>
    <genres>
        <genre></genre>
    </genres>
    <year></year>
</album>
```

DTD

1. Add an internal DTD for the XML document above. Make the DTD as tight as possible.

Solution:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE library [
  <!ELEMENT library (album*)>
  <!ELEMENT album (title,artists,songs,genres,year)>
  <!ELEMENT artists (artist*)>
  <!ELEMENT genres (genre*)>
  <!ELEMENT songs (song+)>
  <!ELEMENT artist (name,country)>
  <!ELEMENT song (title,duration)>
  <!ELEMENT title (#PCDATA)>
  <!ELEMENT name (#PCDATA)>
  <!ELEMENT country (#PCDATA)>
  <!ELEMENT duration (#PCDATA)>
  <!ELEMENT genre (#PCDATA)>
  <!ELEMENT year (#PCDATA)>
]>
<library>
  ...
</library>
```

Note that "*" means zero or more, "+" means one or more and no sign after the text means exactly one. Of these, exactly one is of course the most tight but not necessarily possible for the document we are working with. #PCDATA is *parsed character data*, which we for this module essentially can think of as where we store raw values, i.e. values that do not consist of sub elements.

2. Validate the XML document with its DTD using eXist-db. Try to change the structure of the document so it is invalid and observe the error messages generated by eXist-db.

XPath

Translate the following queries into XPath.
Your answers to the questions should include an XPath query with complete path as well as one with a more concise path, where relevant. Your answers to the questions should include an XPath query with the full syntax and as well as one using the shorthand syntax.
Evaluate the queries using eXist-db. eXist-db only shows 10 elements by default. You should wrap your query in an outer element to easily see all elements, i.e. `<results>{ XPath query }</results>`. For brevity, the solutions are shown without this wrapping.

1. Find the title of the songs.

Solution: Try to find a answer with the explicit axes first (not the `descendant::shorthand`)

- `doc("library.xml")/child::library/child::album/child::songs/child::song/child::title`
- `doc("library.xml")/descendant::song/child::title`
- `doc("library.xml")/descendant::title` does not work (gives the wrong result).

Shorthand

- `doc("library.xml")/library/album/songs/song/title`
- `doc("library.xml")//song/title`
- `doc("library.xml")//title` does not work (gives the wrong result).

2. Find the names of the Indonesian artists interpreting songs published between 1990 and 2000.

Solution:

```
doc("library.xml")/descendant::album[child::year>=1990 and child::year <=2000]
          /child::artists/child::artist[child::country='Indonesia']/child::name

doc("library.xml")//album[year>=1990 and year <=2000]/artists/artist[country='Indonesia']/name
```

3. Find how many songs in the library were interpreted by Anang Ashanty.

Solution:

```
count(doc("library.xml")/child::library/child::album[child::artists/child::artist/child::name='Anang Ashanty']
                        /child::songs/child::song)
```

```
count(doc("library.xml")/library/album[artists/artist/name='Anang Ashanty']/songs/song)


Caution (on concise path):
count(doc("library.xml")//album[//name='Anang Ashanty']//song)
=> Does not work, the condition inside [] became uncorrelated to node album.


The most concise answer is:
count(doc("library.xml")//album[artists//name='Anang Ashanty']//song)
```

4. Find the title of the albums in the library with four songs or more.

Solution:

```
doc("library.xml")/child::library/child::album[count(child::songs/child::song)>=4]/child::title


doc("library.xml")/library/album[count(songs/song)>=4]/title
```

5. Propose an interesting query that uses different axes than "`child`" and "`descendant`". Write the query in English and in XPath.