MONASH University
Information Technology

FIT3176 Advanced Database Design

Week 8 Studio - XML Schemas

Before beginning this work it is very important that you have carefully read the lecture slides and resource material for topic 8 (please ensure you do this before attending your allocated studio).

Data files needed for this week's activities are available in the Moodle archive datafiles-topic08.zip

Task 1 - Lecture Example

Examine the files discussed in the lecture available in the topic 8 lecture archive.

Task 2 - Publications

The XML file pub.xml (folder pub) contains details of texts on XML - extract the file from the archive and add an extra entry for the text 'XML Perspectives Comprehensive' using the following data

New Perspectives on XML 2nd Edition Comprehensive
Patrick Carey
Published: 17-08-2006
No of pages: 736
Price: 102.00

(i) make a copy of the file called pub1.xml and create a schema pub1.xsd which can be used to validate the XML document using only standard XML data types

Note that when oXygen creates a new schema file, it creates it with

elementFormDefault="qualified"

Unless you are working with namespaces and wish to clearly indicate which elements are in which namespace, this clause can be changed to "unqualified" or removed (easier). If you need to place the instance document in a namespace and are using "unqualified", some qualification will be required, the level of qualification will depend on the schema style used.

(ii) make a copy of the file from (i) called pub2.xml and create a schema pub2.xsd which enforces the following

- Add an annotation to describe what the file is and list your name and today's date
- Limit the title string to 10 - 50 characters
- Use a regular expression to set a mask for the payment amount

(iii) make a copy of the file from (ii) called pub3.xml and add an attribute called cover to each title element eg. cover="hard" or cover="soft". Create a schema pub3.xsd which enforces the features from pub1.xsd and also now includes the cover attribute.

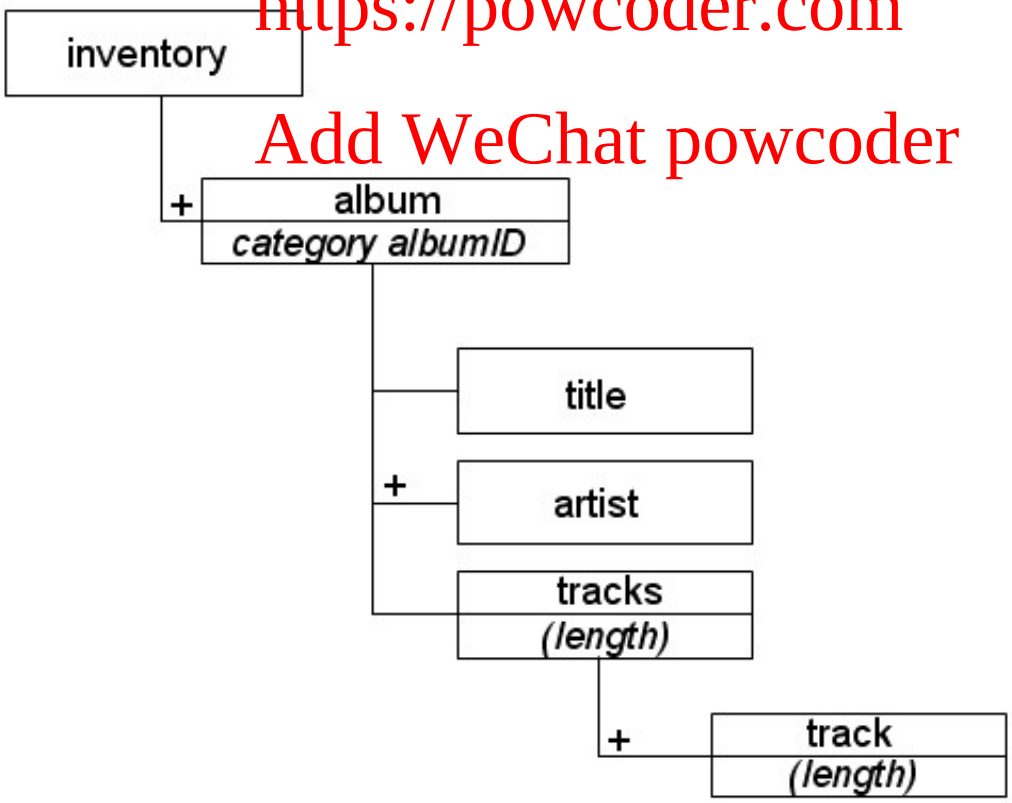
(iv) make a copy of the file from (iii) called pub4.xml and add a namespace to the XML root element (such as "http://books/pub/ns"). Create a schema pub4.xsd which can be used to validate this XML document.

Task 3 - Case Problems

Case Problem 1: The Jazz Warehouse

Data files needed for this Case Problem: **jw.xml**, **music.xsd** (folder case1)

Richard Brooks is working on an XML document to store the inventory of vintage albums sold by the Jazz Warehouse. Figure 4-53 below shows the structure of the vocabulary employed in the document. A description of the elements and attributes used in the music catalog is shown in Figure 4-54.



Element or Attribute	Description
inventory	The root element
album	Element storing information about each album
category	The album category (New Orleans, Swing, Bebop, Modern)
albumID	Album ID number in the form JW###### where # is a digit
title	The album title
artist	The album artist (there may be more than one)
tracks	Element storing information on the album tracks
length	An optional attribute storing the length (in hours:minutes:seconds) of an entire album or track
track	The name of an individual album track

Figure 4-54

Richard needs your help in creating a schema that will validate the data he's already entered and will enter in the future. In order to keep the code compact, you'll use a Russian doll design for the schema layout. Richard is not planning to use this schema with other XML documents, so you do not have to define a namespace for the vocabulary he created.

To complete this task:

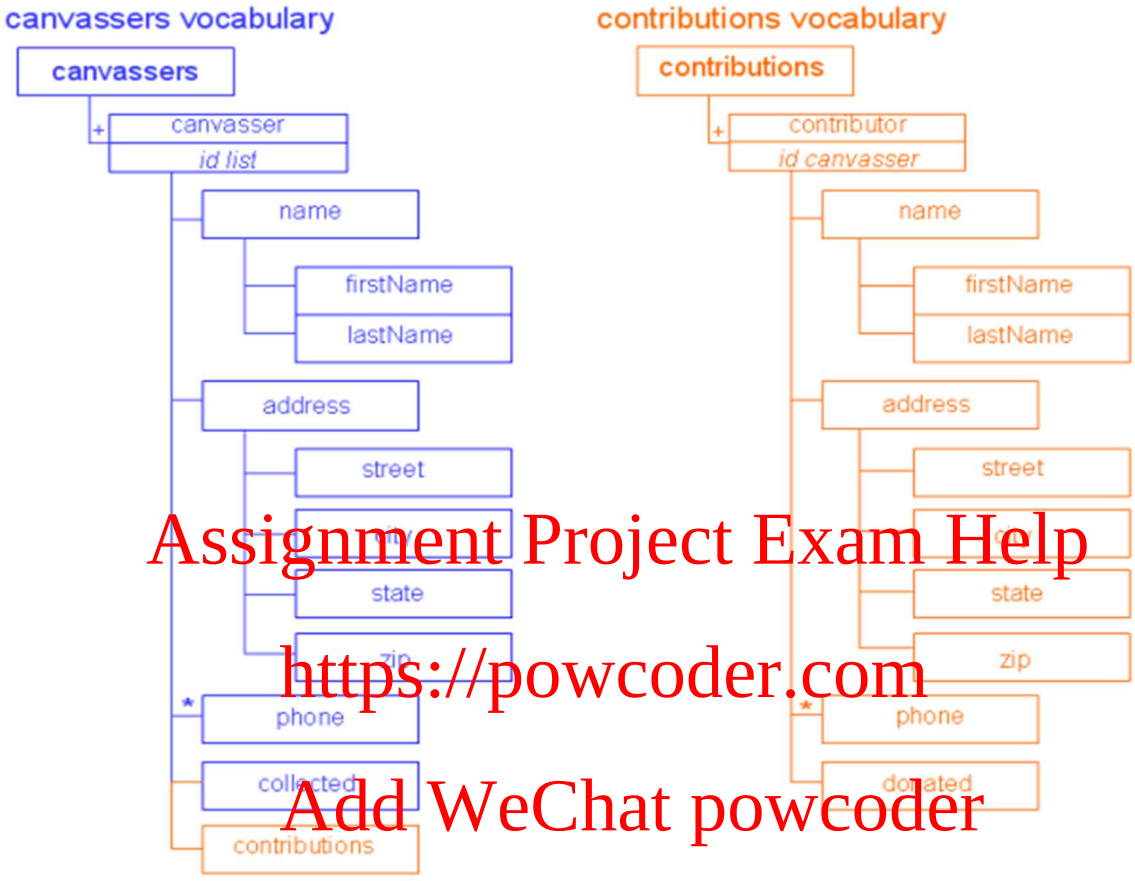
1. Enter your name and the date in the comment section of each file - jw.xml and music.xsd
2. Go to the music.xsd file in your text editor and insert the root schema element. Declare the XML Schema namespace with xsd as the namespace prefix.
3. Define the following data types:
   - albumIDType, based on the ID data type and following the pattern JW######, where # is a digit
   - jazzType, based on the string data type and limited to New Orleans, Swing, Bebop, and Modern
4. Declare the inventory complex element type and nest the album element within in. The album must occur at least once, but its upper level is unbounded.
5. Within the album element, create a Russian-doll layout, first nesting the child elements title, artist, and tracks, and the attributes category and albumID. The title and artist elements are both simple types containing string data. The artist element may occur multiple times, but must occur at least once. The category attribute is required and contains jazzType data. The albumID attribute is also required and contains albumIDType data.
6. The tracks element is a complex type element and contains at least one track element. The tracks element also contains an optional length attribute containing time data.
7. The track element is a complex type element and contains a simple text string and the length attribute. The length attribute is optional and stores time data.
8. Close the music.xsd file, saving your changes.
9. Go to the jw.xml file in your text editor. Within the root inventory element, declare the XML Schema instance namespace. Use xsi as the namespace prefix. Attach the schema file music.xsd to this instance document. Do not place the schema or the instance document in a namespace.
10. Save your changes to the jw.xml file and then validate jw.xml. Correct any data entry errors you find in the instance document.

Case Problem 2: EPAC-MO

Data files needed for this Case Problem: canvlist.xsd, clist.xsd, contrib.xml, lib.xsd, report.xml (folder case2)

EPAC-MO is an environmental political action committee operating in Central Missouri. Sudha Bhatia manages fundraising reports for the committee and has been using XML to record information on contributors and canvassers who collect donations. She's developed a vocabulary for the canvassers to record each canvasser's

name, address, and total amount collected. She's also developed a vocabulary for contributions, recording each contributor's name, address, and total amount donated. Sudha wants your help in developing a schema to validate the information she puts in her documents. Since Sudha will create a compound document displaying information on both canvassers and contributions, she needs the schema to combine information from several namespaces. Figure 4-55 below shows the tree structure for the vocabulary of the compound document.



You notice that several simple and complex types, such as the name and address elements, are repeated in both vocabularies. Rather than repeating the definitions in both schemas, you decide to place the definitions for these common elements in a third schema containing a library of common data types. You'll import that schema into the schema files for both the canvasser and contributions vocabularies.

To complete this task:

1. Enter your name and the date in the comment section of each file - canvlist.xsd, clist.xsd, lib.xsd, and report.xml.
2. Go to the lib.xsd file in your text editor. Within this schema you'll create a library of data types. Add the root schema element and insert the declaration for the XML Schema namespace using the xs prefix. Set the default namespace and target of the schema to the URI http://epacmo.org/library.
3. Create the following data types:
   - stateType. based on the string data type and limited to two uppercase letters
   - zipType, based on the integer data type and following the pattern d#### where d is a digit from 1 to 9 and # is any digit.
   - phoneType, based on the string data type and following the pattern of d##-### where d is a digit from 1 to 9 and # is any digit.

4. Create a complex type named nameType containing the following sequence of elements: firstName and lastName. Both elements should contain string data.
5. Create a complex type named addressType containing the following sequence of elements: street, city, state, and zip. The street and city elements contain string data. The state element contains stateType data. The zip element contains zipType data.
6. Close the lib.xsd file, saving your changes and go to the clist.xsd file in your text editor. Within this file you will create the schema for the contributions vocabulary.
7. Insert the root schema element, declaring the XML Schema namespace with the xs prefix. Declare the library namespace using the URI http://epacmo.org/library and the prefix lib. Set the default namespace and the schema target to the URI http://epacmo.org/contributors.
8. Import the lib.xsd schema file using the URI of the library namespace.
9. Using a Russian doll design, insert the declaration for the complex type element contributions. The element contains the child element contributor, which occurs at least once in the instance document.
10. The contributor element is also a complex type element containing the child elements name, address, phone, and donated, and the attributes id and canvasser. The phone element may occur any number of times.
11. Set the data types of the elements and attributes of the contributor element as follows:
    - The name element contains nameType data (taken from the library namespace).
    - The address element contains addressType data (taken from the library namespace).
    - The phone element contains phoneType data (taken from the library namespace).
    - The donated element contains positive integer data.
    - The id attribute contains ID data.
    - The canvasser attribute contains an IDREF reference.
12. Close the clist.xsd file, saving your changes. Go to the canvlist.xsd file in your text editor. This file contains the schema for the canvasser vocabulary.
13. Insert the root schema element, declaring the XML Schema namespace with the xs prefix. Declare the library namespace using the lib prefix and the contributors namespace using the clist prefix. Set the default namespace and the schema target to the URI http://epacmo.org/canvassers.
14. Import the library and contributors schemas using the appropriate namespace URIs and schema locations (you will need to import elements).
15. Using a Russian doll design, declare the complex type element canvassers. The element has a single child element, canvasser, that occurs at least once. The canvasser element contains the child sequence: name, address, phone, collected, and a reference to the contributions element from the contributions namespace. The canvasser element also contains the id and list attributes. The phone element can occur any number of times.
16. As with the contributions schema, the name, address and phone elements contain nameType, addressType, and phoneType data, respectively. The collected element contains positive integers. The id attribute contains ID data. The list attribute contains a list of ID references.
17. Close the file, saving your changes; and then go to the report.xml file in your text editor.
18. Within the root canvassers element, declare the XML Schema instance namespace using the prefix xsi. Declare the canvassers namespace using the prefix canvlist. Declare the contributions namespace using the prefix contlist. Attach the schemas from the canvlist.xsd and clist.xsd files using the appropriate namespaces.
19. Change the opening and closing tags of the canvassers element to a qualified name using the canvlist prefix. Change the opening and closing tags of the two contributions elements to qualified names using the contlist prefix.
20. Go to the contrib.xml file in your text editor. Copy and paste the contributor information for contributors c001, c002, and c004 within the first set of contributions tags in the report.xml file. Copy and paste the contributor information for contributors c003, c005, and c006 from the contrib.xml file to within the second set of contributions tags in the report.xml file.
21. Save your changes to the report.xml file and then validate the file.