Lab Assignment 3 on XML

CIS612
Dr. Sunnie S. Chung

Semi-structure Data Processing: Transforming XML data to CSV format

For Lab3, You can write in your choice of any languages in any platform.

Parse XML Document using any Parser that supports DOM (Document Object Model) to Extract Schema and Data from XML Documents and Transform them into a Flat Text File Format (CSV/TSV) to Create Tables in a Relational DB

A way to retrieve information from XML files is to use an XML parser. An XML parser is, quite simply, software that reads an XML file and makes available the data in it. You can write your own XML parser that supports the XML Document Object Model (DOM) to build a tree structure that represents XML schema and Data.

The tree that DOM builds from a given XML document consists of different Node types and edges (Parent – Child hierarchy) as covered in class. You may choose to use an existing parser (for example, DOM Parser, SAX Parser) that supports the XML Document Object Model (DOM). The DOM defines a standard set of commands that parsers should expose so you can access HTML and XML document content from your programs. An XML parser that supports the DOM will take the data in an XML document and expose it via a set of objects that you can program against. You can learn how to access and manipulate XML documents via the XML DOM implementation, as exposed by the Microsoft® XML Parser (Msxml.dll) to download below or any other available XML parser that supports DOM. You can use a SAX Parser with Java as you can find. Learn how to use DOM Parser below and download one of any available then add the library into your programming platform like Visual Studio. Read through for more details. You can use XPATH in your program. See instructions on how to set up XPath in the Lab Section on the class webpage.

http://www.w3.org/TR/2000/REC-xml-20001006
http://www.w3.org/DOM/
http://www.dmoz.org/Computers/Programming/Internet/W3C_DOM/
http://www.dmoz.org/Computers/Programming/Languages/Java/XML/Class_Libraries/Parsers/
http://www.saxproject.org/
http://www.microsoft.com/en-us/download/details.aspx?id=3988
http://msdn.microsoft.com/en-us/library/aa468547.aspx

For sample code examples to extract data from a XML document using DOM,

http://www.w3schools.com/dom/dom_examples.asp
http://www.mkyong.com/java/how-to-read-xml-file-in-java-dom-parser/

http://www.mightywebdeveloper.com/coding/mysql-to-xml-php/

http://www.codediesel.com/php/converting-mysql-queries-to-xml/

Write a program that parses a given XML document in a file (in next page) using any existing XML parser that supports DOM to do the followings:

a) Extract data from the document and
b) Transforms the data into structured text files in CSV/TBS and/or into tables in a SQL database.

For this task,
1) Check the links given above to learn about DOM and an existing XML parser (ex: DOM parse, MSXML parser, or SAX Parser)
1) Download any available XML parser that supports DOM library (ex: DOM/MSXML into VS or DOM/SAX Parser for Java) to set up
2) Write a program that loads the given XML documents in a file (in next page) as input and parses to extract terminal data to write into an output file.
3) Write Stored Procedures to read data from the files and then store them in the tables in database
4) For this task, you need to have a mapping strategy to carry the relationships in XML to write data into the output files then store them into tables

Use the following XML document as an input file. Assume that you have <bibs> as a root element and you can assume you have a root in the input file as you need.

Automatic database table creation in a SQL Server in your program using JDBC/ODBC would be the best and recommended. However, you don't need to make the database table creation part in your codes if you don't know how. This is because there is some set up procedure if you are writing in Java that uses JDBC and try to connect to MS SQL Server which uses ODBC. You can use MySQL instead of MS SQL Server to avoid the extra JDBC-ODBC bridge set up if you are using JDBC in JAVA. You can create a Stored Procedure separately to create database tables from your outputs.

This lab is to learn how to handle a multivalued columns, nested and irregular data in semi-structured data to transform them to a correct relational database scheme, which is a common task in real life applications.

There are many ways to do this task. You are to find a way to do this task, any way would be good. Two common ways could be:
I
1) Create a big dirty table in CSV in your program and
2) Create multiple tables in correct scheme reading from the big table in a Stored Procedure in a SQL Server.
II.
1) Design a scheme (multiple CSV file formats) and create multiple CSV files in your program and
2) Create database tables directly from each CSV in a Stored Procedure in a SQL Server.

Design your CSV file formats. There is no one strict CSV file format as a solution. Think about what would be a good format to transform those irregular multi valued data or nested data to a table (or connected multiple tables) so that you can retrieve them from a database easily and efficiently without losing information. As long as it is transformed correctly without losing data, any form would be good.

Submit the followings:

On Blackboard:

1. Lab Report (in doc file) that shows the followings:
   1) Your set up/platform procedure,
   2) The executions to generate the output files in a structured any text file format (in CSV, TSV),
   3) Your outputs in text.
   4) Screen captures of your SQL tables created from the converted files, and
   5) Your source codes/scripts.

2. Submit your report file in doc, all your codes/scripts, and your output files in one zip file.

In Class: A printout of your report.

```
<bibs>
    <bib>
    <book>  <publisher> Addison-Wesley </publisher>
            <author> Serge Abiteboul </author>
            <author> <first-name> Rick </first-name>
                        <last-name> Hull </last-name>
            </author>
            <author> Victor Vianu </author>
            <title> Foundations of Databases </title>
            <year> 1995 </year>
    </book>
    <book price="55">
            <publisher> Freeman </publisher>
            <author> Jeffrey D. Ullman </author>
            <title> Principles of Database and Knowledge Base Systems </title>
            <year> 1998 </year>
    </book>
    </bib>
    <bib>
    <book>  <publisher> Addison-Wesley </publisher>
            <author> Rick Hull </author>
            <author> <first-name> Jane </first-name>
                        <last-name> Widom </last-name>
                    <address><street> Pacific Coast Highway</street>
                                <zip> 90254</zip>
                    </address>
             </author>
             <author> Dan Suci </author>
              <title> Implementation of Databases </title>
                  <year> 1998 </year>
   </book>
   <book price="100">
        <publisher> Freeman </publisher>
         <author><name> Jeffrey D. Ullman </name>
                    <address><street> 414 2nd St</street>
                      <zip> 90254</zip>
                      </address>
         </author>
         <title> Principles of Database and Knowledge Base Systems </title>
         <year> 1998 </year>
   </book>
    <paper price="15">
            <publisher> ACM Press</publisher>
             <author><name> Jeffrey Ullman </name>
                        <address><street> 200 Sepuveda</street>
                                    <zip> 90245 </zip>
                        </address>
             </author>
             <title> Principles of Database and Knowledge Base Systems </title>
             <year> 1998 </year>
    </paper>
    <paper price="10">
            <publisher> IEEE Press </publisher>
             <author> Jeffrey D. Ullman </author>
             <title> Cloud Azure </title>
             <year> 2010 </year>
    </paper>
    </bib>
</bibs>
```