CSCI 3901 Assignment5

Due date: 11:59pm Friday, March 20, 2020 in git.cs.dal.ca at
https://git.cs.dal.ca/courses/2020-winter/csci-3901/assignment-5/xxxx.git
where xxxx is your CSID (this repository already exists, so clone it and then add your code to it).

Problem 1

Goal

Access SQL through Java. Gain some exposure to XML.

Question

You work for the Northwind food distribution company. Management periodically wants a summary of the company's operation over a period of time.

Your job is to extract the summary information from the database. You will store the summary information in a file that follows an XML format. Someone else will then use XML tools (notably XSLT) to convert your information into something that management will review.

Input

Your program will obtain the following information from the keyboard in the following order:

- The starting date for the period to summarize
- The ending date for the period to summarize
- The name of the file for the output

All dates will be in a YYYY-MM-DD format.

Output

Your program will send all of its output to the specified file.

The data that you extract will be in 3 categories:

1. Customer information
    a. Report the customer name, address, number of orders in the period, and total order value
2. Product information
    a. Report, for each product category, the category name and for each product in the category report the name, supplier, units sold, and value of product sold
3. Supplier information
    a. Report the supplier name, address, number of products that we sold, and the total value of business that we sold from this supplier's products.

In all of the reporting, do not report any customers, products, or suppliers who have not had any interaction over the reporting period.

Your output file will be in an XML format. XML uses a set of tags to surround data to let you know what the data is. Some tags can be nested in other tags. HTML in assignment 1 follows an XML format.

We will use a simple version of XML. The first line of your XML file should provide information on the version of XML to use. The following line will be sufficient:

<?xml version="1.0" encoding="UTF-8" ?>

Following this first line, we get a set of nested tags to store the data. The starting tag has the format <…> and the matching ending tag has the format <…/> (differing by the ending slash) where … is the tag name. The outermost tag is year_end_report

Here is a description of the correct nesting (in a data type definition (DTD) format):

<!ELEMENT year_end_summary (year, customer_list, product_list, supplier_list) >
<!ELEMENT year (start_date, end_date) >
<!ELEMENT customer_list (customer*) >
<!ELEMENET customer (customer_name, address, num_orders, order_value) >
<!ELEMENT address (street_address, city, region, postal_code, country) >
<!ELEMENT product_list (category*) >
<!ELEMENT category (category_name, product*) >
<!ELEMENT product (product_name, supplier_name, units_sold, sale_value) >
<!ELEMENT supplier_list (supplier) >
<!ELEMENT supplier (supplier_name, address, num_products, product_value) >

All items without an ELEMENT line are of type #PCDATA (if that matters to you).

As an example to read this information, the tags year_end_summary must contain nested tags for each of year, customer_list, product_list, and supplier_list. The tag customer_list will contain zero or more tags with name "customer", as identified by the * after the "customer" tag in the ELEMENT clause.

In an XML file, the spacing doesn't matter. I encourage you to use spacing and tabs to make the XML file readable by a person.

Information on XML can be found at w3schools.com.

Sample output:

<?xml version="1.0" encoding="UTF-8" ?>
<year_end_summary>
    <year>
        <start_date> 1996-01-30 </ start_date>

```
        <end_date> 1996-02-02 </ end_date>
    </year>
    <customer_list>
        <customer>
            <customer_name> foo </customer_name>
            <address>
                <street_address> 123 Hemming Way </street_address>
                <city> Brandon </city>
                <region> Manitoba </region>
                <postal_code> P3J 4V2 </postal_code>
                <country> Canada </country>
            </address>
            <num_orders> 30 </num_orders>
            <order_value> 11425 </order_value>
        </customer>
    </customer_list>
    <product_list>
        <category>
            <category_name> games </category_name>
            <product>
                <product_name> game1 </product_name >
                <supplier_name> a_supplier </supplier_name>
                <units_sold> 100 </units_sold>
                <sale_value> 500 </sale_value>
            </product>
        </category>
    </product_list>
    <supplier_list>
        <supplier>
            <supplier_name> a_supplier </supplier_name>
            <address>
                <street_address> 456 Falcoln Ridge </street_address>
                <city> Saskatoon </city>
                <region> Saskatchewan </region>
                <postal_code> Q3C 1T8 </postal_code>
                <country> Canada </country>
            </address>
            <num_products> 5 </num_products>
            <product_value> 1250 </product_value>
        </supplier>
    </supplier_list>
</year_end_summary >
```

Constraints

- You may use any data structure from the Java Collection Framework.
- Write your solution in Java. The solution code must be your own.
- Use the mysql JDBC connection for Java.
- If in doubt for testing, I will be running your program on bluenose.cs.dal.ca. Correct operation of your program shouldn't rely on any packages that aren't available on that system.

Notes

- Use SQL vs Java as you deem best.
- Be sure to document your approach and any resources that you use.
- Look at where the bulk of the marks are in the marking scheme to help focus your efforts.
- You can run your queries against the csci3901 database on db.cs.dal.ca I will also make the sql file for the database available to you so that you can create your own copy of the database.
- You are expected to submit
    - Your Java code
    - External documentation
    - An argument as to why your solution is ready to be deployed

Marking scheme

- Documentation (internal and external) for what you added – 3 marks
- Program design, organization, and style – 2 marks
- Proper XML file format, including human readability – 3 marks
- Correct extraction of customer information – 4 marks
- Correct extraction of product information – 4 marks
- Correct extraction of supplier information – 4 marks
- Design elements of your program that would make it easy to report additional information for customers, products, or suppliers with minimal code changes (and avoiding cut-and-paste of code segments) – 2 marks
- The marker's reaction to an argument _that you supply_ to convince him that your code is ready to be used – 3 marks
    - 0 -> the marker is unconvinced by your argument that the code is ready for deployment
    - 1 -> Your approach sounds fine, but the marker has no confidence that your solution will deliver on the required data
    - 2 -> The marker isn't fully convinced that everything will work out, but is willing to run your code in parallel with some manual duplication of work to ensure that everything is caught correctly.
    - 3 -> The marker is convinced that we should use your program as-is. No need to run parallel systems. Awesome job!