Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept.
Second Year, Fall 2019

CS221: Object Oriented Programming
Programming Assignment 3
Assigned: Sunday, Nov 3rd, 2019
Due: Friday, Nov 29th, 2019

Simple DBMS

Objectives

Upon completion of this assignment, you will be able to:

- Design an object-oriented XML-based DBMS
- Getting familiar with SQL.
- Getting familiar with XML parsers
- Draw a UML class diagram that represents your model
- Apply different design patterns to your model

Description

A Computer Database is a structured collection of records or data that is stored in a computer system. On the other hand, a Database Management System (DBMS) is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. DBMS are categorized according to their data structures or types. The DBMS accepts requests for data from the application program and instructs the operating system to transfer the appropriate data.

Extensible Markup Language (XML) (**encoding: ISO-8859-1**) is a set of rules for encoding documents in machine readable form. It is defined in the XML 1.0 Specification produced by the W3C, and several other related specifications, all gratis open standards.

Hint1: Use *System.getProperty("file.separator")* and *file.mkdirs()*. You should create a main folder for all your tests. Also, you should cache the data while you are working, it is not wise to write to file with each change.
Hint2: Using Regex for parsing queries will make it much easier to pass the tests and to make it clean code as well

Recommended Design Patterns

You are free to implement your own design, but you can check these: Visitor, Iterator, Filter, Interpreter, Adapter, Pool, Builder, Thread pool, Singleton, Façade, Command.

Tasks

1. Implement a simple DBMS that handles data stored in XML files.

2. You should support the following SQL Statements:
    - Create database
    - Create table
    - Insert into table
    - Delete from table
    - Drop database
    - Drop table
    - Select from table
    - Update table

3. The DBMS you will develop should control the management and retrieval of data from data files. The DBMS accepts requests for data from application programs (in the form of the SQL queries) and retrieves and transfers the appropriate data from files that are stored physically on disk.

4. SQL is case insensitive in the statement and in the data base name (folder and files).

5. You can find detailed statements descriptions at: http://www.w3schools.com/sql

6. You should support the "SELECT * FROM table;" statement.

7. For statements which contain conditions (i.e. the Where clause), support only the simple conditions: =, >, and <. You are NOT required to support multiple conditions: AND, OR, or NOT. (If you did, it will be bonus).

8. The table should support only two types: varchar and int. "varchar" used to store string, and "int" to store numeric values (no floating point will be supported). Do not support custom type lengths (e.g. varchar(255), or int(11)), just assume all types of the same default length. However, you should make your design extensible.

9. The DBMS interface does NOT handle SQL queries directly. SQL queries should be sent first to another class, say Parser class. The Parser parses them, and then calls the appropriate DBMS function. The Parser should use only the functions supported by the DBMS interface to access the database. The files should be updated, whenever any DBMS function (that requires updating) is called.

10. The Parser should validate the SQL statements and reject bad ones.

11. The DBMS should validate the sent parameters and should throw appropriate exceptions whenever needed.

12. Each table in the database should be stored in a separate XML file. Files should be placed in the database directory.

13. You should maintain “Schema files” that contain data about the tables and columns. Tables should be validated across their schema files (“DTD or XSD Files” can be used for this purpose).

14. This means each table will have at least two files: one for data (XML file), and one for its structure (DTD or XSD file)

15. You should use SAX or DOM, or StAX parsers to parse and validate the XML database files.

16. You should provide a command line interface that accepts SQL queries.

17. You are required to create a UML class diagram.

Project setup

- Download the project from this link.
- Add test_supporting_files.jar to your external jars
- Organize your code under package with name eg.edu.alexu.csd.oop.db.cs<your-two-digits-class-number>
- Just write class number of one member of the team

Deliverables

- You should work in groups of four.
- Develop this assignment in Java.
- You should provide implementation for the given interface.
- You should deliver a report that contains the required UML diagram, describes your design thoroughly, and contains snapshots of your UI and a user guide that explains how to use your application. Any design decisions that you have made should be listed clearly.
- A self-executable Jar: The program should be executable by simply double clicking the icon, given that you have a running JRE.
- One of the team members will create new repo, and the other 3 members will just clone this repo and work simultaneously on this project

- Your final work must be on branch called “*dbms*” (without double quotations)
- Give our account on bitbucket “**OOP TAs**” *read* access in your repo access settings
- A discussion date will be determined later.
- Be creative! The required features are only the beginning of what you can do, add more features or spice up the required ones, bonus marks will be given to those with eye-catching extra features and user-friendly interfaces.
- **Delivering a copy will be severely penalized for both parties, so delivering nothing is so much better than delivering a copy.**

Good luck