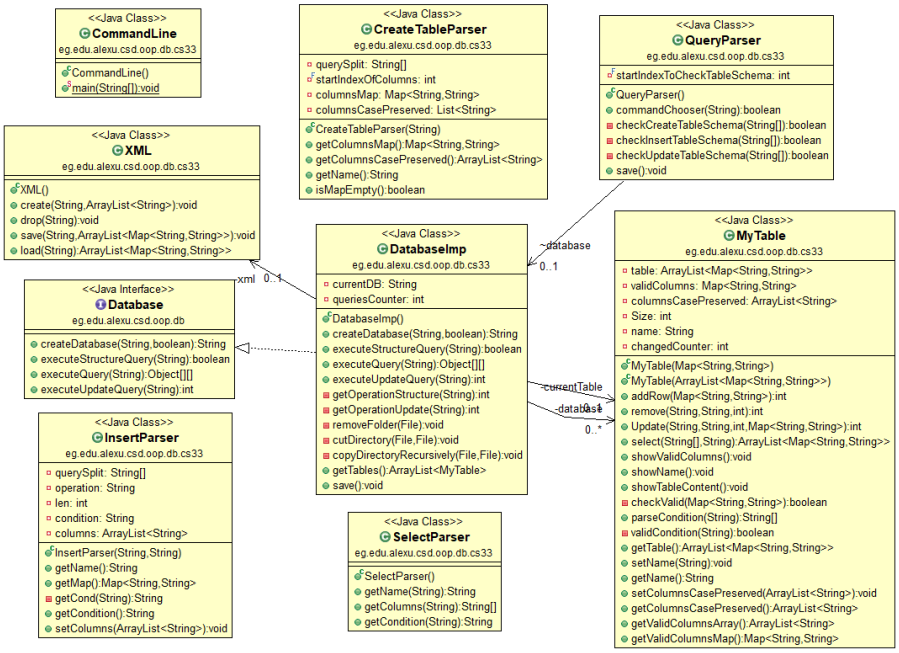
Student Names-IDs:	Elsayed Akram Elsayed-16 Ramez Maher Victor-28 Seif Eldin Ehab Mostafa-33 Youssef Sherif Nashaat-74
Lab Title:	Data Base Management System
Drs:	Dr/Khaled Nagi
TA:	Eng/Abdelrahman Hany

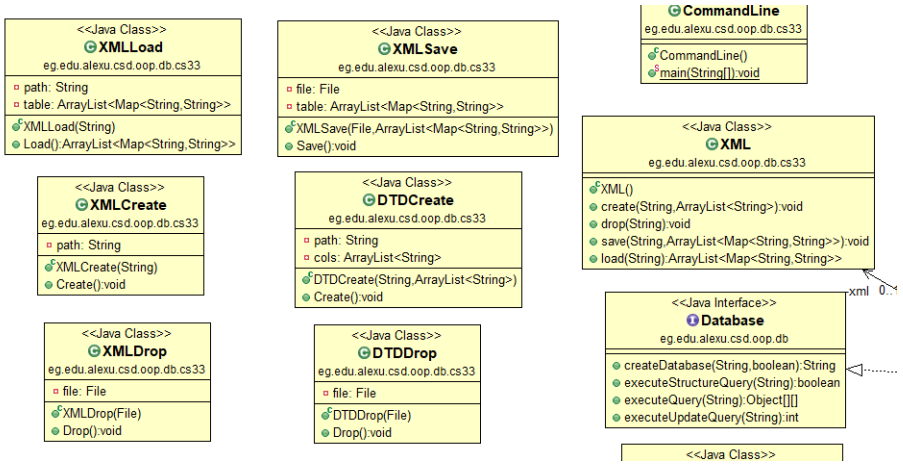
Computer and Systems Engineering Department

1 Introduction

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.

2 UML Diagram





3 Design

****Design decisions are bold****
Assumptions are italic

3.1 Taking Queries from User

- SQL queries are sent to class QueryParser which checks the validity of the statement.
- Valid statements are sent to the appropriate method in the class implementing the interface Database.

3.2 SQL Structure Execution

- Create database:
  1. Checks first if a database with similar name is found to work with it.
  2. Otherwise create a new folder for the database.
- Drop database:
  1. Checks for an existing database folder to delete it.
  2. Throws exception if there is no database created or found.
- Create table:
  1. Checks if a database is created and checks if a table with similar name is found to work with it.
  2. Otherwise create new table in the database folder with the appropriate columns' data types.
  3. **Columns' names and data types are converted to lower case so that a case insensitivity of the query is achieved.**
  4. **The conversion to lower case doesn't affect how it's displayed to the user.**

- Drop table:
    1. Checks for an existing table to delete it.
    2. Otherwise throws an exception if not found.

3.3 Select Query

- Check whether the query contains a condition or select all.
- Parse the condition and return it if found, otherwise return null.
- Check if the table selected from is found, otherwise return null.
- Go to the table class with the condition and retrieve the satisfied data.
- Return the data as a 2D array of objects.

3.4 SQL Update Execution

- Insert Query:
    1. Check if a database is created and table found, otherwise throw exception.
    2. **Data is represented in the table as an array list of maps where the rows are the maps and the columns are the keys.**
    3. **The keys of the map are the column's names and their values are the data entered.**
    4. **The Array List organizes the rows in a sequential order while the map lets the user reach is desired column in a time complexity of $\mathcal{O}(1)$, so searching any element in the table will take linear time complexity $\mathcal{O}(n)$.**
    5. An additional row is created in another class InsertParser as a hash map.
    6. Check for the keys while creating the row and if they are equivalent to the number of values, otherwise return null and print an error to the user.
    7. The new row is added to the table.
- Update Query:
    1. The query is parsed in class InsertParser to retrieve and validate the data.
    2. Check for a condition and parse it if found.
    3. Check if the table is found, otherwise throw an exception.
    4. Update these data in the table.
- Delete Query:
    1. The query is parsed in class InsertParser and checks for a condition.
    2. Check if the table is found, otherwise throw an exception.

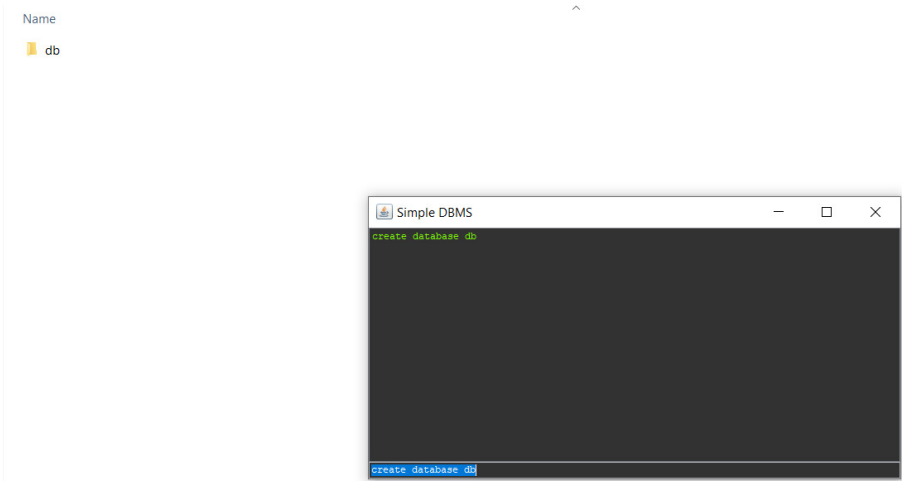
3. Delete the selected data from the table.

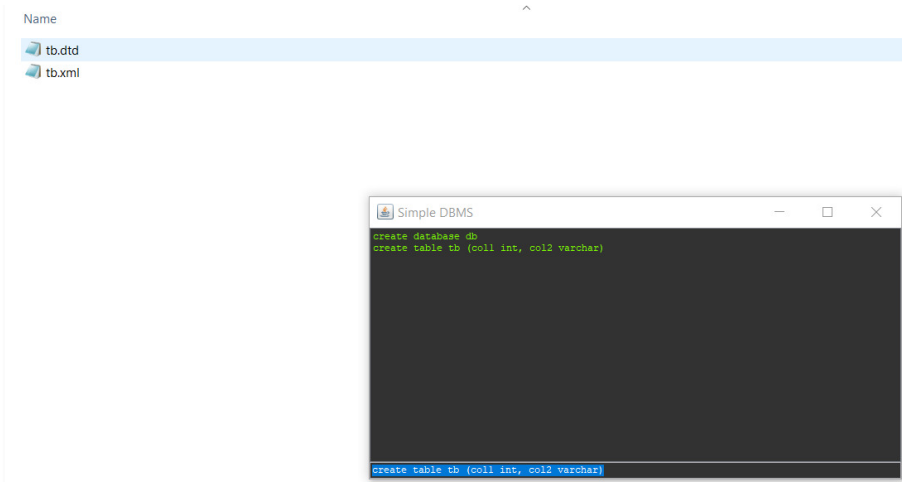
- The updated rows are tracked in the above queries and their number is returned from the function.

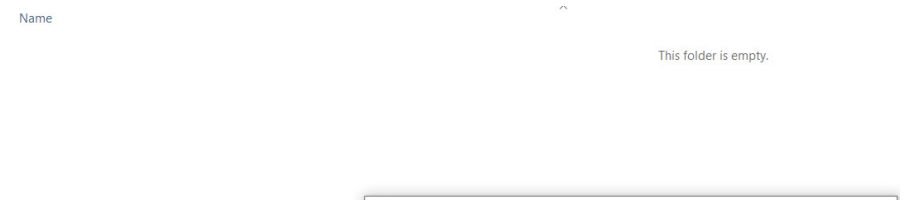
3.5 XML and Schema Files

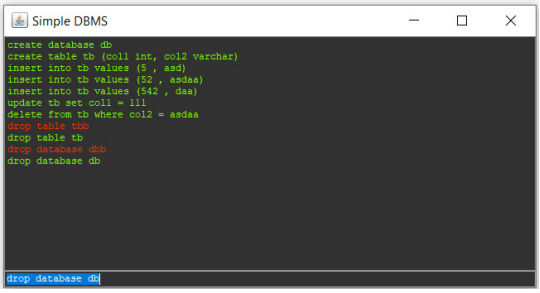
- Creating:
  1. The application creates suitable XML and DTD files once the table is created.
- Saving: **The application saves the current table in 3 cases**
  1. After 5 successfully completed edit queries (Insert, Update or Delete)
  2. After the user switches from one table to another and didn't reach the 5 successful queries counter.
  3. After the user exits the application and didn't reach the 5 successful queries counter.
- When the application saves the file it validates it first through the DTD file in order to continue, if it doesn't match then it keeps the original file before saving.
- Loading:
  1. When the user creates and already existing database it checks if there is any already saved tables inside the database and loads them to the cache after validating them via the DTD files.
- Dropping:
  1. The application deletes the XML and DTD files for the required table to drop.
- *The user must exits using -1 in order to save.*

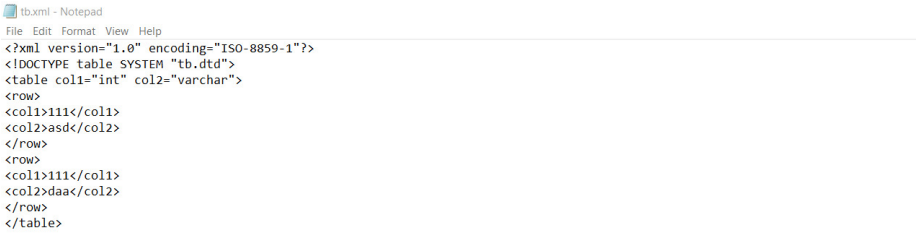
4 Sample Runs

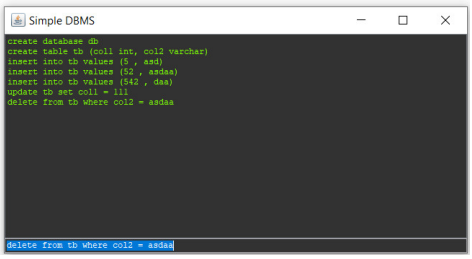



Name
tb.dtd
tb.xml
Simple DBMS
create database db
create table tb (col1 int, col2 varchar)
create table tb (col1 int, col2 varchar)


Name
This folder is empty.


Simple DBMS
create database db
create table tb (col1 int, col2 varchar)
insert into tb values (5 , asd)
insert into tb values (52 , asdaa)
insert into tb values (542 , daa)
update tb set col1 = 111
delete from tb where col2 = asdaa
drop table tbb
drop table tb
drop database dbb
drop database db
drop database db


tb.xml - Notepad
File Edit Format View Help
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE table SYSTEM "tb.dtd">
<table col1="int" col2="varchar">
<row>
<col1>111</col1>
<col2>asd</col2>
</row>
<row>
<col1>111</col1>
<col2>daa</col2>
</row>
</table>


Simple DBMS
create database db
create table tb (col1 int, col2 varchar)
insert into tb values (5 , asd)
insert into tb values (52 , asdaa)
insert into tb values (542 , daa)
update tb set col1 = 111
delete from tb where col2 = asdaa
delete from tb where col2 = asdaa