XML DBMS Project

Dahlia Shehata - Fady Nabil - Mohamed Raafat - Helana Mansour

1. Application Overview

A XML based SQL- like database management system (DBMS) application created by Java programming language, our DBMS supports some simple SQL commands as:

- Use database command (as in mySQL).
- Creating a database.
- Creating a table inside a certain database.
- Inserting data in a table.
- Updating some data in a table.
- Deleting data from a table.
- Selecting (viewing) specified and general data from a table.
- Dropping a database.
- Dropping a table.

• A SQL-like interface that resembles that of real SQL DBMS.
• Files of .xml and .dtd are used to save data and information about it.

2. Application Design

The program is divided to 2 main packages, one is the database and tables functions were each stage from entering the SQL statement till its execution has its one class - following OOP principals in breaking the big task to small independent tasks - , this classes will be explained briefly in the next few lines. The other package is the package responsible for drawing table neatly, this package was downloaded from GitHub.

SQL Statements Parser Class

It class follows the *Singleton Design Pattern* since we only need to create a single object to be used in out program and to control synchronization if threads are used in order to prevent statements execution conflict.

The statement is entered to this class where it checks that the statements a right SQL statement with right syntax and input and if so the suitable function and the SQL command goes to its next step.

Functions Class

It also deals with all SQL orders from creating the database directory and table files (.xml and .dtd) and their equivalent objects and data-structures till dropping a database or table in a certain database, passing through other statements - stated in the project statement - from inserting into table, updating an entry, deleting a table or an entry … etc.

Each single command has its own method to approach final paces in executing the entered command.

XML Handler Class

Final steps are put to have a well built database management system this class is used to handle to / from interaction with table-containing files, as each table has 2 files .xml and .dtd where:

1. Each table (columns and rows) are saved in a well formatted human - readable .xml file with its suitable tags.
2. On loading, load function puts the table's data in a suitable data-structure to be ready for performing any modifications .
3. A .dtd file contains the general overview for each table.

For validating that a .dtd file matches its .xml file for a certain table, a validation method that is implemented for such case to enable tracing an error if it exists.

3. User's Guide

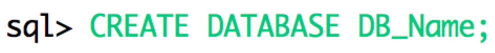
A good application should always be user-friendly through providing a simple, facilitated and realistic user interface (UI) for the user to communicate and interact with the application easily.

We tried to approach a SQL-like UI to complete the working environment we created as a SQL-like DBMS.

Creation Command[1]

1. Database Creation

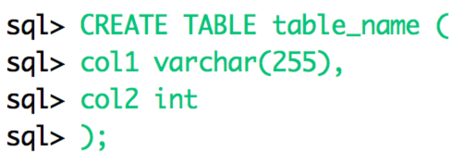
Enter: *CREATE DATABASE DB_Name*[2]*;* to create a new database.



```
sql> CREATE DATABASE DB_Name;
```

2. Table Creation

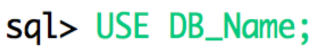
Enter: *CREATE TABLE table_name (col1 varchar(255), col2 int);* to create a new table.



```
sql> CREATE TABLE table_name (
sql> col1 varchar(255),
sql> col2 int
sql> );
```

Use Command

To set certain database available for performing commands ex. creating table, enter:



```
sql> USE DB_Name;
```

[1] SQL commands are case-insensitive, but it is preferred to be in uppercase.

[2] Database names and Tables' names are case-sensitive.

Insert Command

1. Insert Into All Columns[3]
Enter: *INSERT INTO table_name VALUES (value1,value2,value3,...);*

```
sql> INSERT INTO table_name
VALUES (value1,value2,value3,...);
```

2. Insert Into Certain Columns
Enter: *INSERT INTO table_name (column1,column2,column3,…) VALUES (value1,value2,value3,...);*

```
sql> INSERT INTO table_name (column1,column2,column3,...)
VALUES (value1,value2,value3,...);
```

Update Command

1. Update All Rows With the Same Value
Enter: *UPDATE table_name SET column1=value1,column2=value2;*

```
sql> UPDATE table_name
sql> SET column1=value1,column2=value;
```

2. Update Certain Row
Enter: *UPDATE table_name SET column1=value1,column2=value2 WHERE some_column=some_value;*

```
sql> UPDATE table_name
sql> SET column1=value1,column2=value2
sql> WHERE some_column=some_value;
```

Delete Command

1. Erase All Table's Data
Enter: *DELETE FROM table_name; or DELETE * FROM table_name;*

```
sql> DELETE FROM table_name;
```

[3] Enter values in the same order of the columns.

2. Erase Certain Data

Enter: *DELETE FROM table_name WHERE some_column=some_value;*

```
sql> DELETE FROM table_name
WHERE some_column=some_value;
```

Select Command

1. Select All Command

Enter: *SELECT * FROM table_name;*

```
sql> SELECT * FROM table_name;
```

2. Select All with Where

Enter: *SELECT * FROM table_name WHERE column_name operator*[4] *value;*

```
sql> SELECT * FROM table_name WHERE column_name operator value;
```

3. Select Certain Columns

Enter: *SELECT column_name,column_name FROM table_name;*

```
sql> SELECT column_name,column_name FROM table_name;
```

4. Select Certain Columns with Where

Enter: *SELECT column_name,column_name FROM table_name WHERE column_name operator value;*

```
sql> SELECT column_name,column_name FROM table_name
sql> WHERE column_name operator value;
```

[4] =, < or >.

Drop Command

1. **Drop Database**

Enter: *DROP DATABASE db_name;*

```
sql> DROP DATABASE db_name;
```

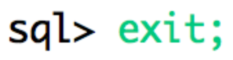
2. **Drop Table**

Enter: *DROP TABLE table_name;*

```
sql> DROP TABLE table_name;
```

Drop Command

Enter: *exit;* to exit from the program.



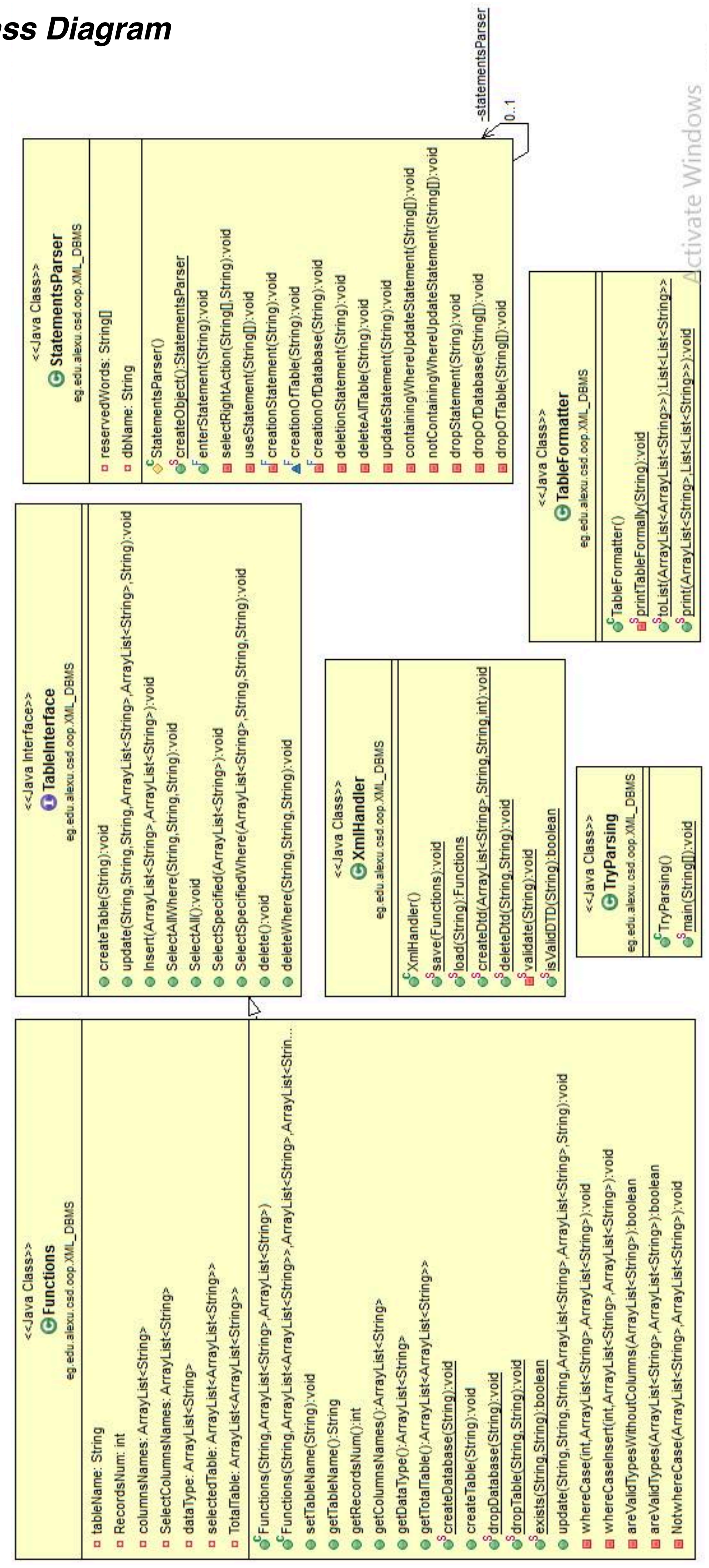
```
sql> exit;
```

4. Code Assumptions

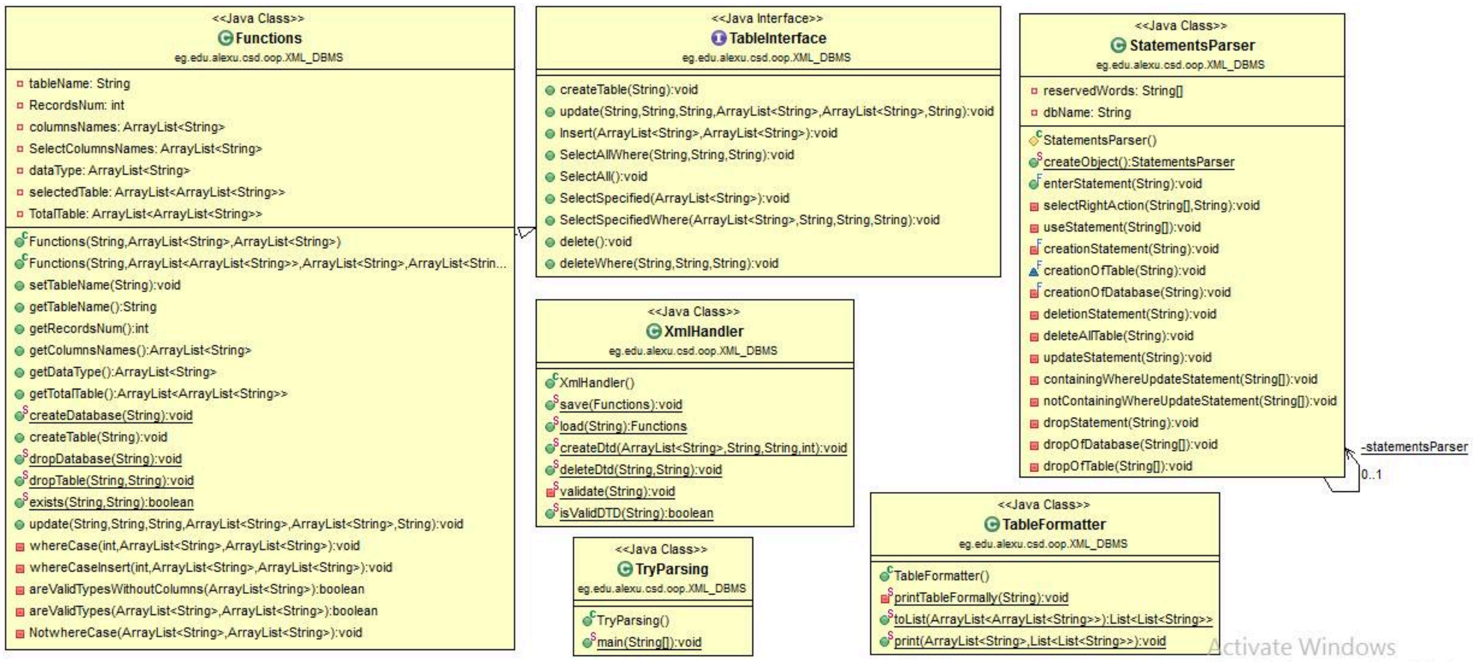
Some assumptions were made to facilitate and pass some stuck problems during code implementation, so we assumed the following:

- All tables' names and database names must be entered with <u>no double or single quotes</u>.
- A string <u>must only</u> be entered between <u>double or single quotes.</u>
- Integers must be entered with <u>no double or single quotes</u>.
- In insertion or update statements, a <u>comma can be written</u>.
- The table is <u>printed after each edit.</u>

5. UML Class Diagram



<<Java Class>>
StatementsParser
eg.edu.alexu.csd.oop.XML_DBMS
reservedWords: String[]
dbName: String
StatementsParser()
createObject():StatementsParser
enterStatement(String):void
selectRightAction(String[],String):void
useStatement(String[]):void
creationStatement(String):void
creationOfTable(String):void
creationOfDatabase(String):void
deletionStatement(String):void
deleteAllTable(String):void
updateStatement(String):void
containingWhereUpdateStatement(String[]):void
notContainingWhereUpdateStatement(String[]):void
dropStatement(String):void
dropOfDatabase(String[]):void
dropOfTable(String[]):void
-statementsParser
0..1
<<Java Interface>>
TableInterface
eg.edu.alexu.csd.oop.XML_DBMS
createTable(String):void
update(String,String,ArrayList<String>,ArrayList<String>,String):void
Insert(ArrayList<String>,ArrayList<String>):void
SelectAllWhere(String,String,String):void
SelectAll():void
SelectSpecified(ArrayList<String>):void
SelectSpecifiedWhere(ArrayList<String>,String,String,String):void
delete():void
deleteWhere(String,String,String):void
<<Java Class>>
XmlHandler
eg.edu.alexu.csd.oop.XML_DBMS
XmlHandler()
save(Functions):void
load(String):Functions
createDtd(ArrayList<String>,String,String,int):void
deleteDtd(String,String):void
validate(String):void
isValidDTD(String):boolean
<<Java Class>>
TableFormatter
eg.edu.alexu.csd.oop.XML_DBMS
TableFormatter()
printTableFormally(String):void
toList(ArrayList<ArrayList<String>>):List<List<String>>
print(ArrayList<String>,List<List<String>>):void
<<Java Class>>
TryParsing
eg.edu.alexu.csd.oop.XML_DBMS
TryParsing()
main(String[]):void
<<Java Class>>
Functions
eg.edu.alexu.csd.oop.XML_DBMS
tableName: String
RecordsNum: int
columnsNames: ArrayList<String>
SelectColumnsNames: ArrayList<String>
dataType: ArrayList<String>
selectedTable: ArrayList<ArrayList<String>>
TotalTable: ArrayList<ArrayList<String>>
Functions(String,ArrayList<String>,ArrayList<String>)
Functions(String,ArrayList<ArrayList<String>>,ArrayList<String>,ArrayList<Strin...
setTableName(String):void
getTableName():String
getRecordsNum():int
getColumnsNames():ArrayList<String>
getDataType():ArrayList<String>
getTotalTable():ArrayList<ArrayList<String>>
createDatabase(String):void
createTable(String):void
dropDatabase(String):void
dropTable(String,String):void
exists(String,String):boolean
update(String,String,String,ArrayList<String>,ArrayList<String>,String):void
whereCase(int,ArrayList<String>,ArrayList<String>):void
whereCaseInsert(int,ArrayList<String>,ArrayList<String>):void
areValidTypesWithoutColumns(ArrayList<String>):boolean
areValidTypes(ArrayList<String>,ArrayList<String>):boolean
NotwhereCase(ArrayList<String>,ArrayList<String>):void


<<Java Class>>
Functions
eg.edu.alexu.csd.oop.XML_DBMS
tableName: String
RecordsNum: int
columnsNames: ArrayList<String>
SelectColumnsNames: ArrayList<String>
dataType: ArrayList<String>
selectedTable: ArrayList<ArrayList<String>>
TotalTable: ArrayList<ArrayList<String>>
Functions(String,ArrayList<String>,ArrayList<String>)
Functions(String,ArrayList<ArrayList<String>>,ArrayList<String>,ArrayList<Strin...
setTableName(String):void
getTableName():String
getRecordsNum():int
getColumnsNames():ArrayList<String>
getDataType():ArrayList<String>
getTotalTable():ArrayList<ArrayList<String>>
createDatabase(String):void
createTable(String):void
dropDatabase(String):void
dropTable(String,String):void
exists(String,String):boolean
update(String,String,String,ArrayList<String>,ArrayList<String>,String):void
whereCase(int,ArrayList<String>,ArrayList<String>):void
whereCaseInsert(int,ArrayList<String>,ArrayList<String>):void
areValidTypesWithoutColumns(ArrayList<String>):boolean
areValidTypes(ArrayList<String>,ArrayList<String>):boolean
NotwhereCase(ArrayList<String>,ArrayList<String>):void
<<Java Interface>>
TableInterface
eg.edu.alexu.csd.oop.XML_DBMS
createTable(String):void
update(String,String,String,ArrayList<String>,ArrayList<String>,String):void
Insert(ArrayList<String>,ArrayList<String>):void
SelectAllWhere(String,String,String):void
SelectAll():void
SelectSpecified(ArrayList<String>):void
SelectSpecifiedWhere(ArrayList<String>,String,String,String):void
delete():void
deleteWhere(String,String,String):void
<<Java Class>>
XmlHandler
eg.edu.alexu.csd.oop.XML_DBMS
XmlHandler()
save(Functions):void
load(String):Functions
createDtd(ArrayList<String>,String,String,int):void
deleteDtd(String,String):void
validate(String):void
isValidDTD(String):boolean
<<Java Class>>
TryParsing
eg.edu.alexu.csd.oop.XML_DBMS
TryParsing()
main(String[]):void
<<Java Class>>
StatementsParser
eg.edu.alexu.csd.oop.XML_DBMS
reservedWords: String[]
dbName: String
StatementsParser()
createObject():StatementsParser
enterStatement(String):void
selectRightAction(String[],String):void
useStatement(String[]):void
creationStatement(String):void
creationOfTable(String):void
creationOfDatabase(String):void
deletionStatement(String):void
deleteAllTable(String):void
updateStatement(String):void
containingWhereUpdateStatement(String[]):void
notContainingWhereUpdateStatement(String[]):void
dropStatement(String):void
dropOfDatabase(String[]):void
dropOfTable(String[]):void
-statementsParser
0..1
<<Java Class>>
TableFormatter
eg.edu.alexu.csd.oop.XML_DBMS
TableFormatter()
printTableFormally(String):void
toList(ArrayList<ArrayList<String>>):List<List<String>>
print(ArrayList<String>,List<List<String>>):void

Thank You