Alexandria University
Faculty of Engineering
Computer and Systems Engineering Dept.
Second Year
Fall 2016

Team:

1. ***Aya Ali Saad Abo Zaid (2)***
2. ***Aya Fouad Metwally (3)***
3. ***Salma Mohammed Elsayed (33)***
4. ***Mona Alaa Darwesh Mostaufa (76)***

We divided our selves into two teams :

1. *Parsing .(Aya Ali Saad and Aya Fouad Metwally)*
2. *Xml queries. (Mona Alaa Darwesh Mostaufa and Salma Mohamed elsayed)*

Report about Simple XML DBMS Assignment

Introduction:

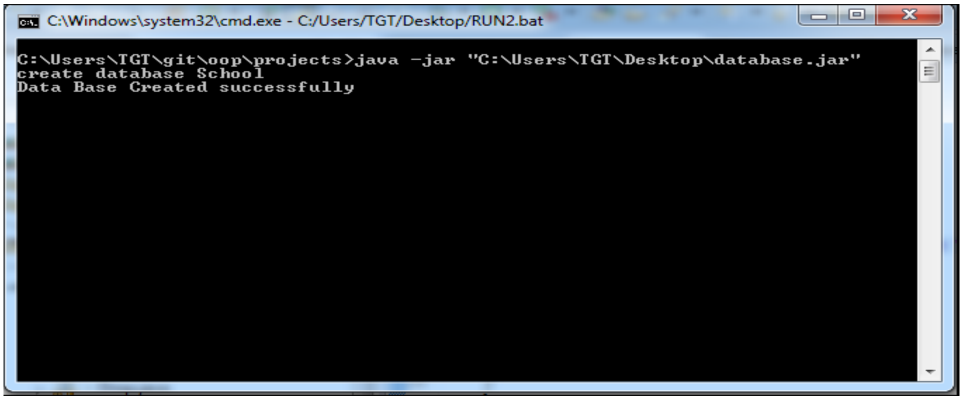
A Computer Database is a structured collection of records or data that is stored in a computer system. On the other hand, a Database Management System (DBMS) is a complex set of software programs that controls the organization, storage, management, and retrieval of data in a database. DBMS are categorized 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. On the other hand, Extensible Markup Language (XML) 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.

We divided the project to two main titles:

- First: Database Machine.

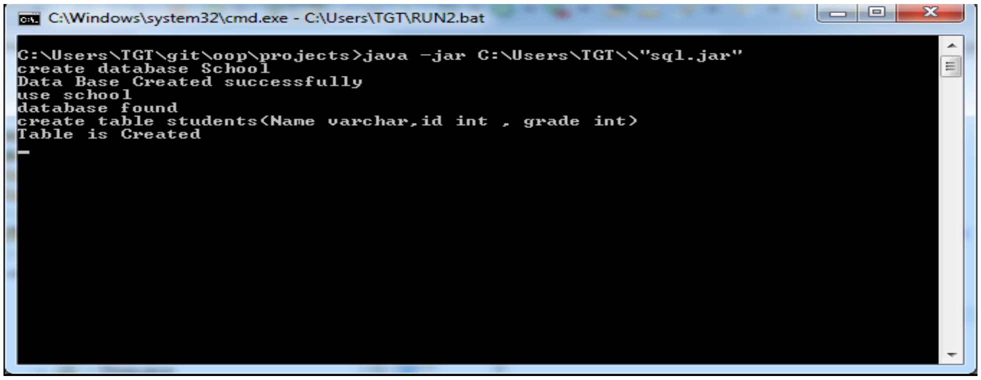
  We made an interface has all the functions that we used to work with the database that the user will enter.

- *We started the data base machine by creating a directory to save all the tables in it (note: you can save the database in reports, flowcharts not only tables) (createDataBase).*
- *In the opposite we wrote a function that can delete this directory (dropDataBase).*

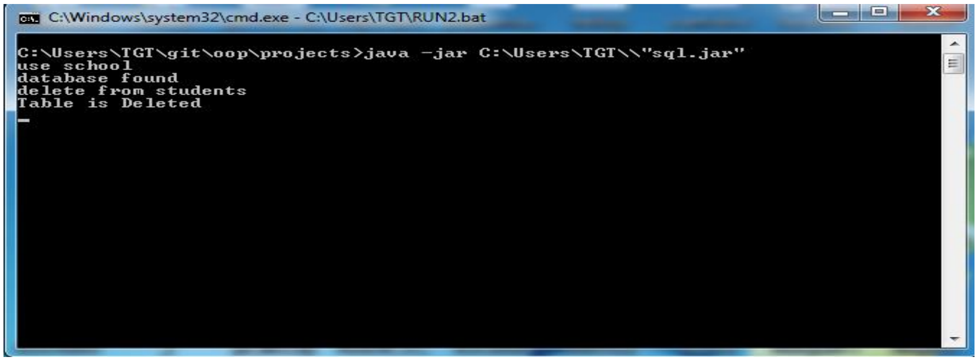


```
C:\Users\TGT\git\oop\projects>java -jar "C:\Users\TGT\Desktop\database.jar"
create database School
Data Base Created successfully
```

- *Then we create table that save all data base in xml file (createTable).*
- *In the opposite we wrote a function to delete a table from the database directory (deleteTable).*



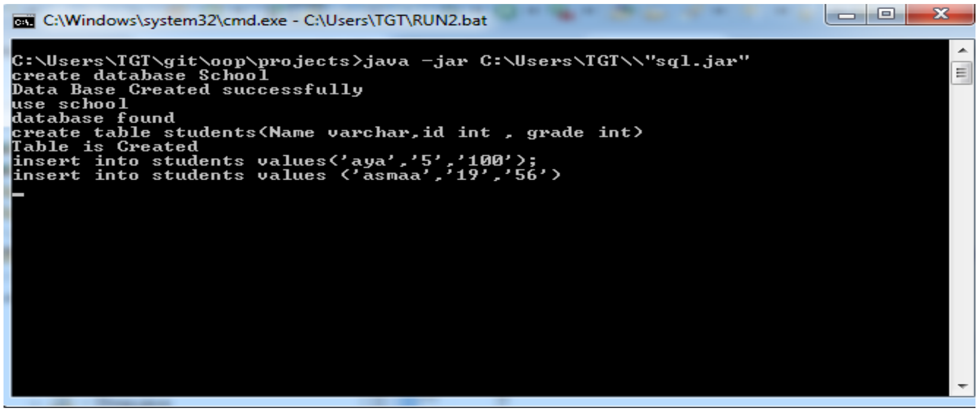
```
C:\Users\TGT\git\oop\projects>java -jar C:\Users\TGT\\"sql.jar"
create database School
Data Base Created successfully
use school
database found
create table students(Name varchar,id int , grade int)
Table is Created
_
```



```
C:\Users\TGT\git\oop\projects>java -jar C:\Users\TGT\\"sql.jar"
use school
database found
delete from students
Table is Deleted
_
```

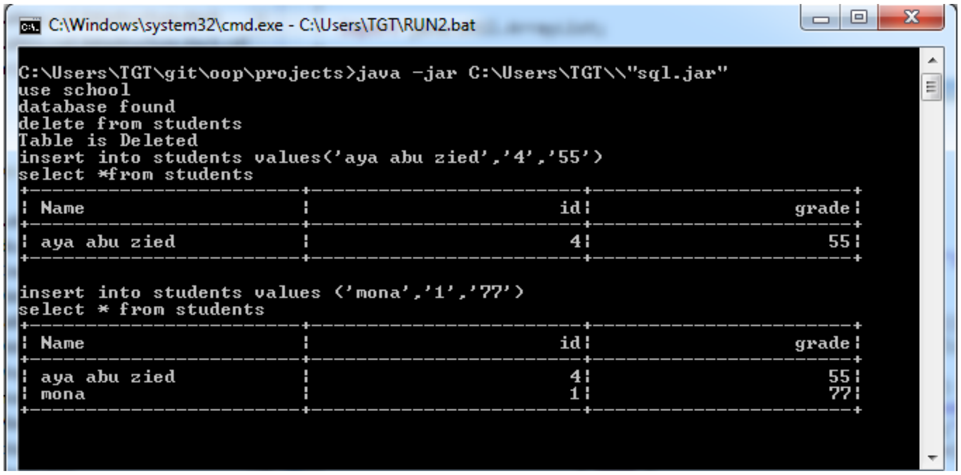
- *Then we add all the queries that the user can want to do on the table like(Insert ,Select,Delete,Update) and every query from them has a lot of roots*

- *Insert: has one root which add row in the table with values that user will enter (insertRow).*



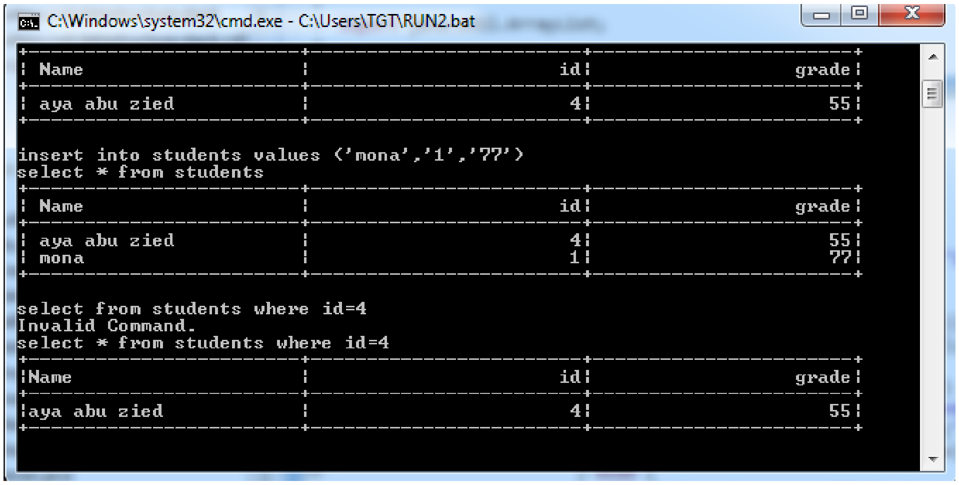
```
C:\Users\TGT\git\oop\projects>java -jar C:\Users\TGT\\"sql.jar"
create database School
Data Base Created successfully
use school
database found
create table students(Name varchar,id int , grade int)
Table is Created
insert into students values('aya','5','100');
insert into students values ('asmaa','19','56')
_
```

- *Select: has four roots which select all columns that print the entire table (selectAllColumns), select all the values of the rows that suitable with special condition given by the user (selectAllWithCondition), select special columns and print the value of these columns in every row (selectColumnsWithCondition) and select some columns and print the values of these columns that for the row that suitable with special condition given by the user (selectColumnsWithCondition).*



```
C:\Users\TGT\git\oop\projects>java -jar C:\Users\TGT\\"sql.jar"
use school
database found
delete from students
Table is Deleted
insert into students values('aya abu zied','4','55')
select *from students
+------------------------+------------------------+-----------------------+
¦ Name                   ¦                      id¦                  grade¦
+------------------------+------------------------+-----------------------+
¦ aya abu zied           ¦                       4¦                     55¦
+------------------------+------------------------+-----------------------+

insert into students values ('mona','1','77')
select * from students
+------------------------+------------------------+-----------------------+
¦ Name                   ¦                      id¦                  grade¦
+------------------------+------------------------+-----------------------+
¦ aya abu zied           ¦                       4¦                     55¦
¦ mona                   ¦                       1¦                     77¦
+------------------------+------------------------+-----------------------+
```

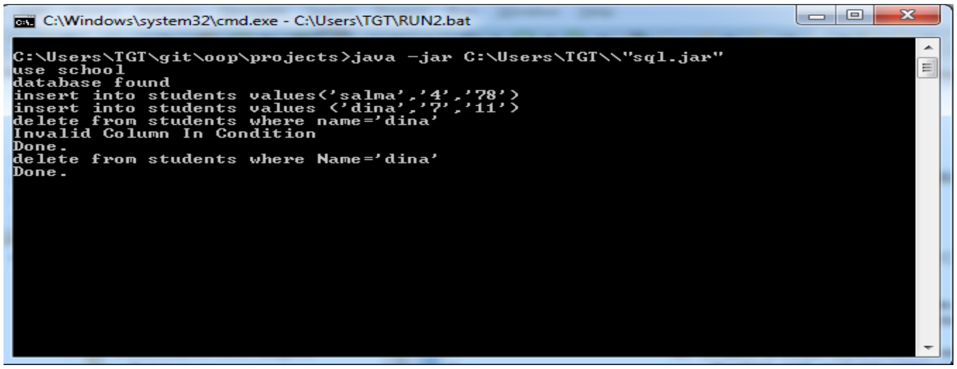


```
+------------------------------+------------------------------+------------------------------+
¦ Name                         ¦                           id¦                         grade¦
+------------------------------+------------------------------+------------------------------+
¦ aya abu zied                 ¦                            4¦                            55¦
+------------------------------+------------------------------+------------------------------+

insert into students values ('mona','1','77')
select * from students
+------------------------------+------------------------------+------------------------------+
¦ Name                         ¦                           id¦                         grade¦
+------------------------------+------------------------------+------------------------------+
¦ aya abu zied                 ¦                            4¦                            55¦
¦ mona                         ¦                            1¦                            77¦
+------------------------------+------------------------------+------------------------------+

select from students where id=4
Invalid Command.
select * from students where id=4
+------------------------------+------------------------------+------------------------------+
¦Name                          ¦                           id¦                         grade¦
+------------------------------+------------------------------+------------------------------+
¦aya abu zied                  ¦                            4¦                            55¦
+------------------------------+------------------------------+------------------------------+
```

- *Delete: has two options first, delete table from database directory and the second is that delete special rows that suitable with the condition given by the user (deleteSubTable).*



```
C:\Users\TGT\git\oop\projects>java -jar C:\Users\TGT\\"sql.jar"
use school
database found
insert into students values('salma','4','78')
insert into students values ('dina','7','11')
delete from students where name='dina'
Invalid Column In Condition
Done.
delete from students where Name='dina'
Done.
```

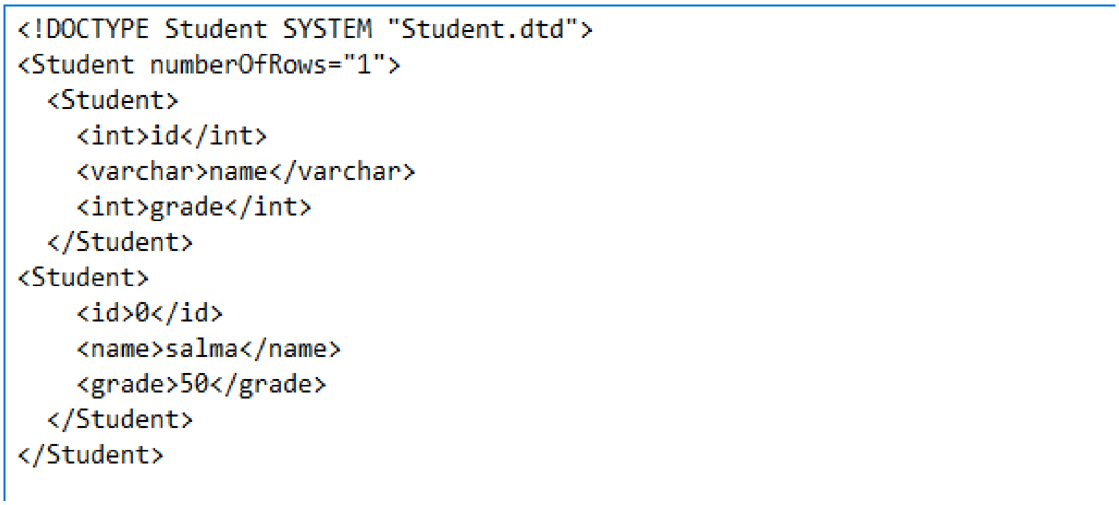
- *Update: change values or rows in the table with special condition (SelectColumnsWithCondition) or without Condition.*

After we saved the data in XMLfile we moved to the second way to save XML file called DTD file, DTD is a way to save XML data and check the validation of XMLfile by the function (validateWithDTDUsingDOM) so when the user create a table the program will create two files with the same name of the table (XML file and DTD file).

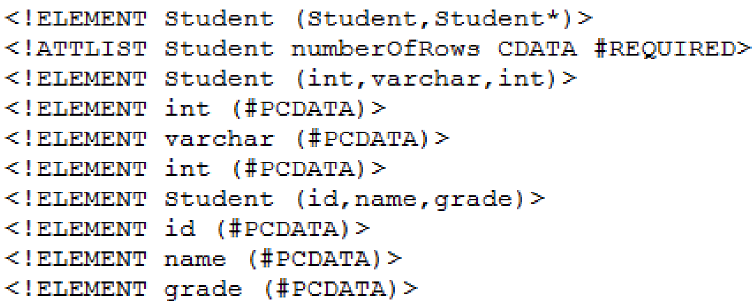
Tables should be validated across their schema files ("DTD Files" can be used for this purpose).

To express the file path we used an environment variable to get the directory path during execution. Example:

```
File table = new File (System.getProperty ("user.home")
+File. Separator +database name +File. Separator + table
name);
```



```
<!DOCTYPE Student SYSTEM "Student.dtd">
<Student numberOfRows="1">
  <Student>
    <int>id</int>
    <varchar>name</varchar>
    <int>grade</int>
  </Student>
<Student>
    <id>0</id>
    <name>salma</name>
    <grade>50</grade>
  </Student>
</Student>
```



```
<!ELEMENT Student (Student,Student*)>
<!ATTLIST Student numberOfRows CDATA #REQUIRED>
<!ELEMENT Student (int,varchar,int)>
<!ELEMENT int (#PCDATA)>
<!ELEMENT varchar (#PCDATA)>
<!ELEMENT int (#PCDATA)>
<!ELEMENT Student (id,name,grade)>
<!ELEMENT id (#PCDATA)>
<!ELEMENT name (#PCDATA)>
<!ELEMENT grade (#PCDATA)>
```

Now we finished database queries that include half of interface methods given by the user, the user entered his queries by SQL Statements (Note: SQL is case insensitive).

- Second: Run Time Machine.

When the user enter his query the statement go to a special class that take every statement given by the user and go to this method.

***Protected void** ChangeToLowerCase (**String** sqlStatement)*

This function is used because of that SQL is case insensitive.

When we make sure of that all the letters in the SQL statement are lowercase and ignore the existence of semicolon or not, this class (Main) send the SQL statement to Parser Class (Parser) to trim all the spaces from the beginning and the end of the statement to deal with this statement without problems.

Then the user starts to write the statements and his query like:

- **USE** database_name: in this statement the user wants to open a directory with the name of database_name so in this case a class does some checks like check of the existence of a directory which name database_name.

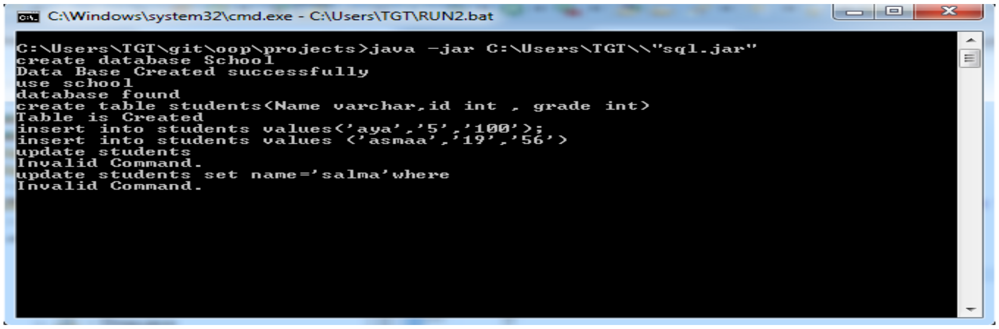
- **Create** table_name (columns_titles): in this statement the program after detect the existence of database directory, creates a table in this directory with the given name.

- Select statements have four options for all the columns of the table **(Select* table_name)** and select all the table with condition**(Select*** table_name **Where** (condition)**)**and select special columns titles with and without condition**(Select** column1,column2 **from** table_name**)(Select** column1,column2 **from** table_name **Where** condition**).**

- Update statements have the same four options in select statement in addition to insert statement and delete statements.

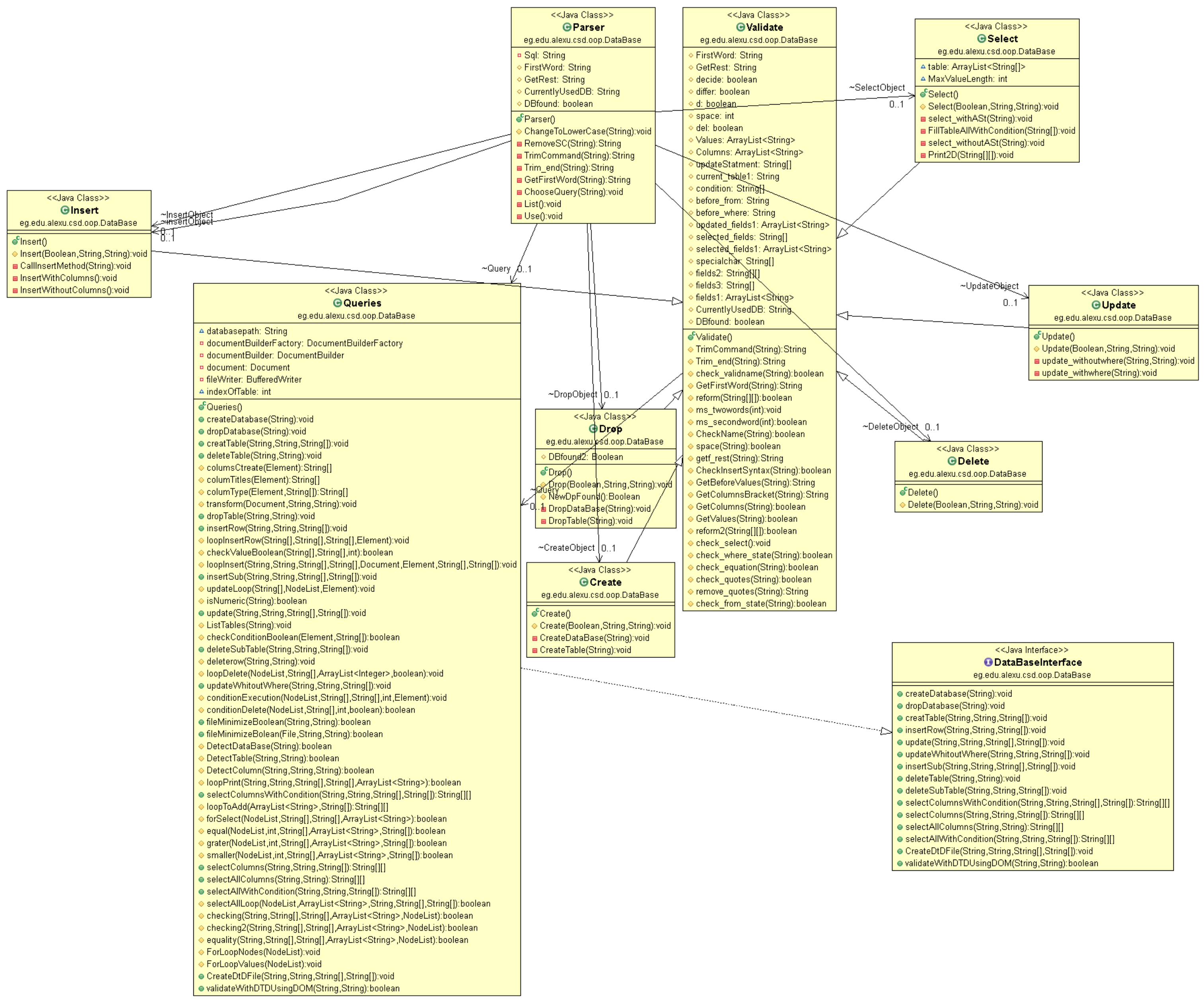
- Every statement has a function in database machine so when the class read special SQL it call the method that do this query .

If the parser class found other statements during execution of the program ,the Parser class prints invalid command



```
C:\Users\TGT\git\oop\projects>java -jar C:\Users\TGT\\"sql.jar"
create database School
Data Base Created successfully
use school
database found
create table students(Name varchar,id int , grade int)
Table is Created
insert into students values('aya','5','100');
insert into students values ('asmaa','19','56')
update students
Invalid Command.
update students set name='salma'where
Invalid Command.
```

UML Diagram:



<<Java Class>>
Parser
eg.edu.alexu.csd.oop.DataBase
Sql: String
FirstWord: String
GetRest: String
CurrentlyUsedDB: String
DBfound: boolean
Parser()
ChangeToLowerCase(String):void
RemoveSC(String):String
TrimCommand(String):String
Trim_end(String):String
GetFirstWord(String):String
ChooseQuery(String):void
List():void
Use():void
<<Java Class>>
Validate
eg.edu.alexu.csd.oop.DataBase
FirstWord: String
GetRest: String
decide: boolean
differ: boolean
d: boolean
space: int
del: boolean
Values: ArrayList<String>
Columns: ArrayList<String>
updateStatment: String[]
current_table1: String
condition: String[]
before_from: String
before_where: String
updated_fields1: ArrayList<String>
selected_fields: String[]
selected_fields1: ArrayList<String>
specialchar: String[]
fields2: String[][]
fields3: String[]
fields1: ArrayList<String>
CurrentlyUsedDB: String
DBfound: boolean
Validate()
TrimCommand(String):String
Trim_end(String):String
check_validname(String):boolean
GetFirstWord(String):String
reform(String[][]):boolean
ms_twowords(int):void
ms_secondword(int):boolean
CheckName(String):boolean
space(String):boolean
getf_rest(String):String
CheckInsertSyntax(String):boolean
GetBeforeValues(String):String
GetColumnsBracket(String):String
GetColumns(String):boolean
GetValues(String):boolean
reform2(String[][]):boolean
check_select():void
check_where_state(String):boolean
check_equation(String):boolean
check_quotes(String):boolean
remove_quotes(String):String
check_from_state(String):boolean
~SelectObject
0..1
<<Java Class>>
Select
eg.edu.alexu.csd.oop.DataBase
table: ArrayList<String[]>
MaxValueLength: int
Select()
Select(Boolean,String,String):void
select_withASt(String):void
FillTableAllWithCondition(String[]):void
select_withoutASt(String):void
Print2D(String[][]):void
~InsertObject
~InsertObject
0..1
0..1
<<Java Class>>
Insert
eg.edu.alexu.csd.oop.DataBase
Insert()
Insert(Boolean,String,String):void
CallInsertMethod(String):void
InsertWithColumns():void
InsertWithoutColumns():void
~Query
0..1
~UpdateObject
0..1
<<Java Class>>
Update
eg.edu.alexu.csd.oop.DataBase
Update()
Update(Boolean,String,String):void
update_withoutwhere(String,String):void
update_withwhere(String):void
<<Java Class>>
Queries
eg.edu.alexu.csd.oop.DataBase
databasepath: String
documentBuilderFactory: DocumentBuilderFactory
documentBuilder: DocumentBuilder
document: Document
fileWriter: BufferedWriter
indexOfTable: int
Queries()
createDatabase(String):void
dropDatabase(String):void
creatTable(String,String,String[]):void
deleteTable(String,String):void
columsCtreate(Element):String[]
columTitles(Element):String[]
columType(Element,String[]):String[]
transform(Document,String,String):void
dropTable(String,String):void
insertRow(String,String,String[]):void
loopInsertRow(String[],String[],String[],Element):void
checkValueBoolean(String[],String[],int):boolean
loopInsert(String,String,String[],String[],Document,Element,String[],String[]):void
insertSub(String,String,String[],String[]):void
updateLoop(String[],NodeList,Element):void
isNumeric(String):boolean
update(String,String,String[],String[]):void
ListTables(String):void
checkConditionBoolean(Element,String[]):boolean
deleteSubTable(String,String,String[]):void
deleterow(String,String):void
loopDelete(NodeList,String[],ArrayList<Integer>,boolean):void
updateWhitoutWhere(String,String,String[]):void
conditionExecution(NodeList,String[],String[],int,Element):void
conditionDelete(NodeList,String[],int,boolean):boolean
fileMinimizeBoolean(String,String):boolean
fileMinimizeBolean(File,String,String):boolean
DetectDataBase(String):boolean
DetectTable(String,String):boolean
DetectColumn(String,String,String):boolean
loopPrint(String,String,String[],String[],ArrayList<String>):boolean
selectColumnsWithCondition(String,String,String[],String[]):String[][]
loopToAdd(ArrayList<String>,String[]):String[][]
forSelect(NodeList,String[],String[],ArrayList<String>):boolean
equal(NodeList,int,String[],ArrayList<String>,String[]):boolean
grater(NodeList,int,String[],ArrayList<String>,String[]):boolean
smaller(NodeList,int,String[],ArrayList<String>,String[]):boolean
selectColumns(String,String,String[]):String[][]
selectAllColumns(String,String):String[][]
selectAllWithCondition(String,String,String[]):String[][]
selectAllLoop(NodeList,ArrayList<String>,String,String[],String[]):boolean
checking(String,String[],String[],ArrayList<String>,NodeList):boolean
checking2(String,String[],String[],ArrayList<String>,NodeList):boolean
equality(String,String[],String[],ArrayList<String>,NodeList):boolean
ForLoopNodes(NodeList):void
ForLoopValues(NodeList):void
CreateDtDFile(String,String,String[],String[]):void
validateWithDTDUsingDOM(String,String):boolean
~DropObject
0..1
<<Java Class>>
Drop
eg.edu.alexu.csd.oop.DataBase
DBfound2: Boolean
Drop()
Drop(Boolean,String,String):void
NewDpFound():Boolean
DropDataBase(String):void
DropTable(String):void
~Query
0..1
~DeleteObject
0..1
<<Java Class>>
Delete
eg.edu.alexu.csd.oop.DataBase
Delete()
Delete(Boolean,String,String):void
~CreateObject
0..1
<<Java Class>>
Create
eg.edu.alexu.csd.oop.DataBase
Create()
Create(Boolean,String,String):void
CreateDataBase(String):void
CreateTable(String):void
<<Java Interface>>
DataBaseInterface
eg.edu.alexu.csd.oop.DataBase
createDatabase(String):void
dropDatabase(String):void
creatTable(String,String,String[]):void
insertRow(String,String,String[]):void
update(String,String,String[],String[]):void
updateWhitoutWhere(String,String,String[]):void
insertSub(String,String,String[],String[]):void
deleteTable(String,String):void
deleteSubTable(String,String,String[]):void
selectColumnsWithCondition(String,String,String[],String[]):String[][]
selectColumns(String,String,String[]):String[][]
selectAllColumns(String,String):String[][]
selectAllWithCondition(String,String,String[]):String[][]
CreateDtDFile(String,String,String[],String[]):void
validateWithDTDUsingDOM(String,String):boolean