DBMS

Data Base Manager System.

BY:

- Omar Mohamed Emam   44
- Islam Yousry   14
- Islam Mohamed   12
- Islam Khlaf   21

Report Content:

1. Program Specifications
2. User Guide
3. UML Diagram
4. Program Description
5. Design Patterns
6. Sample Runs

Database Management System

Program Specifications:

-The Program specify all SQL queries (Create, drop, insert, update, select, delete)

- Program uses dynamic loading for all of your databases, so you can edit them whenever you want

- Program is able to re-load your data no matter where you stored it

- Program is able to understand your SQL query in uppercase or lowercase and with spaces or not

- Program will keep running even if you entered in wrong syntax, you will be able to retry till you want to stop

- Program runs through self-runnable Cmd app

User Guide:

Run "excucation.bat" and enter your SQL query in right syntax and then you can do more queries or exit by typing "end".


SQL CHEAT SHEET http://www.sqltutorial.org

MANAGING TABLES

```
CREATE TABLE t (
    id INT PRIMARY KEY,
    name VARCHAR NOT NULL,
    price INT DEFAULT 0
);
```
Create a new table with three columns

```
DROP TABLE t ;
```
Delete the table from the database

```
ALTER TABLE t ADD column;
```
Add a new column to the table

```
ALTER TABLE t DROP COLUMN c ;
```
Drop column c from the table

```
ALTER TABLE t ADD constraint;
```
Add a constraint

```
ALTER TABLE t DROP constraint;
```
Drop a constraint

```
ALTER TABLE t1 RENAME TO t2;
```
Rename a table from t1 to t2

```
ALTER TABLE t1 RENAME c1 TO c2 ;
```
Rename column c1 to c2

```
TRUNCATE TABLE t;
```
Remove all data in a table

USING SQL CONSTRAINTS

```
CREATE TABLE t(
    c1 INT, c2 INT, c3 VARCHAR,
    PRIMARY KEY (c1,c2)
);
```
Set c1 and c2 as a primary key

```
CREATE TABLE t1(
    c1 INT PRIMARY KEY,
    c2 INT,
    FOREIGN KEY (c2) REFERENCES t2(c2)
);
```
Set c2 column as a foreign key

```
CREATE TABLE t(
    c1 INT, c1 INT,
    UNIQUE(c2,c3)
);
```
Make the values in c1 and c2 unique

```
CREATE TABLE t(
    c1 INT, c2 INT,
    CHECK(c1> 0 AND c1 >= c2)
);
```
Ensure c1 > 0 and values in c1 >= c2

```
CREATE TABLE t(
    c1 INT PRIMARY KEY,
    c2 VARCHAR NOT NULL
);
```
Set values in c2 column not NULL

MODIFYING DATA

```
INSERT INTO t(column_list)
VALUES(value_list);
```
Insert one row into a table

```
INSERT INTO t(column_list)
VALUES (value_list),
       (value_list), ....;
```
Insert multiple rows into a table

```
INSERT INTO t1(column_list)
SELECT column_list
FROM t2;
```
Insert rows from t2 into t1

```
UPDATE t
SET c1 = new_value;
```
Update new value in the column c1 for all rows

```
UPDATE t
SET c1 = new_value,
    c2 = new_value
WHERE condition;
```
Update values in the column c1, c2 that match the condition

```
DELETE FROM t;
```
Delete all data in a table

```
DELETE FROM t
WHERE condition;
```
Delete subset of rows in a table

UML Diagram:
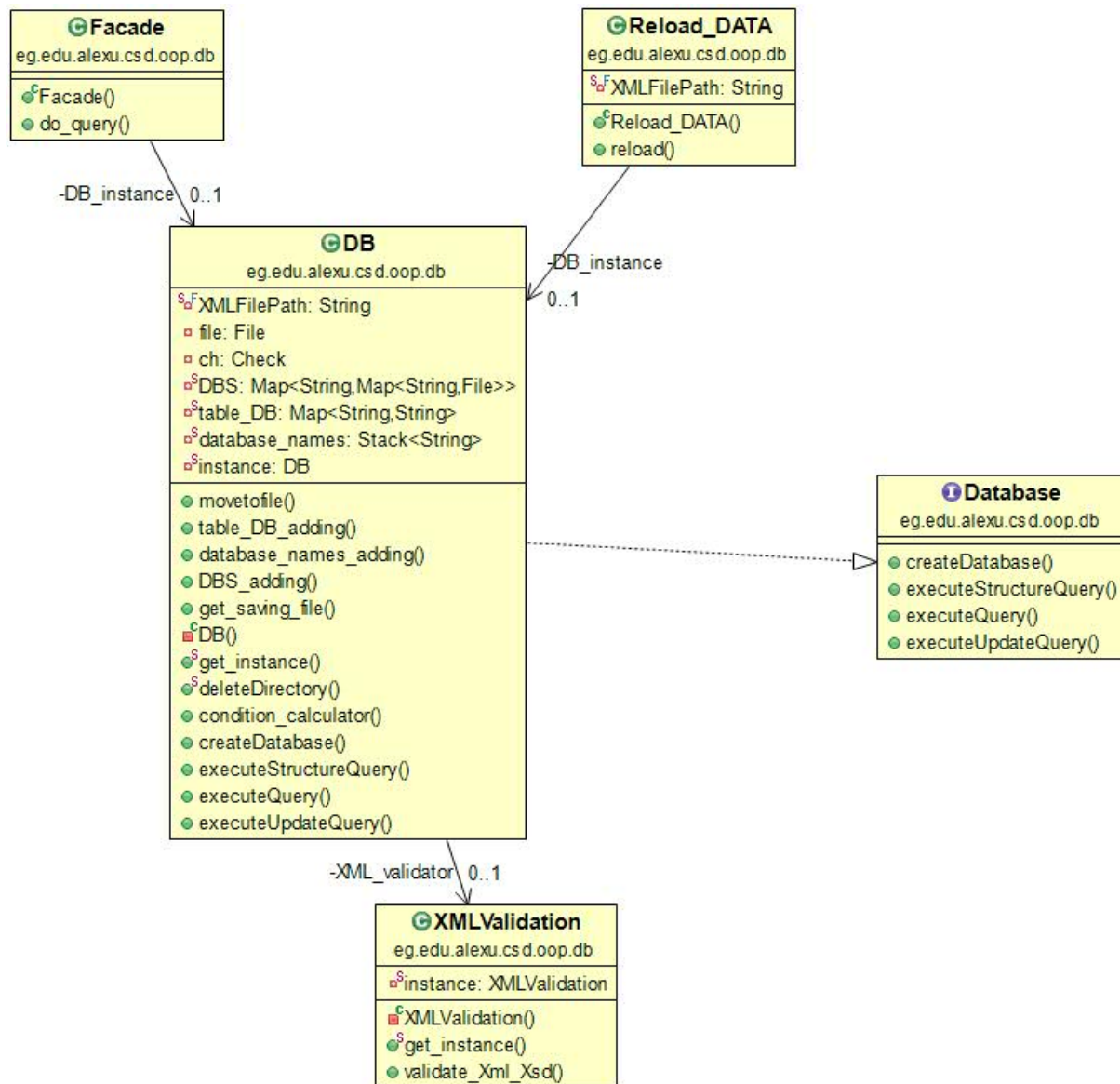
Program Description:

After entering your sql query , it will be check by the regex through :

Check:

The check class is responsible for check if the input that token is in a correct syntax or not.

The check class has 6 function and these functions are : createcheck , updatecheck ,insertcheck , deletcheck ,selectcheck and dropcheck.

And these functions are called by the DB class.

Createcheck:

This function applied on any input start with the word (create) and this function check if the input is (create database database_name) , (create database (path)) , (create table table_name (column_name dataType)) by three regex and then it takes the wanted information by (split and regex) and return it in array and if the input was wrong syntax it returns null.

updatecheck:

This function applied on any input start with the word (update) and this function check if the input is (UPDATE *table_name* SET *column1 = value1, column2 = value2, ...* WHERE *condition*) or (UPDATE *table_name* SET *column1 = value1, column2 = value2, ...*) by two regex and then it takes the wanted information by (split and regex) and return it in array and if the input was wrong syntax it returns null.

insertcheck:

This function applied on any input start with the word (insert) and this function check if the input is (INSERT INTO *table_name* (*column1, column2, column3, ...*) VALUES (*value1, value2, value3, ...*)) or(INSERT INTO *table_name* VALUES (*value1, value2, value3, ...*))   by two regex and then it takes the wanted information by (split and regex) and return it in array and if the input was wrong syntax it returns null.

deletcheck:

This function applied on any input start with the word (delet) and this function check if the input is (DELETE FROM *table_name* WHERE *condition*) or (DELETE FROM *table_name*) by two regex and then it takes the wanted information by (split and regex) and return it in array and if the input was wrong syntax it returns null.

selectcheck:

This function applied on any input start with the word (select) and this function check if the input is (SELECT *column1, column2, ...* FROM *table_name*) or (SELECT *column1, column2, ...* FROM *table_name where condition*) or (SELECT * FROM *table_name*) or (SELECT * FROM *table_name where condition*) by four regex and then it takes the wanted information by (split and regex) and return it in array and if the input was wrong syntax it returns null.

dropcheck:

This function applied on any input start with the word (drop) and this function check if the input is (DROP DATABASE *databasename*) or (DROP TABLE *table_name*) by two regex and then it takes the wanted information by (split and regex) and return it in String and if the input was wrong syntax it returns null.

After Checking The Query:

We use Java DOM(Document Object Model) to create and parse XML files by representing the document in tree structure.

We use Java XSD (XML Schema Definition) to describe the structure of the xml file(data types of each element and the sequence of elements of each row)

The class XMLValidation is used to determine whatever the xml file is good structured or not

Design Patterns:

1- **Facade design pattern**: used to decide which method in the interface execute dependent on the input query.

2- **Singleton design pattern**: to make one instance of object.

Sample Runs:

First run & generated xml

```
D:\DBMS>java -jar DBMS.jar
****************************Welcome TO Our DBMS****************************
Enter Your SQL query in right synatx (to exit just right'end'):
create database test_database
DataBase>> test_database << is Created and Saved
Enter Your SQL query in right synatx (to exit just right'end'):
create table test_table (col1 int , col2 varchar , col3 int , col4 varchar )
Table>> test_table << is Created
Enter Your SQL query in right synatx (to exit just right'end'):
insert into test_table (col1 ) values (100)
You Inserted Into >> test_table
Enter Your SQL query in right synatx (to exit just right'end'):
insert into test_table (col2 , col4 ) values ('try','tryagian')
You Inserted Into >> test_table
Enter Your SQL query in right synatx (to exit just right'end'):
insert into test_table (col3 ) values (300)
You Inserted Into >> test_table
Enter Your SQL query in right synatx (to exit just right'end'):
update test_table set col3 = 500 where col3 = 300
You Updated >> test_table
Enter Your SQL query in right synatx (to exit just right'end'):
end
****************************Hope You Enjoyed Our DBMS****************************

D:\DBMS>pause
Press any key to continue . . .
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
- <test_table>
    - <table_details>
        <col1 type="integer"/>
        <col2 type="string"/>
        <col3 type="integer"/>
        <col4 type="string"/>
      </table_details>
    - <id>
        <col1>100</col1>
      </id>
    - <id>
        <col2>try</col2>
        <col4>tryagian</col4>
      </id>
    - <id>
        <col3>500</col3>
      </id>
  </test_table>
```

After Reloading and editing

```
D:\DBMS>java -jar DBMS.jar
****************************Welcome TO Our DBMS****************************
Enter Your SQL query in right synatx (to exit just right'end'):
select * from test_table
selected values:
===============================================
100     null    null    null
===============================================
null    try     null    tryagian
===============================================
null    null    500     null
===============================================
Enter Your SQL query in right synatx (to exit just right'end'):
drop table test_table
Enter Your SQL query in right synatx (to exit just right'end'):
drop database test_database
DataBase>> test_database << is Dropped and Deleted
Enter Your SQL query in right synatx (to exit just right'end'):
end
****************************Hope You Enjoyed Our DBMS****************************

D:\DBMS>pause
Press any key to continue . . .
```

```
Enter Your SQL query in right synatx (to exit just right'end'):
create database second_test
DataBase>> second_test << is Created and Saved
Enter Your SQL query in right synatx (to exit just right'end'):
create table t1 (c1 int , c2 varchar)
Table>> t1 << is Created
Enter Your SQL query in right synatx (to exit just right'end'):
insert into t1 (c1 ,c2 ) values (50 , 'new')
You Inserted Into >> t1
Enter Your SQL query in right synatx (to exit just right'end'):
select * from t1
selected values:
===========================================
50      new
===========================================
Enter Your SQL query in right synatx (to exit just right'end'):
create table t2 (c1 int , c2 int ,c3 int)
Table>> t2 << is Created
Enter Your SQL query in right synatx (to exit just right'end'):
insert into t2 (c3) values (565)
You Inserted Into >> t2
Enter Your SQL query in right synatx (to exit just right'end'):
drop table t2
Table> t2 << is Dropped
Enter Your SQL query in right synatx (to exit just right'end'):
end
***************************Hope You Enjoyed Our DBMS***************************

D:\DBMS>pause
Press any key to continue . . .
```