NOVEMBER 29, 2016

XML DBMS APPLICATION REPORT

PRESENTED BY:

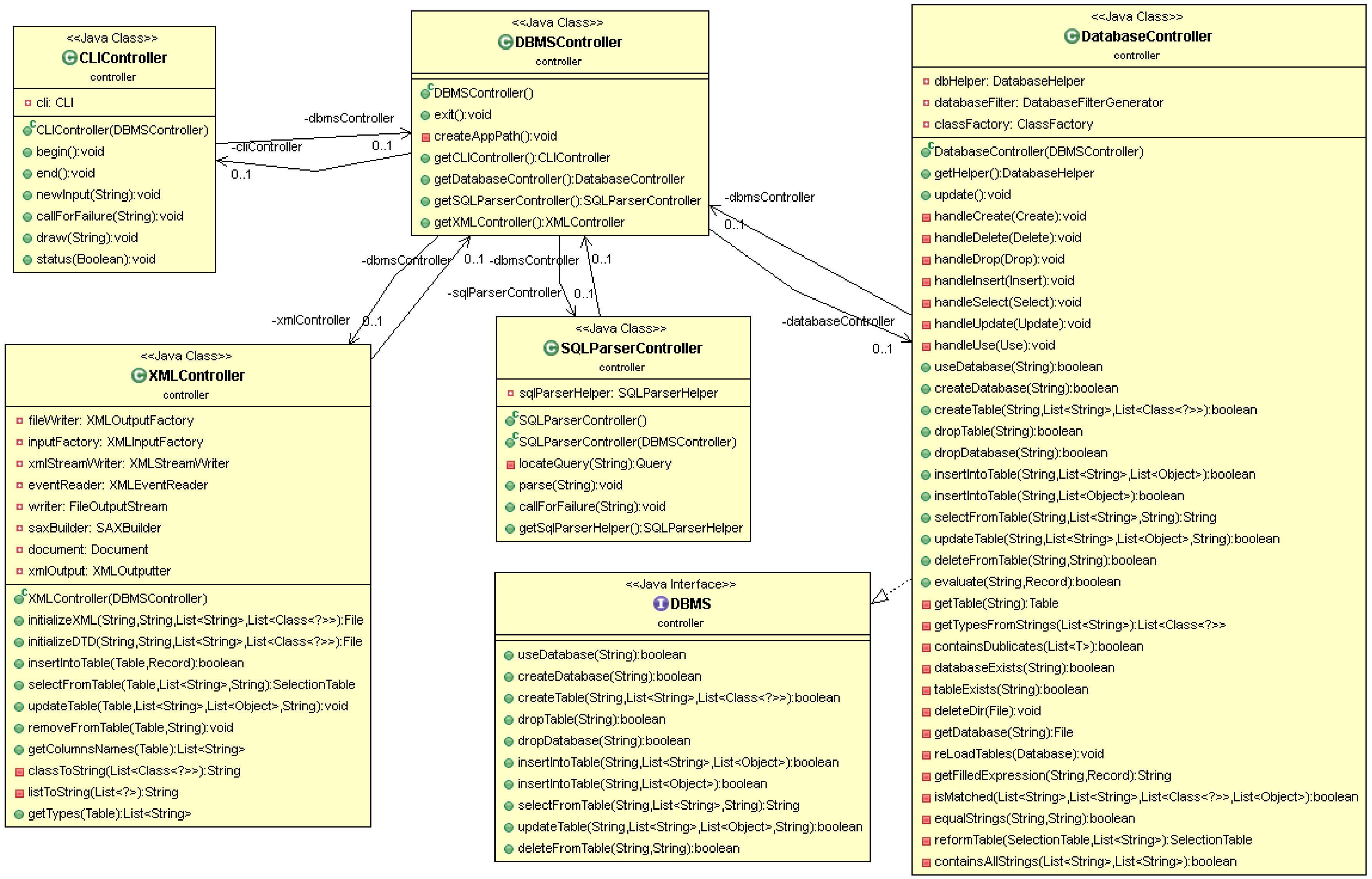
AHMED MAGDY

HESHAM EL SAWAF

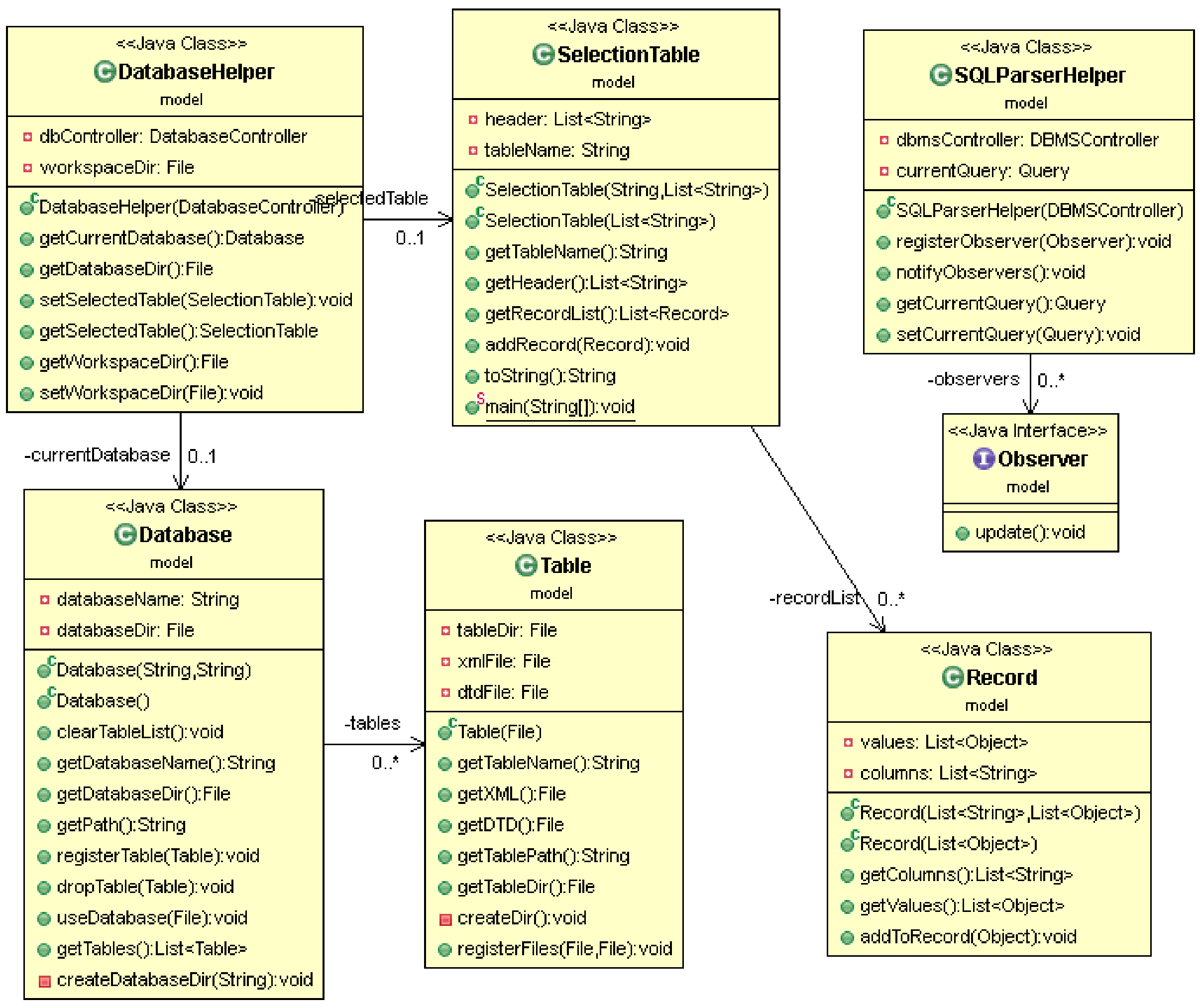
MAHMOUD HUSSEIN

MARWAN TAMMAM

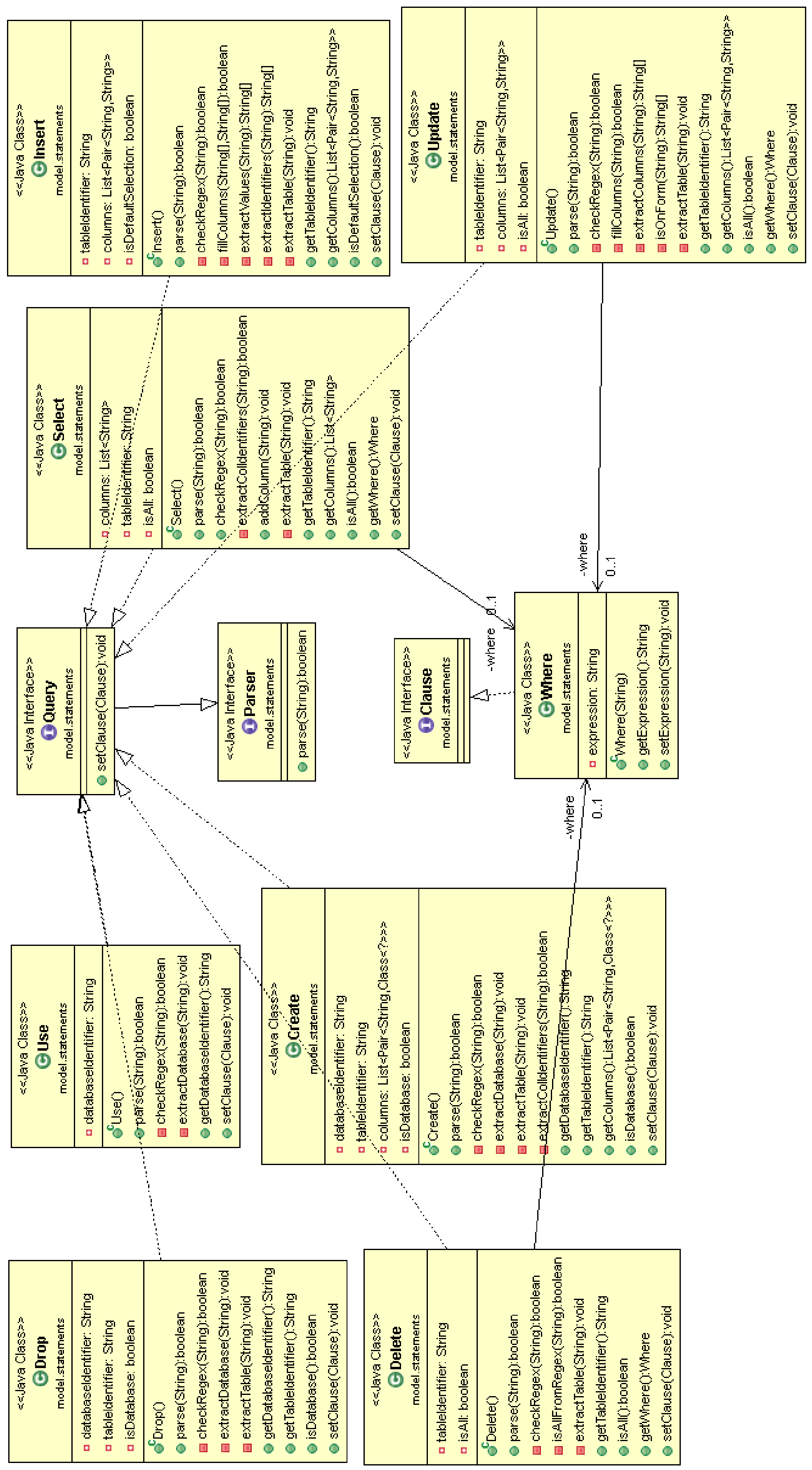
UML DIAGRAM 3


<<Java Class>>
CLIController
controller
cli: CLI
CLIController(DBMSController)
begin():void
end():void
newInput(String):void
callForFailure(String):void
draw(String):void
status(Boolean):void
<<Java Class>>
DBMSController
controller
DBMSController()
exit():void
createAppPath():void
getCLIController():CLIController
getDatabaseController():DatabaseController
getSQLParserController():SQLParserController
getXMLController():XMLController
<<Java Class>>
DatabaseController
controller
dbHelper: DatabaseHelper
databaseFilter: DatabaseFilterGenerator
classFactory: ClassFactory
DatabaseController(DBMSController)
getHelper():DatabaseHelper
update():void
handleCreate(Create):void
handleDelete(Delete):void
handleDrop(Drop):void
handleInsert(Insert):void
handleSelect(Select):void
handleUpdate(Update):void
handleUse(Use):void
useDatabase(String):boolean
createDatabase(String):boolean
createTable(String,List<String>,List<Class<?>>):boolean
dropTable(String):boolean
dropDatabase(String):boolean
insertIntoTable(String,List<String>,List<Object>):boolean
insertIntoTable(String,List<Object>):boolean
selectFromTable(String,List<String>,String):String
updateTable(String,List<String>,List<Object>,String):boolean
deleteFromTable(String,String):boolean
evaluate(String,Record):boolean
getTable(String):Table
getTypesFromStrings(List<String>):List<Class<?>>
containsDublicates(List<T>):boolean
databaseExists(String):boolean
tableExists(String):boolean
deleteDir(File):void
getDatabase(String):File
reLoadTables(Database):void
getFilledExpression(String,Record):String
isMatched(List<String>,List<String>,List<Class<?>>,List<Object>):boolean
equalStrings(String,String):boolean
reformTable(SelectionTable,List<String>):SelectionTable
containsAllStrings(List<String>,List<String>):boolean
<<Java Class>>
XMLController
controller
fileWriter: XMLOutputFactory
inputFactory: XMLInputFactory
xmlStreamWriter: XMLStreamWriter
eventReader: XMLEventReader
writer: FileOutputStream
saxBuilder: SAXBuilder
document: Document
xmlOutput: XMLOutputter
XMLController(DBMSController)
initializeXML(String,String,List<String>,List<Class<?>>):File
initializeDTD(String,String,List<String>,List<Class<?>>):File
insertIntoTable(Table,Record):boolean
selectFromTable(Table,List<String>,String):SelectionTable
updateTable(Table,List<String>,List<Object>,String):void
removeFromTable(Table,String):void
getColumnsNames(Table):List<String>
classToString(List<Class<?>>):String
listToString(List<?>):String
getTypes(Table):List<String>
<<Java Class>>
SQLParserController
controller
sqlParserHelper: SQLParserHelper
SQLParserController()
SQLParserController(DBMSController)
locateQuery(String):Query
parse(String):void
callForFailure(String):void
getSqlParserHelper():SQLParserHelper
<<Java Interface>>
DBMS
controller
useDatabase(String):boolean
createDatabase(String):boolean
createTable(String,List<String>,List<Class<?>>):boolean
dropTable(String):boolean
dropDatabase(String):boolean
insertIntoTable(String,List<String>,List<Object>):boolean
insertIntoTable(String,List<Object>):boolean
selectFromTable(String,List<String>,String):String
updateTable(String,List<String>,List<Object>,String):boolean
deleteFromTable(String,String):boolean
-dbmsController
-cliController
0..1
0..1
-dbmsController
0..1
-databaseController
0..1
-dbmsController
0..1
-xmlController
0..1
-dbmsController
0..1
-sqlParserController
0..1

UML DIAGRAM 3



UML DIAGRAM 3



<<Java Class>>
Insert
model.statements
tableIdentifier: String
columns: List<Pair<String,String>>
isDefaultSelection: boolean
Insert()
parse(String):boolean
checkRegex(String):boolean
fillColumns(String[],String[]):boolean
extractValues(String):String[]
extractIdentifiers(String):String[]
extractTable(String):void
getTableIdentifier():String
getColumns():List<Pair<String,String>>
isDefaultSelection():boolean
setClause(Clause):void
<<Java Class>>
Update
model.statements
tableIdentifier: String
columns: List<Pair<String,String>>
isAll: boolean
Update()
parse(String):boolean
checkRegex(String):boolean
fillColumns(String):boolean
extractColumns(String):String[]
isOnForm(String):String[]
extractTable(String):void
getTableIdentifier():String
getColumns():List<Pair<String,String>>
isAll():boolean
getWhere():Where
setClause(Clause):void
<<Java Class>>
Select
model.statements
columns: List<String>
tableIdentifier: String
isAll: boolean
Select()
parse(String):boolean
checkRegex(String):boolean
extractColIdentifiers(String):boolean
addColumn(String):void
extractTable(String):void
getTableIdentifier():String
getColumns():List<String>
isAll():boolean
getWhere():Where
setClause(Clause):void
-where
0..1
<<Java Interface>>
Query
model.statements
setClause(Clause):void
<<Java Interface>>
Parser
model.statements
parse(String):boolean
<<Java Interface>>
Clause
model.statements
<<Java Class>>
Where
model.statements
expression: String
Where(String)
getExpression():String
setExpression(String):void
<<Java Class>>
Use
model.statements
databaseIdentifier: String
Use()
parse(String):boolean
checkRegex(String):boolean
extractDatabase(String):void
getDatabaseIdentifier():String
setClause(Clause):void
<<Java Class>>
Create
model.statements
databaseIdentifier: String
tableIdentifier: String
columns: List<Pair<String,Class<?>>>
isDatabase: boolean
Create()
parse(String):boolean
checkRegex(String):boolean
extractDatabase(String):void
extractTable(String):void
extractColIdentifiers(String):boolean
getDatabaseIdentifier():String
getTableIdentifier():String
getColumns():List<Pair<String,Class<?>>>
isDatabase():boolean
setClause(Clause):void
<<Java Class>>
Drop
model.statements
databaseIdentifier: String
tableIdentifier: String
isDatabase: boolean
Drop()
parse(String):boolean
checkRegex(String):boolean
extractDatabase(String):void
extractTable(String):void
getDatabaseIdentifier():String
getTableIdentifier():String
isDatabase():boolean
setClause(Clause):void
<<Java Class>>
Delete
model.statements
tableIdentifier: String
isAll: boolean
Delete()
parse(String):boolean
checkRegex(String):boolean
isAllFromRegex(String):boolean
extractTable(String):void
getTableIdentifier():String
isAll():boolean
getWhere():Where
setClause(Clause):void

INTRODUCTION

This project is a simple database management system that takes as input SQL statements from the user in the terminal window and parses executes them handling data stored in XML files.

The supported SQL statements are as follow:

- Create database
- Create table
- Insert into table
- Delete from table
- Drop database
- Drop table
- Select from table
- Update table
- Use table

Each table in a database is stored in two files, an XML file for the data, and a DTD file for the table structure.

USER GUIDE

Upon launching the project the user is presented with a prompt displaying his current directory.

SQL statements are case insensitive. The user enters any of the supported SQL statements and the parser evaluates them. If the statement is false the user is notified with an appropriate error message. Else the parser executes the statement.

Before beginning to deal with tables, the user should enter the USE statement to select the database first.

DESIGN

The main interface of the application is the DBMS interface which contains the following methods:

public boolean useDatabase(String databaseName);

public boolean createDatabase(String databaseName);

public boolean createTable(String tableName, List<String> colNames, List<Class<?>> types);

public boolean dropTable(String tableName);

public boolean dropDatabase(String databaseName);

public boolean insertIntoTable(String tableName, List<String> colNames, List<String> values);

public boolean insertIntoTable(String tableName, List<String> values);

public String selectFromTable(String tableName, List<String> colNames, String condition);

public boolean updateTable(String tableName, List<String> colNames, List<String> values, String condition);

public boolean deleteFromTable(String tableName, String condition);

The database controller implements this interface.

XML CONTROLLER

Applies the queries on the XML files of the database directly.

BOOLEAN EVALUATOR

Evaluates a Boolean expression returning true, false or throws an exception if the expression is invalid.

REGEX EVALUATOR

Evaluates a given pattern on a given string and return matching groups.

SQLPARSERCONTROLLER

Parses the input SQL statements and check the syntax. If correct create a query object and pass it to the SQLParserHelper.

SQLPARSERHELPER

Receives a query object from the SQLParserController and notifies the database controller.

XMLPARSER

Data are written on hard desk and being operated on using SAX, STAX and DOM XML Parsers.

OOP CONCEPTS

DESIGN PATTERNS

OBSERVER DESIGN PATTERN

Used with the SQL parser to notify the database controller that the user has entered a new statement, and that the XML parser should execute the statement and update the files on disk if necessary.

FACTORY DESIGN PATTERN

ClassFactory: given a string representing a class name returns an object of that class if it is a valid data type class supported by the application.

ObjectFactory: given a string representing a specific data type, returns an object of that datatype after casting/parsing it.

MVC

The project uses the MVC design pattern where the model represents the factories, the table columns, the records and the table. In addition, the model represents the classes of the different statements and the where condition.

The view part consists of the CLI class responsible for displaying the prompts and asking the controller to print the table.

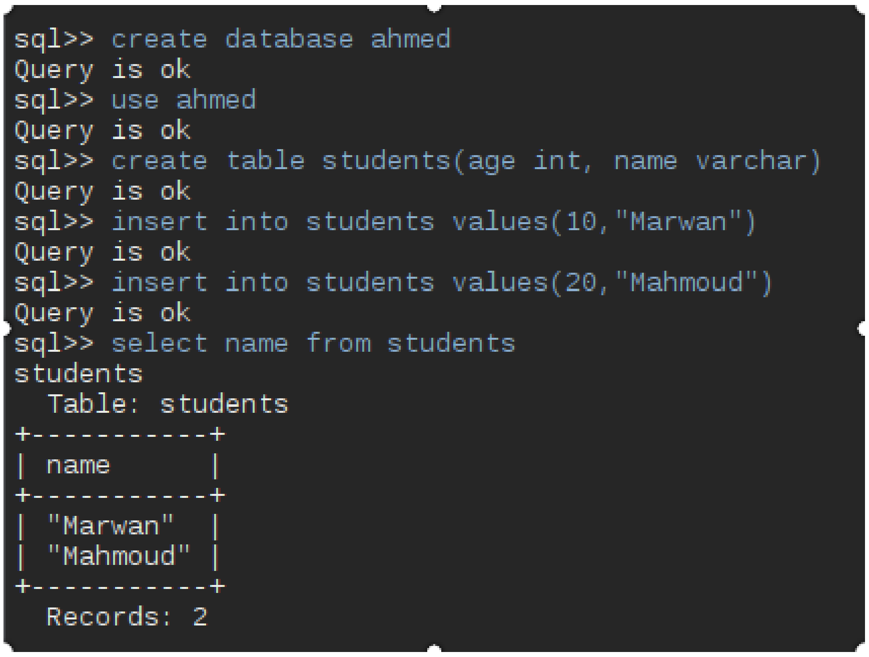
The controller consists mainly of 2 classes. The SQLParser which is responsible for parsing the statements and creating appropriate query objects depending on the statement. The second main part in the controller is the database controller which implements the main interface of the application, the DBMS interface. This class executes the statements. The XML controller is also a main class which deals with the data of the tables stored on the disk.

ASSUMPTIONS & EXTRA FEATURES

- Equality is supported only using “==” sign.
- Support AND, OR and NOT operations using literals or signs.
- All comparisons are supported “<=”, “>=” , “==” , “!=” , “>”, “<”.

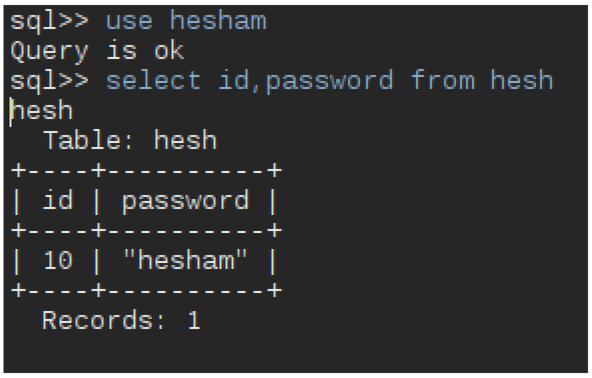
SCREENSHOTS

- Creating a database, use it, create a table into it and insert two new records.



```
sql>> create database ahmed
Query is ok
sql>> use ahmed
Query is ok
sql>> create table students(age int, name varchar)
Query is ok
sql>> insert into students values(10,"Marwan")
Query is ok
sql>> insert into students values(20,"Mahmoud")
Query is ok
sql>> select name from students
students
  Table: students
+-----------+
| name      |
+-----------+
| "Marwan"  |
| "Mahmoud" |
+-----------+
  Records: 2
```

- Using a previously stored database with selecting a table.



```
sql>> use hesham
Query is ok
sql>> select id,password from hesh
hesh
  Table: hesh
+----+----------+
| id | password |
+----+----------+
| 10 | "hesham" |
+----+----------+
  Records: 1
```