**Universitatea Alexandru Ioan Cuza Iași**

**FACULTATEA DE INFORMATICĂ**

**DataTools - Database Implementation**

Authors:

Ecaterina-Mihaela Chilaboc

Vladimir Ventaniuc

**Content**

1. **Problem Definition**
2. **Design and Programming**
3. **Testing**
4. **Assertions**
5. **User Manual**
6. **Contribution**
7. **Problem Definition**

A database is a collection of [information](https://searchsqlserver.techtarget.com/definition/information) that is organized so that it can be easily accessed, managed and updated.

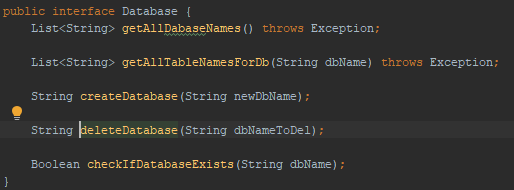
Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added. Databases process workloads to create and update themselves, querying the data they contain and running applications against it.

This project propose a database implementation that will support all common operations that can be performed on a classic database. The proposed syntax is similar to the SQL.

Users can perform actions upon database using the command line interface that supports SQL queries or they can use the graphic interface.

1. **Design and Programming**

The project is divided into two main parts, the command line and the graphic interface. Both parts use the classes from the commands folder. Under interfaces directory we have two interfaces defined. The Database interface contains the definition of all operations that can be performed upon the database.



A database is a directory that is created in the resources directory. We have defined 5 operations as above:

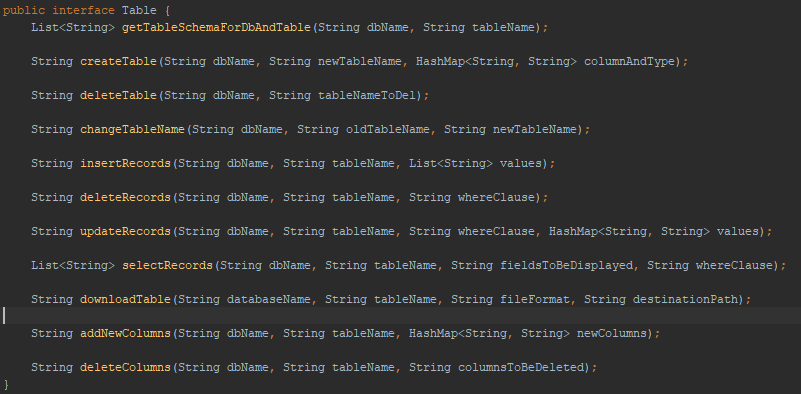
* getAllDatabaseNames – returns a list of strings that contains the name of each directory created under the resource directory.
* getAllTableNamesForDB - recursively browses the directory that is given as a parameter and returns the name of all tables stored in it. A table is stored as an xml file.
* createDatabase – a new directory will be created with the name given as a parameter. An error message is throwed if already a directory with the same name exists.
* deleteDatabase – delete the directory and all the content from it. Can throw an error message if the directory doesn’t exists.
* checkIfDatabaseExists – return true if the directory exist or false if not.

A table is a xml file that have the following structure:



* First line contains the XML declaration.
* First node contains the table definition and all the entries that are stored in the able.
* Name node contains the name of the database.
* NumberOfFields contains the number of columns from the table.
* Fields node has multiple nodes. Each node is a column of the table. Tag is the name of the column and the value from the tag is the accepted type.
* Entries contains nodes of type Entry. An entry had an id that is automatically assigned and the value for each column.

Table interface contains operations that can be performed upon a table.



* getTableSchemaForDbAndTable – takes all the nodes from Fields node. The result is a list that contains the column name concatenated with the type.
* createTable – a new xml file is created in the given database. The file has the format from above. Fields are given as a map in which the key represents the column name and the value is the type.
* deleteTable – deletes a certain file.
* changeTableName – rename a file if the new name is not used by another table. An error message is throwed if so.
* insertRecords – new entries are stored under Entries node. Values are given as a list of strings. The list size needs to have the same size as the columns number from the file in order to not receive an error message.
* deleteRecords – deletes nodes from Entries when the conditions from where clause are met. whereClause is a string that can have multiple conditions. Conditions needs to be separated by “or” or “and” and are interpreted by WhereClauseValidator class.
* updateRecords – entries are modified when the conditions from where clause are met. A map that contains as key the field and the new values is provided.
* selectRecords – works the same as the update but a list with the entries will be returned. We can specify what fields we want to be returned in fieldsToBeDisplayed string. If we pass the “\*” string all the fields will be displayed.
* downloadTable – copy a table on the given path. fileFormat string specify the format of the new file. If the format is XML the file is copied as it is. If the format is JSON, first we convert the file to the desired format and after this the copying process is made. Another formats are not supported, so an error message is displayed.
* addNewColumns – this method alter a table structure by adding the fields provided as a map ( same map that is used on creation ). Fields are added under the Fields node and also under Entries. When we add new column to Entries the value will be the empty string.
* deleteColumns – deletes the provided fields from Fields and Entries.

Under Implementation directory we have two classes: DatabaseImpl and TestImpl that implements the interfaces described above.

Helpers directory contains various classes that are used in TestImpl class:

* Condition class – representation of a condition. Has 3 variable: field – name of a field, value – value that will be checked and equal – a Boolean that specify if a field value needs to be equal or not with the value from condition ( currently we support just “=” and his negation )
* Entry – representation of nodes that are stored under Entities. Contains a map in which the key is the field name and the value is the value associated to the column, line represents the line number at which the entry is stored.
* EntityConvertor – a class that creates Entry objects based on the given properties.
* WhereClauseValidator – check all the entries that needs to be updated, selected or deleted based on the provided where clause. A list that contains the entries is returned.

1. **Testing**

JUnit is a unit testing framework for Java programming language. JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks collectively known as xUnit, that originated with JUnit.

Test can be found under test directory. We tested all major operations that can be performed in this project. We have two test classes : DatabaseTest and TableTest that contains unit tests for all the methods from DatabaseImpl and TableImpl. DatabaseTest contains 9 test and TableImpl 22. Before each testing method is executed the method with annotation @Before is invoked. In this method are created the files that are necessary for the execution of tests.

Under gui package are tests related to the graphic interface. We have tests for: DeleteDatababsePanel, CreateDatabasePanel, DeleteTablePanel, CreateTablePanel and AlterTablePanel(ChangeTableNamePanel, AddNewColumnPanel and DeleteColumnPanel). There is a sum of 48 methods for test regarding GUI. The most of the test methods verifies what’s happen if we select an specific radio button, or an specific checkbox, or what’s happen if we have some combo-boxes that are not selected or selected. Also, we have tests for user input, if the input is processed and verified or not and if message/success errors are displayed to user.

1. **Assertions**

Regarding user graphic interface, we have assertions for those cases when an user is not selecting a table and/or database from combo-box, but it’s compulsory to select.

Most asserts from the project are preconditions. We validate the input provided by checking if the names of the tables or databases are not null or empty, provided arrays contains elements or types for columns or files are supported.

Post-conditions are used to check if resources were created with success after the execution of methods.

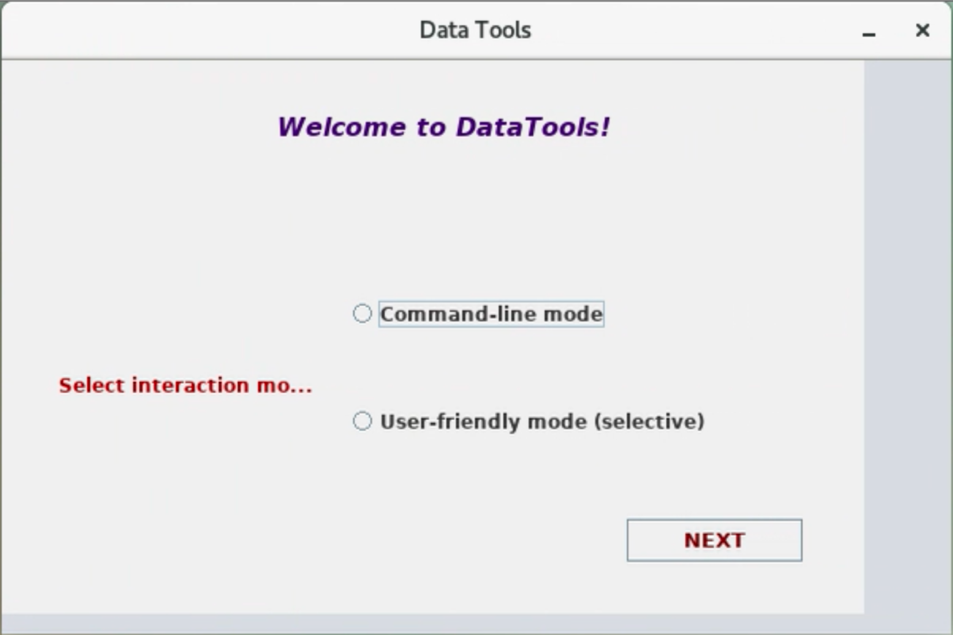
1. **User Manual**
   1. **Command Line**

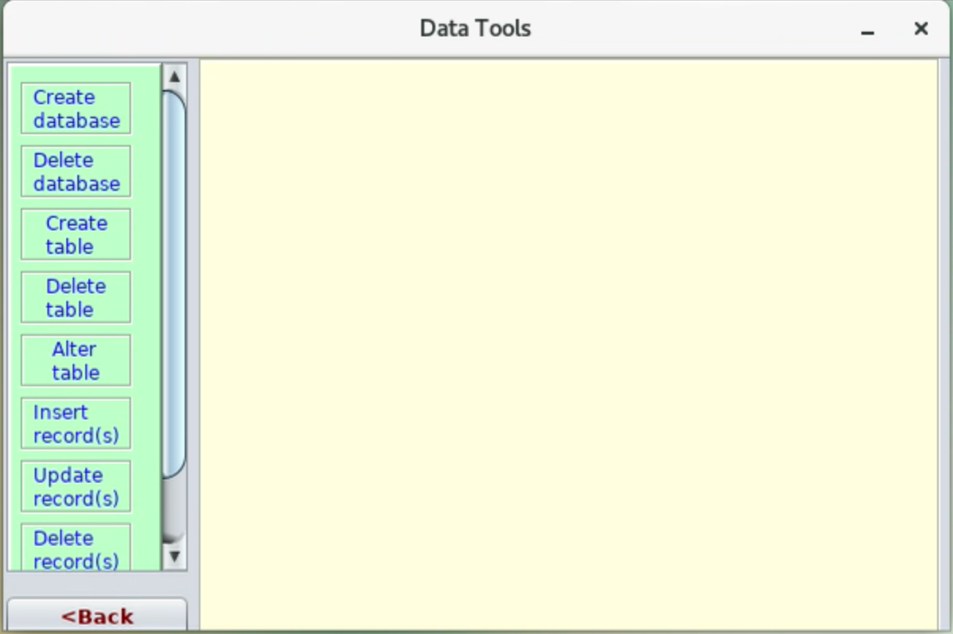
In this chapter will present all operations that can be performed from the command line and the required syntax for each one of them.

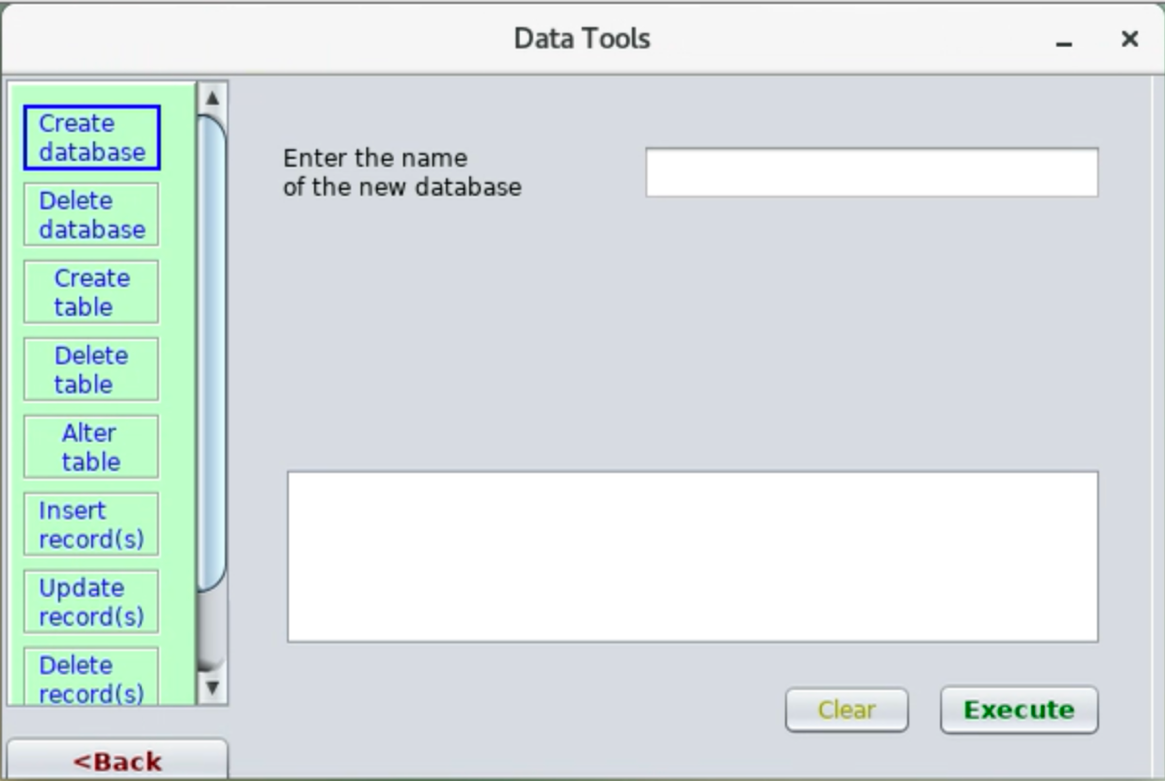
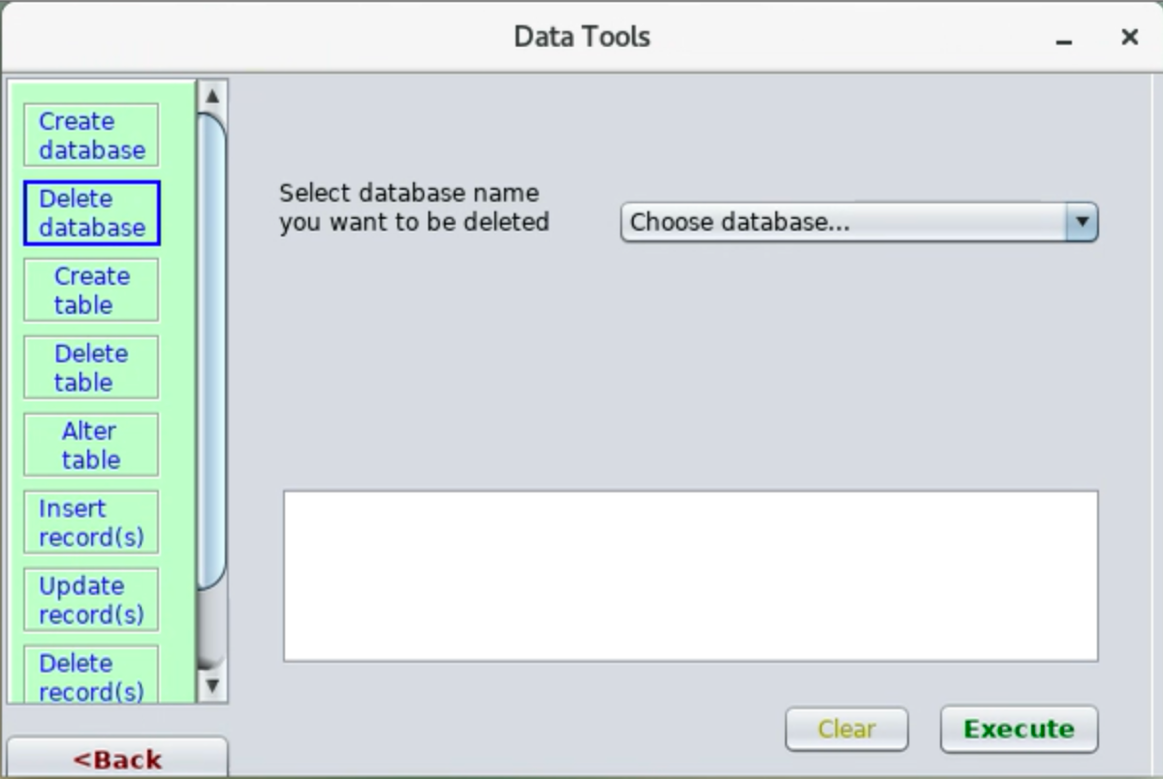
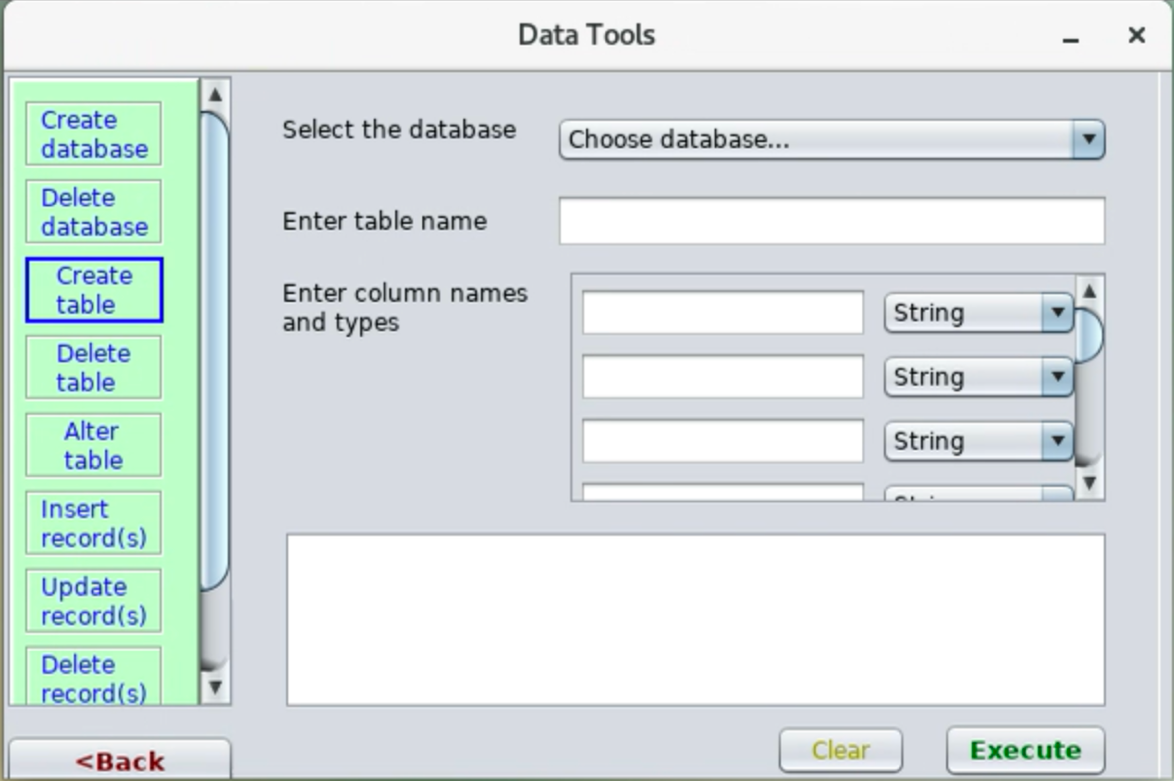
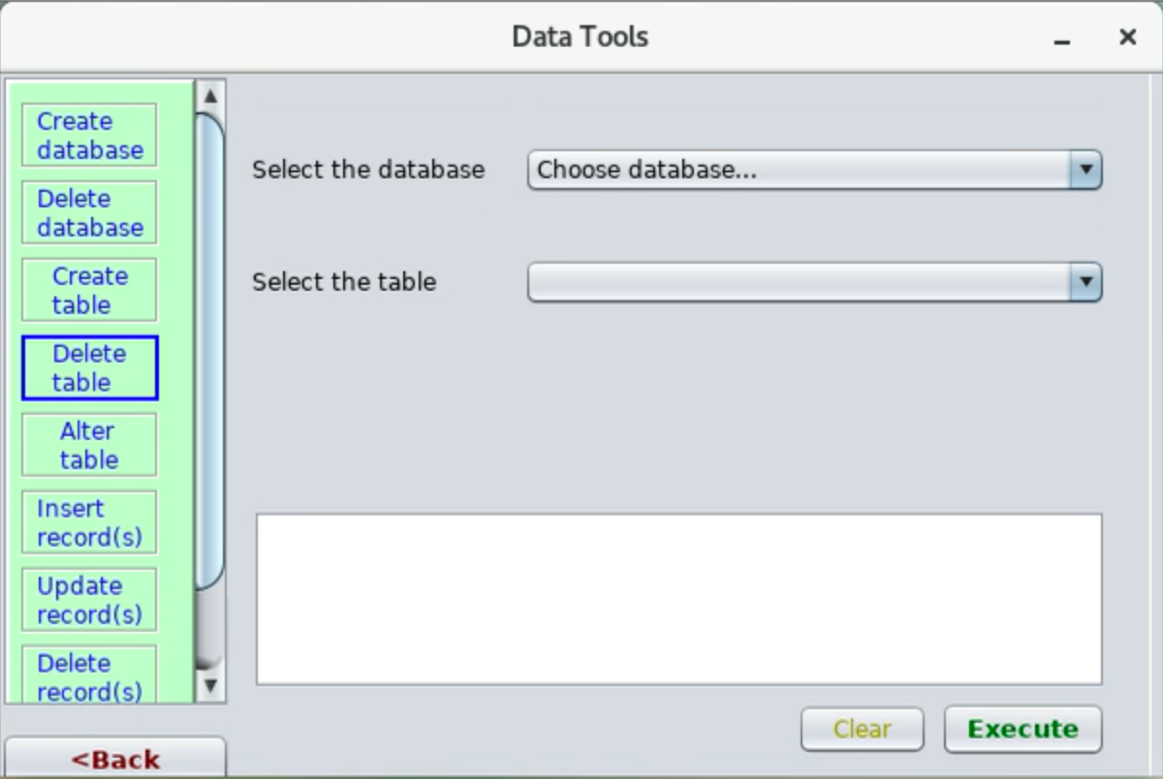
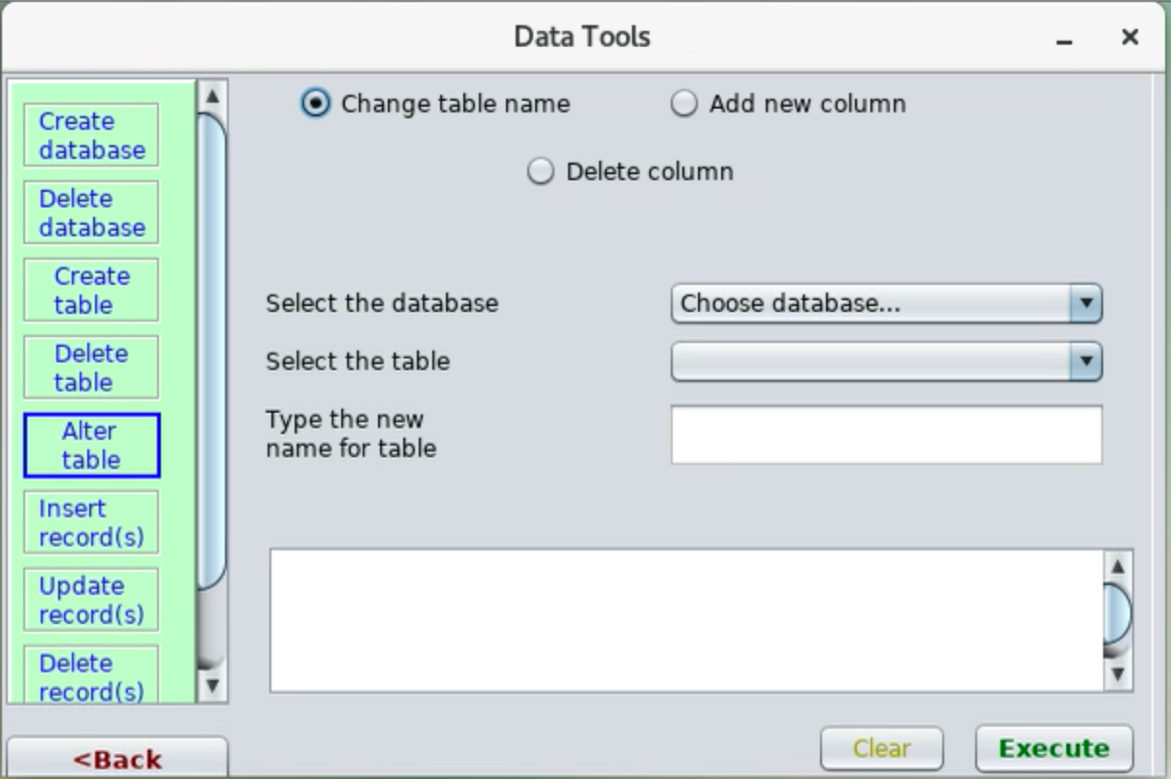
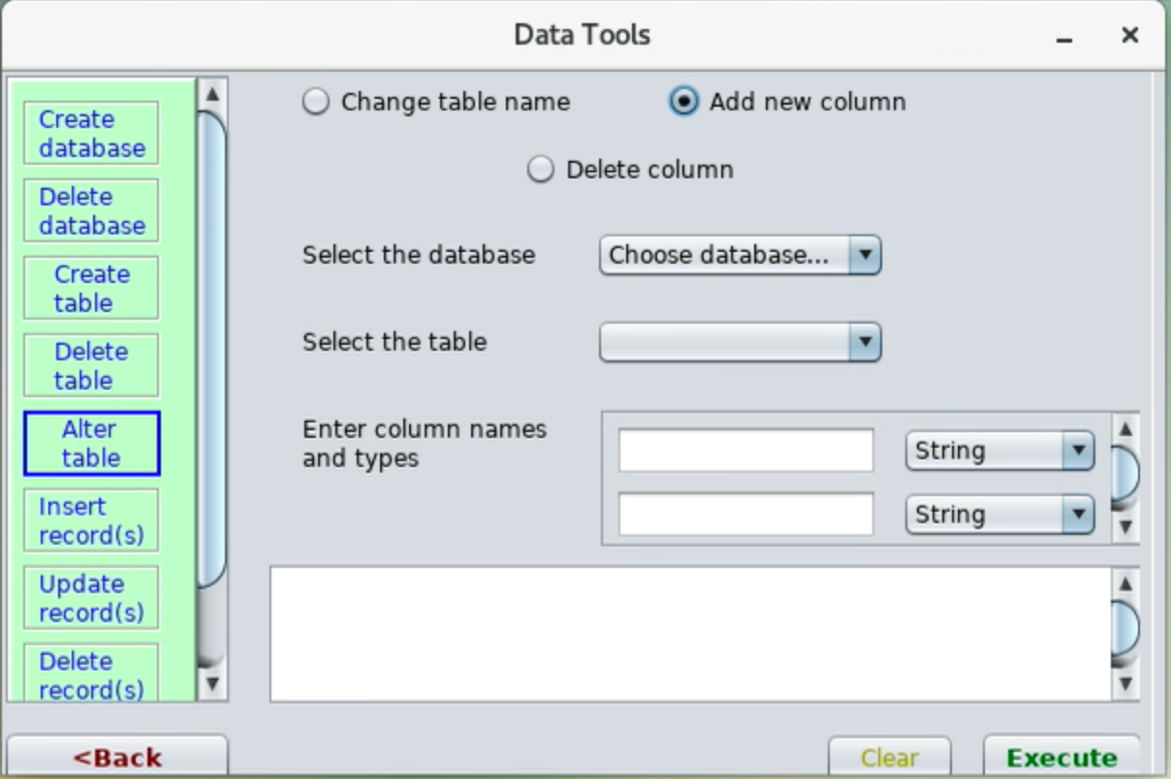
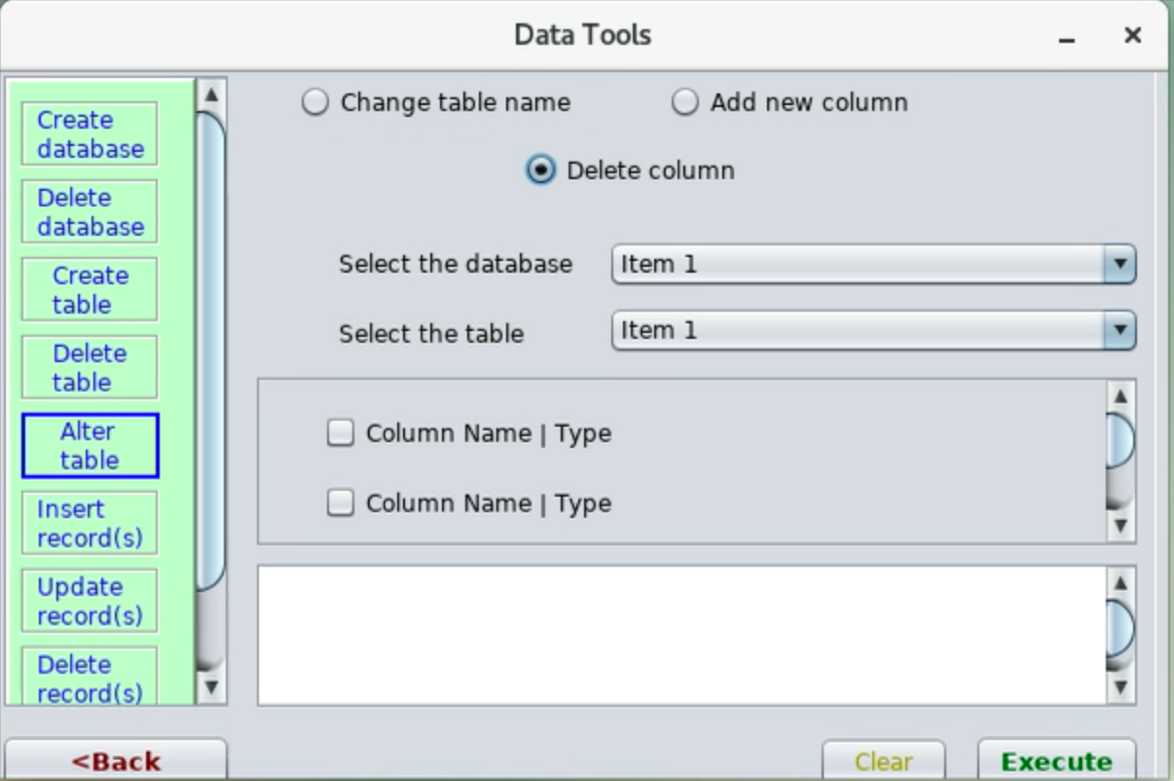
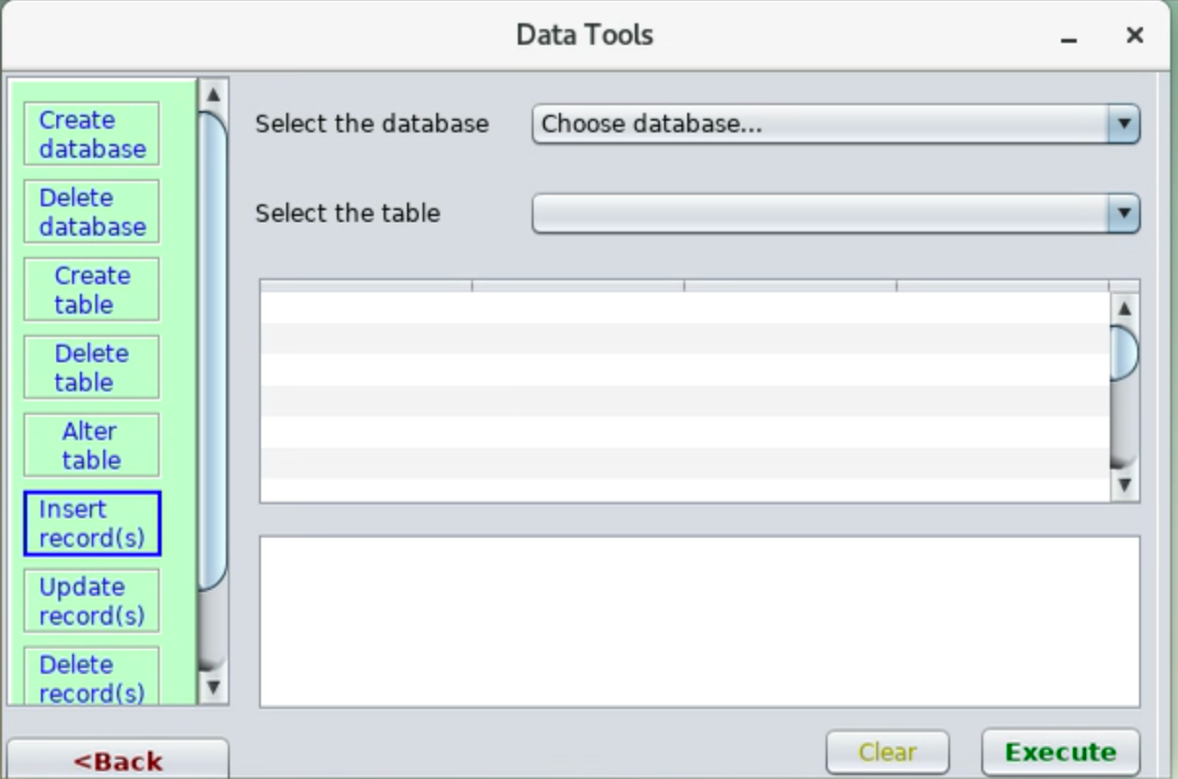
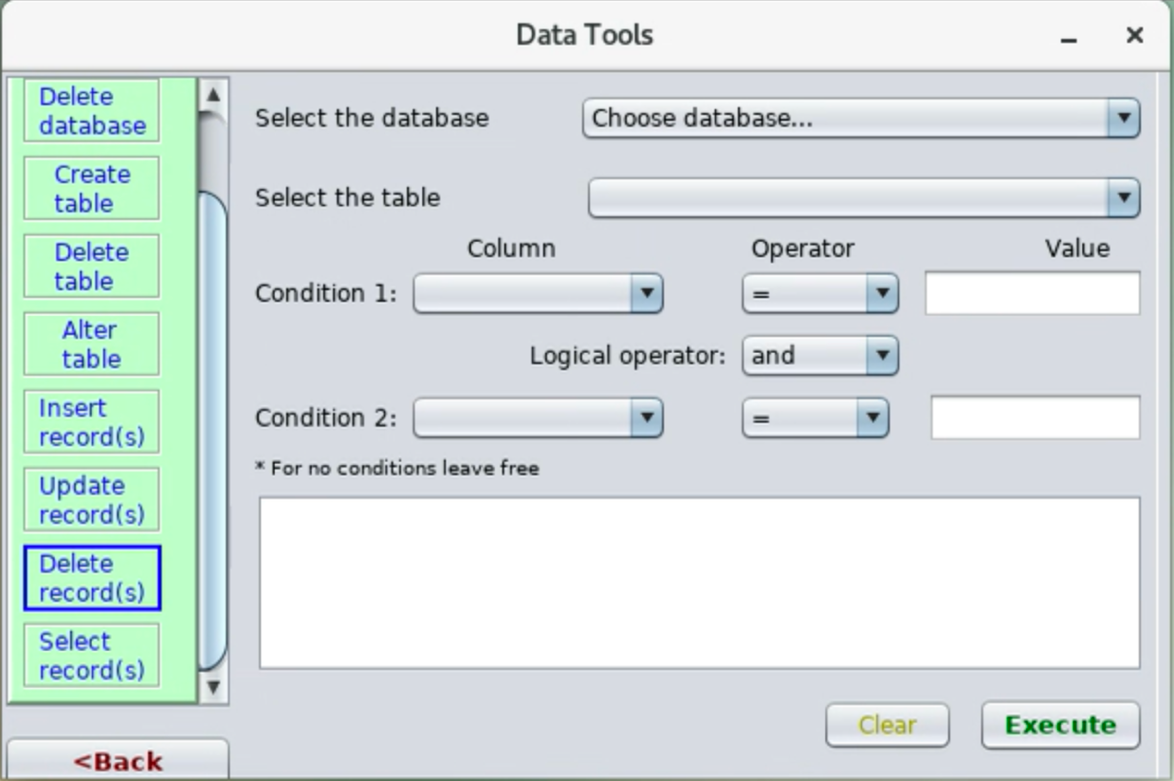
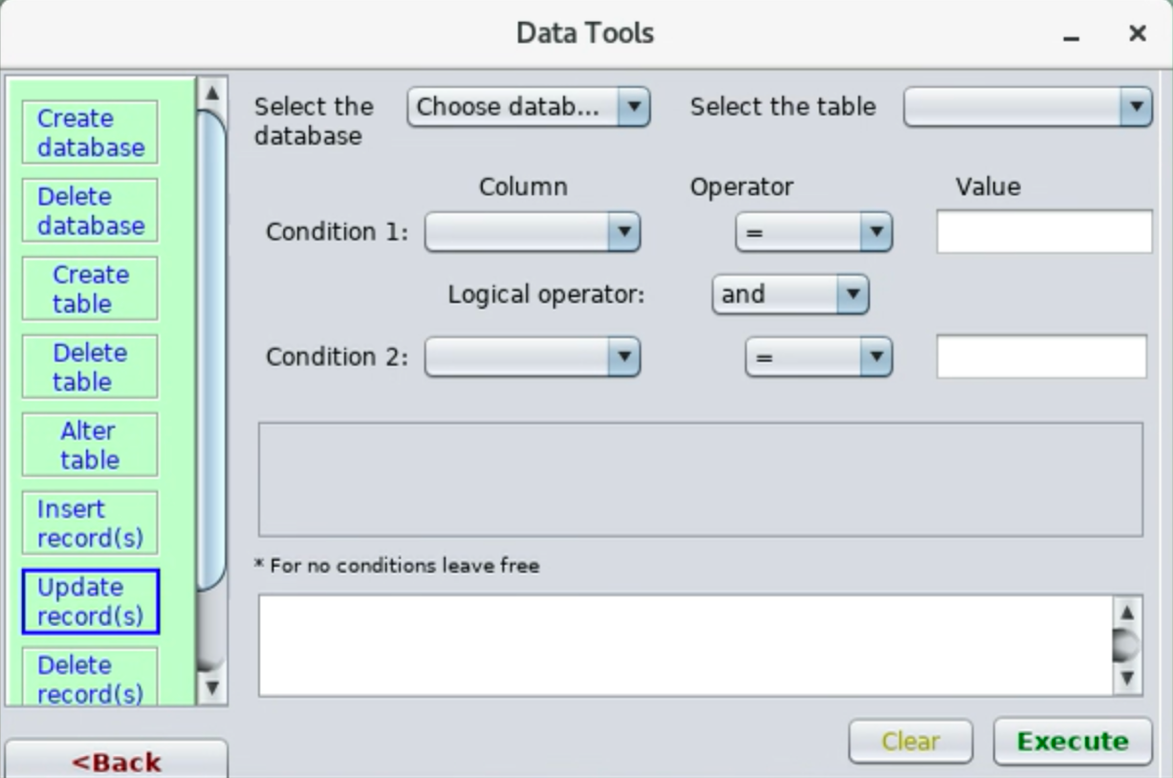
* Display databases – “show databases” a list of databases will be displayed to the console
* Display tables from a database – “show tables from [database name]” all tables from the specified database will be displayed
* Create a database – “create database [database name]” new database with the provided name will be created
* Delete a database – “delete database [database name]” if the database exists will be deleted with all the tables from it
* Select a database – “change database to [database name]” a database is selected. We need to select a database before the following operations can be executed.
* Display a table schema – “get table schema for [table name]” fields and types will be displayed
* Create a table – “create table [table name] ([column name] : [column type], …) create a new table with the columns provided between [parentheses](https://www.google.com/search?q=parentheses&spell=1&sa=X&ved=0ahUKEwiD54zz9JXiAhV3AxAIHR4FBp0QkeECCCooAA)
* Delete a table – “drop [table name]” delete the table with the provided name if exists
* Insert new records in a table – “insert into [table name] values [val1],[val2] … insert into the specified table values provided after “values” word. Values need to be separated by comma.
* Delete records from a table – “delete from [table name] where [where clause]”. Where clause can have multiple conditions separated by “or” or “and”. Example : firstName = [val1] or firstName = [val2] and lastName = [val3] – this is interpreted to ( firstName = [val1] or firstName = [val2]) and lastName = [val3]
* Update records – “update [table name] set [field name] = [val] …. where [where clause] updates specified fields to the new values provided when the where clause is met. Where clause is optional.
* Select records – “select [fields to be displayed] from [table name] where [where clause]” [fields to be displayed] – can be “\*” and in this case all columns are displayer, or can be a list of columns separated by coma. Where clause is optional.
* Download a table – “download [table name] as [type] to [path]” type can be xml or json. A new file is saved to the provided path
* Add new columns to a table – “alter [table name] add [columns]” [columns] must be provided in the same way as at table creation
* Remove columns from a table – “alter [table name] drop column [column]” column needs to be a valid column from the table.
  1. **Graphic interface**

In this chapter will present all operations that can be performed from the graphic user interface.

When the user interacts for the first time with the application, (s)he will be asked the desired mode of interaction: command-line mode (not implemented yet) or user-friendly mode. If (s)he’s choosing user-friendly mode, the next operations are available:

****

****

* ****Create database: the user has to type the new database name into a text field. If the database already exists, an error message will be displayed, other a message of success will be displayed in the below text area.
* ****Delete database: the user has a combo-box where are listed all existent databases. From there a database has to be selected in order to be deleted.
* ****Create table: first of all, the user has to select a database where the table has to be created, after that has to type the new table name and to complete at least one column name with a type (String, Integer, Boolean, Double). If the table name already exists in the selected database, an error message will be displayed and if not, a success message will be showed.
* Delete table: the user has to select for the first time the database name and after that the table name, which (s)he wants deleted.
* ****Alter table – Change table name: first of all the user has to select a database and table name and after that, has to type the new table name. If the table name already exists in the database, an error will be thrown.
* ****Alter table – Add new column: first of all the user has to select a database and table name and after that the user has to type the new column names along with their types.
* ****Alter table – Delete column: (s)he has to select a database and table and after that all table columns will be displayed. From here, the user has to check the desired columns to be deleted.
* ****Insert records: the user has to select the database and table name where (s)he wants to insert. After that a table with table’s columns will be shown and the user can insert new records in the table.
* ****Delete records:the user has to select the database and table name from where (s)he wants to delete. He can create at most two conditions for deleting records. If conditions are empty, all the record from the specified table will be deleted.
* ****Update records: the user has to select the database and table name where (s)he wants to update. (S)He can create at most two conditions for updating records. If conditions are empty, all the record from the specified table will be shown to be updated. After selecting all of this, a table with all records will be shown, and the user can update whatever (s)he wants.
* Select records:the user has to select the database and table name from where (s)he wants to select. (S)He can create at most two conditions for selecting records. If conditions are empty, all the record from the specified table will be shown. After selecting all of this, the user has to select the columns selected and a table with all records will be shown.

****

For executing each command, in graphic interface, the user has an “Execute” button. The error/success message or results after processing data, will be shown in a text area.

For clearing all selections, the user input and the output, the user has an “Clear” button.

For returning in the main menu, the user has an “Back” button.

1. **Contribution**
   1. **Chilaboc Ecaterina-Mihaela**

* Phase 1: created graphic user interface and integrated Vlad’s “back-end” with all panels, combo-boxes, checks, text-fields, buttons and text areas.
* Phase 2: created unit tests for graphic interface’s panels: delete database, create database, create table, delete table, alter table (change table name, add new column, delete column).
* Phase 3: used assertions in graphic interface’s code, for throwing exceptions when a database or table name is not selected.
  1. **Ventaniuc Vladimir**
* Phase 1: created basic operations for the database (inserting, deleting, alter table) and the command line interface.
* Phase 2: created unit tests for all operations from the phase 1.
* Phase 3: Preconditions and post-conditions were added to each method described in Chapter 2.