**Universitatea Alexandru Ioan Cuza Iași**

**FACULTATEA DE INFORMATICĂ**

**Database Implementation**

**Authors:**

**Ecaterina Mihaela Chilaboc**

**Ventaniuc Vladimir**

**Content**

1. **Problem Definition**
2. **Design and Programming**
3. **Testing**
4. **User Manual**
5. **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.

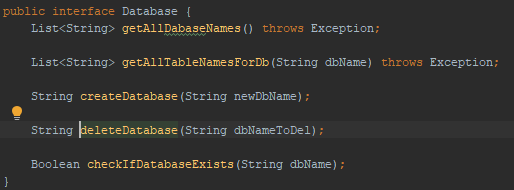
Data is organized into rows, columns and tables, and it is indexed to make it easier to find relevant information. Data gets updated, expanded and deleted as new information is added. Databases process workloads to create and update themselves, querying the data they contain and running applications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

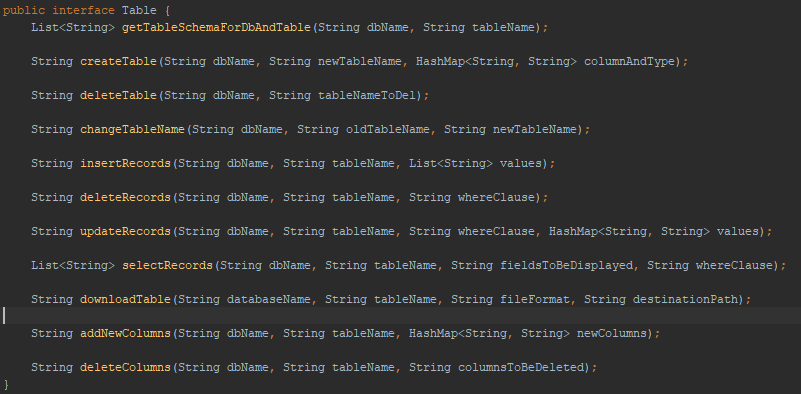
* getAllDatabaseNames – returns a list of strings that contains the name of each directory created under the resource directory.
* getAllTableNamesForDB - recursively browses the directory that is given as a parameter and returns the name of all tables stored in it. A table is stored as an xml file.
* 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. Under gui directory are tests related to the

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 which 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**