Проект 6: Бази от данни

*Изготвил:* Георги Станчев, I курс, ИС, Фак. № 71959

[Глава 1. Увод 3](#_Toc41892490)

[1.1. Описание и идея на проекта 3](#_Toc41892491)

[1.2. Цел и задачи на разработка 4](#_Toc41892492)

[1.3. Структура на документацията 5](#_Toc41892493)

[Глава 2. Преглед на предметната област 5](#_Toc41892494)

[2.1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани 5](#_Toc41892496)

[2.2. Дефиниране на проблеми и сложност на поставената задача 5](#_Toc41892497)

[2.3. Подходи, методи (евентуално модели и стандарти) за решаване на поставените проблемите 6](#_Toc41892498)

[2.4. Потребителски (функционални) изисквания (права, роли, статуси, диаграми, ...) и качествени (нефункционални) изисквания (скалируемост, поддръжка, ...) 6](#_Toc41892499)

[Глава 3. Проектиране 7](#_Toc41892501)

[3.1. Обща архитектура – ООП дизайн 7](#_Toc41892503)

[Глава 4. Реализация, тестване 8](#_Toc41892504)

[4.1. Реализация на класове (включва важни моменти от реализацията на класовете и малки фрагменти от кода) 8](#_Toc41892506)

[4.2. Управление на паметта и алгоритми. Оптимизации 9](#_Toc41892507)

[4.3. Планиране, описание и създаване на тестови сценарии (създаване на примери) 9](#_Toc41892508)

[Глава 5. Заключение 17](#_Toc41892509)

[5.1. Обобщение на изпълнението на началните цели 17](#_Toc41892511)

[5.2. Насоки за бъдещо развитие и усъвършенстване 17](#_Toc41892512)

[Глава 6. Използвана литература 17](#_Toc41892513)

# Увод

* 1. Описание и идея на проекта

Да се реализира програма на C++, поддържаща операции с прости бази от данни. Базите данни се състоят от серии от таблици, като всяка таблица е записана в собствен файл. Базата данни е записана в главен файл (каталог), който съдържа списък от таблиците в базата данни, като за всяка таблица е зададено име и файл, в който таблицата е записана.

Всяка “колона” на таблица в базата данни има тип, като в една таблица може да има едновременно колони от различни типове. Приложението трябва да може да поддържа следните типове:

**Цяло число** – поредица от цифри, без никакви други символи между тях. В началото на числото може да има знак '+' или '-'. Например:

123

-123

+123

**Дробно число** – поредица от цифри, следвана от символ за точка и след нея друга поредица от цифри. В началото на числото може да има знак '+' или '-'. Например:

123.456

-123.456

+123.456

**Символен низ (стринг)** – поредица от произволни символи оградени в кавички. Подобно на низовете в C++, при необходимост да се включи символа за кавичка в даден низ, трябва да се представи като \", за наклонена черта, трябва да се представи като \\. Например:

"Hello world!"

"C:\\temp\\"

"\"This is a quotation\""

Освен конкретна стойност, дадена клетка в даден ред на таблицата може да е “празна”. Такива клетки да се обозначават специално и да се изписват като “NULL”.

* 1. Цел и задачи на разработка

Целта на приложението, е да може да оперира с множество файлове и да може да извършва общи операции (close, save, save as, help и exit). В допълнение е необходимо да извършва и посочените операции в Таблица 1:

|  |  |
| --- | --- |
| import <file name> | Добавя в базата данни нова таблица от файл. Във файла е записана информация за типа на всяка колона. **Всяка таблица има име**. При опит за зареждане на таблица с име, което съвпада с името на някоя вече заредена таблица, системата да дава грешка. Добавената таблица се записва в каталога на базата от данни. |
| showtables | Показва списък с имената на всички заредени таблици |
| describe <name> | Показва информация за типовете на колоните на дадена таблица |
| print <name> | Показва всички редове от дадена таблица. Да се реализира диалогов режим, позволяващ съдържанието на таблицата да се преглежда по страници (такива, че се събират на един екран) със следните команди: следваща страница, предишна страница, изход. |
| export <name> <file name> | Записва таблица във файл |
| select <column-n> <value> <table name> | Извежда всички редове от таблицата, които съдържат стойността “value” в клетката с дадения пореден номер. Да се реализира извеждане по страници |
| addcolumn <table name> <column name> <column type> | Добавя нова колона (с най-голям номер) в дадена таблица. За всички съществуващи редове от таблицата, стойността на тази колона да е празна. |
| update <table name> <search column n> <search value> <target column n> <target value> | За всички редове в таблицата, чиято колона с пореден номер <search column n> съдържа стойността <search column value> се променят така, че колоната им с пореден номер <target column n> да получи стойност <target value>. Да се поддържа стойност NULL. |
| delete <table name> <search column n> <search value> | Изтрива всички редове в таблицата, чиято колона <search column n> съдържа стойността <search column value> |
| insert <table name> <column 1> … <column n> | Вмъква нов ред в таблицата със съответните стойности |
| innerjoin <table 1> <column n1> <table 2> <column n2> | Извършва операцията [Inner Join](https://en.wikipedia.org/wiki/Join_(SQL)#Inner_join) над две таблици спрямо колоните <column n1> в първата таблица и <column n2> във втората. Създава нова таблица и извежда идентификатора и. |
| rename <old name> <new name> | Преименува таблица. Отпечатва грешка, ако новото име не е уникално. |
| count <table name> <search column n> <search value> | Намира броя на редовете в таблицата, чиито колони съдържат дадената стойност |
| aggregate <table name> <search column n> <search value> <target column n> <operation> | Извършва дадена операция върху стойностите от колоната <target column n> на всички редове, чиито колони с номер <search column n> съдържат стойността <search value>. Възможните операции са sum, product, maximum, minimum. Системата да дава грешка, ако колоните не са числови. |

* 1. Структура на документацията

Документа е структуриран в 5/пет/ основни глави, описващи целта и идеята на проекта, както и неговото реализиране.

# Преглед на предметната област

1. 1. Основни дефиниции, концепции и алгоритми, които ще бъдат използвани

Програмата е реализирана на Visual Studio 2019 , C++. Основната идея, по която е разработена, e база данни позволяваща извършването на основните операции с таблиците, от които е съставена. Като цяло приложението представлява работа с четене и запис на данни във файлове файлове, със съответните валидации и проверки, които са естествени за правилното функциониране. Основният файл - Каталог, съставен от имената на таблиците и файловете, които ги представляват, е съществена част от приложението и затова е изграден механизъм за автоматичното му създаване ако липсва такъв. Файловете, в които са записани таблиците, са организирани така, че всеки ред от файла представлява отделна колона.

* 1. Дефиниране на проблеми и сложност на поставената задача

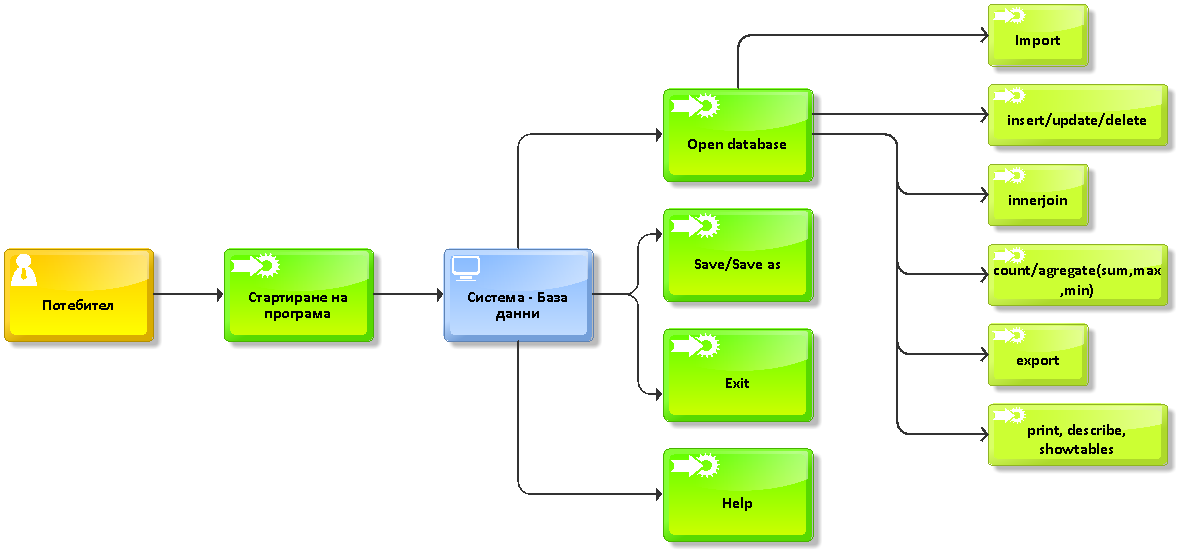
В самото начало постигането на целите на задачата изглеждаше сложно, но в последствие на писане на код и стъпка по стъпка, всичко започна да се подрежда логически. Трудността се състоеше в това, че беше необходимо да се обмисли внимателно съдържанието на файла, което да отразява структура на таблица от база данни и съответно да позволява всички зададени операции по редове и колони – insert, update, delete, join, export, import. От друга страна трябва да се съобрази и реализирането на тяхното зареждане в паметта, за да се постигне максимално удобство за бъдещата обработка.

* 1. Подходи, методи (евентуално модели и стандарти) за решаване на поставените проблемите

След задълбочен анализ върху поставената задача, подходи и методи за реализираното и постигане на крайните цели, избрах да работя основно със символни низове, като при необходимост ги превръщам в числов тип. Проучих и се запознах обстойно как се представят данните и как се оперира с тях в някои различни SQL бази данни, което ми помогна да изградя собствен модел, отговарящ на условията на проекта. Създадох клас, представляващ двойката име на таблица и пътя на файла, където се намира, за да може чрез него да се състави и каталога. След това започнах да съставям нужните функционалности за работа с файловете, а накрая и нужните операции, необходими за работата с таблиците в базата данни с недостатъка, че единствено операцията innerjoin не успях да реализирам. Проверки са правени за идентичност на имената на колоните и на таблиците, както и на имената на файловете, като ако бъде допусната грешка се предоставя ясно съобщение на потребителя.

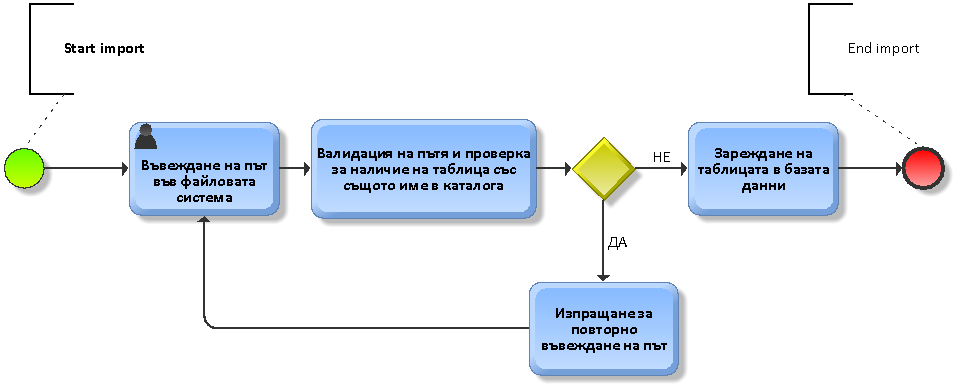
* 1. Потребителски (функционални) изисквания (права, роли, статуси, диаграми, ...) и качествени (нефункционални) изисквания (скалируемост, поддръжка, ...)

Потребителските (функционални) изисквания към приложението (*Фигура* *1*) се дефинират и детайлизират с цел постигане на високо ниво на разбиране за потребностите

на потребителя от функционалността на приложението и дават ясна представа за отделните компоненти, които да се разработят по подходящ метод и алгоритъм.

Фигура

На следващата схема е показана детайлно функционалността на операция „import” (*Фигура* *2*)



Фигура 2

# Проектиране

1. 1. Обща архитектура – ООП дизайн

С помощта на клас, който имплементира двойката име на таблица и пътя на файла, където е записана тази таблица, се създава каталога на базата данни, който всъщност е сред важните компоненти, за които трябва да сa предвидени всякакви случаи. Поради тази причина, ако каталога липсва, той ще бъде създаден автоматично. (*Фрагмент 1*)



Фрагмент

Създадена е функция, която играе роля на разпределител на командите, чрез която програмата вече изпълнява желанието на потребителя в зависимост от неговия вход, като са предвидени и възможно най-много случаи не грешки, при които да може да се изведе нужното съобщение, за да може потребителя да направи корекция. (*Фрагмент 2*)



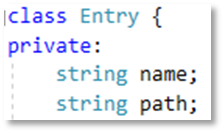
Фрагмент

Важно е да се отбележи, че за разделяне на отделните компоненти на входа от потребителя и на данните във файловете са различни. (*Фрагмент 3*)

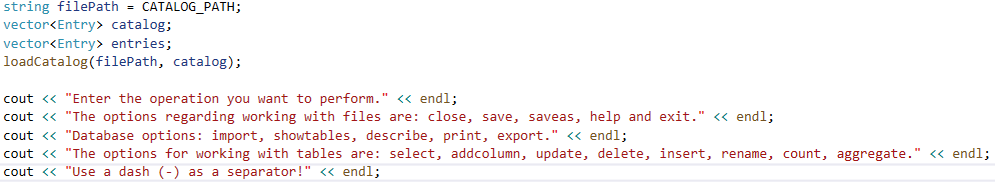
Фрагмент 3

# Реализация, тестване

1. 1. Реализация на класове (включва важни моменти от реализацията на класовете и малки фрагменти от кода)

Представеният фрагмент 4, показва малка част от класа Вписване, който е имплементация на двойката име на таблица и пътя на файла, където е записана тя, като чрез тези записи се съставя каталога на базата данни.

Фрагмент 4

Още в самото начало, ако не е бил създаден от потребителя, автоматично се прави текстов файл и в него вече може да се правят записи на таблици. Като за да се извършват операции с нововъведените таблици в каталога, те първо трябва да се запазят в него, в противен случай стоят заредени в паметта във вектора “entries” и при опит за работа с тях, ще се изведе напомнящо съобщение за запазването им в каталога. (*Фрагмент 5*)

Фрагмент 5

* 1. Управление на паметта и алгоритми. Оптимизации

Изградено е управление на паметта, но предполагам за наличие и на по-оптимално решение по отношение на паметта, която използвам за реализирането на правилната работа на операциите с таблиците, но спрямо целите на заданието, мисля, че е в рамките на допустимото.

* 1. Планиране, описание и създаване на тестови сценарии (създаване на примери)

За постигане на успешното внедряване на системата е необходимо много добре да се планират стъпките и дейностите в етапите на тестване и внедряване. Крайният сорс код ще се окомплектова заедно с помощните материали и публикува в GitHub, като същевременно е спазвано и качването на междинните разработки.

За преглед - [*https://github.com/FantomGSS/Database*](https://github.com/FantomGSS/Database)

С цел проверка на постигнатата реализация на приложението с необходимите критерии по задание, е необходимо да се изпълняват минимум следните функционални тестови сценарии (Таблица 2). Както споменах по-рано, единствено операцията между две таблици – innerjoin не успях да реализирам, но за останалите може да се твърди, че работят добре. Направени са необходимите тестове за проверка, както и допълнителни с повече данни във файловете, представляващи таблиците, с цел да се провери дали всичко работи според очакванията и впоследствие логиката на някои места се пренаписваше.

Таблица 2

|  |  |  |  |
| --- | --- | --- | --- |
| **номер** | **Функция** | **Описание** | **Резултат (Да/Не)** |
| 1 | import <file name> | Добавя в базата данни нова таблица от файл. Във файла е записана информация за типа на всяка колона. Всяка таблица има име. При опит за зареждане на таблица с име, което съвпада с името на някоя вече заредена таблица, системата да дава грешка. Добавената таблица се записва в каталога на базата от данни. | да |
| 2 | showtables | Показва списък с имената на всички заредени таблици | да |
| 3 | describe <name> | Показва информация за типовете на колоните на дадена таблица | да |
| 4 | print <name> | Показва всички редове от дадена таблица. Да се реализира диалогов режим, позволяващ съдържанието на таблицата да се преглежда по страници (такива, че се събират на един екран) със следните команди: следваща страница, предишна страница, изход. | да |
| 5 | export <name> <file name> | Записва таблица във файл | да |
| 6 | select <column-n> <value> <table name> | Извежда всички редове от таблицата, които съдържат стойността “value” в клетката с дадения пореден номер. Да се реализира извеждане по страници | да |
| 7 | addcolumn <table name> <column name> <column type> | Добавя нова колона (с най-голям номер) в дадена таблица. За всички съществуващи редове от таблицата, стойността на тази колона да е празна. | да |
| 8 | update <table name> <search column n> <search value> <target column n> <target value> | За всички редове в таблицата, чиято колона с пореден номер <search column n> съдържа стойността <search column value> се променят така, че колоната им с пореден номер <target column n> да получи стойност <target value>. Да се поддържа стойност NULL. | да |
| 9 | delete <table name> <search column n> <search value> | Изтрива всички редове в таблицата, чиято колона <search column n> съдържа стойността <search column value> | да |
| 10 | insert <table name> <column 1> … <column n> | Вмъква нов ред в таблицата със съответните стойности | да |
| 11 | innerjoin <table 1> <column n1> <table 2> <column n2> | Извършва операцията Inner Join над две таблици спрямо колоните <column n1> в първата таблица и <column n2> във втората. Създава нова таблица и извежда идентификатора и. | не |
| 12 | rename <old name> <new name> | Преименува таблица. Отпечатва грешка, ако новото име не е уникално. | да |
| 13 | count <table name> <search column n> <search value> | Намира броя на редовете в таблицата, чиито колони съдържат дадената стойност | да |
| 14 | aggregate <table name> <search column n> <search value> <target column n> <operation> | Извършва дадена операция върху стойностите от колоната <target column n> на всички редове, чиито колони с номер <search column n> съдържат стойността <search value>. Възможните операции са sum, product, maximum, minimum. Системата да дава грешка, ако колоните не са числови. | да |

По-долу е представено в няколко примера работата на приложението, покривайки основните функционалности, които се изискваха от задачата. Входа от потребителя е означен с долар ($) в примерите.

Пример 1:

Enter the operation you want to perform.

The options regarding working with files are: close, save, saveas, help and exit.

Database options: import, showtables, describe, print, export.

The options for working with tables are: select, addcolumn, update, delete, insert, rename, count, aggregate.

Use a semicolon (;) as a separator!

Command: $ showtables

Clients, Addresses

Command: $ import;credit.db

The table has been successfully added to the catalog!

Command: $ showtables

Clients, Addresses

Command: $ print;Credit

To work with this table, you must save the changes in the catalog!

Command: $ save

===========================================================

All changes are saved successfully in the file Catalog.txt!

===========================================================

Command: $ print Credit

This command is not recognized!

Command: $ print;Credit

The printing of a table is limited to three lines

The possible viewing options are: next page, previous page and exit.

id\_client credit

int double

1 123.45

2 NULL

3 345.55

Operation: $ previous page

There is no previous page yet, because you are still at the beginning of the review!

Operation: $ next page

4 NULL

Command: $ describe;Addresses

int, string, string

Command: export;Addresses;fileExport.db

===========================================================

Successfully created fileExport.db!

===========================================================

Command: $ close

The catalog has been successfully closed!

Command: $ exit

Пример 2:

Enter the operation you want to perform.

The options regarding working with files are: close, save, saveas, help and exit.

Database options: import, showtables, describe, print, export.

The options for working with tables are: select, addcolumn, update, delete, insert, rename, count, aggregate.

Use a semicolon (;) as a separator!

Command: $ showtables

Clients, Addresses, Credit

Command: $ select;2;"Sofia";Addresses

The printing of the rows from your selection is limited to three lines

The possible viewing options are: next page, previous page and exit.

id\_client city address

int string string

1 "Sofia" "Mladost 1"

3 "Sofia" "Banishora"

Command: $ addcolumn;Clients;age;int

The column has been successfully added to the table!

Command: $ print;Clients

The printing of a table is limited to three lines

The possible viewing options are: next page, previous page and exit.

id name marital\_status age

int string string int

1 "Ivan Ivanov" "yes" NULL

2 "Denitsa Dimitrova" "yes" NULL

3 "Krassimir Lefterov" NULL NULL

Operation: $ exit

Command: insert;Clients;5;"Svetlin Dimitrov";"no";52

The row has been successfully added to the table!

Command: $ print;Clients

The printing of a table is limited to three lines

The possible viewing options are: next page, previous page and exit.

id name marital\_status age

int string string int

1 "Ivan Ivanov" "yes" NULL

2 "Denitsa Dimitrova" "yes" NULL

3 "Krassimir Lefterov" NULL NULL

Operation: $ next page

4 "Rumena Marinova" "no" NULL

5 "Svetlin Dimitrov" "no" 52

Command: $ rename;Addresses;Place\_of\_residence

The table was successfully renamed!

Command: $ showtables

Clients, Place\_of\_residence, Credit

Command: $ exit

Пример 3:

Enter the operation you want to perform.

The options regarding working with files are: close, save, saveas, help and exit.

Database options: import, showtables, describe, print, export.

The options for working with tables are: select, addcolumn, update, delete, insert, rename, count, aggregate.

Use a semicolon (;) as a separator!

Command: $ showtables

Clients, Place\_of\_residence, Credit

Command: $ print;Credit

The printing of a table is limited to three lines

The possible viewing options are: next page, previous page and exit.

id\_client credit

int double

1 123.45

2 NULL

3 345.55

Operation: $ next page

4 NULL

Command: $ update;Credit;2;NULL;2;25.48

The table was successfully updated!

Command: $ print;Credit

The printing of a table is limited to three lines

The possible viewing options are: next page, previous page and exit.

id\_client credit

int double

1 123.45

2 25.48

3 345.55

Operation: $ next page

4 25.48

Command: $ delete;Credit;2;25.48

The rows have been successfully deleted from the table!

Command: $ print;Credit

The printing of a table is limited to three lines

The possible viewing options are: next page, previous page and exit.

id\_client credit

int double

1 123.45

3 345.55

Command: $ count;Place\_of\_residence;2;"Varna"

The number of rows where this value is contained in the specified column is 1.

Command: $ aggregate;Credit;1;1;2;sum

The result is: 123.450

Command: $ exit

# Заключение

1. 1. Обобщение на изпълнението на началните цели

На пръв поглед, задачата изглеждаше трудна, но впоследствие след като направих анализ и си насъбрах идеи, всичко почна да изглежда с една идея по-лесно. Представеното решения може би не е сред най-добрите, като се има предвид, и че една от опциите липсва (innerjoin), но по отношение на нужните проверки, които са необходими за една база данни, всичко е предвидено.

* 1. Насоки за бъдещо развитие и усъвършенстване

Като възможно усъвършенстване на приложението в бъдеще, е оптимизацията на паметта и по-добър начин за съхранение на данните от файловете, представляващи таблиците.

# Използвана литература

*Използван алгоритъм за разделяне на думите чрез специфичен разделител –* [*https://www.fluentcpp.com/2017/04/21/how-to-split-a-string-in-c*](https://www.fluentcpp.com/2017/04/21/how-to-split-a-string-in-c/)

*Използван продукт за схеми по фиг.1 и 2 – ARIS Express* [*https://www.ariscommunity.com/aris-express*](https://www.ariscommunity.com/aris-express)