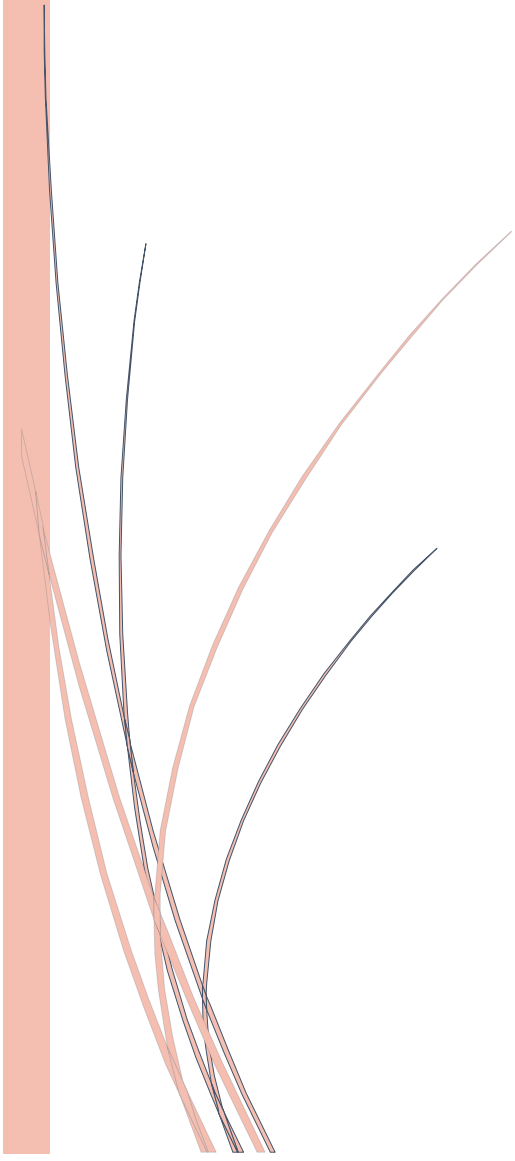
Софтуерна документация за проекта Минималистична Система за Управление на Бази Данни



Хюлиа Мустафа, КН, група 2
ФН: 81779

Съдържание

1.Увод

1.1 Описание и идея на проекта

Настоящият документ запознава читателите с функционалностите, начините за работа, перспективите и функционалните изисквания на проекта минималистична система за управление на бази данни(СУБД).

Проектът минималистична СУБД реализира създаването на система, изградена от таблици, които поддържат произволен брой типизирани колони с възможни типове за колоните: целочислен, число с плаваща запетая, символен низ с произволна дължина и булев.

1.2 Цел и задачи на разработката

Настоящият документ има за цел да запознае читателите с основните функции на проектът минималистична система за управление на бази данни, като е обърнато особено внимание върху реализацията на основните методи и класове. Показани са различни алгоритми за решаване на поставените от клиента задачи.

1.3 Оформление на документа

При написването на документа е използван шрифт Times New Roman. Текстът е с размер 12.Всички заглавни редове са удебелени и с размер 18. Подзаглавията са с размер 14.

2.Преглед на предметната област

2.1 Концепции и алгоритми

Проектът СУБД е реализиран чрез парадигмата за обектно-ориентираното програмиране, която представлява моделиране на предмети както от реалния свят, така и от научните сфери, чрез обекти, които да взаимодействат помежду си. Основни концепции, които се обхващат от ООП(обектно-ориентирано програмиране) са идеята за абстракция на данните, капсулиране(т.е. скриване на съществените данни за потребителя), полиморфизъм(обекти от един и същи тип да имат един и същи интерфейс, но различна реализация) и наследяване.

В проекта СУБД са включени всички основни концепции на ООП , чрез които се изгражда една надеждна и гъвкава система, която е по-проста за разработка и поддръжка.

2. 2 Функционални изисквания

Функционалните изисквания са пряко свързани с конкретната реализация на основните дейности.

Системата трябва да може да:

- ✓ Поддържа таблици с произволен брой типизирани колони

- ✓ Поддържа поле основен ключ
- ✓ Поддържа агрегатни функции

2.2.2 Използване на системата

А) Програма изпълнява диалогов режим за изпълнение на команди на базата от данни.

Б) След като потребителят е стартирал програмата, той трябва да въведе команда според определен синтаксис, която да извърши желанието от потребителя действия

Основни функционалности, които трябва да бъдат предоставени на отделните класове.

А) Функционалности на класа „Колона“.

- ✓ Добавя нова стойност към колоната
- ✓ Получава типа на колоната
- ✓ Проверява дали има дадена стойност на определен ред
- ✓ Променя дадена стойност на определен ред
- ✓ Принтира стойност на определен ред
- ✓ Връща броя на елементите в дадена колона
- ✓ Принтира името на колоната
- ✓ Намира минималната или максималната стойност в дадена колона
- ✓ Намира средно аритметично да елементите в дадена колона
- ✓ Намира сумата на елементите в дадена колона

Класът колона е абстрактен клас и тези функционалности са реализирани в неговите наследници: колона от целочислен тип, колона от булев тип, колона от тип символни низове и колона от тип числа с плаваща запетая.

А) Функционалности на класа „Таблица“.

- ✓ Добавя празна колона към таблицата
- ✓ Добавя динамично заделена колона към таблицата
- ✓ Добавя име на таблица
- ✓ Вмъква празен ред
- ✓ Принтира данните в дадена колона
- ✓ Принтира името на таблицата
- ✓ Принтира съдържанието на таблицата

2.3 Нефункционални изисквания

Изисквания за качеството на софтуера

2.3.1 Адаптивност

Системата трябва да бъде написана по начин, отговарящ на общоприетите добри практики, които да позволят лесна и бърза поддръжка и промяна на интерфейса до 48 часа.

Промените в системата не трябва да водят до загуба на данни

2.3.2 Преизползваемост

Системата трябва да бъде разработена по такъв начин, че да позволява максималното преизползване на код за други системи от този тип.

3.Проектиране

3.1 Обща архитектура- ООП дизайн

Проектът е изграден на базата на основни концепции в обектно-ориентираното програмиране, като за неговата реализация са използвани няколко основни класа „Колона(Column)“, „Колона, поддържаща целочислени типове(IntColumn)“, „Колона поддържаща символни низове(StringColumn)“, „Колона, поддържаща булеви типове(BoolColumn)“ и „Колона, поддържаща числа с плаваща запетая(DoubleColumn)“. Като класът „Колона“ е абстрактен клас, който има за наследници изброените по-горе 4 класа за различните типове колони. Също така, реализиран е и клас „Таблица“ , който поддържа набор от типизирани колони. Проектът съдържа клас „Система за бази данни“(DatabaseSystem), който съдържа списък от таблици.

Всяка една колона се определя от няколко основни фактора:

- ✓ име
- ✓ набор от елементи от един и същи тип

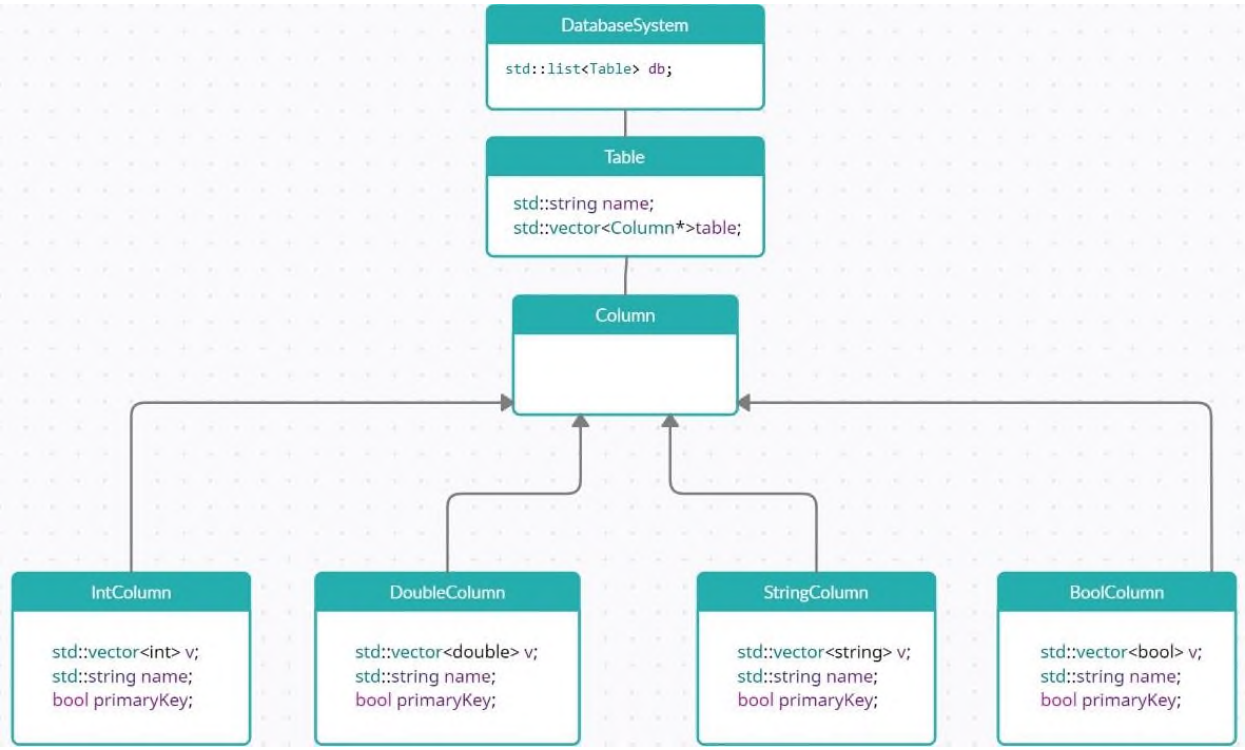
Класът „Таблица“ е изграден от:

- ✓ име
- ✓ набор от типизирани колони

От типа на колоната зависят и наборът от елементи, който тя ще съдържа.

3.2 UML диаграми

Приложената по-долу 1. UML диаграма показва йерархията на проекта СУБД, като визуализира взаимоотношенията между отделните класове. Показани са основни характеристики и функционалности на всеки един отделен клас. Класът DatabaseSystem съдържа `std::list<Table> db;`



1. UML диаграма, показваща йерархията на проекта СУБД

4. Реализация, тестване

4.1 Реализация на класове

Класовете са реализирани чрез обектно-ориентиран код. В проекта са използвани готовите контейнери като „vector” и „string”, предоставени от стандартната библиотека[STL(Standard Library)] на C++.

Приложен е примерен код за декларацията на абстрактния клас „Колона“:

```
class Column
{
public:

    virtual int getType()=0;
    virtual void printName()=0;
    virtual void setName(const std::string &name)=0;
    virtual void addValue(const std::string& val)=0;
    virtual void printValue(int row)=0;
```

```
    virtual std::string getName()=0;
    virtual size_t getSize()=0;
    virtual void updateValue(int row, std::string& val)=0;
    virtual bool hasValueInRow(int row, std::string& val) = 0;
    virtual int count()=0;
    virtual std::string maxValue()=0;
    virtual std::string minValue()=0;
    virtual std::string average()=0;
    virtual std::string sum()=0;
    virtual ~Column(){};

};
```

Приложен е примерен код за част от декларацията на класа „Колона, поддържаща целочислени типове“

```
class Table {
private:
    std::string name;
    std::vector<Column*>table;

    void copy(const Table& other);
    void free();
public:
    Table();
    Table(const std::string& nameVal);
    Table(const Table& other);
    Table& operator=(const Table& other);
    ~Table();


    //adds an empty column
    void addColumn(const std::string& colName, int type, bool
hasPrimaryKey);
    //here the added column must be allocated dynamically on heap;
    void addColumn(Column *c);
    void printName();

    void setName(const std::string& nameVal);
    void printTable();
    void printColumn(int number);
    void insertRow();
    void printRow(int row);
    void select();
    int count(int column,const std::string& key,const std::string&
op);
    void update(int column,const std::string& key,const std::string&
newVal,const std::string& op);
    void saveToFile(const std::string& filename);
    std::string& getName();
```

```
};
```

4.2 Управление на паметта и алгоритми. Оптимизации.

Проектът е реализиран чрез контейнери, предоставени от STL, като е разгледана предварително реализацията на функциите, който са използвани наготово. Класът „Таблица“ съдържа вектор от указатели към колони. Предоставена е примерна дефиниция на конструктора му е един параметър:

```
Table::Table(const std::string &nameVal):name(nameVal) {
    for(int i=0; i<table.size();i++)
    {
        table[i]= nullptr;
    }
}
```

За сортиране е използвана вградената функция sort().

5.Заключение

5.1 Обобщение на изпълнението на началните цели

Проектът реализира по-голямата архитектура от изискванията на клиента.

5.2 Бъдещи перспективи на продукта

Тук се предоставят подобрения, свързани с бъдещата перспектива на продукта:

Проектът СУБД ще предоставя възможност за създаване на списък от потребители, които ще имат достъп до системата, като потребителите ще бъдат разделени на 2 вида:

А) Администратори – те ще поддържат системата

Б) Клиенти – те ще използват системата за извличане, добавяне на нова информация и лесно търсене на определени данни.

Всеки потребител ще има достъп до системата след въвеждане на потребителско име и парола.