# Разработка макета СУБД

## Цель и содержание практики

#  Целями учебной практики «Проектно - технологическая практика» являются:

# - закрепление и углубление теоретической подготовки студентов покурсам «Информатика», «Основы программирования» и «Программирование на основе классов и шаблонов»,

# - приобретение практических навыков программирования с использованием методов групповой разработки программ,

# - разработка библиотеки функций или классов для указанной в задании предметной области, общей для всех членов подгруппы,

# - использование разработанной библиотеки при выполнении индивидуальных заданий КЛР,

# - приобретение навыков разработки технической документации.

## Содержание

Содержание практики.

* Разработать и записать в текстовые файлы таблицы с данными тестового примера для базы данных БИБЛИОТЕКА. Каждая таблица хранится в отдельном файле. Данные хранятся в формате CSV с разделителем ‘|’. Для записи тестовых данных в файл использовать текстовый редактор MS Visual C++. Первая строка файла содержит заголовок таблицы, а остальные строки – данные.
* Разработать структуру DBTable, способную хранить данные произвольных таблиц.
* Разработать функцию readTable() для чтения данных из файла в структуру DBTable (доработать функцию readTable() из ЛР 1так, чтобы она считывала в DBTable не только заголовки, но и строки таблицы).

- Разработать функцию printTable() для печати данных из структуры DBTable

- Разработать функцию writeTable() для записи данных из структуры DBTable в файл в формате CSV;

- Разработать функцию writeTableBin() для записи данных из структуры DBTable в бинарный файл с именем <имя таблицы>.bin;
- Разработать функцию readTableBin() для чтения данных из бинарного файла с именем <имя таблицы>.bin в структуру DBTable;

- Разработать функцию main() для тестирования функций. Для вызова тестируемых функций из main() использовать меню.

* Создать файл DBLibrary.cpp, включающий структуру Table и функции readTable(), writeTable(), printTable(), и подключить его к проекту.
* Перейти от структуры DBTable к классу DBTable.
* Разработать класс DBTableSet, хранящий все таблицы БД.
* Разработать новый (пользовательский) тип данных DBDate для представления времени в таблицах БД.

- Добавить в классы методы, необходимые для выполнения вариантов заданий всех членов группы.

-Установить связи между таблицами.

- Разработать функции для выполнения индивидуального задания КЛР, использующие методы разработанных классов.

Для вызова функций из main() использовать меню.

Перед началом выполнения индивидуального задания разработать документ «Техническое задание» и согласовать его с преподавателем.

## Общие указания

Для выполнения практики студенты разделяются на подгруппы по 3-4 человека. Один из них выбирается бригадиром. Каждая подгруппа выполняет свое задание, используя технологию групповой разработки программ.

Перед началом программирования разработать тестовый пример. В качестве данных для тестового примера использовать таблицы БД, указанной в задании на практику.

***База данных библиотеки*** ***DBLibrary*** содержит служебную таблицу DBTables с именами всех таблиц БД и 4 таблицы с данными:

Abonements, Students, Books и Subjects.

Поля таблиц для БД Library:

Таблица Students: StudentID, Name, Group.

Таблица Subjects: SubjectID, Subject, где:

Subject – тематика книги ( физика, математика, информатика)

Таблица Books: BookID, Title, SubjectID, Quantity, Number, где:

Quantity – количество поступивших в библиотеку экземпляров данной книги.

Number – количество выданных экземпляров данной книги.

Таблица Abonements: StudentID, BookID, OutDate, InDate, где:

OutDate, InDate – дата выдачи, дата возврата книги.

Заполнить файлы тестовыми данными. Ввести не менее трех книг по каждому из трех предметов, не менее 5 записей в таблицу Students и не менее 10 записей в таблицу Abonements.

При заполнении таблицы данными проверять уникальность значений идентификаторов и ссылочную целостность данных, то есть наличие в основных (родительских) таблицах Students, Books и Subjects значений идентификаторов StudentID, BookID и SubjectID, использованных в дочерних таблицах Abonements и Books.

***База данных DBCompany*** компании по оптовой торговле содержит служебную таблицу DBTables с именами всех таблиц БД и 4 таблицы с данными:

Orders, OrderDetails, Customers, Products.

Поля таблиц для БД ***DBCompany***:

Customers: CustomerID, CompanyName, ContactName

OrderDetails: ID, OrderID, ProductID, UnitPrice, Quantity

Orders: OrderID, CustomerID,|OrderDate

Products: ProductID, UnitPrice, UnitInStock

Базы данных хранятся на диске в текстовых файлах с разделителями (формат CSV).

#### Каждая таблица хранится в отдельном файле, <имя файла>=<имя таблицы>.txt.

Спецификация класса DBTable приведена ниже в Приложении 1.

## Варианты заданий

## Вариант 1

Оформление заказа на отпуск товаров со склада (добавить запись и сделать необходимые изменения в таблицах). Перед оформлением заказа сделать проверку наличия товара на складе.

### Вариант 2

Выдача книги в библиотеке. На одних руках не должно быть двух книг по одному предмету. Перед выдачей сделать проверку наличия экземпляра книги в книгохранилище.

### Вариант 3

Для книг, которых нет на полках, вывести список студентов, взявших эту книгу.

### Вариант 4

Подсчитать доход компании от продажи товаров за указанный период времени.

Доход получается за счет разницы оптовой (указана в таблице Products) и отпускной (указана в таблице OrderDetails) цен товара.

### Вариант 5

Подсчитать расходы указанного покупателя за указанный период времени.

### Вариант 6

Вывести список книг по физике, находящихся на руках у заданного абонента.

### Вариант 7

Определить покупателя, который потратил больше всех денег за указанный период времени.

### Вариант 8

Записать студента в библиотеку (обеспечить уникальность ID) и

выдать ему книгу по указанному предмету.

### Вариант 9

Определить студента, который дольше всех держит на руках экземпляр указанной книги.

### Вариант 10

Определить предмет, по которому осталось меньше всего экземпляров книг в хранилище.

## 3. Рекомендации по выполнению работы

### 3.1. Шаги разработки программы

3.1.1. Разработка структуры таблиц для хранения данных в файле и создание в текстовом редакторе файлов с табличными данными.

3.1.2. Разработка структуры DBTable для хранения данных в ОП.

3.1.3. Разработка спецификаций функций для чтения данных из файла в структуру DBTable, печати считанных данных и записи данных из структуры DBTable в файл.

3.1.4. Кодирование и отладка функций чтения, печати и записи данных.

3.1.5. Разработка структуры DBTableSet для хранения данных всех таблиц БД в ОП.

3.1.6. Разработка спецификаций функций для чтения, печати и записи данных для структуры DBTableSet.

3.1.7. Кодирование и отладка функций чтения, печати и записи данных для структуры DBTableSet.

 *Приложение 1.* Файл DBTable.h

#include <map>

#include <string>

#include "DBDate.h"

using namespace std;

typedef map<string, void\*> Row;

typedef map<string, string> Header;

class DBTable

{

private:

 string tableName; //имя таблицы

 Header columnHeaders; //заголовок таблицы

 vector<Row> data;//вектор из строк (записей) таблицы

 void\* GetValue(string value, string columnName); //возвращает

 // указатель void\*, инициализированный адресом

 //значения переменной value, преобразованной из типа

 //string в тип typeName, имя которого указано в заголовке

 //столбца таблицы в CVS-файле.

public:

 DBTable(){} //конструктор по умолчанию

~DBTable(){} //деструктор

 DBTable(string tabName) //конструктор с параметром

 {

 tableName = tabName;

 }

string valueToString(Row& row,string columnName); //возвращает //значение переменной value, преобразованное из типа

//typeName, имя которого указано в заголовке columnName столбца

//таблицы, в тип string.

void ReadTable(string tabName);

void PrintTable(int numcol);

void WriteTable();

int GetSize() //возвращает число строк в таблице

{

 return (int)data.size();

}

Row& operator[](int ind) //перегрузка операции индексации

{

 return data[ind];

}

int GetType(string columnName); //возвращает код типа данных

//в столбце columnName таблицы

bool ValueIsEqual(string columnNameA, void\* objA, void\* objB);

//сравнивает значения данных в столбце columnNameA двух строк //таблицы

vector<Row> SelfRows(string columnName, void\* value);

//возвращае вектор из строк таблицы, в которых значение данных

//в столбце columnName равно \*value

};

Шаблонный класс map<Type1,Type2> - это ассоциативный контейнер, построенный на основе бинарного дерева, в узлах которого хранятся пары значений типа pair<Type1,Type2>.

Type1 – тип ключа, Type2 – тип значения. Значение ключа должно быть уникальным в пределах контейнера. В качестве ключа могут использоваться только те типы, для которых определены операции сравнения.

Контейнер ***map*** можно рассматривать как массив, упорядоченный по значению ключа. Упорядочивание производится при записи данных в контейнер. Например, для добавления значения столбца value в строку заголовка columnName можно использовать оператор:

columnHeaders [columnName]=value;