Документация проекта базы данных на C++

Егоров Александр 210

Содержание

1 Обзор проекта

Проект представляет собой реализацию клиент-серверной системы управления базой данных (СУБД) на языке C++. Система позволяет хранить, управлять и обрабатывать записи о студентах, включая их идентификатор, ФИО, группу, оценку и дополнительную информацию. Проект состоит из трех основных компонентов: сервер (`server.cpp`), клиент (`client.cpp`) и библиотека базы данных (`subd.h`, `subd.cpp`). Все компоненты поддерживают работу с кириллицей (UTF-8 и UTF-16) и обеспечивают многопользовательскую работу с базой данных.

2 Архитектура системы

Система организована по клиент-серверной архитектуре:

- **Сервер** (`server.cpp`): отвечает за обработку запросов клиентов, управление экземплярами базы данных и уведомление клиентов об изменениях.
- **Клиент** (`client.cpp`): предоставляет интерфейс для взаимодействия с сервером, отправки команд и получения результатов.
- **База данных** (`subd.h`, `subd.cpp`): реализует функциональность хранения, выборки, редактирования и удаления записей.

3 Структура базы данных

База данных хранит записи о студентах. Каждая запись представлена структурой `Student` с следующими полями:

- `id`: Уникальный идентификатор (целое число).
- `name`: ФИО студента (строка до 64 символов, кириллица, формат: Фамилия Имя Отчество).
- `group`: Группа (целое положительное число).
- `rating`: Оценка (дробное число от 2.0 до 5.0 с одной цифрой после запятой).
- `info`: Дополнительная информация (строка произвольной длины).

Данные хранятся в памяти в формате UTF-16, а в файлах — в формате UTF-8. Для оптимизации выборки используются три красно-черных дерева (`std::set`) для индексации по полям `name`, `group` и `rating`.

4 Команды

Система поддерживает следующие команды для управления базой данных:

Команда	Сигнатура	Описание
open	<название файла>	Выбор файла базы данных.
save		Сохранение базы данных в файл.
select	[id=<...>, name=<...>, group=<...>, rating=<...>]	Выборка записей по критериям.
reselect	[id=<...>, name=<...>, group=<...>, rating=<...>]	Повторная выборка среди уже выбранных записей.
update	<name=<...>, group=<...>, rating=<...>, info=<...»	Редактирование выбранных записей.
remove		Удаление выбранных записей.
add	<фио> \t <группа> \t <оценка> \t <информация>	Добавление новой записи.
print	[id, name, group, rating, info] [range=<...>] [sort <name/group/rating>]	Вывод выбранных записей с возможностью сортировки и указания диапазона.

4.1 Формат критериев

Критерии для команд select, reselect, update, remove задаются в следующем формате:

Поле	Допустимые значения
id	*, конкретное число (например, 1), диапазон (например, 1-100, *-100, 1-*).
name	*, "*", строка с маской (например, Кузьмин*), точное ФИО (например, "Кузьмин Иван Иванович").
group	*, конкретное число (например, 101), диапазон (например, 101-103, *-105, 104-*).
rating	*, конкретное число (например, 4), диапазон (например, 4-5, *-4, 4-*).

5 Клиент (client.cpp)

Клиент подключается к серверу по IP-адресу и порту, указанным в конфигурационном файле client_config.ini. Он предоставляет консольный ин-

терфейс для ввода команд, отправляет их на сервер и отображает ответы. Дополнительные команды клиента:

- `help`: Выводит справочную информацию о командах.
- `clear`: Очищает консоль.
- `reconnect`: Переподключается к серверу.
- `exit`: Завершает работу клиента.

Клиент обрабатывает уведомления от сервера об изменениях базы данных, запрашивая подтверждение перед выполнением команды, если данные были изменены другим пользователем.

6 Сервер (`server.cpp`)

Сервер обрабатывает подключения клиентов, создавая для каждого клиента отдельный поток и экземпляр базы данных (`Database`). Конфигурация сервера (порт, максимальное количество клиентов) задается в файле `server_config.i`
Сервер поддерживает:

- Многопользовательский доступ к файлам базы данных.
- Уведомления клиентов об изменениях в базе данных.
- Управление экземплярами базы данных для каждого клиента.

7 Реализация базы данных (`subd.h`, `subd.cpp`)

Класс `Database` реализует основную функциональность СУБД:

- Хранение данных в памяти с использованием `std::vector<Student>` и индексация через `std::set` для полей `name`, `group`, `rating`.
- Поддержка операций выборки (`select`, `reselect`), редактирования (`update`), удаления (`remove`) и добавления (`add`).
- Сохранение и загрузка данных из файлов в формате UTF-8.
- Уведомления об изменениях через механизм обратных вызовов (`notifyOnChange`).

Ниже приведены ключевые части реализации.

7.1 Класс Database

```
class Database {
private:
    struct Student {
        int id;
        wchar_t name[64];
        int group;
        double rating;
        std::wstring info;
    };
    std::vector<Student> students;
    std::set<Index, CompareByName> studentsBN;
    std::set<Index, CompareByGroup> studentsBG;
    std::set<Index, CompareByRating> studentsBR;
    std::vector<size_t> selectedStudents;
    std::wstring dbFile;
    int nextId = 1;
    size_t version = 0;
    std::vector<std::function<void()>> changeCallbacks;
public:
    void selectDB(const std::wstring& filename);
    void saveDB();
    void select(const std::wstring& command);
    void reselect(const std::wstring& command);
    void print(const std::wstring& fields) const;
    void update(const std::wstring& command);
    void remove();
    void add(const std::wstring& command);
};
```

7.2 Взаимодействие клиент-сервер

Пример обработки команды на сервере:

```
void handle_client(int clientSocket) {
    std::wstring current_db_file;
    std::shared_ptr<Database> db_ptr;
    while (true) {
        int msgLength;
        ssize_t bytesRead = recv(clientSocket, &msgLength,
           sizeof(int), 0);
        if (bytesRead <= 0) break;
        char* buffer = new char[msgLength + 1];
        //
        std::wstring wmessage = utf8_to_utf16(buffer);
        WcoutRedirect redirect;
        if (wmessage.substr(0, 4) == L"open") {
            current_db_file = wmessage.substr(5);
            db_ptr = std::make_shared<Database>();
            db_ptr->selectDB(current_db_file);
```

```
            db_ptr->notifyOnChange([current_db_file, clientSocket]() {
                notify_clients_db_update(current_db_file);
            });
        } else if (db_ptr) {
            db_ptr->parseCommand(wmessage);
        }
        std::string message = utf16_to_utf8(redirect.getOutput());
        send(clientSocket, &message.size(), sizeof(int), 0);
        send(clientSocket, message.c_str(), message.size(), 0);
    }
}
```

8 Конфигурация

- `client_config.ini`: Содержит `server_ip` (IP-адрес сервера) и `port` (порт для подключения).
- `server_config.ini`: Содержит `port` (порт сервера) и `max_clients` (максимальное количество клиентов).

9 Сборка и запуск

Для сборки проекта требуется компилятор C++ с поддержкой C++17. Пример сборки:

```
g++ -std=c++17 server.cpp subd.cpp -o server
g++ -std=c++17 client.cpp subd.cpp -o client
```

Запуск сервера:

```
./server
```

Запуск клиента:

```
./client
```

10 Заключение

Проект представляет собой полноценную клиент-серверную СУБД с поддержкой многопользовательского доступа и работы с кириллицей. Код оптимизирован для быстрой выборки за счет использования красно-черных деревьев и обеспечивает базовые операции управления данными. Дальнейшее развитие проекта может включать улучшение безопасности, производительности и функциональности.