МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ

образовательное учреждение

высшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации

**

**ПРАКТИЧЕСКАЯ РАБОТА №2**

**«**Разработка сетевого интерфейса

системы управления базой данных.**»**

**по дисциплине: «*Программирование*»**

|  |  |
| --- | --- |
| Выполнил:  Студент гр. «АБ-320», «АВТФ»  *Мартынова Е.Д.*  «28» ноября 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Проверил:  *Заместитель декана «АВТФ»*  *Архипова А.Б.*  «\_\_\_» \_\_\_\_\_\_ 2024г  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (подпись) |

Новосибирск 2024

Оглавление

[Задание к работе 2](#_Toc183722562)

[Описание функций 3](#_Toc183722563)

[Подключенные библиотеки 7](#_Toc183722564)

[Примеры работы программы 9](#_Toc183722565)

[ПРИЛОЖЕНИЕ 11](#_Toc183722566)

Задание к работе

Требуется реализовать сетевой интерфейс для СУБД из практики 1. При запуске приложения из первой практики СУБД должна ожидать соединение по протоколу tcp на порту 7432. При подключении требуется обработать запрос либо в отдельном потоке, либо в отдельном процессе, либо асинхронной задачей и отдать результат в ответ на запрос. Также необходимо позаботится о блокировках на структуре БД если используются потоки или процессы.

# Описание функций

WriteUtf8BOM(ofstream& file) - Записывает BOM для UTF-8 в указанный файл.

FreeTable(RowNode table)\* - Освобождает память, занятую строками и ячейками таблицы.

FreeAllTables(RowNode tables, int count)\*\* - Освобождает память для всех указанных таблиц.

UpdatePrimaryKey(const string& schemaName, const string& tableName, RowNode table)\* - Обновляет последовательность первичного ключа таблицы, записывая количество строк в файл.

countRowsInCSV(const string& csvFilePath) - Считывает и возвращает количество строк в указанном CSV файле.

ReadConfiguration(const string& filename) - Читает конфигурацию из файла и создает директории и CSV файлы для каждой таблицы.

PrintTable(RowNode table) - Выводит содержимое одной таблицы в консоль.

PrintTables(RowNode tables) - Выводит содержимое нескольких таблиц в консоль.

AddTableNames(RowNode table, RowNode\* tables, int countTables) - Добавляет названия таблиц в ячейки строк указанной таблицы.

AddNumColumns(RowNode table, int numColumns[], int countTables)\* - Добавляет номера колонок в ячейки строк указанной таблицы.

InsertInto(RowNode table, const string listString[])\* - Вставляет новую строку в указанную таблицу, используя данные из массива строк.

SelectFromOneTable(RowNode table, int numColumns[]) - Выбирает определенные колонки из одной таблицы и возвращает новую таблицу с выбранными данными.

SelectFromManyTables(RowNode tables, int numColumns[], int countTables) - Выбирает определенные колонки из нескольких таблиц и возвращает новую таблицу с объединенными данными.

SelectFromCartesian(RowNode tables, int countTables) - Выполняет декартово произведение всех указанных таблиц, возвращая новую таблицу.

SplitCondition(const string& filter) - Разделяет условие фильтрации на отдельные логические выражения и возвращает список условий.

CheckingCondition(RowNode row, const string& condition) - Проверяет, соответствует ли строка заданному условию.

CheckingLogicalExpression(RowNode row, Condition condition)\*\* - Оценивает логическое выражение на основе условий, связанных логическими операторами.

FilteringTable(RowNode select, RowNode where, int selectSize, int whereSize, int numColumnsSelect[], int numColumnsWhere[], string condition)\*\* - Фильтрует строки из таблиц на основе заданных условий и возвращает новую таблицу.

DeleteFrom(RowNode table, Condition condition) - Удаляет строки из таблицы, соответствующие заданному условию.

void CreateTable(const string& schemaDir, const string& tableName) - Создает директорию и необходимые файлы для новой таблицы в указанной схеме, включая CSV файл и файлы для первичного ключа и блокировки.

json ConvertTableToJson(RowNode table)\* - Преобразует данные таблицы в формате RowNode в JSON массив, представляющий строки и ячейки таблицы.

RowNode ConvertJsonToTable(const json& jsonData, const string& tableName)\* - Преобразует JSON данные в структуру таблицы RowNode, добавляя строки в таблицу, если они существуют в JSON.

void WriteJsonToCSV(const string& filePath, const json& jsonData) - Записывает данные из JSON массива в CSV файл, добавляя BOM для поддержки UTF-8, если файл пустой.

void splitString(string& str, string tokens)\* - Разделяет строку на токены по запятой, удаляя фигурные скобки и пробелы.

json ReadCSVToJson(const string& filePath) - Читает данные из указанного CSV файла и преобразует их в JSON формат, возвращая массив строк.

void AddColumnsInSchemaJson(const json& newColumns, const string& schemaFilePath, const string& tableName) - Добавляет новые колонки в схему таблицы в JSON формате, обновляя файл схемы.

void RewriteTableSchema(const json& newSchema, const string& schemaFilePath, const string& tableName) - Перезаписывает схему указанной таблицы в файле схемы на новую структуру, переданную в виде JSON.

void RewriteCSVbyJson(const string& filePath, const json& jsonData) - Перезаписывает CSV файл, основываясь на данных из JSON массива, добавляя BOM для поддержки UTF-8.

void incrementSequence(const string& tableName) - Увеличивает значение первичного ключа в соответствующем текстовом файле, хранящемся в директории таблицы.

void decrementLock(const string& tableName) - Уменьшает значение блокировки в соответствующем текстовом файле, хранящемся в директории таблицы.

int getNextCsv(const string& tableName) - Определяет номер следующего CSV файла для таблицы, основываясь на текущем количестве строк и существующих файлах в директории.

int DBMS\_Queries(string& command, ostringstream& toClient) – Вызывает все функции в программе, представляя реализованные команды СУБД.

void SignalCheck(int signal) – Определяет системное прерывание и меняет флаг для завершения программы.

void ConnectionProcessing(int clientSocket, int serverSocket, ostringstream& toClient) – Обрабатывает полученные данные клиента и отправляет ответ.

int main() – Запускает интерфейс сервера/клиента, принимает/отправляет данные.

# Подключенные библиотеки

#include <iostream> - Библиотека для работы с вводом и выводом данных через стандартные потоки (cin, cout).

#include <string> - Библиотека для работы со строками и их манипуляциями. #include <algorithm> - Библиотека с функциями для выполнения операций над контейнерами, например, сортировка и поиск.

#include <sstream> - Библиотека для работы с потоками строк, позволяющая преобразовывать строки в другие типы и обратно.

#include "json.hpp" - Библиотека для работы с JSON данными, обеспечивающая парсинг и генерацию JSON формата.

#include <fstream> - Библиотека для работы с файловыми потоками, позволяющая читать и записывать данные в файлы.

#include <filesystem> - Библиотека для работы с файловой системой, обеспечивающая доступ к файлам и директориям.

#include <locale> - Библиотека для работы с локалями, позволяющая адаптировать программы к различным языковым и культурным условиям.

#include <unistd.h> — Библиотека для работы с низкоуровневыми операциями, такими как управление процессами (например, fork, exec, getpid) и файловыми дескрипторами (например, close).

#include <string.h> — Библиотека для работы с функциями манипуляции строками, такими как memset, memcpy, strcmp, и strlen. Также включает функции для работы с бинарными данными.

#include <arpa/inet.h> — Библиотека, содержащая функции для работы с сетевыми адресами, такие как inet\_addr (преобразует строку в формат адреса) и inet\_ntoa (преобразует IP-адрес в строковое представление).

#include <sys/socket.h> — Библиотека, определяющая структуры и функции для работы с сокетами в сетевых приложениях. Включает символы для различных типов сокетов (например, AF\_INET для IPv4) и протоколов (например, SOCK\_STREAM для TCP).

#include <netinet/in.h> — Библиотека, которая содержит определения для работы с интернет-адресами и портами, включая структуру sockaddr\_in, которая используется для задания адресов в сети в формате IPv4.

# Примеры работы программы

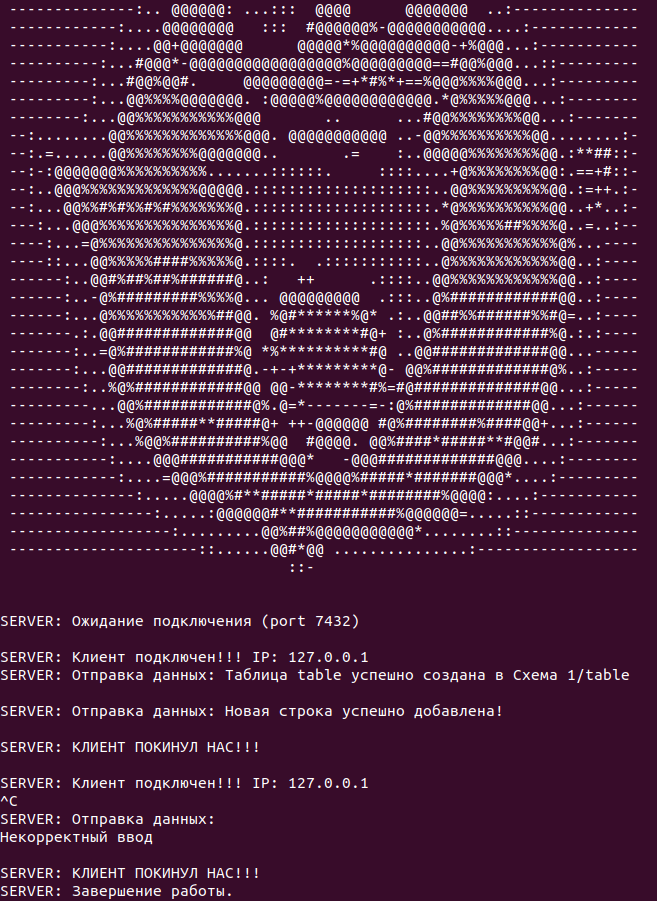


Рисунок 1. Пример работы сервера с подключенным клиентом на локальном хосте.

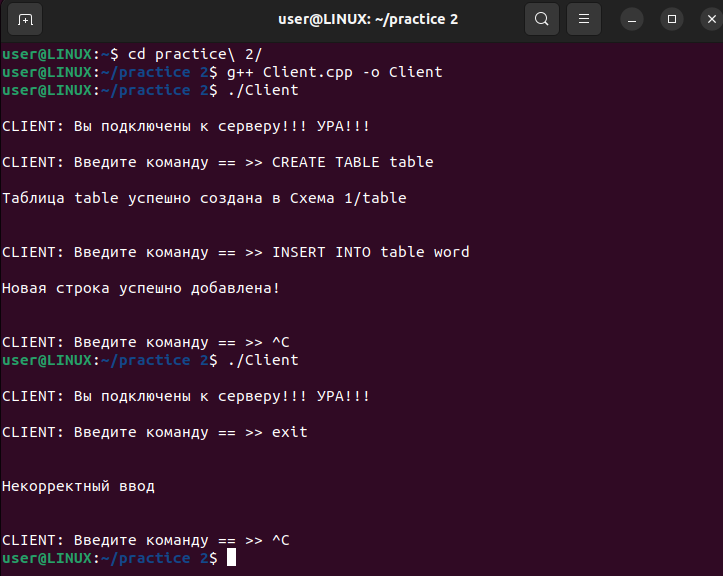


Рисунок 2. Пример работы клиента на локальном хосте.

# ПРИЛОЖЕНИЕ

Ссылка на GitHub: <https://github.com/mamam111a/practice-2>