Национальная научно-образовательная корпорация ИТМО

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

Лабораторная работа №3

по дисциплине

«Низкоуровневое программирование»

Группа: P33312
Выполнил:
Лебедев Вячеслав Владимирович
Преподаватель:
Кореньков Юрий Дмитриевич

Санкт-Петербург

2023

Цели

На базе данного транспортного формата описать схему протокола обмена информацией и воспользоваться существующей библиотекой по выбору для реализации модуля, обеспечивающего его функционирование.

Протокол должен включать представление информации о командах создания, выборки, модификации и удаления данных в соответствии с данной формой, и результатах их выполнения.

Используя созданные в результате выполнения заданий модули, разработать в виде консольного приложения две программы: клиентскую и серверную части.

Серверная часть – получающая по сети запросы и операции описанного формата и последовательно выполняющая их над файлом данных с помощью модуля из первого задания. Имя файла данных для работы получать с аргументами командной строки, создавать новый в случае его отсутствия.

Клиентская часть – в цикле получающая на стандартный ввод текст команд, извлекающая из него информацию о запрашиваемой операции с помощью модуля из второго задания и пересылает ее на сервер с помощью модуля для обмена информацией, получая ответ и выводящая его в человеко-понятном виде в стандартный вывод.

Решение

В качестве библиотеки была выбрана реализация формата Protobuf для языка C: https://github.com/protobuf-c/protobuf-c

Так как эта версия библиотеки содержит только функционал для сериализации и десериализации структур, то для сетевого обмена были использованы средства ОС - TCP сокеты.

Был разработан новый модуль network, содержащий функции для удобной работы с библиотекой, а также схема протокола обмена, описанная в файлах .proto внутри папки proto.

Между клиентом и сервером идет обмен структурами Message - инкапсулирующими произвольное сообщение. При отправке сообщения, в начале в сокет пишется его размер, а потом содержимое. За счет этого принимающая сторона может выделить достаточный буфер для считывания сообщения. После Message десериализуется средствами библиотеки, готовая структура содержит либо Request - запрос клиента, либо Response - ответ сервера.

Request может содержать разные запросы: создание данных, удаление, изменение и т. д. На каждый вид запроса есть свои виды ответов.

Серверная часть была разработана в модуле server, как консольное приложение со следующими аргументами командной строки:

-m, --mode [=create, exist, clear] - обязательный аргумент, для способа открытия файла: create - будет создан новый и проинициализирован внутренними структурами, exist - файл уже есть, clear - файл уже есть, но при открытии внутренние структуры будут очищены

-f, --file - обязательный аргумент для указания пути файла

-p, --port - опциональный аргумент для указания прослушивающего порта, если не указан, используется любой свободный

-h, --help - справка по аргументам

После запуска приложения выводится справочная информация:

```
/home/wieceslaw/CLionProjects/single-file-database/cmake-build-debug/server/Server --mode=exist
  --file=/home/wieceslaw/CLionProjects/single-file-database/test.db
[INFO]: Port if not specified. Using available
[INFO]: Server started with port: 48171
[INFO]: New client
Connection.c::ConnectionRun::114 Connection run
[INFO]: Client disconnect
Connection.c::ConnectionFree::156 Connection free
```

Также сервер прослушивает стандартный вход и может быть gracefully остановлен при помощи команды stop, которая отключит все соединения и освободит ресурсы:

```
/home/wieceslaw/CLionProjects/single-file-database/cmake-build-debug/server/Server --mode=exist
  --file=/home/wieceslaw/CLionProjects/single-file-database/test.db
[INFO]: Port if not specified. Using available
[INFO]: Server started with port: 48171
[INFO]: New client
Connection.c::ConnectionRun::114 Connection run
[INFO]: Client disconnect
Connection.c::ConnectionFree::156 Connection free
stop
[INFO]: Stopping server...
[INFO]: Server stopped
```

В отдельном потоке в цикле принимаются соединения и отправляются на обработку в свои потоки.

Каждое клиентское соединения выполняется в своем потоке и при обращении к объекту базы встает на lock по общему для всех соединений мьютексу.

При отключении клиента поток останавливается и освобождает свои ресурсы.

Клиентская часть была разработана в модуле client в виде консольного приложения, принимающего 2 аргумента командной строки - адрес сервера и порт.

После успешного создания соединения, клиент имеет возможность ввести несколько команд:

\q - gracefully выйти из приложения

\d - вывести список доступных таблиц базы

А также команды на подмножестве SQL для выполнения запросов на:

- создание и удаление таблиц
- вставку, модификацию, удаление и просмотр данных

Команды на SQL преобразуются в синтаксическое дерево при помощи библиотеки из модуля parser, реализованной в Лабораторной №2. Дерево преобразуется в запрос в формате Protobuf, описанный в модуле network и отправляется на сервер. А ответ сервера выводится в консоль.

Полный цикл работы клиента-сервера:

\d - список существующих таблиц

```
database=# \d
List of relations

Table "test2"
------------------
| Column  | Type |
+---------+------+
| test_id | int  |
| label   | text |
------------------

Table "user"
-----------------
| Column | Type |
+--------+------+
| id     | int  |
| name   | text |
-----------------

Table "test"
-----------------
| Column | Type |
+--------+------+
| int    | int  |
```

Создание таблицы:

```
database=# create table group(
database-#  id INT32,
database-#  name TEXT,
database-#  subscribers INT32,
database-#  is_banned BOOLEAN
database-# );
OK
database=#
```

```
database=# \d
List of relations

Table "test2"
------------------
| Column  | Type |
+---------+------+
| test_id | int  |
| label   | text |
------------------

Table "group"
-------------------------
| Column      | Type    |
+-------------+---------+
| id          | int     |
| name        | text    |
| subscribers | int     |
| is_banned   | boolean |
-------------------------
```

Удаление таблицы:

```
database=# delete table test2;
OK
database=# \d
List of relations

Table "group"
-------------------------
| Column      | Type    |
+-------------+---------+
| id          | int     |
| name        | text    |
| subscribers | int     |
| is_banned   | boolean |
-------------------------

Table "user"
```

Добавление и выборка данных:

```
database=# insert into group values (1, 'Cats Fans', 42, false);
OK
database=# select group.id, group.name, group.subscribers, group.is_banned from group;
-----------------------------------------------------------------
| group.id | group.name | group.subscribers | group.is_banned |
+----------+------------+-------------------+-----------------+
| 1        | Cats Fans  | 42                | false           |
-----------------------------------------------------------------
ROWS (1)
database=#
```

Выборка с условием:

```
database=# insert into group values (2, 'Dogs Fans', 42, true);
OK
database=# insert into group values (3, 'Horse Fans', 42, false);
OK
database=# insert into group values (4, 'Chicken Fans', 52, false);
OK
database=# select group.id, group.name, group.subscribers, group.is_banned from group;
-------------------------------------------------------------------
| group.id | group.name   | group.subscribers | group.is_banned |
+----------+--------------+-------------------+-----------------+
| 1        | Cats Fans    | 42                | false           |
| 2        | Dogs Fans    | 42                | true            |
| 3        | Horse Fans   | 42                | false           |
| 4        | Chicken Fans | 52                | false           |
-------------------------------------------------------------------
ROWS (4)
database=# select group.id, group.name, group.subscribers, group.is_banned from group
database-# where group.is_banned = true;
-----------------------------------------------------------------
| group.id | group.name | group.subscribers | group.is_banned |
+----------+------------+-------------------+-----------------+
| 2        | Dogs Fans  | 42                | true            |
-----------------------------------------------------------------
ROWS (1)
database=#
```

Обновление данных:

```
database=# update group
database-# set is_banned = false
database-# where group.is_banned = true;
UPDATED (1)
database=# select group.id, group.name, group.subscribers, group.is_banned from group;
-----------------------------------------------------------------
| group.id | group.name   | group.subscribers | group.is_banned |
+----------+--------------+-------------------+-----------------+
| 1        | Cats Fans    | 42                | false           |
| 2        | Dogs Fans    | 42                | false           |
| 3        | Horse Fans   | 42                | false           |
| 4        | Chicken Fans | 52                | false           |
-----------------------------------------------------------------
ROWS (4)
database=#
```

Удаление данных:

```
database=# select group.id, group.name, group.subscribers, group.is_banned from group;
-----------------------------------------------------------------
| group.id | group.name   | group.subscribers | group.is_banned |
+----------+--------------+-------------------+-----------------+
| 1        | Cats Fans    | 42                | false           |
| 2        | Dogs Fans    | 42                | false           |
| 3        | Horse Fans   | 42                | false           |
| 4        | Chicken Fans | 52                | false           |
-----------------------------------------------------------------
ROWS (4)
database=# delete from group
database-# where group.name = 'Cats Fans' or group.subscribers = 52;
DELETED (2)
database=# select group.id, group.name, group.subscribers, group.is_banned from group;
-----------------------------------------------------------------
| group.id | group.name | group.subscribers | group.is_banned |
+----------+------------+-------------------+-----------------+
| 3        | Horse Fans | 42                | false           |
| 2        | Dogs Fans  | 42                | false           |
-----------------------------------------------------------------
ROWS (2)
database=#
```

Выборка с соединением и фильтрацией:

```
database=# create table user (
database-#  id INT32,
database-#  name TEXT
database-# );
OK
database=# \d
List of relations

Table "user"
-----------------
| Column | Type |
+--------+------+
| id     | int  |
| name   | text |
-----------------
```

```
database=# create table test (
database-# id int32
database-# ,
database-# surname TEXT
database-# );
OK
database=# \d
List of relations

Table "user"
-----------------
| Column | Type |
+--------+------+
| id     | int  |
| name   | text |
-----------------
```

```
database=# insert into test values (1, 'Johnson');
OK
database=# insert into test values (2, 'Topson');
OK
database=# insert into test values (3, 'Bobson');
OK
database=# select test.id, test.surname, user.id, user.name from test join user on test.id = user.id;
-----------------------------------------------
| test.id | test.surname | user.id | user.name |
+---------+--------------+---------+-----------+
-----------------------------------------------
ROWS (0)
```

```
database=# insert into user values (1, 'John');
OK
database=# insert into user values (2, 'Top');
OK
database=# insert into user values (3, 'Bob');
OK
database=# select test.id, test.surname, user.id, user.name from test join user on test.id = user.id;
-----------------------------------------------
| test.id | test.surname | user.id | user.name |
+---------+--------------+---------+-----------+
| 1       | Johnson      | 1       | John      |
| 2       | Topson       | 2       | Top       |
| 3       | Bobson       | 3       | Bob       |
-----------------------------------------------
ROWS (3)
```

```
database=# select test.id, test.surname, user.id, user.name from test
database-# join user on test.id = user.id
database-# where user.name = 'Bob';
-------------------------------------------------
| test.id | test.surname | user.id | user.name |
+---------+--------------+---------+-----------+
| 3       | Bobson       | 3       | Bob       |
-------------------------------------------------
ROWS (1)
database=#
```

Вывод

В ходе выполнения работы были созданы 3 новых модуля:

- network для сетевого взаимодействия на основе Protobuf
- server для обработки клиентских запросов, использующий библиотеку из Лабораторной №1 для работы с файлом
- client для создания запросов к серверу, использующий библиотеку для парсинга SQL запросов из Лабораторной №3