ИТМО

Факультет программной инженерии и компьютерной техники

Направление подготовки: Информатика и вычислительная техника

Дисциплина «Низкоуровневое программирование»

Лабораторная работа №3

Вариант 1

Выполнил:

Богатов А. С.

P33302

Преподаватель

Кореньков Ю.Д.

г. Санкт-Петербург, 2023 г.

Задание:

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 пересылающая её на сервер с помощью модуля для обмена информацией, получающая ответ и выводящая его в человеко-понятном виде в стандартный вывод.

Вариант - XML

Используемые средства:

- В качестве библиотеки для парсинга и валидации по схеме XML запроса была выбрана libxml2
- Для реализации асинхронного ввода/вывода по сети была использована библиотека для C++ - boost.asio.
- Линковка с библиотеками и сборка модуля - CMake
- Результаты утилит Flex & Bison, полученные в лабораторной 2

Разработанный модуль:

Разработанный модуль реализует клиент-сервер взаимодействие - в роли клиента выступает модуль синтаксического анализа языка AQL из лабораторной 2, к модулю добавлен подмодуль по составлению XML запроса к серверу на основе сгенерированного AST.

Клиент запрашивает порт сервера при запуске.

Клиент выводит результаты операций на сервере, в том числе ошибки или сообщения о некорректности запроса.

Основой серверной части является модуль реляционной базы данных из лабораторной 1. Модуль был частично переработан для возможности возвращать результат операции с сервера на клиент в читабельном виде.

При запуске сервер запрашивает порт на котором будет открыто соединение, название файла с БД. При получении запроса от клиента, средствами libxml2 сервер осуществляет валидацию XML запроса в соответствии с XSD схемой.

При успешной валидации вызывается обработка запроса. На сервере поддержаны операции выборки, выборки с соединением, удаления, вставки данных, обновления данных и создания отношения в БД.

XSD Schema:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">

   <xs:complexType name="variant">
       <xs:simpleContent>
           <xs:extension base="xs:string">
               <xs:attribute name="type" type="xs:string" />
           </xs:extension>
       </xs:simpleContent>
   </xs:complexType>

   <xs:complexType name="compare_t">
       <xs:sequence>
           <xs:element name="left_operand" type="variant" />
           <xs:element name="compare_by" type="xs:string"/>
           <xs:element name="right_operand" type="variant"/>
       </xs:sequence>
   </xs:complexType>

   <xs:complexType name="cmp_t">
       <xs:choice>
           <xs:element name="filter" type="compare_t"/>
           <xs:element name="nullable" type="xs:string"/>
       </xs:choice>
   </xs:complexType>

   <xs:complexType name="join_t">
       <xs:choice>
           <xs:element name="nullable" type="xs:string"/>
           <xs:element name="value" type="variant" minOccurs="0"
maxOccurs="1"/>
       </xs:choice>
   </xs:complexType>

   <xs:complexType name="select_t">
       <xs:sequence>
           <xs:element name="table" type="xs:string"/>
```

```
            <xs:element name="cmp" type="cmp_t"/>
            <xs:element name="join" type="join_t"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="delete_t">
        <xs:sequence>
            <xs:element name="table" type="xs:string"/>
            <xs:element name="cmp" type="cmp_t"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="update_t">
        <xs:sequence>
            <xs:element name="table" type="xs:string"/>
            <xs:element name="cmp" type="cmp_t"/>
            <xs:element name="list_values">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="list" type="list_t"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="list_t">
        <xs:sequence>
            <xs:element name="pair" minOccurs="1" maxOccurs="50">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="value" type="variant"
minOccurs="2" maxOccurs="2"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="insert_t">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
```

```
            <xs:element name="list" type="list_t"/>
        </xs:sequence>
    </xs:complexType>

    <xs:complexType name="create_t">
        <xs:sequence>
            <xs:element name="name" type="xs:string"/>
            <xs:element name="list_values">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="list" type="list_t"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>



    <xs:complexType name="command_t">
        <xs:choice>
            <xs:element name="select" type="select_t"/>
            <xs:element name="update" type="update_t"/>
            <xs:element name="insert" type="insert_t"/>
            <xs:element name="create" type="create_t"/>
            <xs:element name="delete" type="delete_t"/>
        </xs:choice>
    </xs:complexType>

    <xs:element name="command" type="command_t"/>

</xs:schema>
```

Пример работы модуля:

Сессия со стороны сервера:

```
File created
Enter port
9030
Waiting for requests
Client connected
```

```
100 HELLO
200 HELLO
300 HELLO
Selected 3 rows
F 2
Joined 1 rows
Updated 1 rows
Deleted 3 rows
```

Сессия со стороны клиента:

```
Enter port
9030
$> --(end of buffer or a NUL)
CREATE table_b { field_one: INT, active: STRING };
created table_b
$> --(end of buffer or a NUL)
INSERT { field_one: 100, active: "HELLO" } IN table_b;
inserted into table_b
$> --(end of buffer or a NUL)
INSERT { field_one: 200, active: "HELLO" } IN table_b;
inserted into table_b
$> --(end of buffer or a NUL)
INSERT { field_one: 300, active: "HELLO" } IN table_b;
inserted into table_b
$> --(end of buffer or a NUL)
FOR a IN table_b FILTER a.active == "HELLO" return a;
 100 HELLO
 200 HELLO
 300 HELLO
 Selected 3 rows
$> --(end of buffer or a NUL)
CREATE table_a { field_one: INT, field_two: INT};
created table_a
$> --(end of buffer or a NUL)
INSERT { field_one: 4, field_two: 1 } IN table_a;
inserted into table_a
$> --(end of buffer or a NUL)
INSERT { field_one: 5, field_two: 2 } IN table_a;
inserted into table_a
$> --(end of buffer or a NUL)
CREATE table_d { field_one: INT, field_two: BOOL};
```

```
created table_d
$> --(end of buffer or a NUL)
INSERT { field_one: 5, field_two: false } IN table_d;
inserted into table_d
$> --(end of buffer or a NUL)
FOR a IN table_a FOR b IN table_d FILTER a.field_one ==
b.field_one RETURN a,b;
 F  2
 Joined 1 rows
$> --(end of buffer or a NUL)
CREATE table_e { field_one: INT, field_two: BOOL};
created table_e
$> --(end of buffer or a NUL)
INSERT { field_one: 1, field_two: true } IN table_e;
inserted into table_e
$> --(end of buffer or a NUL)
INSERT { field_one: 2, field_two: true } IN table_e;
inserted into table_e
$> --(end of buffer or a NUL)
INSERT { field_one: 3, field_two: true } IN table_e;
inserted into table_e
$> --(end of buffer or a NUL)
FOR a IN table_e FILTER a.field_two == true REMOVE a IN table_e;
 Deleted 3 rows
$> --(end of buffer or a NUL)
```

Требования к запуску:

- ОС Windows/*NIX
- C++20 компилятор (Clang 14+/gcc 10)
- CMake 3.21+
- boost 1.8+
- libxml2 2.10.3+

Выводы:

Была изучена работа с библиотеками в C и C++, основы взаимодействия клиент-сервер в C++ с помощью стандартных средств языка и библиотеки boost.asio, опробована библиотека для парсинга и валидации по схеме XML файлов/строк. Был получен опыт комбинирования модулей на C с модулем на C++ и потрачено значительно количество нервов.