Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

Лабораторная работа №3 по дисциплине

«**Низкоуровневое программирование**»

**Выполнил:**

Миху Вадим Дмитриевич

Факультет “ПИиКТ”

Группа: P33301

**Преподаватель:**

Кореньков Юрий Дмитриевич

Вариант 5



Санкт-Петербург, 2023 г.

Оглавление

[Задание 2](#_Toc145016776)

[Выполнение 3](#_Toc145016777)

[Выводы по работе 5](#_Toc145016778)

Задание

**Вариант №5**

Создать 2 консольных приложения, которые обмениваются информацией через XML по своему протоколу. Серверный модуль должен использовать функции, реализованные в первой ЛР и по сути представляет из себя удаленную СУБД. Клиентский модуль должен парсить запросы на языке AQL и пересылать их серверу.

• Изучить доступные средства для парсинга и валидации XML.

• Спроектировать структуры данных для представления запроса.

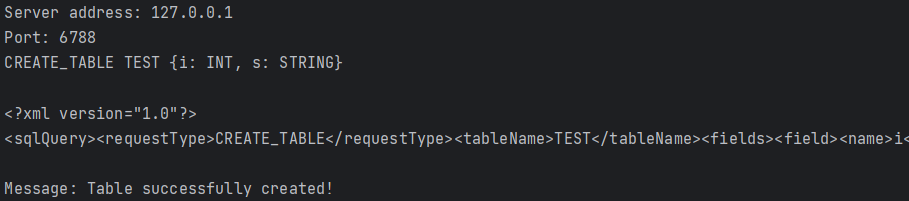
• Реализовать публичный интерфейс для приведенных выше операций

• Реализовать тестовую программу для демонстрации работоспособности решения

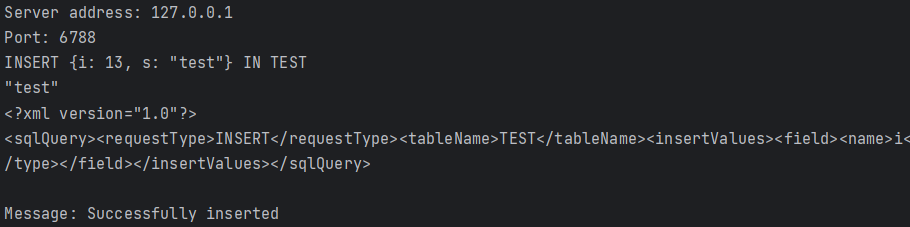
Выполнение

Ссрверу в параметры передается адрес, порт и название файла базы данных, а клиенту только адрес и порт.

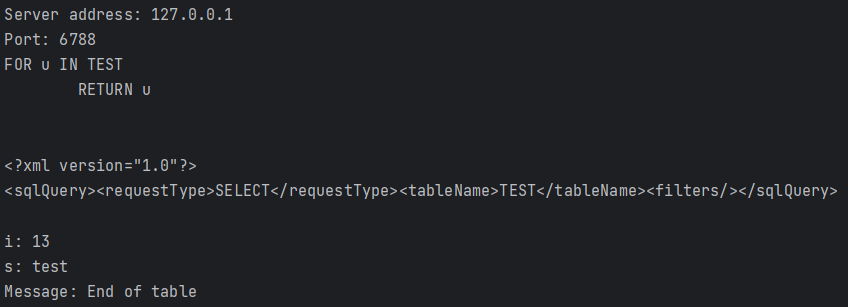
Создание тестовой таблицы



Вставка данных



Выборка данных



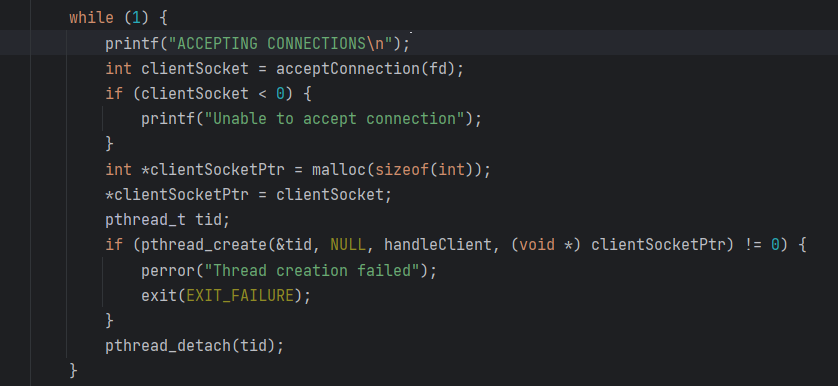
Описание примененных решений

Для разработки использовалась библиотека LIBXML2 — одна из самых популярных библиотек под XML на С. На клиенте и на сервере были реализованы модули net и хml, которые позволяют передавать сообщения и парсить их. Сообщение передается по обычным сокетам, парсинг вида команды происходит с помощью специального тега.

Сервер имеет структуру реквестов, которые может принять и обработать. На клиенте запрос формируется в дерево, после чего конвертируется в xml, отправляется на сервер. Сервер формирует структуру запроса и выполняет его в соответствии с типом.

enum requestType {  
 *CREATE\_TABLE*,  
 *DROP\_TABLE*,  
 *INSERT*,  
 *SELECT*,  
 *DELETE*};  
  
struct createTableRequest {  
 char \*tableName;  
 int32\_t columnNum;  
 enum DataType \*types;  
 const char \*\*names;  
};  
  
struct dropTableRequest {  
 char \*tableName;  
};  
  
struct insertRequest {  
 char \*tableName;  
 int32\_t dataCount;  
 void \*\*data;  
};  
  
struct selectRequest {  
 char \*tableName;  
 struct Condition \*conditions;  
};  
  
  
struct deleteRequest {  
 char \*tableName;  
 int32\_t conditionCount;  
 struct Condition \*conditions;  
};  
  
struct request {  
 enum requestType type;  
 union {  
 struct createTableRequest createTableRequest;  
 struct dropTableRequest dropTableRequest;  
 struct insertRequest insertRequest;  
 struct selectRequest selectRequest;  
 struct deleteRequest deleteRequest;  
 };  
};

Все запросы выполняются в отдельном потоке, это позволяет не блокировать основное ожидание сокета и переиспользовать его.



**Разработанный протокол формата xml в виде xsd файла**

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">  
 <xs:element name="sqlQuery">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="requestType">  
 <xs:simpleType>  
 <xs:restriction base="xs:string">  
 <xs:enumeration value="SELECT"/>  
 <xs:enumeration value="INSERT"/>  
 <xs:enumeration value="DELETE"/>  
 <xs:enumeration value="UPDATE"/>  
 <xs:enumeration value="CREATE\_TABLE"/>  
 <xs:enumeration value="DROP\_TABLE"/>  
 </xs:restriction>  
 </xs:simpleType>  
 </xs:element>  
 <xs:element name="tableName" type="xs:string"/>  
 <xs:choice>  
 <xs:sequence>  
 <!-- If requestType is DELETE or SELECT -->  
 <xs:element name="filters" minOccurs="0">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="filter" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="leftOp">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="isColumnName" type="xs:boolean"/>  
 <xs:element name="value" minOccurs="0">  
 <xs:simpleType>  
 <xs:union memberTypes="xs:string xs:int xs:double xs:boolean"/>  
 </xs:simpleType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <xs:element name="operator">  
 <xs:simpleType>  
 <xs:restriction base="xs:string">  
 <xs:enumeration value="=="/>  
 <xs:enumeration value="&lt;"/>  
 <xs:enumeration value="&gt;"/>  
 </xs:restriction>  
 </xs:simpleType>  
 </xs:element>  
 <xs:element name="rightOp">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="isColumnName" type="xs:boolean"/>  
 <xs:element name="value" minOccurs="0">  
 <xs:simpleType>  
 <xs:union memberTypes="xs:string xs:int xs:double xs:boolean"/>  
 </xs:simpleType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 <xs:element name="jo in">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="joinConditions" type="JoinConditionType"/>  
 <xs:element name="filterConditions" type="FilterConditionType" minOccurs="0"/>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <!-- If requestType is INSERT -->  
 <xs:element name="insertValues" minOccurs="0">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="field" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="name" type="xs:string"/>  
 <xs:element name="value" type="xs:string"/>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <!-- If requestType is UPDATE -->  
 <xs:element name="updateValues" minOccurs="0">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="field" maxOccurs="unbounded">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="name" type="xs:string"/>  
 <xs:element name="type">  
 <xs:simpleType>  
 <xs:restriction base="xs:string">  
 <xs:enumeration value="string"/>  
 <xs:enumeration value="int"/>  
 <xs:enumeration value="double"/>  
 <xs:enumeration value="boolean"/>  
 </xs:restriction>  
 </xs:simpleType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <!-- If requestType is CREATE\_TABLE -->  
 <xs:element name="fields" minOccurs="0">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="field" maxOccurs="unbounded" minOccurs="0">  
 <xs:complexType>  
 <xs:sequence>  
 <xs:element name="name" type="xs:string"/>  
 <xs:element name="type" type="xs:string"/>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 </xs:choice>  
 </xs:sequence>  
 </xs:complexType>  
 </xs:element>  
 <xs:complexType name="JoinConditionType">  
 <xs:sequence>  
 <xs:element name="leftTable" type="xs:string"/>  
 <xs:element name="leftColumn" type="xs:string"/>  
 <xs:element name="operator" type="xs:string"/>  
 <xs:element name="rightTable" type="xs:string"/>  
 <xs:element name="rightColumn" type="xs:string"/>  
 </xs:sequence>  
 </xs:complexType>  
 <xs:complexType name="FilterConditionType">  
 <xs:sequence>  
 <xs:element name="table" type="xs:string"/>  
 <xs:element name="column" type="xs:string"/>  
 <xs:element name="operator" type="xs:string"/>  
 <xs:element name="value" type="xs:string"/>  
 </xs:sequence>  
 </xs:complexType>  
</xs:schema>

Он предоставляет в основном только структуру запроса, не ограничивая типы, валидация типов происходит впоследствии в коде.

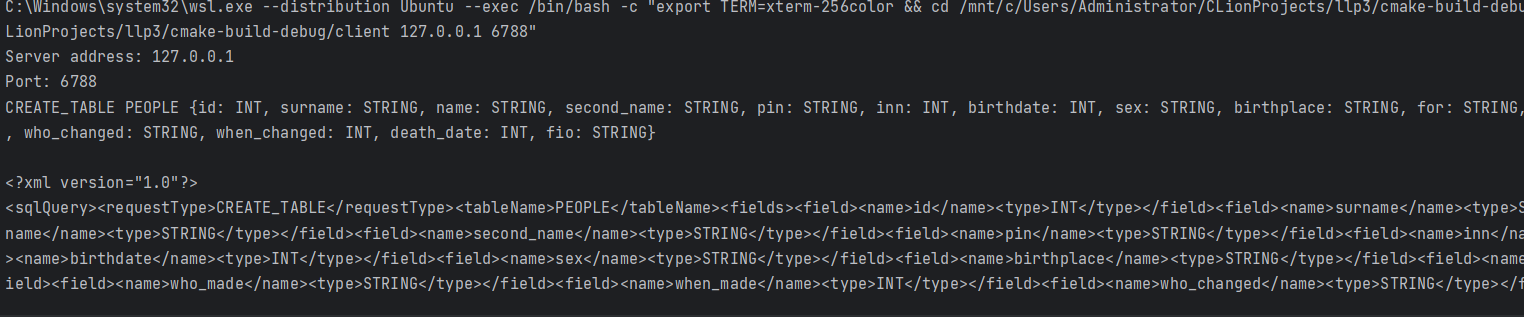
Выводы по работе

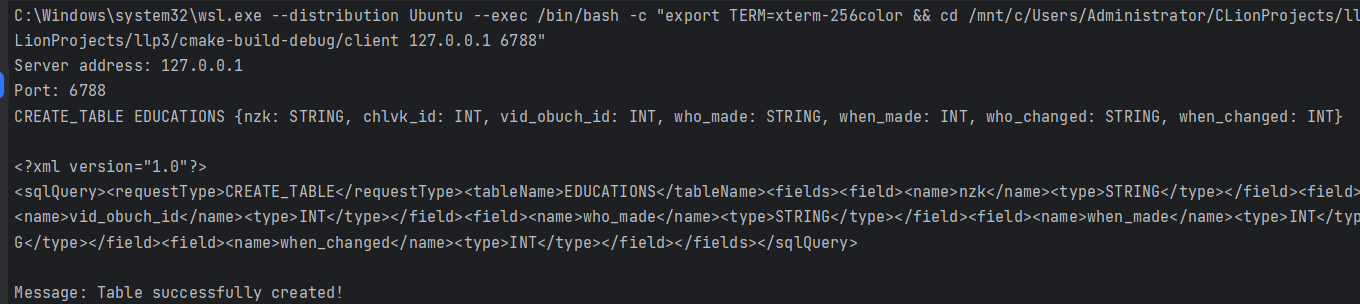
В ходе выполнения лабораторной работы были соединены результаты 1 и 2 лр. Реализовано клиент-серверное взаимодействие с помощью протокола на хml

**Дополнительное задание**

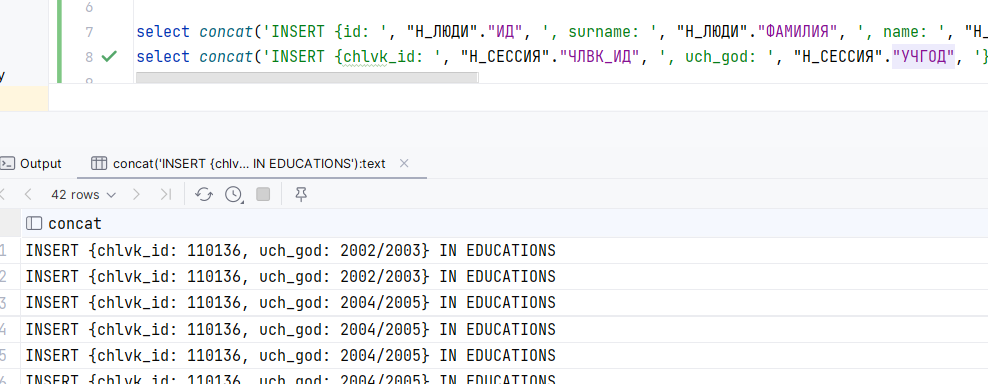
Создадим несколько таблиц с реальными данными и проверим работоспособность соединений на них. Созданная база данных поддерживает только внутренние соединения методом вложенного цикла, поэтому тип соединения дополнительно не указывается.

Для примера использовалась база данных ucheb, хранимая на университетском сервере

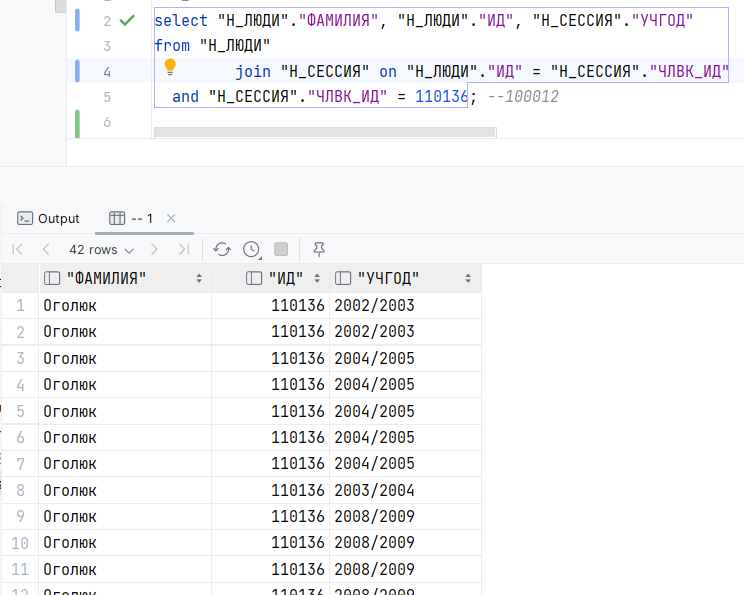




Сгенерированы скрипты для вставки



Ожидаемый результат соединения



Результат на практике

Server address: 127.0.0.1  
Port: 6794  
FOR u IN PEOPLE

FOR j IN EDUCATIONS

FILTER u.id == j.chlvk\_id

FILTER u.id == 110136

RETURN {surname: u.surname, id: u.id, uch\_god: j.uch\_god}  
  
  
<?xml version="1.0"?>  
<sqlQuery><requestType>JOIN</requestType><tableName>JOIN</tableName><join><aliases><alias><table>PEOPLE</table><al>u</al></alias><alias><table>EDUCATIONS</table><al>j</al></alias><alias><table></tabl

e><al></al></alias></aliases><joinConditions><leftAlias>u</leftAlias><rightAlias>j</rightAlias><operator>==</operator><leftColumn>id</leftColumn><rightColumn>chlvk\_id</rightColumn><tableAlias>u</tableAlias><operator>==</operator><column>id</column><value>110136</value></joinConditions><filterConditions/><return/></join></sqlQuery>  
  
surname: Оголюк  
 id: 110136  
uch\_god: 2009/2010  
surname: Оголюк  
 id: 110136  
uch\_god: 2009/2010  
surname: Оголюк  
 id: 110136  
uch\_god: 2009/2010  
surname: Оголюк  
 id: 110136  
uch\_god: 2009/2010  
surname: Оголюк  
 id: 110136  
uch\_god: 2009/2010  
surname: Оголюк  
 id: 110136  
uch\_god: 2009/2010  
surname: Оголюк  
 id: 110136  
uch\_god: 2009/2010  
surname: Оголюк  
 id: 110136  
uch\_god: 2009/2010  
surname: Оголюк  
 id: 110136  
uch\_god: 2010/2011  
surname: Оголюк  
 id: 110136  
uch\_god: 2008/2009  
surname: Оголюк  
 id: 110136  
uch\_god: 2008/2009  
surname: Оголюк  
 id: 110136  
uch\_god: 2006/2007  
surname: Оголюк  
 id: 110136  
uch\_god: 2006/2007  
surname: Оголюк  
 id: 110136  
uch\_god: 2005/2006  
surname: Оголюк  
 id: 110136  
uch\_god: 2006/2007  
surname: Оголюк  
 id: 110136  
uch\_god: 2006/2007  
surname: Оголюк  
 id: 110136  
uch\_god: 2006/2007  
surname: Оголюк  
 id: 110136  
uch\_god: 2005/2006  
surname: Оголюк  
 id: 110136  
uch\_god: 2005/2006  
surname: Оголюк  
 id: 110136  
uch\_god: 2005/2006  
surname: Оголюк  
 id: 110136  
uch\_god: 2007/2008  
surname: Оголюк  
 id: 110136  
uch\_god: 2007/2008  
surname: Оголюк  
 id: 110136  
uch\_god: 2007/2008  
surname: Оголюк  
 id: 110136  
uch\_god: 2010/2011  
surname: Оголюк  
 id: 110136  
uch\_god: 2010/2011  
surname: Оголюк  
 id: 110136  
uch\_god: 2010/2011  
surname: Оголюк  
 id: 110136  
uch\_god: 2010/2011  
surname: Оголюк  
 id: 110136  
uch\_god: 2010/2011  
surname: Оголюк  
 id: 110136  
uch\_god: 2008/2009  
surname: Оголюк  
 id: 110136  
uch\_god: 2008/2009  
surname: Оголюк  
 id: 110136  
uch\_god: 2008/2009  
surname: Оголюк  
 id: 110136  
uch\_god: 2008/2009  
surname: Оголюк  
 id: 110136  
uch\_god: 2008/2009  
surname: Оголюк  
 id: 110136  
uch\_god: 2008/2009  
surname: Оголюк  
 id: 110136  
uch\_god: 2003/2004  
surname: Оголюк  
 id: 110136  
uch\_god: 2004/2005  
surname: Оголюк  
 id: 110136  
uch\_god: 2004/2005  
surname: Оголюк  
 id: 110136  
uch\_god: 2004/2005  
surname: Оголюк  
 id: 110136  
uch\_god: 2004/2005  
surname: Оголюк  
 id: 110136  
uch\_god: 2004/2005  
surname: Оголюк  
 id: 110136  
uch\_god: 2002/2003  
surname: Оголюк  
 id: 110136  
uch\_god: 2002/2003  
Message: End of table  
  
 Process finished with exit code 0