МИНИСТЕРСТВОНАУКИИВЫСШЕГООБРАЗОВАНИЯРФ

# НИУИТМО

**ФАКУЛЬТЕТПРОГРАММНОЙИНЖЕНЕРИИИКОМПЬЮТЕРНОЙ ТЕХНИКИ**

**Лабораторнаяработа№3**

подисциплине

Низкоуровневое программирование

Выполнил:

Студент группы P33102

Манжиков Никита Сергеевич

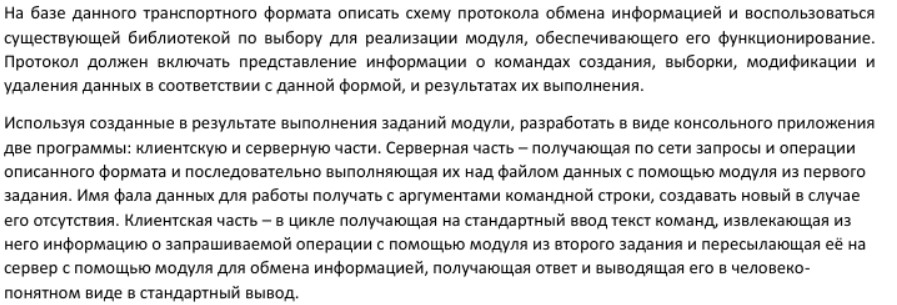
Преподаватель:

Кореньков Юрий Дмитриевич

Санкт-Петербург

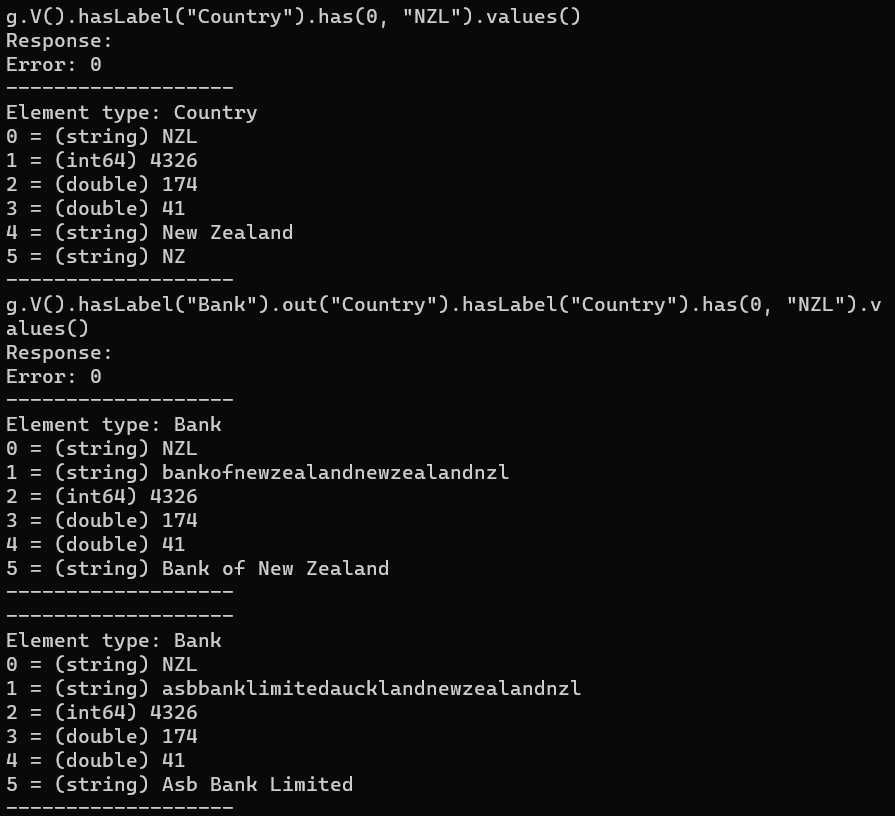
2023

# Цели



# Задачи

Пример сеанса:



Задачи которые были решены:

1. Спроектирован формат запросов и ответов
2. Спроектированы JSON схемы отображающие запросы и ответы
3. Изучены различные варианты JSON сериализаторов-десериализаторов, их достоинства и недостатки
4. Выбрана библиотека C++ nlohmann/json. Язык C++ выбран в силу отсутствия готовых и поддерживаемых библиотек сериализации-десериализации JSON с поддержкой валидации схемы, включающей в себя рекурсивные элементы
5. Создан интерфейс(объявление) класса TcpSocket - обертки над API сокетов
6. Созданы реализации (при помощи препроцессора выбирается какой .cpp файл использовать) для Windows и \*Nix
7. Созданы и реализованы классы Server и Client использующие TcpSocket
8. Модули интегрированы с сериализаторами-десериализаторами и разработанными в предыдущих лабораторных работах программами.

# Описание работы

Проект состоит из следующих модулей:

1. server - содержит сервер, который принимает соединения, читает запросы, валидирует их схемой, и передает на обработку классу StorageAdapter - который обрабатывает соответствующие запросы и возвращает результат в виде Json обратно серверу и сервер его отправляет клиенту.
2. cient - содержит клиент, который парсит запросы на языке гремлин в AST, переводит AST в json, валидирует полученный запрос, отправляет его серверу, получает ответ в виде json, переводит его в тип элемента базы данных и выводит.
3. share - содержит общие для клиента и сервера заголовки структур данных и класс TcpSocket - обертку на Socket API от ос.

Json схема запроса:

{

|  |
| --- |
| "$schema": "http://json-schema.org/draft-07/schema#", |
| "$id": "./query\_schema.json", |
| "title": "Query", |
| "description": "Query json represantation", |
| "definitions": { |
| "element": { |
| "type": "object", |
| "properties": { |
| "elementType": { |
| "type": "string" |
| }, |
| "attributes": { |
| "type": "array", |
| "items": { |

|  |
| --- |
| "oneOf": [ |
| { |
| "type": "string" |
| }, |
| { |
| "type": "number" |
| }, |
| { |
| "type": "boolean" |
| } |
| ] |
| } |
| } |
| }, |
| "required": [ |
| "elementType", |
| "attributes" |
| ], |
| "additionalProperties": false |
| }, |
| "attributesPattern": { |
| "oneOf": [ |
| { |
| "description": "binary operation", |
| "type": "object", |
| "properties": { |
| "patternType": { |
| "const": "BIN\_OP" |
| }, |
| "operation": { |
| "enum": [ |
| "AND", |
| "OR" |
| ] |
| }, |
| "lhs": { |
| "$ref": "#/definitions/attributesPattern" |
| }, |
| "rhs": { |
| "$ref": "#/definitions/attributesPattern" |
| } |
| }, |
| "required": [ |

|  |
| --- |
| "patternType", |
| "operation", |
| "lhs", |
| "rhs" |
| ], |
| "additionalItems": false |
| }, |
| { |
| "description": "condition node", |
| "type": "object", |
| "properties": { |
| "patternType": { |
| "const": "COND" |
| }, |
| "operation": { |
| "enum": [ |
| "EQ", |
| "NEQ", |
| "GT", |
| "LT", |
| "GTE", |
| "LTE" |
| ] |
| }, |
| "index": { |
| "type": "integer" |
| }, |
| "value": { |
| "oneOf": [ |
| { |
| "type": "number" |
| }, |
| { |
| "type": "string" |
| } |
| ] |
| } |
| }, |
| "required": [ |
| "patternType", |
| "operation", |
| "index", |
| "value" |

|  |
| --- |
| ], |
| "additionalItems": false |
| }, |
| { |
| "description": "condition node", |
| "type": "object", |
| "properties": { |
| "patternType": { |
| "const": "COND" |
| }, |
| "operation": { |
| "const": "CONTAINS" |
| }, |
| "index": { |
| "type": "integer" |
| }, |
| "value": { |
| "type": "string" |
| } |
| }, |
| "required": [ |
| "patternType", |
| "operation", |
| "index", |
| "value" |
| ], |
| "additionalItems": false |
| } |
| ] |
| }, |
| "linkPattern": { |
| "type": "object", |
| "properties": { |
| "linkType": { |
| "type": "string" |
| }, |
| "target": { |
| "$ref": "#/definitions/elementPattern" |
| } |
| }, |
| "required": [ |
| "linkType" |
| ], |

|  |
| --- |
| "additionalProperties": false |
| }, |
| "elementPattern": { |
| "type": "object", |
| "properties": { |
| "elementType": { |
| "type": "string" |
| }, |
| "attributes": { |
| "$ref": "#/definitions/attributesPattern" |
| }, |
| "linksIn": { |
| "type": "array", |
| "items": { |
| "$ref": "#/definitions/linkPattern" |
| } |
| }, |
| "linksOut": { |
| "type": "array", |
| "items": { |
| "$ref": "#/definitions/linkPattern" |
| } |
| } |
| }, |
| "required": [ |
| "elementType", |
| "linksIn", |
| "linksOut" |
| ], |
| "additionalProperties": false |
| } |
| }, |
| "type": "object", |
| "oneOf": [ |
| { |
| "properties": { |
| "queryType": { |
| "const": "create\_vertex" |
| }, |
| "element": { |
| "$ref": "#/definitions/element" |
| } |
| }, |

|  |
| --- |
| "required": [ |
| "queryType", |
| "element" |
| ], |
| "additionalProperties": false |
| }, |
| { |
| "properties": { |
| "queryType": { |
| "enum": [ |
| "drop\_vertex", |
| "match" |
| ] |
| }, |
| "pattern": { |
| "$ref": "#/definitions/elementPattern" |
| } |
| }, |
| "required": [ |
| "queryType", |
| "pattern" |
| ], |
| "additionalProperties": false |
| }, |
| { |
| "properties": { |
| "queryType": { |
| "const": "create\_edge" |
| }, |
| "linkType": { |
| "type": "string" |
| }, |
| "sourcePattern": { |
| "$ref": "#/definitions/elementPattern" |
| }, |
| "dstPattern": { |
| "$ref": "#/definitions/elementPattern" |
| } |
| }, |
| "required": [ |
| "queryType", |
| "linkType", |
| "sourcePattern", |

|  |
| --- |
| "dstPattern" |
| ], |
| "additionalProperties": false |
| }, |
| { |
| "properties": { |
| "queryType": { |
| "const": "update\_vertex" |
| }, |
| "pattern": { |
| "$ref": "#/definitions/elementPattern" |
| }, |
| "attributes": { |
| "type": "array", |
| "items": { |
| "type": "object", |
| "properties": { |
| "index": { |
| "type": "integer" |
| }, |
| "value": { |
| "oneOf": [ |
| { |
| "type": "number" |
| }, |
| { |
| "type": "string" |
| }, |
| { |
| "type": "boolean" |
| } |
| ] |
| } |
| }, |
| "required": [ |
| "index", |
| "value" |
| ], |
| "additionalProperties": false |
| } |
| } |
| }, |
| "required": [ |

|  |
| --- |
| "queryType", |
| "pattern", |
| "attributes" |
| ], |
| "additionalProperties": false |
| }, |
| { |
| "properties": { |
| "queryType": { |
| "const": "create\_vertex\_type" |
| }, |
| "attributes": { |
| "type": "array", |
| "items": { |
| "enum": [ |
| "INT32", |
| "INT64", |
| "DOUBLE", |
| "BOOL", |
| "STRING" |
| ] |
| } |
| }, |
| "elementType": { |
| "type": "string" |
| } |
| }, |
| "required": [ |
| "queryType", |
| "attributes", |
| "elementType" |
| ], |
| "additionalProperties": false |
| }, |
| { |
| "properties": { |
| "queryType": { |
| "const": "create\_edge\_type" |
| }, |
| "linkType": { |
| "type": "string" |
| } |
| }, |

|  |
| --- |
| "required": [ |
| "queryType", |
| "linkType" |
| ], |
| "additionalProperties": false |
| }, |
| { |
| "properties": { |
| "queryType": { |
| "const": "drop\_vertex\_type" |
| }, |
| "elementType": { |
| "type": "string" |
| } |
| }, |
| "required": [ |
| "queryType", |
| "elementType" |
| ], |
| "additionalProperties": false |
| }, |
| { |
| "properties": { |
| "queryType": { |
| "const": "drop\_edge\_type" |
| }, |
| "linkType": { |
| "type": "string" |
| } |
| }, |
| "required": [ |
| "queryType", |
| "linkType" |
| ], |
| "additionalProperties": false |
| } |
| ] |
| } |

**Json схема ответа:**

|  |
| --- |
| { |
| "$schema": "http://json-schema.org/draft-07/schema#", |
| "$id": "./response\_schema.json", |
| "title": "Response", |
| "description": "Response json represantation", |
| "type": "object", |
| "properties": { |
| "error": { |
| "enum": [ |
| 0, |
| 1 |
| ] |
| }, |
| "result": { |
| "type": "array", |
| "items": { |
| "type": "object", |
| "properties": { |
| "elementType": { |
| "type": "string" |
| }, |
| "attributes": { |
| "type": "array", |
| "items": { |
| "oneOf": [ |
| { |
| "type": "object", |
| "properties": { |
| "attributeType": { |
| "enum": [ |
| "int64", |
| "int32", |
| "bool" |
| ] |
| }, |
| "value": { |
| "type": "integer" |
| } |
| }, |
| "additionalProperties": false, |
| "required": [ |
| "attributeType", |

|  |
| --- |
| "value" |
| ] |
| }, |
| { |
| "type": "object", |
| "properties": { |
| "attributeType": { |
| "const": "double" |
| }, |
| "value": { |
| "type": "number" |
| } |
| }, |
| "additionalProperties": false, |
| "required": [ |
| "attributeType", |
| "value" |
| ] |
| }, |
| { |
| "type": "object", |
| "properties": { |
| "attributeType": { |
| "const": "string" |
| }, |
| "value": { |
| "type": "string" |
| } |
| }, |
| "additionalProperties": false, |
| "required": [ |
| "attributeType", |
| "value" |
| ] |
| } |
| ] |
| } |
| } |
| }, |
| "required": [ |
| "elementType", |
| "attributes" |
| ], |
| "additionalProperties": false |
| } |
| } |
| }, |
| "required": [ |
| "error" |
| ], |
| "additionalProperties": false |
| } |

# Результаты

Выбранная база данных:

[https://demo.neo4jlabs.com:7473/browser/?dbms=neo4j://fincen@demo.neo4jlabs.c](https://demo.neo4jlabs.com:7473/browser/?dbms=neo4j://fincen@demo.neo4jlabs.com&db=fincen)

[om&db=fincen](https://demo.neo4jlabs.com:7473/browser/?dbms=neo4j://fincen@demo.neo4jlabs.com&db=fincen)

Запрос для вывода всех стран

MATCH p=(c:Country) RETURN p

Запрос для вывода всех банков

MATCH p=(b:Entity) RETURN p

Далее банки отформатированы и убраны те, которые не обладают полем

Country.

Превращаем в запросы к нашей программе:

schema.vertexLabel("Bank").properties().String().String().Int64().Double().Double().St

ring().create()

g.addVertex("Bank", "CHN", "chinaconstructionbankcorporationbeijingchinachn", 4326, 105.0, 35.0, "China Construction Bank Corporation")

…

schema.vertexLabel("Country").properties().String().Int64().Double().Double().String()

.String().create()

g.addVertex("Country", "AFG", 4326, 65.0, 33.0, "Afghanistan", "AF")

…

schema.edgeLabel("Country").create()

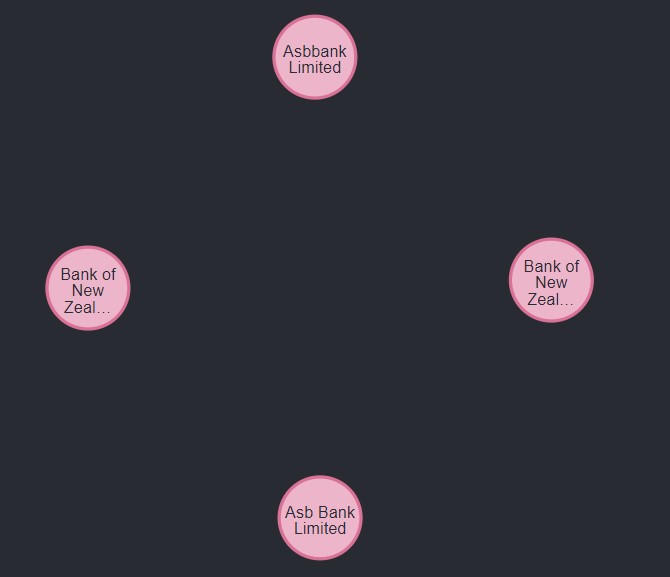
g.addEdge("Country", V().hasLabel("Bank").has(0, "AFG"),

V().hasLabel("Country").has(0, "AFG"))

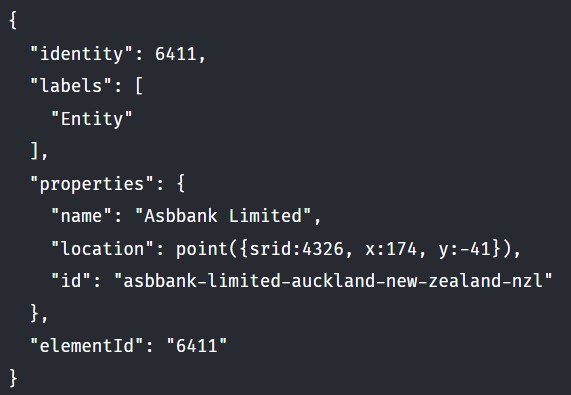
…(генерируем такой запрос для каждой страны)...

Пример запроса к neo4j:

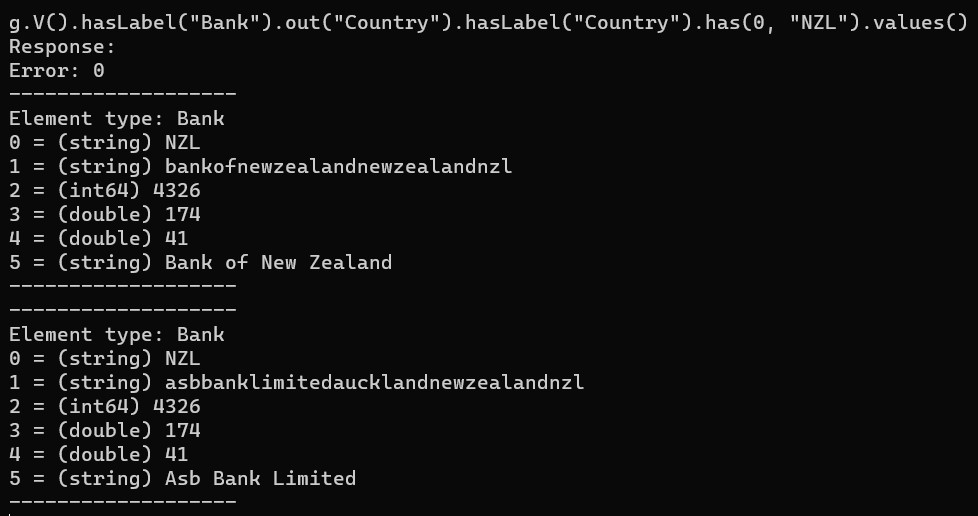
MATCH p=(b)-[r:COUNTRY]->(c) WHERE c.code="NZL" RETURN b выводим все банки Новой Зеландии:



**НО**: у двух банков нет поля Country, поэтому их нет в нашей программе



Вывод аналогичного запроса к программе:



# Выводы

В результате выполнения работы, была разработана JSON схема. Далее при помощи nlohmann/json были сгенерирован серилизаторы, десериализаторы и валидаторы запросов и ответов к бд. Настроено tcp соединения для передачи данных от клиента серверу в формате JSON. Полученные компоненты интегрированы в предыдущие лабораторные работы