Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ « Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ ПО ПРЕДДИПЛОМНОЙ ПРАКТИКЕ

Студент Сукочева Алис
фамилия, имя, отчество

Группа ИУ7-83Б

Тип практики Производственная

Название предприятия НУК ИУ МГТУ им. Н. Э. Баумана

Студент _____ Сукочева А.
подпись, дата *фамилия, и.о.*

Руководитель от предприятия _____ Гаврилова Ю.М.
подпись, дата *фамилия, и.о.*

Руководитель от МГТУ им. Баумана _____ Строганов Ю.В.
подпись, дата *фамилия, и.о.*

Рекомендуемая оценка _____

2022 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ « Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ЗАДАНИЕ
на прохождение производственной практики

на предприятии НУК ИУ МГТУ им. Н. Э. Баумана

Студент Сукочева А. ИУ7-83Б

(фамилия, имя, отчество; инициалы; индекс группы)

Техническое задание необходимо описать используемые инструменты и провести исследование реализуемого метода построения дерева синтаксического анализа языка SQL на основе графических примитивов

Дата выдачи задания « ____ » _____ 20__ г.

Студент	_____	Сукочева А.
	подпись, дата	*фамилия, и.о.*
Руководитель от предприятия	_____	Гаврилова Ю.М.
	подпись, дата	*фамилия, и.о.*
Руководитель от МГТУ им. Баумана	_____	Строганов Ю.В.
	подпись, дата	*фамилия, и.о.*

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время сфера IT набирает все большую и большую популярность. Появляется много платформ, таких как *SAS Data Integration Studio* [1], *Informatica PowerCenter* [2], *Apache NiFi* [3], *DBForge Studio* [4] и многие другие, которые предоставляют графический инструмент для разработки, управления и администрирования баз данных. С ростом популярности языков программирования [5] появляется больше людей, заинтересованных в данной области, что влечет за собой желание больше практиковаться.

SQL (Structured Query Language) – декларативный язык программирования, применяемый для создания, модификации и управления данными в реляционной базе данных, который входит в рейтинг 15 языков программирования, представленный IEEE [5].

Целью данной работы является выбор используемых интсрументов, описание работы ПО, а также исследование метода построения дерева синтаксического анализа на основе графических примитивов.

В рамках реализации проекта должны быть решены следующие задачи:

а) выбран язык программирования и среда разработки;

б) выбраны и описаны средства для работы с грамматикой, а также интсрументов для замеров времени;

в) описаны инструкции для запуска ПО;

г) описана работа с интерфесом и структура прокта;

д) приведены примеры построение дерева синтаксического анализа;

е) описан предмет исследования и произведено исследование количественных характеристик метода.

1 Технологический раздел

1.1 Выбор языка программирования и среды разработки

В качестве языка программирования был выбран C# [6]. Данный язык был выбран по следующим причинам.

а) Он использует объектно-ориентированный подход к программированию.

б) В языке присутствует обилие синтаксических возможностей, которые помогают использовать готовые конструкции, вместо того, чтобы переписывать однотипные строки кода.

в) Данный язык содержит библиотеки и шаблоны, позволяющие не тратить время на изобретение готовых конструкций.

г) Язык является строго типизированным, что позволяет защититься от не проконтролированных ошибок.

В качестве среды разработки была выбрана Visual Studio 2017 [7].

Данная среда была выбрана по следующим причинам.

а) Доступна бесплатная версия для студентов и школьников.

б) Имеется расширение ANTLR Language Support [8], предоставляющее языковую поддержку грамматики ANTLR (V3 и V4) [9].

в) Предоставляет интерактивный интерфейс для работы с диспетчером пактов NuGet [10] и для работы с системой контроля версий.

г) Является кроссплатформенной, что позволяет использовать ее на различных операционных системах.

1.2 Выбор и описание средств для работы с грамматикой

В данном разделе будет обоснован выбор средств для работы с грамматикой.

1.2.1 Выбор инструмента для написания грамматики

В качестве генератора синтаксического анализатора был выбран ANTLR v4 (ANother Tool for Language Recognition) [11]. Данны генератор предоставляет возможность генерировать код, написанный на языке программирования C#, что позволит использовать сгенерированные модуль, подключив его в основную программу.

Разрабатываемая грамматика записывается в файл с расширением *.g4, далее, посредством расширения visual studio для автоматической генерации кода, будут сгенерированы файлы представленные на рис. 1.1. Далее будет использован сгенерированный SqlSelectGrammarParser.cs парсер для построения дерева синтаксического анализа.

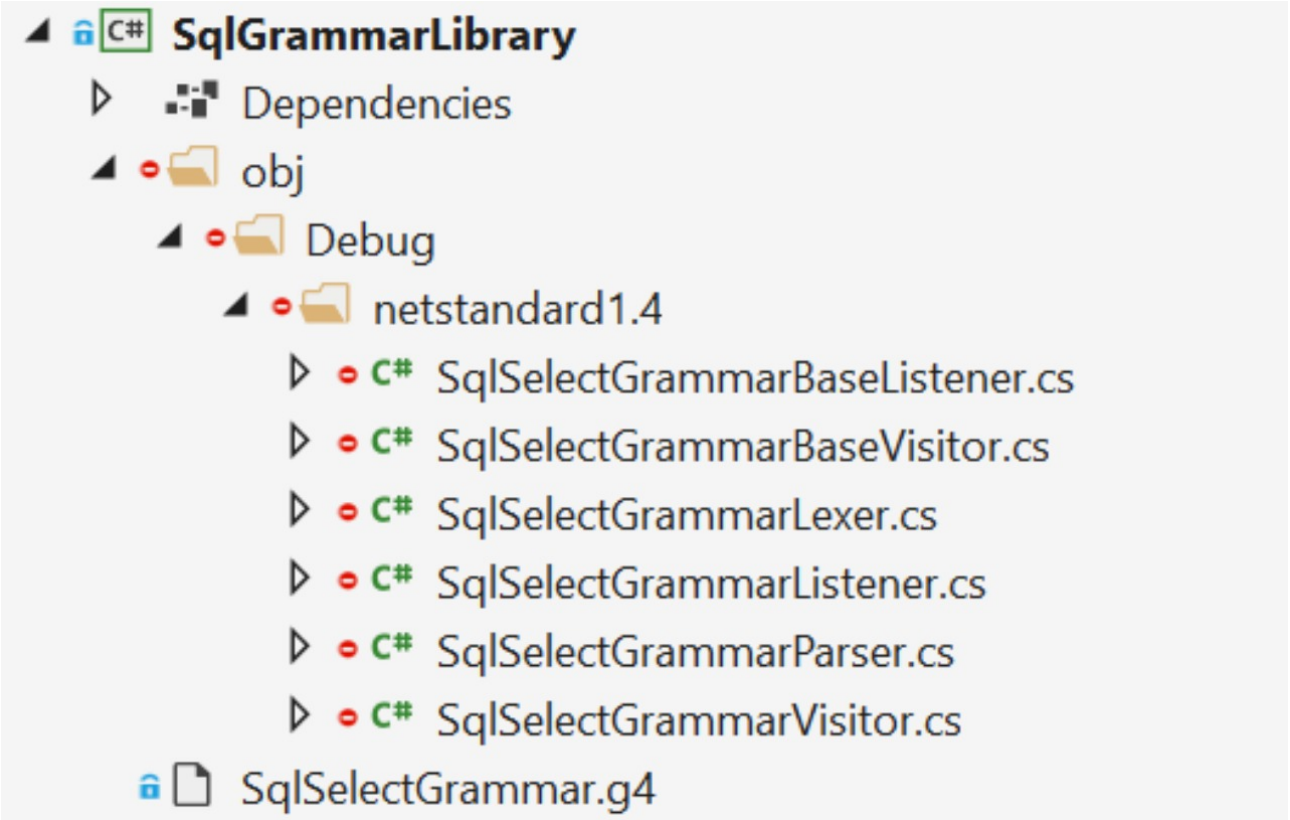

Рисунок 1.1 — Сгенерированные классы с помощью ANTLR v4

1.2.2 Описание работы ANTLR v4

ANTLR v4 - инструмент, используемый для создания новых языков программирования и обработки/перевода структурированных текстовых или

двоичных файлов. ANTLR использует созданную грамматику для генерации парсера, который может создавать и перемещать дерево разбора (или абстрактное синтаксическое дерево, AST). Парсер состоит из выходных файлов на указанном целевом языке. ANTLR v4 поддерживает несколько целевых языков, включая C#. Для работы с IDE в GUI существуют плагин для Visual Studio.

Имена для правил лексера обязаны начинаться с заглавной буквы. После имени идет символ «:», за которым следуют так называемые альтернативы. Альтернативы разделяются сиволом «|» и должны заканчиваться символом «;». Правила для парсера по структуре не отличаются от правил для лексера, а именно имя должно начинаться со строчной буквы. Правила Lexer определяют типы токенов. Их имя должно начинаться с буквы верхнего регистра, чтобы отличать их от правил парсера. Аксиома - правило для парсера, с помощью которого парсер проверяет поток лексем.

После генерации кода в сгенерированном файле SqlSelectGrammarParser.cs будет находиться главный класс SqlSelectGrammarParser, который предоставляет возможность создания дерева синтаксического анализа для реализованной грамматики. Также в этом файле будут реализованы классы аксиомы, созданные из правил парсера.

1.2.3 Выбор инструмента для визуализации дерева запросов

Для визуализации дерева запросов был выбран инструмент Grun.Net [12]. Также данный инструмент предоставляет возможность посредством интерфейса тестировать разработанную грамматику. Для использования данного инструмента необходимо сгенерировать созданную грамматику в виде библиотеки. Далее неоюходимо загрузить сгенерированный dll файл в Grun Net. После чего в окне можно выбрать необходимое правило, которое нужно тестировать и визуализировать. На рис. 1.2 представлен интерфейс программы Grun Net. В правой части экрана изображено аст.

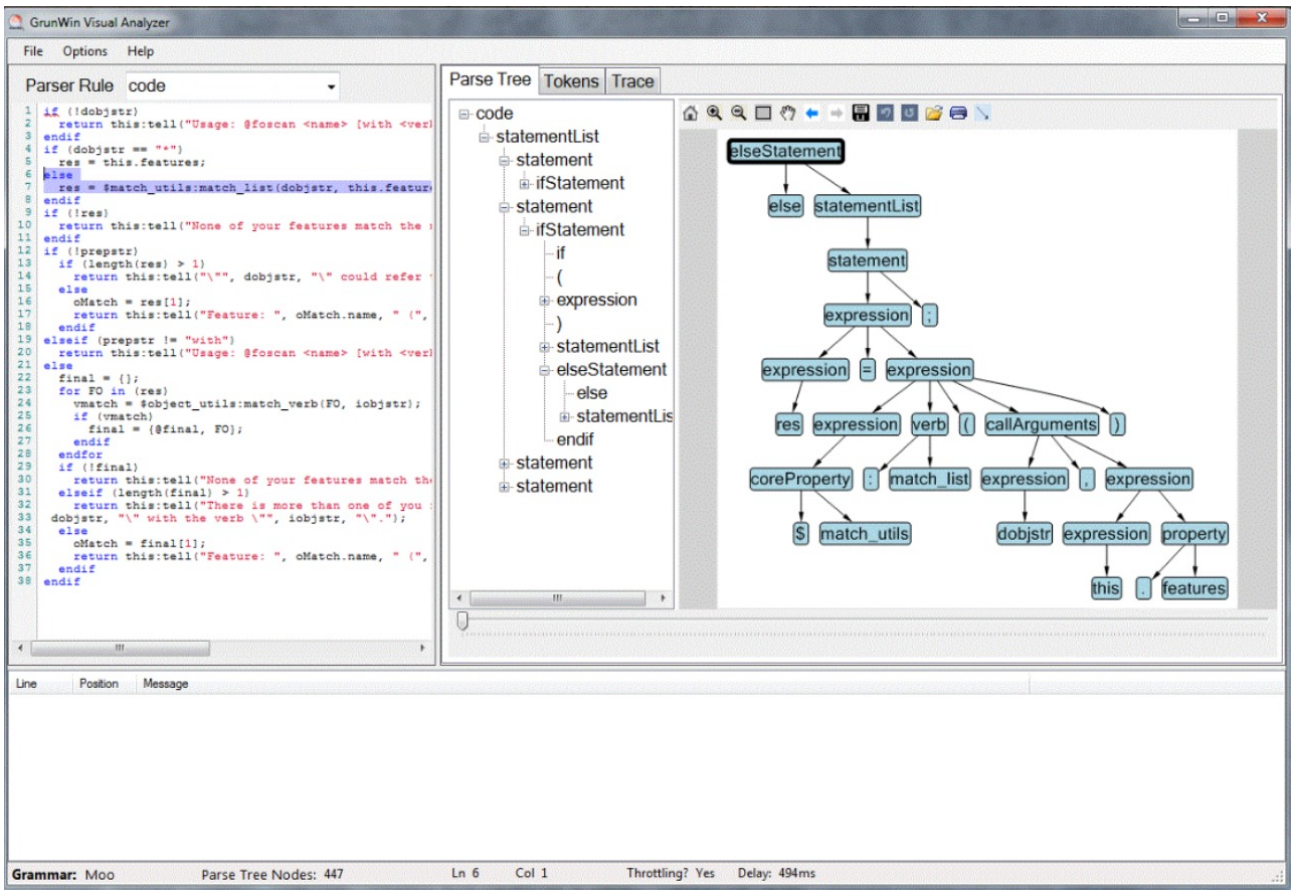

Рисунок 1.2 — Интерфейс программы Grun Net с визуализацией аст

1.3 Выбор средства для проведения замеров времени

BenchmarkDotNet [13] является инструментом, который помогает преобразовывать методы в тесты, отслеживать их эффективность и делиться воспроизводимыми экспериментами по измерению. BenchmarkDotNet гарантирует надежные и точные результаты благодаря статистическому движку perfolizer [14]. Также он защищает от ошибок при тестировании и предупреждает, если что-то не так с тестом или полученными измерениями. Результаты представлены в удобной для пользователя форме, которая выделяет факты об эксперименте. Библиотека используется в более чем 6800 проектах, включая .NET Runtime. BenchmarkDotNet автоматически запускает тесты во всех средах выполнения, объединяет измерения и печатает сводную таблицу с информацией.

1.4 Инструкция для запуска

Для того, чтобы запустить ПО необходимо аппаратное обеспечение с операционной системы Windows. Также необходим установленный .NET Framework.

1.5 Описание работы с интерфейсом

На рисунке 1.3 представлен интерфейс при первом запуске ПО. Экран поделен на две области. В левой области представлен холст, на котором располагаются графические элементы. Изначально холст содержит два основополагающих элемента: запуск и конец. Между этими двумя элементами будет располагаться запрос. В правой области экрана представлена область, для ввода данных в графический элемент. Данная область будет показывать необходимые поля после того, как будет выбран графический элемент, предоставляющий возможность ввода данных. Также в правой части предоставлена возможность построить запрос. Для этого необходимо нажать на кнопку «Process». Результатом будет сообщение о том, удалось ли успешно построить дерево или нет.

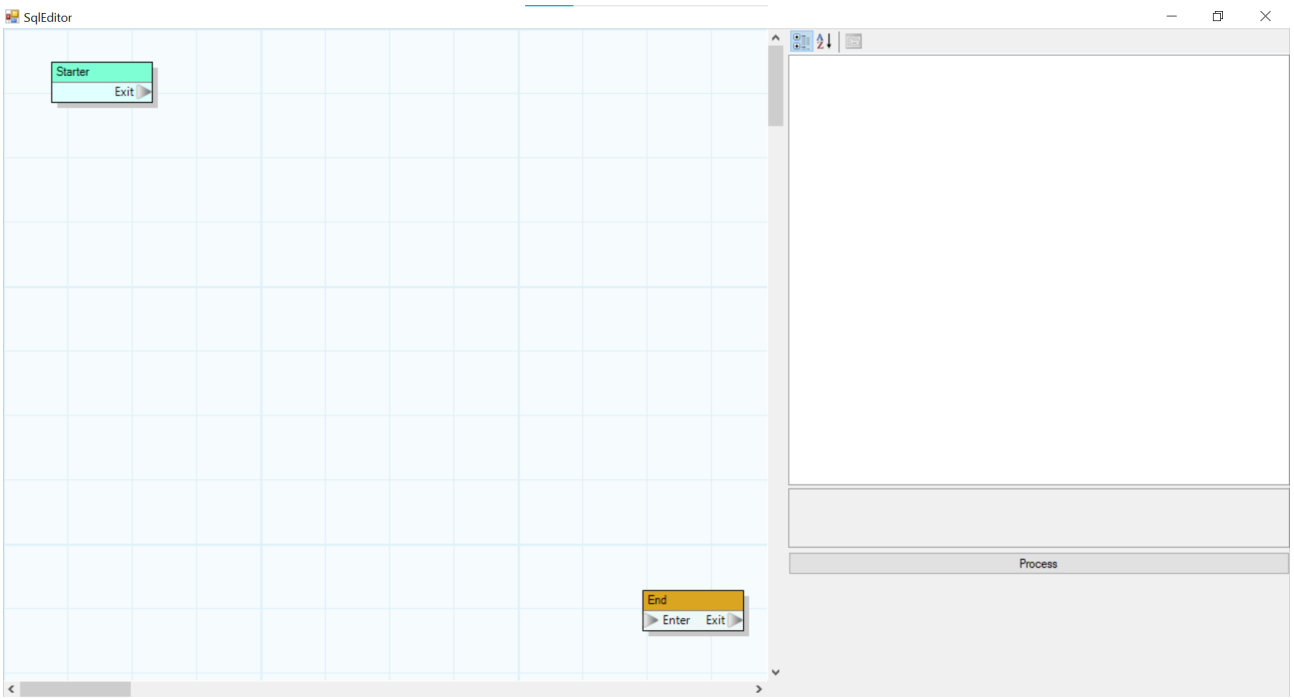

Рисунок 1.3 — Интерфейс при первом запуске ПО

Для того чтобы добавить графический элемент на холст необходимо нажать правой кнопкой мыши по левой области, содержащей холст. После данного действия появится контекстное меню представленное на рисунке 1.4. Данное меню содержит сгруппированные по смыслу элементы. При добавлении графического элемента на холст, он сохраняется в контекстном меню (рис. 1.5).

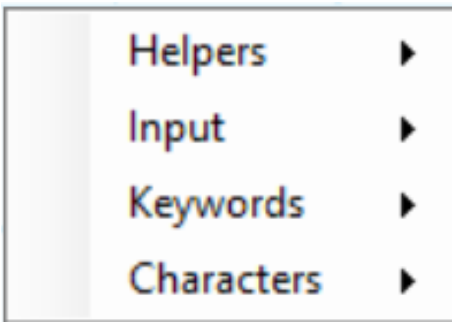

Рисунок 1.4 — Контекстное меню при первом добавлении графического элемента

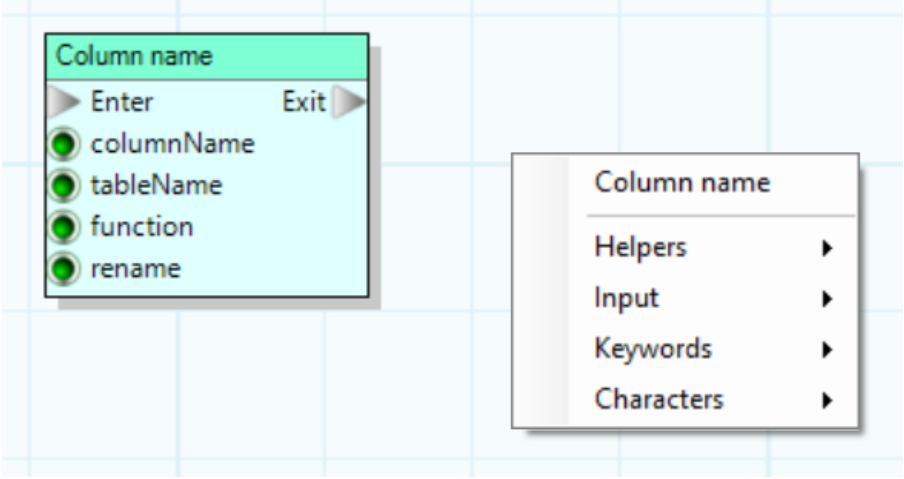

Рисунок 1.5 — Контекстное меню при втором добавлении графического элемента

С графическими элементами возможно производить следующие действия: перемещать по холсту; удалять; дублировать (происходит глубокое копирование, при котором все введенные данные в графический элемент копируются в дубликат); изменение цвета элемента. Для этого необходимо выбрать элемент (активный элемент станет выделен другим цветом) и нажать правой кнопкой мыши. Далее появится следующее контекстное меню представленное на рисунке 1.6.

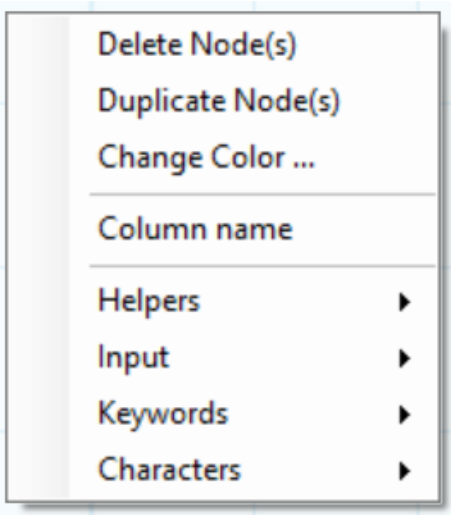

Рисунок 1.6 — Контекстное меню для выбранного графического элемента

Для ввода данных в графический элемент необходимо нажать по нему левой кнопкной мыши. После этого в правой части экрана появится возможность ввода данных. Для элемента «Column name» результат представлен на рисунке 1.7.

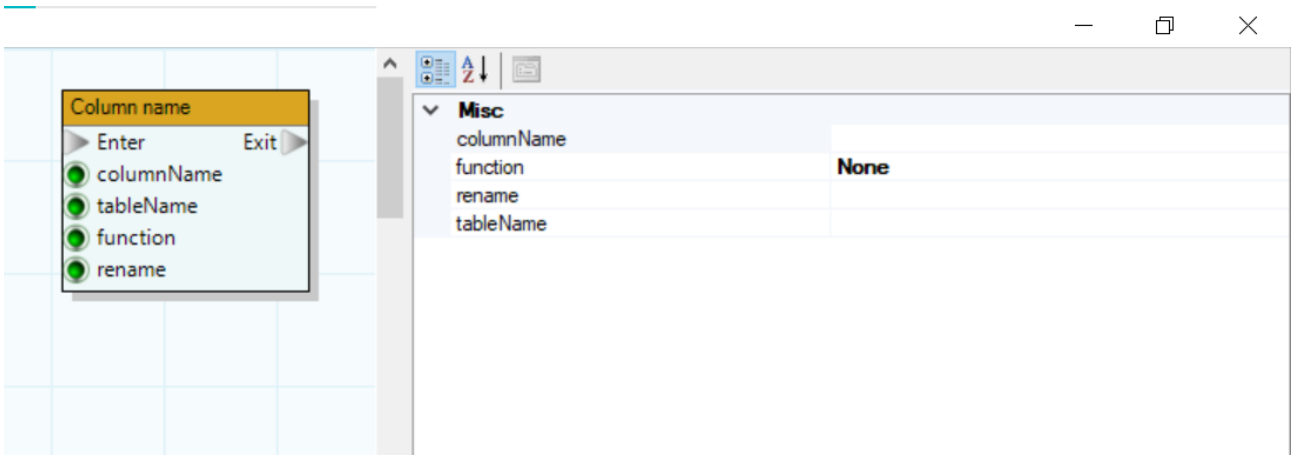

Рисунок 1.7 — Интерфейс для ввода данных в графический элемент «Column name»

После ввода данных необходимо создать цепочку взаимосвязанных между собой графических элементов. Для этого необходимо соединить входы и выходы всех элементов. Процесс связи двух элементов представлен на рисунке 1.8. После того как установлена связь между всеми графическими элементами (от начала до конца) для получения результата необходимо нажать на кнопку «Process». После данного действия появится всплывающее окно.

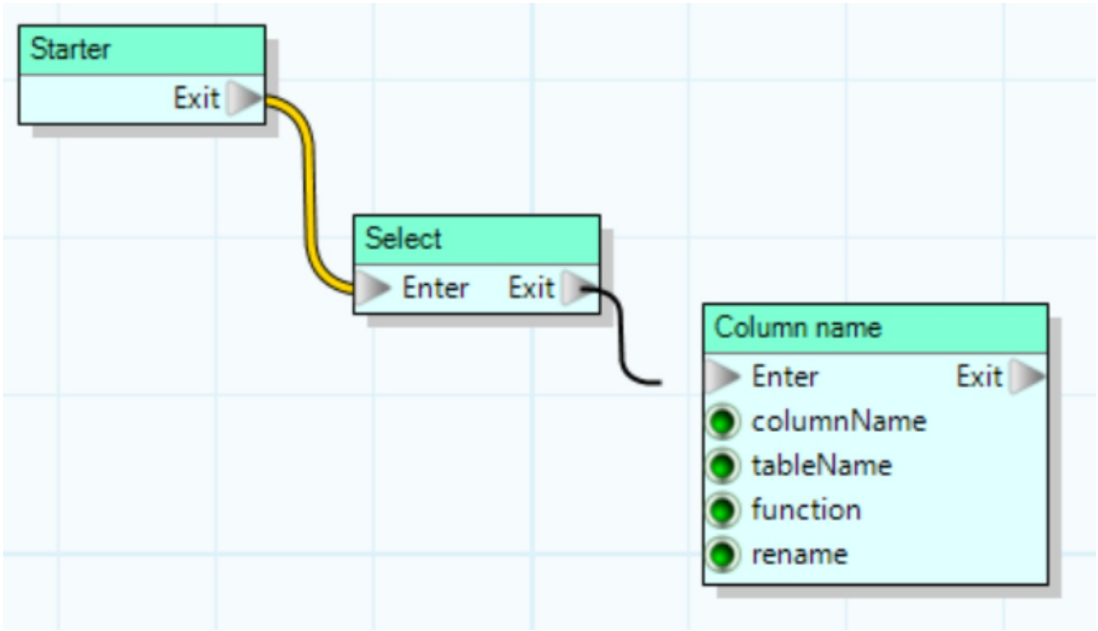

Рисунок 1.8 — Процесс связи двух графических элементов

1.6 Структура проекта

Проект состоит из двух сборок:

а) SqlFront - основная сборка, содержащая весь код разработанного ПО, данная сборка также включает разработанную грамматику;

б) Antlr4.Runtime - сборка проекта antlr с добавлением функционала, который предоставляет возможность использовать парсер без лексера.

На рисунке 1.9 представлена зависимость сборок.

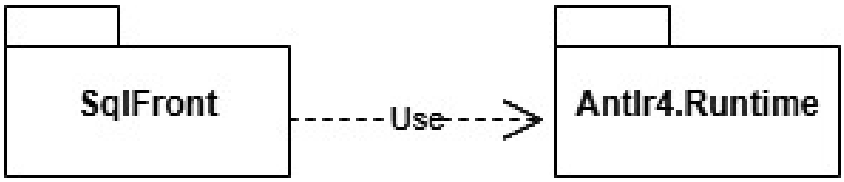

Рисунок 1.9 — Зависимость сборок

Сборка SqlFront содержит следующие пространства имен.

а) Attributes - пространство, содержащее атрибуты для типов токенов.

б) Enums - пространство, содержащее перечисления.

в) Helpers - пространство, содержащее вспомогательные классы.

г) Helpers.Listeners - пространство, содержащее классы, реализующие слушателей.

д) Models - пространство, содержащее модели.

е) StrContext - пространство, содержащее контекст для формирования строкового запроса.

ж) TokensContext - пространство, содержащее контекст для формирования потока токенов.

На рисунке 1.10 предоставлена взаимосвязь пространств имен.

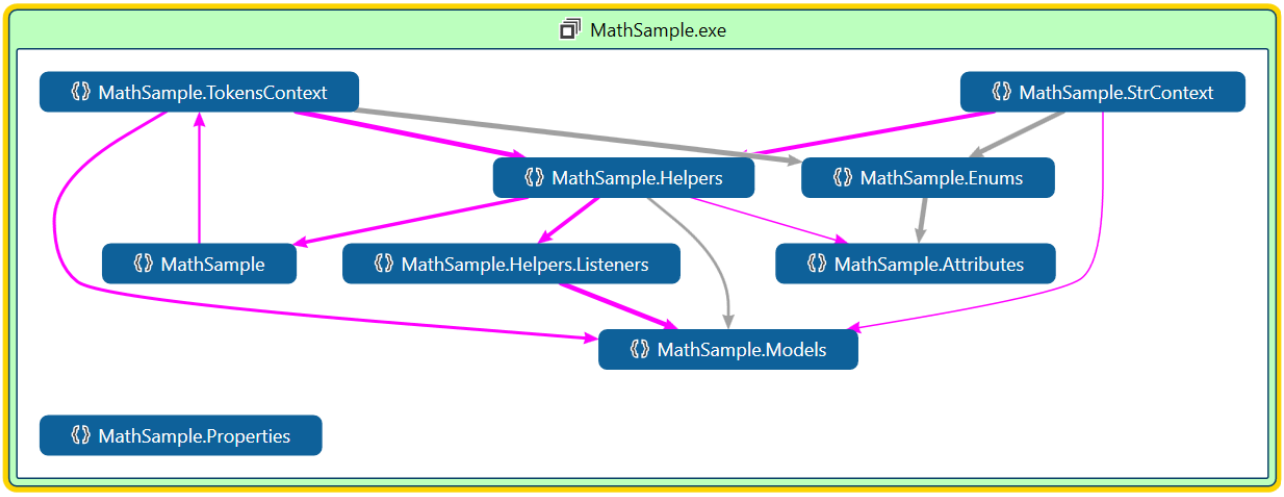

Рисунок 1.10 — Взаимосвязь пространств имен

1.7 Примеры построения дерева

В аналитической части был разобран пример построения дерева синтаксического анализа на основе PostgreSQL (листинг **??**). Также построенное дерево было графически визуализировано. Проведем такую же операцию для разработанного ПО. На рисунке 1.11 представлены взаимосвязанные графические элементы для запроса Capitan Nemo. На рисунке 1.11 визуализировано построенное дерево. По аналогии с PostgreSQL листами данного дерева являются токены, которые содержат в себе основную информацию. Выполняя обход по-данному дерево возможно построение более оптимального дерева, а затем построение плана запроса.

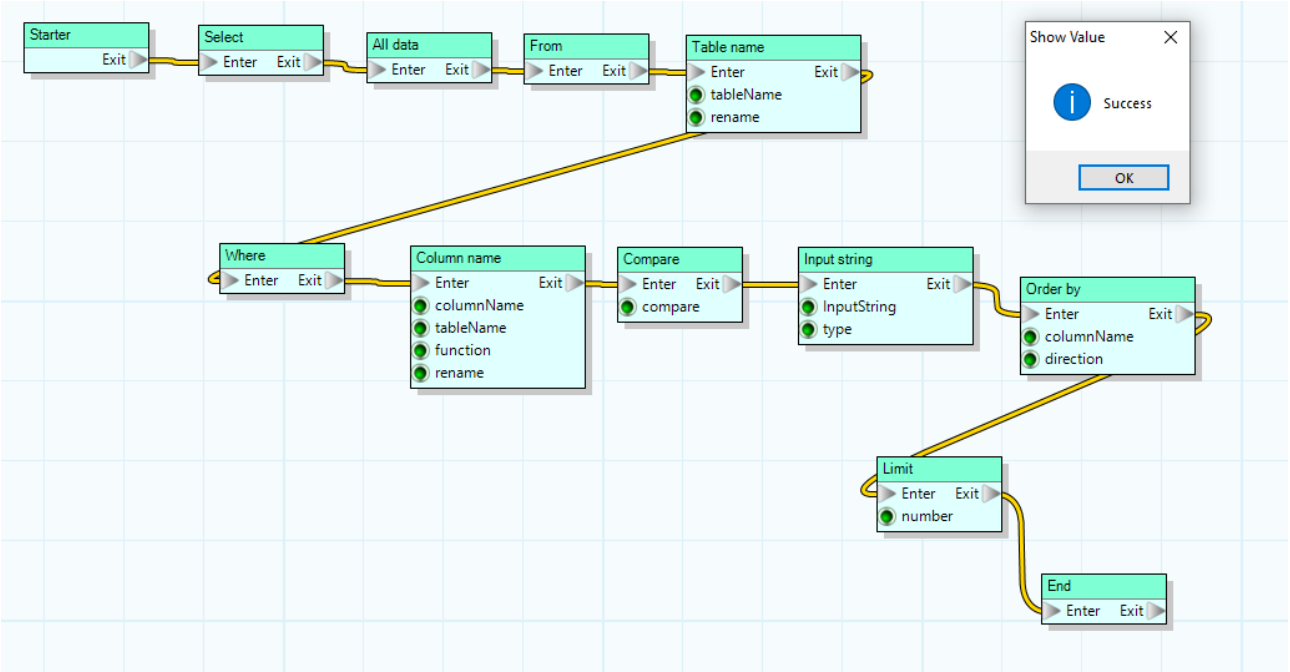

Рисунок 1.11 — Взаимосвязанные графические элементы для запроса Capitan Nemo

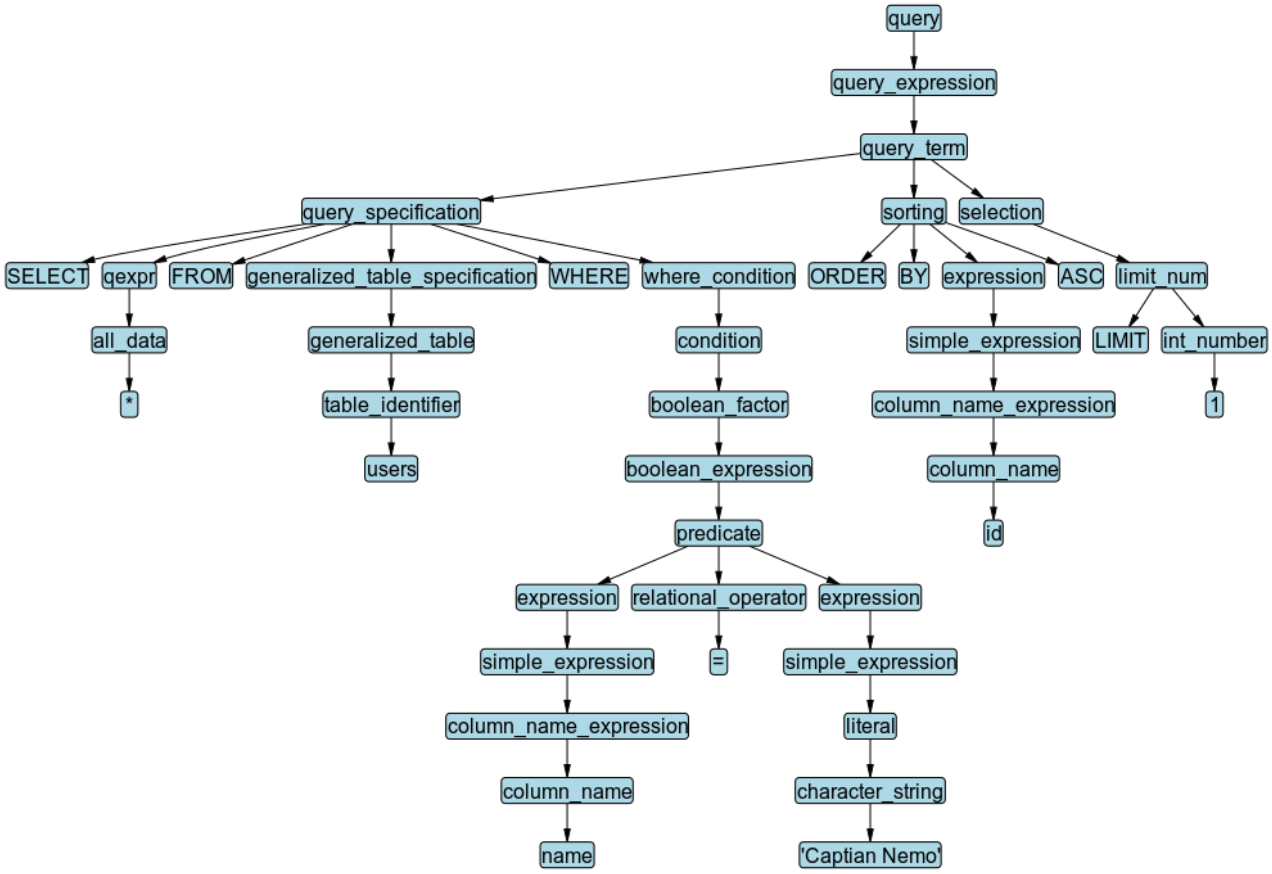

Рисунок 1.12 — Дерево синтаксического анализа для запроса Capitan Nemo

Также поддерживается уровень дерева для порядка операций. Разберем пример запроса, который содержит where-условие представленное на листинге 1.1. Результат построение части дерева синтаксического анализа, связанного с условием where приведен на рисунке 1.13. Из результата видно, что узлы, имеющие более высокий приоритет, расположены ниже, т.к. они должны выполнятся раннее.

Листинг 1.1 — Пример where-условия

```
1 where age > 18 and age < 22 or state = 'Junior'
```

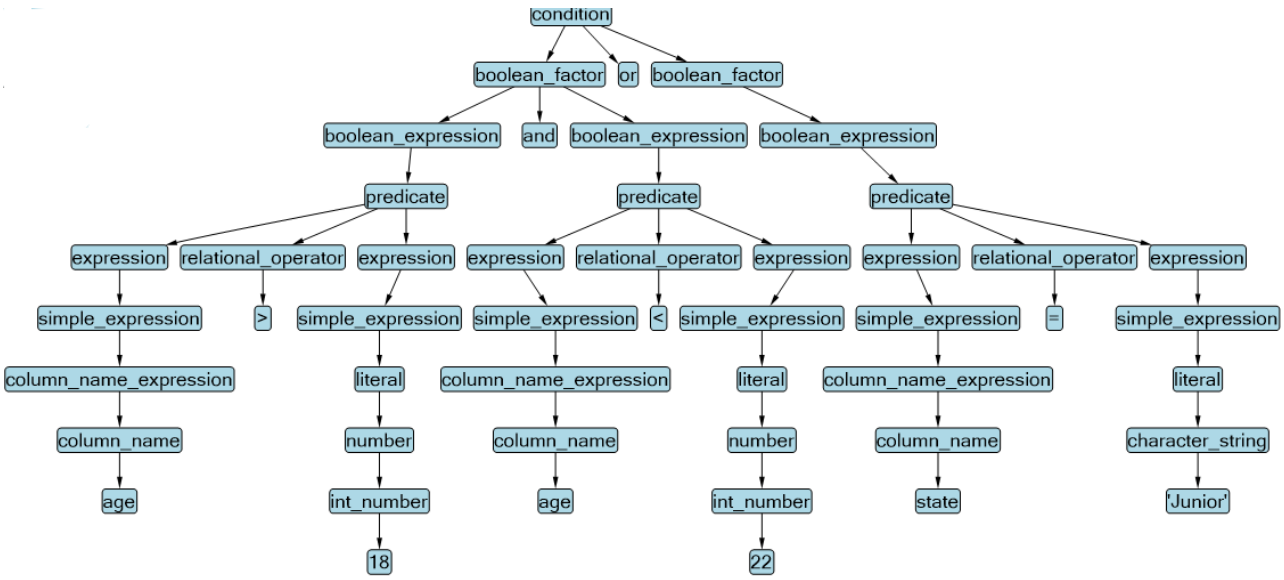

Рисунок 1.13 — Результат построение части дерева синтаксического анализа, связанного с условием where

Также при построении дерева можно увидеть поддержку для следующих приведенных типов (рисунок 1.14).

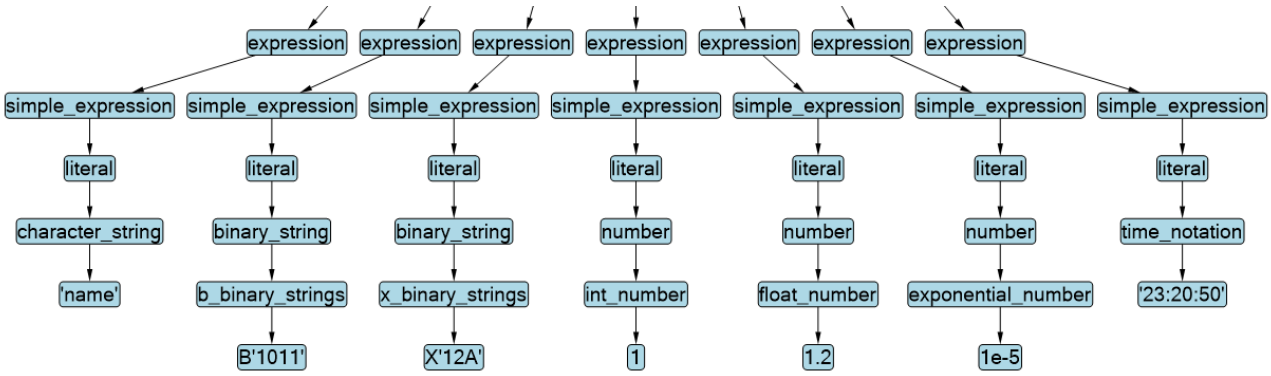

Рисунок 1.14 — Часть дерева синтаксического анализа, содержащая разные типы

Далее будет рассмотрен пример построения дерева с использованием ключевого слова join. На рисунке 1.15 представлена цепочка графических элементов с использованием join. На рисунке 1.16 представлено дерево синтаксического анализа для цепочки графических элементов с использованием join.

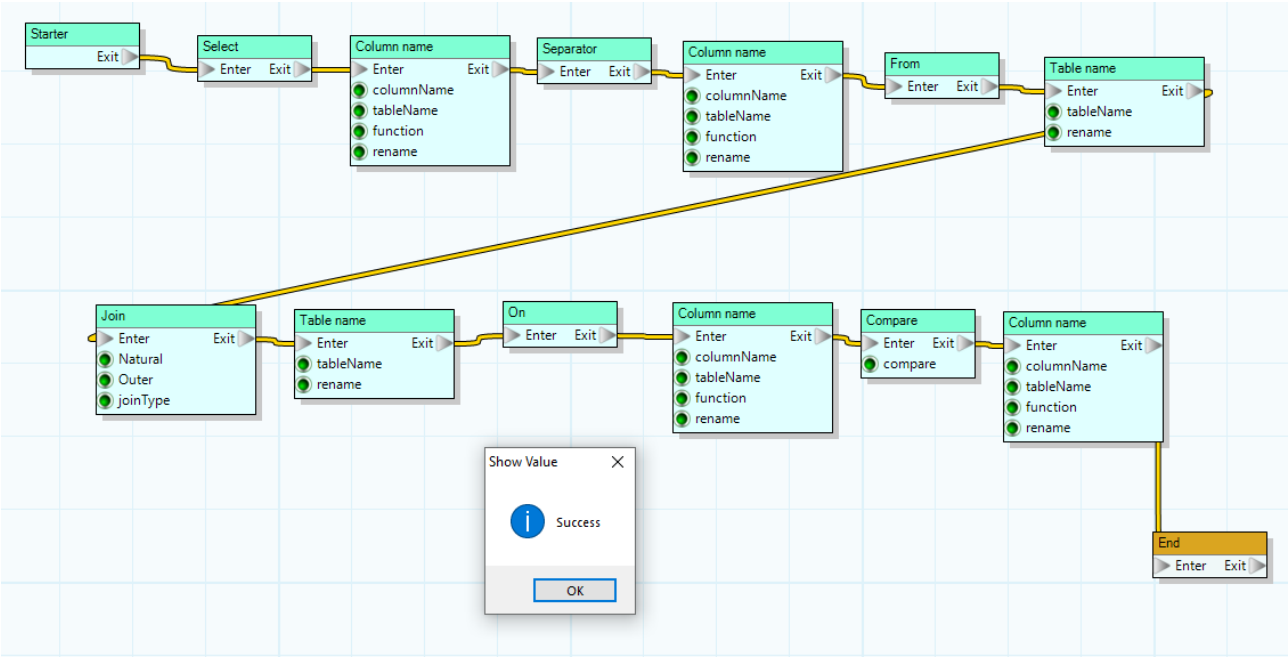

Рисунок 1.15 — Цепочка графических элементов с использованием join

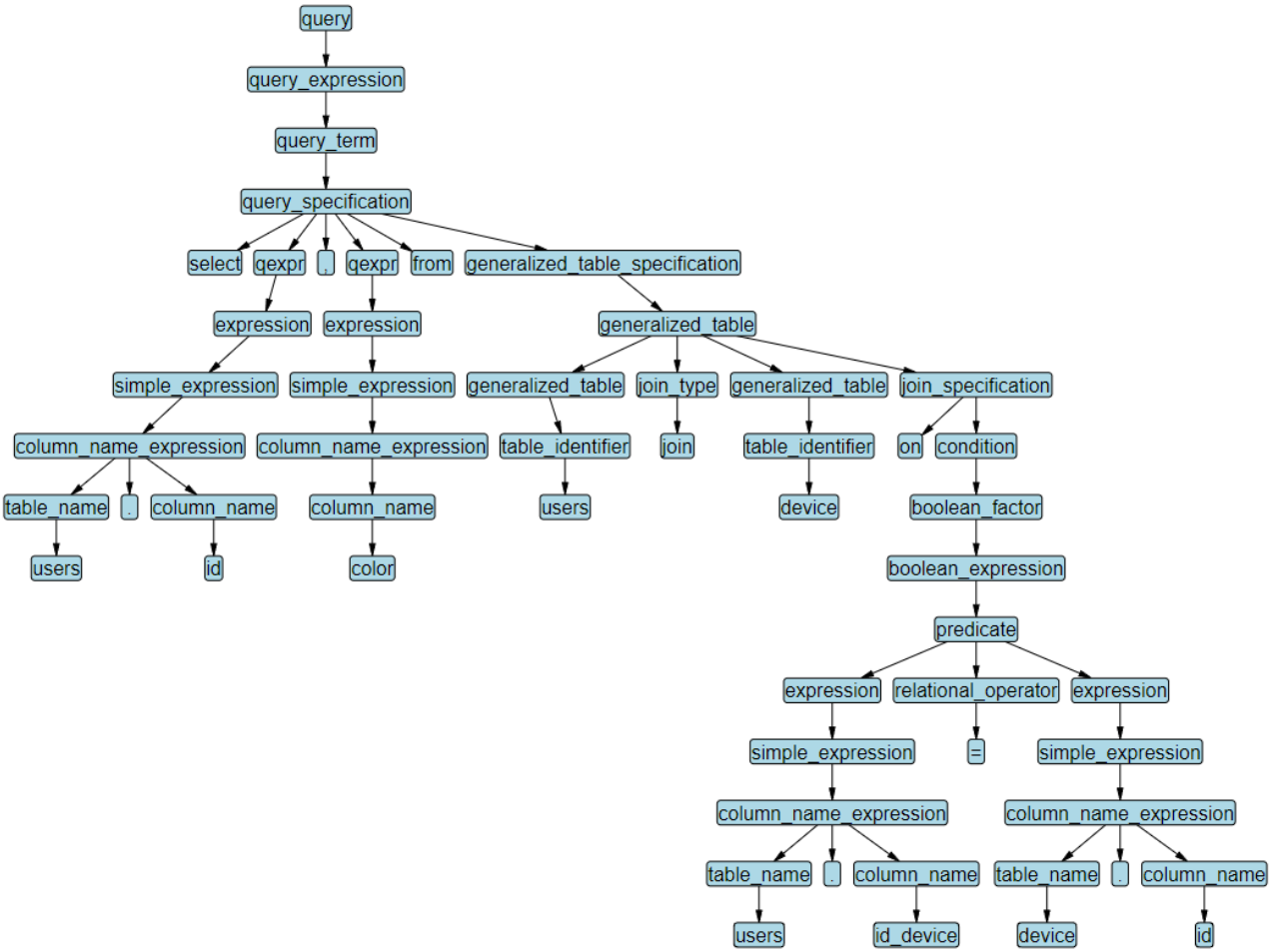

Рисунок 1.16 — Дерево синтаксического анализа для цепочки графических элементов с использованием join

1.8 Выводы из технологического раздела

В данном разделе был выбран и обоснован выбор языка программирования и средств для работы с грамматикой. Также была описана работа ANTLR v4 и выбран инструмент для визуализации дерева запросов. Были описаны инструкции для запуска и описание работы с графическим интерфейсом. Также описана структура проекта и приведены примеры построения дерева синтаксического анализа.

2 Исследовательский раздел

2.1 Предмет исследования

Для проведения исследования будут построены деревья синтаксического анализа для четырех уровней запросов. Построение дерева будет выполняться 10000 раз. Основная задача исследования состоит в сравнение времени построения дерева синтаксического анализа с использованием строки или потока токенов. Запросы будут варьироваться и иметь четыре следующие сложности.

а) VeryEasy - запрос, содержащий только выборку одного элемента из одной таблицы.

б) Easy - запрос, содержащий where-условия и выборку нескольких элементов.

в) Normal - запрос, содержащий join-условие.

г) Hard - запрос, содержащий подзапрос, order by и having-условия.

Запрос уровня VeryEasy представлен на листинге 2.1. Запрос уровня Easy представлен на листинге 2.2. Запросы уровней Normal и Hard представлены на листингах 2.3 и 2.4 соответственно.

Листинг 2.1 — Запрос уровня VeryEasy для benchmarks

```
1 select name from cats
```

Листинг 2.2 — Запрос уровня Easy для benchmarks

```
1 select name, breed, age
2 from cats
3 where breed in ('Sphinx', 'Persian', 'MaineCoon')
4 and age >= 1
5 and age < 5
```

Листинг 2.3 — Запрос уровня Normal для benchmarks

```
"select owners.name, cats.name, breed, cats.age " +
from cats
inner join owners
on owners.cat_name = cats.name
where breed in ('sphinx', 'Persian', 'MaineCoon')
and cats.age >= 1
and cats.age < 5
and cats.eyes = 'green'
and cats.name like '%Barsik%'
and owners.name in ('Alice', 'Katya')
```

Листинг 2.4 — Запрос уровня Hard для benchmarks

```
select owners.name as own, count(owners.name) as con, cats.name as cn, breed as
    b, cats.age as ca, dogs.eyes as de, cats.eyes as ce
from
(select *
from cats)
inner join owners
on owners.cat_name = cats.name
where breed in ('sphinx', 'Persian', 'Maine')
and cats.age >= 1
and cats.age < 5
and cats.eyes = 'green'
and cats.name like '%Barsik%'
and owners.name in ('Alice', 'Katya')
having owners.age >= 5
and owners.age < 18
order by count(owners.name) desc;
```

Характеристики ПО, на котором проводилось исследование представлены ниже.

а) Processor - AMD Ryzen 7 4700U with Radeon Graphics 2.00 GHz.

б) RAM - 16.0 GB.

в) System type - 64-bit operating system, x64-based processor

г) Operating system - Windows 10 Enterprise.

д) Подключен к сети - да.

е) Подключен к питанию - да.

2.2 Результаты исследования

На рисунке 2.1 представлены результаты построения дерева синтаксического анализа для разных типов сложностей запроса. Суффикс Tokens обозначает, что запрос строился на основе токенов, Суффикс Str обозначает, что запрос строился на основе строки. Расшифровка столбцов приведена на рисунке 2.2.

Method	Mean	Error	StdDev	Gen 0	Allocated
VeryEasyQueryTokens	41.33 ms	0.285 ms	0.252 ms	21923.0769	44 MB
VeryEasyQueryStr	62.69 ms	0.394 ms	0.349 ms	26625.0000	53 MB
EasyQueryTokens	208.78 ms	1.135 ms	1.061 ms	85000.0000	170 MB
EasyQueryStr	320.49 ms	2.732 ms	2.556 ms	97000.0000	194 MB
NormalQueryTokens	534.41 ms	4.933 ms	4.615 ms	260000.0000	519 MB
NormalQueryStr	1,671.47 ms	7.143 ms	6.681 ms	652000.0000	1,302 MB
HardQueryTokens	642.57 ms	3.985 ms	3.533 ms	297000.0000	592 MB
HardQueryStr	2,099.53 ms	11.511 ms	10.768 ms	754000.0000	1,504 MB

Рисунок 2.1 — Результаты потсроения дерева синтаксического анализа

```
Mean      : Arithmetic mean of all measurements
Error     : Half of 99.9% confidence interval
StdDev    : Standard deviation of all measurements
Gen 0     : GC Generation 0 collects per 1000 operations
Allocated : Allocated memory per single operation (managed only, inclusive, 1KB = 1024B)
1 ms      : 1 Millisecond (0.001 sec)
```

Рисунок 2.2 — Расшифровка столбцов таблицы Benchmarks

На рисунке 2.3 представлены полученные ранее замеры времени в графическом виде, на рисунке 2.4 представлена расшифровка. Видно, что с ростом уровня сложности запроса растет время построение дерева синтаксического анализа, однако скорость построение дерева синтаксического анализа на основе токенов меньше, нежели чем на основе строки. Это аргументируется тем, что при использовании токенов пропускается первый этап - разбиение строки на поток токенов.

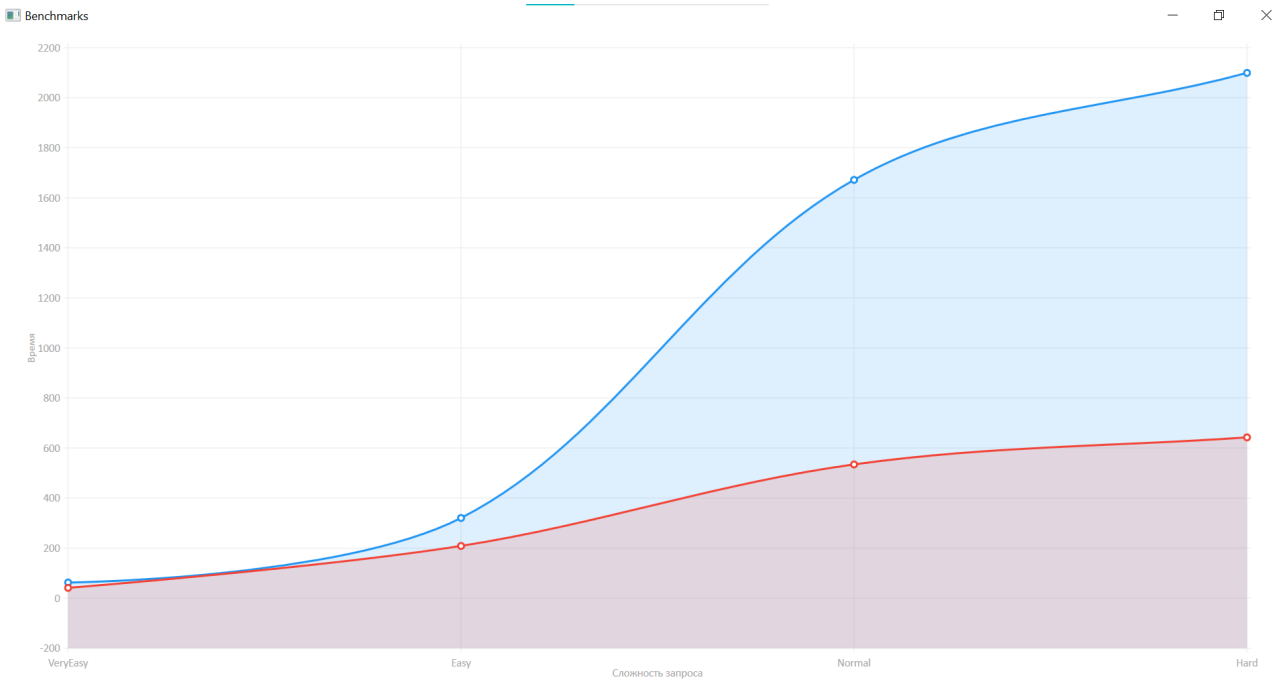

Рисунок 2.3 — Расшифровка столбцов таблицы Benchmarks

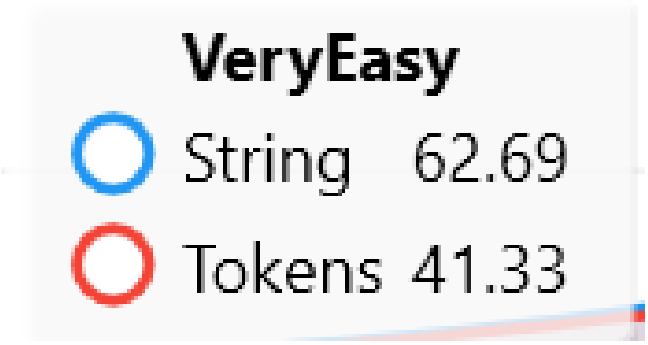

Рисунок 2.4 — Расшифровка графика

2.3 Выводы из исследовательского раздела

В данном разделе была описана предметная область, проведено и проанализировано исследование. Также показано, что время построения дерева синтаксического анализа на основе токенов меньше, чем на основе строки.

ЗАКЛЮЧЕНИЕ

В ходе работы был произведен выбор используемых интсрументов, была описана работа ПО, а также проведено исследование метода построения дерева синтаксического анализа на основе графических примитивов.

В результате были выполнены все поставленные задачи:

а) выбран язык программирования и среда разработки;

б) выбраны и описаны средства для работы с грамматикой, а также интсрументов для замеров времени;

в) описаны инструкции для запуска ПО;

г) описана работа с интерфесом и структура прокта;

д) приведены примеры построение дерева синтаксического анализа;

е) описан предмет исследования;

ж) произведено исследование количественных характеристик метода.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. SAS Data Integration Studio [Электронный ресурс]. — SAS Institute, 1976. — Режим доступа: https://support.sas.com/en/software/data-integration-studio-support.html (Дата обращения 2022-03-05).

2. Informatica PowerCenter [Электронный ресурс]. — Data Integration, 1993. — Режим доступа: https://dis-group.ru/technologies/data-integration/powercenter/ (Дата обращения 2022-03-06).

3. Apache NiFi [Электронный ресурс]. — Apache Software Foundation, 2006. — Режим доступа: https://nifi.apache.org/ (Дата обращения 2022-03-06).

4. DBForge Studio [Электронный ресурс]. — Devart. — Режим доступа: https://www.devart.com/dbforge/ (Дата обращения 2022-03-16).

5. Рейтинг языков программирования [Электронный ресурс]. — Режим доступа: https://spectrum.ieee.org/static/interactive-the-topprogramming-languages-2020 (Дата обращения 2022-05-01).

6. Документация по csharp [Электронный ресурс]. — Microsoft. — Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/ (Дата обращения 2022-04-20).

7. Visual Studio [Электронный ресурс]. — Microsoft. — Режим доступа: https://visualstudio.microsoft.com/ru/vs/older-downloads/ (Дата обращения 2022-04-24).

8. ANTLR Language Support [Электронный ресурс]. — Режим доступа: https://marketplace.visualstudio.com/items?itemName=SamHarwell.ANTLRLanguageSupport (Дата обращения 2022-04-26).

9. ANTLR [Электронный ресурс]. — Режим доступа: https://www.antlr.org/ (Дата обращения 2022-04-23).

10. NuGet [Электронный ресурс]. — Режим доступа: https://www.nuget.org/ (Дата обращения 2022-04-21).

11. Learn tutorials ANTLR [Электронный ресурс]. — Режим доступа: https://learntutorials.net/ru/antlr/topic/4453/ (Дата обращения 2022-04-18).

12. Grun Net [Электронный ресурс]. — Режим доступа: https://github.com/wiredwiz/Grun.Net (Дата обращения 2022-04-18).

13. Benchmarkdotnet [Электронный ресурс]. — Режим доступа: https://benchmarkdotnet.org/articles/overview.html (Дата обращения 2022-05-01).

14. Perfolizer [Электронный ресурс]. — Режим доступа: https://github.com/AndreyAkinshin/perfolizer (Дата обращения 2022-05-01).